

UNIVERSITÀ DEGLI STUDI DI BERGAMO

Facoltà di Ingegneria



Dottorato di Ricerca in

MECCATRONICA, INFORMAZIONE, TECNOLOGIE INNOVATIVE  
E METODI MATEMATICI

- XXVI Ciclo -

**Un approccio mecatronico per la  
progettazione e lo sviluppo di sistemi di  
controllo in asservimento visivo**

Tutor

PROF. PAOLO RIGHETTINI

Dottorando

MATTIA ROSSETTI

Matricola n. 49673

Anni : 2011 - 2014



*Ai miei genitori.*

---

*Un'immagine vale più  
di mille parole.  
(Anonimo)*

*La regola secondo me è: quando sei a un bivio  
e trovi una strada che va in su e una che va in giù,  
piglia quella che va in su.  
È più facile andare in discesa, ma alla fine ti trovi in un buco.  
A salire c'è più speranza.  
È difficile, è un altro modo di vedere le cose,  
è una sfida, ti tiene all'erta.  
(Tiziano Terzani, da "La fine è il mio inizio")*





# Abstract

**Keywords:** Visual Servoing, Image Sensors, Computer Vision, Robot Control, Digital Control, Feedback Control, Feedforward Control, Control System Design, Control System Simulation, Software Design.

Modern robots and mechatronic systems are equipped with a large number of sensors which give them the possibility to accomplish complex tasks. In recent years the integration of robots and vision system has become critical to improve machinery productivity and versatility. Nevertheless, the most widely used approach is still the *look then move* fashion, in which the vision system simply supports a manipulator without taking part in the robot control task. An alternative technique that allows to merge the vision task and the control one is named *visual servoing* or *vision in the loop*. Following this approach, a visual-feedback loop can be added to the standard manipulator control loop in order to drive the robot directly with the information coming from the vision system.

This further controller brings a lot of benefits including: (i) a general increase of the accuracy of the subsystem and its robustness towards environment noise and changes; (ii) the capability to follow fast moving objects; (iii) the possibility to adapt the system to different situations and tasks. On the contrary hand, designing and developing an efficient and robust visual control is typically a difficult task as it involves various issues like computer vision algorithms, control theory, robot kinematics and so on.

The analysis carried out in this thesis aims to highlight the aspects common to all visual servoing tasks in order to provide a general solution schema for this kind of problems. Firstly, the image acquisition process and image elaboration techniques are presented, with particular attention to their application to *tracking* fast moving objects or scenes; secondly, visual servoing controller design is introduced and its kinematic and dynamic issues are

studied; eventually, a software architecture for the implementation of visually guided task is shown along with simple experimental applications used to evaluate the results of the work.

# Ringraziamenti

Arrivato al termine del *Dottorato di Ricerca*, desidero ringraziare le persone che mi hanno guidato durante il cammino che ha portato alla stesura di questa tesi.

Vorrei, innanzitutto, ringraziare il *Prof. Paolo Righettini* per avermi accolto nel modo migliore possibile nel suo gruppo di lavoro e per il numero di sfide cui mi ha sottoposto durante questi anni passati con lui. Non posso sapere cosa avrei imparato in un'altra esperienza lavorativa, ma posso garantire che difficilmente avrei potuto seguire lo stesso sviluppo di crescita professionale. Lo ringrazio inoltre per avermi dato tutta la libertà possibile nell'affrontare i problemi sottopostomi e per la disponibilità con cui ha seguito il lavoro, nonché per il valido contributo di idee ed esperienza.

Il secondo ringraziamento va a tutto il gruppo di ricerca con cui ho condiviso questo percorso di collaborazione, partendo dal *Prof. Roberto Strada*, altra vera anima del gruppo, per poi passare a tutti i miei *colleghi*. A loro va il mio ringraziamento sia per l'aiuto e disponibilità fornitomi nello sviluppo delle varie attività, sia per i momenti di svago che hanno accompagnato ogni giornata di lavoro. Rimanendo nell'ambito della vita di laboratorio, ringrazio anche tutti gli *studenti* dei corsi di Ingegneria Informatica e Meccanica che ho avuto modo di seguire in questi anni. Mi ha fatto piacere conoscere persone coi miei stessi interessi e spero di essere stato in grado di lasciar loro un buon ricordo delle esperienze condivise.

Un grazie grandissimo, infine, a tutti i membri della mia famiglia, che sono sempre stati i miei principali sostenitori nel corso dei miei studi, sia nei momenti felici, ma soprattutto in quelli più difficili che non sono certo mancati. Tra questi devo ringraziare in primo luogo *mia mamma Luisa e mio papà Gianni*, che mi hanno sempre supportato nel modo migliore possibile. A loro va la dedica principale di questo lavoro di tesi. In secondo luogo, per non far torto a nessuno, ringrazio tutti gli altri miei parenti: *nonni, zii e cugini*.

Al termine della tesi di laurea non avrei mai pensato di dovermi ritrovare a scrivere un altro lavoro di tesi e un'altra pagina di ringraziamenti come questa. Tuttavia, devo

dire che il *Dottorato di Ricerca* mi ha portato a vivere situazioni e risolvere problemi che mai avrei pensato di essere in grado di affrontare e che sempre porterò con me nel mio bagaglio di esperienza e nei miei ricordi. Il consiglio a chi legge queste poche parole è proprio quello che mi ha portato ad intraprendere il cammino scelto: non avere paura nell'affrontare situazioni nuove e cogliere tutte le occasioni di crescita che si presentano sul cammino. In poche parole, il solito motto di Orazio: *Carpe Diem!*

*Mattia Rossetti*

# Indice

<b>Abstract</b>	<b>ii</b>
<b>Ringraziamenti</b>	<b>iv</b>
<b>1 Introduzione</b>	<b>1</b>
1 La visione come sensore intelligente . . . . .	2
2 Vantaggi dei sistemi di visione . . . . .	4
3 Svantaggi dei sistemi di visione . . . . .	5
4 Guida Robot . . . . .	7
5 Motivazioni e Struttura della tesi . . . . .	11
 <b>I Elaborazione delle immagini</b>	
<hr/>	
<b>2 Acquisizione delle Immagini</b>	<b>19</b>
1 Sequenza e componenti dell'acquisizione immagini . . . . .	21
2 Modello della telecamera . . . . .	40
3 Calibrazione delle telecamere . . . . .	57
4 Conclusioni . . . . .	64
 <b>3 Tecniche di Elaborazione di Immagini 2D</b>	<b>67</b>
1 Segmentazione . . . . .	70
2 Descrizione . . . . .	92
3 Interpretazione . . . . .	105
4 Finestramento . . . . .	119
5 Conclusioni . . . . .	123
 <b>4 Acquisizione ed Elaborazione di Immagini 3D</b>	<b>127</b>
1 Classificazione dei compiti . . . . .	129
2 Acquisizione e rappresentazione dei dati . . . . .	131

3	Ricostruzione e Descrittori 3D . . . . .	142
4	Riconoscimento . . . . .	159
5	Conclusioni . . . . .	166

## II Visual Servoing

---

<b>5</b>	<b>Controllo in Asservimento Visivo</b>	<b>171</b>
1	Controllo in anello aperto . . . . .	174
2	Vision in the Loop . . . . .	191
3	Position-Based Visual Servoing . . . . .	198
4	Image-Based Visual Servoing . . . . .	206
5	Conclusioni . . . . .	220
<b>6</b>	<b>Metodologie per il Tracking Visivo</b>	<b>223</b>
1	Classificazione e sequenza d'elaborazione . . . . .	226
2	Algoritmi di tracciamento basati su feature . . . . .	235
3	Tracciamento basato su viste multiple di un modello . . . . .	247
4	Conclusioni . . . . .	270
<b>7</b>	<b>Approccio al Progetto di Regolatori in Asservimento Visivo</b>	<b>273</b>
1	Modello del controllo del motore DC . . . . .	275
2	Dinamica del sistema di visione . . . . .	302
3	Schemi di controllo in asservimento visivo . . . . .	315
4	Progetto dei regolatori . . . . .	323
5	Simulazione asse lineare con controllo IBVS . . . . .	331
6	Conclusioni . . . . .	345
<b>8</b>	<b>Criteri di Sviluppo di Componenti Software per l'Asservimento Visivo</b>	<b>347</b>
1	Acquisizione immagini . . . . .	351
2	Elaborazione immagini . . . . .	367
3	Controllore RTAI . . . . .	381
4	Conclusioni . . . . .	403

## III Validazione sperimentale

---

<b>9</b>	<b>Posizionamento di una Tavola Rotante</b>	<b>407</b>
----------	---	------------

---

1	Architettura del sistema . . . . .	409
2	Elaborazione immagini . . . . .	422
3	Sviluppo del controllo . . . . .	428
4	Prove sperimentali . . . . .	435
5	Conclusioni . . . . .	443
<b>10</b>	<b>Controllo di un Pendolo Inverso con Asservimento Visivo</b>	<b>445</b>
1	Descrizione del banco . . . . .	448
2	Controllo classico . . . . .	454
3	Controllo in asservimento visivo . . . . .	466
4	Elaborazione delle immagini . . . . .	477
5	Risultati sperimentali . . . . .	487
6	Conclusioni . . . . .	494
<b>11</b>	<b>Conclusioni e Sviluppi Futuri</b>	<b>495</b>
1	Risultati principali . . . . .	497
2	Tematiche aperte . . . . .	500
3	Sviluppi futuri . . . . .	501
	<b>Appendice A. Metodo dei minimi quadrati</b>	<b>505</b>
	<b>Appendice B. Assegnamento degli autovalori</b>	<b>507</b>
	<b>Bibliografia</b>	<b>516</b>
	<b>Sitografia</b>	<b>517</b>





# Elenco delle figure

1.1	<i>Manipolazione intelligente</i>	2
1.2	<i>Componenti di un sistema di Visione Industriale</i>	3
1.3	<i>Discipline convergenti in un algoritmo di visione</i>	6
1.4	<i>Schema generale della visione d'insieme</i>	8
1.5	<i>Esempio di visione 2D classica</i>	9
1.6	<i>Schema Vision in The Loop</i>	10
1.7	<i>Introduzione di un filo nella cruna di un ago</i>	11
2.1	<i>Componenti ed operazioni coinvolte nella formazione dell'immagine</i>	22
2.2	<i>Struttura di un sensore d'acquisizione immagini</i>	22
2.3	<i>Fenomeno del "blurring"</i>	24
2.4	<i>Schema Bayer di rappresentazione dei colori</i>	25
2.5	<i>Accoppiamento tra sensore d'acquisizione immagini e obbiettivo</i>	26
2.6	<i>Interlacciamento</i>	27
2.7	<i>Bus di comunicazione firewire</i>	30
2.8	<i>Riflessione della luce</i>	33
2.9	<i>Spettro delle radiazioni elettromagnetiche</i>	34
2.10	<i>Funzione di luminosità standard</i>	34
2.11	<i>Retroilluminazione</i>	35
2.12	<i>Illuminazione frontale</i>	36
2.13	<i>Shutter - esempio n. 1</i>	36
2.14	<i>Shutter - esempio n. 2</i>	37
2.15	<i>Effetti del guadagno</i>	37
2.16	<i>Variazione di luminosità</i>	37
2.17	<i>Nitidezza e messa a fuoco</i>	38
2.18	<i>Correzione gamma</i>	38
2.19	<i>Saturazione dei colori</i>	38
2.20	<i>Tonalità dei colori</i>	39

2.21	<i>Temperatura Colore</i>	39
2.22	<i>Bilanciamento del bianco</i>	40
2.23	<i>Modello Pin-Hole</i>	41
2.24	<i>Sistema di riferimento del piano immagine</i>	43
2.25	<i>Calcolo della relazione tra distanze focali obiettivo e modello</i>	44
2.26	<i>Lente sferica e distorsione radiale</i>	45
2.27	<i>Distorsioni radiali di un obiettivo fish-eye</i>	46
2.28	<i>Misura della distorsione radiale</i>	46
2.29	<i>Causa della distorsione tangenziale</i>	47
2.30	<i>Misura della distorsione tangenziale</i>	47
2.31	<i>Trasformazioni da piano immagine a spazio 3D</i>	48
2.32	<i>Problema della ricostruzione 3D</i>	49
2.33	<i>Ricostruzione planare</i>	50
2.34	<i>Ricostruzione con triangolazione attiva</i>	51
2.35	<i>Ricostruzione con triangolazione passiva</i>	53
2.36	<i>Corrispondenze tra modello e osservazione</i>	60
2.37	<i>Inversione di corrispondenze</i>	63
2.38	<i>Robustezza della stima dei minimi quadrati</i>	63
2.39	<i>Individuazione angoli su pattern di calibrazione</i>	64
2.40	<i>Procedura di calibrazione tramite apposito pattern</i>	65
2.41	<i>Applicazione dei parametri estrinseci stimati per la riduzione della distorsione</i>	65
3.1	<i>Sequenza di operazioni per l'elaborazione delle immagini</i>	68
3.2	<i>Scelta della soglia per segmentazione basata su sogliatura</i>	71
3.3	<i>Spazio di colore RGB</i>	73
3.4	<i>Spazio di colore HSV</i>	75
3.5	<i>Esempio di sottrazione dello sfondo</i>	77
3.6	<i>Esempio di filtro spaziale</i>	78
3.7	<i>Filtro passa-basso</i>	79
3.8	<i>Filtro mediano</i>	80
3.9	<i>Filtro passa-alto</i>	81
3.10	<i>Definizione matematica di bordo</i>	81
3.11	<i>Filtro di Sobel</i>	85
3.12	<i>Filtro gaussiano</i>	87

3.13	<i>Algoritmo di Canny: discretizzazione della direzione</i>	87
3.14	<i>Operatore di Canny</i>	88
3.15	<i>Rilevatore di Harris: contorni e angoli</i>	91
3.16	<i>Scale Invariant Features Transform</i>	91
3.17	<i>Speeded Up Robust features</i>	92
3.18	<i>Formazione dell'istogramma</i>	94
3.19	<i>Istogramma della luminosità di un'immagine</i>	94
3.20	<i>Algoritmo Mean Shift</i>	96
3.21	<i>Assi d'inerzia di un oggetto</i>	101
3.22	<i>Applicazione dei momenti per il calcolo delle caratteristiche geometriche</i>	102
3.23	<i>Sistemi di coordinate per rappresentazione dei contorni</i>	102
3.24	<i>Rappresentazione di contorni in coordinate cartesiane: chain code</i>	103
3.25	<i>Rappresentazione di contorni in coordinate polari</i>	103
3.26	<i>Rappresentazione di contorni in coordinate tangenziali: Contour Slope Sequence</i>	104
3.27	<i>Algoritmo di approssimazione polinomiale del contorno</i>	105
3.28	<i>Esempio di template matching</i>	108
3.29	<i>Spazio immagine e spazio dei parametri della trasformata di Hough</i>	110
3.30	<i>Vettore accumulatore per la trasformata di Hough lineare</i>	110
3.31	<i>Coordinate polari per la rappresentazione di una retta</i>	111
3.32	<i>Applicazione della trasformata di Hough</i>	112
3.33	<i>Parametrizzazione della trasformata di Hough generalizzata</i>	112
3.34	<i>Ricerca di oggetti bidimensionali</i>	113
3.35	<i>Corrispondenze per la stima della posa 3D</i>	114
3.36	<i>Sistema di riferimento di una finestra dell'immagine</i>	120
3.37	<i>Scelta della dimensione della finestra per una feature locale</i>	122
3.38	<i>Feature di un'immagine</i>	124
4.1	<i>Compiti tipici della visione 3D</i>	131
4.2	<i>Visione 3D: tracciamento con occlusioni</i>	132
4.3	<i>Calcolo della distanza con tempo di volo</i>	133
4.4	<i>Sensore ad ultrasuoni</i>	134
4.5	<i>Telecamera time-of-flight</i>	134
4.6	<i>Dispositivo per visione stereoscopica</i>	135

4.7	<i>Scanner 3D a triangolazione attiva</i>	136
4.8	<i>Sensore a luce strutturata, principio di funzionamento</i>	137
4.9	<i>Dispositivo per visione stereoscopica con ausilio di pattern luminoso</i>	138
4.10	<i>Projected Texture Stereo Vision: pattern luminoso</i>	138
4.11	<i>Nuvola di punti</i>	139
4.12	<i>Rendering di mesh</i>	140
4.13	<i>Voxelized Cloud</i>	140
4.14	<i>Range Image</i>	141
4.15	<i>Trasformazioni tra rappresentazioni 3D</i>	141
4.16	<i>Esempio di Registrazione 3D</i>	143
4.17	<i>Registrazione 3D: calcolo corrispondenze</i>	143
4.18	<i>Iterative Closest Point</i>	144
4.19	<i>Definizione di vettore normale</i>	145
4.20	<i>Significato geometrico della matrice di dispersione</i>	146
4.21	<i>Normali su range image</i>	147
4.22	<i>Curvatura di un profilo</i>	148
4.23	<i>Curvature principali</i>	148
4.24	<i>Shape Index di una superficie 3D</i>	149
4.25	<i>Descrittore SHOT</i>	152
4.26	<i>Descrittore PFH</i>	153
4.27	<i>Descrittore SC</i>	153
4.28	<i>Descrittore RSD</i>	154
4.29	<i>Descrittore Spin Image</i>	155
4.30	<i>Descrittore Spin Image: intero set di descrittori</i>	155
4.31	<i>Descrittore Multiple-cue</i>	156
4.32	<i>Descrittore CVFH con occlusioni</i>	158
4.33	<i>Descrittore ESF</i>	158
4.34	<i>Esempio di riconoscimento da nuvola di punti</i>	159
4.35	<i>Sequenza di riconoscimento</i>	160
4.36	<i>Camera Roll Histogram</i>	165
5.1	<i>Schema a blocchi del controllo visivo di un robot in anello aperto</i>	174
5.2	<i>Guida robot in anello aperto con oggetti movimentati da nastro trasportatore</i>	176
5.3	<i>Schema di una trasmissione</i>	178

5.4	<i>Rapporto attrito-velocità</i>	178
5.5	<i>Blocco Simulink di un attuatore</i>	179
5.6	<i>Blocco Simulink per la simulazione di un controllore di un robot</i>	180
5.7	<i>Possibili configurazioni della matrice jacobiana</i>	182
5.8	<i>Inserimento della cinematica inversa nello schema Simulink per il controllo del robot</i>	183
5.9	<i>Blocco Simulink del controllore a giunti indipendenti</i>	184
5.10	<i>Schema Simulink del controllore a giunti indipendenti</i>	184
5.11	<i>Utilizzo della dinamica al fine della simulazione</i>	185
5.12	<i>Schema Simulink per la simulazione del controllore di un robot</i>	190
5.13	<i>Schema a blocchi del controllo in asservimento visivo</i>	191
5.14	<i>Tipologie di controllo Visual Servoing</i>	195
5.15	<i>Tipologie di controllo in asservimento visivo in funzione della configurazione telecamera-manipolatore</i>	196
5.16	<i>Classificazione del controllo in asservimento visivo in funzione della posizione della telecamera</i>	196
5.17	<i>Schema del controllo PBVS</i>	199
5.18	<i>Trasformazioni geometriche coinvolte nel controllo PBVS</i>	199
5.19	<i>Esempio PBVS: traiettoria nello spazio immagine</i>	203
5.20	<i>Esempio di PBVS: traiettoria nello spazio di lavoro</i>	204
5.21	<i>Blocco Simulink per la simulazione della telecamera</i>	204
5.22	<i>Blocco Simulink per la simulazione del movimento dell'end-effector</i>	205
5.23	<i>Schema a blocchi Simulink del controllo PBVS</i>	205
5.24	<i>Schema del controllo in asservimento visivo Image-Based</i>	206
5.25	<i>Definizione del task di posizionamento IBVS nel piano immagine</i>	207
5.26	<i>Esempio IBVS: traiettoria nello spazio immagine</i>	213
5.27	<i>Esempio di IBVS: traiettoria nello spazio di lavoro</i>	214
5.28	<i>Esempio di IBVS: effetto della stima della distanza <math>Z</math></i>	215
5.29	<i>Esempio di omotetia a <math>180^\circ</math></i>	216
5.30	<i>Controllo IBVS: passaggio da un punto singolare</i>	216
5.31	<i>Controllo IBVS: passaggio vicino ad un punto singolare</i>	217
5.32	<i>Controllo IBVS: soluzione al passaggio vicino ad un punto singolare</i>	218
5.33	<i>Schema a blocchi Simulink del controllo IBVS</i>	219

6.1	<i>Esempio di tracking basato su modello</i>	224
6.2	<i>Macrotematiche relative al problema del tracciamento</i>	228
6.3	<i>Esempi di modellazione di oggetti per la costruzione di descrittori</i>	229
6.4	<i>Componenti e sequenza di operazioni di un modulo per il tracciamento di oggetti</i>	234
6.5	<i>Aggiornamento del modulo di tracciamento: predizione e aggiornamento</i>	235
6.6	<i>Schematizzazione dell'algoritmo Moving Edges Tracker</i>	236
6.7	<i>Algoritmo LK: esempio</i>	241
6.8	<i>Assunzioni dell'algoritmo Lucas-Kanade</i>	242
6.9	<i>Algoritmo LK ad 1 dimensione</i>	243
6.10	<i>Algoritmo LK a 2 dimensioni</i>	244
6.11	<i>Algoritmo LK: problema dell'apertura</i>	244
6.12	<i>Algoritmo LK piramidale</i>	245
6.13	<i>Piramidi gaussiana e laplaciana</i>	246
6.14	<i>Esempi di applicazioni planari</i>	249
6.15	<i>Distorsione di un oggetto planare al variare della vista</i>	251
6.16	<i>Distorsione di un oggetto tridimensionale</i>	252
6.17	<i>Grafo per la descrizione del modello 3D</i>	253
6.18	<i>Costruzione del descrittore a viste multiple</i>	256
6.19	<i>Sottoinsieme del grafo del descrittore considerato per il tracciamento del- l'oggetto</i>	261
6.20	<i>Limitazione della ricerca in una porzione dell'immagine</i>	262
6.21	<i>Modelli per l'oggetto planare considerato</i>	263
6.22	<i>Invarianza a traslazioni e rotazioni attorno all'asse Z per il modello planare</i>	264
6.23	<i>Riconoscimento dell'oggetto planare sottoposto a variazioni di scala</i>	265
6.24	<i>Stima della posa dell'oggetto planare in presenza di trasformazioni geome- triche più complesse</i>	266
6.25	<i>Modelli per l'oggetto tridimensionale considerato</i>	267
6.26	<i>Riconoscimento dell'oggetto tridimensionale sottoposto a diverse trasforma- zioni e problema della simmetria</i>	268
6.27	<i>Errata stima della posa dell'oggetto tridimensionale</i>	269
6.28	<i>Stima corretta della posa per l'oggetto tridimensionale</i>	269
7.1	<i>Frequenze di campionamento di un sistema multirate</i>	274

7.2	<i>Modelli del motore in corrente continua . . . . .</i>	276
7.3	<i>Schema a blocchi della funzione di trasferimento del motore in corrente continua . . . . .</i>	278
7.4	<i>Risposta in frequenza delle funzione di trasferimento del motore . . . . .</i>	279
7.5	<i>Schema Simulink del motore a corrente continua . . . . .</i>	280
7.6	<i>Curva caratteristica del motore . . . . .</i>	281
7.7	<i>Processo composto da sottosistemi in serie . . . . .</i>	282
7.8	<i>Controllo in cascata . . . . .</i>	282
7.9	<i>Schema a blocchi del controllo in cascata del motore in corrente continua .</i>	282
7.10	<i>Schema a blocchi del controllo in coppia del motore in corrente continua .</i>	283
7.11	<i>Schema a blocchi del controllo in velocità del motore in corrente continua .</i>	286
7.12	<i>Controllo in velocità: wind-up . . . . .</i>	287
7.13	<i>Saturazione della coppia . . . . .</i>	291
7.14	<i>Desaturazione dell'azione integrale per un regolatore PI . . . . .</i>	292
7.15	<i>Schema a blocchi del controllo in posizione del motore in corrente continua</i>	293
7.16	<i>Schema Simulink per l'implementazione del controllo del motore DC . . . .</i>	295
7.17	<i>Implementazione Simulink degli anelli di controllo in posizione e velocità .</i>	295
7.18	<i>Controllo in velocità del motore DC, massime prestazioni . . . . .</i>	298
7.19	<i>Controllo in velocità del motore DC, prestazioni contenute . . . . .</i>	299
7.20	<i>Controllo in posizione del motore DC, massime prestazioni . . . . .</i>	300
7.21	<i>Controllo in posizione del motore DC, prestazioni contenute . . . . .</i>	301
7.22	<i>Diagramma di Bode del ritardo puro e dell'approssimante di Padé di primo grado . . . . .</i>	304
7.23	<i>Schema di controllo digitale a campionamento dell'uscita e del riferimento</i>	305
7.24	<i>Campionatore periodico . . . . .</i>	306
7.25	<i>Mantenitore di ordine zero . . . . .</i>	307
7.26	<i>Risposta all'impulso dello ZOH . . . . .</i>	307
7.27	<i>Schema di controllo ibrido . . . . .</i>	308
7.28	<i>Diagramma di Bode del gruppo campionario-mantenitore puro e dell'approssimante di Padé di primo grado . . . . .</i>	310
7.29	<i>Schema Simulink del modello dinamico del sensore di visione . . . . .</i>	313
7.30	<i>Visual Servoing: schema di base . . . . .</i>	317
7.31	<i>Definizione del task di posizionamento delle immagini . . . . .</i>	318
7.32	<i>Definizione del compito di inseguimento . . . . .</i>	320

7.33	<i>FeedForward sul comando di moto</i>	321
7.34	<i>Feedforward sul comando di moto con stima della velocità del target</i>	322
7.35	<i>Funzione in anello aperto dell'asservimento visivo</i>	324
7.36	<i>Controllore Proporzionale: luogo delle radici</i>	325
7.37	<i>Controllore proporzionale: funzione in anello aperto</i>	326
7.38	<i>Controllore Proporzionale-Derivativo: luogo delle radici</i>	326
7.39	<i>Controllore Proporzionale-Derivativo: luogo delle radici, dettaglio</i>	327
7.40	<i>Controllore proporzionale-derivativo: funzione in anello aperto</i>	328
7.41	<i>Blocco Simulink per la stima della velocità degli oggetti target</i>	330
7.42	<i>Funzione di trasferimento in anello chiuso: posizionamento</i>	331
7.43	<i>Funzione di trasferimento in anello chiuso: inseguimento</i>	331
7.44	<i>Sistema ad un grado di libertà simulato</i>	332
7.45	<i>Interfaccia per l'inizializzazione della simulazione del task di asservimento visivo</i>	333
7.46	<i>Guida lineare: risposta scalino algoritmo IBVS , <math>\mu = 3</math></i>	336
7.47	<i>Guida lineare: risposta scalino algoritmo IBVS , <math>\mu = 5</math></i>	337
7.48	<i>Guida lineare: risposta scalino algoritmo IBVS , <math>\mu = 3</math> , con rumore</i>	338
7.49	<i>Guida lineare: inseguimento, movimento sinusoidale</i>	340
7.50	<i>Guida lineare: inseguimento, movimento triangolare</i>	341
7.51	<i>Guida lineare: tracking, movimento sinusoidale, feedforward</i>	343
7.52	<i>Guida lineare: inseguimento con feedforward, movimento triangolare</i>	344
8.1	<i>Definizione dei task in funzione dell'hardware</i>	348
8.2	<i>Scomposizione del controllo in task logici</i>	349
8.3	<i>Sequenza di acquisizione ed elaborazione immagini</i>	351
8.4	<i>Acquisizione ed elaborazione immagini in parallelo</i>	353
8.5	<i>Struttura di un buffer circolare</i>	357
8.6	<i>Struttura di uno swinging circolare</i>	358
8.7	<i>Schema degli oggetti software per la gestione dell'acquisizione</i>	363
8.8	<i>Schema delle componenti del software di tracciamento implementato</i>	368
8.9	<i>Struttura della libreria software openCv</i>	381
8.10	<i>Componenti di Linux RTAI</i>	382
8.11	<i>Spazio utente LX-RT nell'architettura RTAI</i>	384
8.12	<i>Schema delle classi per l'oggetto ControllerThread</i>	389



8.13	<i>Implementazione di una FIFO full duplex . . . . .</i>	397
9.1	<i>Rappresentazione del banco prova realizzato . . . . .</i>	408
9.2	<i>Componenti principali del sistema . . . . .</i>	409
9.3	<i>Figure sovrainpresse sul disco . . . . .</i>	411
9.4	<i>Telecamera Sony XCD V60-CR . . . . .</i>	411
9.5	<i>Obiettivo Tamron, 4.8 mm . . . . .</i>	412
9.6	<i>Motore brushless Mavilor BLS-055 . . . . .</i>	413
9.7	<i>Azionamento Infranor XtrapulsPac 230 V . . . . .</i>	414
9.8	<i>Scheda d'espansione per il bus firewire . . . . .</i>	418
9.9	<i>National Instruments PCI-6229 . . . . .</i>	418
9.10	<i>Pinout della scheda NI PCI-6229 . . . . .</i>	420
9.11	<i>Line Receiver . . . . .</i>	421
9.12	<i>Schema dei segnali . . . . .</i>	421
9.13	<i>Modalità di costruzione del modello . . . . .</i>	423
9.14	<i>Rotazioni dell'immagine del disco . . . . .</i>	424
9.15	<i>Segmentazione dell'immagine . . . . .</i>	425
9.16	<i>Individuazione dei blob e dei centri geometrici . . . . .</i>	426
9.17	<i>Individuazione dell'oggetto . . . . .</i>	426
9.18	<i>Rappresentazione del sistema in simulazione . . . . .</i>	428
9.19	<i>Schema di controllo del disco IBVS . . . . .</i>	430
9.20	<i>Disco, controllo IBVS, risposta allo scalino . . . . .</i>	431
9.21	<i>Schema di controllo del disco PBVS . . . . .</i>	432
9.22	<i>Luogo delle radici del sistema a tempo continuo . . . . .</i>	433
9.23	<i>Posizionamento del disco visto sul piano immagine . . . . .</i>	433
9.24	<i>Disco, controllo PBVS ideale, risposta allo scalino . . . . .</i>	434
9.25	<i>Luogo delle radici del sistema a tempo discreto . . . . .</i>	435
9.26	<i>Disco, controllo PBVS reale, risposta allo scalino . . . . .</i>	435
9.27	<i>Posizionamento del disco visto sul piano immagine con l'introduzione di rumore . . . . .</i>	436
9.28	<i>Selezione del target da tracciare per il controllo . . . . .</i>	436
9.29	<i>Posizionamento con tracking, <math>K_p = 7</math> . . . . .</i>	437
9.30	<i>Posizionamento con tracking, <math>K_p = 10</math> . . . . .</i>	438
9.31	<i>Posizionamento senza finestramento . . . . .</i>	439

9.32	<i>Sequenza di frame del task di posizionamento . . . . .</i>	441
9.33	<i>Sequenza di frame del task di posizionamento con perdita temporanea del tracciamento . . . . .</i>	442
9.34	<i>Tavola rotante industriale . . . . .</i>	444
10.1	<i>Punti d'equilibrio del pendolo . . . . .</i>	446
10.2	<i>Rappresentazione schematica del sistema . . . . .</i>	447
10.3	<i>Pendolo inverso inquadrato dalla telecamera di controllo . . . . .</i>	449
10.4	<i>Vista del banco pendolo inverso completo . . . . .</i>	450
10.5	<i>Banco pendolo inverso, particolare dei finecorsa . . . . .</i>	451
10.6	<i>Banco pendolo inverso, carrello . . . . .</i>	452
10.7	<i>Banco pendolo inverso, particolare del blocco motore . . . . .</i>	452
10.8	<i>Motore Parker SMB 42-60 . . . . .</i>	454
10.9	<i>Azionamento Parker SLVD1N . . . . .</i>	455
10.10	<i>Modello e forze del banco per il controllo pendolo inverso . . . . .</i>	456
10.11	<i>Modello semplificato del pendolo inverso . . . . .</i>	457
10.12	<i>Analisi della dinamica del pendolo in anello aperto . . . . .</i>	462
10.13	<i>Controllo del pendolo inverso con posizionamento dei poli . . . . .</i>	463
10.14	<i>Schema di controllo del pendolo inverso . . . . .</i>	463
10.15	<i>Introduzione dei filtri di posizione e velocità nello schema di controllo del pendolo inverso . . . . .</i>	463
10.16	<i>Diagrammi di Bode dei filtri di posizione e velocità del pendolo . . . . .</i>	464
10.17	<i>Pendolo inverso: angolo dell'asta in simulazione . . . . .</i>	465
10.18	<i>Pendolo inverso: posizione del carrello in simulazione . . . . .</i>	465
10.19	<i>Pendolo inverso: coppia motrice in simulazione . . . . .</i>	466
10.20	<i>Risultati sperimentali del controllo classico del pendolo inverso, <math>\omega_1 = 4 \text{ rad/s}</math>, <math>\omega_2 = 10 \text{ rad/s}</math> e <math>\xi = 0.9</math> . . . . .</i>	467
10.21	<i>Pendolo Inverso, simulazione: posizione e velocità angolare con sottocam- pionamento a <math>100 \text{ Hz}</math> . . . . .</i>	468
10.22	<i>Pendolo Inverso, simulazione: posizione e velocità del carrello con sotto- campionamento a <math>100 \text{ Hz}</math> . . . . .</i>	469
10.23	<i>Pendolo Inverso, simulazione: posizione di carrello e asta con sottocampio- namento a <math>50 \text{ Hz}</math> . . . . .</i>	470
10.24	<i>Schema di controllo vision-in-the-loop del pendolo inverso . . . . .</i>	471

10.25	<i>Pendolo inverso, simulazione: controllo del pendolo inverso con asservimen- to visivo a 50 Hz</i>	472
10.26	<i>Pendolo inverso, prova sperimentale: sottocampionamento a 50 Hz</i>	474
10.27	<i>Pendolo inverso, prova sperimentale: confronto tra sottocampionamento a <math>f_1 = 50 \text{ Hz}</math> e <math>f_2 = 200 \text{ Hz}</math></i>	475
10.28	<i>Pendolo inverso, prova sperimentale: sottocampionamento a 25 Hz</i>	476
10.29	<i>Schematizzazione delle feature del pendolo</i>	478
10.30	<i>Immagine del pendolo inverso ripresa dal sistema di visione</i>	478
10.31	<i>Sogliatura dell'immagine del pendolo inverso con <math>0 \leq H_{thr} \leq 179</math>, <math>S_{thr} = 0</math>, <math>243 \leq V_{thr} \leq 255</math></i>	480
10.32	<i>Sogliatura dell'immagine del pendolo inverso con <math>0 \leq H_{thr} \leq 179</math>, <math>0 \leq</math> <math>S_{thr} \leq 8</math>, <math>243 \leq V_{thr} \leq 255</math></i>	480
10.33	<i>Segmentazione del pendolo per sottrazione dello sfondo</i>	481
10.34	<i>Segmentazione del pendolo per sottrazione di immagini consecutive</i>	483
10.35	<i>Filtraggio finale dell'immagine differenza del pendolo</i>	484
10.36	<i>Risultato dell'applicazione della trasformata di Hough per la rilevazione del pendolo</i>	485
10.37	<i>Problema dell'individuazione di due pendoli</i>	486
10.38	<i>Algoritmo per la discriminazione di pendolo reale e virtuale nella differenza tra immagini consecutive</i>	486
10.39	<i>Confronto tra le letture delle variabili di stato dei sensori di posizione e del sistema di visione</i>	488
10.40	<i>Sequenza di tracciamento del pendolo inverso</i>	489
10.41	<i>Stima della posizione angolare dell'asta per l'asservimento visivo</i>	490
10.42	<i>Stima della posizione del carrello per l'asservimento visivo</i>	491
10.43	<i>Stima delle velocità del carrello e dell'asta effettuata dal sistema di visione durante il controllo in asservimento visivo</i>	492
11.1	<i>Simulazione di un robot SCARA</i>	502
11.2	<i>Legge di moto ottenuta per il task di posizionamento del robot SCARA</i>	503
11.3	<i>Errore nel piano immagine per il task di posizionamento del robot SCARA</i>	503
B.1	<i>Schema di controllo con retroazione dell'uscita</i>	507
B.2	<i>Schema di controllo con retroazione dello stato</i>	508
B.3	<i>Schema per l'assegnamento degli autovalori con stato misurabile</i>	508



# Elenco delle tabelle

2.1	Tipi di omografia . . . . .	60
4.1	Descrittori Locali 3D . . . . .	151
5.1	Tipologie di controllo in asservimento visivo . . . . .	192
7.1	Dati tecnici del motore Parker SMB40-6000 . . . . .	277
9.1	Telecamera Sony XCD V60-CR, dati tecnici . . . . .	412
9.2	Motore brushless Mavilor BLS-055, dati tecnici . . . . .	413
9.3	Motore brushless Mavilor BLS-055, encoder . . . . .	414
9.4	Azionamento Infranor XtrapulsPac 230 V, morsettiera freno (X8) . . . . .	415
9.5	Azionamento Infranor XtrapulsPac 230 V, morsettiera potenza (X9) . . . . .	416
9.6	Azionamento Infranor XtrapulsPac 230 V, IO . . . . .	417
10.1	Banco pendolo inverso, dimensioni . . . . .	453
10.2	Motore Parker SMB 42-60, dati tecnici . . . . .	454
10.3	Azionamento Parker SLVD1N, dati tecnici . . . . .	455



# Capitolo 1

## Introduzione

### Indice

---

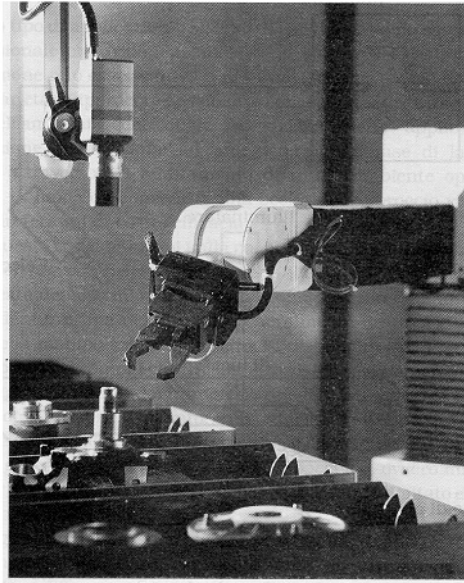
1	La visione come sensore intelligente . . . . .	<b>2</b>
2	Vantaggi dei sistemi di visione . . . . .	<b>4</b>
3	Svantaggi dei sistemi di visione . . . . .	<b>5</b>
4	Guida Robot . . . . .	<b>7</b>
4.1	Visione classica . . . . .	7
4.2	Visual Servoing . . . . .	9
5	Motivazioni e Struttura della tesi . . . . .	<b>11</b>

---

Nel corso degli ultimi anni nel settore dell'automazione industriale si è potuto osservare un notevole incremento nell'utilizzo di sistemi di visione poiché essi risultano applicabili alla soluzione di svariati problemi, quali:

- controllo qualità;
- selezione e scarto;
- lettura di codici a barre e alfanumerici;
- misura;
- guida robot.

Relativamente alla *guida robot* (figura 1.1), in anni ancora più recenti si sono ampiamente diffusi i termini di *meccatronica* e *dispositivi meccatronici*. Spesso si pensa che questo termine indichi solo una commistione di meccanica ed elettronica, mentre in realtà per dispositivo meccatronico si intende un dispositivo in cui diverse discipline come meccanica, elettronica ed informatica sono perfettamente integrate per rendere possibile lo sviluppo



*Figura 1.1: Manipolazione intelligente*

di sistemi per l'esecuzione di compiti complessi. In questo senso, quindi, l'integrazione di un sistema di visione con un robot non prevede solamente l'affiancamento di una telecamera al sistema di controllo del robot, ma bensì l'introduzione delle informazioni visive direttamente nel ciclo di controllo della traiettoria del manipolatore. Si può di conseguenza parlare di una vera e propria *manipolazione intelligente*.

## 1 La visione come sensore intelligente

In primissima analisi è possibile scomporre un sistema robotico in alcuni suoi componenti:

- *Componenti meccaniche*: sono la struttura portante del robot;
- *Attuatori*: permettono la movimentazione delle parti meccaniche;
- *Sensori* attraverso cui il dispositivo mecatronico è in grado di interagire con l'ambiente che lo circonda;
- *Sistema di controllo*: è il software che fornisce l'*intelligenza* attraverso cui la macchina diviene in grado di risolvere i problemi per cui è stata progettata.

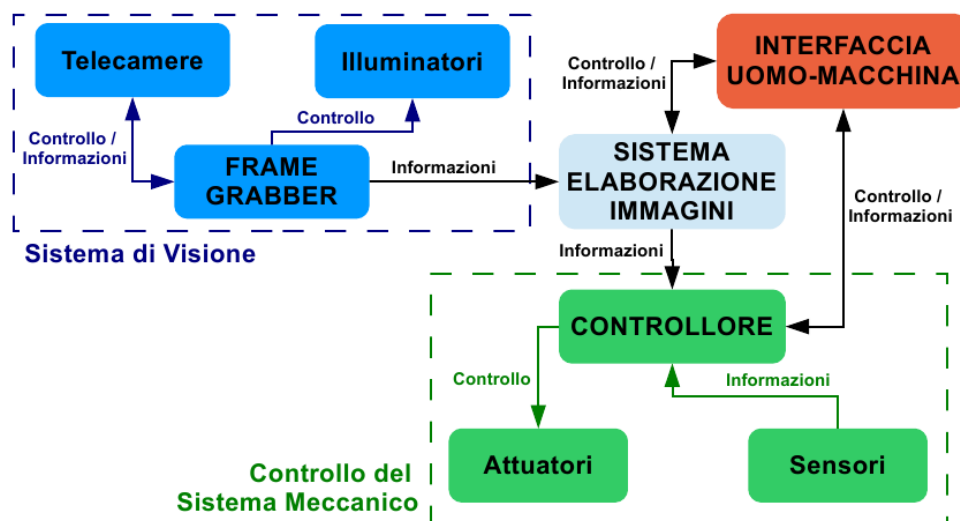
In questo lavoro di tesi l'attenzione è stata posta sull'analisi delle problematiche relative all'acquisizione di informazioni sull'ambiente circostante il manipolatore e alla loro integrazione nel sistema di controllo dello stesso. In particolare, l'utilizzo della visione artificiale permette di ottenere informazioni più significative rispetto ad altri sensori in un numero elevato di applicazioni. Tuttavia, per raccogliere tale informazione non è sufficiente leggere



ed elaborare un singolo segnale, ma il processo di acquisizione passa attraverso più fasi successive. L'ultima di queste fasi concerne l'utilizzo dell'informazione per il controllo del robot.

Tutta la serie di operazioni da compiere dall'acquisizione delle immagini fino alla fruizione del loro contenuto costituiscono un'operazione piuttosto complessa, pertanto si rende necessario un lavoro di raccolta e sintesi delle varie problematiche.

Un sistema di visione integrato nel controllo di un sistema meccanico, infatti, può essere suddiviso in diverse componenti a se stanti, le quali necessitano, tuttavia, di una stretta comunicazione per il completamento del compito per cui sono state impiegate.



*Figura 1.2: Componenti di un sistema di Visione Industriale*

In particolare, come si può vedere in figura 1.2, un sistema di visione industriale è composto da:

- *Sistema di acquisizione immagini.* È costituito dai dispositivi atti all'acquisizione vera e propria delle immagini (telecamere) e dagli strumenti che permettono di agevolarne la successiva elaborazione (illuminatori). L'acquisizione e memorizzazione delle immagini può avvenire direttamente a bordo del dispositivo oppure essere demandata ad un calcolatore (frame grabber) il cui compito è il controllo della sola acquisizione.
- *Sistema di elaborazione immagini.* Il suo compito è quello di estrarre le informazioni dalle immagini messe a disposizione dal frame grabber. Necessita pertanto di un canale di comunicazione col sistema di acquisizione immagini per ricevere i segnali di input per l'elaborazione e di un altro canale col controllore del sistema meccanico per fornire l'informazione necessaria a quest'ultimo per il completamento del task.

- *Controllore*. Rappresenta il centro nevralgico dedicato al controllo del robot. Pertanto, la sua funzione principale è quella di raccogliere tutte le informazioni provenienti dai sensori, sistema di visione incluso, al fine di calcolare i movimenti da imporre alle parti meccaniche per mezzo degli attuatori.
- *Interfacciamento con l'utente*. Fornisce il punto d'ingresso per l'interazione tra la macchina e il suo utilizzatore. L'interfacciamento rende possibile sia la descrizione e programmazione del compito da assolvere, sia il suo completo monitoraggio nelle varie fasi.

## 2 Vantaggi dei sistemi di visione

Come accennato nel paragrafo precedente, il primo vantaggio dell'utilizzo di sensori di visione risiede nella possibilità di estrarre svariate informazioni di diverso tipo da un'unica fonte, ovvero l'immagine. Ad esempio, a seconda del tipo di elaborazione effettuata, del numero e del tipo di dispositivi utilizzati, è possibile estrarre contemporaneamente informazioni su forma, dimensioni e volume degli oggetti. Questa ricchezza di contenuto porta ad un aumento della *flessibilità* del sistema meccanico. Basti pensare, come esempio, ad un robot incaricato di manipolare diversi tipi di oggetti: grazie al sistema di visione il robot riesce ad adattarsi agli oggetti che si trova di fronte senza l'intervento da parte di operatori e/o modifiche al programma da eseguire. In aggiunta, tale flessibilità viene raggiunta senza un incremento eccessivo della complessità del sistema, pertanto l'uso dei sistemi di visione rappresenta una soluzione *compatta* e *robusta* a diversi tipi di problemi.

Il secondo vantaggio viene messo in evidenza passando al punto di vista di chi deve lavorare col manipolatore, ovvero il suo operatore e il programmatore dello stesso. Dato che generalmente il senso più utilizzato dall'uomo è proprio la vista, l'utilizzo di immagini e video può facilitare enormemente l'*interfacciamento uomo-macchina*: monitorare la corretta esecuzione di un compito assegnato ad una macchina avendo a disposizione delle immagini è più semplice rispetto a monitorare il corretto funzionamento di un sensore di diverso tipo.

Un altro vantaggio si può dire che risieda direttamente nel funzionamento intrinseco delle telecamere. Queste ultime sono sensori *contactless*, ovvero non è richiesto un contatto tra il sensore e l'oggetto di indagine per ottenere le informazioni desiderate. Questa caratteristica dei dispositivi di visione è particolarmente apprezzata in quelle applicazioni

in cui gli oggetti da misurare sono fragili (ad esempio frutta e verdura) e applicazioni in cui l'utilizzo di altri sensori può portare all'insorgere di errori (per esempio l'uso di un encoder su un nastro sottoposto ad oscillazioni).

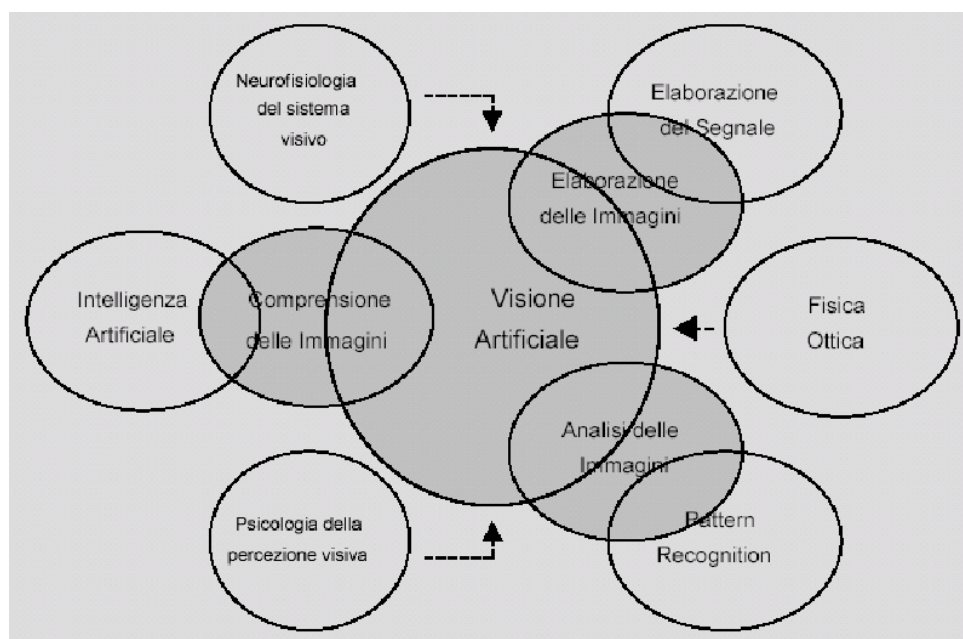
Infine, a tutto questo si aggiunge un vantaggio di tipo *economico-tecnologico*. Dal punto di vista tecnologico si sta osservando un continuo aumento delle prestazioni dei sensori per l'imaging sia dal punto di vista delle risoluzioni che del framerate raggiungibili. Di pari passo, inoltre, stanno proliferando anche le soluzioni software: librerie specifiche per la visione sempre più complete e performanti, protocolli e interfacce di comunicazione studiati appositamente per la trasmissione di immagini. . . . Del resto, come in altre applicazioni, la stessa crescita tecnologica si sta avendo anche sull'hardware utilizzabile per l'elaborazione delle immagini. A fronte di questi miglioramenti, dal lato economico si sta invece assistendo ad una diminuzione dei costi di tali dispositivi, tanto che spesso è più conveniente acquistare una telecamera e investire tutti gli sforzi sullo sviluppo degli algoritmi di visione necessari piuttosto che dotare il sistema automatico di un sensore per la rilevazione diretta dell'informazione di interesse.

### 3 Svantaggi dei sistemi di visione

Fin qui si è discusso di quanto può essere vantaggioso utilizzare un sensore di visione in un'applicazione industriale, ma d'altro canto è utile sottolineare anche che la gestione di un sensore basato su visione è più complessa rispetto a sensoristica di tipo diverso. La ricerca di alcuni punti di partenza per la gestione della complessità delle informazioni, infatti, è una delle motivazioni dello sviluppo di questo lavoro di tesi.

Il grande interesse cresciuto negli ultimi anni per la tematica della visione artificiale ha portato come principale risultato allo sviluppo da parte di enti privati e di ricerca di un elevatissimo numero di algoritmi e soluzioni ai problemi più tipici. Spesso, poi, gli algoritmi non rappresentano una soluzione migliore ad un determinato tipo di problema, ma semplicemente una soluzione diversa. Per i sistemi di visione lo svantaggio principale è dato proprio dalla numerosità e complessità degli algoritmi esistenti: spesso non esiste un unico modo di estrarre un certo tipo di informazione da un'immagine. La mancanza di una soluzione unica rende sempre difficile la scelta dell'algoritmo ottimo per un certo problema e generalmente l'algoritmo vincente è quello col miglior compromesso tra *qualità delle informazioni* estratte e *tempo* necessario all'estrazione di queste informazioni. In questi

casi si è solito dire che la *visione artificiale* è ancora più un'arte piuttosto che una scienza ingegneristica. Nella scelta di un algoritmo di visione si fa spesso uso del classico approccio *trial and error*: scelto un algoritmo, si prova ad applicarlo ad un particolare problema per valutare se la sua applicazione può essere utile a mettere in evidenza alcune informazioni da estrarre dall'immagine. A tutto questo si va ad aggiungere la complicazione intrinseca di un algoritmo di visione: per implementare un algoritmo di visione nel modo migliore è necessario avere padronanza di molte discipline diverse (figura 1.3).



**Figura 1.3:** Discipline convergenti in un algoritmo di visione

Un altro svantaggio può essere ancora una volta ricondotto alla natura attraverso la quale le informazioni vengono presentate da un sistema di visione: l'immagine. Nei paragrafi precedenti si è evidenziato che, all'interno di un dispositivo meccatronico, le telecamere sono in prima e ultima istanza dei sensori, sostituibili e/o affiancabili ad altri sensori. Da questo punto di vista, misurare la velocità di una ruota, la posizione angolare del giunto di un robot o l'intensità dei milioni di pixel che possono formare un'immagine sono tutte attività analoghe, atte essenzialmente all'acquisizione di informazioni. Per molte applicazioni robotiche, se il numero di dati che devono essere acquisiti in *tempo reale* è elevato, per le applicazioni che includono un sistema di visione tale numero è ancora più elevato (nell'ordine di decine o centinaia di  $MB/s$ ). Anche se la capacità di elaborazione dell'hardware moderno è sempre più elevata, quando si parla di applicazioni *real-time* o di *sistemi embedded* la potenza di calcolo e/o il tempo a disposizione per l'elaborazione possono essere limitati. Ancora una volta risulta evidente che, nella valutazione di un

algoritmo di visione, è sempre necessario tenere conto del compromesso tra qualità delle informazioni estratte ed efficienza nell'estrazione delle stesse.

Non solo lo sviluppo del software di un sistema di visione, ma anche l'installazione del sistema di visione può essere sottoposta ad alcune problematiche. Il problema tipico con cui si ha a che fare è quello dell'illuminazione: nel caso delle immagini il rumore sul segnale è rappresentato dalla luce ambientale. Qui la soluzione tipica consiste nel limitare l'influenza dei disturbi ambientali cercando di costruire ambienti di lavoro in cui la luce è controllata (ad esempio tramite l'utilizzo di illuminatori), ma questa difficoltà di adattamento a diverse condizioni ambientali costituisce ad oggi il limite più grande all'utilizzo dei sistemi di visione.

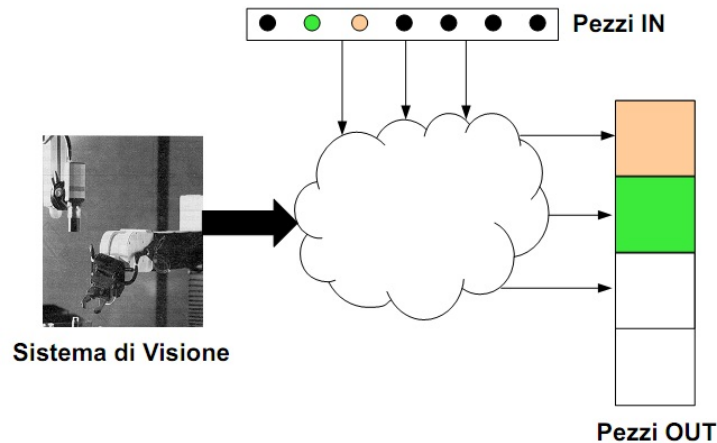
## 4 Guida Robot

In questa tesi, relativamente al problema di trasformare le informazioni derivanti dalla visione in informazioni utilizzabili dal sistema meccanico, ci si intende concentrare sulle problematiche concernenti la *guida robot*. In questo campo applicativo si possono distinguere principalmente due tipologie di applicazione delle tecniche di visione artificiale:

- *Visione classica*: problemi in cui il sistema di visione deve individuare posizione e orientamento di una serie di oggetti. Tipicamente rientrano in questa categoria i problemi di *pick and place* e *packaging*.
- *Vision in the Loop* (o *Visual Servoing*): problemi in cui la visione è utilizzata per retroazionare i movimenti di un manipolatore in base all'osservazione di un singolo oggetto ed ottenere così posizionamenti più precisi sull'oggetto. Esempi tipici sono i microposizionamenti richiesti per la manipolazione di componenti microscopiche o interventi chirurgici.

### 4.1 Visione classica

Il problema della visione d'insieme degli oggetti da manipolare fa riferimento alla seguente situazione tipica: gli oggetti in entrata nella cella (su nastro trasportatore o piazzola d'appoggio) devono essere presi dal manipolatore e spostati in un certo ordine o con certi criteri (*"pick and place"*, figura 1.4).



*Figura 1.4: Schema generale della visione d'insieme*

#### 4.1.1 Descrizione del problema

In un'applicazione tradizionale di integrazione tra visione e sistema meccanico il sistema di visione acquisisce una singola immagine e individua la posizione e l'orientamento (nel sistema di coordinate assolute) degli oggetti da individuare e manipolare. Successivamente il sistema di visione trasmette queste coordinate ad un robot manipolatore, il quale si occupa di eseguire il movimento richiesto nel modo più preciso possibile. Si può fare un'analogia col caso di una persona che, per prendere un oggetto da una scrivania, guarda prima dove questo si trova per poi chiudere gli occhi e provare a prendere l'oggetto.

#### 4.1.2 Limiti

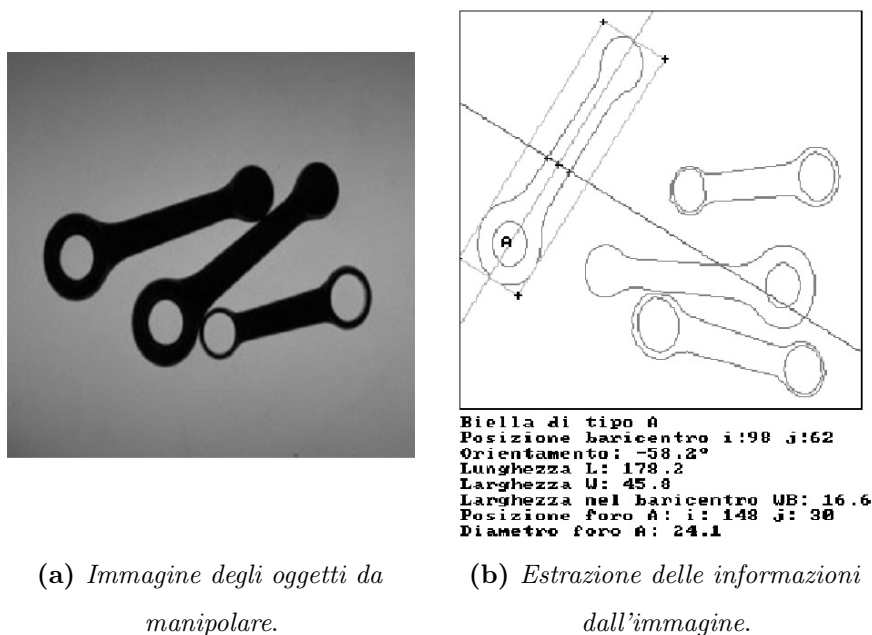
Un primo limite è dato dal fatto che l'operazione ha successo solamente se l'oggetto in questione è sufficientemente grande. Altri fattori che influenzano il successo dell'operazione sono:

- La conoscenza della posizione della telecamera relativamente al robot e dei parametri di proiezione (estrinseci e intrinseci) della telecamera stessa.
- La conoscenza del field of view della telecamera (è richiesta quindi una sorta di calibrazione ...).
- La capacità del robot di muoversi nel modo più accurato (e ripetibile) possibile verso una posizione  $XYZ$ .

Dal momento che non viene eseguita nessuna verifica del fatto che il movimento sia stato eseguito correttamente, il sistema deve assumere che il processo abbia avuto successo per procedere al passo successivo.

### 4.1.3 Esempio applicativo

Su una linea di produzione sono presenti due tipi di bielle per motociclette, aventi rispettivamente uno o due fori in corrispondenza della testa (figura 1.5). Un robot manipolatore deve prelevare le bielle dalla linea e smistarle in funzione del tipo. Si tratta di un problema classico, che richiede fundamentalmente l'individuazione di posizione ed orientamento degli oggetti unitamente alla capacità di riconoscere un oggetto all'interno di un insieme limitato di oggetti possibili.



**Figura 1.5:** Esempio di visione 2D classica

## 4.2 Visual Servoing

Il termine *Vision in the loop* (o *visual servoing*) si riferisce alle metodologie, tecniche e problemi che prevedono l'utilizzo della visione artificiale per la chiusura di anelli di controllo in posizione. Nell'ambito della manipolazione, una branca di problemi si riferisce alla correzione della presa di oggetti, i quali, per essere lavorati correttamente, devono necessariamente essere presi in un determinato modo.

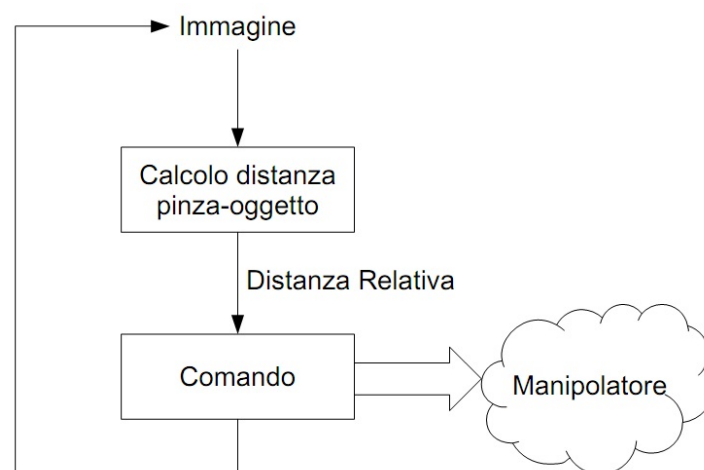
### 4.2.1 Descrizione del problema

La prima differenza col problema precedentemente trattato risiede nel sistema di riferimento più corretto da adottare: se la necessità è quella di correggere la posizione e

l'orientamento di presa, allora è possibile ed è più conveniente avere una telecamera solida-  
le coll'organo di presa. In questo modo, ci si trova a che fare con un sistema di riferimento  
relativo pinza-oggetto. Questo introduce potenzialità in più rispetto alla visione classica;  
in quest'ultima la posizione della telecamera è tipicamente statica e la posizione dell'end-  
effector del manipolatore deriva solo da calcoli cinematici a partire dalla posizione angolare  
dei suoi giunti; nel caso del visual servoing, invece, è possibile far muovere la telecamera  
insieme al manipolatore.

Rispetto alla tradizionale guida robot, cambia soprattutto il modo di pensare il  
procedimento di presa, il quale si avvicina maggiormente ai ragionamenti che normalmente  
compie un essere umano quando si accinge a prendere un oggetto. Per prendere un oggetto,  
un essere umano guarda contemporaneamente mano e oggetto per determinare la distanza  
relativa tra essi e, successivamente, eseguire un movimento incrementale verso l'oggetto. Un  
sistema di visione automatizzato ragiona allo stesso modo: si acquisisce un'immagine e si  
calcola la distanza relativa da percorrere in modo che il software real-time possa ordinare  
una serie di comandi di movimento incrementali all'end-effector del robot. Mentre il  
movimento viene eseguito, il software continua ad analizzare le nuove immagini e aggiorna  
i comandi di movimento (figura 1.6). Questo procedimento continua fino a quando il  
risultato ottenuto è soddisfacente: nel caso degli organi di presa, ad esempio, quando  
l'oggetto viene afferrato in modo corretto.

Il grande vantaggio di questa tecnica di guida robot rispetto a quella tradizionale risiede nel  
fatto che la precisione e ripetibilità del robot incaricato della manipolazione dell'oggetto  
non deve essere elevata come nel caso della guida robot in coordinate assolute.



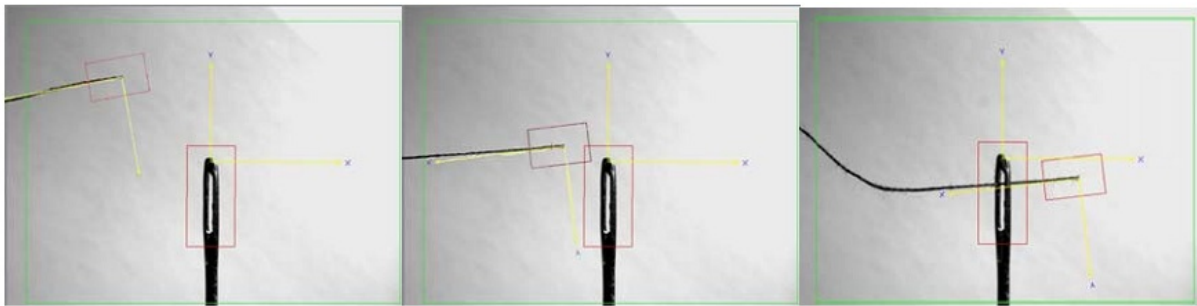
**Figura 1.6:** *Schema Vision in The Loop*



#### 4.2.2 Esempio: introdurre un filo nella cruna di un ago

Il problema è il seguente: una pinza deve afferrare un filo e, successivamente, introdurlo nella cruna di un ago (figura 1.7). E' un problema impossibile da risolvere utilizzando sistemi di *visione classica* perché si ha a che fare con oggetti molto piccoli. Al contrario, utilizzando un sistema di riferimento relativo, ovvero tramite *vision in the loop* è possibile trovare una soluzione a questo problema.

Il sistema di visione analizza una prima immagine e calcola le posizioni relative dell'ago e del filo. A questo punto, invece di mandare un singolo comando di movimento assoluto, si invia una serie di comandi di piccoli movimenti incrementali. Appena la pinza si mette in moto, il software di elaborazione prende in considerazione molte più immagini aggiornando i comandi di moto. Il processo continua fino a quando il sistema di visione conferma che il processo stesso è stato portato a termine.



*Figura 1.7: Introduzione di un filo nella cruna di un ago*

## 5 Motivazioni e Struttura della tesi

Per fronteggiare il problema dell'integrazione di sistemi di visione con dispositivi mecatronici complessi è necessario affrontare tematiche afferenti a discipline diverse, tra cui:

- Acquisizione di immagini in tempo reale.
- Elaborazioni di immagini al fine di riconoscere oggetti e collocarli correttamente all'interno dell'area di lavoro.
- Controllo di attuatori in tempo reale per la trasmissione del movimento desiderato agli organi meccanici.
- Esecuzione di simulazioni complesse per l'analisi degli algoritmi di generazione delle traiettorie.

- Implementazione di software real-time distribuito per l'integrazione del controllore del dispositivo mecatronico con il sistema di visione.

Generalmente, le pubblicazioni che trattano gli argomenti di guida robot e visual servoing cercano di proporre soluzioni innovative per un ben specifico problema. Quello che sembra mancare è una sorta di sintesi delle soluzioni a disposizione: ognuno fornisce la propria soluzione che si va ad aggiungere alla grande vastità di soluzioni già note in letteratura. Tutto questo rende sempre più difficile, per i neofiti, l'approccio alle tecniche di asservimento visivo.

Limitandosi ad altre tesi di dottorato prese in analisi, ad esempio, [31] si concentra soprattutto sull'implementazione dell'algoritmo per il riconoscimento degli oggetti, mentre [37] pone maggiore attenzione sulla presentazione del sistema sviluppato per una ben specifica applicazione. Al fine di presentare una trattazione completa delle diverse tematiche afferenti al problema dell'asservimento visivo, uno dei principali riferimenti di questo lavoro di tesi è [16]. Un'altra fonte d'interesse per lo studio della tematica in tutta la sua complessità è [17].

Scopo di questa tesi è dimostrare la validità dell'approccio ad asservimento visivo per la soluzione di problemi di guida robot ed analizzare un set abbastanza ristretto di soluzioni (sia riguardo all'elaborazione delle immagini che al controllo dei robot) che permettono però di affrontare la maggior parte dei problemi di guida robot. Il contributo principale di questo lavoro di tesi risulta essere un tentativo di *ingegnerizzazione* del problema del visual servoing. Tale processo verrà completato individuando:

- Le problematiche e i passi da seguire per giungere ad una significativa elaborazione delle immagini a partire dall'acquisizione delle stesse, indipendentemente dal tipo di sensore utilizzato.
- Gli algoritmi di visione più adatti ad essere utilizzati in applicazioni in tempo reale senza perdita di efficacia. Un contributo interessante e innovativo verrà dato per gli algoritmi di analisi di superfici 3D, messi a disposizione dai più moderni sensori di acquisizione immagini e per cui in anni recenti sta crescendo l'interesse del mondo industriale.
- La classificazione dei principali algoritmi di asservimento visivo e il loro campo applicativo.
- I parametri fondamentali per la modellazione per l'intero sistema di controllo, dal modello della formazione delle immagini fino al modello della parte meccanica.

- L'architettura di sistema di un generico controllore di un dispositivo mecatronico ad intelligenza distribuita.

Visto il grande numero e, soprattutto, la diversità delle varie tematiche da affrontare, si è deciso di suddividere la tesi in tre parti:

## **PARTE 1: Elaborazione delle immagini**

- **Acquisizione delle Immagini.**

L'acquisizione delle immagini è una tematica che viene spesso sottovalutata in lavori riguardanti il controllo con asservimento visivo. Tuttavia, l'acquisizione di dati in tempo reale ricopre un ruolo fondamentale nello sviluppo di un sistema di controllo retroazionato. Pertanto, in questo paragrafo vengono presentate alcune considerazioni relative ad una possibile struttura software che permette di rendere trasparente il tempo da dedicare all'acquisizione di immagini. Inoltre, vengono descritti il modello prospettico utilizzato per rappresentare la proiezione di una scena tridimensionale su un piano immagine bidimensionale e le modalità di calibrazione dei parametri di tale modello. Vengono infine fornite informazioni riguardanti i sensori 2D e la formazione dell'immagine.

- **Tecniche di Elaborazione di immagini 2D.**

In questo capitolo si elencano alcune delle classiche tecniche di elaborazione di immagini bidimensionali. Queste metodologie rappresentano la base di qualsiasi algoritmo di visione e vedranno la loro applicazione nei capitoli successivi. Visto l'elevato numero di algoritmi di visione 2D esistenti, il lavoro che si è cercato di svolgere è una sintesi degli algoritmi maggiormente efficaci ed utili nell'estrazione e descrizione di oggetti bidimensionali all'interno di una scena. Tali algoritmi rappresentano la base per la descrizione delle tecniche di tracciamento di oggetti su un flusso video e trovano la loro applicazione negli algoritmi "ad hoc" sviluppati per gli esempi di controllo con asservimento visivo prodotti nel corso del lavoro di tesi.

- **Acquisizione ed Elaborazione di Immagini 3D.**

Al contrario dell'elaborazione di immagini bidimensionali, il settore dell'analisi di *nuvole di punti* tridimensionali è abbastanza nuovo. Questo ambito di ricerca della

visione artificiale sta prendendo sempre più piede in ambito industriale grazie all'arrivo sul mercato di sensori sempre più performanti che permettono di acquisire informazioni sulla scena inquadrata direttamente in coordinate tridimensionali. I dispositivi che permettono di ottenere direttamente informazioni tridimensionali delle scene si differenziano dalle classiche telecamere; in questo capitolo si fa una breve descrizione delle tipologie di questi dispositivi. In seguito vengono presentati i principali algoritmi per la descrizione e il riconoscimento di oggetti tridimensionali a partire da osservazioni tridimensionali. Il concetto che si vuole mettere in evidenza riguarda il fatto che questi algoritmi rappresentano l'estensione dei classici algoritmi di visione 2D al caso tridimensionale e quindi possono prenderne il posto nella sequenza di operazioni per il tracciamento di oggetti all'interno di una scena senza intaccare minimamente le altre componenti.

## PARTE 2: Visual Servoing

- **Controllo in Asservimento Visivo.**

Come nel caso degli algoritmi di visione, non esiste un modo univoco per utilizzare le informazioni visive nell'anello di controllo di un manipolatore. Le differenze riguardano principalmente la configurazione e disposizione dei sistemi di visione e la modalità di inserimento delle informazioni visive nell'anello di controllo del sistema meccanico. Pertanto, è innanzitutto necessario presentare le tematiche presenti nell'affrontare questo problema e fornire una prima classificazione delle differenti tipologie di controllo note in letteratura. Il principale problema del controllo con asservimento visivo concerne la mancanza di un legame diretto tra l'informazione proveniente dai sensori e il segnale da controllare. Per tale motivo, un'ulteriore elaborazione delle informazioni è necessaria per l'implementazione del controllore. Questa elaborazione implica essenzialmente l'applicazione delle trasformazioni geometriche inverse alla trasformazione prospettica alla base della formazione dell'immagine. Questa trasformazione fa parte della legge di controllo del sistema meccanico, ma ancora non include gli aspetti dinamici del sistema, pertanto ci si riferisce ad essa come alla *cinematica* del controllo in asservimento visivo.

- **Metodologie per il Tracking Visivo.**

I compiti di guida robot prevedono generalmente l'interazione con parti in movimento, siano esse gli oggetti da manipolare o il manipolatore stesso. Gli algoritmi presentati precedentemente devono quindi essere adattati all'estrazione di informazioni in tempo reale da un flusso di immagini acquisite ad intervalli di tempo differente. Questo compito viene svolto dagli algoritmi di tracciamento. In questo capitolo vengono estrapolati i componenti base di tali algoritmi e se ne vede l'applicazione in due tipologie differenti. Infine, viene presentato un algoritmo originale sviluppato appositamente per questo lavoro di tesi.

- **Approccio al Progetto di Regolatori in Asservimento Visivo.**

L'asservimento visivo introduce essenzialmente un sensore di visione direttamente nel ramo di retroazione del controllore del movimento di un sistema meccanico. Tuttavia, questa operazione implica l'introduzione di componenti non lineari all'interno del controllore e, pertanto, porta all'insorgere di effetti collaterali nel controllo. In questo capitolo si presenta la modellazione delle non linearità del sistema di visione e si forniscono gli strumenti fondamentali per il corretto progetto di un sistema di asservimento visivo.

- **Criteri di Sviluppo di Componenti Software per l'Asservimento Visivo.**

In questo capitolo vengono presentati gli approcci seguiti per la progettazione ed implementazione della libreria software utilizzata per la realizzazione delle applicazioni per la validazione sperimentale. In particolare, viene innanzitutto mostrata la struttura degli strumenti di simulazione e modellazione sviluppati in ambiente MatLab/Simulink per analizzare il comportamento cinematico e dinamico dei controllori con asservimento visivo. Relativamente all'implementazione degli algoritmi di tracking, invece, vengono discusse le strutture software e l'architettura progettata per l'acquisizione, elaborazione e scambio dei dati tra sistema di visione e controllore. Anche per quanto concerne il controllore stesso, infine, si presentano alcune riflessioni sull'architettura software sviluppata.

### **PARTE 3: Validazione sperimentale**

Le applicazioni effettivamente realizzate utilizzano semplici sistemi ad un grado di libertà. La scelta è ricaduta su questo tipo di applicazioni perché permettono di effettuare un'analisi diretta degli effetti dovuti all'introduzione del sistema di visione nell'anello di controllo del sistema. Tale scelta ha comunque permesso il progetto e lo sviluppo delle componenti software fondamentali per affrontare i problemi di controllo con asservimento visivo. Con queste applicazioni si cerca di mettere in luce questi componenti in modo da evidenziare come essi possano essere facilmente estesi a casi molto più complessi.

- **Posizionamento di una Tavola Rotante.**

Questa applicazione fa riferimento al problema di posizionamento di una guida lineare o una tavola rotante in modo da esporre ad un operatore un determinato strumento o parte da lavorare. Quest'ultima definisce il set-point per il compito di posizionamento; la visione apporta il vantaggio di poter compiere il task senza avere movimenti predeterminati, ma solo tramite la scelta per mezzo di immagini del target.

- **Controllo di un Pendolo Inverso con Asservimento Visivo.**

Il problema di mantenere in equilibrio un pendolo inverso è un esercizio abbastanza diffuso per mettere in pratica le regole di controllo di un sistema instabile. Allo stesso modo, decidendo di leggere la posizione angolare del pendolo e la velocità con cui esso oscilla tramite un sistema di visione, permette di mettere in pratica le soluzioni al problema del tracciamento di oggetti in movimento e mettere successivamente in mostra le problematiche derivanti dall'introduzione di un sistema di visione nell'anello di retroazione di un controllore.

### **Conclusioni e Sviluppi Futuri**

Nell'ultimo capitolo vengono presentate le considerazioni finali sul lavoro svolto. Si evidenziano, infine, alcuni possibili sviluppi futuri delle componenti software per la simulazione e l'implementazione di sistemi di controllo con asservimento visivo implementati nel lavoro di tesi.

# Parte I

## Elaborazione delle immagini





# Capitolo 2

## Acquisizione delle Immagini

### Indice

---

1	Sequenza e componenti dell'acquisizione immagini . . . . .	<b>21</b>
1.1	Sensore d'acquisizione . . . . .	22
1.2	Ottiche . . . . .	25
1.3	Condizionamento del segnale . . . . .	26
1.4	Interfaccia di comunicazione . . . . .	29
1.5	Effetti dell'illuminazione . . . . .	32
2	Modello della telecamera . . . . .	<b>40</b>
2.1	Modello pin-hole . . . . .	40
2.2	Distanze focali . . . . .	43
2.3	Distorsioni . . . . .	44
2.4	Ricostruzione 3D . . . . .	48
3	Calibrazione delle telecamere . . . . .	<b>57</b>
3.1	Omografia . . . . .	58
3.2	Stima dell'omografia . . . . .	60
3.3	Pattern di calibrazione . . . . .	63
4	Conclusioni . . . . .	<b>64</b>

---

In figura 1.2 sono state introdotte le diverse componenti di un sistema di visione artificiale e la sua integrazione con il sistema di controllo di un qualsivoglia sistema meccanico. La componente principale da cui inizia lo sviluppo di ogni sistema di visione è sicuramente il *sistema di acquisizione immagini*. Spesso si tende ad includere in questa componente solo i dispositivi di visione in senso stretto, ovvero le telecamere. In realtà, si

dovrebbe parlare di un sistema più complesso, suddividibile a sua volta in più sottoparti, la cui analisi e conoscenza sono necessarie per ottenere l'immagine da elaborare:

- *Sensore d'acquisizione.* È il cuore di ogni dispositivo di visione. Le sue caratteristiche stabiliscono il formato, in termini di dimensione e colore, dell'immagine risultante;
- *Obiettivo.* Definisce la relazione tra dimensione dell'immagine e dimensione del campo inquadrato. Alcuni tipi di obiettivi permettono di regolare fisicamente alcune caratteristiche delle immagini, quali la luminosità;
- *Frame grabber.* Comprende l'elettronica e la logica da impiegare per il *condizionamento* del segnale, ovvero per convertire dei segnali di tensione in un'informazione visiva. Nel caso delle telecamere *digitali*, questo componente è integrato direttamente nel dispositivo di acquisizione, mentre nel caso delle telecamere *analogiche* esso è un dispositivo a parte. L'analisi di tale elemento risulta fondamentale in quanto esso è il principale responsabile del *campionamento* dell'immagine, sia in termini spaziali che in termini temporali;
- *Interfaccia di comunicazione.* Ne fanno parte tutte le componenti, sia hardware che software, che si riferiscono alla modalità di scambio delle informazioni tra la periferica e il componente dedicato alla memorizzazione delle immagini. Per quanto riguarda il lato software, i protocolli di comunicazione dei dispositivi di visione non solo specificano le modalità di acquisizione delle informazioni relative alle immagini, ma offrono anche la possibilità di regolare alcune proprietà delle immagini;
- *Ambiente e illuminazione.* Si tratta di un elemento esterno al dispositivo d'acquisizione vero e proprio. Il problema dell'illuminazione si riferisce sia alla possibile presenza di disturbi ambientali, sia all'analisi delle proprietà degli oggetti da inquadrare. In ambito industriale, si fa spesso uso, ove possibile, di dispositivi di illuminazione esterni per permettere di rendere costanti le condizioni ambientali e ridurre al minimo l'influenza dei disturbi sugli algoritmi di elaborazione delle immagini;
- *Centro di elaborazione delle immagini.* Alcuni dispositivi di visione (*smart camera*), non si limitano a fornire gli strumenti per l'acquisizione delle immagini, ma contengono al loro interno anche un'unità di calcolo cui può essere assegnato il primo stadio di elaborazione dell'immagine. In questo caso, l'interfaccia di comunicazione permette la trasmissione di informazioni più complesse che non siano le semplici immagini. Tuttavia, la configurazione tipica prevede l'assegnamento del compito di elaborazione ad un dispositivo dedicato e separato dalla telecamera.

La scelta di un dispositivo di visione rappresenta il primo e forse più importante passo nello sviluppo di un'applicazione di visione artificiale. La presenza di più tematiche di tipo diverso, tuttavia, rende complessa la scelta dello stesso e fa sorgere la necessità di una classificazione e analisi dettagliata dei processi che portano alla formazione delle immagini.

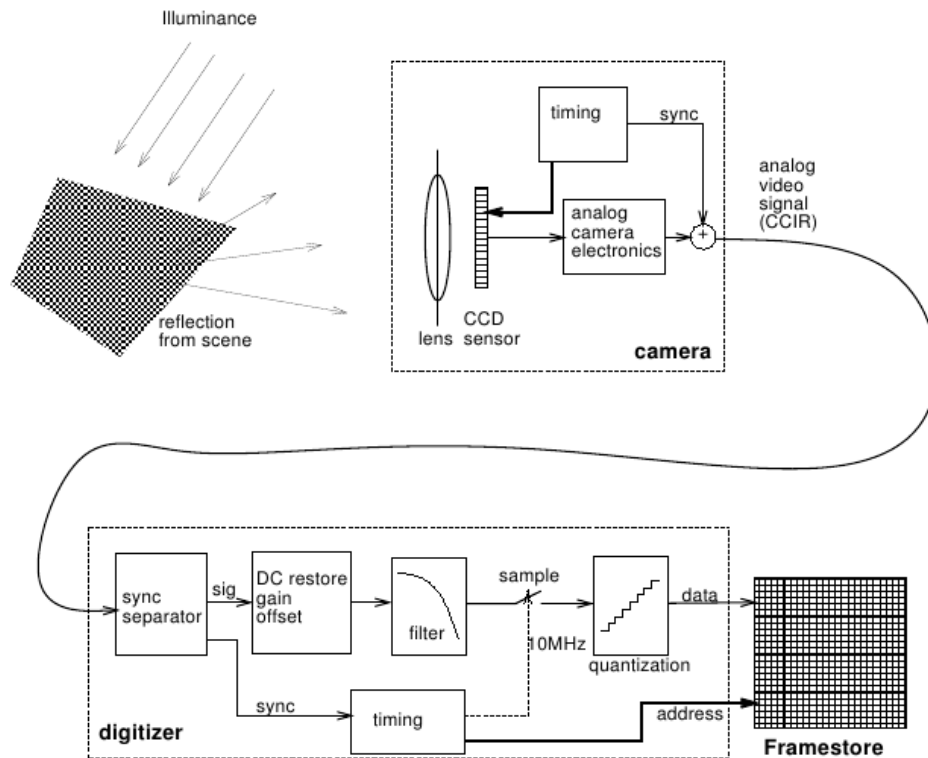
In questo capitolo ci si limita ad analizzare l'hardware di un sistema di visione, mentre il lato software, relativo all'effettiva elaborazione delle immagini verrà presentato nei capitoli 3, 4 e 6. Innanzitutto si presentano le nozioni base per la comprensione dei principi fisici alla base della formazione dell'immagine in funzione di fornire gli strumenti base per la scelta e l'analisi delle prestazioni dei dispositivi di visione (1). L'introduzione di alcuni concetti relativi alla formazione dell'immagine è qui necessaria per meglio comprendere come la scelta di un determinato tipo di telecamera possa influenzare non solo il sistema di acquisizione, ma anche la scelta del tipo di algoritmi di elaborazione delle immagini implementare. In secondo luogo, viene presentata una formulazione matematica del problema della formazione dell'immagine (2) e la soluzione al problema dell'identificazione del modello della telecamera (3). Infine, si traggono le conclusioni (4) relative all'analisi effettuata, al fine di mostrare come i concetti presentati forniscano le basi per la trattazione delle successive componenti del sistema di visione.

I dati e le informazioni sull'hardware dedicato alla visione trattato in questo capitolo sono stati reperiti principalmente da [site14] e [site15].

## 1 Sequenza e componenti dell'acquisizione immagini

In figura 2.1 vengono riassunte le componenti e la sequenza di operazioni necessarie per passare dall'inquadratura di una scena ad una sua rappresentazione sotto forma di immagini. In particolare, il processo rappresentato in figura si riferisce all'acquisizione di immagini da una telecamera *analogica*, ma può essere esteso facilmente al caso di telecamere *digitali* in cui i passi eseguiti per digitalizzare l'immagine sono insiti nel tipo di tecnologia utilizzata nella realizzazione del sensore d'acquisizione.

L'acquisizione avviene per fasi successive: (i) memorizzazione del valore di intensità luminosa all'interno di ogni singolo pixel e (ii) digitalizzazione dei segnali al fine di ottenere (iii) una rappresentazione matriciale della scena inquadrata.

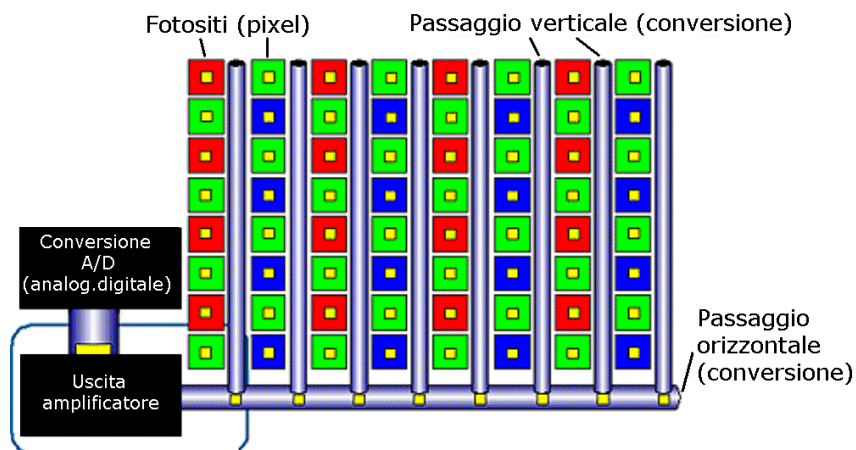


**Figura 2.1:** Componenti ed operazioni coinvolte nella formazione dell'immagine

## 1.1 Sensore d'acquisizione

Il *sensore d'acquisizione* è, insieme all'ottica, il componente fondamentale di una telecamera. Il compito svolto da questo elemento è essenzialmente il *campionamento* della scena inquadrata dal dispositivo di visione.

Il sensore (figura 2.2) è composto da un numero variabile di elementi fotosensibili (*pixel*) in grado di trasformare l'energia assorbita sotto forma di luce in energia elettrica.



**Figura 2.2:** Struttura di un sensore d'acquisizione immagini

I pixel sono distribuiti sull'intera superficie del sensore in modo da formare una matrice

di  $n$  righe e  $m$  colonne. La dimensione della matrice fornisce la *risoluzione* dell'immagine. Questa distribuzione dei singoli elementi in una struttura matriciale permette di effettuare un *campionamento spaziale* della scena, la quale viene quindi scomposta in un numero finito di punti.

Una prima classificazione dei sensori d'acquisizione è data dalla modalità attraverso cui la matrice viene costruita: le *telecamere lineari* utilizzano una singola linea di pixel che permette di costruire l'immagine di oggetti in movimento nel tempo, mentre le *telecamere matriciali* permettono di ottenere direttamente un'immagine bidimensionale.

Un'altra differenza consiste nella tecnologia utilizzata nella realizzazione del sensore. In particolare, si parla di sensori *CCD* o *CMOS*. Nei sensori CCD i pixel sono costituiti da semiconduttori in grado di accumulare una carica elettrica proporzionale al numero di elettroni liberati una volta colpiti dalla radiazione luminosa. Poiché l'intensità luminosa è misurata attraverso una carica, ne risulta che la misura è data dal calcolo dell'integrale dell'intensità di luce in un certo intervallo di tempo (*tempo d'esposizione*). Al termine del periodo d'esposizione, il valore della carica è passato ad un amplificatore e il pixel viene scaricato. Nel caso della tecnologia CMOS, i semiconduttori vengono sostituiti da fotodiodi. La giunzione di ogni diodo è precaricata, per essere progressivamente scaricata una volta colpiti dai fotoni. Ancora una volta, un amplificatore si occupa di convertire questo valore di carica in un livello di corrente o tensione.

Entrambi i tipi di sensori possono soffrire di *blurring*, ovvero quel fenomeno per cui, a causa di distorsioni dei pixel, gli oggetti di una scena tendono ad allungarsi, soprattutto se sottoposti ad un movimento veloce. Nel caso dei sensori CCD questo effetto è provocato dalla misura integrale della luminosità, a causa della quale la misura effettuata da un singolo pixel influenza la misura dei suoi vicini all'interno della matrice. Nel caso del sensore CMOS, al contrario, le distorsioni vengono provocate dal fatto che pixel ai lati opposti del sensore possono avere tempi d'esposizione differenti. Queste due caratteristiche portano ad evidenziare la differenza principale nell'acquisizione di immagini da parte dei due sensori: i CCD permettono di ottenere immagini più uniformi, ma hanno tempi d'adattamento alle variazioni di luminosità più lunghi; al contrario, i CMOS reagiscono più rapidamente alle variazioni di luminosità, ma le immagini risultano spazialmente non uniformi.

Il blurring (figura 2.3) è uno degli effetti con i quali bisogna scontrarsi quando si riprendono scene in movimento.

Nelle fotocamere digitali per uso non industriale, la messa a fuoco degli oggetti in movimento viene effettuata meccanicamente attraverso l'*otturatore* (*shutter*), un elemento



**Figura 2.3:** Fenomeno del “blurring”

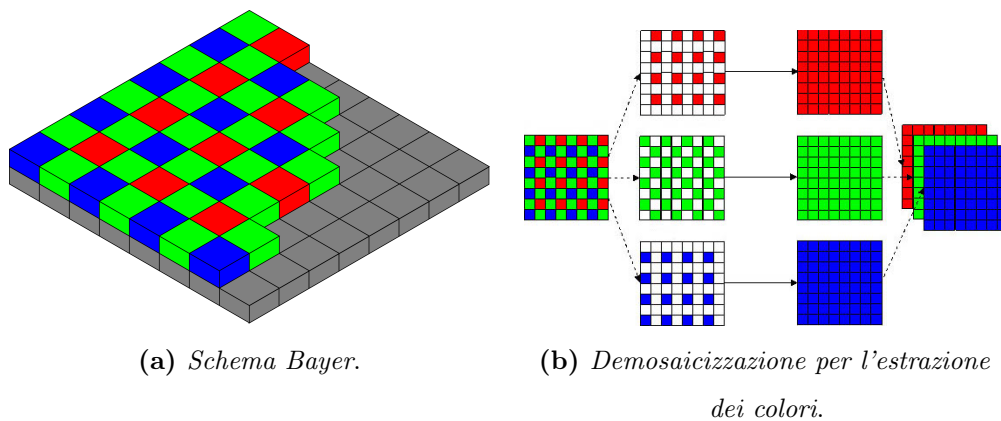
attraverso cui è possibile ridurre il tempo d’esposizione ad un periodo inferiore alla durata del movimento stesso. I sensori CCD permettono di simulare il comportamento dell’otturatore mantenendo i pixel scarichi fino ad un istante di tempo prima dell’inizio del periodo d’esposizione desiderato. Questa tecnica, tuttavia, produce una diminuzione del rapporto segnale-rumore poiché la carica accumulata diminuisce con l’avvicinarsi al termine del periodo d’esposizione. È pertanto necessario, nella procedura di taratura del parametro di shutter, lavorare anche con i parametri che influenzano la *luminosità* dell’immagine, ovvero col guadagno della parte di amplificazione o con l’apertura dell’obiettivo della telecamera, qualora quest’ultimo ne permetta la taratura meccanica. L’otturatore permette di *campionare temporalmente* la scena, nel senso che il valore di luminosità catturato dai pixel è relativo all’istante in cui essi intercettano la radiazione luminosa.

Infine, è possibile distinguere tra telecamere *analogiche* e *digitali*: nel caso delle prime, la misura effettuata dai pixel deve essere ulteriormente elaborata per ottenere un flusso video valido, mentre per le seconde, il segnale in uscita dall’amplificatore rappresenta il valore di intensità luminosa dell’immagine.

### 1.1.1 Rappresentazione dei colori

Come si è visto, l’elemento che costituisce il pixel fornisce una misura della quantità di energia luminosa che incide il sensore. Con una matrice di pixel puri è possibile realizzare una semplice telecamera in scala di grigi, ovvero in grado di misurare solamente la luminosità della scena inquadrata. La resa dei colori viene solitamente realizzata scomponendo la luce nelle sue componenti fondamentali e misurandone la luminosità. La ricostruzione dell’immagine a colori prevede l’aggiunta di una fase di filtraggio da parte del componente dedicato alla digitalizzazione dei segnali.

Lo *schema Bayer*, mostrato in figura 2.4, è il tipico schema per la disposizione degli elementi sensibili ai diversi colori nei sensori usati per l'acquisizione di immagini digitali. La sua caratteristica è quella di raggruppare i sensori per i tre colori fondamentali necessari per la sintesi additiva (RGB, rosso, verde e blu) in celle di due fotositi per due. Ogni cella contiene due elementi verdi, uno rosso e uno blu. Lo schema Bayer prevede che nelle otto cellule adiacenti ad ogni fotosito, ve ne siano almeno due di ognuno degli altri colori. Quindi rende possibile ricostruire il valore della luminosità, ad esempio, del rosso in corrispondenza di un elemento verde o blu, deducendolo dagli elementi rossi circostanti. Il processo, chiamato *demosaicizzazione*, costituisce il filtraggio aggiuntivo per ottenere l'immagine a colori.



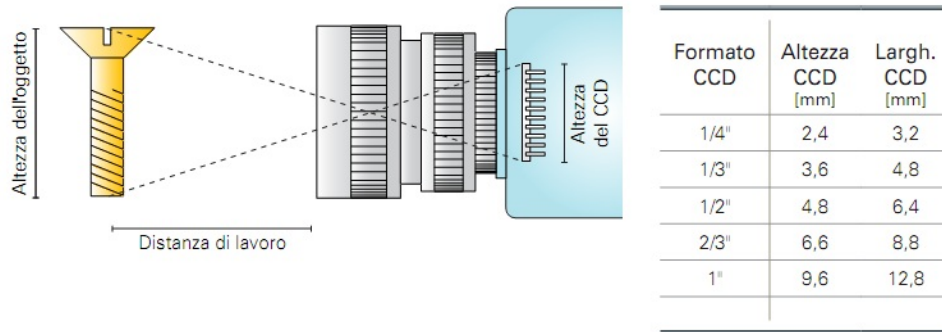
**Figura 2.4:** Schema Bayer di rappresentazione dei colori

## 1.2 Ottiche

L'altro elemento fondamentale per comprendere come la scena inquadrata da una telecamera venga resa sul piano immagine è l'*obbiettivo*. Il suo scopo è quello di indirizzare la luce verso il sensore del dispositivo d'acquisizione. La caratteristica che permette di distinguere un obbiettivo dall'altro è la *lunghezza focale*. Considerando l'obbiettivo come una semplice lente, la distanza focale di questi è la misura espressa in millimetri della distanza che separa la lente dal piano focale. Essendo gli obbiettivi composti da più gruppi di lenti, tale distanza non si misura da una lente in particolare all'interno degli stessi ma dal centro ottico dell'obbiettivo che viene definito *punto nodale posteriore* e in genere si trova in prossimità del diaframma. In sostanza, la distanza focale indica la distanza fra il punto nodale posteriore di un obbiettivo e il piano su cui i soggetti all'infinito sono messi a fuoco. La scelta della distanza focale permette quindi di regolare la dimensione dell'area

inquadrata dalla telecamera, mentre il diaframma è l'elemento meccanico che permette di regolare la quantità di luce che attraversa la lente dell'obiettivo e raggiunge il sensore di visione.

L'accoppiamento del corretto obiettivo al sensore scelto (figura 2.5) è il passo che permette di definire la scena inquadrata sia qualitativamente che dal punto di vista delle dimensioni dell'area inquadrata.



**Figura 2.5:** Accoppiamento tra sensore d'acquisizione immagini e obiettivo

Dati come parametri di progetto la distanza operativa  $D_O$  tra telecamera e oggetti inquadrati e la dimensione dell'area da inquadrare, ovvero il *field of view* ( $L_{FOV}$ ,  $H_{FOV}$ ), è possibile calcolare la distanza focale necessaria. Come si vede dalle equazioni (2.1), anche la dimensione fisica del sensore ( $L_{sensore}$ ,  $H_{sensore}$ ) influisce sulla scelta dell'obiettivo.

$$\begin{cases} F_L = \frac{(D_O) \cdot (L_{sensore})}{(L_{FOV}) + (L_{sensore})} \\ F_H = \frac{(D_O) \cdot (H_{sensore})}{(H_{FOV}) + (H_{sensore})} \end{cases} \quad (2.1)$$

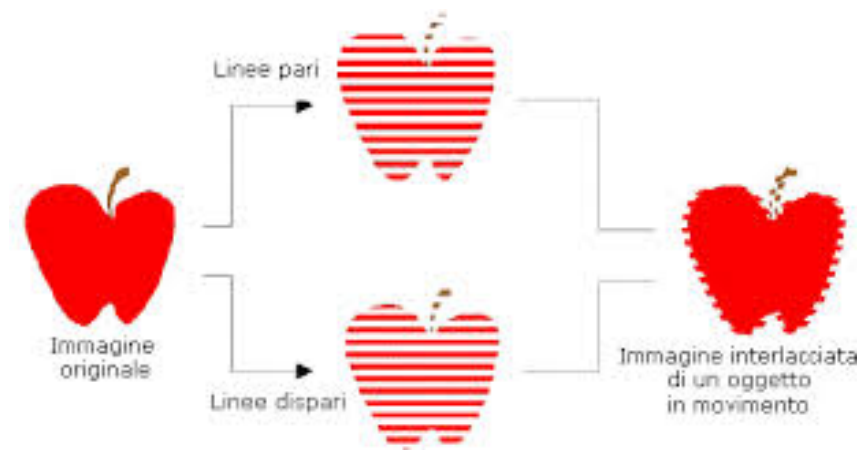
Nel paragrafo 2 di questo capitolo verrà mostrato come inserire la distanza focale dell'obiettivo in un modello matematico per la modellazione della trasformazione prospettica che permette di calcolare le coordinate immagine di qualsiasi punto della scena a partire dalle sue coordinate spaziali.

### 1.3 Condizionamento del segnale

Nel caso di un sensore CCD di una telecamera monocromatica, lo stadio di amplificazione genera un segnale dalla cui elaborazione, successivamente, nasce il segnale video vero e proprio. L'intera immagine viene suddivisa in un numero di linee che devono essere elaborate sequenzialmente. L'elaborazione procede linea per linea dall'alto verso il basso



considerando prima tutte le linee dispari e in seguito le linee pari. L'immagine completa viene infine ricostruita come composizione di due mezze immagini successive. Questa tecnica è nota col nome di *interlacciamento* (figura 2.6) ed essa permette di aggiornare l'immagine alla frequenza di acquisizione della telecamera oppure a metà della frequenza. Conoscere la frequenza d'acquisizione e di aggiornamento delle immagini è fondamentale per stimare l'effettiva frequenza di lavoro del dispositivo di visione e, quindi, stimare la massima velocità di movimento della scena ammessa per limitare o evitare l'insorgenza del fenomeno del blurring.



**Figura 2.6:** *Interlacciamento*

Il segnale ottenuto tramite interlacciamento è di tipo analogico, mentre gli algoritmi di visione vengono generalmente applicati ad immagini digitali. I dispositivi d'acquisizione più utilizzati integrano delle componenti elettroniche il cui scopo è la conversione del segnale analogico in un'immagine digitale. Questi componenti operano quindi come convertitori A/D e vengono chiamati *frame grabber*. In particolare, il frame grabber esegue il *campionamento* e la *quantizzazione* del segnale analogico e il risultato che si ottiene è la memorizzazione dei valori misurati dalla matrice di pixel in un array di dati. La presenza del frame grabber è necessaria per i sensori CCD, mentre nel caso della tecnologia CMOS il convertitore A/D può essere realizzato a livello di ogni singolo pixel. Per questo motivo, un sensore CMOS permette l'accesso casuale ad un sottoinsieme specifico di pixel, garantendo quindi un aumento della frequenza di aggiornamento delle immagini rispetto ad un analogo sensore CCD.

Un'immagine digitale è una rappresentazione, campionata nello spazio, della funzione continua  $I(u, v)$  che definisce l'intensità della luce incidente il pixel di coordinate  $(u, v)$  del piano immagine. La funzione  $I(u, v)$  è chiamata generalmente *livello di grigio* perché

senza ulteriore elaborazione una matrice di pixel fornisce immagini a livello di grigio della scena inquadrata. In figura 2.1, l'ultima parte del processo d'acquisizione è costituita dalla *digitalizzazione* del segnale, che può essere scomposta nelle seguenti sottofasi:

1. *Eliminazione della componente continua.* Durante la fase di amplificazione, il segnale potrebbe avere accumulato una componente continua. Questo offset deve essere eliminato perché altrimenti potrebbe risulterne uno spostamento dei valori misurati verso il basso o l'alto;
2. *Condizionamento del segnale.* Prima di essere digitalizzato, il segnale viene filtrato per ridurre gli effetti di *aliasing*. Il filtraggio ha però alcuni effetti negativi sulla qualità delle immagini prodotte, come la riduzione del contrasto e l'introduzione di un'ulteriore ritardo nella formazione dell'immagine dovuto alla dinamica del filtro introdotto;
3. *Campionamento.* Nel caso ideale, le porzioni di segnale di ogni elemento del sensore dovrebbero essere costanti e determinate. Tale corrispondenza non è riscontrata nel caso reale, ma risulta invece distorta. Il campionamento del segnale permette di estrarre l'informazione  $I(r, c)$  dal singolo elemento fotosensibile del sensore ed associarla all'informazione  $I(u, v)$  del piano immagine risultante. Per portare a termine questa operazione, viene utilizzato un segnale di sincronizzazione. La frequenza di acquisizione delle immagini  $f_d$  è generalmente un sottomultiplo della frequenza di sincronizzazione  $f_s$ . Infine, non è detto che sussista una corrispondenza uno-a-uno tra la risoluzione del sensore e la risoluzione delle immagini risultanti dal filtraggio. Il parametro che mette in relazione le due quantità è il *rapporto di campionamento*  $\beta$  definito come

$$\beta = \frac{\# \text{ elementi fotosensibili per linea}}{\# \text{ pixel per linea}} . \quad (2.2)$$

4. *Quantizzazione.* L'ultima fase del processo di formazione dell'immagine è la quantizzazione del segnale analogico di ogni singolo pixel in un numero finito di intervalli. Ogni campione è quantizzato in un intero a  $n$  bit, in modo da avere un range di  $2^n$  valori ammissibili. Solitamente allo 0 viene fatto corrispondere il livello del nero, mentre il bianco viene associato al livello più alto. Tipicamente  $n = 8$  perché 256 livelli di grigio permettono di ottenere una buona descrizione della scena per la maggior parte delle applicazioni e, inoltre, l'immagine può essere rappresentata come un vettore di Byte. L'equazione (2.3) permette di esprimere il segnale quantizzato

$x_q(t)$  come funzione del segnale originale  $x(t)$  e del rumore di quantizzazione  $e_q(t)$ , modellabile come una distribuzione uniforme nell'intervallo  $[-\frac{1}{2}, \frac{1}{2}]$ .

$$x_q(t) = x(t) + e_q(t) . \quad (2.3)$$

## 1.4 Interfaccia di comunicazione

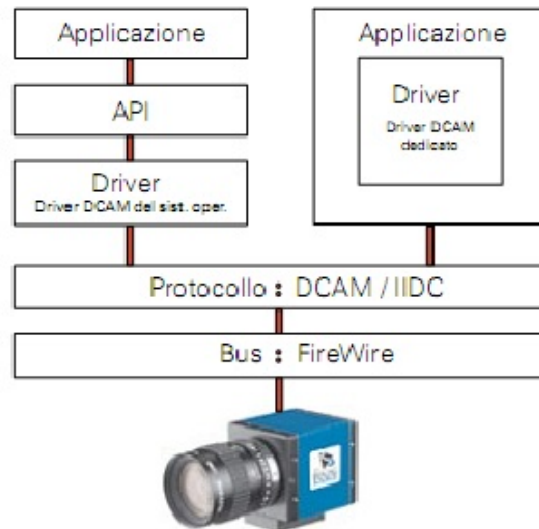
Nei paragrafi precedenti sono stati analizzati i singoli passi del processo di formazione delle immagini. Una volta ottenuta l'immagine digitale, tuttavia, sorge la necessità di estrarre le informazioni dal dispositivo per proseguire con l'elaborazione dell'immagine stessa. La fase di comunicazione tra dispositivo di acquisizione e il componente dedicato all'elaborazione vera e propria dell'immagine introduce un'ulteriore ritardo nella propagazione dell'immagine e risulta quindi necessario tenere conto delle specifiche del canale di comunicazione per comprendere a quale istante temporale si riferiscono le informazioni estratte.

Le telecamere presenti sul mercato mettono a disposizione diverse tipologie di interfacce di comunicazione, tra cui le più utilizzate sono:

- *IEEE1394 (firewire;*
- *Camera Link;*
- *USB;*
- *Gigabit Ethernet.*

### 1.4.1 Firewire

Il *firewire* è un'interfaccia standard per la comunicazione di dati tra dispositivi connessi su un bus seriale. Supporta due modalità di trasferimento dati: *sincrona* e *asincrona*. Quest'ultima modalità viene utilizzata quando il dato spedito viene ricevuto dall'altra parte del cavo solo su richiesta e una tantum. La ricezione è garantita perché è previsto il reinvio dell'informazione nel caso il canale di comunicazione risultasse occupato. Nelle telecamere questa tipologia di comunicazione viene utilizzata per la cosiddetta modalità *one-shot* in cui il frame viene inviato solo su richiesta. La modalità sincrona, al contrario, prevede un invio di dati attraverso un flusso continuo in *tempo reale*. Questa seconda modalità di comunicazione è quella tipicamente utilizzata nelle telecamere in cui, dopo un opportuno avvio, l'acquisizione delle immagini avviene in continuo. In figura 2.7 è riportato un esempio di collegamento di più periferiche al bus firewire e se ne evidenzia lo stack da implementare per l'attivazione della comunicazione.



**Figura 2.7:** Bus di comunicazione firewire

Il bus firewire supporta il collegamento di fino a 63 periferiche, distinguibili dall'indirizzo assegnato loro al momento dell'inserimento sul bus, esattamente come avviene per una rete LAN. La trasmissione di dati in parallelo è possibile qualora il flusso totale di dati non saturi la banda complessiva offerta dal canale di comunicazione. Il collegamento *plug and play* è supportato e anche l'alimentazione dei dispositivi è spesso fornita dal bus stesso.

Esistono due diverse specifiche del protocollo: *Firewire-A* e *Firewire-B*. Lo standard B è retrocompatibile con il primo poiché l'unica differenza tra i due risiede proprio nella banda passante. Il firewire A permette la trasmissione di dati fino a 400 Mb/s, mentre con la seconda versione del protocollo la banda viene raddoppiata fino a 800 Mb/s. I principali vantaggi del firewire non risiedono però nell'ampiezza di banda raggiungibile. Un primo vantaggio risiede ma nella definizione di un protocollo di comunicazione universale, noto come *1394-based Digital Camera Specification* (DCAM) che permette di gestire non solo l'acquisizione delle immagini, ma anche l'impostazione dei parametri della telecamera. Esistono implementazioni di questo standard, sotto forma di libreria per i più diffusi sistemi operativi ([site4]).

Il secondo vantaggio è dato dalla possibilità di acquisire immagini in tempo reale. Infatti, sempre rimanendo al di sotto dell'occupazione di banda ammessa, le richieste di immagini ai dispositivi ricevono una risposta in un tempo determinato e, pertanto, stimabile a priori; qualora questo limite temporale non possa essere rispettato, si genera un errore. Questa caratteristica del firewire risulta rilevante quando l'acquisizione di immagini si

mette al servizio di un sistema che necessita l'estrazione in tempo reale di informazioni dalle immagini.

Nel caso trattato in questo lavoro di tesi, ovvero lo sviluppo di controllori per sistemi meccanici complessi facendo uso di informazioni visive, si vedrà nel capitolo 7 come sia necessario inserire anche il tempo di formazione e trasmissione dell'immagine nello schema del controllore.

### 1.4.2 Camera Link

Anche l'interfaccia *Camera Link* descrive uno standard di comunicazione seriale. Tuttavia, esso nasce con la specifica esigenza di standardizzare l'interfaccia di comunicazione per dispositivi di visione ad elevato livello tecnologico. Pertanto, caratteristica principale dello standard è quella di permettere la trasmissione di una grande quantità di dati. La banda passante è infatti di ben 2 Gb/s. Ad ogni modo, l'elevata banda passante richiede anche un'elevata capacità computazionale del componente dedicato all'acquisizione ed elaborazione delle immagini e dell'elettronica dedicata al campionamento dei segnali generati dal sensore d'acquisizione; di conseguenza, l'interfaccia Camera Link viene utilizzata per aumentare la risoluzione delle immagini piuttosto che il framerate d'acquisizione.

Il raggiungimento di una banda passante così elevata è possibile perché il tipo d'informazione che passa sul canale di comunicazione non è una corrispondenza uno-a-uno dell'immagine acquisita, ma è il risultato di una permutazione effettuata sui valori assunti dai singoli pixel. Le periferiche Camera Link includono una componente hardware dedicata alla codifica e decodifica dell'informazione. Generalmente, infatti, l'informazione non viene codificata o compressa poiché senza la presenza di un hardware opportuno, le operazioni di estrazione dell'informazione andrebbero ad aggiungersi sequenzialmente alle operazioni di acquisizione dell'immagine, provocando un aumento del tempo d'acquisizione stesso. Infine, come nel caso del firewire, la ripetitività dello scambio delle informazioni è garantita dall'invio di un segnale di sincronizzazione della linea e dei vari dispositivi.

### 1.4.3 Universal Serial Bus

Lo standard *USB* è uno dei più utilizzati anche per il collegamento dei dispositivi d'acquisizione immagine industriali. La grande diffusione è dovuta soprattutto alla semplicità d'utilizzo, dato dal supporto per il collegamento a caldo e dalla possibilità di alimentare i dispositivi direttamente col bus. A differenza del firewire, il sistema USB è asimmetrico,

ovvero necessita di un dispositivo che faccia da master. Nel caso della visione industriale, il master è costituito dal calcolatore incaricato dell'acquisizione delle immagini, mentre le telecamere fungono da dispositivo slave. La banda passante del canale di comunicazione è di 480 *Mb/s* per lo standard USB 2.0, fino ad arrivare a 4.8 *Gb/s* con la versione 3.0. La possibilità di ottenere bande passanti così elevate porta alla completa eliminazione dei ritardi di trasmissione delle immagini. Tuttavia, le telecamere con interfaccia di comunicazione basata su USB soffrono della mancanza di un protocollo di comunicazione unico. Il protocollo e, di conseguenza, i driver di controllo dei dispositivi, sono proprietari, non rendendo così possibile lo sviluppo di applicazioni indipendenti dall'hardware scelto. Un ulteriore svantaggio è dato dalla mancanza di meccanismi di sincronizzazione real-time, pertanto il tempo di risposta ad una richiesta di dati potrebbe essere piccolo a piacere (nel caso dell'USB 3.0), ma in realtà non determinabile o stimabile a priori.

#### 1.4.4 Gigabit Ethernet

Un'altra interfaccia di comunicazione particolarmente apprezzata e utilizzata per applicazioni che necessitano la trasmissione di un'elevata quantità di dati è fornita dallo standard *Gigabit Ethernet* (*GigE*). Il principale vantaggio nell'utilizzo dello standard Ethernet consiste nella facilità di creazione ed estensione di reti di dispositivi facilmente identificabili attraverso il loro indirizzo univoco. Inoltre, anche il Gigabit Ethernet è in grado di fornire l'alimentazione ai dispositivi di visione (*Power Over GigE*). Il principale svantaggio è lo stesso dello standard USB, ovvero la mancanza di un protocollo di comunicazione standard, anche se tale svantaggio è in parte mitigato dalla possibilità di non dover utilizzare driver aggiuntivi per instaurare la comunicazione tra un dispositivo e l'altro.

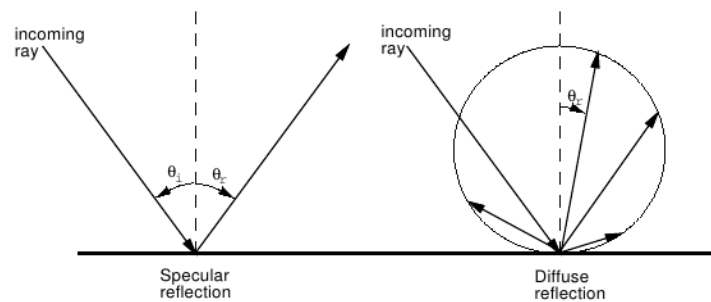
### 1.5 Effetti dell'illuminazione

Una trattazione rigorosa delle leggi fisiche che regolano le modalità con cui le superfici riflettono la luce e, conseguentemente, della misura con cui questa arriva al sensore del dispositivo di visione, è complessa e non è uno degli obiettivi di questo lavoro di tesi. Pertanto, di seguito ci si limita a fornire una classificazione delle diverse tipologie di illuminazione cui può essere sottoposta una scena e a mostrare alcuni esempi di come le condizioni di luminosità influenzano il modo in cui una scena viene rappresentata.

### 1.5.1 Tipologie di illuminazione

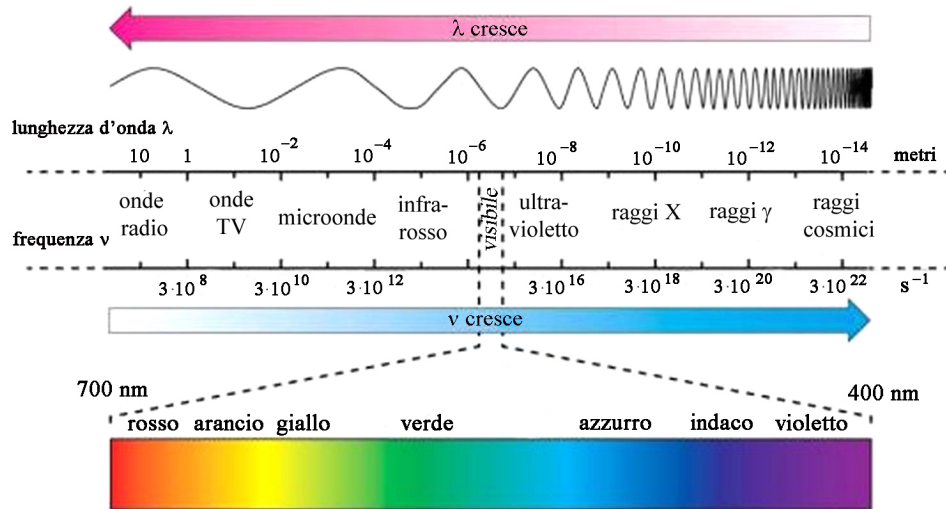
Per caratterizzare una tipologia di illuminazione, si devono analizzare le seguenti proprietà:

- *Direzione.* La direzione della luce dipende dalla posizione delle fonti di luce. Nel caso di un'unica fonte di luce si ottiene la formazione di un'ombra netta, mentre l'utilizzo di più punti luce permette tipicamente di eliminare le ombre dalle immagini. Solitamente viene considerata la direzione dei raggi emessi dalla fonte luminosa e la direzione dei raggi riflessi dalla superficie incidente. Si possono individuare due tipi principali di riflessione: speculare e diffusa (figura 2.8). Nel caso della riflessione speculare l'angolo di riflessione è lo stesso dell'angolo di incidenza, mentre nel caso della riflessione diffusa l'angolo di riflessione è variabile e dipende dal punto in cui la luce incide l'oggetto;



*Figura 2.8: Riflessione della luce*

- *Lunghezza d'onda.* Identifica la posizione di una radiazione elettromagnetica sullo spettro della radiazione luminosa (figura 2.9). Nel caso di illuminazione naturale si lavora nella banda della luce visibile che va dalle frequenze del rosso al viola. A lunghezze d'onda maggiori si parla di radiazione infrarossa, mentre dall'altro lato si sfiora nel campo degli ultravioletti;
- *Polarizzazione.* Per un'onda elettromagnetica, indica la direzione dell'oscillazione del vettore campo elettrico durante la propagazione dell'onda nello spazio-tempo. Nel campo della visione artificiale, la polarizzazione viene sfruttata attraverso l'utilizzo di filtri polarizzatori, in modo tale da discriminare alcune radiazioni luminose d'interesse. caso tipico è l'utilizzo per l'individuazione di laser, i quali sono infatti fonti di luce a lunghezza d'onda fissa e determinata;
- *Intensità.* È proporzionale alla potenza o quantità d'energia emessa dalla fonte di luce per ogni unità di angolo solido. Nel caso dell'illuminazione a led, ad esempio, si ottiene

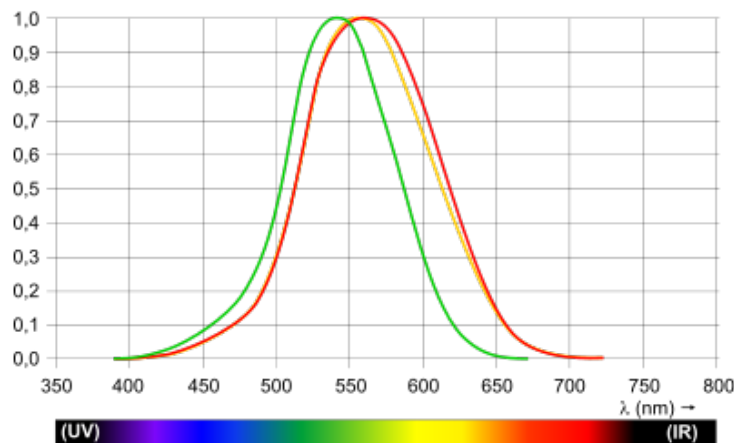


**Figura 2.9:** Spettro delle radiazioni elettromagnetiche

un'intensità maggiore semplicemente aumentando la tensione d'alimentazione dei led fino ad arrivare alla tensione di rottura dei diodi. L'unità di misura dell'intensità luminosa  $I_v$  è il candela ( $cd$ ) e matematicamente è data dalla somma integrale dell'intensità luminosa generata dalle radiazioni di ogni lunghezza d'onda

$$I_v = 683 \cdot \int_0^\infty \bar{y}(\lambda) \cdot \frac{dI_e(\lambda)}{d\lambda} d\lambda, \quad (2.4)$$

dove  $I_e$  è l'intensità radiante in watt per steradiano ( $W/sr$ ) e  $\bar{y}(\lambda)$  è la funzione di luminosità standard, la quale fornisce la componente di intensità in funzione della lunghezza d'onda (in figura 2.10 è riportata a titolo d'esempio la funzione d'intensità per radiazioni nelle frequenze di luce visibile del verde e del rosso).



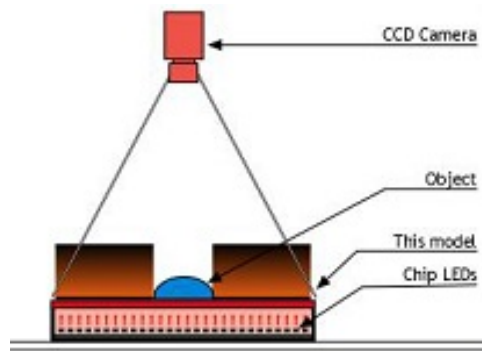
**Figura 2.10:** Funzione di luminosità standard



- *Uniformità*. Descrive la distribuzione della luce all'interno della scena. Più la luce è uniforme, minore è la presenza di parti oscure all'interno dell'immagine.

Una prassi tipica della visione industriale prevede l'utilizzo di illuminatori per fare in modo che la scena inquadrata abbia l'illuminazione desiderata. In funzione della posizione dell'illuminatore, si possono distinguere due principali tipi di illuminazione: *frontale* o *retro-illuminazione*.

Il secondo tipo è uno dei più utilizzati per la realizzazione di aree di prelievo oggetti in cui l'individuazione degli oggetti è possibile a partire semplicemente dalla loro silhouette. L'illuminatore è costituito tipicamente da un pannello di led posto dietro all'oggetto, come mostrato in figura 2.11.



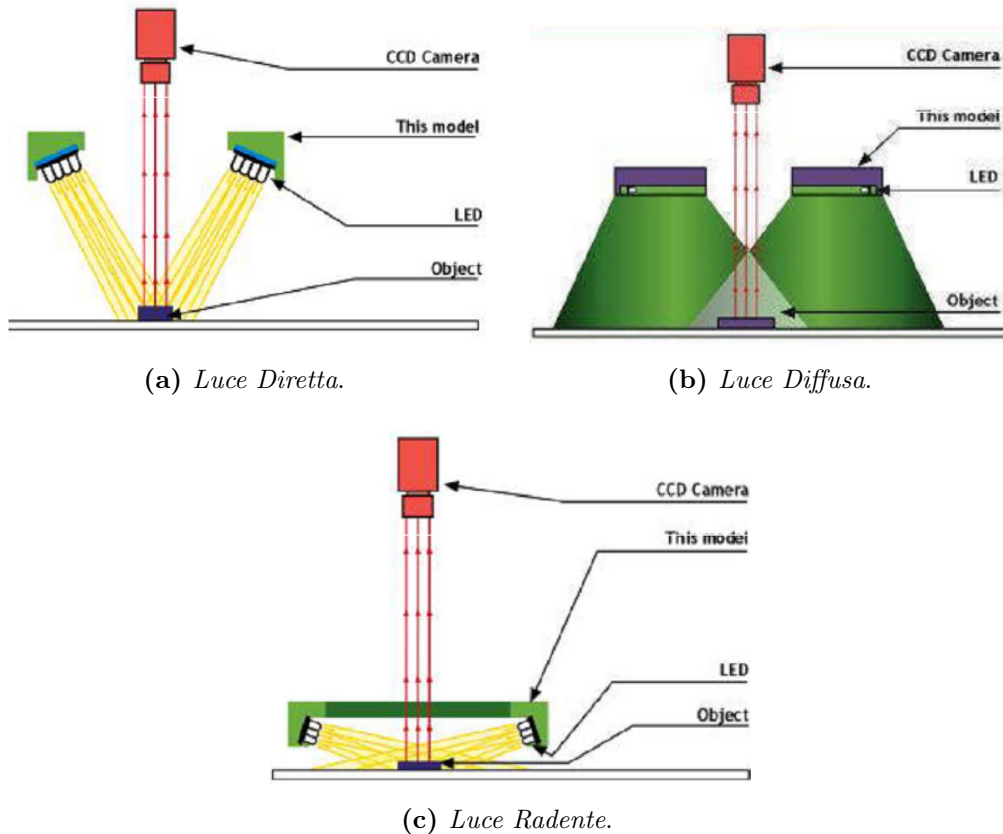
**Figura 2.11:** Retroilluminazione

Per quanto riguarda l'illuminazione frontale, invece, esistono diversi tipi di illuminatori in funzione di quale caratteristica dell'oggetto voglia essere messa in evidenza. In figura 2.12 si riportano alcuni esempi di illuminazione frontale. L'utilizzo delle luci diretta e diffusa permette di evidenziare i colori e le proprietà delle superfici della scena rispettivamente per un'area ridotta o più estesa. Al contrario, gli illuminatori a luce radente annullano le proprietà delle superfici per evidenziare i contorni degli oggetti.

Infine, per alcune applicazioni vengono utilizzati illuminatori laser (a linea o meno) per permettere la ricostruzione dell'informazione 3D della scena osservata.

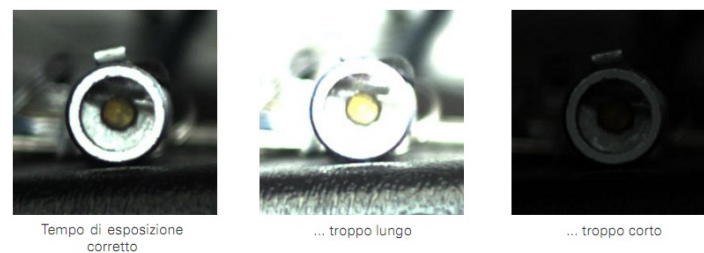
### 1.5.2 Parametri per la modifica dell'aspetto delle immagini

Gli esempi relativi a come l'illuminazione di una scena modifica il modo in cui essa viene rappresentata su un'immagine vengono elencati insieme al parametro dell'immagine da modificare per ottenere un particolare effetto desiderato. I parametri elencati vanno a modificare la configurazione degli elementi di filtraggio ed elaborazione del segnale utilizzati per il campionamento e digitalizzazione dell'immagine.



**Figura 2.12:** Illuminazione frontale

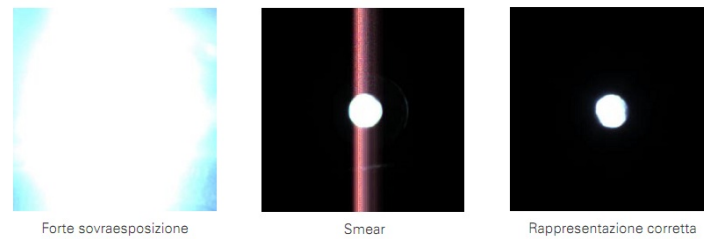
**Shutter.** Come si è visto (paragrafo 1.1), lo shutter determina il tempo di esposizione del CCD. Le prime tre immagini in figura 2.13 mostrano, a titolo d'esempio, un led ripreso con diversi tempi di esposizione: corretto, troppo corto e troppo lungo.



**Figura 2.13:** Shutter - esempio n. 1

In figura 2.14 si mostrano gli effetti dell'accensione del led: l'immagine ripresa con l'otturatore configurato come in precedenza è talmente sovraesposta che si vede soltanto una grande macchia bianca.

Riducendo il tempo di esposizione, l'immagine torna nitida. Nonostante la correzione dell'esposizione, una striscia verticale disturba ancora l'immagine. Questo effetto, tipico dei CCD, viene detto *smearing* ed è uno degli effetti indesiderati dovuto alla misura integrale



**Figura 2.14:** *Shutter - esempio n. 2*

della luminosità. Chiudendo il diaframma e allungando il tempo di esposizione è possibile ridurre o eliminare questo effetto.

**Guadagno e Contrasto.** Col termine *guadagno* ci si riferisce al guadagno del circuito di amplificazione del segnale proveniente dal sensore CCD. L'effetto di modifiche al valore del guadagno è una variazione della luminosità generale dell'immagine. Tuttavia, all'aumentare del guadagno si ha una riduzione del rapporto segnale-rumore (figura 2.15).



**Figura 2.15:** *Effetti del guadagno*

**Luminosità.** Col parametro di luminosità si aggiunge una componente costante al segnale in uscita dal CCD. Questa componente agisce come un offset su tutti i livelli di grigio dell'immagine (figura 2.16).



**Figura 2.16:** *Variazione di luminosità*

**Esposizione.** L'esposizione regola il tempo d'esposizione del sensore alla luce. In genere è possibile configurare una regolazione automatica del tempo d'esposizione. In questo

caso è il dispositivo stesso che esamina la luminosità media delle immagini e cerca di portare l'immagine ad avere tale valore pari al valore medio del range di luminosità consentito. Tuttavia il tempo d'esposizione va ad aggiungersi direttamente al tempo di aggiornamento delle immagini, pertanto nello sviluppo di un'applicazione industriale è importante mantenere questo tempo il più costante possibile.

**Nitidezza.** La nitidezza è collegata alla messa a fuoco delle immagini. In figura 2.17 vengono mostrate le differenze tra un'immagine sfocata e un'immagine troppo nitida.



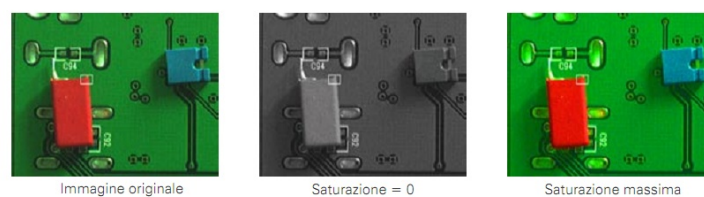
*Figura 2.17: Nitidezza e messa a fuoco*

**Gamma.** La *gamma* (figura 2.18) produce una variazione dei livelli di grigio medi e viene modificata per correggere le non-linearità e/o ridurre il rumore nelle immagini.



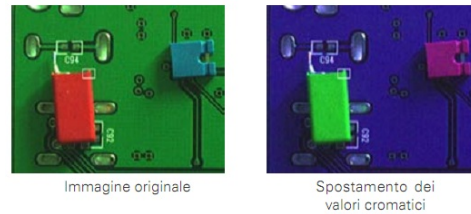
*Figura 2.18: Correzione gamma*

**Saturazione.** La saturazione (figura 2.19) indica il grado di purezza dei colori dell'immagine. Le immagini in scala di grigio hanno il minor grado di saturazione possibile.



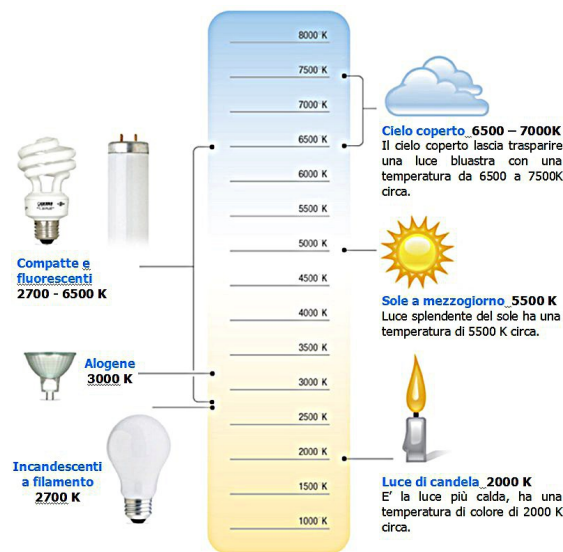
*Figura 2.19: Saturazione dei colori*

**Tonalità.** Questo parametro permette di spostare il piano colore dell'immagine su una gamma cromatica differente, mantenendo invariata la relazione tra i colori. Un esempio in figura 2.20.



*Figura 2.20: Tonalità dei colori*

La tonalità viene solitamente misurata dalla temperatura colore, definita come la temperatura che dovrebbe avere un corpo nero affinché la radiazione luminosa emessa da quest'ultimo appaia cromaticamente la più vicina possibile alla radiazione considerata. In figura 2.21 la misura di temperatura colore per alcune fonti d'illuminazione tipiche.



*Figura 2.21: Temperatura Colore*

**Bilanciamento del bianco.** Il bilanciamento del bianco regola la quantità di rosso e di blu dell'immagine. Questa regolazione permette di ottenere una corretta rappresentazione dei colori dell'immagine andando a massimizzare il valore di luminosità associato ai pixel di oggetti bianchi. Come per il tempo d'esposizione, è possibile impostare un bilanciamento del bianco assoluto o fare in modo che questo venga ricalcolato e corretto ad ogni immagine acquisita. In figura 2.22 viene mostrato come una corretta rappresentazione dei colori può essere fondamentale per ottenere immagini col corretto contrasto e per identificare parti della scena in base al loro colore.



**Figura 2.22:** Bilanciamento del bianco

## 2 Modello della telecamera

Per comprendere e fornire una formulazione matematica di come gli oggetti di una scena vengano campionati e proiettati sulla matrice di pixel del sensore di una telecamera si può sfruttare il cosiddetto *modello pin-hole*. Tale modello si basa sulla semplificazione della telecamera, ovvero si pensa ad essa come un muro con un piccolo foro al centro. Il muro permette di bloccare tutte le radiazioni eccetto quelle passanti per il foro, dietro al quale è posto il sensore d'acquisizione, considerato puntiforme. Il modello pin-hole è il modo più semplice di modellare una telecamera, ma sfortunatamente questo sistema non è realizzabile in realtà perché la formazione dell'immagine dipende dal passaggio di luce dal foro: se il foro è troppo piccolo si rischia di non riuscire ad assorbire una quantità di luce sufficiente ad ottenere un'immagine nitida. Per ovviare a questo problema, nella realtà si utilizzano gli obbiettivi in modo da aumentare la quantità di luce che raggiunge il sensore d'acquisizione. Tuttavia, le lenti utilizzate per realizzare gli obbiettivi introducono distorsioni nelle immagini, le quali rappresentano una non-linearità del modello matematico della telecamera. Esse vengono quindi trattate complicando ulteriormente il modello stesso.

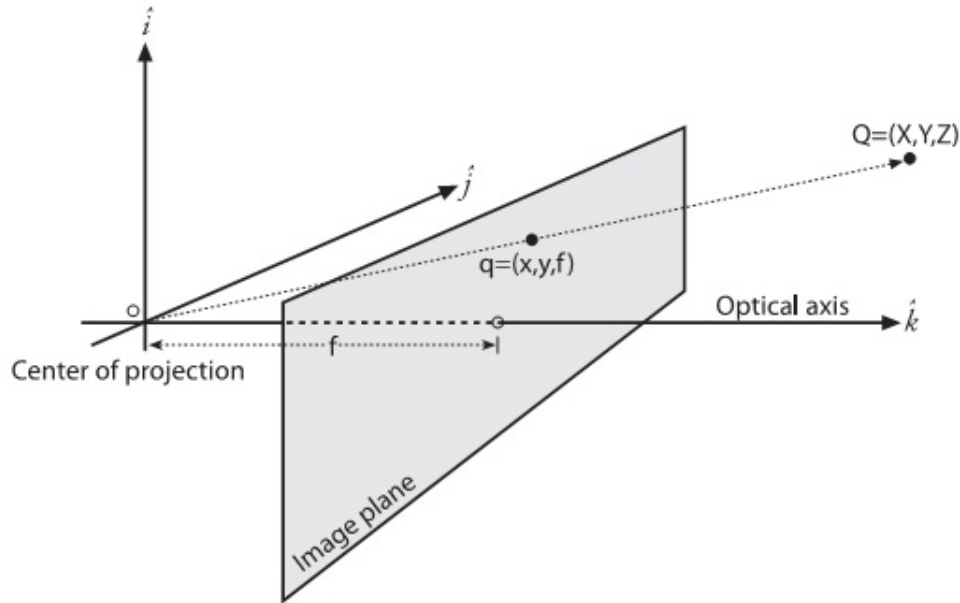
### 2.1 Modello pin-hole

Il modello *pin-hole*, come già accennato, fornisce un semplice modello della trasformazione prospettica che descrive la relazione dei punti della scena reale (3D, in unità metriche) con quelli del piano immagine (2D, in pixel). Ovviamente, come tutti i modelli anche il modello pin-hole si basa su alcune semplificazioni del dispositivo fisico.

La prima approssimazione riguarda il modo in cui la luce viene assorbita dal sensore: questo viene considerato come un piccolo foro dal quale viene assorbito un singolo raggio luminoso da ogni punto della scena inquadrata. Il raggio luminoso viene quindi approssimato con una retta passante dal punto della scena 3D e dal foro (l'obiettivo della telecamera,  $O$ ). Infine, il punto  $\mathbf{Q} = [X, Y, Z]$  viene proiettato su una singola superficie (piano immagine) considerata esattamente perpendicolare alla direzione dell'asse ottico della telecamera



( $z$ ). La proiezione è detta  $\mathbf{q} = [x, y, f]$ . Data tale approssimazione, il piano immagine risultante è sempre messo a fuoco e la massima dimensione dell'immagine ottenibile dipende solamente da un parametro: la *distanza focale* ( $f$ ) della telecamera. Nel modello idealizzato, la distanza focale corrisponde esattamente alla distanza tra l'apertura dell'obiettivo e lo schermo (il piano immagine). La figura 2.23 mostra una rappresentazione schematica di questo semplice modello.



**Figura 2.23:** Modello Pin-Hole

Il sistema di riferimento comunemente usato nelle telecamere è un sistema di riferimento destrorso tale per cui l'asse  $z$  è parallelo all'asse ottico e ha verso uscente dal centro ottico. Per tale motivo spesso ci si riferisce alla coordinata  $Z$  col termine di *profondità* perché essa misura la distanza di un qualsiasi punto dello spazio rispetto all'obbiettivo del dispositivo di visione.

Basandosi sulle similitudini tra i triangoli formati dall'origine dell'obbiettivo  $O$  e dalle proiezioni dei punti  $\mathbf{Q}$  e  $\mathbf{q}$  sul piano formato dagli assi  $z$  e  $x$ , si ricava la proiezione di un punto sul piano immagine:

$$x = f \cdot \frac{X}{Z} .$$

Questa equazione è vera solamente nel caso in cui l'asse ottico della telecamera passi esattamente per il centro dell'immagine. In realtà, il pixel posizionato al centro del sensore d'acquisizione raramente corrisponde al centro dell'immagine. Pertanto, è necessario introdurre due nuovi parametri ( $c_x, c_y$ ) per modellare la posizione del centro dell'immagine.

La trasformazione prospettica risultante dal modello pin-hole è quindi la seguente:

$$\begin{cases} x = f_x \cdot \left(\frac{X}{Z}\right) + c_x \\ y = f_y \cdot \left(\frac{Y}{Z}\right) + c_y \end{cases} \quad (2.5)$$

I parametri presenti nell'equazione (2.5) sono i cosiddetti parametri *intrinseci* della telecamera ( $\mathbf{P_I} = [f_x, f_y, c_x, c_y]$ ) perché sono associati alla trasformazione geometrica con cui una scena viene proiettata su un piano immagine.

Su questo sistema di equazioni si possono fare alcune considerazioni. Innanzitutto si possono far notare le unità di misura delle variabili che entrano nella trasformazione prospettica: mentre le coordinate del punto  $\mathbf{Q} = (X, Y, Z)$  sono tipicamente espresse con unità di misura metriche (metri o millimetri), la proiezione  $\mathbf{q} = (x, y, f)$  deve necessariamente essere espressa in pixel. Infatti, le coordinate  $(x, y)$  fanno riferimento alla posizione, in termini di riga e colonna, del pixel eccitato. Di conseguenza, sia l'offset del centro dell'immagine  $(c_x, c_y)$  che le distanze focali  $(f_x, f_y)$  devono essere espresse in pixel. Si vedrà successivamente (paragrafo 2.2) come è possibile passare dall'informazione di distanza focale ottenibile dal datasheet di un obiettivo alla corrispondente distanza focale del modello pin-hole.

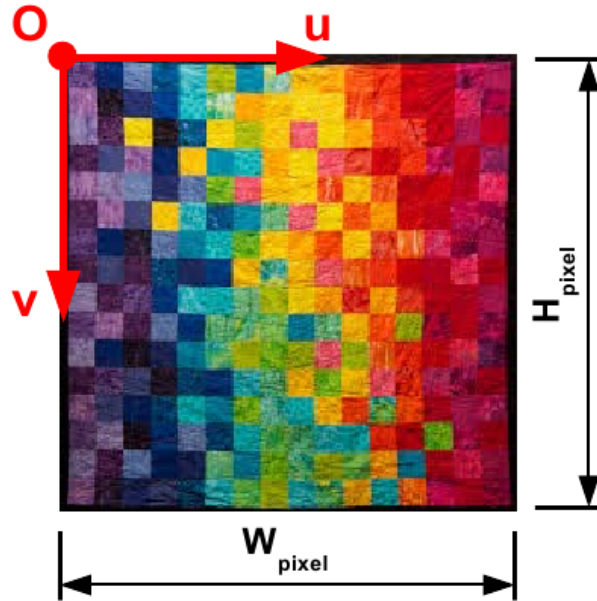
Spesso, per evidenziare l'appartenenza del punto  $\mathbf{q}$  ad un altro sistema di riferimento, ovvero il riferimento del piano immagine, ci si riferisce ad essi come  $\mathbf{q} = (u, v)$ , tralasciando la distanza focale.

Infine, il sistema di riferimento del piano immagine ha generalmente la sua origine nel pixel della prima riga in alto e della prima colonna a sinistra, pertanto l'offset  $(c_x, c_y)$  viene utilizzato anche per il cambiamento di coordinate. Facendo riferimento alla figura 2.24, in assenza di distorsioni si ottengono le relazioni (2.6).

$$\begin{cases} c_x = \frac{W_{pixel}}{2} \\ c_y = \frac{H_{pixel}}{2} \end{cases} \quad (2.6)$$

Infine, è possibile notare che sono state introdotte nel modello due diverse distanze focali per la proiezione dei punti lungo le due direzioni dell'immagine. Infatti, come viene specificato nel paragrafo 2.2, nel calcolo della distanza focale rientra la risoluzione dell'immagine. Pertanto, il rapporto tra le due distanze focali è equivalente al formato dell'immagine, ovvero al rapporto tra la risoluzione orizzontale e quella verticale.





*Figura 2.24: Sistema di riferimento del piano immagine*

### 2.1.1 Rappresentazione matriciale

Per semplificare la scrittura e lo sviluppo delle equazioni che utilizzano il modello pin-hole, è possibile riscrivere il sistema di equazioni (2.5) in forma matriciale. Pertanto, dati un generico punto nello spazio  $\mathbf{Q} = (X, Y, Z)$  e la sua proiezione sull'immagine  $\mathbf{q} = (x, y, 1)$  è possibile definire la matrice  $\mathbf{M}$  dei parametri intrinseci della telecamera

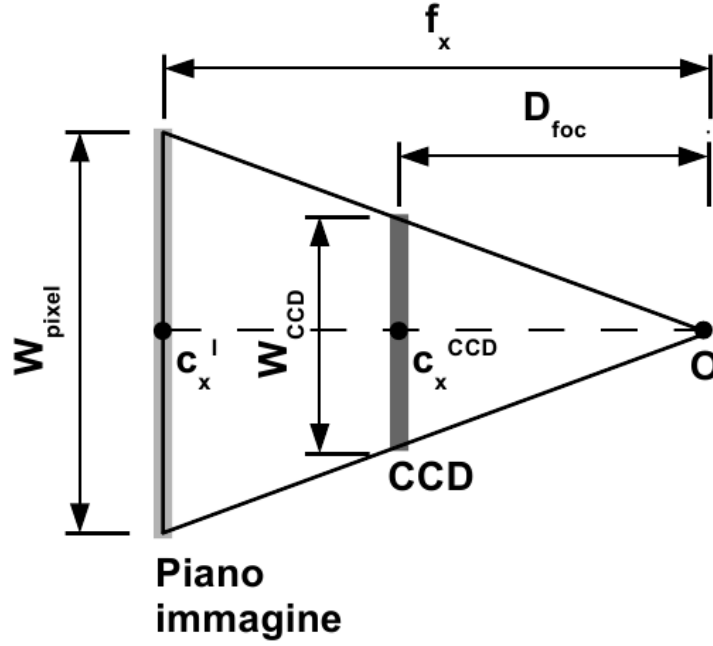
$$\mathbf{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.7)$$

in modo da riscrivere il sistema di equazioni (2.5) come

$$\mathbf{q} = \frac{1}{Z}(\mathbf{M}\mathbf{Q}) . \quad (2.8)$$

## 2.2 Distanze focali

Nel paragrafo 1.2 è stato introdotto il concetto di distanza focale e i suoi effetti sulla formazione delle immagini. Tuttavia, le distanze focali  $f_x$  e  $f_y$  utilizzate nel modello della telecamera dato dal sistema (2.5) sono espresse in pixel, mentre la focale  $D_{foc}$  ricavabile dal datasheet degli obiettivi è espressa in  $mm$ . Di conseguenza, un'ulteriore relazione è necessaria per passare da una definizione di distanza focale all'altra. Data la dimensione in  $mm$  del sensore di acquisizione  $(W_{CCD}, H_{CCD})$  e la massima risoluzione in pixel delle



**Figura 2.25:** Calcolo della relazione tra distanze focali obiettivo e modello

immagini ottenibili con il dispositivo  $(W_{pixel}, H_{pixel})$ , la relazione tra le distanze focali è ottenibile operando una similitudine sui triangoli in figura 2.25.

Per quanto riguarda la distanza focale  $f_x$  la relazione risultante è la seguente:

$$\frac{D_{foc}}{\left(\frac{W_{CCD}}{2}\right)} = \frac{f_x}{\left(\frac{W_{pixel}}{2}\right)} \quad (2.9)$$

$$f_x = \frac{W_{pixel}}{W_{CCD}} \cdot D_{foc}$$

e procedendo in modo analogo per  $f_y$  si ottengono entrambe le relazioni desiderate

$$\begin{cases} f_x = \frac{W_{pixel}}{W_{CCD}} \cdot D_{foc} \\ f_y = \frac{H_{pixel}}{H_{CCD}} \cdot D_{foc} \end{cases} \quad (2.10)$$

## 2.3 Distorsioni

Nel paragrafo introduttivo di questa sezione si è già parlato della necessità dell'utilizzo di lenti per la realizzazione nella pratica degli obbiettivi. Tuttavia, le lenti portano all'introduzione di fenomeni di *distorsione* nell'immagine. A livello teorico è possibile definire alcuni principi per la realizzazione di una lente che non introduca o comunque riduca le distorsioni, ma in pratica questi principi non sono applicabili per due principali ragioni:

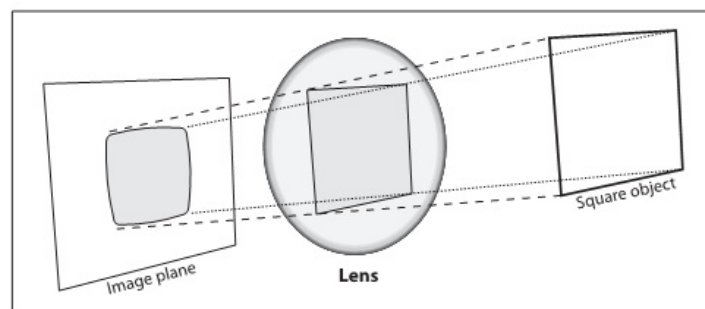
- una lente non sferica (ad esempio parabolica) ridurrebbe la distorsione, ma in pratica tutte le lenti sono sferiche perché tecnologicamente più semplici da realizzare;
- è meccanicamente difficile allineare perfettamente la lente e il sensore di acquisizione.

Vista l'impossibilità di eliminazione delle distorsioni, queste devono essere matematicamente trattate come non-linearità del modello. Si rende quindi necessario introdurre, nel modello (2.5) dei parametri che permettano di modellizzare gli effetti distorsivi introdotti dalle lenti.

In particolare, è possibile classificare le distorsioni in due categorie distinte:

- *distorsioni radiali*, derivanti dalla forma della lente;
- *distorsioni tangenziali*, derivanti dal non preciso accoppiamento della lente, in senso di obbiettivo, al sensore di acquisizione immagini montato sulla scheda della telecamera.

Per quanto riguarda le distorsioni radiali le lenti degli obbiettivi distorcono le immagini soprattutto per quanto riguarda i pixel più esterni, ovvero molto vicini ai bordi del sensore. Questo fenomeno viene chiamato effetto a *occhio di pesce* (*fish-eye effect*) ed è dovuto all'utilizzo di lenti sferiche (figura 2.26).



**Figura 2.26:** Lente sferica e distorsione radiale

L'origine del nome risiede nel fatto che questo tipo di distorsione aumenta all'aumentare del raggio di curvatura delle lenti. Un tipo particolare di obbiettivo è, appunto, l'obbiettivo fish-eye (dal profilo tondeggiante come l'occhio di un pesce) che permette di avere un angolo di visione di  $180^\circ$ , ovvero tra le 3 e 6 volte maggiore degli obbiettivi standard. L'effetto che si ha sulle immagini ottenute è mostrato in figura 2.27, in cui sembra che la scena sia disposta appunto sulla superficie di una sfera. Infatti, i punti appartenenti ad una retta appaiono disposti su una curva.

La distorsione radiale è nulla nel centro dell'immagine e aumenta man mano che ci si sposta verso la periferia. Di conseguenza, detta  $r$  la distanza di un punto dell'immagine

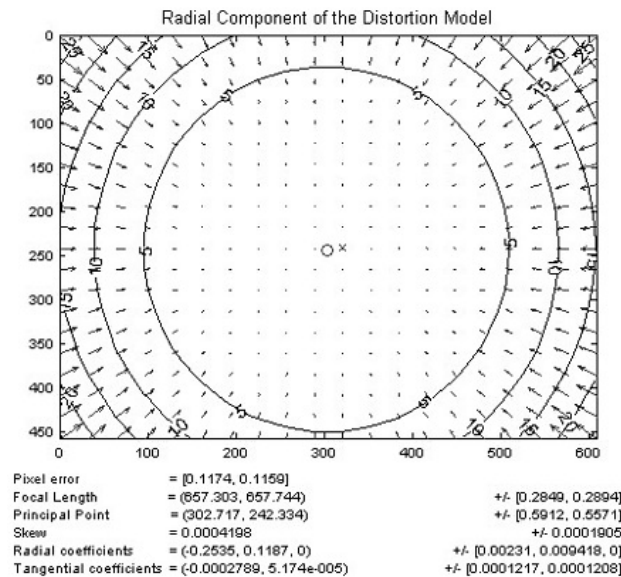


**Figura 2.27:** Distorsioni radiali di un obiettivo fish-eye

dal centro, questo effetto può venire modellizzato come una serie di Taylor attorno ad  $r = 0$ . L'ordine che deve avere la serie per ottenere risultati soddisfacenti dipende dal tipo di dispositivo e di ottica utilizzati: serie di grado maggiore riescono a modellare distorsioni più complesse, come quelle dovute all'effetto fish-eye. Come espresso nella (2.11), buoni risultati si ottengono con una serie di ordine  $n = 6$ .

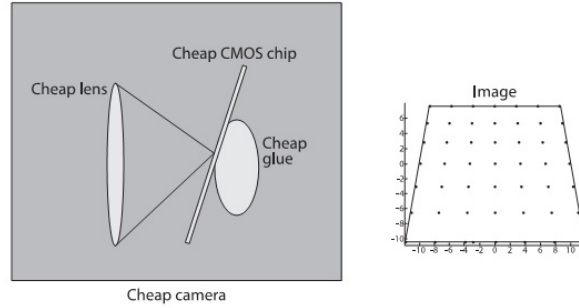
$$\begin{cases} x_{corrected} = x \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{corrected} = y \cdot (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \quad (2.11)$$

Le coordinate  $(x, y)$  si riferiscono alla posizione originale di un pixel sull'immagine, mentre  $x_{corrected}$  e  $y_{corrected}$  sono le coordinate della posizione corretta. Un modo efficace per misurare la distorsione radiale è quello di ricorrere ad una griglia quadrata di punti: se l'effetto della distorsione è elevato, questi punti tenderanno a disporsi lungo circonferenze piuttosto che lungo i lati della griglia, come in figura 2.28.



**Figura 2.28:** Misura della distorsione radiale

La distorsione tangenziale, invece, è dovuta ad un disallineamento tra il centro ottico del sensore della telecamera e il fuoco dell'obiettivo montato sulla telecamera stessa (figura 2.29).

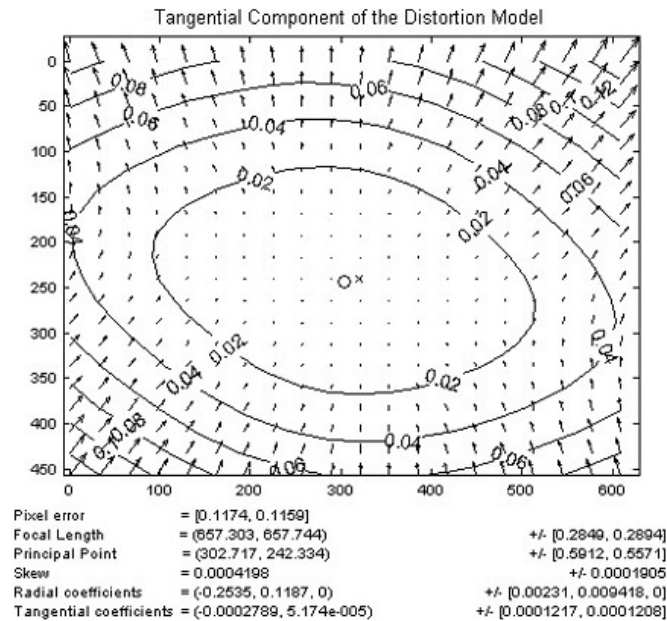


**Figura 2.29:** Causa della distorsione tangenziale

Rispetto alla distorsione radiale, la modellazione matematica della distorsione tangenziale è più complessa. Le relazioni (2.12) mostrano come modellare la distorsione tangenziale con l'aggiunta di due ulteriori coefficienti:  $p_1$  e  $p_2$ .

$$\begin{cases} x_{corrected} = x + [2p_1y + p_2(r^2 + 2x^2)] \\ y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2x] \end{cases} \quad (2.12)$$

La figura 2.30 mostra il significato della modellazione (2.12).



**Figura 2.30:** Misura della distorsione tangenziale

I punti sottoposti a distorsioni tangenziali si distribuiscono lungo delle curve approssimabili a degli ellissoidi. Il valore di distorsione, infatti, dipende non solo dalla distanza del punto

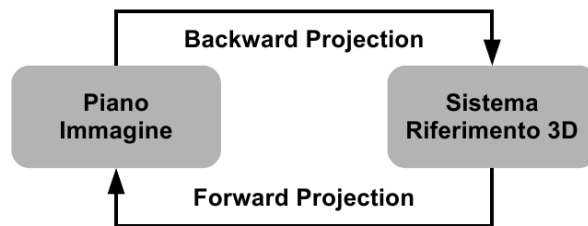
dal centro dell'immagine, ma anche dalle sue coordinate  $(x, y)$ .

In totale, quindi, è necessario calcolare cinque parametri per avere il modello completo delle distorsioni introdotte dalle lenti. I cinque parametri vengono solitamente inseriti nel vettore delle distorsioni, noto anche come vettore dei parametri *estrinseci* della telecamera:

$$\mathbf{d} = \begin{bmatrix} k_1 & k_2 & k_3 & p_1 & p_2 \end{bmatrix} . \quad (2.13)$$

## 2.4 Ricostruzione 3D

I modelli della telecamera e delle distorsioni dell'ottica permettono di definire tutte le relazioni necessarie ad effettuare le trasformazioni dal sistema di riferimento dello spazio al piano immagine e viceversa. In figura 2.31 si mostra la definizione delle due operazioni: si parla di *backward projection* quando si cerca di stimare la posizione 3D di un punto a partire dalle immagini, mentre l'operazione opposta, eseguita implicitamente dal modello della telecamera, è detta *forward projection*.



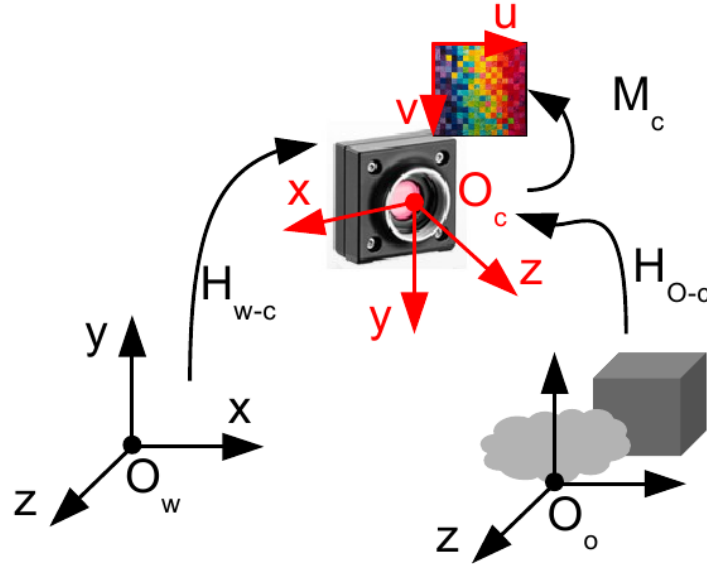
**Figura 2.31:** Trasformazioni da piano immagine a spazio 3D

Come si vedrà più avanti (capitolo 6) quest'ultima operazione risulta utile per trovare le corrispondenze tra un oggetto noto inserito nello spazio con una certa posa e la sua proiezione sul piano immagine. Al contrario, l'operazione di backward projection è nota anche come *ricostruzione 3D* perchè permette di ricostruire la posizione degli oggetti all'interno della scena a partire da una o più immagini.

Il problema generale della ricostruzione 3D, riassunto in figura 2.32, prevede due passi:

1. Calcolo delle coordinate di un punto nello spazio rispetto ad un sistema di riferimento solidale con la telecamera, dove l'origine del sistema di riferimento telecamera  $O_c = (0, 0, 0)$  coincide con il punto focale del dispositivo di visione, l'asse  $Z$  è parallelo all'asse ottico dell'obiettivo e uscente da esso, l'asse  $X$  ha stessa direzione e verso dell'asse  $u$  del piano immagine e l'asse  $Y$  ha stessa direzione, ma verso opposto all'asse  $v$  del piano immagine;

2. Rappresentazione delle coordinate in un sistema di riferimento assoluto, nota la trasformazione geometrica  $H_W^C$  che descrive come è posizionata la telecamera nel sistema di riferimento assoluto.



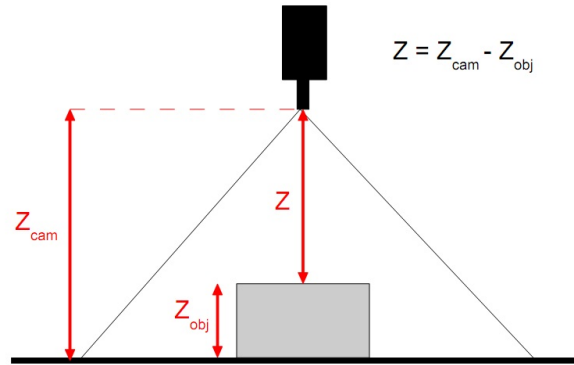
**Figura 2.32:** Problema della ricostruzione 3D

Di seguito vengono elencate e descritte alcune delle tecniche utilizzate per la soluzione del primo punto, ovvero la ricostruzione delle coordinate 3D dei punti di una scena nel sistema di riferimento telecamera. Come risulta evidente dalla 2.5, il sistema di equazioni che regolano la trasformazione tra piano immagine e spazio della scena è un sistema sottodimensionato: note le coordinate  $(x, y)$  di un punto del piano immagine, per ottenere le coordinate  $(X, Y, Z)$  si ha a che fare con un sistema di 3 incognite e 2 sole equazioni. Pertanto, le tecniche di ricostruzione della scena sfruttano sempre una conoscenza a priori che permette di aggiungere equazioni per sovradeterminare il problema e rendere quindi possibile la sua soluzione. A queste tecniche, le quali si basano solo sulla conoscenza dei parametri intrinseci della telecamera, verrà aggiunta la descrizione, nel paragrafo 3, di una modalità per la ricostruzione della posa 3D di un oggetto nella scena a partire dal confronto con un suo modello tridimensionale.

#### 2.4.1 Distanza nota

Nel caso in cui il task di visione sia (i) relativo all'individuazione di oggetti planari o con scarsa e determinata profondità con (ii) l'asse ottico della telecamera perpendicolare al piano d'appoggio degli oggetti e (iii) sia nota la posizione della telecamera rispetto

all'oggetto, allora si può considerare costante la distanza  $Z_O^C$  tra oggetto e telecamera (figura 2.33).



**Figura 2.33:** Ricostruzione planare

La soluzione è quindi data dal sistema di equazioni (2.14).

$$\begin{cases} X = \frac{x - c_x}{f_x} \cdot Z_O^C \\ Y = \frac{y - c_y}{f_y} \cdot Z_O^C \end{cases} \quad (2.14)$$

Questo metodo è ovviamente il più usato nella soluzione dei classici problemi di visione 2D a causa dell'estrema semplicità d'implementazione.

### 2.4.2 Triangolazione attiva

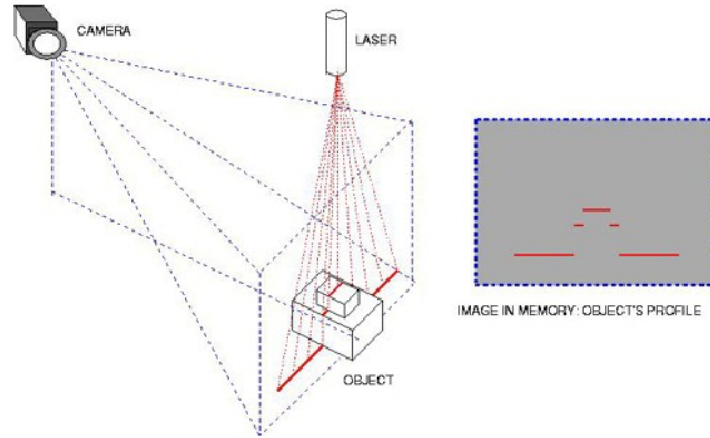
Le tecniche di ricostruzione tridimensionale basate su *triangolazione attiva* prevedono l'aggiunta di informazione alla scena tramite l'utilizzo di una fonte d'energia esterna. L'approccio più utilizzato è quello della *triangolazione laser* in cui si fa uso di un illuminatore laser con un'ottica lineare (linea laser) montato su un particolare sistema di movimentazione. Questo sistema permette di muovere la linea laser all'interno della scena, in modo da "spazzolare" gli oggetti conoscendo nel contempo la posizione e orientamento relativi di laser e telecamera. La ricostruzione tridimensionale passa attraverso l'individuazione del laser all'interno della scena e, successivamente, l'applicazione di semplici trasformazioni geometriche. Tale individuazione è facilitata dal fatto che la luce introdotta dal laser tende ad essere saturata dal sensore di acquisizione e, pertanto, la sua luminosità all'interno di una generica immagine può essere regolata modificando l'esposizione della telecamera.

Un'alternativa alla movimentazione del laser è costituita dal mettere in movimento gli oggetti: l'applicazione tipica prevede oggetti movimentati da un nastro trasportatore, col

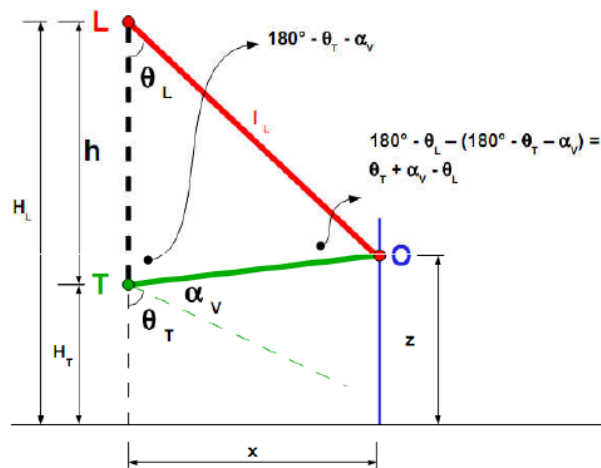


sistema di visione che man mano ottiene le informazioni 3D desiderate.

In figura 2.34 è riportato un semplice modello di un sistema di triangolazione attiva a telecamera fissa e movimentazione del laser.



(a) Configurazione del sistema.



(b) Modello del sistema.

**Figura 2.34:** Ricostruzione con triangolazione attiva

Dato l'orientamento della telecamera  $\theta_T$  e del laser  $\theta_L$ , dall'individuazione della coordinata  $v$  di un pixel dell'immagine illuminato dal fascio laser è possibile ricavare  $\alpha_V$  come

$$\alpha_V = \arctan\left(\frac{v - c_y}{f_y}\right).$$

Una soluzione analoga è ottenuta per la coordinata  $u$  dello stesso pixel:

$$\alpha_H = \arctan\left(\frac{u - c_x}{f_x}\right).$$

Infine, nota la distanza  $h$  tra sorgente laser e telecamera, si ha:

$$l_L = \frac{\sin(\theta_T + \alpha_V)}{\sin(\theta_T + \alpha_V - \theta_L)} \cdot h.$$

Sia  $H_L$  l'altezza della sorgente laser dal piano di riferimento del sistema. La soluzione al problema della ricostruzione 3D è dato dalle equazioni della (2.15), dove nel sistema di riferimento assoluto  $x$  è la distanza tra punto analizzato e telecamera,  $z$  la quota e  $y$  la coordinata restante.

$$\begin{cases} x_O = l_L \cdot \sin(\theta_L) \\ y_O = x_O \cdot \tan(\alpha_H) \\ z_O = H_L - l_L \cdot \cos(\theta_L) \end{cases} \quad . \quad (2.15)$$

### 2.4.3 Triangolazione passiva

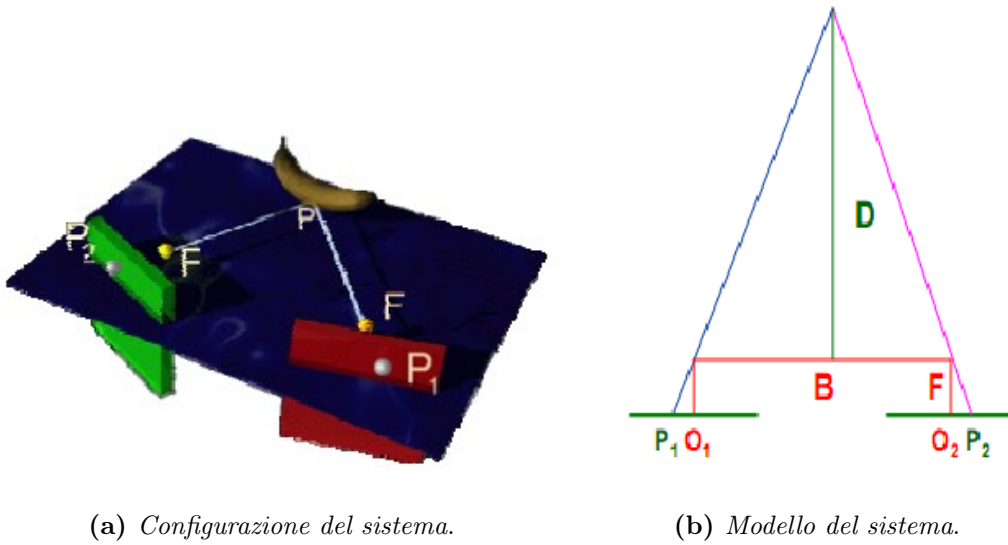
Se due dispositivi posti a distanza nota possono vedere lo stesso punto di un oggetto, allora la distanza dell'oggetto può essere calcolata conoscendo la posizione e l'orientamento relativo dei due dispositivi, ovvero per *triangolazione* nel vero senso della parola.

La tecnica di ricostruzione 3D più nota ed analizzata in letteratura è infatti la *triangolazione passiva*, nota anche come *ricostruzione stereoscopica*. Mentre la triangolazione attiva utilizza una fonte d'energia esterna per sovradeterminare il sistema di equazioni della ricostruzione 3D, la stereoscopia raggiunge lo stesso risultato aggiungendo una telecamera, ovvero un altro set di due equazioni. Pertanto la soluzione del problema di ricostruzione tridimensionale è ottenuto risolvendo un sistema di 4 equazioni in 3 incognite. In figura 2.35 si mostra la possibile schematizzazione e modellazione di un sistema stereoscopico. Tuttavia, l'aggiunta di un ulteriore dispositivo di visione aggiunge maggiore complessità all'identificazione degli oggetti: non è più sufficiente identificare l'oggetto in una singola immagine, ma in due immagini differenti e sincronizzate, ovvero acquisite nello stesso istante temporale. Questo problema è noto in letteratura col termine *ricerca delle corrispondenze*.

I passi fondamentali da seguire per implementare un algoritmo di stereoscopia sono i seguenti:

1. acquisizione di due immagini distinte, conoscendo la distanza tra le telecamere;
2. tracciamento delle linee su cui giacciono i punti tridimensionali reali;
3. intersezione di tali linee le linee.

Il principale problema da risolvere è quindi quello del *matching*, cioè la ricerca di corrispondenza di punti nelle due immagini, al fine di misurare la disparità tra i due punti corrispondenti. Una soluzione a questo problema è stata trovata da *Marr e Poggio*, il cui algoritmo lavora sui seguenti vincoli:



**Figura 2.35:** Ricostruzione con triangolazione passiva

- *compatibilità*: i punti neri possono corrispondere solo a punti neri;
- *unicità*: quasi sempre un punto nero di un'immagine può corrispondere a non più di un punto nero sull'altra immagine;
- *continuità*: la disparità varia linearmente quasi ovunque nell'immagine, eccetto ai bordi.

I metodi di triangolazione permettono la ricostruzione 3D a partire dal vettore delle corrispondenze tra i punti di due immagini. Si può distinguere tra la (i) stereoscopia classica che utilizza corrispondenze tra immagini riprese da dispositivi diversi, oppure la (ii) triangolazione del moto che sfrutta la conoscenza del moto dell'oggetto e l'individuazione delle corrispondenze in immagini acquisite da uno stesso dispositivo ad intervalli di tempo differenti.

**Sistemi di riferimento.** Si considerano i seguenti sistemi di riferimento:

- Riferimento assoluto centrato in  $O$ ;
- Riferimento relativo alla telecamera e centrato nel fuoco dell'ottica della prima telecamera (o della telecamera nel primo istante temporale). La posa della telecamera nel sistema di riferimento assoluto è definita dalla matrice di rotazione  $\mathbf{R}_O^1$  e dal vettore di traslazione  $\mathbf{t}_O^1$ ;
- Riferimento relativo alla seconda telecamera o posa, espresso in termini della matrice  $\mathbf{R}_O^2$  e del vettore  $\mathbf{t}_O^2$ ,

Sempre riferendosi al sistema di riferimento assoluto, ulteriori matrici di rotazione ( $\mathbf{R}_{\text{obj}}$ ) e vettori traslazione ( $\mathbf{t}_{\text{obj}}$ ) vengono utilizzati per modellare l'eventuale movimento degli oggetti all'interno della scena. L'oggetto è completamente descritto da  $n$  punti  $\mathbf{P}_{\text{obj}} = [P_1, P_2, \dots, P_n]$ , le cui coordinate spaziali sono espresse nel sistema di riferimento della telecamera principale (una delle due telecamere del sistema stereoscopico).

Risultano quindi definite le trasformazioni rigide per il passaggio di coordinate dal sistema di riferimento telecamera al sistema di riferimento assoluto e viceversa:

$$\begin{aligned} \mathbf{P}_i^O &= \mathbf{R}_O \cdot \mathbf{P}_i + \mathbf{t}_O \\ \mathbf{P}_i &= \mathbf{R}_O^{-1} \cdot (\mathbf{P}_i^O - \mathbf{t}_O) \end{aligned} \quad (2.16)$$

**Informazione dei sistemi di visione.** Sfruttando le relazioni della trasformazione prospettica (2.5) per entrambe le coppie telecamere-posa, si ottiene il sistema (2.17) formato da 4 equazioni, il quale rappresenta il problema della ricostruzione 3D stereoscopia.

$$\begin{cases} X_1 = \frac{u_1 - u_1^0}{f_1^u} \cdot Z_1 \\ Y_1 = \frac{v_1 - v_1^0}{f_1^v} \cdot Z_1 \\ X_2 = \frac{u_2 - u_2^0}{f_2^u} \cdot Z_2 \\ Y_2 = \frac{v_2 - v_2^0}{f_2^v} \cdot Z_2 \end{cases} \quad (2.17)$$

La stessa operazione può essere eseguita utilizzando un numero qualsiasi  $n \geq 2$  di pose. In questo modo si ottiene sempre un sistema sovradeterminato di  $2n$  equazioni e 3 incognite. L'utilizzo di più telecamere potrebbe portare ad una migliore stima della ricostruzione delle coordinate 3D dei punti individuati, mentre l'utilizzo di più pose ad istanti di tempo diversi permette di stimare nel tempo il movimento della scena.

**Oggetto in moto ripreso dalla stessa telecamera.** Per risolvere in via numerica il sistema ottenuto può essere vantaggioso eseguire alcune trasformazioni di coordinate. Innanzitutto, è possibile introdurre la trasformazione geometrica che mette in relazione la posa dell'oggetto nei sistemi di riferimento delle due scene:

$$\mathbf{P}_O = \mathbf{R}_O^i \cdot \mathbf{P}_O^i + \mathbf{t}_O^i \quad .$$

Da qui in avanti si procede mostrando la soluzione del problema di stereoscopia classico, ovvero con la ripresa della stessa scena da parte di due telecamere differenti di cui si conosce

la posa rispetto all'origine del sistema di riferimento, ovvero  $(\mathbf{P}_O^{\text{cam2}}) = \mathbf{P}_O^{\text{cam1}}$ . Lo stesso approccio può tuttavia essere utilizzato per ricavare le equazioni utili che permettono di stimare il moto degli oggetti in immagini acquisite a istanti temporali differenti:

- Oggetto in moto e telecamera ferma. In questo caso la posizione della telecamera rispetto al sistema di riferimento assoluto non cambia ( $\mathbf{R}_O^1 = \mathbf{R}_O^2$ ,  $\mathbf{t}_O^1 = \mathbf{t}_O^2$ ). Il problema diventa la stima dello spostamento effettuato dall'oggetto nota la posizione dello stesso al punto precedente. L'aggiornamento della posizione è dato da:  $\mathbf{P}_O^2 = \mathbf{R}_{p1}^{\mathbf{P}^2} \cdot \mathbf{P}_O^1 + \mathbf{t}_{p1}^{\mathbf{P}^2}$ .
- Oggetto fermo e telecamera in movimento attorno all'oggetto. In questo caso, la posizione dell'oggetto è costante nel tempo ( $\mathbf{P}_O^{\text{cam2}} = \mathbf{P}_O^{\text{cam1}}$ ) e nel vettore delle incognite entrano i termini relativi allo spostamento della telecamera, ovvero  $\mathbf{R}_O^2 = \mathbf{R}_{\text{cam}} \cdot \mathbf{R}_O^1$  e  $\mathbf{t}_O^2 = \mathbf{R}_{\text{cam}} \cdot \mathbf{t}_O^1 + \mathbf{t}_{\text{cam}}$ .

Utilizzando le trasformazioni di coordinate (2.16) si ottiene l'equazione

$$\mathbf{R}_O^2 \cdot \mathbf{P}_O^2 + \mathbf{t}_O^2 = \mathbf{R}_O^1 \cdot \mathbf{P}_O^1 + \mathbf{t}_O^1 \quad (2.18)$$

che permette di mettere in relazione quanto osservato da una telecamera con l'osservazione della seconda.

Essendo il vettore delle coordinate del punto nello spazio ( $\mathbf{P}_i$ ) composto da tre elementi, l'equazione (2.18) è in realtà un sistema di 3 equazioni. La  $i$ -esima equazione è data da:

$$\begin{aligned} R_O^2(i, 1) \cdot X_2 + R_O^2(i, 2) \cdot Y_2 + R_O^2(i, 3) \cdot Z_2 - R_O^1(i, 1) \cdot X_1 \\ - R_O^1(i, 2) \cdot Y_1 - R_O^1(i, 3) \cdot Z_1 = t_O^1(i) - t_O^2(i) \end{aligned}$$

Date  $n$  coppie di telecamere è possibile ottenere  $3 \cdot (n - 1)$  equazioni aggiuntive.

**Notazione matriciale.** L'intento di tutte queste operazioni è quello di arrivare a scrivere il sistema di equazioni in notazione matriciale, in modo da poter applicare il metodo dei *minimi quadrati* (vedi 3) per ottenere facilmente una soluzione del problema. Il sistema (2.17) viene riscritto nella forma

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \quad , \quad (2.19)$$

dove

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \frac{-u_1+u_1^0}{f_1^u} & 0 & 0 & 0 \\ 0 & 1 & \frac{-v_1+v_1^0}{f_1^v} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \frac{-u_2+u_2^0}{f_2^u} \\ 0 & 0 & 0 & 0 & 1 & \frac{-v_2+v_2^0}{f_2^v} \\ -R_O^1(1,1) & -R_O^1(1,2) & -R_O^1(1,3) & R_O^2(1,1) & R_O^2(1,2) & R_O^2(1,3) \\ -R_O^1(2,1) & -R_O^1(2,2) & -R_O^1(2,3) & R_O^2(2,1) & R_O^2(2,2) & R_O^2(2,3) \\ -R_O^1(3,1) & -R_O^1(3,2) & -R_O^1(3,3) & R_O^2(3,1) & R_O^2(3,2) & R_O^2(3,3) \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ t_O^1(1) - t_O^2(1) \\ t_O^1(2) - t_O^2(2) \\ t_O^1(3) - t_O^2(3) \end{bmatrix}.$$

Il vettore delle incognite è dato da:

$$\mathbf{x} = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ X_2 \\ Y_2 \\ Z_2 \end{bmatrix}.$$

La soluzione è ottenuta tramite il metodo dei minimi quadrati, cioè passando dalla matrice pseudo-inversa:

$$\mathbf{x} = \dagger(\mathbf{A}) \cdot \mathbf{b}, \quad \dagger(\mathbf{A}) = (\mathbf{A}^T \cdot \mathbf{A})^{-1} \cdot \mathbf{A}^T. \quad (2.20)$$

**Notazione compatta.** La matrice  $\mathbf{A}$  e il vettore  $\mathbf{b}$  possono essere costruiti (indipendentemente dal numero di pose considerate) introducendo una notazione compatta per ogni sottoparte.

Date  $n$  pose, la matrice  $\mathbf{A}$  ha dimensioni  $(2n + 3(n-1) \times 3n)$ .

Per quanto riguarda la generica telecamera  $i$ , dalle equazioni di proiezione si ottiene la seguente sottomatrice

$$\mathbf{M}_{\text{cam}}^i = \begin{bmatrix} 1 & 0 & \frac{u_0^i - u^i}{f_u^i} \\ 0 & 1 & \frac{v_0^i - v^i}{f_v^i} \end{bmatrix}. \quad (2.21)$$

La matrice  $\mathbf{A}$  generica è quindi pari a:

$$\mathbf{A} = \begin{bmatrix} M_{cam}^1 & \mathbf{0}^2 & \mathbf{0}^3 \dots & \mathbf{0}^n \\ \mathbf{0}^1 & M_{cam}^2 & \mathbf{0}^3 \dots & \dots & \mathbf{0}^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}^1 & \mathbf{0}^2 & \dots & \mathbf{0}^{n-1} & M_{cam}^n \\ \dots & \dots & \dots & \dots & \dots \\ -R_O^1 & R_0^2 & \mathbf{0}^3 & \dots & \mathbf{0}^n \\ -R_O^1 & \mathbf{0}^2 & \dots & \mathbf{0}^{n-1} & R_0^n \end{bmatrix}. \quad (2.22)$$

Allo stesso modo si può definire il vettore  $\mathbf{b}$  generico

$$\mathbf{b} = \begin{bmatrix} \mathbf{0}^0 \\ \vdots \\ \mathbf{0}^{2n} \\ t_O^1 - t_O^2 \\ t_O^1 - t_O^n \end{bmatrix}. \quad (2.23)$$

### 3 Calibrazione delle telecamere

Le tecniche di *calibrazione* sono necessarie per una migliore comprensione delle immagini perché permettono di:

- ricavare i parametri intrinseci del dispositivo di acquisizione (telecamera e obiettivo), necessari per ottenere la trasformazione dalle coordinate del piano immagine verso le coordinate del mondo reale;
- ridurre gli effetti della distorsione introdotti dalle lenti;
- affinare il modello geometrico della telecamera.

Lo scopo finale della calibrazione consiste nell'identificazione dei parametri del modello della telecamera presentati nel paragrafo 2, ovvero i parametri *intrinseci* della matrice di proiezione e quelli *estrinseci* che permettono di modellare gli effetti di distorsione introdotti dalle ottiche.

In questo paragrafo viene innanzitutto rivisto il modello di trasformazione prospettica da scena inquadrata a piano immagine introducendo il concetto di *omografia*, al fine di mostrare successivamente come identificarne i parametri e applicarli per la limitazione o rimozione degli effetti distorsivi delle lenti.

### 3.1 Omografia

In visione artificiale si definisce *omografia planare* una mappa di proiezione da un piano ad un altro. Nel caso della calibrazione delle telecamere la trasformazione che descrive il cambio di coordinate dal sistema di riferimento tridimensionale della scena al piano 2D dell'immagine costituisce l'omografia fondamentale. Matematicamente, è possibile e utile esprimere questa mappa in termini di moltiplicazioni tra matrici di coordinate omogenee. Siano, al solito, definite le coordinate dei punti della scena  $\mathbf{Q} = [X \ Y \ Z \ 1]^T$  e della loro proiezione sul piano immagine  $\mathbf{q} = [x \ y \ 1]^T$ . La trasformazione omografica risulta definita, a meno di un fattore di scala  $s$ , come

$$\mathbf{q} = s\mathbf{H}\mathbf{Q} . \quad (2.24)$$

La matrice  $\mathbf{H}$  è l'omografia e, nel caso delle telecamere, risulta essere una combinazione di una trasformazione prospettica, data dalla matrice dei parametri intrinseci della telecamera ( $\mathbf{M}$ ), e di una trasformazione fisica, data dalla matrice di rototraslazione ( $\mathbf{W}$ ) che permette di ottenere le coordinate dei punti della scena nel sistema di riferimento della telecamera. Pertanto, l'equazione (2.24) può essere riscritta esplicitandone le varie componenti, ovvero come

$$\mathbf{q} = s\mathbf{M}\mathbf{W}\mathbf{Q} , \quad (2.25)$$

dove

$$\mathbf{W} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$$

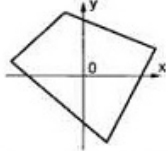
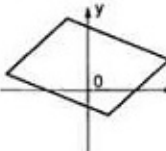
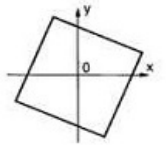
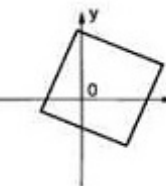
e

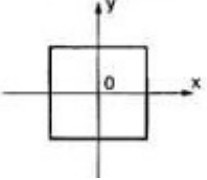
$$\mathbf{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} .$$



### 3.1.1 Esempi di omografie

In tabella 2.1 vengono mostrati alcuni tipi di omografia, la loro forma matriciale e le proprietà geometriche invarianti.

Omografia	Vincoli su $\mathbf{H}$	Esempio 2D	Invarianti
Proiezione	$\det(\mathbf{H}) \neq 0$		Collinearità, tangente, rapporto dimensioni
Trasformazione affine	$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$ $\det(\mathbf{A}) \neq 0$		<p>Invarianti a proiezioni, più:</p> <ul style="list-style-type: none"> <li>• Parallelismo;</li> <li>• Rapporto lunghezze rette parallele;</li> <li>• Rapporto tra aree;</li> <li>• Combinazioni lineari rispetto al centroide.</li> </ul>
Similarità	$\mathbf{H} = \begin{bmatrix} s\mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$ $\mathbf{R}^T \mathbf{R} = \mathbf{I}$ $\det(\mathbf{R}) = 1$ $s > 0$		<p>Invarianti affini, più:</p> <ul style="list-style-type: none"> <li>• Angoli;</li> <li>• Rapporti tra lunghezze.</li> </ul>
Trasformazione metrica	$\mathbf{H} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$ $\mathbf{R}^T \mathbf{R} = \mathbf{I}$ $\det(\mathbf{R}) = 1$		<p>Invarianti similarità, più:</p> <ul style="list-style-type: none"> <li>• lunghezze;</li> <li>• area (volume).</li> </ul>

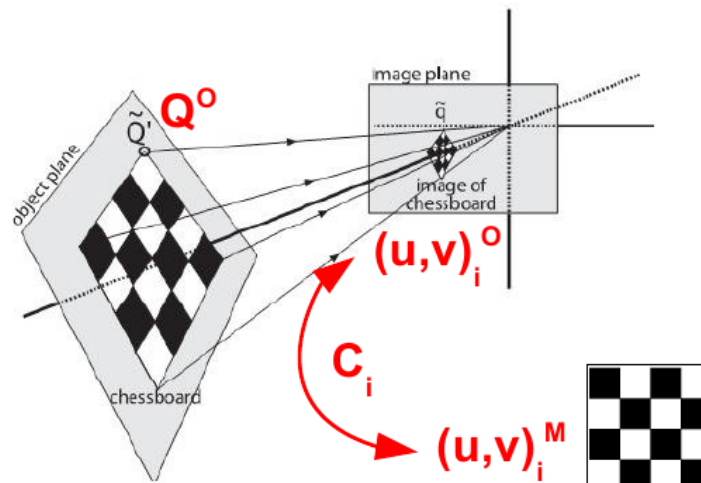
Identità	$H = I$		Caso banale: tutto è invariante
----------	---------	---	---------------------------------

**Tabella 2.1:** Tipi di omografia

L'analisi delle proprietà geometriche invarianti è fondamentale per la visione artificiale perché permette di individuare un sottoinsieme di misure da effettuare in funzione del tipo di movimento e trasformazioni geometriche presenti nella scena inquadrata.

### 3.2 Stima dell'omografia

Un compito frequente in visione artificiale è il calcolo dell'omografia a partire dal vettore delle corrispondenze di punti in diverse immagini. Per *corrispondenza* si intende qui un insieme di coppie ordinate di  $m$  punti ( $C_{i=1}^m = [(u, v)_i^1, (u, v)_i^2]_{i=1}^m$ ) individuati in due diverse immagini. Le coppie possono essere individuate a partire da due immagini dello stesso oggetto in pose differenti, ma per la soluzione del problema della calibrazione, come mostrato in figura 2.36, il primo punto della coppia è solitamente preso da un modello di un oggetto da utilizzare per la calibrazione stessa.



**Figura 2.36:** Corrispondenze tra modello e osservazione

Di conseguenza, ci si può riferire al primo punto come punto del modello ( $(u, v)_i^1 = (u, v)_i^M$ ) e all'altro come punto dell'osservazione ( $(u, v)_i^2 = (u, v)_i^O$ ) e le corrispondenze permettono di passare dal sistema di riferimento del modello al riferimento delle osservazioni.

La stima della matrice  $\mathbf{H}$  passa dalla risoluzione del sistema omogeneo di equazioni lineari (2.26) avente come incognite l'omografia  $\mathbf{H}$  e il fattore di scala  $\alpha_i$ .

$$\alpha_i \mathbf{u}_i^1 = \mathbf{H} \mathbf{u}_i^2, \quad i = 1, \dots, m. \quad (2.26)$$

Avendo a disposizione  $d$  immagini, il sistema risulta avere  $m(d+1)$  equazioni e  $m+(d+1)^2-1$  incognite con  $m$  diversi  $\alpha_i$  possibili e  $(d+1)^2$  componenti della matrice  $\mathbf{H}$ . Pertanto, sono necessarie  $m = d + 2$  corrispondenze per determinare  $\mathbf{H}$  in modo univoco. A volte le corrispondenze formano una configurazione degenera, nel senso che non risulta possibile trovare un'unica soluzione per  $\mathbf{H}$  anche se  $m \geq d + 2$ .

Quando si hanno più di  $d + 2$  corrispondenze, il sistema non ha una soluzione generale a causa del rumore introdotto nelle corrispondenze. In questo caso diventa necessario cercare una soluzione ottima del sistema attraverso metodi statistici.

### 3.2.1 Stima di massima verosimiglianza

Il metodo statistico più adatto per la ricerca della soluzione ottima è la *stima di massima verosimiglianza*. Si consideri il caso  $d = 2$ , cioè si cerchi di stimare l'omografia a partire da sole due immagini. Si assuma che i punti delle due immagini abbiano distribuzione normale indipendente su ogni componente  $(u, v)$ , valori medi  $[u_i^{\hat{M}}, v_i^{\hat{M}}]^T$  e  $[u_i^{\hat{M}}, v_i^{\hat{M}}]^T$  e uguale varianza. A questo punto la stima dell'omografia si ottiene dalla minimizzazione della relazione (2.27) su  $9 + 2m$  variabili.

$$\min_{\mathbf{H}, u_i^O, v_i^O} \sum_{i=1}^m [(u_i^O - \hat{u}_i^O)^2 + (v_i^O - \hat{v}_i^O)^2 + (\frac{[u_i^O, v_i^O, 1] \mathbf{h}_1}{[u_i^O, v_i^O, 1] \mathbf{h}_3} - \hat{u}_i^{\hat{M}})^2 + (\frac{[u_i^O, v_i^O, 1] \mathbf{h}_2}{[u_i^O, v_i^O, 1] \mathbf{h}_3} - \hat{v}_i^{\hat{M}})^2]. \quad (2.27)$$

La matrice  $\mathbf{H}$  è scomposta nelle sue righe  $h_i$ . L'operazione di minimizzazione non è né lineare né convessa e tipicamente si trovano diversi minimi locali. Pertanto il problema è generalmente scomposto in due passi successivi: l'individuazione di una prima stima tramite la ricerca di un buon minimo locale e la successiva minimizzazione nell'intorno di questo punto.

### 3.2.2 Stima lineare

Per trovare una buona, ma non ottima, stima iniziale è possibile risolvere il sistema (2.26) utilizzando un metodo algebrico di soluzione per sistemi sovradeterminati. Questo metodo è noto in letteratura come *minimizzazione della distanza algebrica* (o semplicemente stima lineare).

Innanzitutto, è necessario rappresentare i punti corrispondenti nelle diverse immagini in coordinate omogenee:  $\mathbf{u} = [u \ v \ w]^T$ . Per trovare la soluzione è possibile sfruttare due artifici in modo da mantenere il problema in forma matriciale. In primo luogo si può eliminare  $\alpha$  dal sistema moltiplicando il primo membro dell'equazione per una matrice  $\mathbf{G}(\mathbf{u}^M)$  scelta "ad hoc" le cui righe siano ortogonali a  $\mathbf{u}^M$ . Con questa operazione si ha che  $\mathbf{G}(\mathbf{u}^M)\mathbf{u}^M = \mathbf{0}$  e quindi si ottiene l'equazione (2.28).

$$\mathbf{G}(\mathbf{u}^I)\mathbf{H}\mathbf{u} = \mathbf{0}. \quad (2.28)$$

Visto che i punti immagine sono bidimensionali, cioè  $w = 1$ , una buona scelta per la matrice  $\mathbf{G}(\mathbf{u})$  è

$$\mathbf{G}(\mathbf{u}) = \mathbf{G}([\mathbf{u}, \mathbf{v}, \mathbf{1}]^T) = \begin{bmatrix} 1 & 0 & -u \\ 0 & 1 & -v \end{bmatrix} = [\mathbf{I}] - \mathbf{u}]. \quad (2.29)$$

Il secondo artificio prevede di riorganizzare l'equazione (2.28) così da spostare l'incognita  $\mathbf{H}$  nel membro di destra. Per fare questo si può utilizzare l'identità  $\mathbf{A}\mathbf{B}\mathbf{c} = (\mathbf{c}^T \otimes \mathbf{A})\mathbf{b}$ , dove  $\mathbf{b}$  è il vettore costituito dagli elementi della matrice  $\mathbf{B}$  messi in colonna e  $\otimes$  rappresenta il prodotto di Kronecker tra matrici (<sup>1</sup>). Applicando l'identità, l'equazione da risolvere diventa è ora data da

$$\mathbf{G}(\mathbf{u}^M)\mathbf{H}\mathbf{u}^O = [\mathbf{u}^{O^T} \otimes \mathbf{G}(\mathbf{u}^M)]\mathbf{h} = \mathbf{0}, \quad (2.30)$$

dove  $\mathbf{h}$  è un vettore contenente le 9 componenti della matrice omografia  $\mathbf{H}$ :

$$\mathbf{h} = [h_{11} \ h_{21} \ \dots \ h_{23} \ h_{33}]^T. \quad (2.31)$$

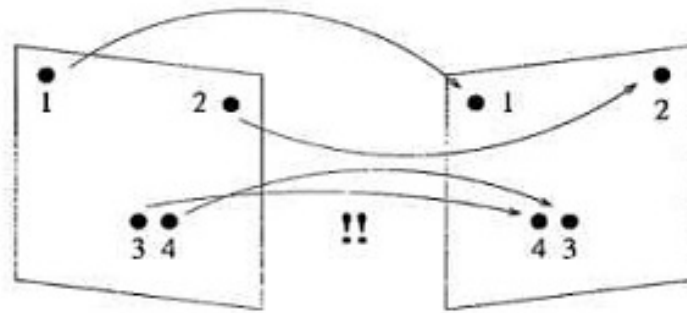
A questo punto la matrice  $\mathbf{H}$  viene trovata calcolando  $\mathbf{h}$  e riorganizzandone gli elementi. Si noti la differenza tra la dimensione del problema di massima verosimiglianza e il problema lineare. Mentre il primo ha  $9 + 2m$  variabili, il secondo ha solo 9 incognite: più  $m$  è grande e più la differenza è computazionalmente importante. Tuttavia, la stima di massima verosimiglianza fornisce sempre la soluzione ottima, mentre l'approssimazione lineare nella maggioranza dei casi si avvicina solamente all'ottimo, pertanto solitamente la stima lineare rappresenta il metodo più usato per avvicinarsi al minimo locale per l'ottimizzazione del problema di massima verosimiglianza.

---

<sup>1</sup>  $\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}$

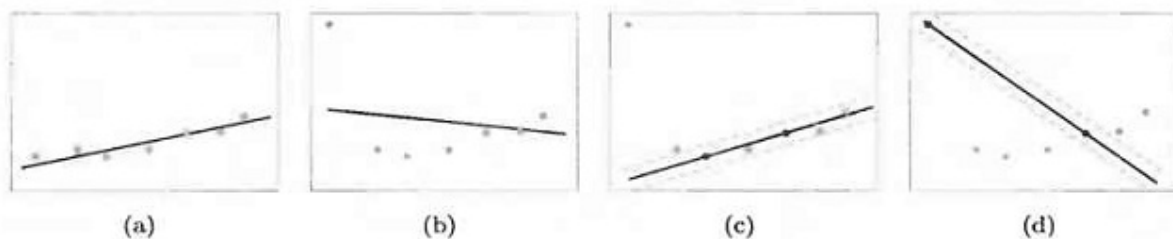
### 3.2.3 Robustezza della stima

La robustezza della stima dell'omografia dipende dalla precisione delle corrispondenze trovate. Assumendo che il vettore delle corrispondenze venga influenzato e corrotto da rumore gaussiano additivo, l'entità del disturbo può portare all'inversione di una corrispondenza con l'altra (figura 2.37). Se il numero di corrispondenze invertite è elevato, la stima risulta affetta da errore.



*Figura 2.37: Inversione di corrispondenze*

Un semplice esempio dell'effetto del rumore è costituito dal fitting di punti su linee tramite la ricerca della retta dei minimi quadrati. Se i punti sono affetti solo da rumore gaussiano, si ottiene che pochi punti non sono allineati e l'errore di allineamento è ridotto. Di conseguenza, la retta dei minimi quadrati rappresenta una stima ottimale della distribuzione dei punti. Al contrario, se il rumore non è gaussiano, i punti non sono più allineati e quindi la minimizzazione ai minimi quadrati non costituisce una buona stima della distribuzione dei punti nel piano (figura 2.38).



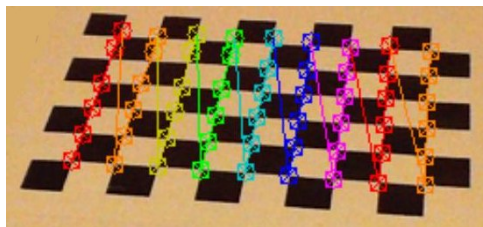
*Figura 2.38: Robustezza della stima dei minimi quadrati*

## 3.3 Pattern di calibrazione

Limitatamente al caso della calibrazione delle telecamere, l'identificazione delle corrispondenze tra più immagini è facilitata dall'utilizzo di oggetti con caratteristiche dell'immagine

facilmente individuabili, come scacchiere o griglie di pallini. In generale, infatti, è difficile individuare le corrispondenze su oggetti tridimensionali. Al contrario, un oggetto piano fortemente “texturizzato”, ovvero con molti tratti grafici, presenta molti punti per cui non è difficile trovare corrispondenze tra più immagini o col modello dell’oggetto. L’individuazione delle corrispondenze viene effettuata tramite algoritmi di *pattern matching*.

Uno tra i calibratori più utilizzati è una scacchiera. In questo caso il modello è costituito dal numero di righe e colonne di scacchi e dalla loro dimensione. Le corrispondenze da ricercare fanno riferimento ai punti d’intersezione tra gli scacchi, individuati solitamente tramite un algoritmo di *individuazione degli angoli* (*corner detection*) come mostrato in figura 2.39.



**Figura 2.39:** Individuazione angoli su pattern di calibrazione

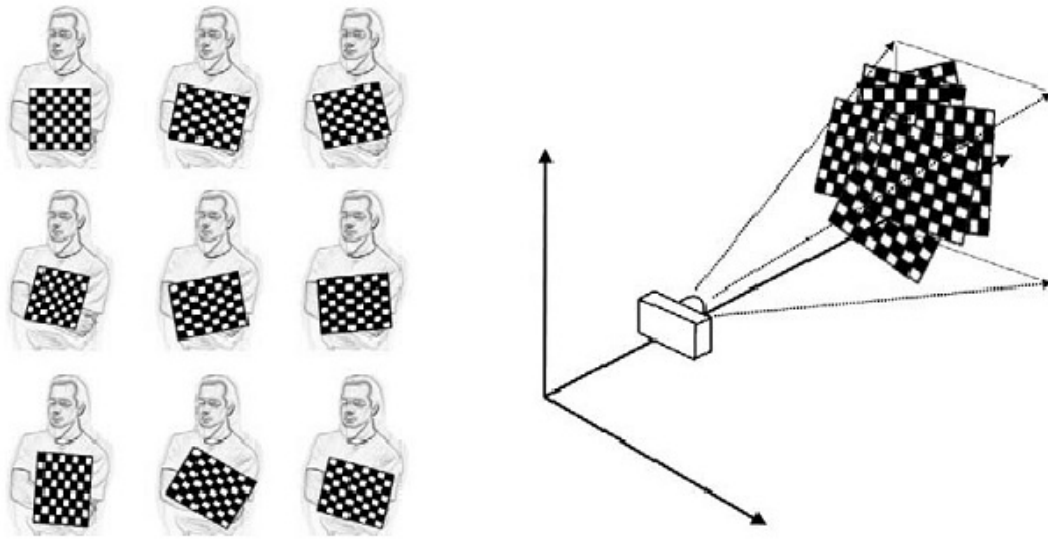
La procedura di calibrazione è riassunta in figura 2.40. Questa prevede di muovere la scacchiera nello spazio, in modo che essa venga mostrata alla telecamera da diverse angolazioni, distanze e su tutta l’ampiezza del piano immagine. Una volta raccolte le corrispondenze col modello su tutte le immagini acquisite, l’omografia è stimata tramite le metodologie descritte nel paragrafo 3.2.

Per finire, in figura 2.41 si mostra l’applicazione dei parametri estrinseci individuati per ridurre l’effetto della distorsione delle lenti. Infatti, si può notare come i punti appartenenti ad una linea, ma distorti su curve, tornano ad essere allineati su rette.

## 4 Conclusioni

In questo capitolo sono state presentate le varie tematiche coinvolte nel processo di formazione dell’immagine e ne è stata presentata la formulazione matematica più utilizzata per comprendere le trasformazioni prospettiche coinvolte nel passaggio da un sistema di riferimento tridimensionale ad un sistema planare.

Dall’analisi dei processi fisici coinvolti nella formazione di un’immagine, è possibile estrapolare una serie di caratteristiche e proprietà di cui si deve tener conto nella scelta



*Figura 2.40: Procedura di calibrazione tramite apposito pattern*



*Figura 2.41: Applicazione dei parametri estrinseci stimati per la riduzione della distorsione*

o nell'analisi delle prestazioni di un dispositivo di visione. In particolare, si possono considerare:

- *Risoluzione*. Data in pixel, limita il massimo passo di campionamento spaziale dell'immagine.
- *Framerate*. Costituisce la frequenza base di aggiornamento delle immagini. Fornisce informazioni riguardanti le prestazioni dinamiche del dispositivo di visione, ovvero riguardanti la capacità dello stesso di tenere traccia ed estrarre informazioni da scene che evolvono nel tempo.
- *Shutter*. Regola la quantità di luce che raggiunge il sensore di visione. La sua regolazione è fondamentale per ottenere immagini con un'illuminazione sufficiente ad evidenziare le caratteristiche degli oggetti che si vogliono individuare. Lo shutter

influisce anche sul tempo d'acquisizione delle immagini: laddove sia necessario un tempo d'esposizione più lungo, è logico aspettarsi una risposta più lenta a variazioni all'interno della scena.

- *Guadagno e/o apertura obbiettivo.* Insieme alla regolazione dello shutter è fondamentale per ottenere la corretta illuminazione dell'immagine. Dalla corretta configurazione della coppia di parametri formata da shutter e guadagno dipende il rapporto segnale-rumore dell'immagine.
- *Formato delle immagini e interfaccia di comunicazione.* Al tempo d'acquisizione delle immagini è necessario aggiungere il tempo di attraversamento del canale di comunicazione. Questo dipende dall'interfaccia di comunicazione scelta, ma anche dalla dimensione, tipologia (colore o scala di grigi) e formato delle immagini.
- *Distanza focale.* Tutte le caratteristiche elencate fanno riferimento al solo sensore di acquisizione immagine e permettono la regolazione della qualità e velocità d'acquisizione delle immagini. La distanza focale, collegata alla scelta del giusto obbiettivo, regola invece la dimensione fisica della scena ripresa. Questa proprietà è alla base della formulazione matematica del modello della telecamera.

A partire dalle immagini, è possibile effettuare una doppia tipologia di elaborazione: (i) limitandosi all'elaborazione a livello dei pixel, ovvero un'elaborazione 2D, oppure (ii) cercare di ricostruire la posa di un oggetto all'interno del sistema di riferimento tridimensionale. Per questo secondo tipo di elaborazione, un modello matematico che regoli le trasformazioni di coordinate da un piano all'altro è necessario. Inoltre, si è visto come modellizzare gli effetti di distorsione introdotti dall'utilizzo di lenti. Questo è il primo esempio di non-linearità da analizzare nell'elaborazione delle immagini.

Una stima dei parametri intrinseci della telecamera a partire dalle proprietà fisiche del sensore e dell'obbiettivo non sempre dà i risultati sperati. Inoltre, per quanto concerne i parametri estrinseci, non è possibile una stima a partire dai dati fisici delle lenti utilizzate nelle ottiche. Per tale motivo, la stima dei parametri del modello viene effettuata tramite una procedura di identificazione che utilizza le immagini riprese dallo stesso sistema che si intende modellizzare: la calibrazione. Nell'analizzare le procedure di calibrazione, è stato introdotto il concetto di omografia e stima della trasformazione omografica, i cui metodi sono utilizzabili anche per la ricostruzione della posa 3D di un oggetto noto a partire da un'immagine bidimensionale che lo contiene.



# Capitolo 3

## Tecniche di Elaborazione di Immagini 2D

### Indice

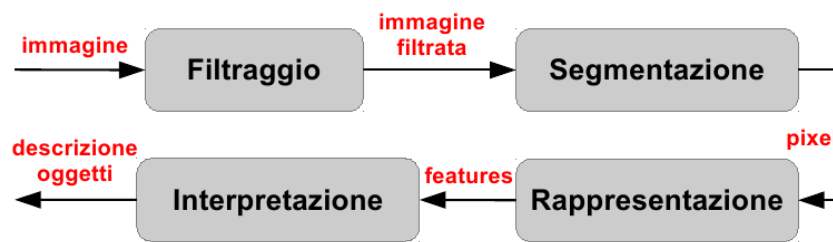
---

1	Segmentazione . . . . .	<b>70</b>
1.1	Ricerca di aree . . . . .	70
1.2	Rilevazione di contorni . . . . .	77
1.3	Punti d'interesse (keypoint) . . . . .	89
2	Descrizione . . . . .	<b>92</b>
2.1	Istogrammi . . . . .	93
2.2	Mean shift . . . . .	95
2.3	Momenti e descrittori geometrici . . . . .	97
2.4	Descrittori per contorni . . . . .	101
3	Interpretazione . . . . .	<b>105</b>
3.1	Template Matching . . . . .	107
3.2	Trasformata di Hough . . . . .	109
3.3	Stima della posa da modello 3D . . . . .	113
4	Finestramento . . . . .	<b>119</b>
4.1	Descrizione della finestra . . . . .	120
4.2	Calcolo delle feature locali . . . . .	121
5	Conclusioni . . . . .	<b>123</b>

---

In riferimento alla figura 1.2, si può dire che il cuore vero e proprio del sistema di visione sia costituito dalla componente dedicata all'elaborazione delle immagini. Infatti,

è solo tramite questo elemento che avviene l'estrazione delle informazioni dalle immagini al fine di convertirne il contenuto in un'informazione utilizzabile da altri componenti del sistema. Tale informazione, a differenza di quella ottenibile dai sensori tipicamente utilizzati nell'ambito dell'automazione industriale, può essere molto ricca e varia. Pertanto, l'elaborazione delle immagini non si limita al condizionamento ed analisi di un solo segnale, ma è più complessa e computazionalmente più onerosa. In figura 3.1 vengono elencate le fasi in cui questo processo può essere suddiviso.



**Figura 3.1:** Sequenza di operazioni per l'elaborazione delle immagini

In breve, queste fasi sono:

- *Filtraggio e pre-processing.* Rientrano in questa categoria tutte le operazioni che permettono di ottenere un'immagine idonea all'estrazione delle caratteristiche desiderate. Esempi di operazioni molto utilizzate sono le trasformazioni dello spazio colore e i filtri per l'eliminazione del rumore.
- *Segmentazione.* È la fase che permette di isolare solo i pixel appartenenti agli oggetti d'interesse della scena. Solitamente, il risultato di questa elaborazione è un'immagine *binaria*, ovvero con sfondo nero e pixel d'interesse bianchi. Di conseguenza, come la fase di filtraggio, anche la segmentazione è un'operazione da immagine a immagine.
- *Rappresentazione.* L'insieme di pixel individuati tramite segmentazione non costituisce ancora un'informazione sufficiente a descrivere la scena inquadrata, il cui ottenimento è compito della fase successiva. Tale informazione ha spesso una formulazione matematica ed è tipicamente costituita da un insieme di caratteristiche dell'immagine, dette *feature*. Il nome "rappresentazione" deriva dal fatto che essa effettua proprio una trasformazione dallo spazio dei pixel ad una rappresentazione matematica della scena.
- *Interpretazione.* Per alcune applicazioni l'elaborazione dell'immagine termina con la rappresentazione della scena. Se però è necessaria una comprensione più profonda dell'informazione contenuta nelle immagini, è necessaria una fase di interpretazione

delle feature estratte. Un tipico esempio di operazione svolta in questa fase è il *pattern matching* in cui si cerca di capire se le feature individuate descrivono un particolare tipo di oggetto o forma ricercata. In uscita si ha una descrizione della scena a livello più profondo, generalmente costituito dall'insieme di oggetti individuati all'interno della scena e la loro relativa posa.

L'elaborazione delle immagini viene affrontata, in questo capitolo come nel proseguo della tesi, con lo scopo di mostrare tecniche e modalità utilizzabili in applicazioni di *guida robot*, ovvero con lo scopo di individuare determinati oggetti nelle immagini e calcolarne la *posa* (posizione e orientamento) nello spazio di lavoro del manipolatore.

Praticamente tutti gli algoritmi presentati in questo capitolo trovano implementazione nella libreria software open source più utilizzata per lo sviluppo di applicazioni di visione artificiale: le *openCv*. Il libro [7] contiene la descrizione di tutti i moduli offerti dalla libreria e la relativa descrizione della formulazione e composizione degli algoritmi implementati. Per uno studio più approfondito e rigoroso delle tematiche di visione artificiale, l'altra fonte utilizzata è [61].

Gli articoli trovati in letteratura hanno come scopo principale la presentazione dell'applicazione di un determinato algoritmo al fine di arrivare alla soluzione di uno specifico problema di riconoscimento di oggetti. In particolare, si possono classificare tali algoritmi in funzione delle natura dei descrittori degli oggetti, ovvero contorni ([35], [50]), punti ([38], [8]), invarianti locali ([65]) e proprietà geometriche ([54], [20], [36]). Un approccio comunemente usato è inoltre costituito dall'utilizzo di tecniche di integrazione di caratteristiche diverse (*feature fusion*, [63]).

Tra le altre fonti utilizzate per la ricerca degli algoritmi di analisi delle immagini 2D si citano le seguenti dispense per corsi universitari: [19], [27], [39] e [49].

In questo capitolo viene fornita una panoramica sulle classiche tecniche di elaborazione delle immagini 2D. La trattazione segue esattamente lo stesso flusso di operazioni dell'elaborazione immagini: tralasciando il filtraggio, viene innanzitutto analizzata la segmentazione (1), per presentare successivamente la rappresentazione matematica delle caratteristiche dell'immagine (2) al fine di arrivare ad un'interpretazione della scena osservata (3). Per quanto riguarda la scelta degli algoritmi mostrati, è stata effettuata una ricerca degli algoritmi non particolarmente onerosi dal punto di vista computazionale e quindi utilizzabili in applicazioni real-time. Infine, viene dedicato un paragrafo (4) per alcune considerazioni sulle tecniche di *finestramento* delle immagini che permettono

di ridurre l'onere computazionale dell'elaborazione. Per concludere, si aggiungono delle considerazioni finali (5) sulle metodologie presentate.

## 1 Segmentazione

La *segmentazione* ha come obiettivo l'estrazione dall'immagine dei soli pixel utili a classificare e descrivere la scena inquadrata. I passaggi da eseguire sono due: la *classificazione* e *presentazione* dei pixel.

Nel primo passo i pixel vengono suddivisi in sottoinsiemi in funzione della loro appartenenza ad una determinata regione dell'immagine. Gli algoritmi di segmentazione vengono classificati in funzione del tipo di regioni considerate. In generale, si distingue tra segmentazione basata su *area*, *contorni* o *punti*. Gli algoritmi basati sulla ricerca di aree hanno come obiettivo la suddivisione dell'immagine in sottoregioni e si basano fondamentalmente sul filtraggio per *soglia* e sulla rappresentazione statistica (*istogrammi*) del contenuto dell'immagine. Nel tipo di segmentazione basata su contorni, invece, l'operazione di filtraggio è più complessa e ha come obiettivo l'individuazione dei *gradienti* di luminosità dell'immagine. In entrambi questi tipi di operazioni si può distinguere tra una fase di *filtraggio* e un'altra di *caratterizzazione geometrica* dei punti individuati. Il terzo tipo di segmentazione analizzato, basata sull'individuazione di punti d'interesse, al contrario, prevede unicamente la fase di filtraggio perché la descrizione della scena è costituita direttamente dall'insieme di punti individuati.

La fase di presentazione, invece, prevede la riorganizzazione dei pixel individuati in strutture dati quali vettori, grafi ed alberi, più adatte per le successive fasi di elaborazioni. L'approccio più semplice è quello di marcare con un colore specifico i pixel individuati direttamente sull'immagine.

### 1.1 Ricerca di aree

Nell'ambito della visione artificiale, la definizione di *area* differisce da quella utilizzata nella vita quotidiana. Per area, infatti, si intende una regione dell'immagine che possiede una determinata caratteristica. La caratteristica più semplice su cui suddividere l'immagine è la luminosità dei pixel: appartengono ad una stessa regione tutti i pixel con un valore di luminosità compreso in uno specifico intervallo. Pertanto, non è detto che l'area sia formata interamente da pixel contigui, ma spesso l'obiettivo della segmentazione è proprio

quello di individuare le aree nel vero senso della parola, ovvero nel senso di sottoinsiemi disgiunti di pixel. Al contrario, una regione di pixel congiunti viene chiamata *blob* e, come si vedrà nel paragrafo 2, l'analisi di questi oggetti permette una semplificazione del compito di descrizione poiché è possibile calcolarne certe proprietà collegate direttamente alle caratteristiche geometriche delle zone trovate.

### 1.1.1 Sogliatura

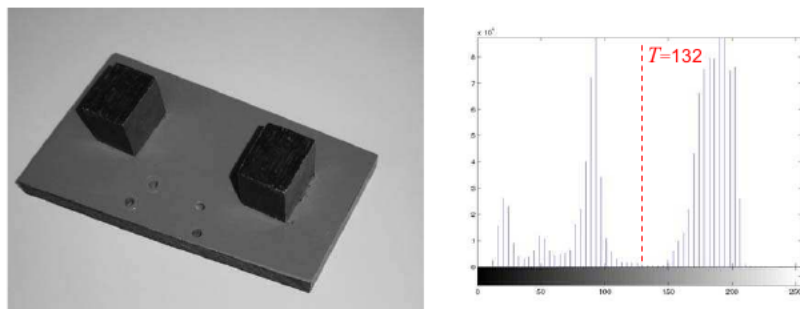
L'operazione più semplice per suddividere un'immagine in diverse regioni è la *sogliatura*. Questa procedura prevede la valutazione, per ogni pixel dell'immagine, della semplice condizione (3.1). Se il valore di luminosità del pixel è tale da rispettare tale condizione, esso viene marcato come appartenente alla sottoregione desiderata, ovvero gli viene assegnato il valore di luminosità massimo. Generalmente, l'obiettivo principale delle operazioni di sogliatura è la sottrazione dei pixel appartenenti allo sfondo.

$$\begin{cases} p(x, y) = 0 & , \quad \text{se } p(x, y) < T \\ p(x, y) = 255 & , \quad \text{se } p(x, y) \geq T \end{cases} \quad (3.1)$$

Il problema, quindi, si riduce essenzialmente alla scelta della soglia  $T$ . Le strade da seguire per la scelta del parametro  $T$  sono principalmente due:

- Utilizzo di metodi per l'ottimizzazione della soglia;
- Scelta di  $T$  in modo empirico, ad esempio tramite procedura di taratura o calibrazione.

Ad esempio, in figura 3.2 si mostra come il valore di soglia possa essere preso in corrispondenza di concavità nell'istogramma (vedi 2.1) che descrive la distribuzione statistica della luminosità dell'immagine.



**Figura 3.2:** Scelta della soglia per segmentazione basata su sogliatura

### Algoritmo per scelta del valore di soglia ottimo.

1. Si assuma di non avere alcuna conoscenza dell'esatta posizione degli oggetti all'interno dell'immagine. Si consideri a priori che i quattro angoli dell'immagine appartengano allo sfondo. Quest'ultima condizione non è particolarmente stringente perché in un'immagine che possa essere considerata valida per l'estrazione delle informazioni gli oggetti d'interesse devono essere posti al centro dell'immagine stessa. Il valore medio dei pixel nei quattro angoli rappresenta il valore di partenza per la stima della luminosità dello sfondo.
2. A partire dal secondo pixel della prima riga, stimarne l'appartenenza all'oggetto o allo sfondo, in funzione della soglia di sfondo stimata. Al passo  $t$  si calcolino  $\mu_B^t$  e  $\mu_O^t$  come media dei valori di intensità dei pixel appartenenti rispettivamente allo sfondo e agli oggetti:

$$\begin{cases} \mu_B^t = \frac{\sum_{(i,j) \in background} f(i,j)}{\sum pixel_{background}} \\ \mu_O^t = \frac{\sum_{(i,j) \in objects} f(i,j)}{\sum pixel_{object}} \end{cases} . \quad (3.2)$$

3. Il valore di soglia da utilizzare per il pixel successivo, è dato dalla media del valore medio d'intensità dello sfondo e degli oggetti:

$$T(t+1) = \frac{\mu_B^t + \mu_O^t}{2} . \quad (3.3)$$

4. I passi 2 e 3 vanno iterati fino a quando il valore di soglia stimato non subisce variazioni significative, ovvero

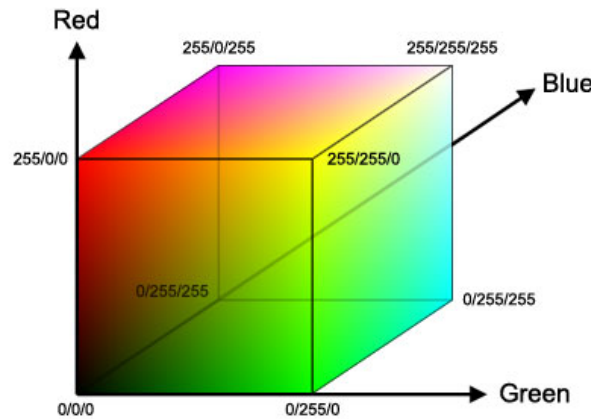
$$T(t+1) \approx T(t) . \quad (3.4)$$

**Immagini a più canali.** Fino ad ora, parlando di funzione di luminosità dei pixel, si è sempre fatto riferimento ad immagini ad un singolo canale, ovvero immagini in scala di grigi. L'utilizzo di immagini a più canali, ovvero in grado di rappresentare l'informazione relativa ai colori, può facilitare l'operazione di sogliatura in applicazioni in cui la separazione delle regioni può essere eseguita semplicemente distinguendo le aree di diverso colore. Tuttavia, nel passare da un'immagine a singolo canale ad una a più canali, l'operazione di sogliatura si complica perché la scelta della soglia corretta deve essere effettuata separatamente per ogni singolo canale. Per di più, l'informazione di colore, generalmente, non è immediatamente disponibile nella rappresentazione a più canali, ma dipende dalla fusione delle informazioni dei singoli canali.

Un esempio di rappresentazione di colori in cui il tipo di colore deriva dalla fusione di più informazioni è lo spazio di colori *RGB* (*Red*, *Green*, *Blue*). Utilizzando questa rappresentazione, il colore viene scomposto, attraverso un apposito filtraggio, in tre colori base, corrispondenti a forme d'onda di periodo fissato. Nel caso della rappresentazione *RGB* si utilizza:

- Rosso, con una lunghezza d'onda di 700 *nm*;
- Verde, con una lunghezza d'onda di 546.1 *nm*;
- Blu, con una lunghezza d'onda di 455.8 *nm*.

Il modello RGB è additivo: unendo i tre colori con la loro intensità massima si ottiene il bianco (tutta la luce viene riflessa). Tutte le altre combinazioni permettono di ottenere l'intero spettro della radiazione luminosa. Tali componenti si distribuiscono su uno spazio *cubico*, come mostrato in figura 3.3.



**Figura 3.3:** Spazio di colore RGB

Per quanto visto, lo spazio colori RGB non è adatto ad una suddivisione delle regioni di colore diverso. Un piano di colori alternativo e più adatto a questo scopo è il piano *HSV* (*Hue*, *Saturation*, *Value*). Il modello HSV è particolarmente orientato alla prospettiva umana, essendo basato sulla percezione che si ha di un colore in termini di tinta, sfumatura e tono. In teoria dei colori, una *tonalità* è un colore puro, ovvero caratterizzato da una singola lunghezza d'onda all'interno dello spettro della luce visibile. Nella rappresentazione dei colori nello spazio RGB, la tonalità può essere pensata come un angolo  $\phi$ . Se  $R$ ,  $G$  e  $B$  sono le coordinate di colore nello spazio RGB (in una scala da 0 a 1),  $\mu$  è la luminosità e  $\sigma$  la saturazione, la tonalità  $\phi$  è data da:

$$\phi = \arccos\left(\frac{R - \mu}{\sigma\sqrt{2}}\right). \quad (3.5)$$

Pertanto,  $\phi = 0^\circ$  corrisponde al rosso,  $\phi = 120^\circ$  al blu, e  $\phi = 240^\circ$  al verde. La relazione tra spazio colore HSV e RGB è data da:

$$\begin{cases} R = \mu + \sigma\sqrt{2} \cos \phi \\ G = \mu + \sigma\sqrt{2} \cos(\phi + 240^\circ) \\ B = \mu + \sigma\sqrt{2} \cos(\phi + 120^\circ) \end{cases} \quad (3.6)$$

Nella teoria dei colori e in discipline correlate come la fotografia, la saturazione è l'intensità di una specifica tonalità. Una tinta molto satura ha un colore vivido e squillante; al diminuire della saturazione, il colore diventa più debole e tendente al grigio. Se la saturazione viene completamente annullata, il colore si trasforma in una tonalità di grigio. Di conseguenza, per esempio, la desaturazione di una fotografia digitalizzata è una delle tecniche con cui si può trasformare un'immagine a colori in una in bianco e nero.

La saturazione di un colore dipende dall'intensità della luce e dallo spettro di lunghezze d'onda su cui viene distribuita. Il colore puro si ottiene quando la luce è su una singola lunghezza d'onda (come nel caso di una sorgente laser ideale). Nell'ambito dell'elaborazione delle immagini, la luminosità è considerata sinonimo di brillantezza (*brightness*) e in questo senso è la quantità totale di luce che una sorgente luminosa appare emettere o che una superficie appare riflettere.

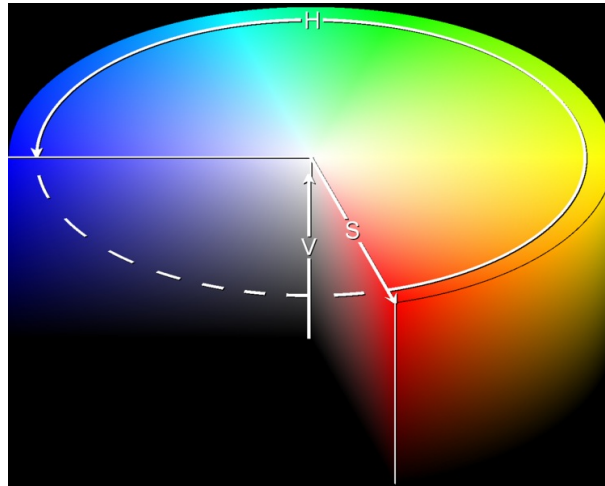
Il sistema di coordinate risultante è lo spazio cilindrico di figura 3.4, dove:

- La tonalità  $H$  viene misurata da un angolo intorno all'asse verticale, con il rosso a  $0^\circ$ , il verde a  $120^\circ$  e il blu a  $240^\circ$ ;
- L'altezza del cilindro rappresenta la luminosità ( $V$ ) con lo zero che rappresenta il nero e l'uno il bianco;
- La saturazione ( $S$ ), invece, ha valore zero sui punti appartenenti all'asse del cilindro e aumenta radialmente avvicinandosi alla superficie esterna del cilindro.

### 1.1.2 Sottrazione di sfondo

L'obiettivo ultimo della sogliatura è la separazione degli oggetti in primo piano dallo sfondo della scena inquadrata. Tuttavia, il corretto risultato dell'operazione di sogliatura è strettamente dipendente dalla scelta del valore di soglia  $T$ . Inoltre, la robustezza dell'operazione viene meno in presenza di sfondi complessi (non uniformi) o con componenti di luminosità paragonabili a quella degli oggetti ricercati. Per ovviare a questi problemi si ricorre, ove possibile, ad una tecnica che sfrutta una conoscenza a priori dello sfondo





*Figura 3.4: Spazio di colore HSV*

della scena. Se tale conoscenza è espressa da un'immagine dello sfondo senza oggetti in primo piano, l'operazione più semplice per suddividere la scena nelle due componenti fondamentali è la *sottrazione dello sfondo*: scorrendo ogni pixel dell'immagine, il pixel risultante è dato dalla sottrazione dello sfondo dall'immagine osservata. La segmentazione ha successo solamente se si verificano certe condizioni, tra cui:

- Telecamera non in movimento. In caso contrario lo sfondo subisce variazioni tali da renderlo diverso dallo sfondo modellato in partenza.
- Sfondo costante. Sulla tinta e luminosità del colore di sfondo possono influire vari parametri, tra i quali riveste notevole importanza l'illuminazione dell'ambiente esterno. Per ottenere risultati migliori, sfondo e oggetti devono avere una composizione di colore la più diversa possibile (il caso ideale è quello di oggetti chiari su sfondo scuro o viceversa).

Il concetto di sfondo può variare anche notevolmente a seconda dell'applicazione in oggetto di studio e quindi non è possibile fornirne una definizione valida in termini assoluti. Una prima definizione è quella per cui lo sfondo deve contenere tutto ciò che è statico nel tempo oppure che si può muovere periodicamente a intervalli di tempo fissati. Come già anticipato nei prerequisiti visti in precedenza, per ottenere buoni risultati anche le condizioni ambientali (soprattutto la luminosità) devono rimanere pressoché costanti. Di conseguenza l'ambiente influisce sulla scelta dell'algoritmo di sottrazione da utilizzare: nel caso di ambienti con luce controllata (ambienti chiusi o con utilizzo di illuminatori) può essere sufficiente limitarsi ad un'operazione di differenza assoluta tra pixel, mentre nel caso di ambienti esterni questa non può più bastare. In questo secondo caso si passa all'utilizzo

di algoritmi adattativi, in cui lo sfondo non è più un parametro costante, ma variabile nel tempo.

Il metodo più semplice di sottrazione dello sfondo prevede semplicemente l'ottenimento di una nuova immagine a partire dalla differenza assoluta tra la luminosità dei pixel dell'immagine in analisi  $i(x, y)$  e la luminosità degli stessi pixel in un'immagine di sfondo precedentemente acquisita  $b(x, y)$ :

$$d(x, y) = |i(x, y) - b(x, y)| . \quad (3.7)$$

Ad ogni modo, questa semplice operazione può non bastare perché anche nelle condizioni le più stabili possibili alcuni pixel possono avere intensità diversa da quella originale. Le cause di questo fenomeno possono essere svariate:

- Leggere modifiche della luce ambientale;
- Influenza della presenza degli oggetti sullo sfondo: ombre, ...;
- Leggere variazioni dell'apertura e del tempo d'esposizione della telecamera.

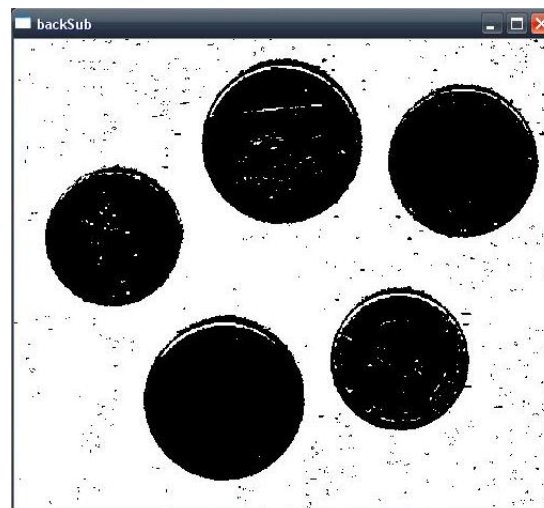
Queste piccole variazioni portano all'insorgenza, nell'immagine differenza, del cosiddetto *rumore granulare*. Questo si presenta come piccoli raggruppamenti di pixel che, pur facendo parte dello sfondo, sono visti come appartenenti agli oggetti da analizzare.

Inoltre, il risultato della segmentazione deve essere un'immagine binaria (*silhouette*) costituita, ad esempio, da oggetti neri su sfondo bianco o viceversa. Al contrario, il valore  $d(x, y)$  è un valore qualsiasi compreso tra il massimo e il minimo valore di luminosità dell'immagine. Di conseguenza, l'operazione di differenza delle immagini è sempre seguita da un'operazione di *sogliatura*, come mostrato in figura 3.5.

Il concetto di *sottrazione dello sfondo* è alla base degli algoritmi di *rilevazione del movimento* (o *optical flow*). Nel caso il movimento sia dato dagli oggetti, ovvero con telecamera e sfondo statici, il moto degli oggetti può essere tracciato aggiornando il modello dello sfondo da sottrarre. L'algoritmo può essere suddiviso nelle seguenti fasi:

1. Inizializzazione. La prima differenza di immagini viene ottenuta sottraendo uno sfondo noto a priori, oppure a partire dalla seconda immagine acquisita, dove lo sfondo è rappresentato dalla prima immagine.
2. Aggiornamento dello sfondo. L'immagine di sfondo da utilizzare al passo  $t + 1$  è l'immagine acquisita al passo  $t$ .
3. Tracciamento. L'immagine differenza è ottenuta sottraendo lo sfondo acquisito al tempo  $t - 1$  dall'immagine acquisita al tempo  $t$ .

Questa tecnica è utilizzata, ad esempio, per le applicazioni di videosorveglianza.

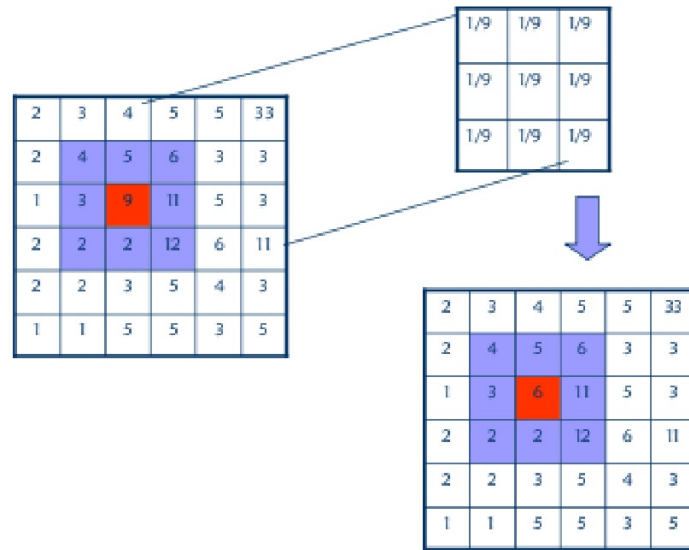
(a) *Immagine.*(b) *Sfondo.*(c) *Silhouette.***Figura 3.5:** Esempio di sottrazione dello sfondo

## 1.2 Rilevazione di contorni

La rilevazione di contorni all'interno di un'immagine in scala di grigi è un'operazione più complessa rispetto alla sogliatura. Infatti, gli operatori da applicare sono *locali* e non puntuali come nei casi visti fino ad adesso. Per operatore puntuale si intende una trasformazione che interviene direttamente a livello di pixel, ovvero permette di ottenere un valore di luminosità in uscita tramite una trasformazione lineare del valore di luminosità di partenza. Con l'utilizzo degli operatori locali, al contrario, il valore in uscita dipende dal valore di partenza del pixel interessato e dei pixel nel suo *vicinato*, ovvero degli 8 pixel che lo circondano. Tipici operatori locali sono i *filtri spaziali*.

Un *filtro spaziale*, detto anche *maschera* o *kernel*, è una regione rettangolare caratterizzata da un'operazione predefinita. L'operazione viene eseguita sui pixel dell'immagine

corrispondenti alla suddetta regione e le modifiche indotte dal filtro generano la cosiddetta immagine filtrata. Il filtro opera sovrapponendosi all'immagine partendo generalmente dall'angolo in alto a sinistra. Quando il punto centrale (*anchor point*) coincide con il pixel da modificare, quest'ultimo verrà trasformato nel nuovo pixel dell'immagine di output. Il kernel "spazzola" l'intera immagine spostandosi da sinistra verso destra e dall'alto verso il basso, modificando i valori del pixel centrale nel modo previsto dalla funzione di filtraggio. L'onerosità computazionale di un filtro dipende quindi dalla dimensione della maschera di filtraggio e dalla risoluzione dell'immagine. Il *filtro della media* di figura 3.8, ad esempio, definisce l'operazione di media su nove pixel; la media viene memorizzata nel punto centrale della maschera di filtraggio.



**Figura 3.6:** Esempio di filtro spaziale

Il filtro viene applicato mediante un procedimento di *convoluzione*, definita come

$$g(x, y) = \sum_{s=-a}^{s=a} \sum_{t=-b}^{t=b} w(s, t) f(x + s, y + t) , \quad (3.8)$$

dove  $f(x, y)$  è la funzione di luminosità del pixel dell'immagine originale,  $w$  è il kernel di convoluzione  $2a \times 2b$ , mentre  $g(x, y)$  è il valore di luminosità del pixel ottenuto con il filtraggio. L'operazione di convoluzione viene eseguita su tutti i pixel dell'immagine. Per i pixel posti sulla cornice dell'immagine, per i quali non è disponibile un vicinato di pixel completo, la soluzione consiste nell'inserimento di  $2a$  o  $2b$  zeri sui lati dell'immagine. La complessità computazionale della convoluzione è elevata: un kernel di dimensioni  $m \times m$  richiede  $m^2$  somme e altrettanti prodotti per ogni pixel dell'immagine. In genere, per una

maschera di dimensioni dispari e con due assi di simmetria si può snellire l'onere computazionale tramite tecniche di separazione, applicando cioè due filtri monodimensionali in serie invece di un unico filtro bidimensionale.

Due dei filtri più utilizzati nell'elaborazione immagini sono il *filtro passa-basso* (detto anche *filtro della media*) e *passa-alto*. Il filtro passa-basso (figura 3.7) esegue una media su un vicinato di pixel, andando a filtrare le componenti in alta frequenza dell'immagine a favore delle componenti continue. Pertanto, è un filtro utilizzato per la rimozione del rumore granulare. L'effetto indesiderato del filtro passa-basso è una sfocatura dell'immagine. Essendo i coefficienti unitari, l'operazione non richiede alcuna moltiplicazione: per un filtro di dimensione  $3 \times 3$  è sufficiente sommare le intensità dei pixel 8-vicini dell'immagine e dividere per 9. Per tale motivo, tale filtro è un esempio di *filtro spaziale lineare*.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(a) *Maschera del filtro.*



(b) *Immagine di partenza.*

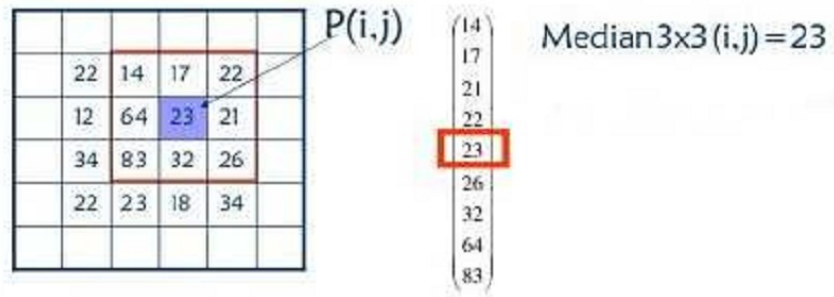


(c) *Immagine filtrata.*

**Figura 3.7:** *Filtro passa-basso*

Un *filtro spaziale non lineare* esegue operazioni più complesse rispetto alle semplici operazioni lineari. Il filtro non lineare maggiormente usato è il *filtro mediano* (figura 3.8), il quale sostituisce ad ogni pixel dell'immagine il valore mediano dei suoi vicini, incluso il pixel stesso. Ha quindi il vantaggio di non introdurre nuovi valori di grigio nell'immagine elaborata. Il valore è calcolato ordinando tutti i pixel dell'intorno prescelto e sostituendo al pixel in esame il valore centrale dell'insieme ordinato. In particolare, se l'intorno contiene un numero pari di pixel si prende la media dei due valori centrali. Questo filtro viene

applicato principalmente per ridurre il rumore di tipo impulsivo (“salt & pepper”), senza causare la sfocatura dell’immagine tipico del filtro passa-basso.



(a) Maschera del filtro.



(b) Immagine di partenza.



(c) Immagine filtrata.

**Figura 3.8:** Filtro mediano

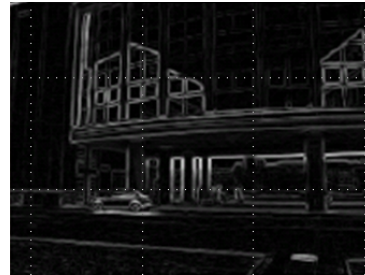
Il filtro passa-alto di figura 3.9, al contrario del filtro passa-basso, permette di accentuare le componenti di discontinuità delle immagini. I *contorni* degli oggetti sono costituiti da quei punti che ne permettono la separazione dallo sfondo, andando per l'appunto ad introdurre una discontinuità nei valori di luminosità dell'immagine. Pertanto, si può concludere che il filtro passa-alto è il più semplice filtro per l'individuazione dei contorni. Un semplice filtro  $3 \times 3$  è costituito da 4 fattori moltiplicativi  $w(s, t)$ , posti nei 4 punti cardinali della maschera di filtraggio, il cui valore è scelto in modo che la loro somma col valore del fattore applicativo del pixel centrale sia nulla.

### 1.2.1 Rilevatori di bordi

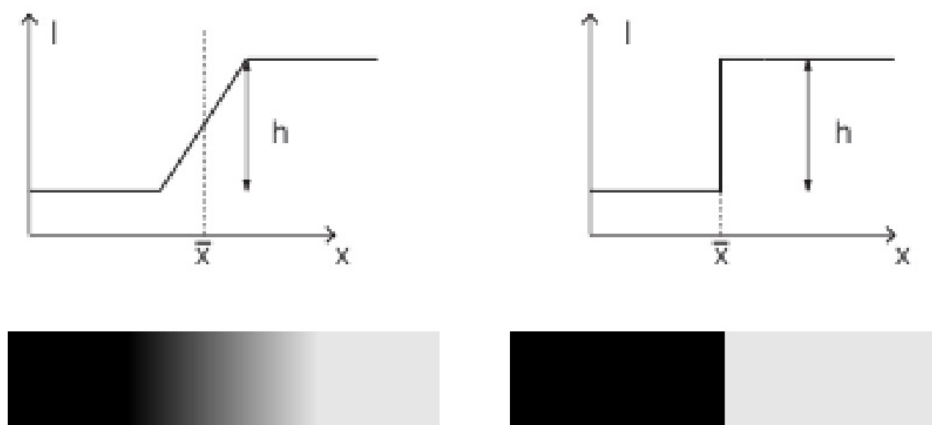
Per *bordo* (*edge*) si intende il confine tra un oggetto e lo sfondo, o il confine tra oggetti sovrapposti. Questo significa che, qualora fosse possibile identificare in maniera accurata tutti i contorni in un'immagine, allora tutti gli oggetti contenuti in essa potrebbero venire localizzati e proprietà di base come area, perimetro e forma essere calcolate.



0	-1	0
-1	4	-1
0	-1	0

(a) *Maschera del filtro.*(b) *Immagine di partenza.*(c) *Immagine filtrata.***Figura 3.9:** *Filtro passa-alto*

Esistono diverse possibili definizioni per i tipi di bordo, ognuna delle quali si applica in circostanze specifiche. Una delle definizioni più comunemente usate è quella di *bordo a gradino ideale*. In questo caso il bordo rappresenta semplicemente un cambiamento netto di intensità da nero a bianco che occorre in una posizione specifica. In realtà, considerando un'immagine ad un singolo canale, in corrispondenza di un bordo si ha una variazione graduale di intensità, quindi a *rampa*, come mostrato in figura 3.10. Maggiore è la pendenza della rampa di variazione di intensità, più semplice è la localizzazione del bordo.

**Figura 3.10:** *Definizione matematica di bordo*

Vi sono tuttavia delle complicazioni nell'applicazione di questa definizione alla ricerca

dei bordi all'interno di un'immagine. La prima difficoltà è dovuta al processo di campionamento spaziale dell'immagine: è poco probabile che un'immagine venga campionata in maniera tale che tutti i bordi contenuti in essa corrispondano esattamente a un bordo netto di 1 o 2 pixel. In realtà, il cambiamento di intensità può estendersi su un numero non precisato di pixel; in questa situazione si può prendere come posizione del bordo il centro della rampa che porta l'intensità da un livello basso a uno alto.

La seconda complicazione è rappresentata dalla presenza di *rumore*, che può essere causato da molti fattori, tra cui il tipo del dispositivo di acquisizione, la lente utilizzata, i cambiamenti di luminosità, il movimento, la temperatura, gli effetti atmosferici, la polvere, .... A causa del rumore diventa altamente improbabile che due pixel cui dovrebbe corrispondere la stessa intensità di grigio abbiano effettivamente lo stesso livello nell'immagine. Essendo il rumore un fenomeno casuale esso può essere caratterizzato e descritto solo statisticamente, così da introdurre una variazione casuale del livello di intensità di pixel in pixel, cosicché le linee sinuose e le rampe che costituirebbero bordi ideali non vengono mai incontrati nella realtà.

Esistono essenzialmente due tipi comuni di operatori per la localizzazione dei bordi. Il primo tipo racchiude gli *operatori derivativi*, cioè progettati per identificare le locazioni in cui vi sono grandi cambiamenti di intensità. Il secondo tipo richiama gli schemi di *template matching*, dove i contorni sono modellati da piccole immagini che esibiscono le proprietà astratte di bordi ideali.

Tipicamente un task di estrazione dei bordi è formato da tre fasi principali e una fase opzionale:

1. Calcolo dell'intensità e dell'orientamento dei bordi, o *gradiente*;
2. Rilevamento dei massimi locali;
3. Binarizzazione o sogliatura;
4. Miglioramento tramite elaborazione morfologica (opzionale). Tipici operatori morfologici sono l'*apertura* e la *chiusura*, che permettono rispettivamente di allontanare o avvicinare i bordi individuati.

Il calcolo dell'intensità (modulo) e orientamento (fase) del gradiente (fase 1) viene effettuato mediante l'utilizzo di operatori di convoluzione del tutto simile a quelli introdotti precedentemente. Dopodiché, si tratta di costruire l'immagine binaria che contenga la mappa finale dei contorni. Il modo più semplice e immediato per far ciò è applicare un



semplice algoritmo di sogliatura marcando di bianco ciò che è considerato bordo e lasciando in nero ciò che è considerato sfondo. Questa procedura però non fornisce, in generale, ottimi risultati; tenendo presente il fatto che, come risultato dal calcolo del gradiente, ai pixel di contorno è associata una certa direzione, occorre tenere conto di essa durante la fase di sogliatura. In particolare, un'operazione fondamentale è la soppressione dei valori non-massimi presenti nel gradiente percorrendo i contorni nella direzione dello stesso (fase 2). Come risultato di questa operazione, solo i massimi locali del gradiente vengono mantenuti: in altre parole, si applica una procedura di riduzione (*thinning*) dei bordi grossolani forniti dal gradiente, e si ottengono contorni spessi al più un pixel. La posizione e contiguità dei pixel appartenenti ad un bordo dipendono dall'algoritmo di estrazione utilizzato. Dopo questa operazione, è possibile applicare un algoritmo di sogliatura per costruire la mappa binaria finale (fase 3).

**Operatori Derivativi** Dal momento che un bordo è caratterizzato da un certo cambiamento di intensità, l'operatore da utilizzare per la rilevazione dei contorni deve essere sensibile a questo tipo di variazioni. Ciò è esattamente quanto svolto da un *derivatore*. Infatti, una possibile interpretazione del concetto di derivata prevede che tramite essa possa essere misurato il tasso di cambiamento dei valori di una funzione; nelle immagini questo tasso è grande vicino ai bordi e piccolo in aree quasi costanti. Dal momento che le immagini sono segnali numerici a due dimensioni, è importante considerare i cambiamenti di livello in più direzioni. Per questa ragione, vengono usate le derivate parziali dell'immagine rispetto alle direzioni principali (ascisse e ordinate). Una stima della direzione effettiva dei contorni può essere quindi ottenuta usando le derivate in  $x$  e in  $y$  come componenti della direzione effettiva lungo gli assi, e calcolandone la somma vettoriale. L'operatore di cui si parla è per l'appunto il *gradiente*, la cui definizione per quanto concerne la funzione di luminosità  $I$  è data dall'equazione seguente:

$$\nabla I(x, y) = \left( \frac{\delta I}{\delta x}, \frac{\delta I}{\delta y} \right) \quad (3.9)$$

per cui ogni pixel è rappresentato come un vettore e le proprietà del bordo sono l'*intensità* e *direzione* del gradiente, calcolate come nella (3.10).

$$\begin{cases} G_{mag} = \sqrt{\left(\frac{\delta A}{\delta x}\right)^2 + \left(\frac{\delta A}{\delta y}\right)^2} \\ g_{dir} = \arctan \frac{\left(\frac{\delta A}{\delta y}\right)}{\left(\frac{\delta A}{\delta x}\right)} \end{cases} . \quad (3.10)$$

**Operatori basati su template.** Come dice il nome stesso della categoria di tali metodi, l'idea che sta dietro a questo tipo di rilevamento è quella di utilizzare un piccolo *template* (nella fattispecie, una piccola matrice di punti) come modello di un bordo. L'individuazione dei contorni si ottiene cercando le regioni dell'immagine in cui si osserva una certa similarità con il template stesso (da cui il termine *template matching*).

Spesso il template è dato sotto forma di un modello delle variazioni di intensità dei bordi, o come approssimazione di un derivatore. Di rilevatori di questo tipo ne esistono molti e, a dispetto dei loro svantaggi, vengono tuttora utilizzati in diverse applicazioni grazie alla loro efficienza computazionale.

Tra i più comuni rilevatori basati su template, uno dei più usati è l'*operatore di Sobel*. Tale operatore applica due kernel  $3 \times 3$ , quindi due matrici di convoluzione, all'immagine originaria per calcolare dei valori approssimati delle derivate: una per la direzione orizzontale, ed una per la direzione verticale. Se  $\mathbf{A}$  è la matrice di pixel dell'immagine sorgente, e  $G_x$  e  $G_y$  le due immagini i cui punti rappresentano rispettivamente i valori approssimati delle derivate in orizzontale ed in verticale, l'operazione è descritta dalle seguenti relazioni<sup>1</sup>:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A . \quad (3.11)$$

Il gradiente ha segno positivo se l'intensità aumenta da sinistra verso destra lungo l'asse  $x$  e dall'alto verso il basso lungo  $y$ . In ciascun punto dell'immagine i valori approssimati del gradiente possono essere combinati per calcolare il valore totale del gradiente e la sua direzione, come esplicitato nelle equazioni (3.12).

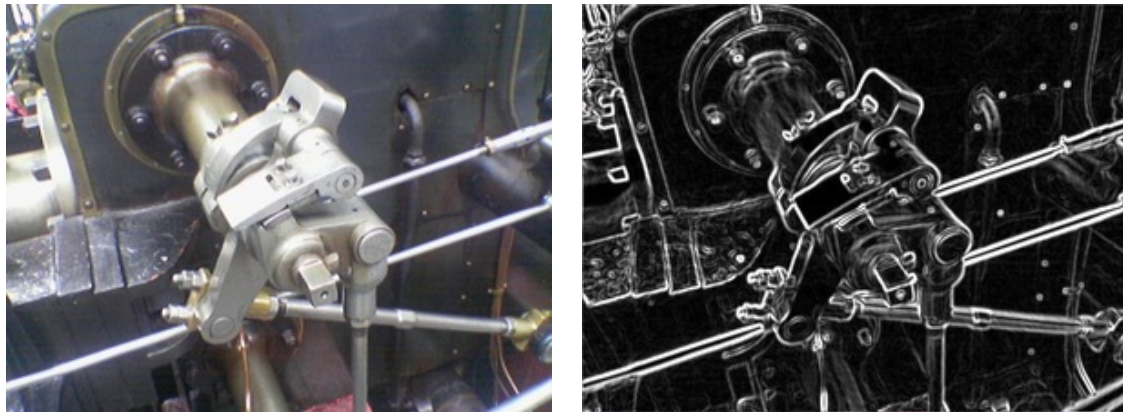
$$\begin{cases} G = \sqrt{G_x^2 + G_y^2} \\ \theta = \arctan\left(\frac{G_y}{G_x}\right) \end{cases} . \quad (3.12)$$

Ad esempio,  $\theta = 0$  per un contorno verticale più "scuro" sul lato sinistro.

Una volta ottenuto il modulo del gradiente, tipicamente si applica una forma di sogliatura per scartare i pixel troppo deboli (ad esempio dovuti al rumore) e considerare tutto il resto come contorno. Si può osservare che la valutazione del gradiente richiede il calcolo di due quadrati e di una radice quadrata, operazioni piuttosto onerose dal punto di vista computazionale; per questo motivo, non è raro vedere utilizzata direttamente la somma dei moduli  $(|\frac{\delta A}{\delta x}| + |\frac{\delta A}{\delta y}|)$  come approssimazione.

In figura 3.11 è riportato un esempio dell'applicazione del filtro di Sobel.

<sup>1</sup>\* indica l'operatore di convoluzione bidimensionale



(a) Immagine di partenza.

(b) Immagine dopo applicazione del filtro di Sobel.

**Figura 3.11:** Filtro di Sobel

Gli operatori basati su template danno generalmente risultati migliori degli operatori derivativi, ma sono comunque ancora troppo sensibili al rumore. Un modo per alleviare il problema consiste nell'espandere le maschere utilizzate.

Poiché la funzione che definisce l'intensità luminosa di un'immagine digitale è nota solo in punti discreti, le derivate di questa funzione non possono a rigore essere definite a meno di assumere l'esistenza di una sottostante funzione di luminosità continua che sia stata campionata nei punti dell'immagine. Per tale motivo, la precisione di calcolo della derivata è fortemente dipendente dalla frequenza di campionamento spaziale dell'immagine, ovvero della risoluzione.

L'operatore di Sobel fornisce un'approssimazione poco accurata del gradiente dell'immagine, ma ottenibile velocemente con solo un passo di filtraggio. Più esattamente, la maschera di filtraggio di Sobel usa i valori di luminosità solo in una regione  $3 \times 3$  intorno ad ogni punto dell'immagine.

**Operatore di Canny.** Nel 1986, John Canny ha definito un insieme di obiettivi che un rilevatore di bordi ideale dovrebbe raggiungere, e ha descritto un metodo ottimale per perseguirli. I tre obiettivi principali sono i seguenti:

1. Un rilevatore dovrebbe rilevare tutti e soli i bordi (*corretto rilevamento*);
2. La distanza tra i pixel di bordi identificati e i bordi effettivi dovrebbe essere la più piccola possibile (*corretta localizzazione*);
3. Un rilevatore dovrebbe rilevare ogni bordo una sola volta (*risposta singola*).

Il rilevatore di Canny è quindi un filtro il cui compito è quello di localizzare i bordi attenuando nel contempo la componente dovuta al rumore. Il problema consiste nella ricerca di un filtro reale che ottimizzi i tre criteri sopra enunciati.

In particolare, non è stata trovata una soluzione analitica che ottimizza i tre criteri, ma Canny ha fornito un'approssimazione efficiente del filtro sotto forma di derivata prima di una distribuzione gaussiana. Uno dei grandi vantaggi di questa procedura risiede anche nei tempi di calcolo: realizzare una convoluzione è semplice ma computazionalmente oneroso, anche passando al dominio delle frequenze; tuttavia, essendo la distribuzione bivariata gaussiana una funzione separabile, la convoluzione a due dimensioni può essere separata in due convoluzioni con due gaussiane monodimensionali, una lungo le ascisse  $x$  e la seconda lungo le ordinate  $y$ .

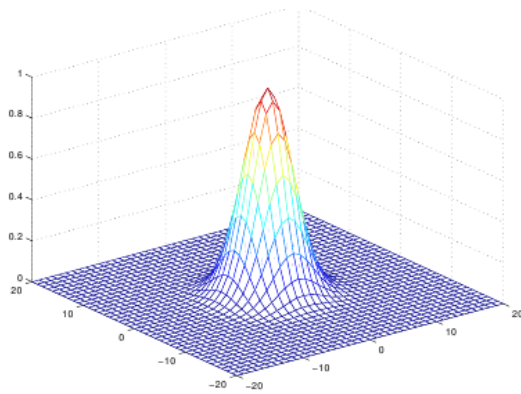
L'algoritmo proposto da Canny consiste in quattro passi:

1. Filtro gaussiano (*image smoothing*).
2. Calcolo del gradiente.
3. *Thinning*.
4. Sogliatura.

Il primo passo effettua lo smoothing dell'immagine di input mediante un filtro Gaussiano, introducendo una sfocatura dell'immagine tale da eliminare le variazioni di luminosità troppo piccole. Questo influenza i risultati generati dall'algoritmo, poiché filtri di piccole dimensioni producono una minore sfocatura e consentono di riconoscere dettagli più fini, mentre filtri più grandi producono una maggiore sfocatura e sono indicati per riconoscere dettagli più ampi. In tale filtro, i fattori moltiplicativi della maschera di filtraggio seguono l'andamento di una distribuzione gaussiana con media nel centro del kernel ( $w(s, t) = \exp - \frac{s^2 + t^2}{2\sigma^2}$ ). Il parametro fondamentale del filtro, da cui dipende il livello dei dettagli da individuare, è la deviazione standard  $\sigma$ . In figura 3.12 viene riportato un esempio di maschera di convoluzione gaussiana di dimensione  $7 \times 7$  e  $\sigma = 3$ .

La differenziazione è il passo successivo dell'algoritmo e permette di calcolare il gradiente dell'immagine ottenuta dal filtraggio gaussiano. Ne segue il calcolo del modulo  $M(x, y)$  e della fase  $\theta(x, y)$  del vettore gradiente.

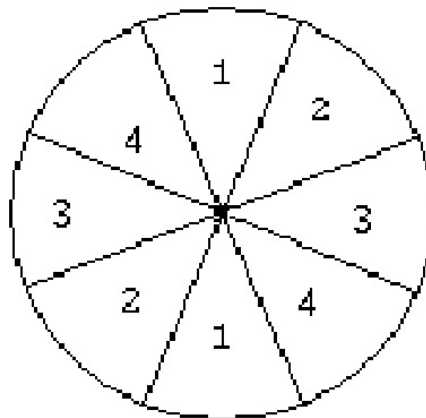
Il terzo passo consiste nella soppressione dei non-massimi. Un pixel appartiene ad un bordo se e solo se l'intensità del gradiente in una determinata direzione è massima rispetto allo stesso valore calcolato nei pixel appartenenti al suo vicinato. Per ogni punto bisogna quindi individuare la direzione del gradiente e confrontare il suo modulo con quello dei

(a) Gaussiana con  $\sigma = 3$ .

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

(b) Kernel gaussiano  $7 \times 7$ .**Figura 3.12:** Filtro gaussiano

vicini giacenti lungo la direzione stessa. Se almeno uno dei due vicini ha modulo maggiore del pixel in esame, questo viene soppresso ponendo il modulo a zero. In una regione di dimensione  $3 \times 3$  si possono definire quattro orientamenti per un bordo passante attraverso il punto centrale, ossia la direzione orizzontale, verticale e le due diagonali ( $45^\circ$  e  $135^\circ$ ), come mostrato in figura 3.13. La direzione del bordo è data dalla direzione principale del gradiente: se il vettore ha valore angolare compreso nel settore 1 si ha un bordo orizzontale, nel settore 3 si ha un bordo verticale e, infine, nei rimanenti un bordo diagonale.

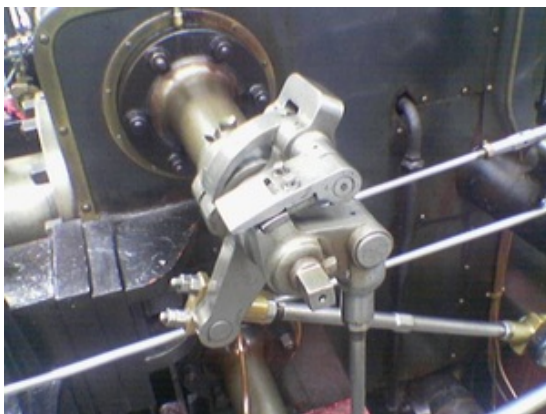
**Figura 3.13:** Algoritmo di Canny: discretizzazione della direzione

Nel quarto passo (*edge thresholding*), si esegue la sogliatura dell'immagine ottenuta al passo precedente. Si devono definire due soglie, una bassa  $T_{low}$  ed una alta  $T_{high}$ , dal cui confronto si ottengono i pixel del contorno. Se il valore è inferiore alla soglia minore  $T_{low}$ , il punto viene scartato. Se invece è superiore alla soglia maggiore  $T_{high}$ , il punto viene accettato come appartenente ad un bordo. Infine, se l'intensità del pixel è compresa tra le soglie, il punto viene accettato solo se contiguo ad un punto precedentemente accettato. L'uso delle

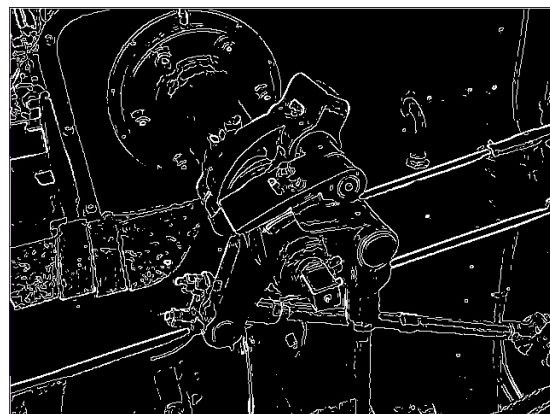
soglie  $T_{low}$  e  $T_{high}$  permette di ridurre la presenza di falsi bordi rispetto ad una sogliatura basata su un singolo valore di soglia  $T$ . Utilizzando una soglia singola si rischia infatti di porla ad un valore troppo basso ottenendo dei falsi bordi; in questo caso un semplice disturbo potrebbe essere inglobato nei bordi dell'immagine. Se invece la soglia viene posta ad un valore troppo alto, i punti appartenenti ad un bordo potrebbero venire eliminati provocando in questo modo la perdita di informazione significativa. Canny suggerisce che i risultati migliori si ottengono scegliendo il valore  $T_{high}$  due o tre volte più grande rispetto al valore  $T_{low}$ . In pratica la soglia  $T_{high}$  serve a localizzare le strutture significative dell'immagine, mentre quella inferiore serve a connetterle.

Le prestazioni dell'algoritmo di Canny dipendono pesantemente dalla deviazione standard  $\sigma$  del filtro Gaussiano e dai valori di soglia adottati nell'isteresi. Il valore  $\sigma$  controlla anche la dimensione del filtro poiché essa deve essere estesa in modo da contenere tutte le componenti aventi una certa influenza; ad un valore alto di  $\sigma$  corrisponde quindi un filtro di dimensioni grandi, il che implica l'aumento della sfocatura e bordi più grossolani. La scelta di un valore di  $\sigma$  alto è però necessario per ridurre il rumore all'interno delle immagini. Ovviamente in questo modo la localizzazione dei bordi è meno accurata. A valori piccoli di  $\sigma$ , d'altra parte, corrispondono sia bordi più fini e meglio localizzati che una maggiore sensibilità al rumore.

In figura 3.14 il risultato ottenuto con l'applicazione dell'operatore di Canny.



(a) Immagine di partenza.



(b) Immagine dopo applicazione dell'operatore di Canny.

**Figura 3.14:** Operatore di Canny

### 1.3 Punti d'interesse (keypoint)

Un *punto d'interesse* (*keypoint*) è un punto che deve possedere alcune caratteristiche locali che altri punti non hanno e che ne permettano quindi l'identificazione all'interno di un'immagine. Per esempio, nel caso di proiezioni poligonali di oggetti, punti d'interesse possono essere gli angoli del contorno; in questo caso, la caratteristica che li rende unici è il netto cambio d'orientamento della direzione della retta tangente al contorno.

Un *angolo* (*corner*) è una caratteristica locale perché il cambio di direzione della tangente è modellabile con due rette e, quindi, con 3 punti. Pertanto, gli angoli vengono individuati utilizzando identificatori locali: data un'immagine a livelli di grigio si ottiene un'immagine il cui valore d'intensità dei pixel rappresenta la probabilità che il pixel appartenga ad un angolo. Sogliando questa immagine, si ottiene un vettore contenente le posizioni (riga e colonna) dei punti trovati e un descrittore delle caratteristiche del punto, ovvero i gradienti del contorno.

Gli algoritmi di *corner detection* si fondano tutti sul principio per cui un angolo corrisponde al punto in cui si osserva un cambio d'orientamento dei contorni. In quest'ottica, il corner detector più semplice è il cosiddetto *rilevatore di Moravec*, il quale restituisce un valore massimo per i pixel che presentano un vicinato con alto contrasto:

$$MO(i, j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{t=j-1}^{j+1} |f(k, t) - f(i, j)| . \quad (3.13)$$

Un rilevatore di angoli migliore e, forse, quello più utilizzato in assoluto è il rilevatore di *Harris*. Questo migliora il rilevatore di Moravec considerando il differenziale dello *score* (somma delle differenze al quadrato) dei singoli pixel. Sia  $f$  un'immagine in scala di grigi e  $W$  un'immagine alternativa ottenuta da  $f$  traslando tutti i suoi pixel di una quantità  $(\Delta_x, \Delta_y)$ . Lo score  $S_W$  è quindi definito come

$$S_W(\Delta_x, \Delta_y) = \sum_{x_i \in W} \sum_{y_i \in W} (f(x_i, y_i) - f(x_i - \Delta_x, y_i - \Delta_y))^2 . \quad (3.14)$$

Come descritto dalla (3.15), l'immagine  $W$  può essere approssimata da una serie di Taylor del primo ordine.

$$f(x_i - \Delta_x, y_i - \Delta_y) \approx f(x_i, y_i) + \left[ \frac{\delta f(x_i, y_i)}{\delta x}, \frac{\delta f(x_i, y_i)}{\delta y} \right] \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} . \quad (3.15)$$

Pertanto, il valore minimo di score può essere ottenuto analiticamente. Sostituendo il valore approssimato dell'immagine  $W$  (3.15) nell'equazione dello score 3.14 si ottiene la

definizione della *matrice di Harris*  $\mathbf{A}$ . In (3.16) è riportato lo svolgimento dei calcoli che permettono di arrivare alla definizione di tale matrice.

$$\begin{aligned}
 S(x, y) &= \sum_{x_i \in W} \sum_{y_i \in W} (f(x_i, y_i) - f(x_i, y_i) - \begin{bmatrix} \frac{\delta f(x_i, y_i)}{\delta x} & \frac{\delta f(x_i, y_i)}{\delta y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix})^2 \\
 &= \sum_{x_i \in W} \sum_{y_i \in W} (+ \begin{bmatrix} \frac{\delta f(x_i, y_i)}{\delta x} & \frac{\delta f(x_i, y_i)}{\delta y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix})^2 \\
 &= \sum_{x_i \in W} \sum_{y_i \in W} [\Delta x, \Delta y] \begin{pmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{pmatrix} \begin{bmatrix} \frac{\delta f}{\delta x} & \frac{\delta f}{\delta y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
 &= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \left( \sum_{x_i \in W} \sum_{y_i \in W} \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix} \begin{bmatrix} \frac{\delta f}{\delta x} & \frac{\delta f}{\delta y} \end{bmatrix} \right) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \\
 &= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} A_W(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}
 \end{aligned} \tag{3.16}$$

Si può notare che la *matrice di Harris*  $\mathbf{A}$  definita dalla relazione (3.17)) contiene le informazioni relative alla derivata seconda di  $S_W$  attorno al punto  $(x, y) = (0, 0)$ .

$$\mathbf{A}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sum_{x_i \in W} \sum_{y_i \in W} \frac{\delta^2 f(x_i, y_i)}{\delta x^2} & \sum_{x_i \in W} \sum_{y_i \in W} \frac{\delta f(x_i, y_i)}{\delta x} \frac{\delta f(x_i, y_i)}{\delta y} \\ \sum_{x_i \in W} \sum_{y_i \in W} \frac{\delta f(x_i, y_i)}{\delta x} \frac{\delta f(x_i, y_i)}{\delta y} & \sum_{x_i \in W} \sum_{y_i \in W} \frac{\delta^2 f(x_i, y_i)}{\delta y^2} \end{bmatrix}. \tag{3.17}$$

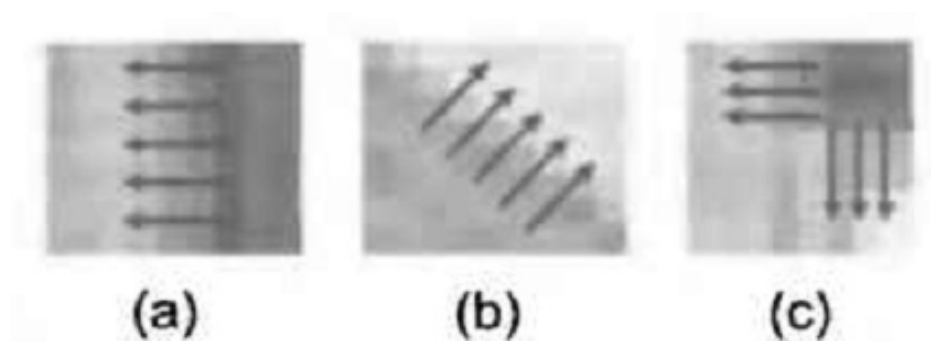
Si possono distinguere tre casi, illustrati in figura 3.15, a seconda dei valori assunti dagli autovalori della matrice  $\mathbf{A}$ :

- Entrambi gli autovalori sono al di sotto di una certa soglia: il pixel esaminato non appartiene ad un contorno, quindi non è un angolo;
- Un autovalore è al di sotto della soglia e l'altro al di sopra: una variazione significativa nell'immagine è causata da un piccolo spostamento in una direzione. Questo accade se il pixel appartiene ad un contorno, ma probabilmente non è un angolo;
- Entrambi gli autovalori sono sopra la soglia: un piccolo spostamento in una qualsiasi delle due direzioni provoca una variazione consistente nell'immagine. Il pixel è un punto d'angolo del bordo.

Le fasi attraverso cui passa il detector di Harris (per ogni pixel) sono quindi le seguenti:

1. Calcolo della matrice  $\mathbf{A}$ .

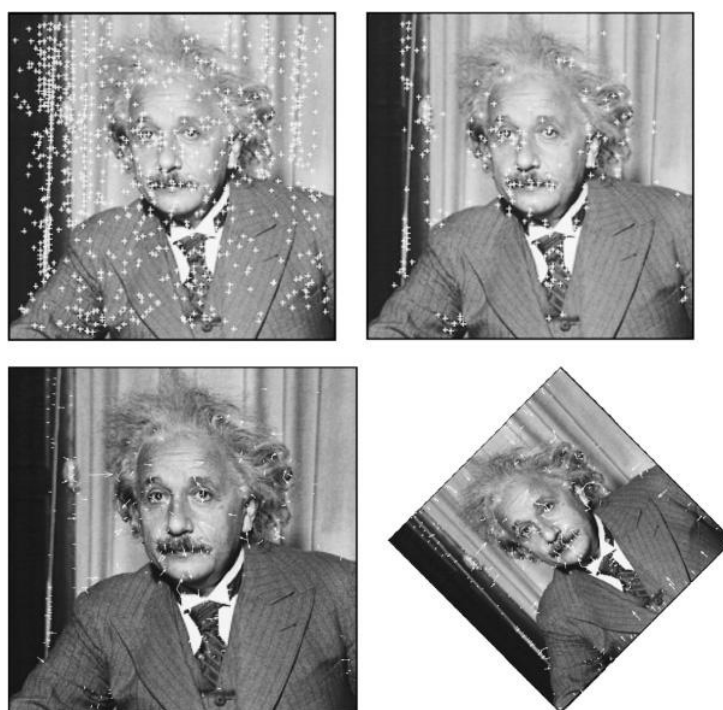




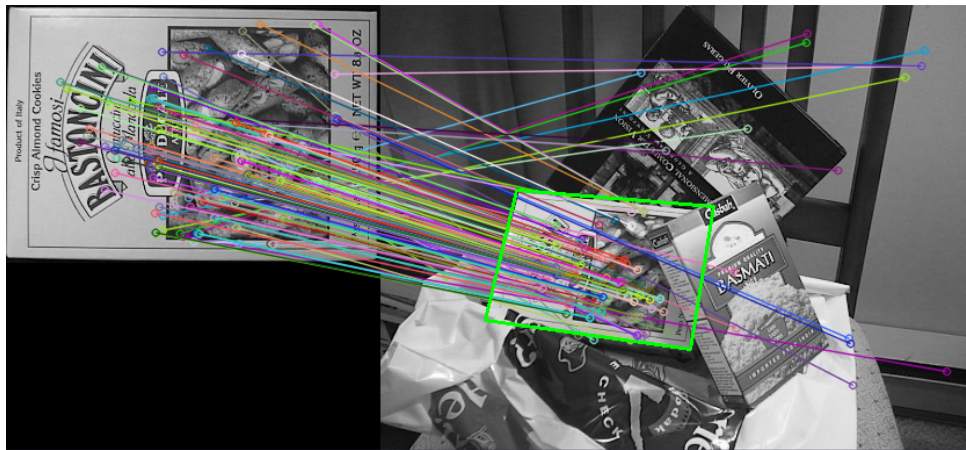
**Figura 3.15:** Rilevatore di Harris: contorni e angoli

2. Calcolo degli autovalori della matrice.
3. Sogliatura degli autovalori e decisione.

A questo punto, le feature acquisite in immagini differenti possono essere associate tramite algoritmi che sfruttano il metodo dei minimi quadrati (vedi 3. Esempi di questi algoritmi sono l'algoritmo *SIFT* (*Scale Invariant Features Transform*, figura 3.16) o *SURF* (*Speeded Up Robust features*, figura 3.17), i quali sfruttano il rilevatore di Harris per l'individuazione dei keypoint e una stima dei minimi quadrati per ottenere la stima del vettore delle corrispondenze tra due immagini.



**Figura 3.16:** Scale Invariant Features Transform



*Figura 3.17: Speeded Up Robust features*

## 2 Descrizione

La fase di *descrizione* si inserisce subito dopo la fase di segmentazione, con lo scopo di rappresentare matematicamente il contenuto dell'immagine. Tale rappresentazione fornisce una cifra di merito per la costruzione di modelli della scena osservata, utilizzabili in seguito all'individuazione di una determinata feature per definirne l'appartenenza o meno ad un particolare oggetto da ricercare. I descrittori vengono classificati in base al tipo di informazioni che li costituiscono e alla loro estensione sull'immagine.

Per quanto riguarda il tipo di informazioni, si distingue tra:

- *Descrittori geometrici.* Partendo da un insieme di pixel, permettono di calcolare alcune proprietà geometriche dei punti, quali l'orientamento, il centro geometrico e l'approssimazione geometrica su linee o poligoni. Per il calcolo di queste caratteristiche è necessario passare dalla fase di segmentazione e ottenere prima un sottoinsieme di punti, regioni o contorni.
- *Descrittori statistici.* L'immagine viene descritta tramite proprietà non necessariamente con contenuto geometrico, ad esempio può essere utilizzata direttamente il valore di luminosità e/o colore di un insieme di pixel. In questo caso non è quindi necessario passare dalla segmentazione delle immagini.

Relativamente all'estensione e applicabilità del descrittore, invece, la classificazione viene fatta tra:

- *Descrittori locali.* Appartengono a questa categoria tutti i descrittori di singoli oggetti e aree della scena. Possono essere di tipo geometrico o statistico. Generalmente questi

descrittori sono costituiti da un vettore di caratteristiche geometriche (punti, linee, contorni, ...) o di proprietà calcolate da un'area (perimetro, area, orientamento, ...).

- *Descrittori globali*. Permettono di descrivere la scena utilizzando tutti i pixel dell'immagine. Solitamente, i descrittori globali sono di tipo statistico. Un esempio di questi descrittori è costituito dagli istogrammi di luminosità e/o colore che permettono di rappresentare la distribuzione di probabilità della luminosità di un'immagine.

I keypoint (vedi paragrafo 1.3), infine, sono un tipo particolare di descrittore. Infatti, il calcolo del descrittore non viene eseguito in fasi successive come per i descrittori geometrici, ma è ottenuto insieme all'identificazione dei punti stessi. Ad esempio, per quanto riguarda gli angoli, l'individuazione del punto d'interesse prevede il calcolo dell'intensità del gradiente nelle differenti direzioni dell'immagine; l'intensità del gradiente nelle varie direzioni viene in seguito utilizzata sia per individuare un sottoinsieme di punti d'interesse all'interno dell'immagine, sia per classificare il punto all'interno di un modello della scena.

## 2.1 Istogrammi

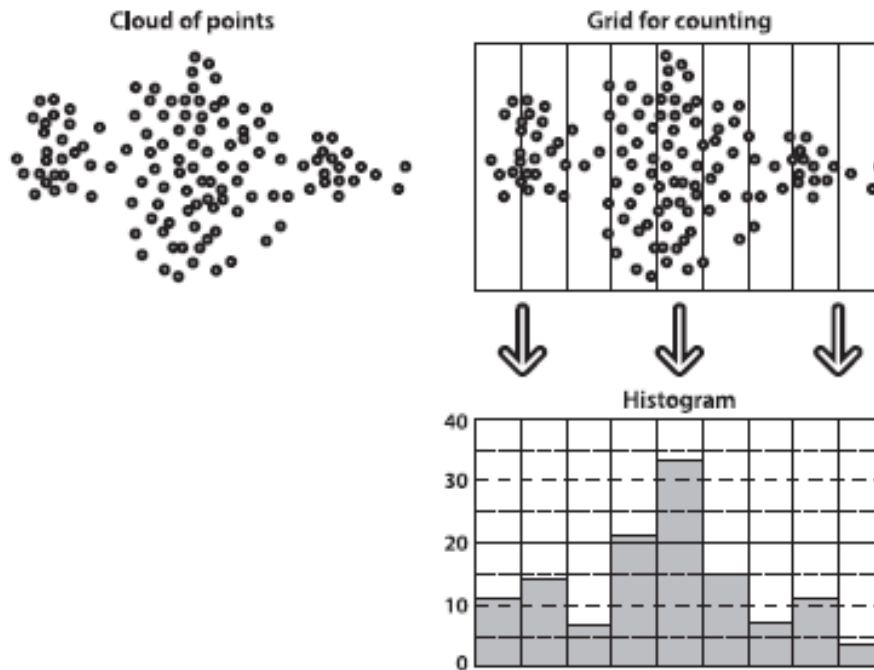
L'*istogramma* è una rappresentazione grafica di una distribuzione di probabilità costruita a partire da un campione di dati. Gli istogrammi sono quindi costituiti da una raccolta di dati raggruppati in insiemi predefiniti, detti *bin*. Tale processo di formazione dell'istogramma è mostrato in figura 3.18. I bin possono esser popolati da un numero di feature calcolate direttamente a partire dalle informazioni dei pixel dell'immagine, come ad esempio direzione e modulo del gradiente, colori o altre caratteristiche.

L'accuratezza di un istogramma dipende dalla dimensione della griglia, ovvero dalla distanza tra un bin e l'altro: più questa è ridotta, più l'istogramma risulta accurato. D'altra parte, una griglia troppo fitta può portare a risultati poco mediati e quindi più difficilmente utilizzabili per l'identificazione di modelli.

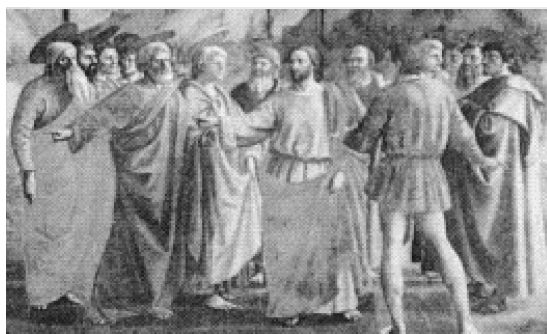
Nell'elaborazione di immagini si fa largo uso degli istogrammi per la rappresentazione della distribuzione di alcune proprietà delle immagini, tipicamente la luminosità dei pixel (figura 3.19).

Il motivo dell'ampio uso degli istogrammi è dato dal fatto che essi permettono di definire facilmente alcune trasformazioni di luminosità, tra cui:

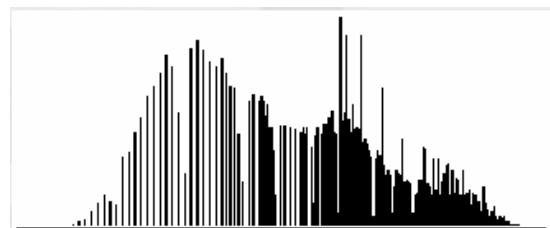
- *Negativo* di un'immagine. Corrisponde all'inversione della luminosità dei pixel: l'istogramma viene specchiato rispetto al suo asse verticale.



**Figura 3.18:** Formazione dell'istogramma



(a) Immagine.



(b) Istogramma.

**Figura 3.19:** Istogramma della luminosità di un'immagine

- *Normalizzazione.* L'istogramma viene ampliato lungo l'asse  $y$ , ovvero si fa in modo che l'immagine contenga tutti i valori di luminosità ammessi.
- *Equalizzazione.* L'istogramma viene ampliato lungo l'asse  $x$  e appiattito, per far sì che ogni componente di luminosità sia presente con la stessa frequenza.
- *Intensificazione.* Viene modificata la luminosità media dell'immagine.
- *Correzione del contrasto.* Il range di luminosità presente nell'immagine viene scalato su un range diverso.

L'identificazione di una scena modellata attraverso istogrammi avviene attraverso il confronto dei due istogrammi, a seguito del quale si decide se vi è o meno una certa corrispondenza tra le osservazioni e la scena ricercata. Tra le misure di confronto tra istogrammi più utilizzate si possono elencare:

- *Correlazione*, (3.18). Per correlazione, un risultato alto rappresenta un match molto buono tra i due istogrammi considerati. I valori variano tra  $-1, 0, 1$ , dove  $-1$  indica la massima differenza ottenibile,  $1$  il migliore e  $0$  assenza di correlazione.

$$d_{corr}(H_1, H_2) = \frac{\sum_i H'_1(i)H'_2(i)}{\sqrt{\sum_i H'^2_1(i)H'^2_2(i)}} . \quad (3.18)$$

- *Chi-square*, definito dalla (3.19), dove un risultato basso rappresenta il match migliore.

$$d_{chi-square}(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)} . \quad (3.19)$$

- L'*intersezione* rappresenta anch'esso un metodo dove un risultato alto si riferisce al match migliore. La distanza di intersezione è definita dalla (3.20).

$$d_{intersection}(H_1, H_2) = \sum_i \min(H_1(i), H_2(i)) . \quad (3.20)$$

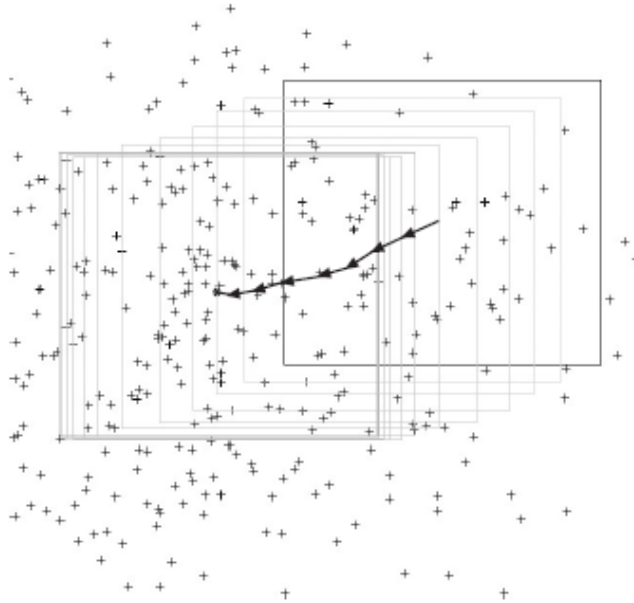
- *Distanza di Bhattacharyya*, definita dalla (3.21), per cui un risultato basso equivale ad un buon match.

$$d_{Bhattacharyya}(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i)H_2(i)}}{\sqrt{\sum_i H_1(i) \sum_i H_2(i)}}} . \quad (3.21)$$

## 2.2 Mean shift

L'algoritmo *mean shift* è un metodo robusto per individuare gli estremi locali in un insieme denso di dati, come mostrato in figura 3.20. La robustezza è da intendersi nel senso statistico della parola, poiché l'algoritmo permette di ignorare gli outlier nei dati e cioè quei punti che sono lontani dal valor medio dei dati. L'algoritmo segue questi passi:

1. Selezione di una finestra (o kernel) di ricerca: posizione iniziale, tipo (uniforme, polinomiale, esponenziale, Gaussiana), forma (simmetrica, ruotata, rettangolare, arrotondata), dimensione;
2. Calcolo del centro di massa della finestra (valore medio);
3. Centraggio della finestra nel centro di massa;
4. Iterazione dei punti 2 e 3 finché la finestra non smette di muoversi.



**Figura 3.20:** *Algoritmo Mean Shift*

Il centro di massa può essere ottenuto calcolando il valore medio del set di  $n$  dati:

$$\begin{cases} x_c = \frac{\sum_{i=1}^n x_i}{n} \\ y_c = \frac{\sum_{i=1}^n y_i}{n} \end{cases} . \quad (3.22)$$

Sia

$$\mathbf{K}(\mathbf{X} - \mathbf{X}_i) = ck \left( \left\| \frac{\mathbf{X} - \mathbf{X}_i}{h} \right\|^2 \right)$$

il kernel considerato in partenza cui corrisponde la distribuzione di probabilità  $P(x) = \frac{1}{n} \sum_{i=1}^n \mathbf{K}(\mathbf{x} - \mathbf{x}_i)$ , ovvero la probabilità di osservare le feature  $\mathbf{x}$  date le osservazioni al passo precedente  $\mathbf{x}_i$ . Il gradiente della distribuzione di probabilità è dato da:

$$\nabla P(x) = \frac{1}{n} \sum_{i=1}^n \nabla \mathbf{K}(\mathbf{x} - \mathbf{x}_i) .$$

Se  $g(x) = -k(x)$  è la derivata del kernel, allora il gradiente di probabilità può essere ridefinito come:

$$\nabla P(x) = \frac{\epsilon}{n} \sum_{i=1}^N \nabla K_i = \frac{\epsilon}{n} \left[ \sum_{i=1}^N g_i \left( \frac{\|x - x_i\|^2}{h} \right) \right] \left[ \frac{\sum_{i=1}^n x_i g_i \left( \frac{\|x - x_i\|^2}{h} \right)}{\sum_{i=1}^n g_i \left( \frac{\|x - x_i\|^2}{h} \right)} - x \right] , \quad (3.23)$$

dove  $h$  è la dimensione in pixel (area) della finestra considerata e  $x$  è il vettore di spostamento della finestra (*mean shift vector*). Pertanto, tramite questo metodo si riescono a mettere in relazione lo spostamento delle feature all'interno della scena con la variazione

del valore delle feature stesse. Per tale motivo, l'algoritmo mean shift è spesso utilizzato per risolvere i problemi di tracciamento di oggetti in movimento, ovvero il problema del tracking (vedi paragrafo subsection::tracking::features::meanshift, capitolo 6). Il vettore di spostamento della finestra viene utilizzato per riposizionare la finestra stessa al passo successivo dell'algoritmo. Pertanto, la condizione di arresto dell'algoritmo può essere rivista come l'ottenimento di un vettore di spostamento nullo.

La finestra individuata al termine dell'algoritmo è quella che contiene il massimo contenuto d'informazione del set di caratteristiche preso in analisi e il descrittore ottenuto è quindi costituito dalle caratteristiche della finestra stessa: centro geometrico e dimensioni.

## 2.3 Momenti e descrittori geometrici

Uno dei modi più semplici e, di conseguenza, meno onerosi per ottenere informazioni su un oggetto e fornirne una rappresentazione geometrica consiste nel calcolare i *momenti* a partire dai punti del suo contorno ([30]).

### 2.3.1 Definizione

Sia  $f(x, y)$  la funzione di luminosità del pixel  $p_{x,y}$  di un'immagine. In particolare, per un immagine binaria,  $f(x, y)$  è descritta tramite la funzione non continua

$$f(x, y) = b(x, y) = \begin{cases} 1 & , \text{ Oggetto} \\ 0 & , \text{ Sfondo} \end{cases} . \quad (3.24)$$

Come si vede nella (3.25), il momento della funzione  $f(x, y)$  è definito come l'integrale della funzione di luminosità sull'intera immagine o sulla regione dell'immagine individuata ritagliata dai punti del contorno in esame.

$$m_{p,q} = \iint_A x^p y^q f(x, y) dx dy . \quad (3.25)$$

I momenti vengono classificati in funzione del loro ordine. L'ordine di un momento è dato dagli indici  $p$  e  $q$  dell'integrale (3.25). La somma  $p + q$  è detta ordine del momento  $m_{p,q}$ . Generalmente vengono calcolati i seguenti momenti:

- Momento di ordine 0:  $(p, q) = (0, 0)$

$$m_{0,0} = \iint f(x, y) dx dy$$

che descrive l'*area*  $A$  della regione.

- Momento di ordine 1:  $(p, q) = (1, 0)$  o  $(0, 1)$

$$\begin{cases} m_{1,0} = \iint x f(x, y) dx dy \\ m_{0,1} = \iint y f(x, y) dx dy \end{cases}$$

da cui si ricavano le coordinate del *centro geometrico* dell'oggetto:

$$\begin{cases} x_c = \frac{m_{1,0}}{m_{0,0}} \\ y_c = \frac{m_{0,1}}{m_{0,0}} \end{cases} . \quad (3.26)$$

- Momenti del second'ordine:  $(p, q) = (2, 0)$  p  $(0, 2)$  o  $(1, 1)$

$$\begin{cases} m_{2,0} = \iint x^2 f(x, y) dx dy \\ m_{0,2} = \iint y^2 f(x, y) dx dy \\ m_{1,1} = \iint xy f(x, y) dx dy \end{cases} .$$

### 2.3.2 Momenti centrali

Le definizioni date fin qui si riferiscono ai *momenti spaziali*. Centrando le coordinate dei pixel della regione considerata sul centro geometrico della regione stessa, è possibile ottenere i *momenti centrali*:

$$\mu_{p,q} = \iint (x - x_c)^p (y - y_c)^q f(x, y) dx dy . \quad (3.27)$$

I momenti centrali hanno lo stesso contenuto d'informazione dei momenti spaziali, ma il loro valore non dipende dalla posizione della regione all'interno dell'immagine.

Per i momenti fino al primo ordine si ha

$$\begin{cases} \mu_{0,0} = m_{0,0} \\ \mu_{1,0} = \mu_{0,1} = 0 \end{cases} .$$

Il calcolo dei momenti centrali per ordini superiori al primo è dato dalla relazione (3.28).

$$\mu_{p,q} = \frac{m_{p,q}}{m_{0,0}} - \left( \frac{m_{1,0}}{m_{0,0}} \right)^p \cdot \left( \frac{m_{0,1}}{m_{0,0}} \right)^q . \quad (3.28)$$

### 2.3.3 Momenti invarianti

I cosiddetti *momenti invarianti* costituiscono la base di molti descrittori di forme geometriche. Infatti, i momenti centrali permettono di ottenere le proprietà geometriche delle forme, ma essi dipendono dall'orientamento e dalla posa dell'oggetto osservato. Il loro valore, infatti, non cambia in presenza di traslazioni, mentre non è invariante a rotazioni e variazioni di scala. Particolari combinazioni dei momenti centrali permettono di ottenere misure invarianti anche a rotazioni e fattori di scala.



**Invarianza ai cambi di scala.** I momenti invarianti ai fattori di scala si ottengono dalla *normalizzazione* dei momenti centrali. La normalizzazione è ottenuta dividendo un momento centrale per il momento di ordine 0 scalato in modo appropriato:

$$\eta_{i,j} = \frac{\mu_{i,j}}{(\mu_{0,0})^{1+\frac{i+j}{2}}} . \quad (3.29)$$

I momenti così ottenuti vengono detti *momenti centrali normalizzati*.

**Invarianza alle rotazioni** L'insieme di momenti invarianti più utilizzati sono stati definiti da Hu ([28]), la cui definizione è mostrata in (3.30).

$$\begin{aligned} I_1 &= \eta_{2,0} + \eta_{0,2} \\ I_2 &= (\eta_{2,0} + \eta_{0,2})^2 + 4\eta_{1,1}^2 \\ I_3 &= (\eta_{3,0} - 3\eta_{1,2})^2 + (3\eta_{2,1} - \eta_{0,3})^2 \\ I_4 &= (\eta_{3,0} + \eta_{1,2})^2 + (\eta_{2,1} + \eta_{0,3})^2 \\ I_5 &= (\eta_{3,0} - 3\eta_{1,2})(\eta_{3,0} + \eta_{1,2})[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2] + \\ &\quad (3\eta_{2,1} - \eta_{0,3})(\eta_{2,1} + \eta_{0,3})[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2] \\ I_6 &= (\eta_{2,0} - \eta_{0,2})[(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2] + 4\eta_{1,1}(\eta_{3,0} + \eta_{1,2})(\eta_{2,1} + \eta_{0,3}) \\ I_7 &= (3\eta_{2,1} - \eta_{0,3})(\eta_{3,0} + \eta_{1,2})[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2] - \\ &\quad (\eta_{3,0} - 3\eta_{1,2})(\eta_{2,1} + \eta_{0,3})[3(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2] \end{aligned} \quad (3.30)$$

Il primo momento  $I_1$  è l'analogo del momento d'inerzia attorno al centroide dell'immagine. Il momento  $I_7$ , invece, è invariante rispetto all'operazione di riflessione delle immagini. Al set di momenti di Hu originali è possibile aggiungere un ulteriore momento di terzo ordine, mostrato in (3.31).

$$I_8 = \eta_{1,1}[(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{0,3} + \eta_{2,1})^2] - (\eta_{2,0} - \eta_{0,2})(\eta_{3,0} + \eta_{1,2})(\eta_{0,3} + \eta_{2,1}) . \quad (3.31)$$

### 2.3.4 Caratteristiche geometriche

Mentre il momento spaziale di ordine 0 ha un intrinseco significato geometrico, i momenti spaziali di ordine superiore non forniscono direttamente un'informazione geometrica. Tuttavia, essi possono venire utilizzati per derivare alcuni parametri geometrici d'interesse, il cui insieme costituisce il descrittore basato sui momenti. Di seguito vengono riassunte le principali proprietà ottenibili tramite combinazione dei momenti geometrici:

- *Area dell'oggetto.*

$$A = m_{0,0} . \quad (3.32)$$

- *Centro geometrico.*

$$\begin{cases} x_c = \frac{m_{1,0}}{A} = \frac{m_{1,0}}{m_{0,0}} \\ y_c = \frac{m_{0,1}}{A} = \frac{m_{0,1}}{m_{0,0}} \end{cases} . \quad (3.33)$$

- *Tensore dei momenti.*

Analogamente ai momenti meccanici i tre momenti centrali di secondo ordine sono le componenti del tensore d'inerzia della rotazione dell'oggetto attorno al suo centro geometrico, come mostrato nella (3.34).

$$\mathbf{J} = \begin{bmatrix} \mu_{2,0} & -\mu_{1,1} \\ -\mu_{1,1} & -\mu_{0,2} \end{bmatrix} . \quad (3.34)$$

Utilizzando il tensore d'inerzia è possibile ricavare:

- *Assi d'inerzia principale e secondario*

L'asse d'inerzia principale può essere ricavato calcolando gli autovalori della matrice rappresentante il tensore d'inerzia:

$$\lambda_{1,2} = \sqrt{\frac{1}{2} \cdot (\mu_{2,0} + \mu_{0,2}) \pm \sqrt{4 \cdot \mu_{1,1}^2 - (\mu_{2,0} - \mu_{0,2})^2}} .$$

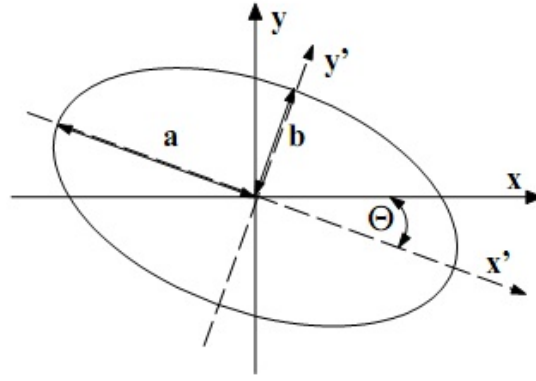
Gli assi principali d'inerzia sono dati dagli autovettori corrispondenti che giacciono sugli assi di un'ellissi che può essere utilizzata come prima approssimazione dell'area dell'oggetto (figura 3.21). L'asse secondario è quello attorno al quale l'oggetto viene ruotato con la massima inerzia, mentre l'asse principale fa riferimento alla rotazione con minima inerzia.

- *Orientamento dell'oggetto*

L'orientamento  $\theta$  dell'oggetto è definito come l'angolo compreso tra l'asse d'inerzia principale e l'asse delle ascisse. Lungo questa direzione l'oggetto ha la sua massima estensione.

Per quanto riguarda il calcolo, l'orientamento corrisponde all'autovettore relativo al minimo autovalore, come detto nella (3.35).

$$\theta = \frac{1}{2} \arctan \frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}} . \quad (3.35)$$



**Figura 3.21:** Assi d'inerzia di un oggetto

- *Eccentricità e compattezza.*

La compattezza ((3.36)) è una misura della distribuzione radiale dei punti del contorno. Può assumere valori compresi tra 0 e 1, dove il valore minimo si ha per una retta e il valore massimo per un cerchio. È un parametro indipendente dalla rotazione dell'oggetto.

$$M_{cmp} = \frac{1}{2\pi} \cdot \frac{S}{\mu_{2,0} + \mu_{0,2}} . \quad (3.36)$$

L'eccentricità ((3.37)) misura invece il grado di elongazione di un oggetto (tra 0 e 1). Per un disco l'elongazione vale 0, mentre per una linea è massima.

$$M_{ect} = \frac{\sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}{\mu_{20} + \mu_{02}} . \quad (3.37)$$

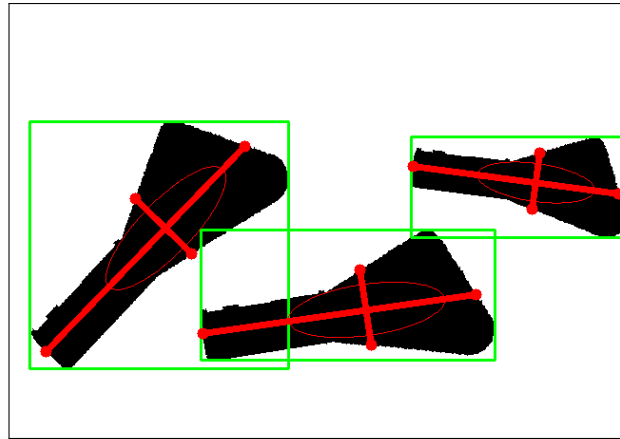
L'eccentricità è un descrittore in generale meno robusto e più variabile della compattezza.

In figura 3.22 è riportato un esempio dell'individuazione dei momenti e delle caratteristiche geometriche di oggetti disposti in modo variabile nel piano immagine.

## 2.4 Descrittori per contorni

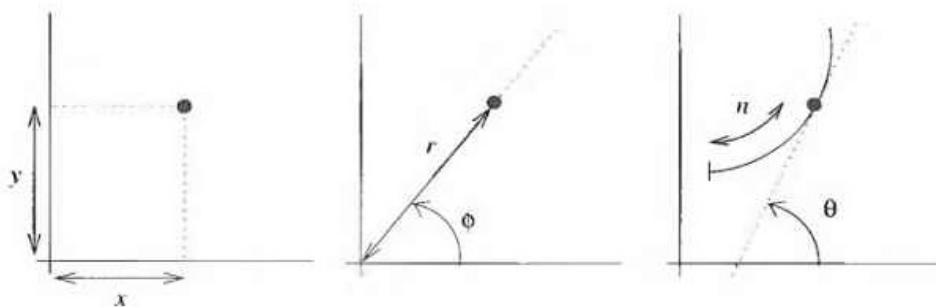
I pixel dell'immagine appartenenti ad un contorno individuato con opportuno filtraggio devono essere espressi in una formulazione matematica per poter essere memorizzati in apposite strutture dati e, successivamente, elaborati. In particolare, la figura 3.23 mostra i tre sistemi di coordinate più utilizzati:

- *Coordinate rettangolari:* pixel rappresentati in un sistema di coordinate cartesiane;



**Figura 3.22:** Applicazione dei momenti per il calcolo delle caratteristiche geometriche

- *Coordinate polari:* le classiche coordinate cartesiane vengono rappresentate in uno spazio formato da angolo e distanza;
- *Coordinate tangenziali:* punto descritto dalla distanza sul contorno del punto in esame rispetto ad un punto di partenza ( $n$ ) e dalla direzione della tangente al contorno nel punto.

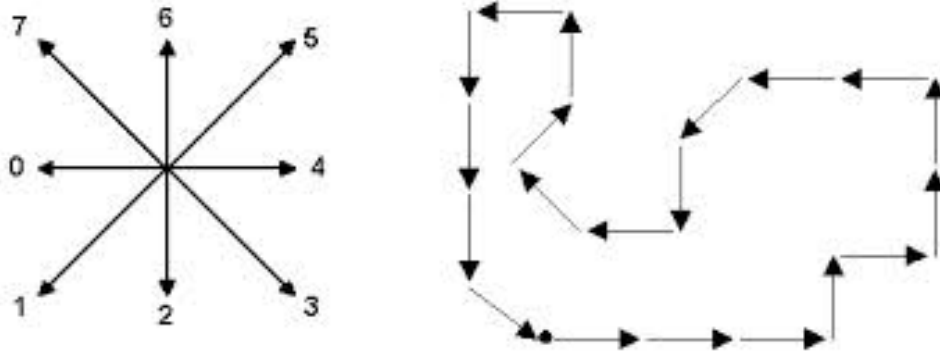


**Figura 3.23:** Sistemi di coordinate per rappresentazione dei contorni

Poiché ogni oggetto può essere caratterizzato e identificato dai punti che lo separano dallo sfondo, cioè dal suo contorno, la rappresentazione e memorizzazione di tali punti è una delle operazioni più comunemente usate in visione artificiale. Pertanto, le strutture dati utilizzate per la descrizione del contorno sono generalmente più complesse di una semplice sequenza o vettore di coordinate.

Per quanto riguarda le *coordinate cartesiane*, il contorno è descritto dal cosiddetto *chain code* (figura 3.24), che sfrutta la discretizzazione dell'immagine in righe e colonne di pixel.

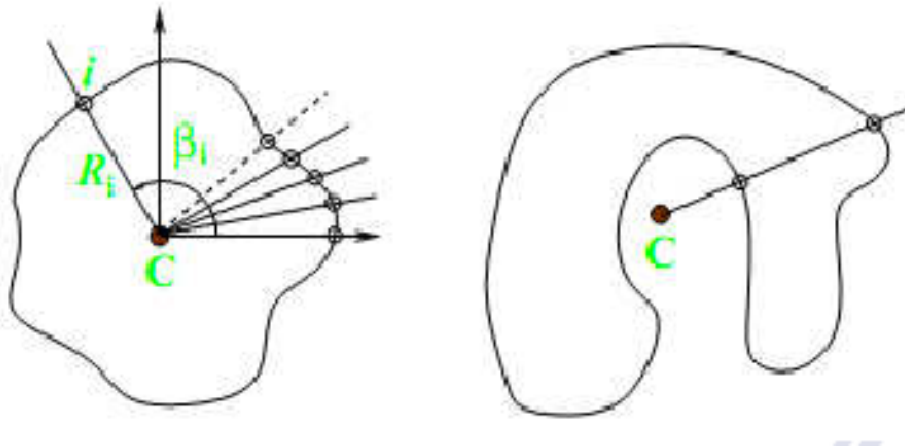
Note le coordinate del punto di partenza  $(r_0, c_0)$ , il contorno è completamente caratterizzato da una sequenza di codici che descrivono il moto da effettuare sul vicinato del pixel in



**Figura 3.24:** Rappresentazione di contorni in coordinate cartesiane: chain code

analisi per passare al pixel successivo. Il codice 0 corrisponde, ad esempio, ad un movimento verso sinistra, mentre i 7 codici successivi descrivono i movimenti sul vicinato procedendo in senso antiorario. La struttura dati ottenuta è *rigenerativa*, ovvero è sufficiente per ricostruire un contorno da zero. Eventuali traslazioni dipendono esclusivamente dalla scelta del punto di partenza.

Per quanto riguarda le *coordinate polari* (figura 3.25), il contorno risulta completamente descritto dalla *funzione radiale*  $R(\alpha)$ .



**Figura 3.25:** Rappresentazione di contorni in coordinate polari

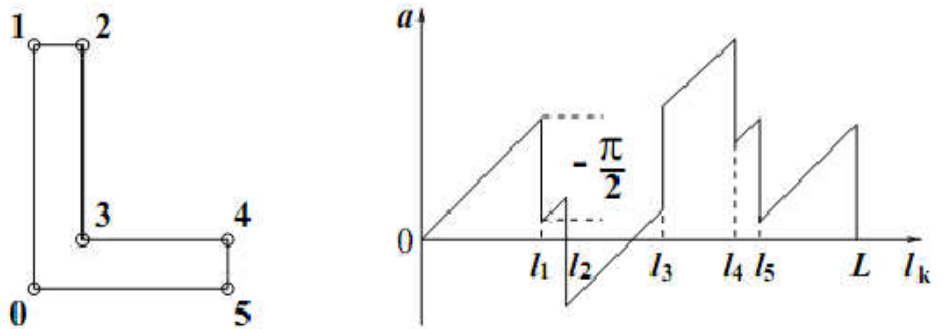
Dato un certo angolo  $\alpha$  la funzione radiale è data dalla distanza del punto del contorno  $(x, y)$  da un punto base scelto come sistema di riferimento  $(x_0, y_0)$ :

$$\begin{aligned} R &= \sqrt{(x - x_0)^2 + (y - y_0)^2} \\ \alpha &= \arctan\left(\frac{y - y_0}{x - x_0}\right) \end{aligned} \quad (3.38)$$

La rappresentazione del contorno assume significati diversi in funzione dell'ordine con cui le coppie  $(\alpha, R)$  vengono memorizzate nella struttura dati. Nel caso venga mantenuto

l'ordine dato dal chain code, le coordinate polari permettono di evidenziare la variazione di distanza tra un punto e il successivo, ovvero il grado di inclinazione del contorno. Se, invece, le coppie vengono ordinate per angolo, si potrebbe perdere la continuità della sequenza. Infatti, in presenza di concavità nel contorno, si avrebbero due o più coppie per uno stesso valore di  $\alpha$ . Tuttavia, tale rappresentazione potrebbe risultare utile per eliminare le concavità e approssimare il contorno col suo inviluppo convesso. Per tale particolarità, le coordinate polari sono efficienti per rappresentare forme convesse per cui risulta vantaggioso scegliere il punto base all'interno della forma. Le coordinate polari, infine, sono rigenerative per traslazioni, rotazioni e riflessioni.

Il sistema di coordinate più adatto a mettere in evidenza l'inclinazione del contorno è il sistema di *coordinate tangenziali*, da cui deriva la struttura *Contour Slope Sequence* (figura 3.26).



**Figura 3.26:** Rappresentazione di contorni in coordinate tangenziali: *Contour Slope Sequence*

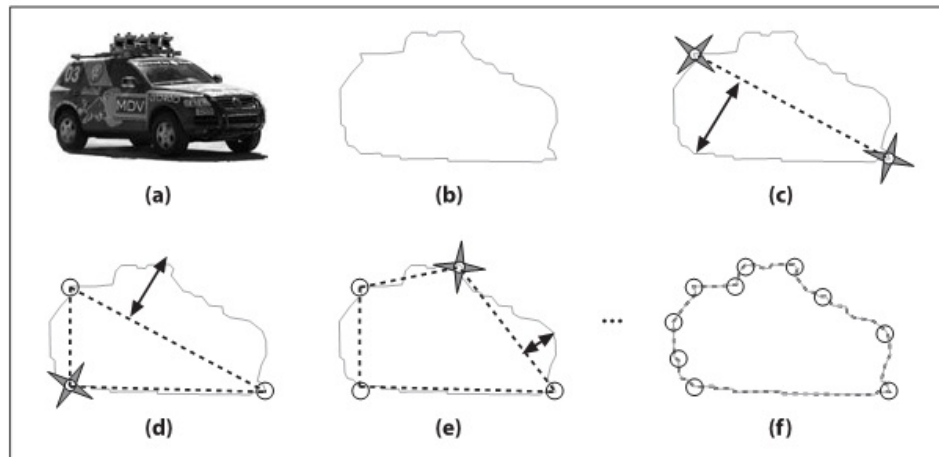
Questa struttura dati permette di rappresentare il contorno come una coppia di lunghezze ed angoli  $(l_k, \alpha_k)$ . Date le coordinate del punto di partenza  $(r_0, c_0)$ ,  $l_k$  rappresenta la lunghezza del tratto di contorno k-esimo, mentre  $\alpha_k$  ne è l'inclinazione. Detta  $L$  la lunghezza totale del contorno, le coppie angolo-lunghezza possono essere riassunte nella funzione di pendenza (*slope*)  $a_k$ :

$$a_k = \alpha_k + \frac{2\pi}{L} \cdot l_k . \quad (3.39)$$

Questa struttura dati è rigenerativa nei confronti di rotazioni e traslazioni.

### 2.4.1 Approssimazione del contorno

Per ridurre l'onere computazionale dell'elaborazione di un contorno, è possibile ridurre il numero di punti che lo compongono. Uno dei metodi più utilizzati per l'approssimazione



**Figura 3.27:** Algoritmo di approssimazione polinomiale del contorno

del contorno è l'algoritmo di *Douglas-Peucker*, il cui sviluppo è mostrato in figura 3.27.

A partire dal contorno completo si cerca di ottenerne un'approssimazione per passi successivi cercando, iterativamente, i punti dominanti. Inizialmente si prendono i due punti del contorno a massima distanza. Il punto da selezionare al passo successivo è quello che massimizza la distanza dalla retta congiungente i primi due punti scelti. Si procede in questo modo fino a quando si è ottenuta l'approssimazione desiderata.

Il parametro da scegliere per un corretto utilizzo dell'algoritmo è pertanto il numero di punti  $N$  con cui si vuole approssimare il contorno. Ad esempio, si ha:

- $N = 2$ . Contorno approssimato con una retta: si perdono tutte le informazioni relative all'area dell'oggetto, ma si può ottenere velocemente un'informazione sul suo orientamento.
- $N = \frac{\# \text{ total points}}{2}$ . Buona approssimazione.
- $N > \frac{\# \text{ total points}}{2}$ . Scarsa approssimazione. In pratica si ottiene un contorno quasi identico a quello di partenza.

### 3 Interpretazione

La fase di *interpretazione* delle immagini è la sede della vera intelligenza di ogni algoritmo di elaborazione delle immagini. Al termine della fase di descrizione si hanno a disposizione una serie di proprietà geometriche o statistiche che permettono di riassumere il contenuto delle immagini in una forma elaborabile da un sistema di intelligenza artificiale. Tuttavia, rispetto all'elaborazione delle informazioni visive da parte del cervello umano, questa

informazione è sì strettamente correlata con quanto percepibile nella scena, ma non riesce ancora a fornire un'informazione utilizzabile da un sistema.

Nel caso di applicazioni di automazione industriale, l'esempio tipico di domanda cui ci si aspetta il sistema di visione riesca a trovare una risposta è relativa alla presenza o meno di un particolare oggetto all'interno della scena. Per di più, se l'oggetto deve essere manipolato da un robot, la sola presenza dell'oggetto non è sufficiente poiché è necessario stimare anche la posa dell'oggetto nello spazio di lavoro del manipolatore.

Quello appena descritto è una tipica esemplificazione del problema del *matching*, ovvero del confronto delle osservazioni sulla scena con un *modello* di quanto ci si aspetta di vedere. Praticamente tutte le procedure di matching si suddividono in due fasi distinte: (i) la costruzione di un *modello* (*pattern*) dell'oggetto da ricercare nell'immagine e (ii) l'individuazione dell'oggetto in una scena. Generalmente il modello è costituito dalla stessa tipologia di descrittori che vengono utilizzati successivamente per l'estrazione delle informazioni dalla scena.

Il matching viene eseguito proiettando le feature del modello nello spazio dell'immagine, attraverso un'associazione con i dati dell'immagine. L'associazione consiste in generale nella formulazione di una o più ipotesi che vi sia una corrispondenza tra le feature osservate nell'immagine e le feature date dal descrittore del modello. Il risultato del processo di associazione è la 5-tupla di informazioni mostrata in (3.40), dove  $x$  è il livello dell'output,  $(z, h)$  sono le coppia di associazioni immagine-modello,  $e$  l'innovazione e  $R$  la covarianza della misura del rumore.

$$Z \equiv z, h, H, e, R_x . \quad (3.40)$$

La matrice  $H$  racchiude le misurazioni previste dello Jacobiano:

$$\mathbf{H} \equiv \frac{\delta h}{\delta s}$$

ovvero contiene informazioni riguardo la variazione delle feature nello spazio immagine.

Le misure ottenute e contenute nella 5-tupla, possono esser classificate principalmente in tre livelli di output  $x$ :

- Misurazioni sui *pixel*. Rientra in questo gruppo il matching basato su template, per cui la descrizione della scena non è data da caratteristiche di tipo geometrico o dall'individuazione di feature immagine, ma direttamente a livello dei singoli pixel.



- Insieme di *feature*. Il modello di confronto è costituito da una serie di caratteristiche tipicamente geometriche (punti, linee, aree, ...), la cui disposizione va ricercata all'interno dell'immagine.
- Livello *Oggetto*. Prevede il confronto della scena con un modello dell'oggetto indipendente dalla scena inquadrata. Lo scopo di questo confronto è l'individuazione della posa dell'oggetto nel sistema di riferimento della telecamera.

### 3.1 Template Matching

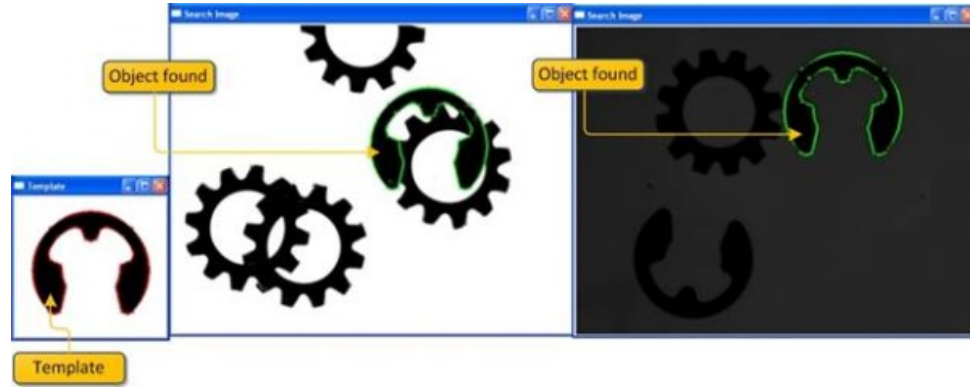
Un *template* è un modello di un oggetto tipicamente completamente texturizzato. Ciò significa che le caratteristiche che permettono di differenziare l'oggetto ricercato da altri oggetti risiedono direttamente nell'apparenza dell'oggetto stesso.

Il confronto basato su template, quindi, consiste nell'utilizzo delle informazioni della superficie di un oggetto come feature globale per il matching. Tale informazione è tipicamente calcolata a livello dei pixel, ovvero senza passare da una ricostruzione geometrica dell'oggetto in questione. L'esempio tipico è infatti costituito dall'utilizzo delle informazioni di luminosità e/o colore dei pixel, raccolte sotto forma di istogrammi o meno.

Il principale vantaggio derivante dal confronto di template risiede nel fatto di dover applicare poche o nessuna tecnica di elaborazione delle immagini per l'estrazione delle feature: ad esempio, nel caso di confronto di luminosità dei pixel è sufficiente l'informazione contenuta nell'istogramma dell'immagine. Inoltre, pur nella sua semplicità, il template contiene tutta l'informazione immagazzinabile ed estraibile da un'immagine. Tuttavia, la mancanza di una fase dedicata esclusivamente all'estrazione dall'immagine di un set limitato di caratteristiche costituisce anche il principale svantaggio del metodo. L'estrazione di un set di caratteristiche permette di sintetizzare il contenuto dell'immagine in pochi parametri, riducendo così il numero di misurazioni da confrontare per risolvere il matching. Basare il confronto sull'utilizzo dei template, al contrario, rende il matching computazionalmente oneroso, poiché vanno prese in considerazione grandi porzioni delle immagini. Sempre a causa dell'assenza della fase di estrazione e sintesi delle caratteristiche dell'immagine, l'approccio risulta poco robusto alle variazioni nelle misure di tali caratteristiche dovute al cambiamento delle condizioni ambientali della scena.

Il confronto del template consiste nel proiettare sul piano immagine l'intero template e verificarne la differenza con i pixel osservati nell'immagine della scena (figura 3.28). Il problema ha diversi gradi di libertà, poiché il template deve essere proiettato tenendo

conto di tutte le possibili trasformazioni geometriche cui può essere sottoposto l'oggetto contenuto nella scena (traslazioni, rotazioni, deformazioni, ...). Per ridurre questa complessità, è possibile *sottocampionare* l'immagine acquisita, andando a ridurre la risoluzione e il numero di punti da confrontare col template.



**Figura 3.28:** Esempio di template matching

Per la misura dell'errore del confronto, possono essere utilizzati diversi indici. Di seguito si elenca:

- Valutazione delle *differenze al quadrato*. Questo metodo associa la differenza dei quadrati (equazione (3.41)). Se il risultato, ovvero l'errore, è nullo significa che il match è perfetto altrimenti cresce al crescere delle differenze tra gli istogrammi confrontati.

$$R_{sq-diff}(x, y) = \sum_{x', y'} [T(x', y') - I(x + x', y + y')]^2 . \quad (3.41)$$

- Confronto della *correlazione*. La valutazione della correlazione (relazione (3.42)) è analoga a quanto già visto per gli istogrammi (vedi 2.1). Il match migliore è quello che presenta un risultato più alto, ovvero viene trovata una dipendenza più o meno elevata tra i due set di dati considerati.

$$R_{corr}(x, y) = \sum_{x', y'} [T(x', y') I(x + x', y + y')]^2 . \quad (3.42)$$

- *Normalizzazione*. Per ognuno dei tre metodi appena descritti, esiste una versione normalizzata. La normalizzazione in tutti i casi prevede lo stesso coefficiente:

$$Z(x, y) = \sqrt{\sum_{x', y'} T(x', y')^2 \sum_{x', y'} I(x + x', y + y')^2} . \quad (3.43)$$

Dividendo per questo fattore  $Z(x, y)$  i risultati dei metodi precedenti, si ottiene il risultato normalizzato.

Per ridurre il carico computazionale degli algoritmi di template matching è possibile rinunciare al confronto dell'informazione completa, utilizzando invece una sintesi costituita dagli istogrammi. Pertanto, il confronto si basa sulle cifre di merito per la valutazione delle differenze negli istogrammi già elencate nel paragrafo 2.1.

## 3.2 Trasformata di Hough

In generale, gli algoritmi di visione artificiale più efficienti e utilizzati per la stima della posa di un oggetto nello spazio sono quelli che presentano un approccio geometrico, ovvero permettono di riconoscere delle primitive geometriche all'interno dell'immagine.

La *trasformata di Hough* è una tecnica che permette di individuare rette, curve oppure forme predefinite presenti all'interno di un'immagine a partire dalla loro proiezione puntiforme in uno spazio dei parametri che viene definito come *Spazio di Hough*.

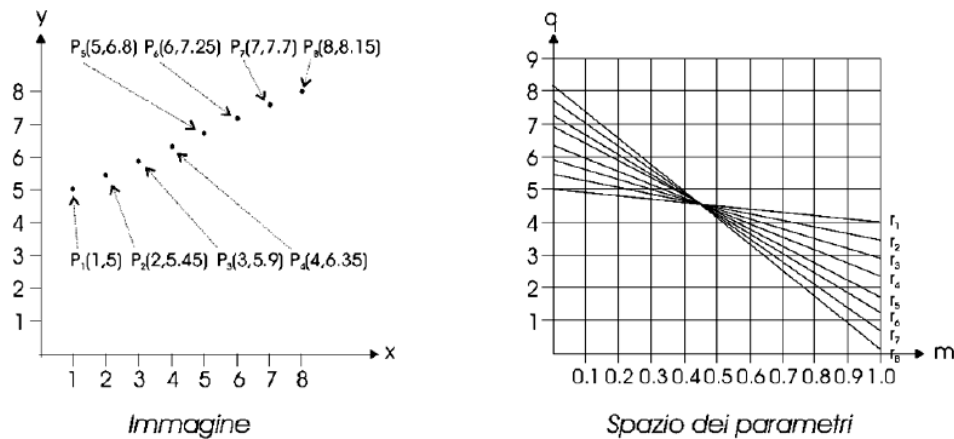
Il principio su cui si basa la trasformata di Hough è la ricerca di punti dell'immagine disposti lungo una particolare curva. Il primo utilizzo di questo metodo è la ricerca di punti allineati su una retta. In geometria analitica, una generica retta viene espressa dall'equazione (3.44), in cui i parametri  $m$  e  $q$  rappresentano rispettivamente la pendenza e l'intercetta sull'asse  $y$  della retta.

$$y - mx - q = 0 . \quad (3.44)$$

Lo spazio immagine è costituito dall'insieme di punti  $(x, y)$ , mentre lo spazio dei parametri è dato da  $(m, q)$ . Una volta fissati i parametri  $(\hat{m}, \hat{q})$ , l'equazione (3.44) può essere vista come una mappatura di tipo biunivoco dallo spazio dei parametri allo spazio dell'immagine e restituisce i valori dei punti presenti nell'immagine appartenente alla retta di parametri  $(\hat{m}, \hat{q})$ . Se, al contrario, vengono fissati i punti  $(\hat{x}, \hat{y})$  dello spazio immagine, la mappatura è diversa, ovvero identifica i parametri  $(m, q)$  che nello spazio immagine rappresentano le rette passanti per il punto  $(\hat{x}, \hat{y})$ .

Data una generica forma da individuare (nell'esempio una retta) rappresentata mediante un insieme di parametri, ogni punto dell'immagine determina una curva nello spazio dei parametri compatibili con quel punto (figura 3.29). Questo concetto può essere facilmente esteso al caso di ricerca di circonferenze, ellissi o qualsiasi tipo di curva, sostituendo l'equazione di partenza con l'equazione della curva che si intende ricercare.

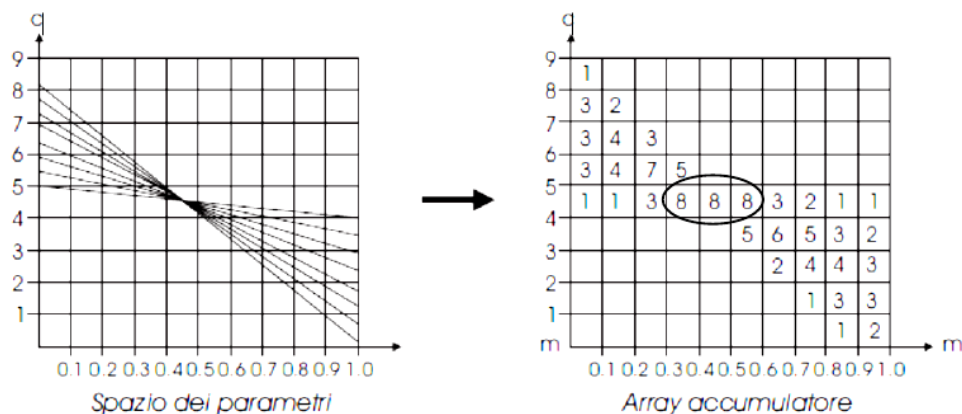
Il fatto che un elevato numero di curve presenti nello spazio dei parametri si incontrino in un determinato punto porta a concludere che un elevato numero di punti nello spazio



**Figura 3.29:** Spazio immagine e spazio dei parametri della trasformata di Hough

immagine siano disposti sulla curva ricercata.

Al fine di implementare la trasformata di Hough è necessario quantizzare in maniera adeguata lo spazio dei parametri, definendo cioè la dimensione in funzione dei suoi estremi e del passo tra un campione e il successivo. L'insieme dei campioni dello spazio dei parametri costituisce il vettore dei parametri delle curve individuabili all'interno dell'immagine in analisi (anche detto *vettore accumulatore*). Per ogni punto dello spazio immagine, si calcola il set di parametri di ogni curva che lo attraversa, si individua la cella corrispondente nel vettore accumulatore e se ne incrementa un contatore. Questo processo è la fase di *voto* in cui si contano il numero di punti dell'immagine che si dispongono su una particolare curva. Al termine dell'elaborazione, ogni cella del vettore accumulatore contiene un valore che rappresenta il numero di voti, perciò le celle con valore più elevato corrispondono a dei punti nello spazio dei parametri che definiscono curve che attraversano un elevato numero di punti. In figura 3.30 un esempio di vettore accumulatore per la trasformata di Hough lineare.



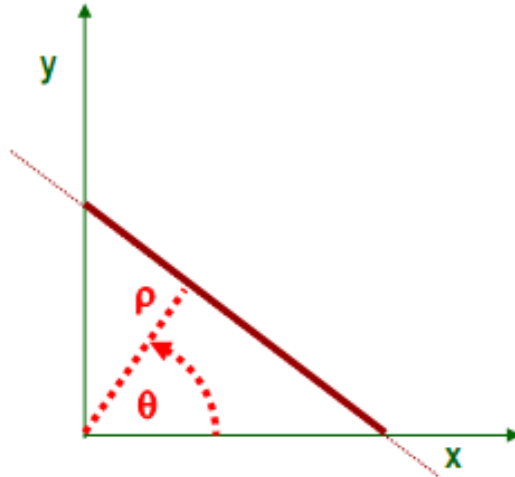
**Figura 3.30:** Vettore accumulatore per la trasformata di Hough lineare

L'utilizzo del vettore accumulatore e del processo di voto rende l'approccio della trasformata di Hough robusto alla presenza di rumore nelle immagini: è statisticamente improbabile che punti causati dal rumore si distribuiscano lungo una specifica curva.

Nel caso delle rette, la parametrizzazione (3.44) non è la più adatta per la quantizzazione dello spazio dei parametri. Infatti, entrambi i parametri possono tendere all'infinito: nel caso di rette verticali o quasi, sia la pendenza  $m$  che l'intercetta  $q$  tendono all'infinito. Pertanto, nel caso delle rette è vantaggioso passare ad una rappresentazione in coordinate polari:

$$\rho = x \cos \theta + y \sin \theta . \quad (3.45)$$

La parametrizzazione della retta in coordinate polari è mostrata in figura 3.31).



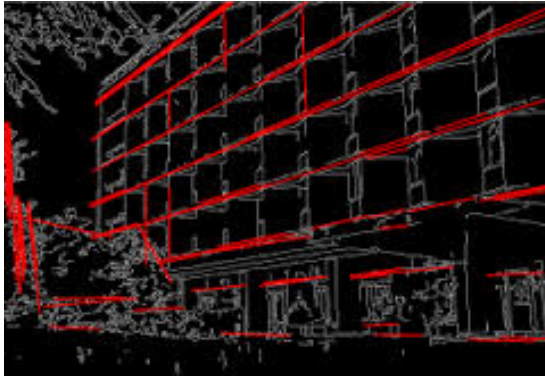
**Figura 3.31:** Coordinate polari per la rappresentazione di una retta

Così facendo, il parametro  $\theta$  risulta definito dall'intervallo  $[-\pi, \pi]$  ed il solo parametro  $\rho$  può tendere all'infinito. In realtà, lavorando direttamente nello spazio immagine, anche il parametro  $\rho$  risulta limitato dalla risoluzione dell'immagine.

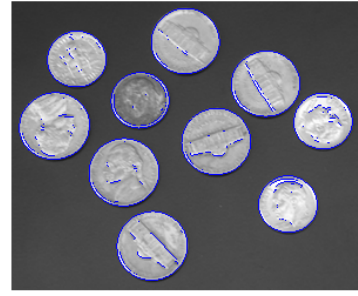
In figura 3.32 si riporta, infine, un esempio di applicazione della trasformata di Hough lineare e circolare.

### 3.2.1 Trasformata di Hough generalizzata

Per quanto presentato fino ad ora, per l'applicazione della trasformata di Hough si ha bisogno della funzione analitica della curva ricercata. In realtà, poiché questa serve unicamente alla definizione e quantizzazione dello spazio dei parametri, la trasformata di Hough lineare è estendibile alla ricerca di una qualsiasi curva opportunamente parametrizzata.



(a) Trasformata di Hough lineare.

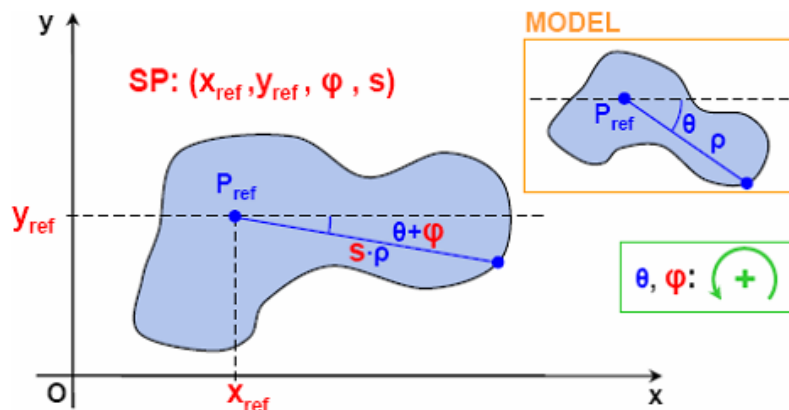


(b) Trasformata di Hough circolare.

**Figura 3.32:** Applicazione della trasformata di Hough

La *generalizzazione* della trasformata di Hough, per l'appunto, permette l'individuazione di punti disposti lungo una curva qualsiasi. Se tale curva è chiusa, si ottiene l'approccio per il matching di un *contorno* noto. Come per le procedure di matching basate su template, la procedura si suddivide in due fasi: la costruzione di un modello e l'individuazione di tale modello nelle immagini.

La fase di costruzione del modello prevede la scelta di un punto di riferimento  $(x_{ref}, y_{ref})$  che costituisce il centro del sistema di riferimento delle coordinate polari. Facendo riferimento alla figura 3.33, lo spazio dei parametri risulta definito da due soli parametri: *orientamento*  $\theta$ , *distanza*  $\rho$ . Per rendere l'identificazione robusta a variazioni di scala e orientamento degli oggetti è possibile estendere lo spazio dei parametri con due ulteriori parametri: lo *sfasamento*  $\phi$  e il *fattore di scala*  $S$ .

**Figura 3.33:** Parametrizzazione della trasformata di Hough generalizzata

Dette  $(x, y)$  le coordinate di un punto del contorno, la formulazione analitica della trasfor-

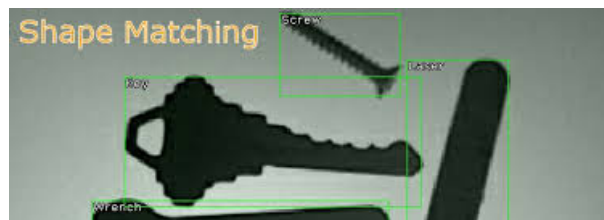
mata di Hough è data da:

$$\begin{cases} x = x_{ref} + S \cdot \rho \cdot \cos(\theta + \phi) \\ y = y_{ref} + S \cdot \rho \cdot \sin(\theta + \phi) \end{cases} \quad (3.46)$$

Nel caso di contorni chiusi e convessi una scelta conveniente per il centro del sistema di riferimento è rappresentata dal centro geometrico del contorno in esame, poiché in questo modo ogni punto ha solo un parametro di orientamento possibile. Inoltre, la scelta del centro geometrico rende semplice la proiezione del centro del sistema di riferimento sull'immagine in analisi.

Il vettore accumulatore viene costruito allo stesso modo del caso della trasformata di Hough lineare. L'unica differenza risiede nell'aumento della dimensionalità del problema, poiché la tupla che identifica una cella del vettore accumulatore è formata adesso da 4 elementi. Il limite della trasformata di Hough generalizzata risiede proprio nel compromesso tra la scelta di un passo di quantizzazione piccolo che permette l'ottenimento di maggiore precisione, ma con carico computazionale elevato, e uno maggiore che rende invece applicabile la trasformata di Hough anche in applicazioni real-time.

In figura 3.34 viene mostrato un esempio di utilizzo della trasformata di Hough generalizzata per un'applicazione di visione planare: dato un database di oggetti e la rappresentazione dei loro contorni in coordinate polari, è possibile eseguire il matching di tale contorno all'interno della scena osservata, individuando la presenza dell'oggetto, al sua scala e l'orientamento.



**Figura 3.34:** Ricerca di oggetti bidimensionali

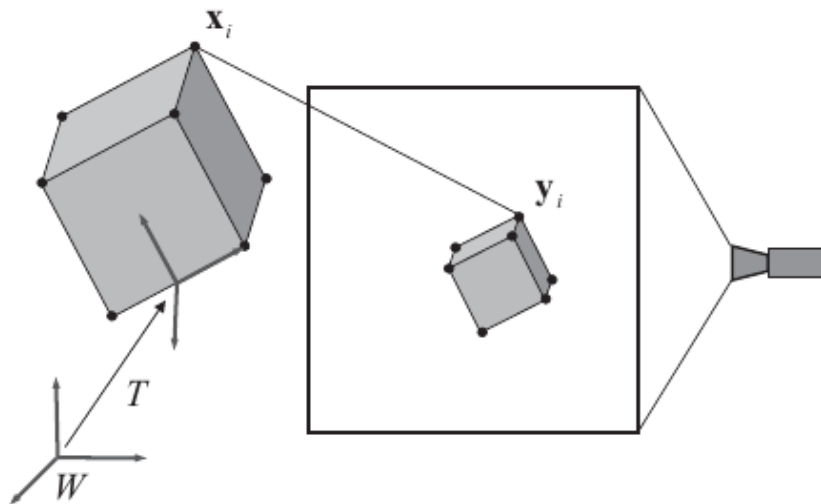
### 3.3 Stima della posa da modello 3D

Per quanto concerne l'ambito dell'automazione industriale, il compito tipico dei sistemi di visione è il riconoscimento della posa degli oggetti da manipolare. Se nel caso di oggetti fortemente bidimensionali possono essere sufficienti il calcolo dei momenti geometrici (vedi paragrafo 2.3) e le tecniche di matching viste fino ad ora, nel caso di oggetti tridimensionali

l'approccio deve necessariamente essere più complesso. La soluzione tipica prevede il confronto dell'informazione estratta dalle immagini, data dall'insieme dei punti appartenenti all'oggetto, con un modello noto a priori del medesimo oggetto.

I parametri che determinano la posa di un oggetto 3D definiscono la sua posizione nel sistema di riferimento del mondo  $W$ . Il problema della stima della posa di un oggetto inquadrato nella scena rispetto ad un modello si affronta individuando le corrispondenze tra i punti del modello e i punti della scena.

Siano  $\mathbf{x}_i$  le coordinate dei punti di un modello 3D e  $\mathbf{y}_i$  le proiezioni di questi punti su un piano immagine, come mostrato in figura 3.35. Si desidera ottenere la stima la matrice di trasformazione  $T$  che permette di passare dal modello tridimensionale alla sua proiezione osservata a partire dall'individuazione di  $N$  corrispondenze tra i due set di punti. Questo obiettivo può essere raggiunto attraverso la minimizzazione di un errore per mezzo dei metodi dei *minimi quadrati* (vedi capitolo 3).



**Figura 3.35:** Corrispondenze per la stima della posa 3D

Il problema può essere formulato in termini di due diversi errori da minimizzare che portano a soluzioni in coordinate omogenee o non-omogenee:

- *Errori geometrici.* Corrispondono alle distanze euclidee calcolate direttamente sul piano immagine. La funzione obiettivo è spesso non lineare e la sua soluzione richiede una buona scelta iniziale per convergere ad un minimo globale.
- *Errori algebrici.* Sono formulati utilizzando equazioni omogenee e forniscono residui algebrici ai quali non corrisponde una particolare interpretazione geometrica. Di conseguenza, la minimizzazione di un errore algebrico potrebbe non corrispondere ad



una stima di massima verosimiglianza della posa 3D. D'altro canto, i minimi quadrati sull'errore algebrico è un problema lineare, quindi risolvibile in un solo passo, e rappresenta una buona scelta iniziale per affrontare il problema della minimizzazione dell'errore geometrico.

### 3.3.1 Errore geometrico

Il problema è posto nei seguenti termini: dati  $N$  punti del modello  $\mathbf{x}_i = (x_i^1, x_i^2, x_i^3, 1)^T$  e le loro corrispondenze nei punti della scena, ad esempio sul piano immagine,  $\mathbf{y}_i = (y_i^1, y_i^2)$ , trovare la trasformazione ottima  $T_{geo}^*$  tale da rispettare la relazione

$$T_{geo}^* = \arg \min \sum_{i=1}^N \|\pi(p \cdot T \cdot \mathbf{x}_i) - \mathbf{y}_i\|^2, \quad (3.47)$$

dove  $P$  è la matrice di proiezione di punti tridimensionali sul piano immagine

$$P = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

e  $\pi$  permette di convertire le coordinate da omogenee a non-omogenee:

$$\pi(z^1, z^2, z^3) = [z^1/z^3, z^2/z^3]^T. \quad (3.48)$$

La soluzione viene ottenuta attraverso il metodo dei *minimi quadrati non lineari* (*Non-Linear LSE*), a partire da una trasformazione  $T_0$  presa come scelta iniziale e sufficientemente vicina a  $T_{geo}^*$  per assicurare la convergenza dell'algoritmo. Tale scelta iniziale è spesso data dalla soluzione della minimizzazione dell'errore algebrico. Nel caso il problema consista nella stima di trasformazioni 2D-2D o 3D-3D, l'errore geometrico è lineare e può essere risolto senza l'utilizzo dell'errore algebrico.

### 3.3.2 Errore algebrico

Utilizzando le coordinate omogenee, il problema della minimizzazione dell'errore geometrico diviene la stima della trasformazione  $T^*$  tale per cui è vera la relazione

$$P \cdot T^* \cdot \mathbf{x}_i = \mathbf{y}_i, \quad (3.49)$$

dove  $\mathbf{x}_i$  sono le coordinate del punto del modello e  $\mathbf{y}_i$  i punti corrispondenti nella scena. Come si può vedere, la formulazione in termini omogenei mantiene l'ambiguità del fattore

di scala dovuto alla proiezione dei punti su un piano immagine. Questo problema sparisce se sia la scena che il modello sono rappresentati tramite dati tridimensionali.

L'equazione 3.49 può essere espressa per mezzo del prodotto vettoriale

$$\mathbf{y}_i \times (P \cdot T^* \cdot \mathbf{x}_i) = 0 . \quad (3.50)$$

Come mostrato nella (3.51), la trasformazione  $T$  può essere stimata tramite minimizzazione:

$$T_{alg}^* = \arg \min \sum_{i=1}^N \|\mathbf{y}_i \times (P \cdot T \cdot \mathbf{x}_i)\|^2 . \quad (3.51)$$

Infine, se  $T$  è parametrizzata in modo lineare, l'intero problema è lineare. La soluzione ottimale è ottenuta in un solo passo tramite la scomposizione della matrice ai valori singolari ( $SVD$ ) e il metodo risultante è noto col nome di *Trasformazione Lineare Diretta* ( $DLT$ , *Direct Linear Transform*).

**Scomposizione ai valori singolari.** Relativamente al problema della stima dell'omografia che permette di proiettare un modello tridimensionale su un piano immagine, la *scomposizione ai valori singolari* può essere applicata alla scomposizione della trasformazione omografica per metterne in evidenza le rotazioni e traslazioni coinvolte. Il problema è riassunto dalla relazione

$$\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \mathbf{R} + \mathbf{t}\mathbf{n}^T . \quad (3.52)$$

La matrice diagonale degli autovalori  $\mathbf{\Lambda}$  può essere a sua volta rappresentata come composizione di una matrice di rotazione e un vettore traslazione:

$$\mathbf{\Lambda} = d_{\Lambda}\mathbf{R}_{\Lambda} + \mathbf{t}_{\Lambda}\mathbf{n}_{\Lambda}^T . \quad (3.53)$$

Con le opportune sostituzioni si ottiene la definizione delle matrici di rotazione e traslazione dell'omografia

$$\begin{cases} \mathbf{R} = \mathbf{U}\mathbf{R}_{\Lambda}\mathbf{V}^T \\ \mathbf{t} = \mathbf{U}\mathbf{t}_{\Lambda} \\ \mathbf{n} = \mathbf{V}\mathbf{n}_{\Lambda} = \mathbf{n}_{\Lambda}^T\mathbf{V}^T \end{cases} \quad (3.54)$$

dove  $\mathbf{n}$  è un vettore con norma unitaria,  $\mathbf{U}$  e  $\mathbf{V}$  sono matrici ortogonali.

Usando la base canonica  $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ , ovvero  $\mathbf{e}_1 = [1 \ 0 \ 0]^T$ , e definendo  $\mathbf{n}_{\Lambda}$  come composizione di vettori  $\mathbf{n} = \mathbf{x}_1\mathbf{e}_1 + \mathbf{x}_2\mathbf{e}_2 + \mathbf{x}_3\mathbf{e}_3 = [x_1 \ x_2 \ x_3]^T$  è possibile riscrivere

l'equazione  $i$ -sima derivante da (3.53), ricavando la (3.55), dove  $d_i$  è l' $i$ -simo autovalore della matrice  $\mathbf{\Lambda}$ , ordinati in modo da avere  $d_1 > d_2 > d_3$ .

$$d_i \mathbf{e}_i = \mathbf{R}_{\mathbf{\Lambda}} \mathbf{e}_i + \mathbf{t}_{\mathbf{\Lambda}} x_i \quad , \quad \text{per } i = 1, 2, 3 . \quad (3.55)$$

Dato che  $\mathbf{n}$  ha norma unitaria e  $\mathbf{V}$  ha norma unitaria, dalla relazione sui vettori normali in (3.54) segue che anche  $\mathbf{n}_{\mathbf{\Lambda}}$  ha norma unitaria, pertanto  $\sum_{i=1}^3 x_i^2 = 1$ . Pertanto, è possibile eliminare  $\mathbf{t}_{\mathbf{\Lambda}}$  attraverso una combinazione lineare dell'equazione (3.55)  $i$ -sima e  $j$ -sima:

$$d_{\mathbf{\Lambda}} \mathbf{R}_{\mathbf{\Lambda}} (x_j \mathbf{e}_i - x_i \mathbf{e}_j) = d_i x_j \mathbf{e}_i - d_j x_i \mathbf{e}_j \quad i \neq j . \quad (3.56)$$

Poiché  $\mathbf{R}_{\mathbf{\Lambda}}$  conserva la norma del vettore, si ottiene il seguente set di equazioni:

$$\begin{cases} (d_{\mathbf{\Lambda}}^2 - d_2^2)x_1^2 + (d_{\mathbf{\Lambda}}^2 - d_1^2)x_2^2 = 0 \\ (d_{\mathbf{\Lambda}}^2 - d_2^2)x_2^2 + (d_{\mathbf{\Lambda}}^2 - d_3^2)x_3^2 = 0 \\ (d_{\mathbf{\Lambda}}^2 - d_1^2)x_3^2 + (d_{\mathbf{\Lambda}}^2 - d_3^2)x_1^2 = 0 \end{cases}$$

che costituisce un sistema di equazioni lineari nelle incognite  $x_1^2$ ,  $x_2^2$  e  $x_3^2$ . Dato che si ricerca una soluzione non nulla, il determinante del sistema deve essere nullo:

$$(d_{\mathbf{\Lambda}}^2 - d_1^2)(d_{\mathbf{\Lambda}}^2 - d_2^2)(d_{\mathbf{\Lambda}}^2 - d_3^2) = 0 .$$

Le soluzioni  $d_{\mathbf{\Lambda}} = \pm d_1$  e  $d_{\mathbf{\Lambda}} = \pm d_3$  non sono ammesse perché portano ad una soluzione nulla.

Se  $d_1 \neq d_3$ , la soluzione è data dal sistema

$$\begin{cases} x_1 = \epsilon_1 \sqrt{\frac{d_1^2 - d_2^2}{d_1^2 - d_3^2}} \\ x_2 = 0 \\ x_3 = \epsilon_3 \sqrt{\frac{d_2^2 - d_3^2}{d_1^2 - d_3^2}} \end{cases} , \quad \epsilon_1, \epsilon_3 = \pm 1 , \quad (3.57)$$

dove  $d_{\mathbf{\Lambda}}$  permette di specificare il verso dell'osservazione e, quindi, rende possibile la gestione delle trasformazioni di riflessione dell'omografia. Di seguito vengono mostrate le soluzioni per il caso  $d_{\mathbf{\Lambda}} > 0$ , comunque analogo al caso  $d_{\mathbf{\Lambda}} < 0$ . La soluzione può essere suddivisa in differenti casi, a seconda della molteplicità degli autovalori della matrice:

- $d_1 \neq d_2 \neq d_3$  e  $d_\Lambda = \pm d_\Lambda$ . In riferimento all'equazione (3.55), si ha  $\mathbf{R}_\Lambda \mathbf{e}_2 = \mathbf{e}_2 \mathbf{R}_\Lambda$ , ovvero  $\mathbf{R}_\Lambda$  definisce una rotazione attorno all'asse di rotazione  $\mathbf{e}_2$ :

$$\mathbf{R}_\Lambda = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}.$$

Dalle equazioni (3.57), (3.55) e (3.56) è possibile ottenere la soluzione:

$$\begin{cases} \sin(\theta) = (d_1 - d_3) \frac{x_1 x_3}{d_2} \\ \cos(\theta) = \frac{d_1 x_3^2 + d_3 x_1^2}{d_2} \\ \mathbf{t}_\Lambda = (d_1 - d_3) \begin{bmatrix} x_1 \\ 0 \\ -x_3 \end{bmatrix} \end{cases}.$$

- $d_1 = d_2 \neq d_3$  o  $d_1 \neq d_2 = d_3$ , con  $d_\Lambda = \pm d_\Lambda$ . Si ottiene  $x_1 = x_2 = 0$  e  $x_3 = \pm 1$ , ottenendo la seguente soluzione:

$$\begin{cases} \mathbf{R}_\Lambda = \mathbf{I} \\ \mathbf{t}_\Lambda = (d_3 - d_1) \mathbf{n}_\Lambda \end{cases}.$$

- $d_1 = d_2 = d_3$  e  $d_\Lambda = \pm d_\Lambda$ . La soluzione è indefinita e pertanto si ha:

$$\begin{cases} \mathbf{R}_\Lambda = \mathbf{I} \\ \mathbf{t}_\Lambda = 0 \end{cases}.$$

**Trasformazione lineare diretta.** Nel caso di stima della posa da modello 3D ad osservazione 2D,  $\mathbf{T}$  è una matrice  $4 \times 4$  e  $\mathbf{P}$  la matrice di proiezione  $3 \times 4$ , anche detta omografia. Il problema è la stima dell'omografia  $\mathbf{H}$  a partire da un set di punti di un modello  $\mathbf{X}_i = [x_1 \ x_2 \ x_3 \ 1]^T$  e dei corrispondenti punti proiettati sul piano immagine  $\mathbf{Y}_i = [y_1 \ y_2 \ 1]^T$ . Di conseguenza, il problema qui trattato è del tutto simile al problema già affrontato (vedi paragrafo 3.2, capitolo 2) per la calibrazione della telecamera, ma qui lo scopo non è l'individuazione dei parametri della telecamera, ma della trasformazione rigida che permette il posizionamento dei punti osservato nella scena.

Detta  $h_i$  l'i-sima riga della matrice  $\mathbf{H}$ , il problema è descritto dall'equazione

$$\mathbf{Y}_i = \mathbf{H} \mathbf{X}_i = \begin{bmatrix} h_1 \mathbf{X}_i \\ h_2 \mathbf{X}_i \\ h_3 \mathbf{X}_i \end{bmatrix}. \quad (3.58)$$

Affinché l'equazione (3.58) sia verificata, è necessario che il prodotto vettoriale tra i punti rilevati sulla proiezione e la proiezione del modello tridimensionale sia nullo, ovvero:

$$\mathbf{Y}_i \times \mathbf{H}\mathbf{X}_i = \mathbf{0} .$$

Sviluppando i conti, si ottiene:

$$\begin{bmatrix} y_2 h_3 \mathbf{X}_i - y_3 h_2 \mathbf{X}_i \\ -y_1 h_3 \mathbf{X}_i + y_3 h_1 \mathbf{X}_i \\ y_1 h_2 \mathbf{X}_i - y_2 h_1 \mathbf{X}_i \end{bmatrix} = \mathbf{0} .$$

Infine, poiché  $h_j \mathbf{X}_i = \mathbf{X}_i^T h_j$ , si ricava il sistema di equazioni lineari

$$\begin{bmatrix} \mathbf{0}^T & -y_3 \mathbf{X}_i^T & y_2 \mathbf{X}_i^T \\ y_3 \mathbf{X}_i^T & \mathbf{0}^T & -y_1 \mathbf{X}_i^T \\ -y_2 \mathbf{X}_i^T & y_1 \mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} h_1^T \\ h_2^T \\ h_3^T \end{bmatrix} = 0 \quad (3.59)$$

con le righe di  $\mathbf{H}$  come incognite.

Tramite la scomposizione ai valori singolari è possibile trovare la soluzione del sistema di equazioni lineari e, in un passo successivo, scomporre la matrice di omografia nelle sue componenti di rotazione e traslazione.

Per approfondimenti sull'utilizzo e analisi della *Direct Linear Transform* si sono consultati [43] e [2].

## 4 Finestramento

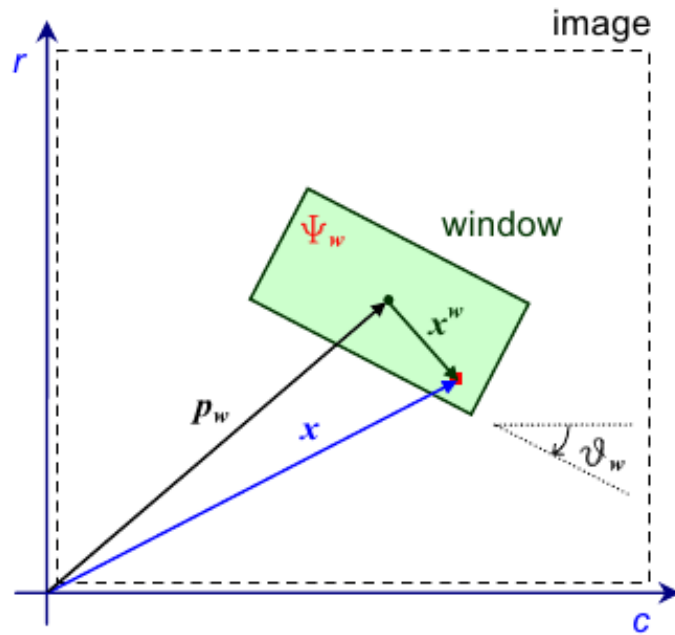
Sotto il termine *finestramento* (*windowing*) possono essere raccolte tutte le tecniche che permettono di suddividere e scomporre le immagini in piccole sottoparti. All'interno della sequenza di elaborazione delle immagini di figura 3.1, il finestramento occupa generalmente la prima posizione. Esso, infatti, viene implementato nella fase di pre-processamento dell'immagine, ma, al contrario delle operazioni di filtraggio, non è presente in tutti gli algoritmi di elaborazione delle immagini. L'utilizzo principale, infatti, lo si trova nell'elaborazione di un flusso di immagini per il quale, ad ogni iterazione dell'algoritmo di visione principale, è disponibile un'informazione a priori relativa alla posizione dei punti o delle aree individuate al passo di elaborazione precedente. Di conseguenza, l'iterazione successiva dell'algoritmo può essere resa meno onerosa restringendo l'area di ricerca delle caratteristiche dell'immagine ad un intorno dell'area o del punto già individuati. Infatti,

per estrarre le caratteristiche dalle immagini seguenti non è necessario analizzare l'intera immagine, ma solo una porzione circoscritta della stessa.

La diminuzione dell'onere computazionale dato dal partizionamento dell'immagine è direttamente proporzionale al rapporto tra la dimensione della finestra estratta e la dimensione dell'immagine di partenza. Pertanto, i risultati migliori si ottengono se la finestra viene creata attorno a feature locali, come angoli e buchi.

## 4.1 Descrizione della finestra

Le coordinate dei pixel dell'immagine sono descritti tramite il noto sistema di riferimento immagine:  $\mathbf{x} = [r \ c]^T = [u \ v]^T$  per l'immagine  $I(\mathbf{x}, t)$  catturata al tempo  $t$ . Una finestra dell'immagine è costituita da un insieme di pixel sottoposti ad una trasformazione geometrica bidimensionale che permette di passare dal sistema di riferimento dell'immagine al sistema di riferimento della finestra, come mostrato in figura 3.36.



**Figura 3.36:** Sistema di riferimento di una finestra dell'immagine

La trasformazione è rigida e completamente definita dal vettore traslazione  $\mathbf{p}_w = [r_w \ c_w]^T$  e dalla rotazione  $\theta_w$ . Questa trasformazione è uno dei parametri che caratterizza la finestra insieme alla dimensione in pixel  $S = W \times H$ .

L'algoritmo di *inseguimento* delle feature opera in due passi consecutivi. Durante la prima fase, vengono recuperate tutte le finestre calcolate al passo precedente (l'intera immagine se il finestramento deve ancora essere inizializzato) e tutti i pixel al loro interno vengono

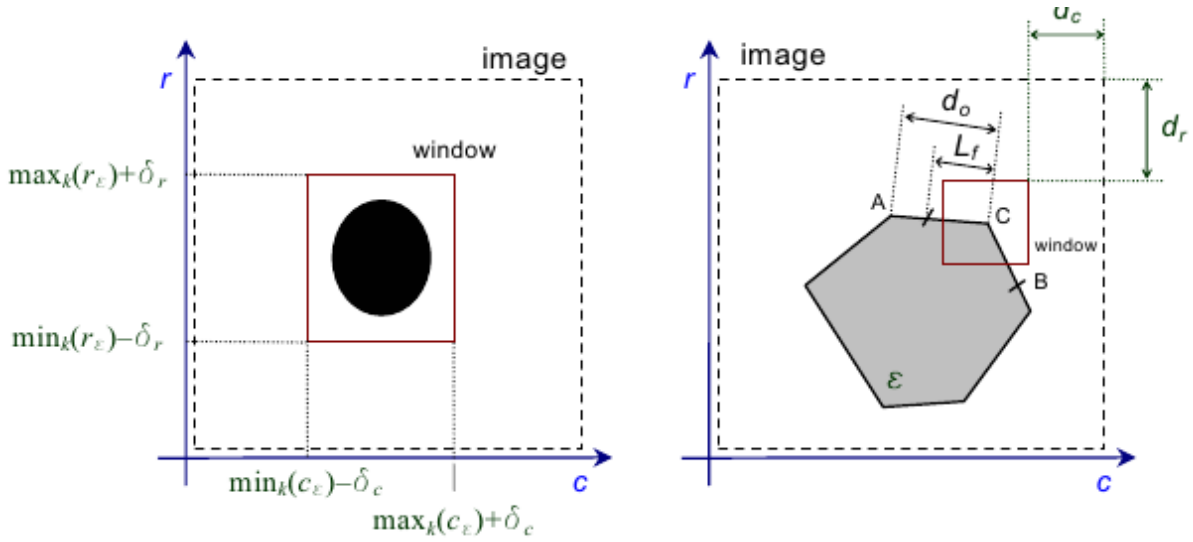
proiettati in un'immagine 2D per la successiva elaborazione. In questo modo, il contenuto dell'immagine viene suddiviso in più immagini. Il secondo passo prevede appunto la localizzazione delle feature d'interesse all'interno di ognuna delle sotto-immagini ottenute. Le feature vengono memorizzate e le fasi successive (ad esempio il matching) proseguono come nel caso dell'elaborazione sull'immagine intera senza finestramento. Infine, la posizione della feature nel sistema di riferimento dell'immagine viene utilizzata per aggiornare la posizione e orientamento della finestra nel passo successivo dell'elaborazione. L'aggiornamento si basa essenzialmente sulla predizione del moto della feature, in modo da cercare di capire quale sarà la sua posizione nell'immagine acquisita al passo successivo d'elaborazione. In letteratura sono stati proposti diversi approcci per predire il movimento di una feature all'interno di immagini. Tra le tecniche più utilizzate si possono elencare il *filtro di Kalman*, i *filtri  $\alpha$ - $\beta$*  e modelli *auto-regressivi* (*AR* o *ARX*).

Apparentemente, il fatto di avere a che fare con l'elaborazione di più immagini potrebbe sembrare uno svantaggio. In realtà, tale elaborazione risulta più semplice di quella che andrebbe applicata all'intera immagine per vari motivi:

- La finestra viene costruita nell'intorno della feature ricercata, quindi in ogni finestra va ricercata, idealmente, un'unica feature e non un intero set.
- La ricerca della feature viene effettuata su un'immagine di dimensioni ridotte, con vantaggi sia per quanto riguarda la robustezza della ricerca (ridotta possibilità di trovare outlier) sia per le risorse necessarie all'elaborazione.
- Salvandosi la storia delle rotazioni e traslazioni applicate per l'aggiornamento delle finestre, è possibile aggiungere un'ulteriore variabile all'algoritmo di identificazione delle caratteristiche: il trend di aggiornamento della finestra è strettamente collegato al movimento effettuato dalla feature stessa all'interno dell'immagine.

## 4.2 Calcolo delle feature locali

Come accennato nell'introduzione, le tecniche di finestramento sono particolarmente efficaci nell'inseguimento di feature locali perché per la loro identificazione è sufficiente una finestra di piccole dimensioni. In questo caso, la finestra contiene solo una feature ed è pertanto centrata sul centro della feature stessa. Inoltre, poiché angoli e buchi sono caratteristiche senza orientamento, l'orientamento della finestra è nullo. Di conseguenza, l'unico parametro da scegliere rimane la dimensione della finestra, con l'unico obiettivo di contenere una porzione sufficientemente grande dell'immagine.



**Figura 3.37:** Scelta della dimensione della finestra per una feature locale

Data un'approssimazione del contorno di un buco  $\epsilon$ , come in figura 3.37, la finestra risulta centrata in

$$\begin{cases} r_w = \frac{1}{2} (\max_k(r_\epsilon^k) + \min_k(r_\epsilon^k)) \\ c_w = \frac{1}{2} (\max_k(c_\epsilon^k) + \min_k(c_\epsilon^k)) \end{cases} \quad (3.60)$$

con dimensione

$$\begin{cases} W_w = \max_k(r_\epsilon^k) - \min_k(r_\epsilon^k) + 2\sigma_r \\ H_w = \max_k(c_\epsilon^k) - \min_k(c_\epsilon^k) + 2\sigma_c \end{cases}, \quad (3.61)$$

dove  $(r_w, c_w)$  sono le coordinate del centro della finestra,  $(r_\epsilon^k, c_\epsilon^k)$  le coordinate del  $k$ -simo punto della feature  $\epsilon$  e  $(\sigma_r, \sigma_c)$  permettono di definire quanto l'area della finestra deve essere maggiore rispetto all'area della feature. La scelta di  $(\sigma_r, \sigma_c)$  definisce univocamente la finestra e dipende principalmente dal massimo spostamento della feature osservabile da un frame acquisito al successivo. Finestre più grandi permettono di seguire movimenti più ampi delle feature, ma questa scelta può causare un'eccessiva sovrapposizione con finestre generate da altre feature.

Per quanto riguarda la scelta della dimensione di una finestra per un angolo, invece, è possibile stabilire un criterio di dimensionamento. Detta  $d_o$  la distanza dell'angolo  $\epsilon_k$  dall'angolo  $\epsilon_j$  più vicino ( $d_o = \|\epsilon_k - \epsilon_j\|$ ), è possibile definire una distanza di sicurezza tale da evitare l'eccessiva sovrapposizione con le finestre di altre feature:

$$L_s = \frac{d_o}{C_f}$$



dove  $C_f > 1$  è un fattore di scala. A questo punto, data la feature  $\epsilon_k$  e i bordi che la collegano alle feature contigue, è possibile calcolare le distanze su righe e colonne da due punti  $p_m = (r_m, c_m)$  e  $p_l = (r_l, c_l)$  posizionati sul bordo a distanza  $L_s$ . Il parametro per il dimensionamento della finestra  $\Delta$  è la massima distanza così ottenuta

$$\Delta = \max(|r_k - r_m|, |r_k - r_l|, |c_k - c_m|, |c_k - c_l|) . \quad (3.62)$$

La finestra risultante è centrata sul centro della feature  $(r_k, c_k)$  e ha dimensione  $2\Delta$ .

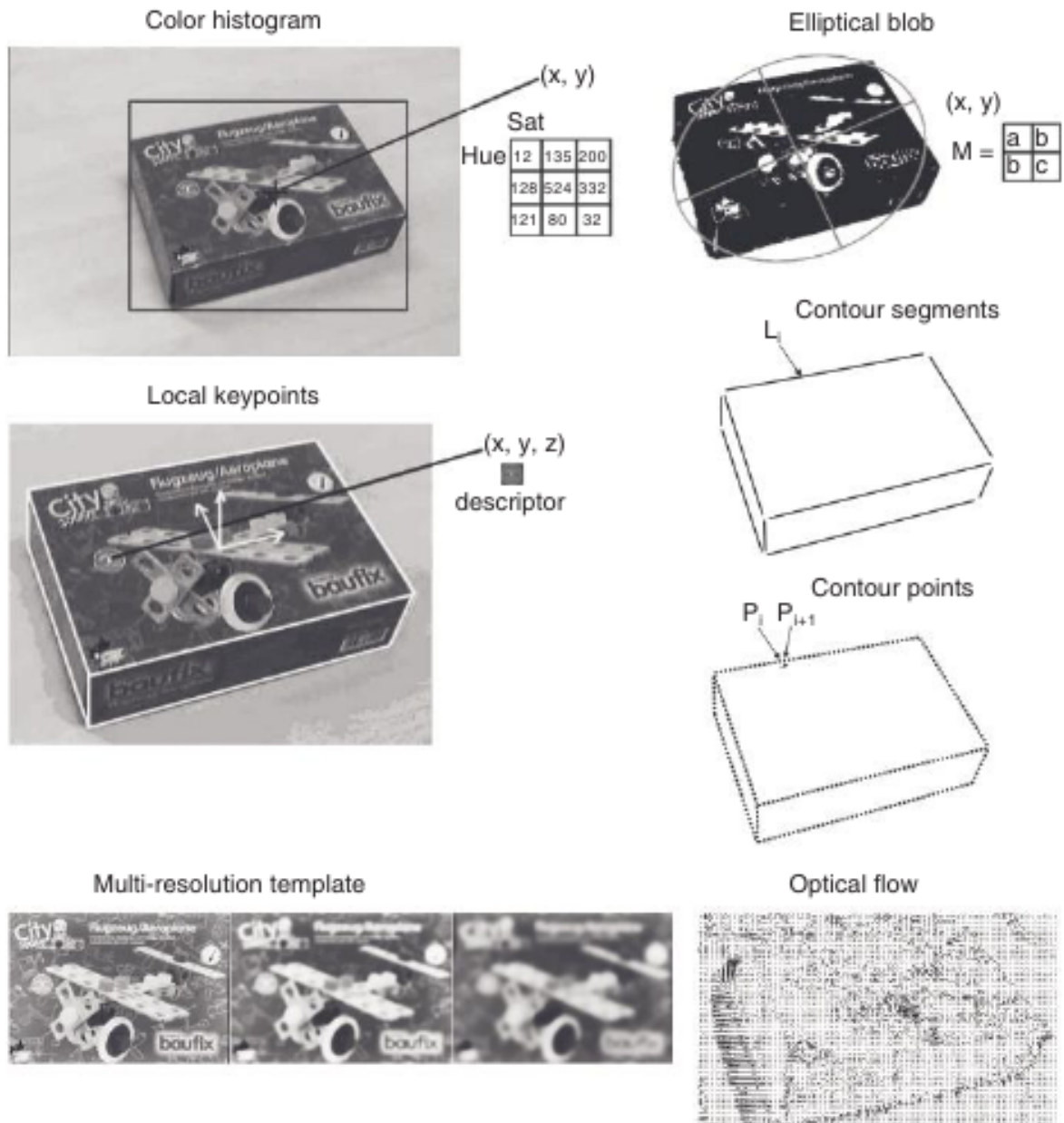
Poiché l'orientamento della finestra è nullo, questa scelta di  $\Delta$  è sufficiente a garantire l'assenza di sovrapposizioni. Infatti, il caso peggiore è quello in cui l'angolo relativo tra le feature  $\epsilon_j$  e  $\epsilon_k$  è multiplo di  $90^\circ$ : in questo caso, il bordo sinistro della finestra della prima feature risulta sovrapposto al bordo destro della seconda feature.

## 5 Conclusioni

In questo capitolo è stata analizzata la sequenza di operazioni da effettuare per ridurre l'informazione contenuta in un'immagine ad un insieme ridotto di caratteristiche facilmente elaborabili da un calcolatore. Tipicamente si tratta di proprietà geometriche relative al sottoinsieme di pixel individuato.

In particolare, sono state descritte tre diverse tipologie di segmentazione che possono essere utilizzate in condizioni operative differenti. La segmentazione basata su aree è efficiente nel caso in cui la silhouette degli oggetti all'interno della scena è più che sufficiente per la loro descrizione. L'individuazione della silhouette è semplice nel caso di scene in cui la luminosità degli oggetti in primo piano è nettamente diversa da quella dello sfondo. Una simile assunzione è alla base anche della segmentazione basata su contorni, ma in questo caso l'individuazione dei pixel è possibile anche in caso di mancanza di una separazione netta tra sfondo e oggetti in primo piano. Infatti, l'informazione che ne permette la distinzione passa dall'essere assoluta (la differenza di luminosità tra una regione e l'altra) ad essere locale (il gradiente di luminosità tra un pixel e i suoi vicini). Infine, la segmentazione basata su punti sfrutta l'individuazione di pixel con un alto contenuto d'informazione. Questi punti possono avere sia una componente geometrica come nel caso dell'individuazione degli angoli di un contorno, sia una componente puramente statistica. Questo è l'approccio più utilizzato nel caso di individuazione di oggetti fortemente texturizzati, ovvero caratterizzati dalla presenza di etichette, disegni o zone di colore univoci.

Tutte le tipologie di segmentazione descritte sono particolarmente efficienti dal punto di vista computazionale perché la fase di filtraggio è costituita da una sola operazione puntuale su tutti i pixel dell'immagine. Al termine della fase di filtraggio, la complessità dell'immagine risulta già ridotta, in modo tale da rendere possibile applicare le metodologie per il calcolo dei descrittori della scena. In figura 3.38 vengono mostrati alcuni dei descrittori di cui si è parlato nel capitolo.



**Figura 3.38:** Feature di un'immagine

Per l'analisi dell'ultima fase, l'interpretazione, si è accennato alla tecnica che prevede il confronto della scena osservata con un pattern noto. Lo scopo dell'analisi è stato mo-

strare la complessità computazionale dell'approccio presentato, in modo da evidenziarne l'inutilizzabilità in un algoritmo per l'individuazione di oggetti real-time.

Infine, sono state presentate alcune considerazioni su come è possibile ridurre la difficoltà e il carico computazionale dell'elaborazione tramite l'utilizzo del finestramento dell'immagine. Nel capitolo 6 viene mostrato come il finestramento ed altre tecniche possano venire utilizzate per estendere gli algoritmi di base per l'interpretazione delle immagini, in modo da renderli usabili anche per il tracciamento real-time di oggetti in movimento all'interno della scena.

Gli algoritmi presentati costituiscono la base dell'elaborazione dell'immagine, ma essi sono puramente bidimensionali. Infatti, l'elaborazione e i risultati prodotti fanno riferimento al solo piano immagine, senza prendere in considerazione la presenza della terza coordinata dello spazio: la profondità. Nel capitolo 4 vengono mostrati degli approcci all'elaborazione di dati tridimensionali e ne viene mostrata la loro utilità nel descrivere scene o oggetti più complessi.



# Capitolo 4

## Acquisizione ed Elaborazione di Immagini 3D

### Indice

---

1	Classificazione dei compiti . . . . .	<b>129</b>
1.1	Ricostruzione di oggetti tridimensionali . . . . .	130
1.2	Riconoscimento . . . . .	130
1.3	Comprensione, segmentazione e tracking . . . . .	131
2	Acquisizione e rappresentazione dei dati . . . . .	<b>131</b>
2.1	Acquisizione di informazioni tridimensionali . . . . .	132
2.2	Strutture per la rappresentazione di dati 3D . . . . .	138
3	Ricostruzione e Descrittori 3D . . . . .	<b>142</b>
3.1	Registrazione . . . . .	142
3.2	Caratteristiche geometriche . . . . .	144
3.3	Descrittori 3D . . . . .	149
4	Riconoscimento . . . . .	<b>159</b>
4.1	Training . . . . .	160
4.2	Riconoscimento tramite descrittori locali . . . . .	161
4.3	Riconoscimento tramite descrittori globali . . . . .	163
4.4	Postprocessing: affinamento dei risultati . . . . .	165
4.5	Stima della posa dal modello 3D . . . . .	166
5	Conclusioni . . . . .	<b>166</b>

---

Le classiche tecniche di elaborazione delle immagini bidimensionali permettono di aggiungere importanti fonti d'informazione per svariati tipi di applicazioni. Tuttavia,

esse presentano alcune limitazioni tecnologiche dovute alla difficoltà di dover interpretare un mondo tridimensionale avendo a disposizione solo informazioni riguardo ad una sua proiezione su un sensore bidimensionale. Ci sono molte ragioni per cui le tecniche di elaborazione di immagini bidimensionali non sono applicabili per un'efficiente e precisa analisi di oggetti tridimensionali:

- Il sensore di acquisizione di una telecamera (come anche l'occhio umano) acquisisce informazione sotto forma di una proiezione prospettica del mondo, con successiva e considerevole perdita d'informazione. A causa di questa trasformazione, un punto proiettato su un'immagine si può trovare fisicamente in uno qualsiasi dei punti allineati sulla retta congiungente il centro focale del sensore col punto in questione: si perde cioè l'informazione di profondità;
- La relazione tra l'intensità dei pixel dell'immagine e la geometria 3D degli oggetti inquadrati dipende da diversi parametri delle superfici degli oggetti stessi, quali l'orientamento, il tipo e la posizione delle fonti di luce, la posizione dell'oggetto, gli indici di riflessione e assorbimento della superficie, ...;
- L'occlusione totale o parziale di oggetti da parte di altri oggetti rende ancora più complicato distinguere l'oggetto posto al di sopra degli altri.

Il sistema di visione ideale è quello in grado di analizzare il mondo tridimensionale a partire da informazioni tridimensionali, ovvero apparentemente senza trasformazioni di coordinate 2D-3D. Lo sviluppo tecnologico degli ultimi anni e, in particolare, l'aumento esponenziale delle risorse computazionali ha permesso lo sviluppo di sensori e algoritmi di visione che permettono l'acquisizione ed elaborazione di questo tipo di informazioni.

L'implementazione degli algoritmi presentati in questo capitolo viene fornita da alcune librerie software, tra le quali le più utilizzate sono le librerie *OpenCv* ([site10]) e la *Point-Cloud Library* (*PCL*, [site12]). In particolar modo, mentre i moduli offerti dalle *OpenCv* offrono implementazioni di algoritmi ad ampio raggio e solo nelle versioni più recenti è stata dedicata una sezione dedicata alla trattazione di dati tridimensionali, la *PCL* nascono specificatamente per l'utilizzo con questa tipologia d'informazione. Pertanto, gli algoritmi implementati e l'architettura software messa a disposizione sono ottimizzati per l'analisi delle nuvole di punti ([1]). Lavori svolti e presentati di recente mostrano le potenzialità delle tecniche di analisi di informazioni tridimensionali in task di manipolazione ([14]), in presenza di occlusioni nella tipica applicazione di svuotamento di un cassone ([13]) e per l'individuazione e tracciamento di oggetti deformabili ([58]).

La classificazione e scelta degli algoritmi da discutere nel capitolo è tratta principalmente da [64].

In questo capitolo si affrontano le tematiche che permettono di affrontare il problema della ricostruzione di una scena tridimensionale a partire da un'informazione tridimensionale. Innanzitutto, si fornisce una classificazione dei compiti demandati ad un sistema di visione 3D rispetto a quelli già risolvibili con un classico sistema di visione bidimensionale (1). In secondo luogo si presentano i dispositivi specificatamente usati per la misura delle distanze e le strutture dati utilizzabili per il trasferimento, elaborazione e salvataggio dell'informazione tridimensionale (2). In seguito si descrivono le metodologie per la ricostruzione e descrizione di oggetti tridimensionali (3) al fine di riconoscerne la presenza all'interno di una scena (4). Tali algoritmi rappresentano un'estensione alle classiche tecniche di elaborazione delle immagini, dovuta allo sfruttamento dell'informazione aggiuntiva data dalla distanza tra punti della scena ed il dispositivo di visione. Alla fine del capitolo si trovano alcune considerazioni generali ed esempi applicativi degli algoritmi presentati (5).

## 1 Classificazione dei compiti

L'ambito di ricerca della *visione 3D* è relativamente nuovo e, pertanto, non è possibile fornirne una sola definizione. Di seguito vengono fornite alcune possibili interpretazioni di obiettivi ottenibili dall'elaborazione di informazioni tridimensionali della scena.

La *teoria dei sistemi* fornisce i fondamenti per comprendere fenomeni complessi utilizzando una formulazione matematica. La complessità del compito di visione 3D è risolto distinguendo tra *oggetti* e *sfondo* della scena, dove col termine “oggetti” si intende tutto ciò che è d'interesse per l'obiettivo finale del sistema di visione. Gli oggetti e le loro proprietà vengono tipicamente caratterizzati tramite un *modello* matematico formale, il quale specifica un set finito di parametri. Il modello è generalmente costruito a partire da immagini, ma in alcune applicazioni industriali è interessante creare modelli partendo direttamente dal modello CAD degli oggetti. La creazione di un sistema di visione 3D prevede sempre tre fasi iniziali, indipendenti dal tipo di algoritmo utilizzato e propedeutiche alle fasi di ricostruzione e riconoscimento presentate in precedenza:

1. Individuazione delle *caratteristiche* (feature) osservabili in un'immagine.

2. *Rappresentazione*: questo problema è relativo alla scelta del modello migliore per la rappresentazione degli oggetti d'interesse.
3. *Interpretazione*: si tratta di capire come le feature scelte sono mappate nel mondo reale.

## 1.1 Ricostruzione di oggetti tridimensionali

La definizione di ricostruzione tridimensionale è la seguente: da un'immagine o serie di immagini di una scena, derivarne un'accurata descrizione geometrica tridimensionale e determinare quantitativamente le proprietà degli oggetti presenti.

Quindi, il primo compito della visione 3D è quello di trovare delle modalità per costruire una *descrizione matematica* di forme anche complesse in un sistema di coordinate indipendente dall'osservatore. La soluzione di questo problema passa attraverso due fasi distinte: l'*acquisizione* di informazioni tridimensionali, ovvero la profondità di ogni singolo pixel dell'immagine oltre alla consueta informazione di intensità luminosa o colore e l'*individuazione* e *classificazione* di proprietà geometriche e/o matematiche per la descrizione della scena 3D. Le tematiche da affrontare riguardo a questo compito sono:

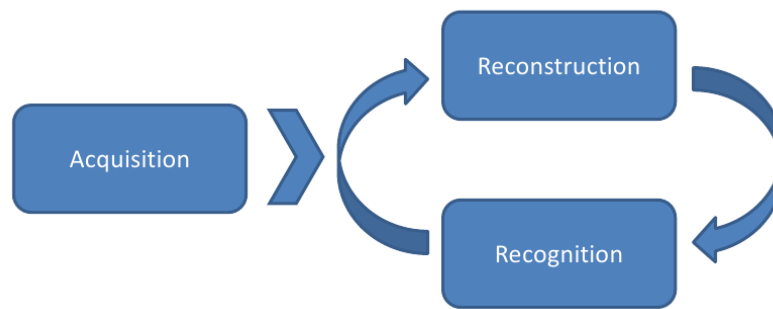
- Metodi di acquisizione della profondità di ogni pixel (sensori 3D).
- Modello del sensore d'acquisizione e suoi parametri (ottenibile tramite calibrazione).
- Caratteristiche geometriche di forme, superfici e punti 3D.

## 1.2 Riconoscimento

La domanda cui risponde il *riconoscimento* è la seguente: data una scena 3D e la conoscenza a priori degli oggetti da ricercare all'interno della scena stessa, l'oggetto è presente nella scena? E in più, se presente, in che posizione si trova?

In alcune applicazioni è possibile fermarsi solo al riconoscimento della presenza dell'oggetto, ad esempio capire se su un nastro trasportatore è presente un certo prodotto in catalogo. L'estensione del compito di riconoscimento con quello di *calcolo della posa* dell'oggetto è necessario in applicazioni di guida robot, in cui è fondamentale capire la posizione e orientamento dell'oggetto nel sistema di riferimento del robot per poter comandare il manipolatore in modo che svolga il compito programmato. Come mostrato in figura 4.1, acquisizione, ricostruzione e riconoscimento formano le basi di ogni algoritmo di visione 3D.





*Figura 4.1: Compiti tipici della visione 3D*

### 1.3 Comprensione, segmentazione e tracking

Un altro problema può essere posto nel seguente modo: da una sequenza di immagini di un oggetto in movimento (o statico) riprese da un osservatore in movimento (o statico), comprendere le proprietà tridimensionali degli oggetti.

Questa definizione è simile al riconoscimento, ma l'uso della parola *comprendere* necessita l'introduzione di un approccio di tipo diverso. In questo caso, infatti, non è più necessario solamente individuare una descrizione della scena osservata, ma capire come gli oggetti interagiscono tra loro all'interno della scena. L'esempio classico è per l'appunto l'individuazione del movimento degli oggetti all'interno della scena relativamente a movimenti compiuti nella terza dimensione: capire se gli oggetti si stanno avvicinando all'osservatore, se si stanno sovrapponendo, etc. . . . Il problema in questo caso è distinguere gli oggetti appartenenti ad una scena. Se non è disponibile una piccola conoscenza a priori, come nel caso della visione umana, la comprensione è complicata. Il caso opposto è quello in cui si ha a disposizione la completa conoscenza a priori degli oggetti con cui si deve interagire: il problema diventa quello del *tracciamento*, trattato in modo più approfondito nel capitolo 6.

In figura 4.2 è riportato un esempio di tracciamento di movimenti di persone per un'applicazione di monitoraggio degli accessi, in cui si vede come la distinzione di oggetti sovrapposti sia fondamentale per raggiungere la comprensione della scena.

## 2 Acquisizione e rappresentazione dei dati

L'*acquisizione* di informazioni tridimensionali sulla scena inquadrata da dispositivi atti alla visione 3D rappresenta sempre il primo passo da compiere per lo sviluppo di un'applicazione



**Figura 4.2:** *Visione 3D: tracciamento con occlusioni*

di visione tridimensionale. Tali informazioni tridimensionali vengono immagazzinate nel calcolatore sotto forma di *nuvola di punti* (*point cloud*).

Le problematiche relative all'acquisizione di immagini 3D tramite appositi dispositivi sono ben descritte in [62].

## 2.1 Acquisizione di informazioni tridimensionali

In generale, per ottenere la nuvola di punti completa è necessario passare attraverso le seguenti fasi successive:

1. Misura delle distanze tra sensore e punti delle superfici degli oggetti della scena;
2. Integrazione delle informazioni prospettiche e di distanza in coordinate 3D  $(x, y, z)$  attraverso i parametri ottenuti dalla calibrazione del sensore;
3. Aggiunta di ulteriori informazioni alle coordinate, ad esempio informazioni di colore (in tal caso si parla di dati *RGB-D*);
4. Generalmente i sensori 3D sono in grado di acquisire informazioni solamente sulla porzione degli oggetti inquadrata (dati *2.5D*). Per tale motivo, nel caso si volesse ottenere una rappresentazione 3D completa (a  $360^\circ$ ) di tutte le superfici dell'oggetto si dovrebbe passare attraverso un'ulteriore fase di unione delle misure, chiamata *registrazione*.

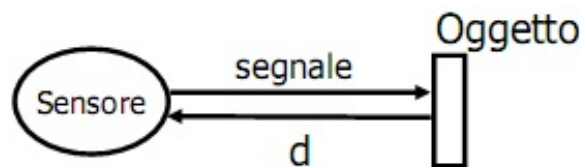
I sensori 3D sono classificati in base al mezzo con cui “perturbano” l'ambiente esterno per ottenere l'informazione di profondità:

- *Sensori di contatto*. Non sono trattati in questa sede perché prevedono il contatto con gli oggetti, mentre i sistemi di visione permettono misure senza contatto;

- *Sensori attivi.* Utilizzano una fonte di energia esterna (laser o infrarossa) per illuminare la scena. L'informazione 3D è ottenuta valutando la rifrazione (in termine di intensità o tempo di volo) di tale fonte di illuminazione da parte delle superfici degli oggetti della scena. La descrizione dettagliata delle trasformazioni e relazioni necessarie all'utilizzo della visione attiva è stata fornita nel paragrafo 2.4.2 del capitolo 2 e in questa sede ci si limita a presentarne vantaggi e svantaggi in relazione ad altre tecniche di acquisizione delle informazioni 3D;
- *Sensori passivi.* Non utilizzano nessuna fonte di energia esterna. L'informazione di profondità è ottenuta combinando più immagini della stessa scena prese da punti diversi (stereoscopia) oppure ad istanti di tempo diversi (*structure from motion*), come già descritto nel paragrafo 2.4.3 del capitolo 2. Come nel caso della triangolazione attiva, in questo paragrafo si presentano i vantaggi e gli svantaggi di questi approcci in relazione ad altri tipi di tecniche.

### 2.1.1 Sensori Time-Of-Flight

La misura della distanza di un oggetto è data dalla misura del tempo che un dato segnale (ad es. un impulso sonoro) impiega a raggiungere l'oggetto e a tornare indietro (*tempo di volo*, time-of-flight, TOF) come schematizzato in figura 4.3.



**Figura 4.3:** Calcolo della distanza con tempo di volo

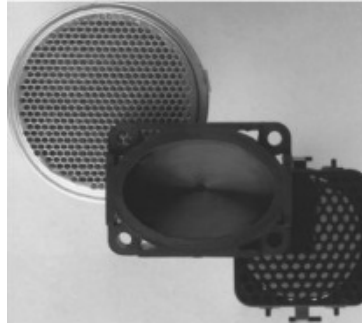
La distanza percorsa è data dalla semplice relazione

$$d = \frac{v \cdot t}{2} . \quad (4.1)$$

Questo metodo è impiegato da diversi anni nei telemetri laser o nei sensori ad ultrasuoni, un cui esempio è mostrato in figura 4.4.

Un sensore ad ultrasuoni è formato da due componenti principali:

- trasduttore di ultrasuoni (che funziona sia da emettitore che da ricevitore);
- elettronica per il calcolo della distanza.



*Figura 4.4: Sensore ad ultrasuoni*

Il tipico ciclo operativo include i seguenti passi: l'elettronica pilota il trasmettitore per inviare ultrasuoni e, nel contempo, disabilita momentaneamente il ricevitore per evitare la ricezione di false risposte. Passato un certo intervallo di tempo, il ricevitore viene riabilitato, il che permette ad esso di ricevere un segnale in risposta (verosimilmente quello giusto). Il segnale ricevuto viene quindi amplificato con un guadagno crescente, per compensare la diminuzione di intensità con la distanza. Infine, gli eco di ritorno che superano una certa soglia vengono presi in considerazione e associati a delle distanze calcolate in base al tempo trascorso dalla trasmissione.

Più recentemente sono comparse sul mercato le cosiddette *TOF-cameras* di figura 4.5, ovvero dispositivi che per ogni pixel aggiungono alla consueta informazione di intensità luminosa anche l'informazione di distanza.



*Figura 4.5: Telecamera time-of-flight*

Allo stesso modo del sensore ad ultrasuoni, un illuminatore ad infrarossi emette un impulso luminoso; l'impulso di ritorno può essere analizzato in due modi differenti:

- *Misura dello sfasamento.*

- *Range-gated imager*: ogni pixel ha un otturatore (shutter) che inizia a chiudersi nel momento in cui l'impulso è emesso; la distanza del punto è inversamente proporzionale all'intensità luminosa rilevata.

Il principale vantaggio di questo tipo di sensori risiede nel fatto di non avere bisogno di motori per direzionare l'impulso luminoso. Inoltre, permettono acquisizioni Real-Time (tra 30 e 100 fps) e hanno un costo piuttosto contenuto. Gli svantaggi derivano dal fatto di avere bisogno di una fonte di illuminazione esterna, che può quindi soffrire di disturbi ambientali (luce solare, altri illuminatori infrarossi) e disturbi dipendenti dalle riflessività delle superfici degli oggetti inquadrati (superfici nere sono difficili da acquisire e superfici lucide possono introdurre errori di misura).

### 2.1.2 Stereoscopia

In figura 4.6 è mostrato un esempio di dispositivo per l'acquisizione di nuvole di punti tramite stereoscopia.



**Figura 4.6:** *Dispositivo per visione stereoscopica*

La stereoscopia presenta molti vantaggi, tra cui il costo, l'acquisizione ad alto frame rate e la possibilità di associare direttamente l'informazione di colore letta da una delle due telecamere all'informazione di profondità per ottenere dati RGB-D. Un altro considerevole vantaggio sta nelle risoluzioni raggiungibili, che sono generalmente più alte rispetto agli altri tipi di sensori. Il principale problema risiede invece nella necessità di avere una perfetta sincronizzazione tra i differenti dispositivi di visione: il disallineamento dei tempi d'acquisizione può portare problemi di misurazione in presenza di scene non statiche e oggetti mobili. L'accuratezza della misura, inoltre, tende a diminuire in caso di presenza di superfici lineari (con poche texture e/o bordi) poiché il calcolo della distanza è possibile solo a seguito dell'individuazione di un certo numero di corrispondenze tra due immagini. Infine, l'area di visione effettiva risulta essere relativamente ristretta, poiché corrisponde solo alla porzione di scena inquadrata da entrambe le telecamere.

### 2.1.3 Triangolazione attiva

Le tecniche di triangolazione attiva offrono una via di mezzo tra gli approcci di calcolo del tempo di volo e la triangolazione passiva. Infatti, utilizzano una fonte di energia esterna come le prime, ma la profondità è ottenuta per triangolazione come le seconde.

Il vantaggio principale rispetto alla triangolazione passiva sta nell'evitare i problemi riguardanti il matching di punti tra due immagini, ovvero la fase di calcolo delle corrispondenze. Tuttavia, vengono introdotti ulteriori problemi riguardanti la rilevazione del segnale emesso dalla sorgente. In particolare, le difficoltà risiedono nella scelta della tipologia di illuminazione da utilizzare, la quale scelta va effettuata a seconda del tipo di ambiente in cui si intende utilizzare il sensore stesso e della tipologia di oggetti da individuare.

**Luce strutturata.** E' un'estensione della triangolazione attiva che utilizza un pattern luminoso anziché un punto solo e calcola la distanza in base alla distorsione della figura che se ne ottiene. Questo è il principio alla base di:

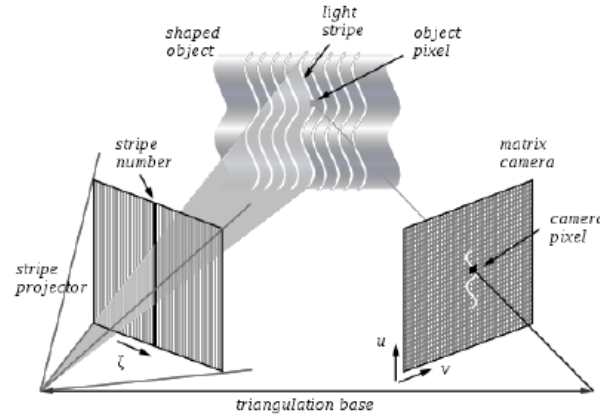
- *Scanner 3D*: il pattern è generalmente una linea che viene traslata e/o ruotata all'interno della scena per mezzo di motori. In alternativa sono gli oggetti ad essere posti in movimento e la posizione relativa della sorgente laser e del dispositivo di visione è costante, come in figura 4.7.



*Figura 4.7: Scanner 3D a triangolazione attiva*

Il vantaggio di questi dispositivi è l'alta precisione raggiungibile. Questa va però a discapito di un limitato field-of-view, con tempi di scansione molto elevati e un costo tipicamente alto. Questa tipologia di dispositivi è infatti utilizzata in applicazioni di ricostruzione 3D non real-time in cui è però necessaria un alto grado di precisione (ad esempio nella ricostruzione di edifici in 3D);

- *Sensori a triangolazione di pattern statistico*: il pattern è una trama di punti o linee. Proiettando una serie di linee permette l'acquisizione di informazioni su più superfici contemporaneamente, ma rende necessario affrontare il problema dell'interferenza e possibilità di confondere linee vicine. I moderni sensori a luce strutturata combinano una telecamera nella banda del visibile (RGB) e una nell'infrarosso per ottenere set di dati RGB-D, come schematizzato in figura 4.8.



**Figura 4.8:** Sensore a luce strutturata, principio di funzionamento

I vantaggi principali sono dati da un'acquisizione real-time di dati (tra 30 e 60 fps) e una precisione di misura al di sotto del millimetro. Per tali motivi rappresentano una soluzione ad un costo particolarmente contenuto per quasi tutte le applicazioni di visione tridimensionale. A fronte di questi vantaggi, si hanno anche alcuni svantaggi: il range di distanze d'utilizzo è contenuto e dipende dal tipo di dispositivo; la misura è particolarmente influenzabile da sorgenti di luce esterne e dalle caratteristiche delle superfici illuminate, il che rende questi sensori ideali per applicazioni industriali ad illuminazione controllata, ma deboli per risolvere problemi di navigazione; difficoltà di utilizzo combinato di due sensori nello stesso field-of-view a causa dell'interferenza tra le varie emissioni luminose.

#### 2.1.4 Projected Texture Stereo Vision

Alcuni dispositivi di recente concezione, ad esempio il dispositivo di visione 3D *IDS Ensensio N10* ([site2] di figura 4.9) uniscono le tecniche di triangolazione attiva e passiva per ottenere prestazioni ancora più elevate. Questo sistema è composto da:

- Due telecamere CMOS a global shutter (per permettere acquisizioni sincrone);

- Una sorgente infrarossa per l'emissione di un pattern casuale a punti per la triangolazione delle distanze;
- Un rilevatore di emissioni infrarosse per la misurazione sul pattern;



**Figura 4.9:** *Dispositivo per visione stereoscopica con ausilio di pattern luminoso*

L'integrazione delle tecniche standard di stereoscopia con la triangolazione a luce strutturata (figura 4.10) permette di ottenere prestazioni elevate in tutte le situazioni operative grazie all'unione dei punti di forza di entrambe le tecnologie: la luce strutturata aiuta nel caso di superfici ad elevato coefficiente di riflessione, mentre la stereoscopia risulta la scelta vincente con superfici poco uniformi.



**Figura 4.10:** *Projected Texture Stereo Vision: pattern luminoso*

Un altro vantaggio risiede nell'ampio range ammesso per la distanza operativa: le telecamere della serie Ensenso operano a distanze comprese tra 280 mm e 1400 mm con field-of-view dipendente dall'ottica montata. Il framerate raggiungibile è di 30 fps a fronte di una risoluzione spaziale inferiore al decimo di mm.

## 2.2 Strutture per la rappresentazione di dati 3D

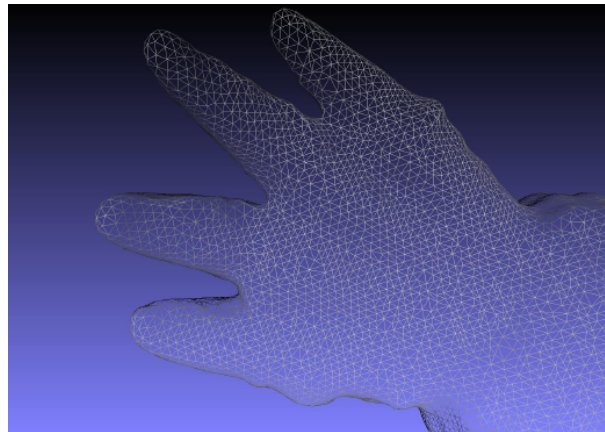
I dati 3D possono essere classificati in base a:

- *Ordinamento*. I dati di una nuvola di punti possono essere ordinati o meno;



- *Tipologia d'informazione.* I dati possono essere puramente 3D (le tre coordinate di un sistema cartesiano) oppure contenere anche informazioni di colore (RGB-D);
- *Vista.* Si può avere a disposizione la visuale completa della scena (*full 360 degrees view*) oppure solo la visuale da un particolare punto d'osservazione (*2.5D view*).

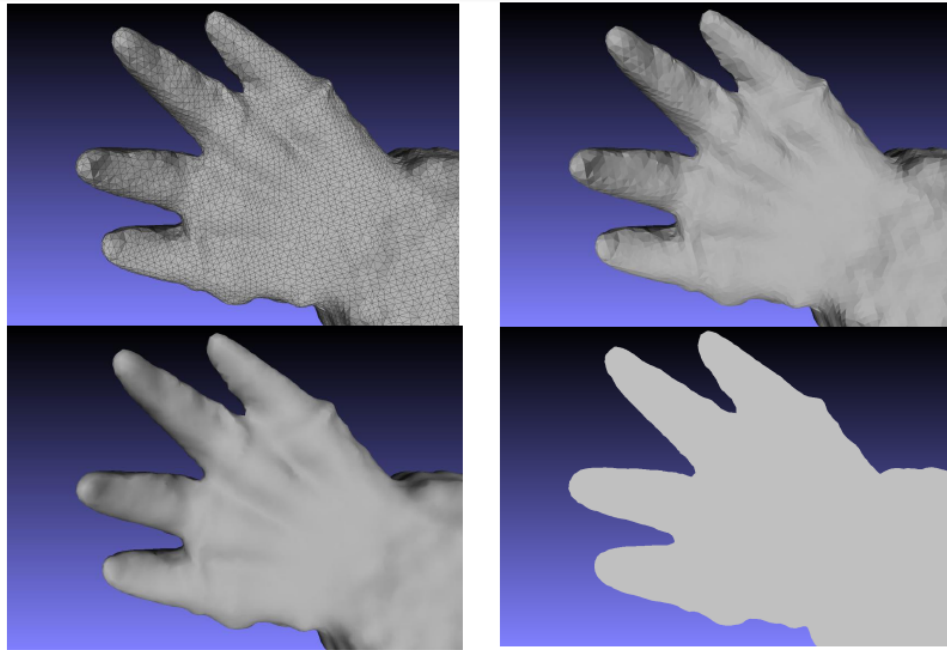
La rappresentazione 3D più usata, data la facilità d'estrazione, è sicuramente la *nuvola di punti*, un cui esempio è riportato in figura 4.11. Nella loro definizione pura, esse sono essenzialmente una collezione non ordinata di vertici. Non avendo a disposizione un “ordine” tra i vertici, è molto difficile eseguire su di esse algoritmi di ricerca sul vicinato dei punti (*neighbor searches*), a meno di non costruire prima una rappresentazione gerarchica dei punti (ad esempio il *KD-tree*). Inoltre, non avendo a disposizione neanche la topologia dei punti, ovvero il set di connessioni tra di essi, risulta complicato discriminare le superfici esterne di un oggetto da quelle interne.



**Figura 4.11:** *Nuvola di punti*

Determinare la direzione delle superfici è il problema tipico del *rendering* (figura 4.12) delle superfici, utilizzato in *computer graphics*, ad esempio, per la resa delle ombre. Le *mesh* sono infatti nuvole di punti a cui viene aggiunta una topologia, ovvero il set di connessioni tra i punti stessi.

L'introduzione di un particolare ordinamento tra i punti di una nuvola origina le cosiddette *voxelized cloud* (figura 4.13), dove il *voxel* è il corrispettivo tridimensionale del pixel. Una *voxelized cloud* è quindi una griglia 3D di valori di intensità luminosa. La voxellizzazione è la rappresentazione di dati più adatta ad ottenere algoritmi di analisi 3D efficienti, poiché le coordinate di un particolare punto sono definite implicitamente dall'indice che rappresenta la posizione del punto all'interno della griglia 3D di punti. La presenza di tale indice comporta che questo sia una rappresentazione ordinata di punti, senza topologia



**Figura 4.12:** *Rendering di mesh*

(ma facilmente ottenibile dalla griglia), su cui è facile eseguire algoritmi di ricerca e ottima per la visualizzazione delle scene 3D (il livello di dettaglio dipende infatti dalla risoluzione della griglia). L'unico svantaggio risiede nel fatto che, a seconda del dispositivo utilizzato, è necessario un passo ulteriore per ottenere la voxelized cloud a partire da altre rappresentazioni. Alcuni particolari tipi di sensori, come i body scanner e i metal detector permettono invece di ottenere direttamente la voxelized cloud.



**Figura 4.13:** *Voxelized Cloud*

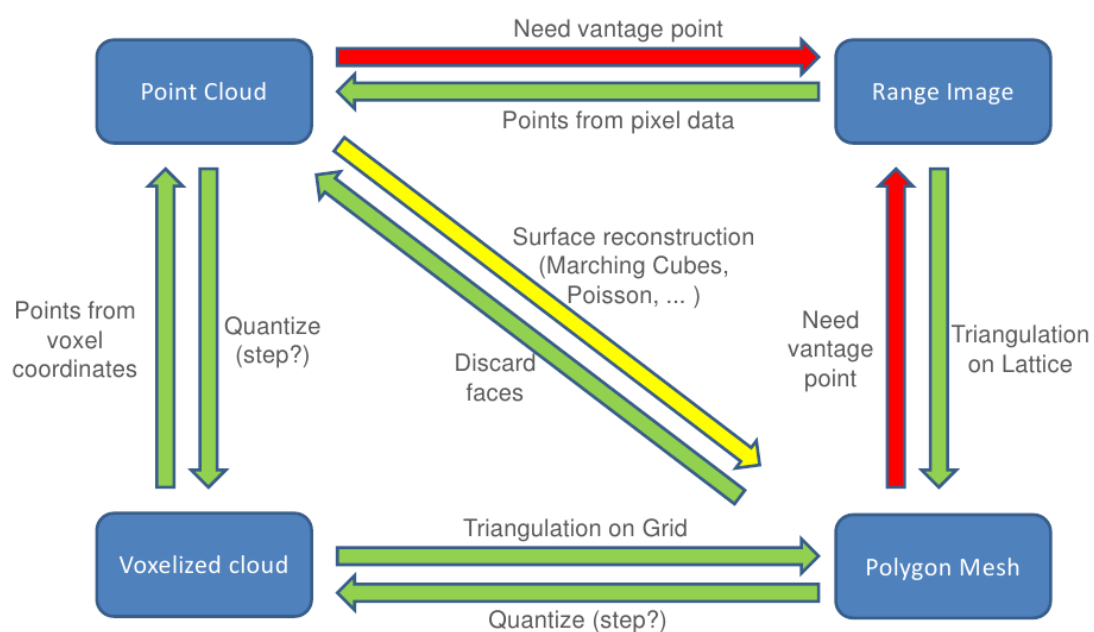
La proiezione di una voxelized cloud su una griglia 2D è detta *range image* (figura 4.14), la quale è anch'essa una rappresentazione ordinata di una scena tridimensionale. Anche questa rappresentazione è particolarmente utile per l'analisi, segmentazione e descrizione di scene 3D. In letteratura questo termine è utilizzato indifferentemente per indicare

un'immagine a canale singolo o multiplo in cui l'intensità dei singoli pixel codifica le coordinate del punto nello spazio cartesiano con il dispositivo di acquisizione nell'origine. Posizionando il sensore di acquisizione nell'origine del sistema cartesiano, la range image permette di ottenere solamente una vista 2.5D della scena.



*Figura 4.14: Range Image*

Per finire, la figura 4.15 schematizza le trasformazioni attraverso cui è possibile passare da una rappresentazione all'altra. Grazie a queste trasformazioni è possibile applicare qualsiasi tipo di algoritmo indipendentemente dal tipo di informazioni messe a disposizione dal sensore utilizzato.



*Figura 4.15: Trasformazioni tra rappresentazioni 3D*

### 3 Ricostruzione e Descrittori 3D

L'obiettivo della *ricostruzione 3D* è l'ottenimento di strutture dati e forme atte a descrivere scene tridimensionali. Si possono distinguere principalmente i seguenti livelli di descrizione:

- Livello *analitico*. Contiene l'informazione completa della scena 3D sotto forma del set completo di punti in coordinate tridimensionali (*nuvola di punti*). All'informazione della posizione è possibile aggiungere, ad ogni punto, l'informazione di colore o intensità luminosa. Questo livello è utilizzato a livello pratico solamente per la visualizzazione della scena 3D, ma l'acquisizione delle informazioni puntuali rappresenta un passo obbligato per ogni analisi successiva.
- Livello *sintetico*. A partire dalla nuvola di punti completa è possibile ottenere un set ristretto di punti aventi un contenuto informativo maggiore degli altri. Tale contenuto informativo è dato da *proprietà geometriche* o da *descrittori* statistici. Perdere alcune delle informazioni a disposizione nella nuvola di punti è un passaggio fondamentale per tutti gli algoritmi di visione 3D. Una nuvola di punti, infatti, occupa molto spazio in memoria poiché ogni punto è caratterizzato dalla sua posizione (e colore nel caso di dati RGB-D) e con i moderni sensori d'acquisizione si hanno a disposizione molti punti per immagine (ad esempio  $640 \times 480 = 307200$  punti). Ridurre il numero di punti necessari a descrivere la scena senza eccessiva perdita di informazioni è un passo necessario per rendere gli algoritmi di visione molto più veloci (si potrebbe dire real-time) senza perdita d'efficienza. Il passaggio a questo livello è quindi fondamentale per la *classificazione* di oggetti e per la successiva fase di *riconoscimento* di tali oggetti all'interno di una scena tridimensionale complessa.

#### 3.1 Registrazione

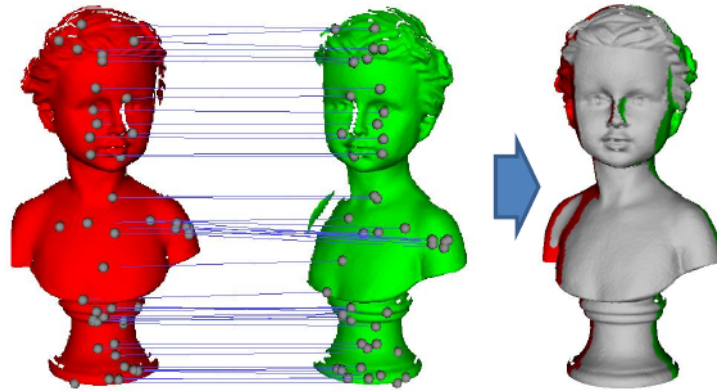
Per classificare i vari oggetti da ricercare all'interno di una scena tridimensionale non è sufficiente avere a disposizione una sola (o un set) di viste 2.5D. Risulta indispensabile, infatti, ottenere un modello 3D completo dell'oggetto d'interesse; questo compito è noto in letteratura col nome di *registrazione 3D* (figura 4.16).

Gli algoritmi di registrazione permettono di ottenere un modello 3D completo di un oggetto a partire da una serie di dati 2.5D acquisiti da punti di vista diversi e che presentano delle sovrapposizioni una con l'altra (figura 4.17).

Uno degli algoritmi di registrazione più utilizzati è l'algoritmo *ICP* (*Iterative Closest Point*). Tale algoritmo permette di allineare due viste 2.5D; di seguito i suoi fondamenti:



*Figura 4.16: Esempio di Registrazione 3D*



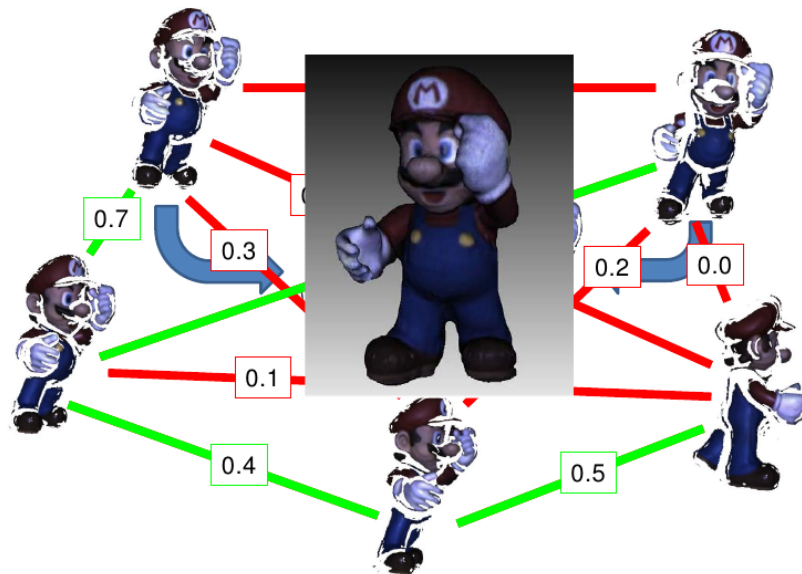
*Figura 4.17: Registrazione 3D: calcolo corrispondenze*

- Input: due differenti set di punti 3D  $\{\mathbf{M}, \mathbf{S}\}$ ;
- Output: la matrice di rotazione e il vettore di traslazione  $\mathbf{R}, \mathbf{t}$  che meglio allineano i due set di punti;
- Data una trasformazione iniziale (scelta ad esempio a partire da un'informazione a-priori), iterare i seguenti passi fino alla convergenza:
  1.  $\forall p \in M$  calcolare il vicinato  $NN(p) \in S$ ;
  2. Trovare l'orientamento assoluto della superficie nel punto: stimare  $\{\mathbf{R}, \mathbf{t}\}$  che minimizzano l'errore quadratico medio tra i set di coppie  $(p, NN(p))$  considerato, ovvero

$$\sum_{p \in M} ||NN(p) - (r \cdot p + t)||^2 . \quad (4.2)$$

- Il criterio di convergenza è dato da una soglia sull'errore minimo ammissibile o da un numero di iterazioni massimo;
- La soluzione dell'algoritmo dipende dalla scelta della trasformazione iniziale. Per ottenere la soluzione ottimale si procede per prove successive cambiando il punto di partenza iniziale.

In figura 4.18 è riportato un esempio di costruzione di un grafo rappresentante i punti vicini individuati.



*Figura 4.18: Iterative Closest Point*

### 3.2 Caratteristiche geometriche

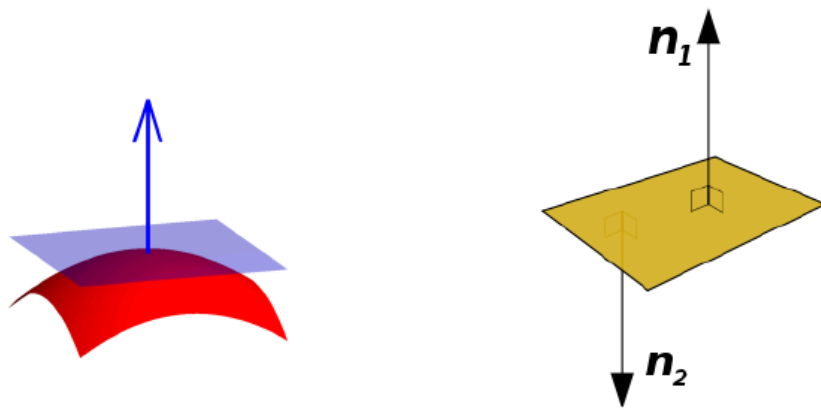
Nella visione 2D classica, l'individuazione delle caratteristiche geometriche di una particolare area dell'immagine è uno dei passi fondamentali per ottenere informazioni riguardo agli oggetti presenti nella scena. Per il calcolo di tali proprietà geometriche si passa tipicamente attraverso l'analisi dei momenti e, più in generale, dei contorni delle forme individuate. Anche nella visione 3D sono disponibili alcune proprietà tipiche degli oggetti 2D per mezzo di una semplice estensione del concetto di momento geometrico ad un set di coordinate 3D:

- *Centroide*: è il punto in cui si può pensare concentrata la “massa” dell'oggetto. Nel caso delle nuvole di punti si considera ogni punto avente massa unitaria, pertanto il centroide si troverà più vicino alla zona con la massima concentrazione di punti;
- *Volume*: è il corrispettivo dell'area per gli oggetti 2D;
- *Convex Hull*: è il volume convesso minimo contenente tutti i punti della nuvola;
- *Punti a massima distanza*;
- *Momenti invarianti*.

Tuttavia, nel caso di forme tridimensionali queste caratteristiche, seppure utilizzabili in alcuni algoritmi per l'individuazione di punti importanti nelle forme, sono difficilmente utilizzabili per classificare un particolare oggetto. Nel caso delle tre dimensioni, infatti, i classificatori geometrici più utilizzati fanno riferimento ad informazioni relative alle superfici degli oggetti come il *vettore normale* ad una superficie (3.2.1) e la curvatura della superficie (3.2.2).

### 3.2.1 Normale alla superficie

Il concetto di vettore normale ad una superficie trova molte applicazioni nel campo della grafica tridimensionale, ad esempio per il corretto posizionamento delle ombre dovute ad una particolare illuminazione. Nel dominio continuo si definisce *vettore normale alla superficie in un punto  $P$*  un vettore che è perpendicolare al piano tangente alla superficie nel punto  $P$ . L'ambiguità di questa definizione risiede solo nel verso da assegnare a tale vettore: dalla schematizzazione di figura 4.19 si vede come una normale non ha un verso univoco.



**Figura 4.19:** Definizione di vettore normale

La migliore rappresentazione di dati per il calcolo delle normali è la *mesh*: calcolate le facce orientate a partire dalla topologia dei vertici, la normale alla superficie nel punto  $P$  è data dalla media delle normali alle facce incidenti  $P$ . Anche il verso è definito considerando l'orientamento delle facce.

Calcolare le normali su una nuvola di punti, al contrario, è un compito molto più difficile. L'algoritmo per il calcolo della normale alla superficie nel punto  $P$  di una nuvola di punti passa attraverso tre fasi successive:

1. *Identificazione del vicinato.* Per ogni punto è necessario trovare  $N$  punti vicini per poter trovare la migliore approssimazione possibile alla superficie. Questo compito è risolto tramite una *ricerca radiale* o una *ricerca k-NN*. Nel primo caso il parametro da definire è l'ampiezza (nell'unità di misura della nuvola di punti) del raggio di ricerca, mentre nel secondo caso il parametro è la numerosità del vicinato;
2. *Stima del vettore normale.* Il metodo più usato per la soluzione di questo problema è basato sui *Minimi Quadrati Totali* (*TLS*, *Total Least Squares*). Seguendo la definizione, la normale è definita come il vettore perpendicolare al piano tangente alla

superficie in un punto della superficie:

$$n_x \cdot x + n_y \cdot y + n_z \cdot z - d = 0 . \quad (4.3)$$

Dato un insieme di  $k$  vicini  $\mathbf{p}_i$  del punto  $\mathbf{p}$ , i minimi quadrati totali permettono di trovare i parametri dell'equazione del piano tangente e, di conseguenza, la normale  $\mathbf{n}$  minimizzando l'equazione

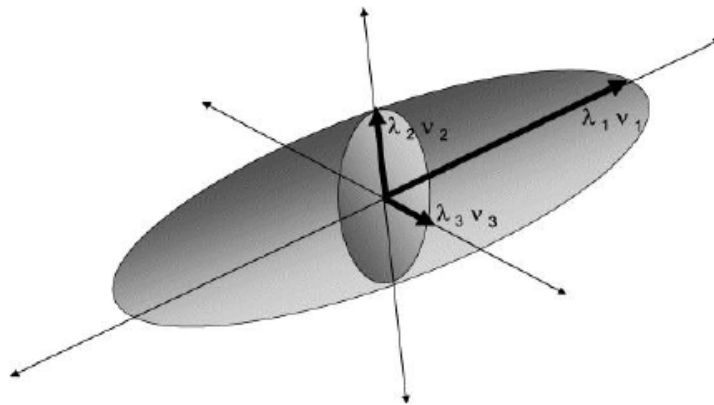
$$\sum_{i=1}^k (\mathbf{P}_i \cdot \mathbf{n} - d)^2 \quad \text{con} \quad |\mathbf{n}| = 1 . \quad (4.4)$$

La soluzione è ottenuta tramite l'analisi delle componenti principali (*PCA*, *Principal Components Analysis*): il vettore cercato corrisponde all'autovettore corrispondente al minimo autovalore della *matrice di dispersione*  $\mathbf{M}$ :

$$\mathbf{M} = \frac{1}{k} \sum_{i=1}^k (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T \quad (4.5)$$

con  $\bar{\mathbf{p}} = \frac{1}{k} \sum_{i=1}^k \mathbf{p}_i$ .

Intuitivamente, dato che tale matrice modella la dispersione dei punti appartenenti al vicinato, l'autovettore corrispondente al minimo autovalore rappresenta la direzione lungo la quale vi è la minima variazione di coordinate e, quindi, rappresenta la direzione perpendicolare al piano tangente, come raffigurato in figura 4.20.



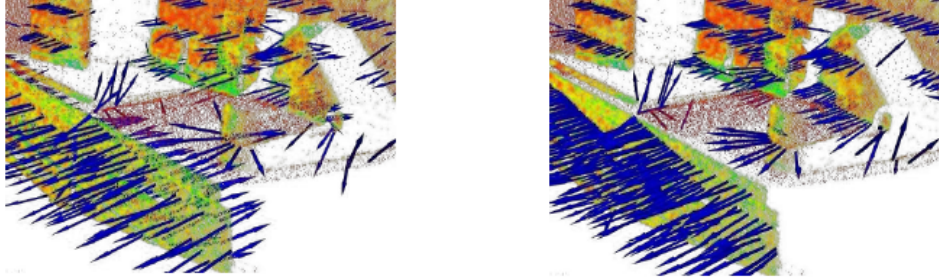
**Figura 4.20:** Significato geometrico della matrice di dispersione

3. *Definizione del verso(segno) del vettore normale.* La soluzione a questo punto non è disponibile utilizzando solo nuvole di punti e in assenza di informazioni note a priori. Considerando invece una *range image*, esiste una semplice definizione del verso delle normali (figura 4.21: poiché in una range image è nota la posizione  $\mathbf{v} = \{0, 0, 0\}$



del sensore è possibile definire il verso positivo dei vettori normali come quello che punta al sensore, ovvero il verso che rispetta la seguente proprietà:

$$(\mathbf{v} - \mathbf{p}) \cdot \mathbf{n} > 0 .$$



**Figura 4.21:** *Normali su range image*

Un altro approccio per il calcolo delle normali, valido anch'esso solo per una range image, è basato sul fatto che il vettore normale può essere ottenuto come prodotto vettoriale tra due vettori non paralleli appartenenti entrambi al piano tangente (*vettori tangente*). Selezionando come vettori tangente il gradiente medio orizzontale e verticale per ogni punto della range image, la normale è data dall'equazione

$$\mathbf{n} = \sum_{i=1}^n \mathbf{I}_x^{(i)} \times \sum_{i=1}^n \mathbf{I}_y^{(i)} . \quad (4.6)$$

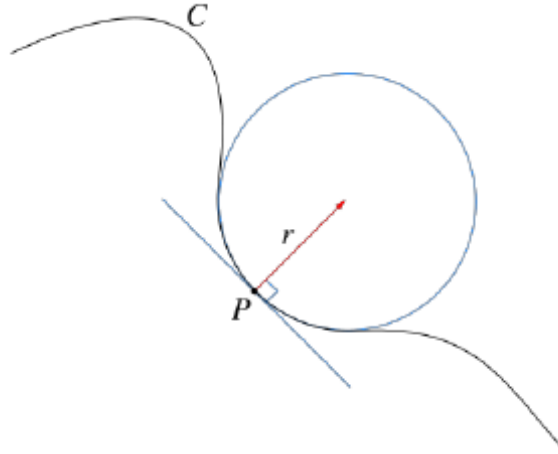
### 3.2.2 Curvatura

L'altra caratteristica geometrica fondamentale per descrivere l'andamento di superfici 3D è la *curvatura*. Intuitivamente è possibile descrivere la curvatura come la *quantità di cui una superficie devia dall'essere piatta*. Un cerchio ha sempre curvatura pari al reciproco del suo raggio:

$$C = \frac{1}{R} . \quad (4.7)$$

La curvatura di un profilo generico è definita dalla curvatura del suo cerchio osculatore in ogni punto, mostrato in figura 4.22.

Quando si passa da un sistema di riferimento bidimensionale ad uno tridimensionale, il concetto di curvatura diviene più complesso. In particolare, in ogni punto della superficie è possibile definire *piano normale* il piano che contiene il vettore normale. Ogni piano così definito taglia la superficie in una curva piana, ognuna avente differente curvatura.

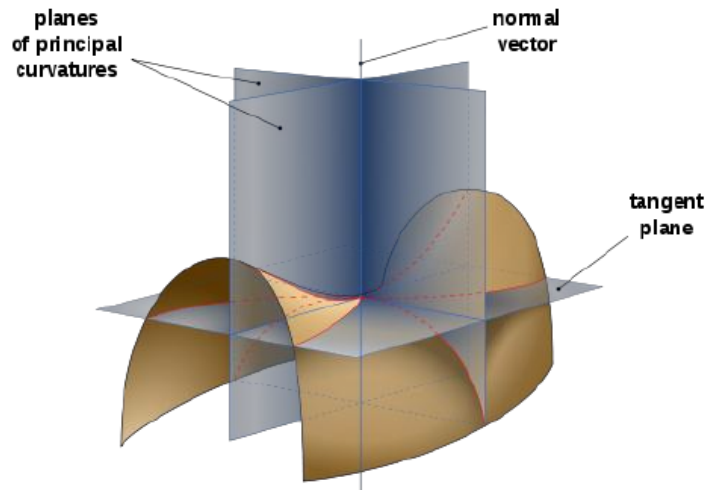


**Figura 4.22:** *Curvatura di un profilo*

Di conseguenza, in ogni punto della superficie non esiste una sola curvatura, ma è invece possibile definire la *curvatura direzionale* nel punto  $\mathbf{p}$  dato il *vettore tangente*  $T$ :

$$C(T) = \frac{1}{R(T)} . \quad (4.8)$$

Come nel caso dei vettori tangenti visti in precedenza, nel punto  $\mathbf{p}$  è possibile definire due direzioni principali della superficie, a cui corrispondono le due *curvature principali*  $C_1$  e  $C_2$  raffigurate in figura 4.23.



**Figura 4.23:** *Curvature principali*

I due vettori tangente da cui derivano le curvature sono sempre ortogonali tra loro e sono detti *direzioni principali*. Le informazioni contenute nelle due curvature principali possono essere sintetizzate in alcune caratteristiche più generali:

- *Curvatura media* ( $H$ ): è data dalla media delle due curvature principali:

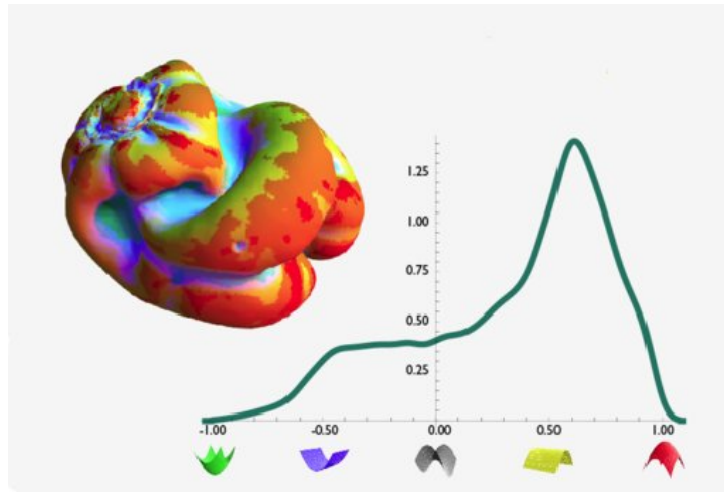
$$H = \frac{C_1 + C_2}{2} . \quad (4.9)$$

- *Curvatura gaussiana* ( $K$ ): è il prodotto delle curvature principali:

$$K = C_1 C_2 . \quad (4.10)$$

- *Shape Index*. Misura angolare, sviluppata da Koenderink e Van Doorn, atta a descrivere la topologia delle superfici locali (figura 4.24). La relazione dello shape index è

$$SI = \frac{2}{\pi} \arctan \left( \frac{C_2 + C_1}{C_2 - C_1} \right) . \quad (4.11)$$



**Figura 4.24:** Shape Index di una superficie 3D

Se si lavora con dati organizzati in nuvole di punti, un'approssimazione delle curvature principali è data dall'analisi delle componenti principali vista per il calcolo della normale ad una superficie. Gli autovalori minimo e massimo sono utilizzati come curvature principali, mentre i corrispondenti autovettori sono le direzioni principali.

### 3.3 Descrittori 3D

I *descrittori* sono comunemente utilizzati in visione artificiale per descrivere e successivamente comparare superfici al fine di trovare corrispondenze tra due set di immagini complete o parziali (*surface matching*).

A seconda del tipo di rappresentazione di dati e delle caratteristiche prese in esame per costruire il descrittore, è possibile distinguere tra:

- *Descrittori puntuali*. La descrizione è basata su una o più caratteristiche valutate direttamente su un punto, come normali, triangoli, Shape Index, ... Tra i vantaggi di questo tipo di descrittori vi è sicuramente la semplicità di costruzione e utilizzo, a fronte di una scarsa robustezza in presenza di rumore e poca capacità descrittiva per alcune applicazioni;

- *Descrittori locali.* La descrizione include informazioni calcolate sulle caratteristiche del vicinato attorno al punto d'interesse. La capacità descrittiva aumenta a fronte di un pari aumento della dimensionalità e complessità del descrittore;
- *Descrittori globali.* La descrizione è costruita basandosi sull'intera superficie a disposizione. Per calcolare correttamente il descrittore è necessario avere a disposizione la completa superficie 3D, senza occlusioni, al fine di evitare errori di classificazione che potrebbero in seguito portare ad errori di riconoscimento. Il risultato che si ottiene è molto più stabile e, di conseguenza, perfetto per task di *classificazione* (creazione di un database di oggetti).

### 3.3.1 Descrittori Locali

I descrittori locali 3D vengono spesso utilizzati in applicazioni di registrazione e riconoscimento di oggetti perché garantiscono un buon compromesso tra efficienza anche in caso di oggetti occlusi e onere computazionale. La descrizione di un oggetto completo si basa sull'associarne ogni suo punto con un descrittore delle geometria locale del punto stesso.

I descrittori locali vengono classificati in base al modo in cui rappresentano l'informazione:

- *Segnatura.* Per ogni punto vengono memorizzate una serie di informazioni calcolate prendendo in considerazione i punti appartenenti al vicinato. Per il calcolo delle proprietà geometriche è richiesta la definizione di un *sistema di riferimento locale* (*LRF*, *Local Reference Frame*);
- *Istogrammi.* Le caratteristiche topologiche o geometriche vengono rappresentate statisticamente per mezzo di istogrammi. Si perde l'informazione sui singoli punti, ma si ottiene una rappresentazione completa della struttura dell'oggetto;
- *Ibridi.* Per ogni punto della superficie in analisi viene memorizzata un'informazione statistica a corredo di un'altra informazione geometrica.

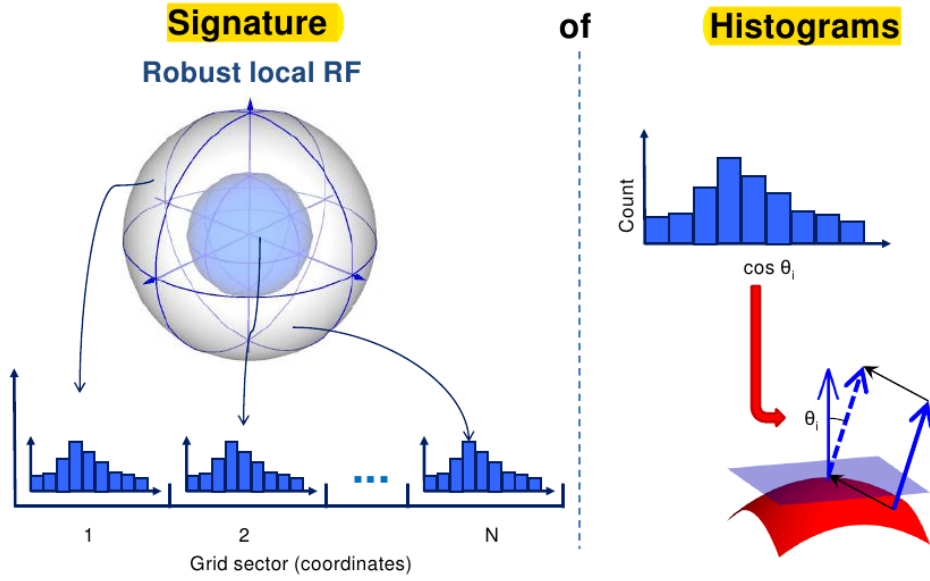
La tabella 4.1 elenca alcuni dei descrittori locali disponibili.

**Tabella 4.1:** *Descrittori Locali 3D*

Descrittore	Categoria	LRF Univoco	Texture
Struct. Indexing [Stein 92]	Segnatura	No	No
PS [Chua 97]	Segnatura	No	No
3DPF [Sun 01]	Segnatura	No	No
3DGSS [Novatnack 08]	Segnatura	No	No
KPQ [Mian 10]	Segnatura	No	No
3D-SURF [Knopp 10]	Segnatura	Sì	No
SI [Johnson 99]	Istogramma	RA	No
LSP [Chen 07]	Istogramma	RA	No
3DSC [Frome 04]	Istogramma	No	No
ISS [Zhong 09]	Istogramma	No	No
USC [Tombari 10b]	Istogramma	Sì	No
PFH [Rusu 08]	Istogramma	RA	No
FPFH [Rusu 09]	Istogramma	RA	No
Tensor [Mian 06]	Istogramma	No	No
HKS [Sun 09]	Altro	-	No
MeshHoG [Zaharescu 09]	Ibrido	Sì	Sì
SHOT [Tombari 10]	Ibrido	Sì	Sì

**Signature of Histograms of Orientations.** Il descrittore *SHOT* (figura 4.25) codifica le caratteristiche topologiche dell’oggetto 3D sotto forma di una serie di istogrammi, ognuno associato ad un angolo diverso; in questo modo si ottiene un descrittore invariante a rotazioni e traslazioni e robusto a rumore e occlusioni.

Si tratta di un descrittore *ibrido* perché formato da un’unione di caratteristiche geometriche o topologiche con proprietà statistiche. Il descrittore SHOT è costruito calcolando le caratteristiche geometriche d’interesse relativamente ai settori di una griglia a sfera costruita nell’intorno del punto in esame. Per ogni settore della griglia si costruisce un istogramma monodimensionale contenente le misure dell’angolo tra la normale alla superficie nel punto d’interesse e la normale di ogni punto appartenente alla struttura sferica di supporto. Il descrittore finale è formato dalla sovrapposizione di tutti gli istogrammi.



**Figura 4.25:** Descrittore SHOT

**Point Feature Histogram.** Il descrittore *PFH*, come il *FPFH* (*Fast Point Feature Histogram*) e il *VFH* (*Viewpoint Feature Histogram*) sono descrittori basati sul calcolo di proprietà geometriche delle superfici. In particolare, il *PFH* prevede il calcolo dell'orientamento relativo delle normali e delle distanze tra coppie di punti. Le coppie di punti sono costituite dai punti della superficie ( $p_i$ ) e dai punti del loro vicinato ( $p_j$ ). Il descrittore prevede il calcolo di una terna di angoli ( $\alpha$ ,  $\phi$  e  $\theta$ ) dato un sistema di riferimento fissato. Tale sistema di riferimento è formato dai seguenti assi:

- La normale  $\mathbf{n}$  nel punto  $p_i$ :

$$\mathbf{u} = \mathbf{n}_i; \quad (4.12)$$

- Il versore avente per direzione la congiungente dei punti  $p_i$  e  $p_j$ :

$$\mathbf{v} = \frac{\mathbf{p}_i - \mathbf{p}_j}{\|\mathbf{p}_i - \mathbf{p}_j\|^2} \times \mathbf{u}. \quad (4.13)$$

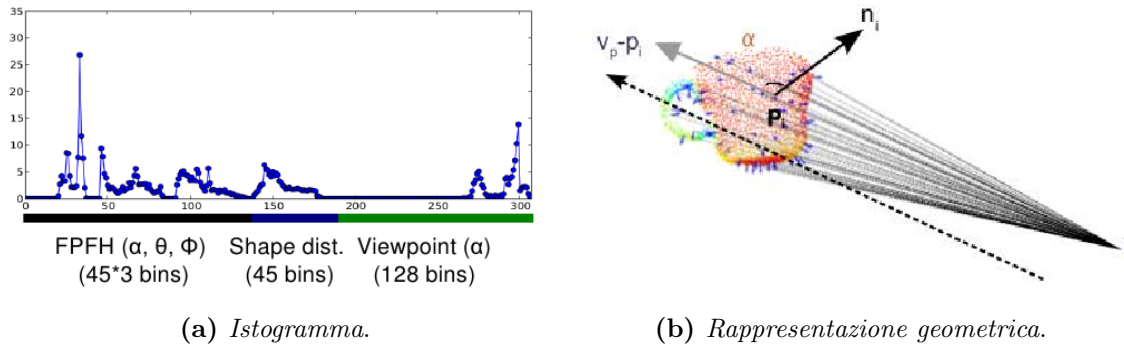
- Il vettore normale ai due vettori appena calcolati:

$$\mathbf{w} = \mathbf{u} \times \mathbf{v}. \quad (4.14)$$

Dato questo sistema di riferimento, i tre angoli d'interesse sono calcolati nel seguente modo:

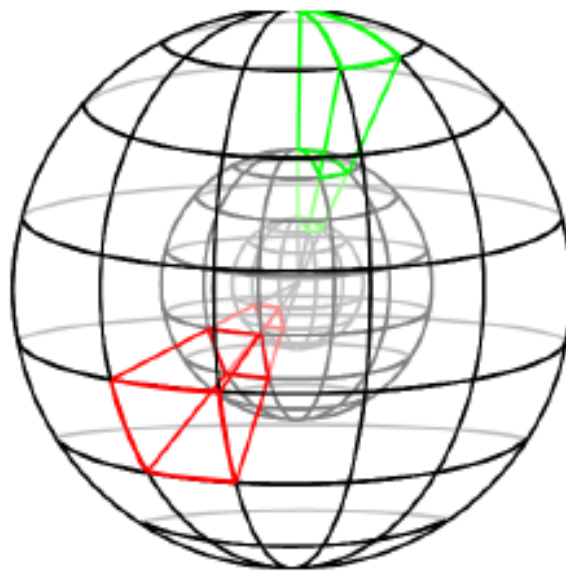
$$\begin{cases} \alpha = \mathbf{v}^T \cdot \mathbf{n}_i \\ \phi = \mathbf{u}^T \cdot \mathbf{v} \\ \theta = \arctan\left(\frac{\mathbf{w}^T \cdot \mathbf{n}_i}{\mathbf{u}^T \cdot \mathbf{n}_i}\right) \end{cases}. \quad (4.15)$$

Nel caso del descrittore *PFH*, questi angoli sono calcolati per ogni possibile coppia del vicinato di  $p_i$  e, in seguito, i valori ottenuti sono condensati in un istogramma a 33 o 45 bin (figura 4.26).



**Figura 4.26:** Descrittore PFH

**3D Shape Context.** Il descrittore *3D SC* (figura 4.27) è calcolato utilizzando una griglia sferica centrata sul punto di interesse, come nel caso del descrittore SHOT. La griglia è suddivisa in un insieme di settori definendo un set di valori equispaziati per quanto riguarda l'azimut e l'elevazione e un insieme di valori spazati logarithmicamente per quanto concerne le dimensioni del raggio. L'informazione è rappresentata ancora sotto forma di istogrammi (uno per ogni combinazione di raggio, azimut ed elevazione), in cui ogni valore è calcolato come la somma pesata del numero di punti inclusi nel corrispondente settore della griglia. Per quanto riguarda il sistema di riferimento, si ha che il polo nord della sfera è allineato con la direzione della normale nel punto in esame. Il set completo di istogrammi è ottenibile ruotando la sfera attorno a tale asse principale.



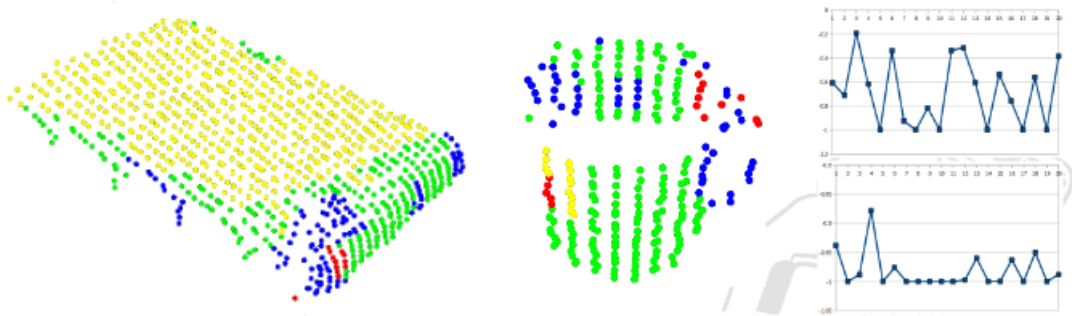
**Figura 4.27:** Descrittore SC

**Unique Shape Context.** Questo descrittore è un'estensione del descrittore precedente, dove si identifica un unico sistema di riferimento per la griglia a sfera da sovrapporre al punto da prendere in considerazione. Il polo nord della sfera è posizionato sempre nel verso della normale alla superficie, mentre gli assi  $x$  e  $y$  sono posizionati lungo le due direzioni principali della superficie (le direzioni delle curvature principali). In questo modo è possibile ridurre la complessità computazionale del descrittore ed eliminare le ambiguità derivanti dall'avere a disposizione più scelte per l'orientamento degli assi della sfera.

**Radius-Based Surface Descriptor.** Il descrittore RSD (figura 4.28)) descrive la geometria di un set di punti appartenenti ad un vicinato locale per mezzo di una stima delle loro relazioni radiali. Il raggio può essere stimato assumendo che ogni coppia di punti giaccia su una sfera, trovando quindi il raggio considerando che la relazione tra la distanza  $d$  tra i punti e l'angolo tra le normali alla superficie nei due punti è data da:

$$d(\alpha) = \sqrt{2r}\sqrt{1 - \cos \alpha} \sim r\alpha + r\alpha^3/24 + O(\alpha^5) . \quad (4.16)$$

Questa relazione è quasi lineare per  $\alpha \in (0, \pi/2)$ . Per stimare il minimo e massimo raggio  $r$  in un vicinato si possono applicare dei metodi di regressione lineare sulle minime e massime coppie  $(\alpha, d)$ . Il raggio assume valore massimo (infinito) nel caso di un piano e valore minimo per superfici con elevata curvatura. Un esempio dell'applicazione di questo risultato consiste nel distinguere superfici sferiche da altre cilindriche. Per un cilindro si ha raggio minimo pari al raggio del cilindro e valore massimo pari ad infinito.

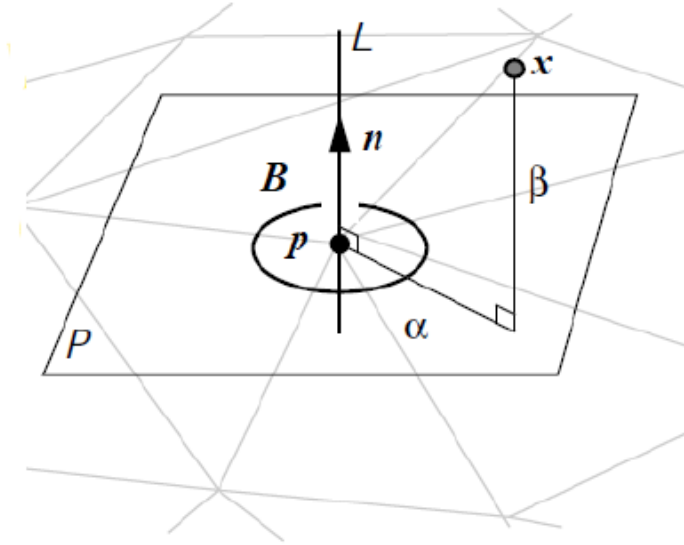


**Figura 4.28:** Descrittore RSD

**Spin Image.** Il descrittore  $SI$  è uno dei più utilizzati nell'ambito della visione 3D. Il suo utilizzo è limitato alle nuvole di punti in cui l'informazione relativa alle normali è disponibile per ogni punto della superficie in esame. Data la normale in un punto  $p$ , un piano contenente la normale viene fatto ruotare attorno ad essa. L'intersezione di tale



piano con la superficie in esame permette di individuare un set di punti  $s$ . Dato il punto  $p$ , per ogni punto  $s_i \in s$  è possibile calcolare la distanza tra le normali. Vengono accettate solamente distanze inferiori ad una certa soglia (figura 4.29).



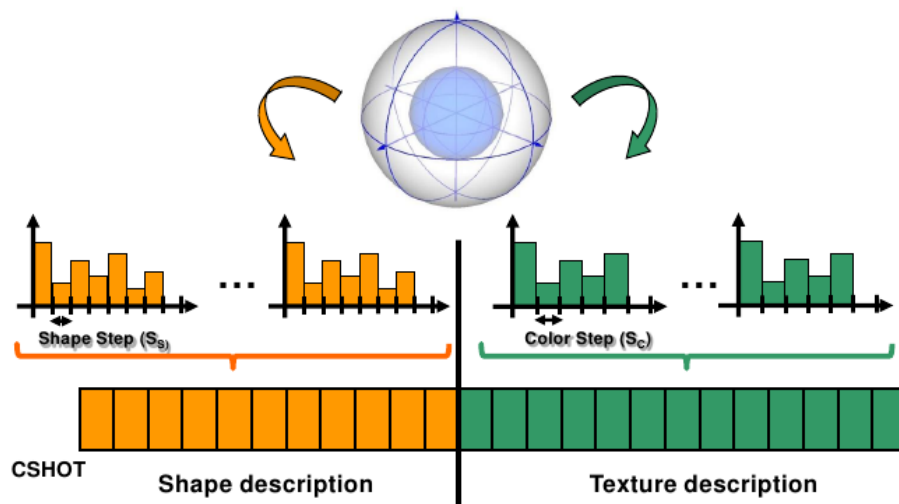
**Figura 4.29:** *Descrittore Spin Image*

Per ogni punto, dato l'angolo di cui è stato ruotato il piano base si ottiene un set di distanze, organizzato sotto forma di istogramma. La descrizione completa è ottenuta facendo ruotare il piano che origina il set di punti  $s$  di un valore equispaziato su 360 gradi (figura 4.30).



**Figura 4.30:** *Descrittore Spin Image: intero set di descrittori*

**Descrittori a caratteristiche multiple.** Nel caso si avesse a disposizione anche l'informazione relativa alle texture delle superfici (ad esempio con dati RGB-D), si potrebbe unire la descrizione delle proprietà geometriche delle superfici con l'istogramma dei valori di intensità dei pixel corrispondenti (figura 4.31). In questo modo si andrebbe a creare un descrittore ancora più robusto, ma per certi versi maggiormente influenzabile dalle condizioni di luminosità della scena.



*Figura 4.31: Descrittore Multiple-cue*

### 3.3.2 Descrittori Globali

I *descrittori globali* sono rappresentazioni ad elevata dimensionalità delle caratteristiche geometriche di un oggetto. Il loro utilizzo è simile a quello dei descrittori locali; vengono impiegati, quindi per la classificazione, riconoscimento e caratterizzazione geometrica di oggetti. Tuttavia, a differenza dei descrittori locali, a causa dell'alto costo computazionale, i descrittori globali sono generalmente calcolati su una nuvola di punti ristretta, ovvero un sottoinsieme della nuvola di punti rappresentante la scena completa. Tale nuvola di punti rappresenta l'oggetto di interesse per l'applicazione ed è ottenuta a partire da un modello 3D (per la classificazione) o dalla fase di *segmentazione* che precede la fase di riconoscimento di oggetti in una scena 3D complessa.

**Point Feature Histogram.** Il descrittore *PFH globale* rappresenta l'estensione del descrittore PFH locale. Allo stesso modo dell'ambito locale (3.3.1), tale descrittore viene costruito calcolando gli angoli  $\alpha$ ,  $\phi$  e  $\theta$  per tutte le coppie di punti appartenenti ad un vicinato e rappresentando queste informazioni tramite un istogramma. Nel caso del descrittore globale, tuttavia, non viene utilizzata solo l'informazione relativa agli angoli, ma anche altre caratteristiche geometriche, ad esempio la distanza tra i punti della coppia. Il numero di intervalli dell'istogramma dipende dalle dimensioni dello spazio delle caratteristiche scelto. Infine, nel caso del descrittore globale scompare il problema di individuare il raggio ottimale per la determinazione dei punti appartenenti ad un vicinato: dato che si lavora su un'intera nuvola di punti, il raggio corrisponde alla distanza tra i punti più lontani tra loro (massima distanza della nuvola di punti).

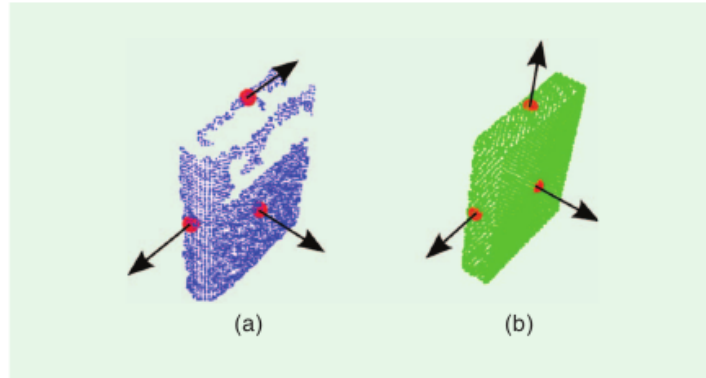
**Viewpoint Feature Histogram.** Ulteriore estensione del descrittore PFH è il descrittore *VFH*. Per costruire questo descrittore, gli angoli  $\alpha$ ,  $\phi$  e  $\theta$  vengono calcolati prendendo in considerazione la coppia di punti formata da un punto  $p$  della nuvola e il centroide  $p_c$  della nuvola stessa. Gli angoli tra il vettore normale al punto  $p$  e il vettore  $\frac{(p_c - p)}{(\|p_c - p\|)}$ , inseriti in un istogramma, rappresentano la caratteristica del descrittore dipendente dal punto di vista. Il calcolo di questa caratteristica rende questo descrittore utile per determinare la posa di un oggetto: una volta riconosciuto l'oggetto all'interno di una scena, è sufficiente valutare queste le differenze tra gli angoli del modello e quelli della scena per calcolare la trasformazione geometrica che lega l'oggetto della scena e il modello. Il descrittore prevede il calcolo di un'altra componente da aggiungere a quella degli angoli: la distanza tra i punti della nuvola e il centroide.

Si può notare che per l'utilizzo di questo descrittore non è necessario individuare nessun punto in particolare all'interno della nuvola di punti, se non il centroide che è però sempre disponibile e facilmente calcolabile. Il descrittore VFH è quindi un descrittore in grado di codificare la geometria di un qualsiasi oggetto come vista da un particolare punto di osservazione in un singolo insieme di istogrammi.

**Clustered Viewpoint Feature Histogram.** Il descrittore *CVFH* è un'ulteriore estensione del VFH basata sull'idea che la struttura geometrica di un qualsiasi oggetto è suddivisibile in un numero  $N$  di regioni disgiunte. Ognuna di queste regioni può essere utilizzata in modo indipendente per calcolare un insieme di  $N$  istogrammi VFH. Per esempio, il  $k$ -simo istogramma usa il centroide  $p_{c_k}$  e la normale  $n_{c_k}$  calcolati mediando i punti e le normali della regione  $k$ -sima. Il problema si riduce alla determinazione delle regioni in cui suddividere l'oggetto d'interesse; un possibile procedimento prevede di:

1. rimuovere i punti ad alta curvatura che rappresentano generalmente rumore o i bordi tra una regione e l'altra;
2. applicare l'algoritmo di "accrescimento" delle regioni sui restanti punti: scelti  $k$  punti, collegare ad ognuno di essi i punti vicini, in modo da ottenere  $k$  regioni distinte.

A differenza dei descrittori visti fino ad ora, il CVFH è una rappresentazione statistica multivariata delle caratteristiche geometriche di un oggetto. Pertanto, lavorando su  $k$  regioni, il CVFH risulta particolarmente robusto alle problematiche derivanti dalla segmentazione, quali la presenza di rumore in prossimità delle superfici e occlusioni di parti degli oggetti (figura 4.32).

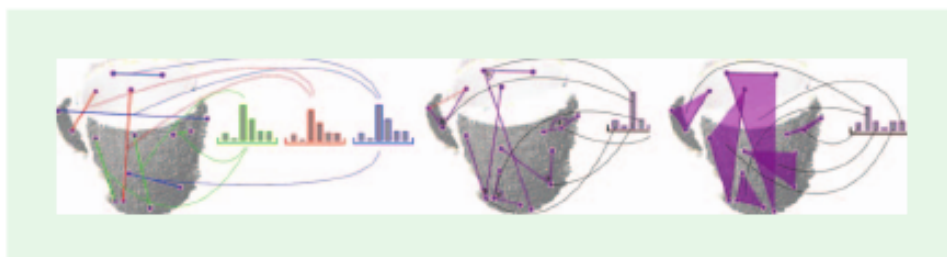


**Figura 4.32:** *Descrittore CVFH con occlusioni*

**Insieme di Fattori di Forma.** Questo descrittore viene costruito calcolando un insieme di istogrammi rappresentanti caratteristiche geometriche della nuvola di punti (*Ensemble of Shape Function, ESF*). Ogni istogramma è suddiviso in 64 intervalli e le caratteristiche geometriche comprendono proprietà quali angoli, distanze e aree. Per calcolare tali caratteristiche geometriche è vantaggioso avere a disposizione la griglia di voxel dell’oggetto. Tuttavia, come si può osservare dalla figura 4.33, il descrittore ESF può essere calcolato in modo efficiente a partire da una semplice nuvola di punti senza la necessità di svolgere ulteriori fasi di preprocessing della nuvola, mentre per gli altri descrittori potrebbe essere fondamentale eseguire i seguenti passi:

- “smoothing” di superfici;
- riempimento di buchi;
- calcolo delle normali;

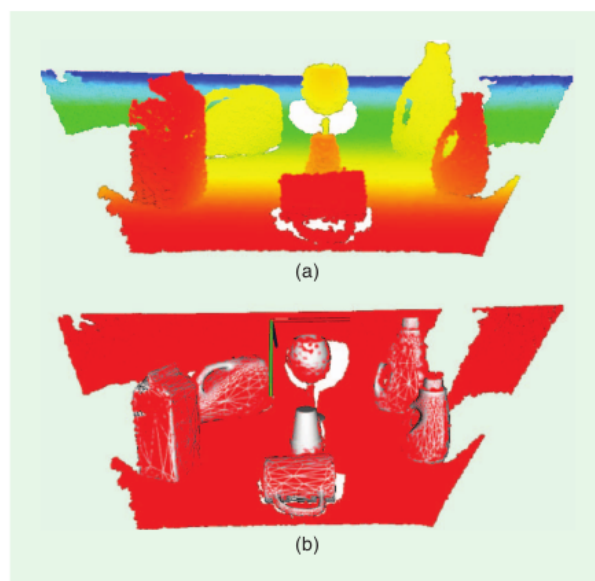
Lavorare direttamente sulla nuvola di punti rende automatica anche la gestione degli outlier, rumore e bordi non rilevati correttamente.



**Figura 4.33:** *Descrittore ESF*

## 4 Riconoscimento

Il *riconoscimento* è il compito attraverso cui è possibile riconoscere la presenza di un oggetto (idealmente rigido) all'interno di una scena e stimarne la posizione e orientamento nel mondo reale (posa a 6 gradi di libertà, 6DOF). In figura 4.34 si può vedere il risultato del procedimento di riconoscimento: a partire da una nuvola di punti ottenuta tramite un sensore a luce strutturata, si arriva ad individuare i vari oggetti della scena e a calcolarne la posa nel sistema di riferimento fissato, tipicamente rispetto alla posizione e orientamento del sensore di visione.



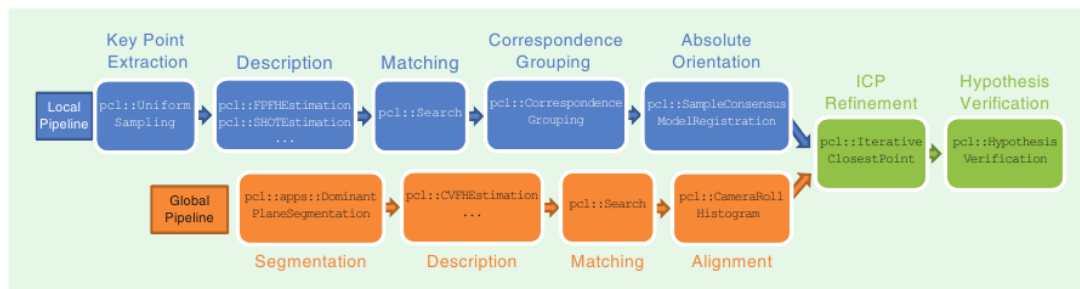
**Figura 4.34:** Esempio di riconoscimento da nuvola di punti

Come si vede in figura 4.35, il riconoscimento di oggetti tridimensionali prevede diverse tematiche da affrontare:

- *Training.* È la fase di addestramento del sistema di visione, al quale si devono fornire in ingresso tutte le informazioni necessarie al riconoscimento degli oggetti all'interno delle scene. Ad oggi, grazie ai moderni sistemi di disegno tridimensionale (CAD), l'individuazione delle caratteristiche geometriche viene fatta a partire da modelli tridimensionali rappresentati sotto forma di mesh. In alternativa all'utilizzo di modelli 3D, è possibile descrivere gli oggetti utilizzando nuvole di punti acquisite direttamente dalla scena.
- *Riconoscimento.* In questa fase vengono calcolati i descrittori 3D direttamente sulla scena osservata, in modo da poterli confrontare con gli stessi descrittori calcolati

sul modello. L'obiettivo del confronto è riuscire a capire se nella scena inquadrata è presente un particolare oggetto ricercato.

- *Stima della posa.* Una volta nota l'informazione relativa alla presenza o meno di un determinato oggetto all'interno di una scena, il passo successivo è quello di calcolarne la posizione e orientamento nello spazio, in modo che questa informazione possa essere utilizzata per interagire con l'oggetto stesso.



*Figura 4.35: Sequenza di riconoscimento*

## 4.1 Training

Per “addestrare” il sistema di riconoscimento le mesh rappresentanti l’oggetto 3D devono essere trasformate in nuvole di punti parziali, in modo da simulare l’acquisizione di tali modelli da parte di un dispositivo di visione 3D. Pertanto, una telecamera virtuale deve essere posta tutt’attorno all’oggetto, sulla superficie di una sfera di raggio sufficiente per far sì che l’oggetto venga inquadrato nella sua interezza da ogni singolo punto di vista. Per ottenere un campionamento uniforme su tutta la superficie della sfera, questa è generata utilizzando un icosaedro e suddividendo successivamente ogni faccia triangolare in quattro triangoli equilateri. In questo modo si ottengono 80 diverse posizioni per la telecamera. Ulteriori livelli di ricorsione possono essere raggiunti suddividendo ulteriormente ogni triangolo in altri quattro triangoli. La telecamera virtuale viene posizionata nel baricentro di ogni triangolo, con l’asse del sistema ottico diretto verso l’oggetto e parallelo al vettore normale al triangolo. I due parametri fondamentali da cui dipende il procedimento di costruzione dei modelli 3D sono:

- il livello di dettaglio scelto, ovvero il numero di ricorsioni con cui si approssima la sfera;
- la risoluzione delle immagini di profondità costruite.

In generale, per modelli sufficientemente semplici e oggetti grandi, per non richiedere un elevato sforzo computazionale al sistema di visione è possibile limitare il dettaglio dei modelli a 80 posizioni per la telecamera con una risoluzione di 150x150 pixel.

A partire dalle nuvole di punti appena costruite vengono calcolate le caratteristiche geometriche d'interesse e i descrittori. Caratteristiche e descrittori vengono archiviati per poter essere utilizzati in fasi successive. Per la ricostruzione della posa 6DOF dell'oggetto nella scena inquadrata è fondamentale salvare anche l'informazione relativa alla posizione della telecamera virtuale che ha generato la nuvola di punti analizzata. La trasformazione inversa permette di convertire la vista nelle coordinate del modello.

## 4.2 Riconoscimento tramite descrittori locali

In figura 4.35 vengono mostrate le principali fasi che permettono il riconoscimento di oggetti a partire da modelli costituiti da un set di descrittori locali:

- estrazione dei keypoint;
- descrizione e confronto (matching);
- raggruppamento delle corrispondenze;
- calcolo dell'orientamento assoluto.

Al termine di questo processo è possibile aggiungere due ulteriori processi di affinamento e consolidamento dei risultati che prevedono l'utilizzo di tecniche in comune coi metodi di riconoscimento a partire dai descrittori globali.

### 4.2.1 Estrazione dei keypoint

Il primo passo di ogni sequenza di processi per il riconoscimento 3D è costituito dall'estrazione di *keypoint* 3D dalla nuvola di punti in esame. Un keypoint è definibile come un punto avente caratteristiche geometriche e/o statistiche particolari che permettono di distinguerlo e differenziarlo da qualsiasi altro punto della nuvola.

Nella scelta di un algoritmo per l'estrazione dei keypoint è necessario tenere presente le caratteristiche che deve possedere:

- *Ripetibilità*. Lo stesso keypoint deve poter essere individuato in scene diverse sottoposte ad un differente tipo di rumore, illuminazione, ...;
- *Distinguibilità*. Un keypoint deve possedere un'elevata capacità di descrizione della superficie cui appartiene, in modo da poter essere successivamente utilizzato nelle fasi di descrizione, matching e classificazione di oggetti. La distinguibilità dipende anche dalla tipologia dei descrittori locali applicati al keypoint;
- *Efficienza computazionale*.

### 4.2.2 Descrizione e Matching

Una volta individuati i keypoint, questi vengono associati ad un descrittore locale. Tutti i descrittori locali (vedi 3.3.1) si basano sull'utilizzo di una struttura geometrica come supporto per la determinazione del vicinato di punti da usare per il calcolo delle caratteristiche geometriche di una superficie. La fase di *descrizione* viene completata calcolando il set di descrittori locali scelto per tutti i modelli di oggetti da ricercare all'interno delle scene. La fase successiva (*matching*) prevede di calcolare gli stessi descrittori su tutti i keypoint individuati nella scena, al fine di confrontarli con i descrittori dei modelli. Per gestire correttamente il caso in cui la scena contenga più oggetti dello stesso tipo, il confronto viene eseguito attraverso un'iterazione sui punti della scena (e non a partire dai punti dei modelli). Una coppia di punti  $(p_m, p_s)$ ,  $p_m$  punto del modello e  $p_s$  punto della scena, può essere definita *corrispondenza* se la loro distanza euclidea è inferiore ad una certa soglia, scelta a priori. Per rendere ancora più efficiente la fase del confronto è possibile applicare particolari schemi di confronto sui vicinati della corrispondenza trovata: un esempio è l'algoritmo *FLANN* (*Fast Approximate NN*).

### 4.2.3 Calcolo delle corrispondenze

Il risultato della fase di confronto è l'ottenimento di un insieme di corrispondenze punto-punto. Questo è lo stesso risultato ottenibile da un algoritmo di confronto su immagini 2D. Nel caso di insiemi di dati 3D è però necessaria un'ulteriore fase, detta *raggruppamento delle corrispondenze*, che permette di filtrare l'insieme di corrispondenze per rinforzare la consistenza geometrica dell'insieme di corrispondenze stesse. Una delle assunzioni alla base della fase di riconoscimento, infatti, prevede che la trasformazione tra il modello di un oggetto e la sua istanza nella scena sia completamente rigida. Per ottenere questa consistenza, l'intero gruppo di corrispondenze è suddiviso in diversi sottogruppi, ognuno dei quali contiene l'informazione (o consenso) relativamente ad una specifica rotazione e traslazione del modello sulla scena. I sottogruppi formati da un numero troppo piccolo di punti (basato su una certa soglia) vengono scartati. Questa prima scrematura dei sottogruppi è fondamentale poiché per calcolare la posa 6DOF di un oggetto è necessario un numero minimo di tre corrispondenze.

Per ottenere sottogruppi geometricamente consistenti è possibile procedere nel seguente modo:



- individuare una corrispondenza  $c_i = (p_i^m, p_i^s)$  forte, ovvero con distanza euclidea tra  $p_i^m$  e  $p_i^s$  sufficientemente piccola;
- considerare tutte le corrispondenze  $c_j = (p_j^m, p_j^s)$  non ancora inserite in nessun sottogruppo;
- aggiungere  $c_j$  allo stesso sottogruppo  $G_i$  cui appartiene  $c_i$  se

$$| \|p_i^m - p_j^m\|^2 - \|p_i^s - p_j^s\|^2 | < \epsilon$$

dove  $\epsilon$  è la soglia che rappresenta la dimensione dell'insieme di consenso.

#### 4.2.4 Calcolo dell'orientamento

Nella sezione 4.2.3 si è visto come, a differenza del caso 2D, nel riconoscimento 3D è necessario avere a disposizione set di punti che mantengano una consistenza geometrica. Anche questa selezione, tuttavia, non permette di garantire che tutte le corrispondenze rimaste siano consistenti con un'unica posa 6DOF, ovvero con un'unica rotazione 3D rigida e una traslazione del modello sulla scena.

Pertanto, è necessario aggiungere ancora un passo di stima basata su uno stimatore della media il cui scopo è quello di ridurre ulteriormente il numero di punti in ogni gruppo eliminando quelli non consistenti con una ben specifica posa. Un esempio di stimatore è il *RANSAC* (*Random Sample Consensus*). Dato un insieme di corrispondenze esatte tra due nuvole di punti, lo stimatore determina la matrice di rotazione  $3 \times 3$   $\bar{\mathbf{R}}$  e il vettore di traslazione  $\bar{\mathbf{t}}$  che meglio definiscono la trasformazione rigida che permette di passare dal modello alla scena. Questo problema (vedi 4.5) è generalmente risolto tramite una minimizzazione ai minimi quadrati: dato un insieme di  $N$  corrispondenze  $c_1, \dots, c_n$ ,  $\bar{\mathbf{R}}$  e  $\bar{\mathbf{t}}$  sono ottenute dalla minimizzazione (4.17):

$$\arg \min_{\mathbf{R}, \mathbf{t}} \sum_{i=1}^N \|p_i^s - \mathbf{R} \cdot p_i^m - \mathbf{t}\|^2. \quad (4.17)$$

La soluzione può essere espressa in coordinate omogenee o non-omogenee.

### 4.3 Riconoscimento tramite descrittori globali

La sequenza di fasi per il riconoscimento di oggetti 3D utilizzando descrittori globali è simile a quella dei descrittori locali se non fosse per una fondamentale differenza: nel caso dei descrittori globali l'oggetto viene descritto nella sua interezza e non solo attraverso alcuni suoi punti. Pertanto, prima di arrivare alle fasi di descrizione, matching e calcolo della posa, è necessario individuare i vari oggetti componenti la scena tramite il processo di *segmentazione*.

### 4.3.1 Segmentazione

Nel caso delle immagini 2D la segmentazione è un problema ben noto. Alcune soluzioni al problema della segmentazione basate sull'analisi di immagini a colori o in scala di grigio sono:

- sottrazione dello sfondo;
- blob search (segmentazione basata su aree);
- segmentazione basata su contorni.

Nel caso delle nuvole di punti RGB (dati RGB-D), i metodi utilizzabili in prima istanza sono esattamente quelli appena elencati. A questi, tuttavia, vanno aggiunte una serie di tecniche basate sull'informazione di profondità aggiuntiva. Tra questi si possono elencare:

- *Differenze tra nuvole di punti*. Simile al metodo della sottrazione dello sfondo;
- *Raggruppamento euclideo*. Suddivisione della nuvola di punti in una serie di sottogruppi distanti tra loro una certa quantità (simile alla segmentazione k-mean);
- *Estrazione di poligoni e solidi*. Rilevamento all'interno della nuvola di punti rappresentante la scena di sottogruppi di punti disposti come solidi (cilindri, sfere, ...);
- *Segmentazione delle normali*. Estrazione di superfici le cui normali hanno una particolare direzione. L'esempio tipico è l'estrazione di superfici planari, caratterizzate da un elevato numero di punti con normali alla superficie in una stessa direzione.

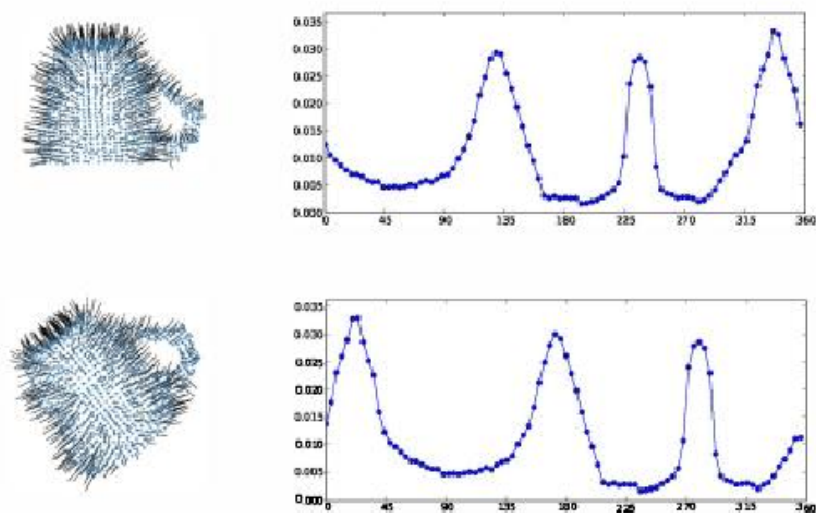
### 4.3.2 Descrizione e Matching

Il risultato ideale della segmentazione è una nuvola di punti formata unicamente dai punti appartenenti all'oggetto da estrarre dalla scena. La forma e geometria di ognuno di questi oggetti è descritto per mezzo di un descrittore globale e rappresentato da un singolo istogramma (o, nel caso del descrittore CVFH, 3.3.2, da multipli istogrammi). Questo istogramma viene comparato indipendentemente con l'istogramma ottenuto nella fase di training, ottenendo le migliori  $N$  corrispondenze. Le corrispondenze non sono coppie di punti come nel caso dei descrittori locali (4.2.3), ma l'insieme di  $N$  viste del modello che possono essere sovrapposte alla scena applicando una trasformazione. Tipicamente il confronto è effettuato con un approccio a forza bruta: ogni istogramma della scena è confrontato punto-punto con ogni istogramma di ogni punto di vista del modello. Tale approccio è possibile perché, a differenza dei descrittori locali, il numero totale di confronti è limitato dal fatto che il numero totale di istogrammi utilizzati per descrivere il modello di un oggetto è proporzionale al livello di dettaglio scelto nella fase di training (4.1).

### 4.3.3 Camera Roll Histogram

Le viste parziali ottenute come candidate per il riconoscimento possono essere utilizzate per allineare il modello alla scena utilizzando i rispettivi centroidi per ottenere una posa 5DOF. Il rimanente grado di libertà è collegato alla rotazione attorno all'asse della telecamera. Le caratteristiche geometriche usate per la costruzione dei descrittori globali sono invarianti rispetto a questa rotazione, dal momento che essa non introduce variazioni nella geometria della parte visibile dell'oggetto.

Una possibile soluzione al problema dell'allineamento del sesto grado di libertà è data dal calcolo del Camera Roll Histogram (figura 4.36): nella fase di training è possibile costruire un ulteriore istogramma ruotando la telecamera attorno al proprio asse per poi utilizzare tale istogramma nella fase di ottimizzazione della posa dell'oggetto.



*Figura 4.36: Camera Roll Histogram*

## 4.4 Postprocessing: affinamento dei risultati

Al termine di entrambe le sequenze di riconoscimento può essere necessario migliorare il risultato del riconoscimento. Un primo passo è l'utilizzo dell'algoritmo *Iterative-Closest-Points* (3.1) per raffinare la stima della posa 6DOF dell'oggetto all'interno della scena.

Un passo successivo, noto in letteratura col termine *verifica d'ipotesi*, ha lo scopo di ridurre il numero di punti falsi individuati, mantenendo inalterato il numero di corrispondenze calcolate correttamente. La verifica viene eseguita a partire dalle caratteristiche geometriche che possono essere calcolate facilmente una volta ottenuto il primo

allineamento tra modello e oggetto della scena. Caratteristiche usate tipicamente sono la percentuale di punti di corrispondenze forti e la percentuale di outlier (numero di punti visibili appartenenti al modello che non hanno una controparte nei punti della scena).

## 4.5 Stima della posa dal modello 3D

Nel paragrafo 3.3 del capitolo 3 si è già discusso del problema di individuazione della posa 3D di un oggetto a partire dalla sua proiezione sul piano immagine. La formulazione del problema è la stessa anche nel caso in cui le informazioni estratte dalle immagini siano puramente tridimensionali. In particolare, si tratta sempre di trovare una soluzione ai problemi *algebrico* e *geometrico*. Tuttavia, la soluzione è resa più semplice dal semplice fatto di dover ricercare una trasformazione geometrica tra due spazi geometrici aventi stesse dimensioni.

Se lo spazio del modello e della scena hanno le stesse dimensioni  $n$ ,  $\mathbf{x}$  e  $\mathbf{y}$  hanno la stessa dimensione e la matrice di proiezione  $P$  si riduce ad un'omografia  $n \times n$ . Di conseguenza i dati dell'immagine  $\mathbf{y}_i$  possono essere elaborati per rimuovere  $P$ , ottenendo la relazione (4.18).

$$\bar{\mathbf{y}}_i = P^{-1} \mathbf{y}_i . \quad (4.18)$$

La soluzione dell'errore algebrico è riformulato dalla (4.19).

$$T_{alg}^* = \arg \min \sum_{i=1}^N \|\bar{\mathbf{y}}_i \times (T \cdot \mathbf{x}_i)\|^2 . \quad (4.19)$$

Il problema di minimizzazione può essere risolto in un passo solo se  $T(\mathbf{p})$  è parametrizzata in modo lineare. Infine, se l'ultima riga della matrice  $T$  è  $[0, \dots, 0, 1]$  anche la soluzione dell'errore geometrico si riduce al problema di minimizzazione (4.20), anch'esso lineare per se  $T$  è parametrizzata in modo lineare.

$$T^* = \arg \min \sum_{i=1}^N \|T \cdot \mathbf{x}_i - \bar{\mathbf{y}}_i\|^2 . \quad (4.20)$$

## 5 Conclusioni

Il capitolo ha proposto una panoramica delle tecniche di acquisizione ed elaborazione di dati tridimensionali. L'utilizzo dell'informazione di profondità, in aggiunta alla classica

informazione presente in un'immagine bidimensionale, permette di irrobustire il processo di estrazione delle caratteristiche principali di una scena. Inoltre, l'informazione 3D risulta fondamentale per certe tipologie di applicazioni, in cui l'individuazione di un oggetto e della sua posa all'interno di una scena è impossibile a partire dalle semplici proiezioni dell'oggetto su un piano immagine. Tuttavia, nella pratica l'integrazione di questo tipo di tecnologia in sistemi di manipolazione complessi non è semplice. Se infatti da un lato il costo dei dispositivi di acquisizione di informazioni tridimensionali è continuamente in calo, d'altro canto lo sforzo computazionale richiesto per l'elaborazione di dati tridimensionali è nettamente superiore a quello richiesto dalla controparte bidimensionale. Ciò porta ad un aumento dei costi, o comunque della complessità, delle componenti dedicate all'elaborazione delle immagini. Inoltre, come si vedrà in seguito, l'utilizzo dei sistemi di visione per applicazioni di movimentazione e manipolazione di oggetti rende necessario l'affronto di tematiche e problemi che insorgono dal movimento della scena. La tematica più importante concerne probabilmente il tempo d'elaborazione: inseguire e tracciare i movimenti di una scena richiede non solo frequenze d'acquisizione elevate, ma anche tempi d'elaborazione estremamente ridotti. Per tali motivi, alcuni degli algoritmi presentati in questo capitolo non sono ancora completamente utilizzabili in applicazioni di tracciamento del movimento. Di conseguenza, la loro ottimizzazione costituisce uno degli aspetti di ricerca più interessanti nell'ambito della visione artificiale.



## Parte II

# Visual Servoing





# Capitolo 5

## Controllo in Asservimento Visivo

### Indice

---

1	Controllo in anello aperto . . . . .	<b>174</b>
1.1	Componenti e scopi del controllo . . . . .	176
1.2	Modello dei giunti . . . . .	177
1.3	Cinematica e dinamica dei manipolatori . . . . .	180
2	Vision in the Loop . . . . .	<b>191</b>
2.1	Integrazione del controllo del robot . . . . .	193
2.2	Spazio dell'errore . . . . .	194
2.3	Configurazione della telecamera . . . . .	195
3	Position-Based Visual Servoing . . . . .	<b>198</b>
3.1	Calcolo della funzione errore . . . . .	199
3.2	Legge di controllo . . . . .	202
3.3	Schema Simulink . . . . .	204
4	Image-Based Visual Servoing . . . . .	<b>206</b>
4.1	Jacobiano immagine . . . . .	207
4.2	Generazione del controllo . . . . .	212
4.3	Punti di singolarità . . . . .	214
4.4	Schema Simulink . . . . .	218
5	Conclusioni . . . . .	<b>220</b>

---

La maggior parte dei robot oggi presente sul mercato appartiene alla categoria dei robot industriali (anche detti manipolatori), cioè quelle macchine che lavorano in un ambiente costruito a loro misura. La possibilità di avere un elevato controllo sull'ambiente in cui il robot si trova ad operare permette di ottenere una grandissima affidabilità, efficienza

e ripetibilità dei compiti che il robot è chiamato a svolgere. D'altro canto, però, il numero di compiti che il manipolatore è in grado di portare a termine correttamente sono limitati dalla sua conoscenza a priori dell'ambiente e degli oggetti con cui esso ha a che fare.

Il modo più utilizzato per cercare di staccarsi da questa visione statica dell'ambiente di lavoro e raggiungere quindi una maggiore flessibilità è quello di avvalersi di differenti tipi di sensori per cercare di ottenere una rappresentazione *dinamica* dell'ambiente. Questo concetto è bene espresso dal termine *sensor fusion*, sotto il quale termine vengono raccolti gli approcci e le tecniche che permettono di unire informazioni provenienti da sensori di tipo diverso allo scopo di aumentare la comprensione dell'ambiente circostante.

Come si è detto nel capitolo 1, la visione rappresenta uno strumento importante nell'ottica dell'utilizzo per la fusione di informazioni, perché la natura di tali informazioni è completamente diversa da quella ottenibile con l'utilizzo di qualsiasi altro tipo di sensori. L'utilizzo di sistemi di visione per la guida robot prende il nome di *Visual Servoing*.

Il controllo visivo di un robot ottiene sostanziali vantaggi in tutte quelle applicazioni in cui gli oggetti con i quali il manipolatore deve interagire occupano una posizione imprecisata. In questo contesto operativo un sistema di asservimento visuale è in grado di ricostruire la posizione di un oggetto utilizzando le immagini riprese da una o più telecamere ed impartire al robot i comandi di movimentazione che permettono ad esso di raggiungere l'obiettivo prefissato.

Il Visual Servoing nasce come disciplina a se stante negli anni '70. A causa delle scarse prestazioni hardware ottenibili con i sistemi di visione d'allora, i primi lavori di fusione tra robotica e visione artificiale utilizzavano una metodologia del tipo *guarda e muovi* (*look then move*), che prevede i seguenti passi, eseguiti in modo sequenziale:

1. Acquisizione delle immagini;
2. Elaborazione delle immagini ed individuazione degli oggetti;
3. Stima della posa (posizione e orientamento) dell'oggetto;
4. Invio dei comandi al robot.

In questo modo il robot viene comandato alla cieca e l'accuratezza dell'operazione dipende sia dalla definizione dell'immagine e successiva precisione nell'elaborazione che dalla precisione dei movimenti del manipolatore. Questa pratica è ancora la più utilizzata nell'ambito dell'integrazione tra sistemi automatici e sistemi di visione e ci si può riferire ad essa col termine *controllo visivo in anello aperto*. Tuttavia, i contesti di utilizzo di questo tipo di sistemi sono limitati poiché un eventuale movimento dell'oggetto ad elaborazione già

avvenuta rende il sistema completamente inaffidabile. Pertanto, la realizzazione di questo tipo di sistemi è possibile solo predisponendo delle aree apposite dove gli oggetti rimangono fermi in attesa di essere manipolati. Esempi tipici di queste applicazioni sono la presa e spostamento di un oggetto (*pick and place*). L'estensione di questa applicazione di base è data dal caso in cui gli oggetti sono movimentati da un nastro trasportatore. Questo è però un problema decisamente più complesso del precedente perché prevede l'inseguimento del moto (*tracking*) degli oggetti.

L'estensione della classica sequenza di elaborazione delle immagini con successiva generazione del moto al problema dell'inseguimento è possibile solo integrando, all'informazione proveniente dal sistema di visione, un'informazione sensoriale di tipo diverso. Nel caso di oggetti in movimento su un nastro trasportatore è possibile, ad esempio, utilizzare la lettura tramite encoder della posizione del nastro trasportatore stesso e collegarla all'informazione temporale dell'acquisizione dell'immagine. Questo compito è reso difficile dalla necessità di dover trovare un metodo di sincronizzazione tra l'acquisizione delle immagini e l'elaborazione delle leggi di moto con cui pilotare il manipolatore.

Inoltre, l'elaborazione delle immagini comporta un onere computazionale elevato. Fino a una decina di anni fa, la limitazione dell'utilizzo di questo tipo di sistemi era imposta dall'esigua potenza dei sistemi di calcolo allora esistenti. Non appena lo sviluppo tecnologico ha permesso di ottenere elaborazioni delle immagini più veloci (ad esempio 30 o 60 frame/s) è divenuto possibile realizzare sistemi *real-time* in grado di correggere la loro strategia di azione durante l'esecuzione dell'azione stessa in funzione di ciò che il sistema di visione osserva. In poche parole è stata introdotta una vera e propria retroazione basata direttamente sulle immagini. I primi lavori di questo tipo utilizzavano il termine *visual feedback* per sottolineare che il robot veniva retroazionato direttamente sul sistema di visione. Oggi, col termine *Visual Servoing* (o *Vision in the loop*) ci si riferisce alle tecniche di controllo che prevedono l'introduzione del sistema di visione direttamente nell'anello di retroazione del controllore di un sistema meccanico generico e, nei casi più complessi, di un manipolatore.

In letteratura esiste una vasta documentazione riguardante la tematica del controllo in asservimento visivo. La conoscenza di base del problema, necessaria ad apprendere le basi di questo approccio al controllo e individuarne gli elementi fondamentali, viene fornita da [29], [32], [10] e [11]. L'analisi che viene compiuta permette di porre le basi del controllo di un sistema per lo svolgimento di compiti tipici dell'asservimento visivo, quali

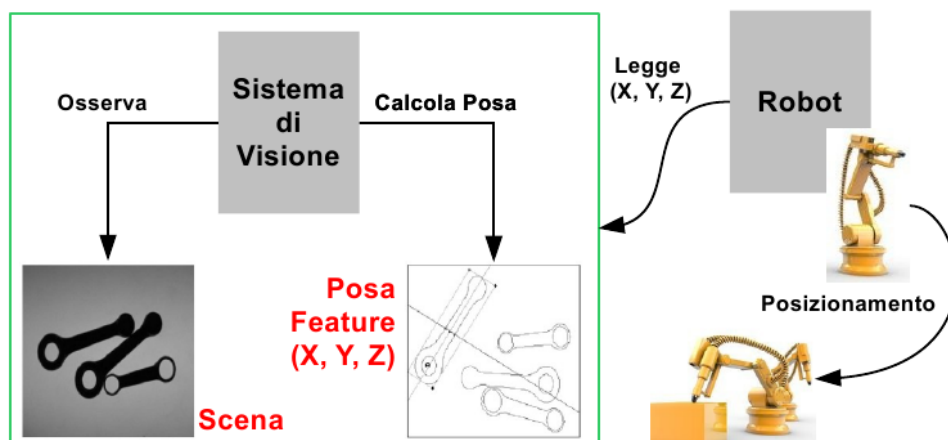
microposizionamenti ([46]) e microassemblaggi ([53]).

Per l'analisi delle tecniche di controllo dei manipolatori si è fatto riferimento a [6].

In questo capitolo vengono presentati i principali schemi di controllo in asservimento visivo. Innanzitutto, si offrono alcuni cenni al controllo in anello aperto (1) presentando i concetti base relativi al controllo di un manipolatore industriale. In secondo luogo si elencano le possibili classificazioni dei sistemi di controllo con sistema di visione in retroazione (2) per concentrarsi in seguito sull'analisi cinematica delle due tipologie di calcolo della legge di moto più usate: una basata sulla stima della posa degli oggetti nella scena (*position based visual servoing*, 3) e l'altra sull'identificazione e valutazione diretta delle feature estratte dall'immagine (4). Infine, nel paragrafo conclusivo (5) vengono riassunte le differenze tra gli approcci presentati e si traggono alcune conclusioni sulle tematiche introdotte dagli schemi di controllo, ma che necessitano di ulteriore approfondimento.

## 1 Controllo in anello aperto

Quando si parla di *controllo visivo in anello aperto*, si intende la classica integrazione di un sistema di visione con un manipolatore industriale. Questo approccio è tuttora il più usato nell'ambito dell'automazione industriale perché l'integrazione delle due componenti permette il raggiungimento di un'elevata flessibilità con uno sforzo minimo. In figura 5.1 è mostrato lo schema a blocchi delle operazioni che portano il sistema di visione a guidare i movimenti del robot.



**Figura 5.1:** Schema a blocchi del controllo visivo di un robot in anello aperto

Tale sequenza di operazioni inizia con l'acquisizione delle immagini della scena da parte del sistema di visione. In seguito, la componente dedicata all'elaborazione delle immagini si

occupa della stima della posa degli oggetti da manipolare. Infine, le informazioni relative a posizione e orientamento dell'oggetto vengono passate al controllore del robot che, autonomamente, si occupa di spostare l'end-effector del manipolatore nella posizione desiderata in modo da manipolare opportunamente l'oggetto, ad esempio portando a termine un'operazione di presa.

Come si può vedere, l'integrazione della visione col controllore del robot è semplice perché ognuna delle due componenti lavora in modo indipendente dall'altra: il sistema di visione non sa nulla di come il manipolatore si muove e il robot non conosce come le informazioni relative alla posa dell'oggetto vengono calcolate. Pertanto, programmato il controllore del robot e il sistema di acquisizione ed elaborazione delle immagini, per integrare le due componenti è sufficiente concordare:

- *Interfaccia e protocollo di comunicazione.* Definisce il tipo e il numero di dati da scambiare e come la comunicazione deve avvenire;
- *Sistema di riferimento comune.* La posa degli oggetti deve venire calcolata e comunicata al manipolatore rispetto ad un sistema di riferimento ad esso noto. È possibile utilizzare un sistema di riferimento assoluto, oppure è necessario che una delle due componenti conosca l'origine del sistema di riferimento dell'altra, in modo da operare le trasformazioni geometriche necessarie.

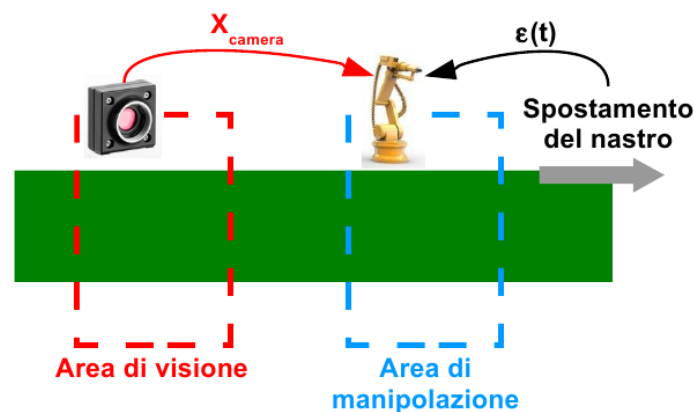
Gli algoritmi di elaborazione delle immagini trattati nei capitoli 3 e 4 permettono di ottenere la posizione di un punto nel sistema di riferimento della telecamera. Ipotizzando di conoscere la trasformazione geometrica rigida  $\mathbf{T}_{\text{robot}}^{\text{camera}}$  tra il sistema di riferimento del manipolatore e quello del sistema di visione, è possibile esprimere tutte le coordinate dei punti individuati dalla visione direttamente nello spazio di lavoro del robot:

$$\mathbf{X}_{\text{robot}} = \mathbf{T}_{\text{robot}}^{\text{camera}} \cdot \mathbf{X}_{\text{camera}} + \epsilon(\mathbf{t}) \quad (5.1)$$

in cui la componente  $\epsilon(t)$  rappresenta l'informazione non ottenibile dal sistema di visione, ma comunque necessaria alla corretta individuazione della posa degli oggetti da manipolare. Si può notare che questa componente dipende dal tempo perché permette generalmente di mappare il movimento degli oggetti target. In particolare, nell'ambito dell'automazione industriale si possono individuare due casi tipici:

- Oggetti posizionati in un'area di prelievo dedicata. In questo caso  $\epsilon(t) = 0$ , poiché gli oggetti da manipolare sono fermi in attesa dell'arrivo del manipolatore;

- Oggetti movimentati da un *nastro trasportatore*, che è anche il piano d'appoggio dell'area di manipolazione, come mostrato in figura 5.2. Allo stesso modo del caso precedente, il sistema di visione riesce ad individuare e ottenere informazioni solo in un'area limitata. Se l'area di manipolazione coincide con l'area di visione, la componente  $\epsilon(t)$  non è comunque nulla, ma risulta pur sempre stimabile dalla componente di elaborazione immagini; infatti, la stima delle coordinate risulta influenzata dai ritardi di elaborazione e comunicazione delle informazioni. Il caso peggiore, invece, è quello in cui le due aree non coincidono, ovvero l'area di manipolazione è posta più avanti lungo il percorso del nastro trasportatore. In questa situazione il contributo fornito da  $\epsilon(t)$  è a tutti gli effetti una componente esogena, stimabile solo tramite l'utilizzo di altre tipologie di sensori, quali, ad esempio, encoder per il tracciamento del movimento del nastro; l'errore di posizionamento risulta pertanto influenzato non solo dalla precisione del sistema di visione, ma anche da eventuali errori di lettura della posizione del nastro, dovute, ad esempio, da deformazioni fisiche dello stesso.



**Figura 5.2:** Guida robot in anello aperto con oggetti movimentati da nastro trasportatore

Nel proseguo del paragrafo vengono accennati i concetti basilari necessari alla comprensione delle varie componenti coinvolte nel controllo di un manipolatore.

## 1.1 Componenti e scopi del controllo

In prima istanza è possibile definire un manipolatore come un insieme di componenti meccaniche (*link*) connessi tra loro tramite dei *giunti*. Ogni giunto ha tipicamente un solo grado di libertà, sia esso traslazionale (*giunti prismatici*) o rotazionale (*giunto rotazionale*). In corrispondenza di ogni giunto è presente un attuatore per la trasmissione del moto. Lo scopo degli algoritmi e delle architetture di controllo del manipolatore, implementate

all'interno del *controllore* del robot, è quello di fornire il segnale di comando agli attuatori dei giunti per ottenere che la punta operativa (*end-effector*) del manipolatore esegua il compito assegnato nello *spazio cartesiano*. Tali compiti vengono definiti ad un livello gerarchico superiore tramite la *pianificazione della traiettoria*, ma la presenza di disturbi, errori del modello, perturbazioni additive e/o parametriche rende necessario un controllo in catena chiusa del moto del robot.

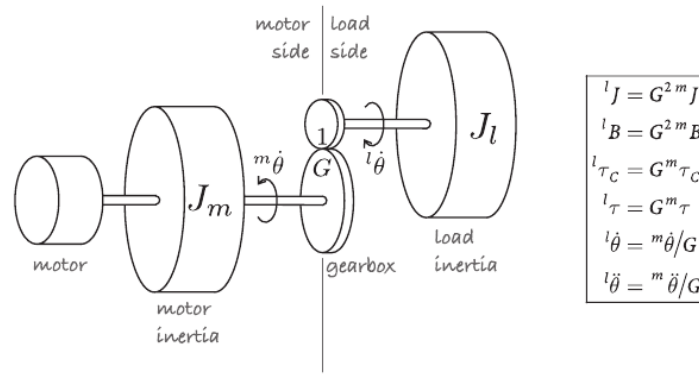
L'architettura di controllo del moto che si è venuta ad imporre fin dai primi robot realizzati è il *controllo a giunti indipendenti* che permette di trattare ogni singolo giunto come un sistema disaccoppiato indipendente dagli altri. Gli effetti di accoppiamento non lineari presenti nella dinamica propri del robot, in tale modo, vengono ridotti a disturbi trattabili localmente in ogni giunto. Il principale motivo per cui questa tecnica di controllo risulta la più utilizzata risiede nel fatto di poter suddividere il controllo in due componenti principali:

- *Controllo locale*. Il suo compito è quello di comandare, mediante una retroazione locale, il movimento dei singoli giunti: l'attuatore di ogni giunto deve essere in grado di raggiungere in modo autonomo il riferimento di coppia, velocità o posizione imposto;
- *Controllo globale*. Si occupa di gestire la traiettoria dell'end-effector del robot, ovvero di calcolare la posizione di ogni giunto ad ogni istante di tempo per ottenere la traiettoria desiderata.

## 1.2 Modello dei giunti

Ad oggi, la grande maggioranza dei robot utilizza *attuatori elettrici* per la movimentazione dei giunti. Tipicamente i robot industriali utilizzano motori *brushless*, mentre per applicazioni da laboratorio il motore più utilizzato è il *motore in corrente continua*. Questi motori sono caratterizzati da dimensioni ridotte e un'alta efficienza che permette loro di generare velocità elevate, ma in genere non un valore di coppia sufficientemente grande. Pertanto, vengono utilizzati dei *meccanismi di trasmissione* o *riduttori* per aumentarne la coppia. Lo svantaggio risiede in un aumento dei costi e del peso e nell'introduzione di non linearità, quali attriti, all'interno del modello del robot. Detto  $G$  il rapporto di riduzione tra la velocità di rotazione del motore e la velocità del carico

$$G = \frac{\dot{\theta}^m}{\dot{\theta}^l}$$



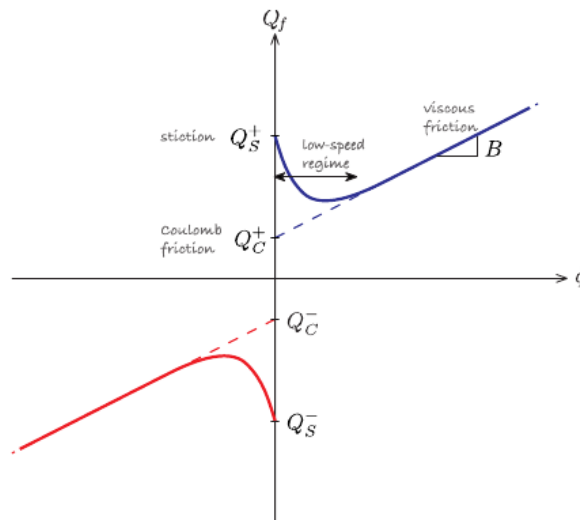
**Figura 5.3:** Schema di una trasmissione

le relazioni tra l'inerzia, la coppia e l'accelerazione percepita a monte e valle del riduttore sono espresse in figura 5.3.

L'inerzia  $J_{tot}$  vista dal motore ha due componenti principali: una dovuta all'inerzia del rotore del motore stesso ( $J_m$ ) e l'altra dovuta al carico collegato al giunto ( $J_l$ ), in cui entra il rapporto di riduzione scelto:

$$J_{tot} = J_m + \frac{1}{G^2} J^l. \quad (5.2)$$

L'altra componente fondamentale per la modellazione dei giunti e, in particolare modo, della *coppia resistente* vista da un giunto è l'*attrito*. Per ogni motore, la relazione tra coppia dovuta all'attrito e velocità ha una forma simile a quella mostrata in figura 5.4.



**Figura 5.4:** Rapporto attrito-velocità

A velocità nulla, al fine di mettere in moto il motore deve essere vinta la coppia d'attrito



*statico*. Una volta in movimento, la forza d'attrito statico diminuisce rapidamente e domina la coppia dovuta all'attrito *dinamico*, ovvero

$$Q_f = B\dot{q} + Q_c . \quad (5.3)$$

La componente coulombiana dell'attrito  $Q_c$  è modellata tramite la funzione non lineare:

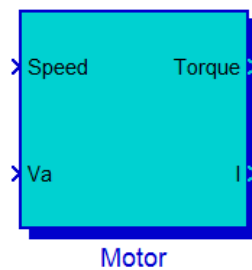
$$Q_c = \begin{cases} 0 & , \quad \dot{q} = 0 \\ Q_c^+ & , \quad \dot{q} > 0 \\ Q_c^- & , \quad \dot{q} < 0 \end{cases} .$$

Il calcolo del carico inerziale e delle componenti d'attrito risulta fondamentale per la creazione di un modello dell'attuatore di un giunto del manipolatore. Tale modello, infine, permette l'analisi della dinamica dell'attuatore, la quale risulta fondamentale per lo sviluppo e studio della dinamica dei controllori del moto del manipolatore.

In generale, un attuatore elettrico è comunque modellabile tramite una funzione che permette di convertire un segnale elettrico  $V$  in un valore di coppia  $Q$ :

$$Q = f_{actuator}(V_a) . \quad (5.4)$$

In figura 5.5 è mostrato il blocco Simulink sviluppato per descrivere il comportamento del motore in corrente continua. Il segnale  $V_a$  è la tensione in ingresso, **Torque** la coppia in uscita, **Speed** rappresenta un'eventuale ingresso di un ramo di feedforward utile a migliorare l'inseguimento delle traiettorie.



**Figura 5.5:** Blocco Simulink di un attuatore

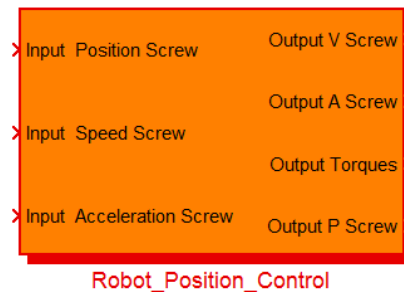
Nel paragrafo 1 del capitolo 7 viene descritto in dettaglio il contenuto di questo blocco, presentando il modello dinamico di un motore a corrente continua, al fine di valutarne l'impatto sullo sviluppo dei controllori in asservimento visivo.

### 1.3 Cinematica e dinamica dei manipolatori

Nel controllo a giunti indipendenti, ma, più in generale, nella descrizione del moto del manipolatore, si possono distinguere due spazi di coordinate:

- *Spazio operativo* (o *spazio cartesiano*). Definisce la posizione dell'end-effector del manipolatore ( $\mathbf{x}$ ) in un sistema di riferimento direttamente collegato allo spazio di lavoro del robot;
- *Spazio dei giunti*. È costituito dall'insieme delle posizioni assunte dai giunti ( $\mathbf{q}$ ), ovvero dagli attuatori del robot.

In generale, quindi, il controllore di un robot può essere modellizzato come un blocco, schematizzato in figura 5.6, avente in ingresso, ad ogni istante temporale, il vettore delle coordinate dell'end-effector nello spazio di lavoro. In realtà, nota la posizione dell'end-effector nel tempo, risultano definiti anche i suoi vettori velocità e accelerazione. La simulazione del controllore del robot permette di ottenere la dinamica del sistema, ovvero l'andamento nel tempo di posizione, velocità e accelerazione dell'end-effector e le coppie ai giunti applicate per ottenere questo andamento.



**Figura 5.6:** Blocco Simulink per la simulazione di un controllore di un robot

#### 1.3.1 Cinematica

La *cinematica* è la branca della meccanica che studia il moto dei corpi senza considerare le forze agenti su di essi. In relazione al controllo del manipolatore, la trasformazione *cinematica diretta*  $K$  permette di passare dallo spazio dei giunti allo spazio cartesiano, mentre viceversa si parla di trasformazione *cinematica inversa*  $K^{-1}$ :

$$\begin{aligned} \mathbf{x} &= K(\mathbf{q}) \\ \mathbf{q} &= K^{-1}(\mathbf{x}) \end{aligned} \quad (5.5)$$

Queste relazioni permettono di definire una traiettoria nello spazio di lavoro del manipolatore e, tramite la cinematica inversa, calcolare la posizione dei giunti che permettono l'esecuzione della traiettoria desiderata a livello dell'end-effector.

La relazione tra le velocità dell'end-effector e dei giunti è ottenibile sfruttando la definizione di *matrice jacobiana*. Lo jacobiano è la matrice ottenuta dalla derivazione di un vettore rispetto ad un altro vettore: data la funzione  $\mathbf{y} = F(\mathbf{x})$ , con  $\mathbf{x} \in \mathbb{R}^n$  e  $\mathbf{y} \in \mathbb{R}^m$  la matrice jacobiana è la matrice  $m \times n$ :

$$\mathbf{J} = \frac{\delta \mathbf{F}}{\delta \mathbf{x}} = \begin{bmatrix} \frac{\delta y_1}{\delta x_1} & \cdots & \frac{\delta y_1}{\delta x_n} \\ \vdots & \ddots & \vdots \\ \frac{\delta y_m}{\delta x_1} & \cdots & \frac{\delta y_m}{\delta x_n} \end{bmatrix}.$$

Applicando questa definizione alla cinematica diretta, si ottiene l'equazione (5.6) per la relazione delle velocità tra coordinate di lavoro e coordinate ai giunti:

$$\boldsymbol{\nu} = \dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \quad (5.6)$$

dove  $\boldsymbol{\nu} = \{v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\}$  è la *velocità spaziale* con le sue componenti traslazionali e rotazionali.

Tuttavia, l'operazione necessaria al controllo della traiettoria dell'end-effector di un manipolatore è ancora una volta la cinematica inversa, con lo scopo di ottenere le velocità da imporre ai giunti per ottenere il moto dell'end-effector desiderato. Se  $\mathbf{J}$  è una matrice quadrata e non singolare, ovvero

$$\det(\mathbf{J}) \neq 0$$

essa è invertibile e quindi la cinematica inversa è data semplicemente da:

$$\dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1}\boldsymbol{\nu}. \quad (5.7)$$

Questo accade quando il sistema non è *ridondante*, ovvero si utilizzano  $N$  giunti per ottenere il movimento in  $N$  gradi di libertà. Se, invece, il sistema è ridondante, cioè si hanno più giunti che gradi di libertà, oppure lo jacobiano è singolare, la soluzione non è ottenibile tramite semplice inversione. Nel caso di sistema ridondante la soluzione viene ottenuta tramite minimizzazione ai minimi quadrati (vedi capitolo 3). In particolare, la matrice  $\mathbf{J}$  viene resa quadrata e invertibile moltiplicandola per la sua trasposta  $\mathbf{J}^T$ :

$$\begin{aligned} \mathbf{J}(\mathbf{q})^T \boldsymbol{\nu} &= \mathbf{J}(\mathbf{q})^T \mathbf{J}(\mathbf{q}) \dot{\mathbf{q}} \\ \dot{\mathbf{q}} &= [\mathbf{J}(\mathbf{q})^T \mathbf{J}(\mathbf{q})]^{-1} \mathbf{J}(\mathbf{q})^T \boldsymbol{\nu} \end{aligned}$$

La matrice  $(\mathbf{J}(\mathbf{q})^T \mathbf{J}(\mathbf{q}))^{-1} \mathbf{J}(\mathbf{q})^T$  è detta matrice *pseudoinversa* ( $\mathbf{J}^+$ ) dello jacobiano e quindi la trasformazione cinematica inversa per la velocità spaziale è data da:

$$\dot{\mathbf{q}} = \mathbf{J}^+ \nu . \quad (5.8)$$

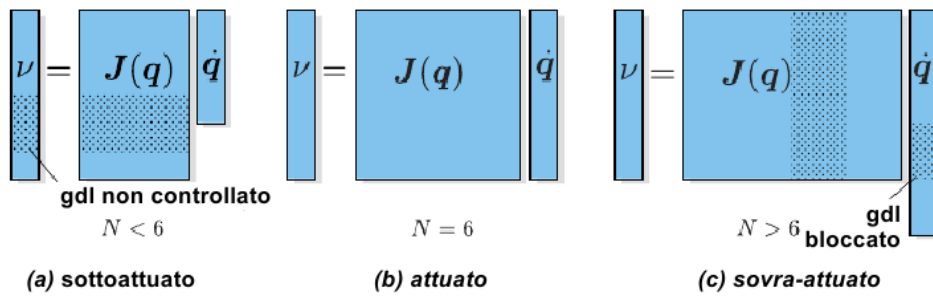
Nel caso il sistema meccanico si trovi vicino ad un punto di singolarità, il determinante dello jacobiano tende a zero. In questo caso la trasformazione cinematica inversa è scomponibile nella seguente:

$$\dot{\mathbf{q}} = (\mathbf{J}(\mathbf{q}) + p \cdot \mathbf{I})^{-1} \nu$$

dove  $p$  è una piccola costante aggiunta alla diagonale della matrice jacobiana in modo da renderne il determinante non nullo. Tuttavia, l'introduzione di questa costante porta all'insorgere di un errore in  $\dot{\mathbf{q}}$  che, integrato nel tempo, porterebbe ad avere un errore rilevante nel posizionamento dell'end-effector. Tale errore è riducibile introducendo un controllo proporzionale dell'errore di posizionamento. In particolare, la legge di controllo ad un certo istante  $k$  è esprimibile tramite le relazioni (5.9), in cui, dato l'intervallo di campionamento  $\delta t$ , la posizione  $\mathbf{q}_{k+1}^*$  è stimata ad ogni passo a partire dalla velocità imposta e la velocità da imporre è proporzionale all'errore tra la posizione desiderata  $\zeta_k^*$  e la posizione raggiunta.

$$\begin{cases} \mathbf{q}_{k+1}^* = \mathbf{q}_k + K_p \cdot (\delta t \cdot \dot{\mathbf{q}}_k^*) \\ \dot{\mathbf{q}}_k^* = \mathbf{J}(\mathbf{q}_k)^{-1} \cdot \zeta_k^* - \mathbf{q}_k \end{cases} . \quad (5.9)$$

In figura 5.7 vengono esemplificate le possibili configurazioni di un sistema meccanico rispetto ai suoi gradi di libertà e viene mostrata la forma assunta dalla matrice jacobiana.



**Figura 5.7:** Possibili configurazioni della matrice jacobiana

Per concludere la trattazione della cinematica, è possibile definire le relazioni tra le *accelerazioni* dello spazio di lavoro e dello spazio dei giunti. La relazione della cinematica

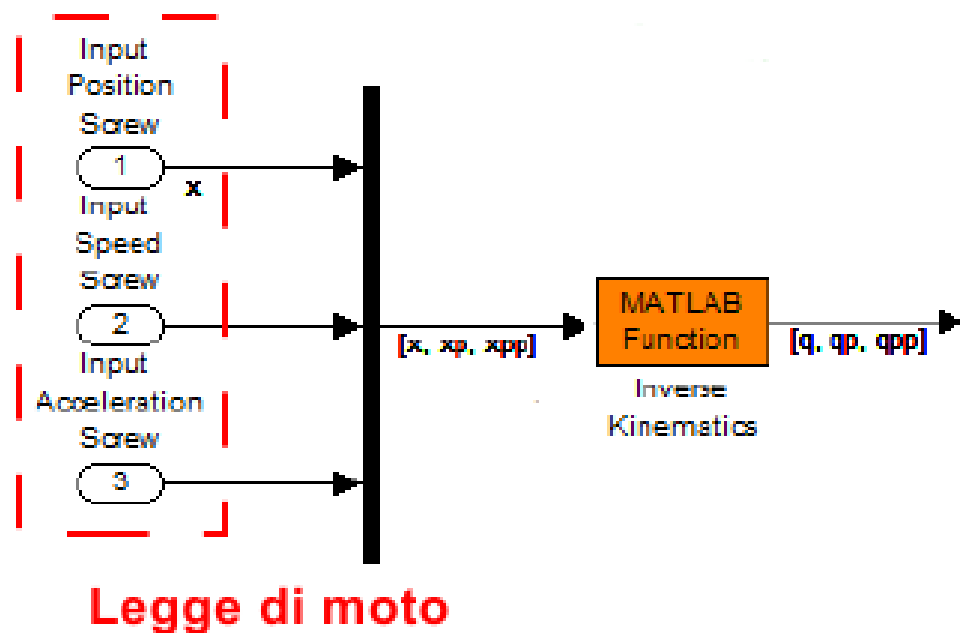
diretta è facilmente ottenibile derivando l'equazione (5.6):

$$\dot{\nu} = \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} . \quad (5.10)$$

Per quanto riguarda la cinematica inversa, tramite considerazioni simili a quelle fatte per l'applicazione alle velocità, si ottiene:

$$\ddot{\mathbf{q}} = \mathbf{J}(\mathbf{q})^+(\dot{\nu} - \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}) . \quad (5.11)$$

Le relazioni della cinematica inversa vengono utilizzate nello schema Simulink in figura 5.8 per ottenere i vettori di coordinate, velocità ed accelerazioni dei giunti.



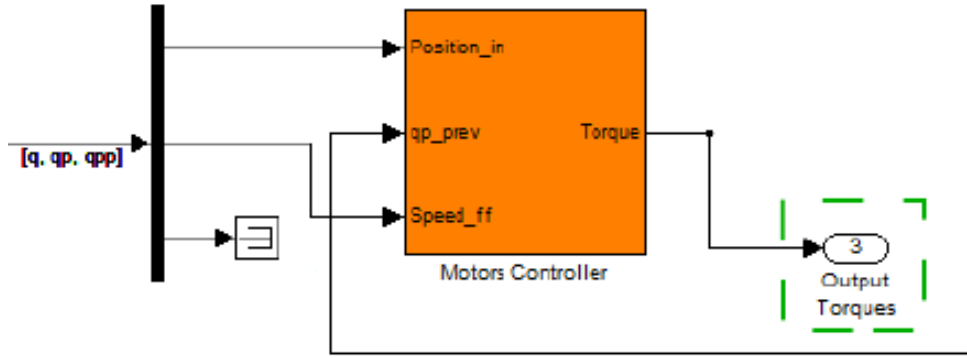
**Figura 5.8:** Inserimento della cinematica inversa nello schema Simulink per il controllo del robot

Il risultato della cinematica fornisce il punto d'ingresso per il controllore vero e proprio dei giunti.

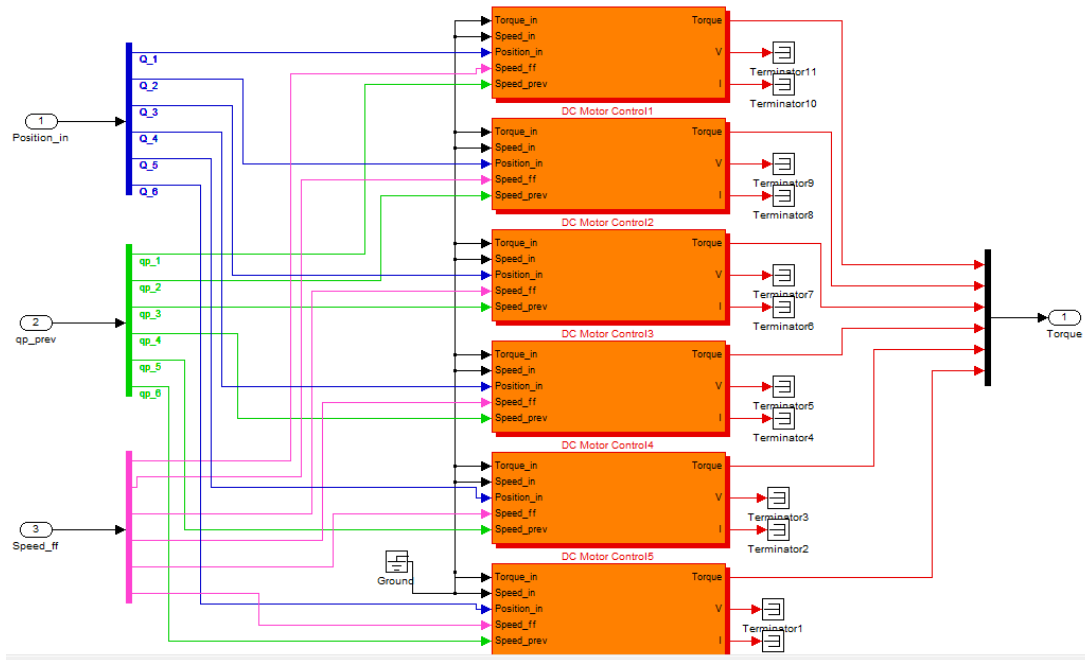
In figura 5.9 è riportato il blocco Simulink del controllore. Esso è formato da una serie di blocchi motore (figura 5.5) innestati all'interno di un anello di controllo in posizione (figura 5.10), i cui schemi con relativa dinamica vengono presentati nel paragrafo 1 del capitolo 7. In uscita si ha il vettore delle coppie ai giunti  $\mathbf{Q}$ .

### 1.3.2 Dinamica

Infine, la *dinamica* permette di estendere lo studio cinematico tramite l'introduzione delle forze agenti sui corpi. Applicandola allo studio del moto dei manipolatori, l'analisi



*Figura 5.9: Blocco Simulink del controllore a giunti indipendenti*



*Figura 5.10: Schema Simulink del controllore a giunti indipendenti*

dinamica permette di analizzare, a partire dall'ultimo link, come il suo moto influenza i link precedenti. Dati i link  $j$  e  $j - 1$ , collegati dal giunto  $j$ , si ha che l'inerzia vista dal motore  $j$  dipende sì dalla sua inerzia, ma anche da quella derivante dal carico che, in questo caso, è una composizione delle inerzie date dai giunti successivi. L'equazione (5.12) riassume tutti i contributi da considerare per lo studio della dinamica del manipolatore, in cui il risultato  $\mathbf{Q}$  è il vettore delle forze e coppie degli attuatori associati alle coordinate  $\mathbf{q}$ .

$$\mathbf{Q} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{J}(\mathbf{q})^T \mathbf{g} . \quad (5.12)$$

Nell'equazione appena vista, compaiono diverse componenti:

- $\mathbf{q}$ ,  $\dot{\mathbf{q}}$  e  $\ddot{\mathbf{q}}$  rappresentano posizione, velocità e accelerazione dei giunti;

- $\mathbf{G}(\mathbf{q})$  è il termine dovuto alla forza associata all'accelerazione di gravità. Si può notare come il contributo della gravità dipenda dalla posizione dei giunti del manipolatore;
- $\mathbf{M}(\mathbf{q})$  è la *matrice d'inerzia* del manipolatore. Gli elementi della diagonale  $M_{j,j}$  descrivono l'inerzia vista dal giunto  $j$ , mentre gli elementi  $M_{i,j}$  permettono di relazionare l'accelerazione del giunto  $j$  con la forza percepita dal giunto  $i$ .
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$  è la *matrice di Coriolis* che contiene le forze centripete applicate ai giunti;
- $\mathbf{F}(\dot{\mathbf{q}})$  rappresenta le forze resistenti dovute all'attrito;
- $\mathbf{J}(\mathbf{q})^T \mathbf{g}$  permette di mappare gli effetti di un'eventuale torsione  $\mathbf{g}$  applicata all'end-effector.

L'equazione (5.12) permette di risolvere il problema della *dinamica inversa*, ovvero l'ottenimento delle coppie ai giunti a partire dalle loro posizioni, velocità ed accelerazioni. La soluzione di questo problema trova applicazione nella pianificazione della traiettoria del manipolatore e nella scelta degli attuatori da associare ai diversi giunti del manipolatore. Per la simulazione numerica della dinamica è utile anche l'operazione inversa, ovvero la *dinamica diretta* di equazione (5.13) che permette di ottenere le accelerazioni dei giunti a partire dalle coppie e forze cui essi sono sottoposti.

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}(\mathbf{Q} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{F}(\dot{\mathbf{q}}) - \mathbf{G}(\mathbf{q}) - \mathbf{J}(\mathbf{q})^T \mathbf{g}) . \quad (5.13)$$

In figura 5.11 si può vedere come la dinamica diretta ed inversa vengano utilizzate ai fini della simulazione tramite schema Simulink per ricostruire il movimento effettivo del sistema a partire dalle coppie ai giunti imposte.

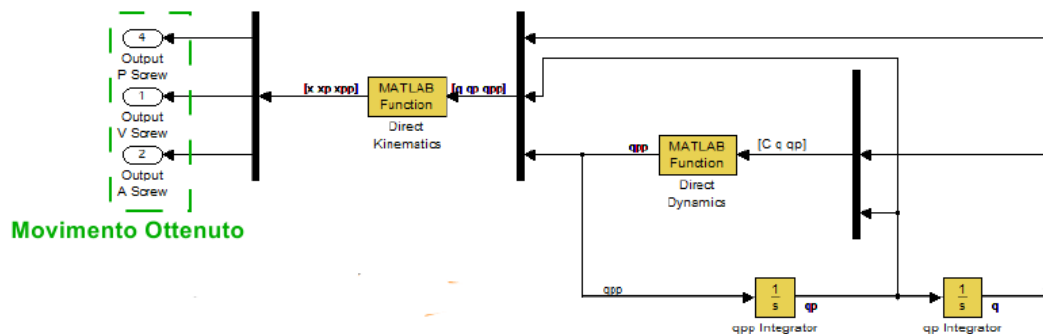


Figura 5.11: Utilizzo della dinamica al fine della simulazione

### 1.3.3 Schema completo

Per completezza, nella figura 5.12 viene mostrato l'intero schema Simulink utilizzato per la simulazione del controllore del manipolatore che imposta la traiettoria in funzione della posizione. Rispetto agli estratti mostrati in precedenza, lo schema completo presenta alcune aggiunte, tra cui:

- Modellazione del *campionamento* della legge di moto data come set-point;
- Introduzione di un offset sulle coordinate ai giunti dato dalla *posizione di azzeramento* del manipolatore.

Infine, per la modellazione del robot è stata creata la classe Matlab *Robot* (5.1), contenente la dichiarazione dei metodi per l'implementazione delle relazioni cinematiche e dinamiche.

**Codice 5.1:** Classe Matlab per la gestione del robot

---

```

1  classdef Simulink_Robot < handle
2      properties (SetAccess = private)
3          % homing position
4          x0;
5
6          % position limits
7          limit_up;
8          limit_down;
9
10         % Motors
11         motors;
12         tau_red_motors;
13     end
14
15     methods
16         function R = Simulink_Robot()
17             % Here, the initialization of fields
18             % ...
19             % ...
20         end
21
22         % Assign Motor to Joint
23         function setMotor (R, index, motor)
24             R.motors{index} = motor;
25             R.tau_red_motors(index) = motor.tau_red;

```



```

26     end
27
28     % Limits given into joint coordinates
29     function set_limits_joint(R, limit_up, limit_down, joint)
30         R.limit_up(joint) = limit_up;
31         R.limit_down(joint) = limit_down;
32     end
33
34     % Limits given into workspace coordinates
35     function set_limits(R, limit_up, limit_down)
36         % Limits
37         [ limit_up_v ] = R.kin_inv(limit_up , zeros(6,1), zeros(6,1) )
38         ;
39         R.limit_up = limit_up_v;
40         [ limit_down_v ] = R.kin_inv(limit_down , zeros(6,1), zeros
41         (6,1) );
42         R.limit_down = limit_down_v;
43     end
44
45     % x0 set into workspace coordinates
46     function set_x0(R, x0)
47         R.x0 = x0;
48     end
49
50     % Check limits : joint coordinates
51     function [q qp qpp] = check_limits (R, q, qp, qpp)
52         for i=1:length(q)
53             if (q(i)>R.limit_up(i))
54                 disp 'TOP LIMIT REACHED'
55                 q(i)=R.limit_up(i);
56                 qp(i) = 0;
57                 qpp(i) = 0;
58             end
59             if (q(i)<R.limit_down(i))
60                 disp 'BOTTOM LIMIT REACHED'
61                 q(i)=R.limit_down(i);
62                 qp(i) = 0;
63                 qpp(i) = 0;
64             end
65         end
66     end

```

```

64     end
65 end
66
67 methods (Abstract)
68     [ qpp ] = din_dir( robot, Q, q, qp )
69     % DIN_DIR Computes joint accelerations
70     % - C: Joint Torques (input)
71     % - q: Joint Position (input)
72     % - qp: Joint Velocity (input)
73     % - qpp: Joint Accelerations (output)
74
75     [ x, xp, xpp ] = kin_dir( robot, q, qp, qpp )
76     % KIN_DIR Computes direct kinematics of the R
77     % - q, qp, qpp = joint-space position, velocity, acceleration
78     % - x, xp, xpp = working-space position, velocity,
79         acceleration
80
81     [ Q ] = din_inv( robot, q, qp, qpp )
82     % DIN_INV Computes torques and forces from movement
83     % - C: Forces
84     % - x, xp, xpp: working-space coordinates
85
86     [ q, qp, qpp ] = kin_inv( robot, x, xp, xpp )
87     % KIN_INV Computes inverse kinematics of the R
88     % - x, xp, xpp = working-space position, velocity,
89         acceleration
90     % - q, qp, qpp = joint-space position, velocity, acceleration
91
92     [Jac, Jac_p] = calc_Jac (q);
93     % CALC_JAC Computes the Jacobian Matrix given joint
94         coordinates q
95     % - q = joint-space position
96     % - Jac, Jac_p = Jacobian matrix and its derivative
97
98     [M, C, F, G] = calc_dynamics (q, qp);
99     % CALC_DYNAMICS Computes Robot Dynamics
100    % - q, qp = joint-space position, velocity
101    % M = inertia matrix
102    % C = coriolis matrix
103    % F = friction vector

```

---

```
101      % G = gravity force
102
103      plot_robot (R, x_ee, label_ee);
104      % PLOT_ROBOT Function used to plot the robot
105      % - x_ee = end-effector position
106      % - label_ee = string to print near the end-effector plot
107  end
108 end
```

---

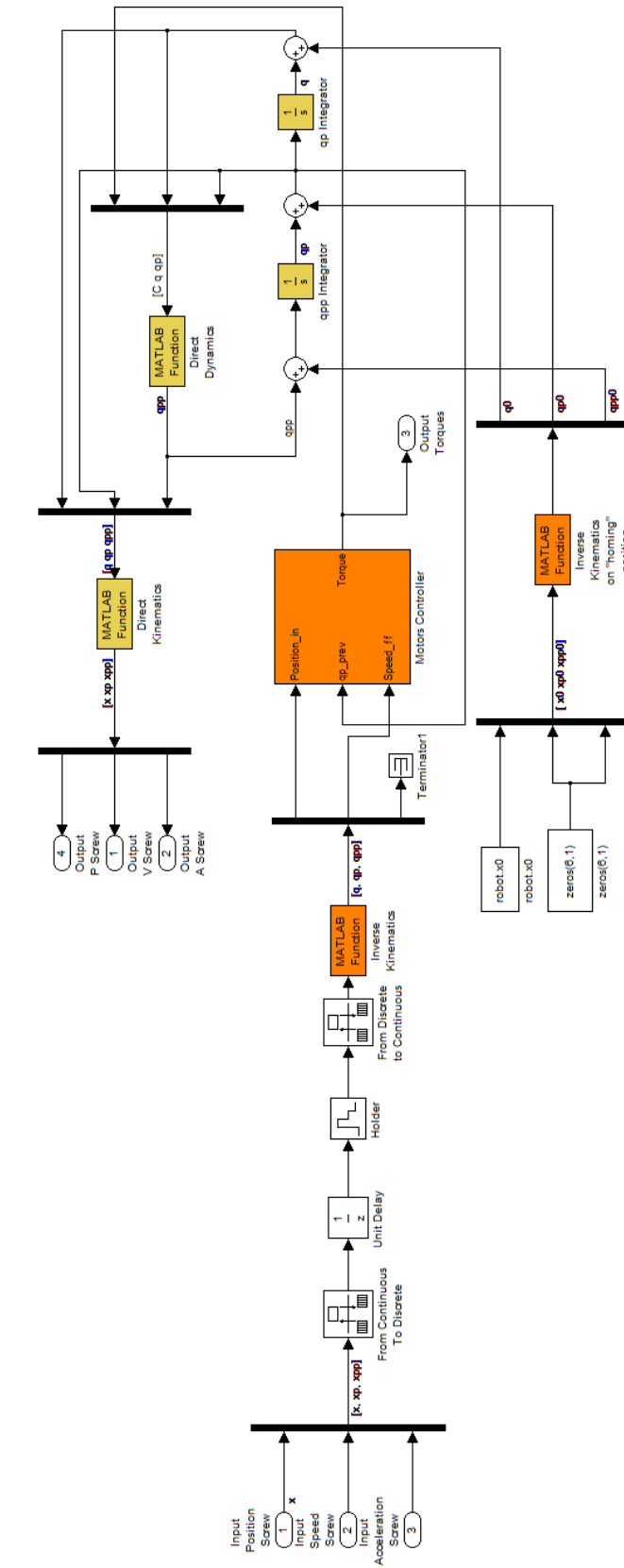
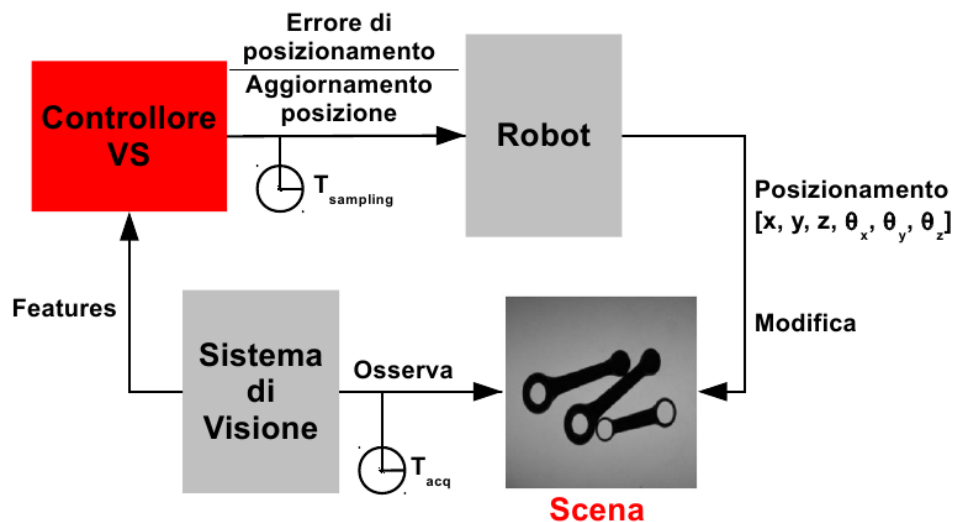


Figura 5.12: Schema Simulink per la simulazione del controllore di un robot

## 2 Vision in the Loop

Il corretto posizionamento di un task di guida robot in anello aperto è dato essenzialmente da due fattori: la (i) precisione con cui il sistema di visione identifica la posa degli oggetti da manipolare e la (ii) precisione e ripetibilità con le quali il manipolatore riesce a posizionarsi nei punti del suo spazio di lavoro. Relativamente a questo secondo punto, si è visto che un attuatore può essere visto come un dispositivo in grado di generare una coppia o forza a partire da un segnale elettrico. In realtà, le trasformazioni cinematiche permettono di ottenere la posizione del manipolatore in funzione delle coordinate ai giunti, dipendenti dalla posizione raggiunta dall'attuatore.

Per il passaggio concettuale dalla guida in anello aperto all'*asservimento visivo* può essere fatto un ragionamento del tutto analogo. Nel controllo in anello aperto il sistema di visione è visto come un sensore puntuale, in grado di acquisire informazioni solo in un certo istante temporale e sotto certe condizioni. Se, al contrario, il sistema di visione è in grado di acquisire informazioni e successivamente scambiarle col controllore del manipolatore in continuazione e per un periodo di tempo prolungato, è possibile realizzare un controllore in tempo reale, dove il set-point del task è continuamente aggiornato (figura 5.13). Dal punto di vista dello schema di controllo, questo obiettivo è raggiunto inserendo il sistema di visione nell'anello di retroazione del controllo (*Vision in the Loop*), in modo tale che esso influisca direttamente sul calcolo dell'errore di posizionamento e, quindi, sulla generazione del set-point di posizione o velocità da imporre all'end-effector del robot.



**Figura 5.13:** Schema a blocchi del controllo in asservimento visivo

In secondo luogo, è possibile ridefinire il set-point rendendolo funzione dell'immagine. In

questo senso, il compito del robot non è più quello di raggiungere un certo punto dello spazio in un certo modo, ma quello di posizionarsi in modo che il sistema di visione riesca ad osservare una certa scena definita a priori. Così facendo, il posizionamento diventa indipendente dalla precisione di stima della posa del sistema di visione e dalla precisione del controllo del manipolatore, ma dipende esclusivamente dalla capacità del sistema di visione di riconoscere una determinata scena.

Dal primo sistema di *Visual Servoing*, sviluppato nei primi anni '80, i progressi nel controllo visuale dei robot è stato abbastanza lento. I questi anni si sta assistendo ad un'accelerazione a causa della disponibilità di una potenza computazionale sempre maggiore. Ad ogni modo, la classificazione delle tecniche di controllo di asservimento visivo dovuta a Sanderson e Weiss ([55]) e datata 1980 è ancora valida. Il loro schema di classificazione pone essenzialmente due domande:

1. La struttura di controllo è gerarchica, con la visione che fornisce l'ingresso al controllore del robot, o è direttamente la visione a comandare il robot a livello dei giunti?
2. L'errore di posizionamento è definito nello spazio di lavoro del robot , o relativamente alle feature delle immagini acquisite?

Le risposte ad ognuna di queste domande sono due, e combinate tra loro generano quattro tipi di paradigmi d visual servoing, che si differenziano, quindi a seconda del tipo di controllo che si opera sul robot ed in base alle informazioni che si ottengono dall'algoritmo di visione. Tali paradigmi sono riassunti nella tabella 5.1.

**Tabella 5.1:** *Tipologie di controllo in asservimento visivo*

		Tipo di controllo	
		Dynamic Look	Direct Visual
		And Move	Servo
Informazioni	Position-Based	I Tipo	II Tipo
	Image-Based	III Tipo	IV Tipo

## 2.1 Integrazione del controllo del robot

La prima distinzione è relativa all'entità incaricata del controllo del robot:

- *Dynamic look and move.* La visione fornisce i punti nel quale il manipolatore si deve portare ed il controllore del manipolatore (diverso e separato dal sistema di visione) aziona i giunti in modo da assumere la configurazione necessaria per il raggiungimento di tali punti;
- *Direct visual servo.* Il controllore del robot viene completamente assimilato dal sistema di elaborazione delle immagini: il sistema di visione si occupa sia dell'acquisizione dell'informazione visiva che della sua trasformazione nella legge di controllo del manipolatore.

Quasi tutti i sistemi adottano il primo tipo d'architettura in quanto prevede la suddivisione dei task su più entità. Questo fatto porta innanzitutto un vantaggio dal punto di vista computazionale ed implementativo, poiché i compiti che devono essere eseguiti rispettivamente dal sistema di acquisizione ed elaborazione immagini e dal controllore del manipolatore sono ben separati e indipendenti l'uno dall'altro. A differenza dell'approccio *look then move*, tuttavia, il sistema d'elaborazione delle immagini deve avere a disposizione la conoscenza e gli strumenti necessari a costruire una legge di moto del manipolatore più avanzata. I problemi relativi al controllo del moto del robot vengono però interamente delegati al suo controllore. Infatti, il trattamento delle singolarità cinematiche del meccanismo vengono completamente separate dal controllo in asservimento visivo propriamente detto e l'integrazione presenta pertanto una difficoltà in meno. Infine, dal punto di vista dell'implementazione software delle varie componenti (sistema di visione e di controllo), si può dire che questa implementazione è la più semplice, visto che assicurare l'elaborazione delle immagini e il contemporaneo controllo della cinematica del manipolatore è un compito complesso sia dal punto di vista computazionale che del controllo.

Al contrario, dal punto di vista del raggiungimento della completa integrazione tra un sistema meccanico e i suoi sensori, l'approccio diretto è quello che unisce più strettamente il progetto del controllore in asservimento visivo con l'effettivo pilotaggio degli attuatori dei giunti del robot. Questa completa integrazione, seppur più complessa da realizzare, permette la costruzione di sistemi più compatti e, di conseguenza, offre la possibilità di costruire un prodotto più compatto, chiuso e definito.

## 2.2 Spazio dell'errore

Una seconda classificazione si basa, invece, sul modo attraverso cui si converte la posa dell'oggetto in un comando di movimentazione del robot:

- *Position-based*. Le informazioni estratte dall'immagine (feature) vengono usate per ricostruire la posa (posizione/orientamento) 3D corrente dell'oggetto rispetto alla telecamera. Si genera un segnale di errore cartesiano dalla differenza tra la posa 3D desiderata e quella attuale;
- *Image-based*. L'errore viene calcolato in 2D nel piano immagine, senza stimare la posizione degli oggetti esaminati. Il robot si muove in modo da portare le feature 2D correnti osservate verso i loro valori desiderati.

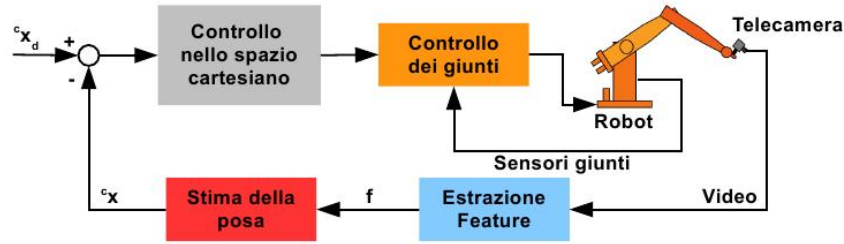
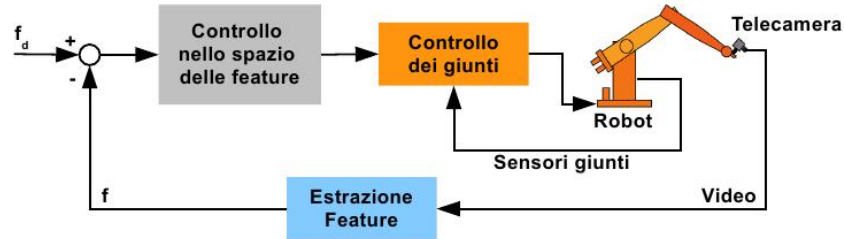
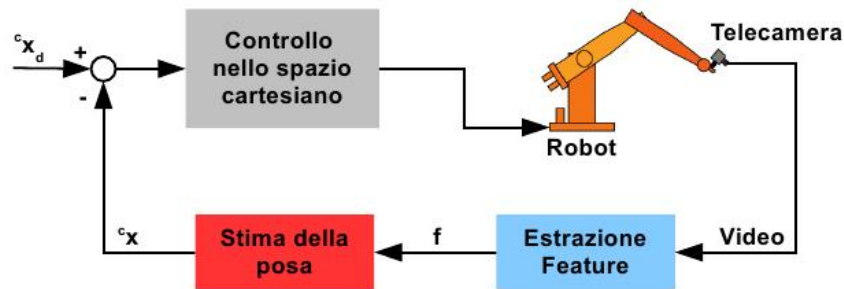
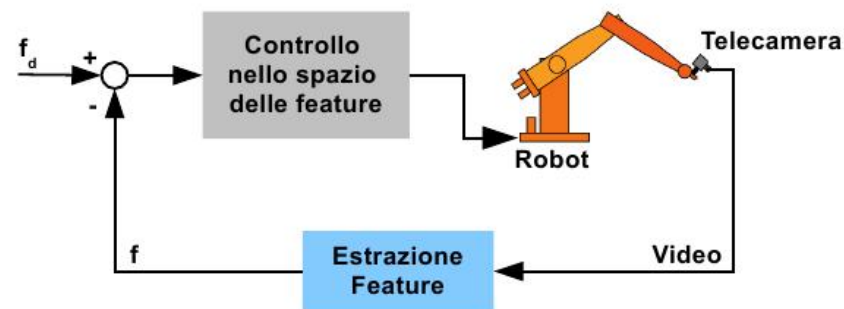
In figura 5.14 vengono riportati gli schemi di controllo dei quattro tipi di asservimento visivo ottenuti combinando la classificazione basata sull'integrazione del controllo col sistema di elaborazione immagini e sulla tipologia di informazioni da cui dipende il calcolo della legge di controllo.

L'approccio basato su immagini presenta alcuni vantaggi, tra cui si possono enumerare:

- riduzione del ritardo computazionale perché l'elaborazione della legge di controllo sfrutta direttamente le feature dell'immagine, senza la necessità di passare prima dalla stima della posizione dei punti nello spazio cartesiano;
- eliminazione di interpretazione dell'immagine. Un eventuale interpretazione serve solo nel caso in cui l'identificazione delle feature all'interno dell'immagine debba passare dal riconoscimento di un modello;
- eliminazione degli errori dovuti alla modellazione dei sensori e alla calibrazione della telecamera e possibilità di correggere errori di posizionamento dell'end-effector. Non è necessario passare dalla fase di creazione di un modello degli oggetti da manipolare, poiché la fase di apprendimento del set-point può essere eseguita mostrando direttamente al sistema di visione come deve vedere la scena.

Tuttavia, dal punto di vista del controllo questo rappresenta l'approccio più complesso da implementare perché consiste nel controllo di un sistema *MIMO* (*Multiple Input Multiple Output*) non lineare i cui ingressi sono fortemente accoppiati. Come si vedrà in seguito, infatti, il set-point del controllo viene fornito direttamente nello spazio delle feature dell'immagine ed è pertanto costituito da un vettore di punti in coordinate del piano immagine.

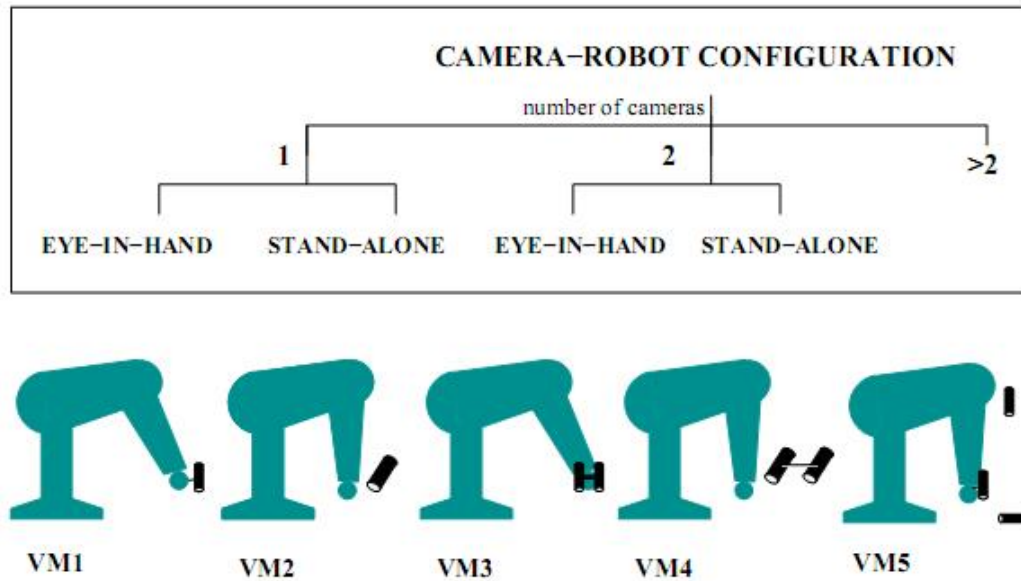


(a) *Dynamic Position-Based Look And Move.*(b) *Dynamic Image-Based Look And Move.*(c) *Position Based Visual Servo.*(d) *Image Based Visual Servo.***Figura 5.14:** *Tipologie di controllo Visual Servoing*

## 2.3 Configurazione della telecamera

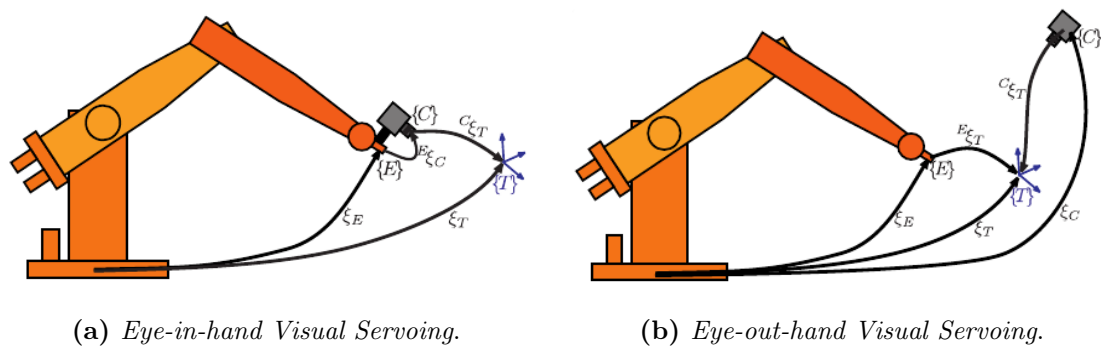
Infine, è possibile fornire ulteriori criteri di classificazione in funzione della posizione relativa tra l'end-effector del manipolatore e la telecamera, di cui in figura 5.15 viene mostrata la classificazione utilizzata in letteratura.

Nel caso in cui la telecamera inquadrì sia l'oggetto target che l'end-effector del manipolatore



**Figura 5.15:** Tipologie di controllo in asservimento visivo in funzione della configurazione telecamera-manipolatore

si parla di sistemi *Eye to Hand*, caratterizzati da una telecamera che mantiene una posizione fissa o al più con orientamento variabile. Nei sistemi *Eye in Hand*, al contrario, la telecamera è fissata sul manipolatore, tipicamente sul suo organo terminale, e tramite il movimento del robot si cerca di mantenere il target nella visuale della telecamera.



**Figura 5.16:** Classificazione del controllo in asservimento visivo in funzione della posizione della telecamera

In figura 5.16 vengono mostrate le due configurazioni possibili insieme alle trasformazioni di coordinate necessarie al calcolo del controllo. In entrambi i casi deve essere nota la posa della telecamera  $\{C\}$  rispetto all'origine del sistema di riferimento del manipolatore. Tuttavia, nel caso eye-out-hand la posizione della telecamera è nota perché fissa, mentre nel caso eye-in-hand essa è data dalla serie della soluzione della cinematica diretta combinata

con la trasformazione rigida che descrive la posizione della telecamera rispetto all'end-effector. A questo punto, la posizione dell'oggetto target  $\{T\}$  è ottenuta nel sistema di riferimento della telecamera, ma risulta sempre definibile nel sistema di riferimento del manipolatore. Il caso eye-out-hand prevede generalmente l'utilizzo di una trasformazione geometrica in più, data dalla posa relativa dell'oggetto rispetto all'end-effector.

Una variazione a questi modelli include il posizionamento di telecamere multiple, con una telecamera montata sul braccio con lo scopo di agganciare l'oggetto target ed una telecamera esterna che monitora il compito del robot, osservando contemporaneamente l'oggetto da manipolare e l'end-effector del robot. L'uso di telecamere multiple porta a soluzioni ibride che compensano i difetti delle due tipologie, ma, oltre ad avere un costo più elevato, presentano anche problemi di gestione, sincronizzazione ed elaborazione di immagini multiple e pertanto non vengono in generale prese in considerazione per l'implementazione della legge di controllo. Una soluzione ampiamente utilizzata prevede invece l'utilizzo di una sola telecamera, ma con più compiti. Un caso d'uso che si può prendere in considerazione per la spiegazione di questo concetto è quello di una telecamera montata su un robot mobile fornito di un braccio meccanico con lo scopo di aprire una porta; la prima fase di identificazione e avvicinamento alla maniglia può essere eseguito tramite una tecnica position-based con stima della posa 3D della maniglia all'interno dell'ambiente in cui si muove il robot; la fase di apertura della porta prevede invece un posizionamento più preciso della pinza del braccio robotico sulla maniglia. Per questo motivo, la stessa telecamera può essere utilizzata con una logica image-based, con set-point calcolato direttamente a partire dall'immagine che deve avere della maniglia per permettere alla pinza di portare a termine il compito prefissato.

Per concludere, l'ultima distinzione si riferisce a quali elementi della scena debbano essere inquadrati ed individuati dal dispositivo di visione. Si parla quindi di:

- *Sistemi End-Point Open-Loop (EOL)* se si osserva solo il movimento (all'interno dell'immagine) dell'oggetto target;
- *Sistemi End-Point Closed-Loop (ECL)* se si osserva il movimento sia del target che dell'end-effector.

Quest'ultima distinzione è strettamente connessa alla configurazione telecamera-manipolatore scelta. Infatti, la prima tipologia è tipica di un sistema eye-in-hand in cui il movimento dell'end-effector è legato al movimento dell'oggetto sul piano immagine. La seconda tecnica di controllo è invece necessaria quando la telecamera è stand-alone e si vuole movimentare

il robot senza fidarsi completamente del suo controllore, ad esempio per la correzione di errori di posizionamento dovuti ad una modellazione incompleta del sistema meccanico.

### 3 Position-Based Visual Servoing

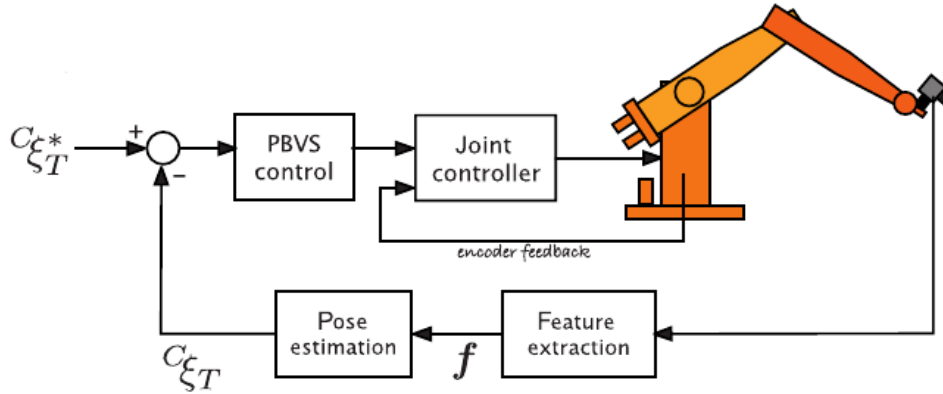
I metodi di asservimento visuale basati sulla stima della posa degli oggetti (*Position-Based Visual Servoing, PBVS*) prevedono per l'appunto l'utilizzo delle feature estratte dalle immagini per la stima della posizione relativa del target rispetto alla telecamera. In seguito, tramite la successiva elaborazione di questa informazione, si definisce un *segnale d'errore* tra la posizione attuale e quella desiderata. Poiché il sistema di visione produce un'informazione nello spazio geometrico e non nello spazio immagine, il riferimento viene definito all'interno dello *spazio di lavoro* del robot. Esso può essere, a seconda del numero di gradi di libertà del sistema, una semplice funzione distanza (1 gdl) fino ad arrivare ad una completa trasformazione geometrica (6 gdl). La caratteristica fondamentale del segnale di riferimento è però data dal fatto che esso non è assoluto, ma *relativo*. Infatti, non viene definita una posizione assoluta nello spazio di lavoro del manipolatore, ma la posa relativa tra il suo end-effector e l'oggetto da manipolare. La definizione del set-point viene fatta completamente offline senza avere a disposizione alcuna immagine acquisita dal sistema di visione. Ad esempio, è possibile definire la trasformazione geometrica che permette di descrivere la posizione dell'end-effector nel sistema di riferimento dell'oggetto da manipolare, definendone così la posa relativa. Il sistema di riferimento scelto per la rappresentazione dell'errore coincide generalmente col sistema di riferimento della telecamera, in modo da non dover sottoporre le misure ad un'ulteriore trasformazione geometrica.

Il task di posizionamento risulta rappresentato da una funzione  $\mathbf{E} : \mathcal{T} \rightarrow \mathcal{R}^m$ . Il compito è completato quando l'end-effector si trova in posizione  $\mathbf{x}_e$ , ovvero se l'errore cinematico è nullo:  $\mathbf{E}(\mathbf{x}_e) = 0$ . Se si considera una posizione generale  $\mathbf{x}_e$  per la quale il task è considerato completato, la funzione d'errore vincola un certo numero  $d$  degli  $m$  gradi di libertà del manipolatore ( $d \leq m$ ). La funzione d'errore cinematico, pertanto, rappresenta un vincolo cinematico virtuale tra l'end-effector ed il target.

Una volta definita in modo appropriato la funzione d'errore cinematico e individuate le feature all'interno di un'immagine si può definire un regolatore in grado di portare a 0 (zero) il valore stimato della funzione errore.

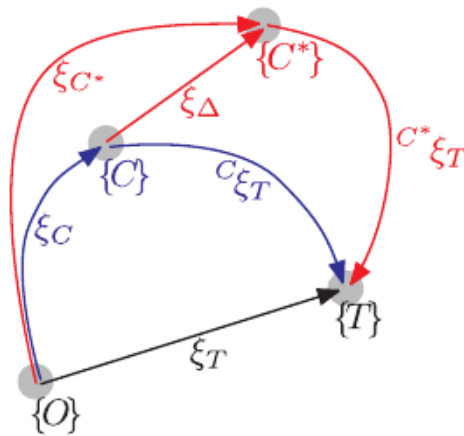
### 3.1 Calcolo della funzione errore

In figura 5.17 è mostrato lo schema a blocchi del controllo in asservimento visivo basato sulla posizione. È possibile notare che il compito del sistema di visione è duplice: l'estrazione delle feature e la stima della posa dell'oggetto nel suo sistema di riferimento.



**Figura 5.17:** Schema del controllo PBVS

Si consideri un sistema di asservimento visivo eye-in-hand. In figura 5.18 sono evidenziate le trasformazioni geometriche da considerare.



**Figura 5.18:** Trasformazioni geometriche coinvolte nel controllo PBVS

Lo scopo del sistema di visione è il calcolo della posa  ${}^C\xi_T$  del target rispetto all'obiettivo. La stima della posa è stata già discussa nei capitoli 3 e 4 e richiede tipicamente la conoscenza di (i) un modello dell'oggetto target, dei (ii) parametri intrinseci ed estrinseci della telecamera e delle (iii) caratteristiche salienti dell'immagine per l'estrazione delle feature. Il segnale di riferimento viene definito in funzione della posa desiderata della telecamera rispetto all'oggetto target  ${}^{C^*}\xi_T$ . L'obiettivo del controllo è la determinazione del movimento dell'end-effector richiesto per spostare la telecamera dalla sua posa di partenza  $\xi_C$

alla posa  $\xi_C^*$ , la cui differenza  $\xi_\Delta$  costituisce l'errore tra riferimento e segnale misurato. La relazione tra le due pose della telecamera è data da:

$$\begin{aligned}\xi_C^* &= \xi_C \cdot \xi_\Delta \\ \xi_\Delta &= (\xi_C)^{-1} \cdot \xi_C^*\end{aligned}\quad (5.14)$$

Nel caso d'esempio trattato, tuttavia, la distanza d'interesse per il controllo del robot è quella tra le pose di partenza ( $\xi_E$ ) e desiderata ( $\xi_E^*$ ) dell'end-effector e non della telecamera. Procedendo in modo analogo a quanto fatto per la posa della telecamera si ottiene la distanza  $\xi_{\Delta E}$ , la cui relazione con le due pose è espressa come

$$\begin{aligned}\xi_E^* &= \xi_E \cdot \xi_{\Delta E} \\ \xi_{\Delta E} &= (\xi_E)^{-1} \cdot \xi_E^*\end{aligned}\quad (5.15)$$

La trasformazione geometrica  ${}^C\xi_E$  che permette di passare dal sistema di riferimento dell'end-effector al sistema di riferimento della telecamera è costante e valida sia per la posa di partenza che per quella desiderata. La trasformazione è esplicitata da:

$$\begin{aligned}\xi_E &= {}^C\xi_E \cdot \xi_C \\ \xi_E^* &= {}^C\xi_E \cdot \xi_C^*\end{aligned}\quad (5.16)$$

Lo scopo del sistema di visione è quello di relazionare la posa della telecamera  $\xi_C$  alla posa dell'oggetto target  $\xi_T$ . Nel task di posizionamento la posa dell'oggetto rispetto al sistema di riferimento del robot è fissa perché l'obiettivo del controllo è quello di spostare l'end-effector in una certa posizione relativa all'oggetto stesso. La trattazione di un eventuale movimento dell'oggetto non è diversa, poiché esso verrebbe trattato dal controllo come un disturbo esterno da compensare. Il sistema di visione permette di stimare la posa dell'oggetto rispetto al sistema di riferimento della telecamera

$${}^T\xi_C = ({}^C\xi_T)^{-1}.$$

Eguagliando le pose oggetto espresse in funzione della sua posa rispetto alla telecamera e della posa della telecamera stessa in corrispondenza delle posizioni di partenza e desiderata, si ottiene la relazione

$$\begin{aligned}\xi_T &= \xi_C \cdot {}^C\xi_T = \xi_C^* \cdot {}^C\xi_T^* \\ \xi_C^* &= \xi_C \cdot {}^C\xi_T \cdot ({}^C\xi_T^*)^{-1}\end{aligned}\quad (5.17)$$

Combinando le equazioni (5.15) e (5.17) si ottiene l'espressione della distanza  $\xi_{\Delta\mathbf{E}}$ , come mostrato nella (5.18).

$$\begin{aligned}
 \xi_{\Delta\mathbf{E}} &= (\xi_{\mathbf{E}})^{-1} \cdot \xi_{\mathbf{E}}^* \\
 &= (\xi_{\mathbf{E}})^{-1} \cdot ({}^{\mathbf{C}}\xi_{\mathbf{E}} \cdot \xi_{\mathbf{C}}^*) \\
 &= (\xi_{\mathbf{E}})^{-1} \cdot ({}^{\mathbf{C}}\xi_{\mathbf{E}} \cdot (\xi_{\mathbf{C}} \cdot {}^{\mathbf{C}}\xi_{\mathbf{T}} \cdot ({}^{\mathbf{C}}\xi_{\mathbf{T}}^*)^{-1})) \\
 &= (\xi_{\mathbf{E}})^{-1} \cdot ({}^{\mathbf{C}}\xi_{\mathbf{E}} \cdot (\xi_{\mathbf{C}} \cdot \xi_{\Delta}))
 \end{aligned} \tag{5.18}$$

Si noti che il calcolo dell'errore dipende da vari fattori:

- Posizione di partenza dell'end-effector ( $\xi_{\mathbf{E}}$ ). Questa viene stimata tramite la cinematica del manipolatore;
- Posizione di partenza dalla telecamera ( $\xi_{\mathbf{C}}$ ), stimata a partire dalla posizione dell'end-effector e tramite il successivo cambio di coordinate rigido tra end-effector e telecamera;
- Trasformazione geometrica per il cambio di coordinate tra sistema di riferimento della telecamera e riferimento dell'end-effector ( ${}^{\mathbf{C}}\xi_{\mathbf{E}}$ );
- Stima della posa oggetto nel sistema di riferimento della telecamera nella posa di partenza ( ${}^{\mathbf{C}}\xi_{\mathbf{T}}$ ). In questo fattore entrano gli eventuali errori di stima da parte del sistema di visione. È per ottenere questa stima che diventa necessario conoscere i parametri intrinseci ed estrinseci della telecamera, ovvero all'esecuzione della procedura di calibrazione;
- Metodo di indicazione della posa desiderata ( ${}^{\mathbf{C}}\xi_{\mathbf{T}}^*$ ). Se questa viene calcolata direttamente a partire dalle immagini, l'eventuale errore di stima è lo stesso compiuto nella fase di stima della posa online. Se, invece, la posa viene definita senza l'utilizzo delle immagini, l'errore dipende dalla precisione del modello del sistema e del target.

L'analisi di questi punti mette in evidenza come l'utilizzo dell'errore nelle tecniche PBVS permette di correggere eventuali errori di stima o posizionamento, ma il calcolo dell'errore stesso non è immune dall'introduzione di errori stocastici (i) da parte del controllore del manipolatore per quanto riguarda l'analisi della cinematica e (ii) da parte del sistema di visione per quanto concerne l'individuazione delle feature, la stima della loro posizione nello spazio e, nel complesso, la stima della posa 3D dell'oggetto target.

Il caso banale, in cui la telecamera viene fatta coincidere con l'end-effector ( ${}^{\mathbf{T}}\xi_{\mathbf{C}} = \mathbf{I}$ ) si

riduce all'equazione

$$\begin{aligned}\xi_{\Delta E} &= (\xi_E)^{-1} \cdot ({}^C\xi_E \cdot (\xi_C \cdot \xi_\Delta)) \\ &= (\xi_E)^{-1} \cdot (\mathbf{I} \cdot (\xi_E \cdot \xi_\Delta)) \\ &= \xi_\Delta\end{aligned}\tag{5.19}$$

Il caso del controllo eye-out-hand con sistema di visione in grado di tracciare il movimento sia degli oggetti target che dell'end-effector viene trattato in modo simile a quanto visto per il caso eye-in-hand. Tuttavia, la possibilità di stimare la posa dell'end-effector permette di rimuovere dal problema gli eventuali errori di posizionamento dovuti ad approssimazioni nella scrittura delle equazioni cinematiche del sistema. In questo caso, infatti, gli unici errori introdotti diverrebbero quelli dovuti al sistema di visione per la stima di entrambe le pose.

### 3.2 Legge di controllo

Il calcolo della distanza (o errore) tra la posa della telecamera attuale e quella desiderata è il primo passo che il controllore in asservimento visuale deve compiere. Il secondo passo prevede la generazione di un segnale tramite il quale è possibile movimentare i giunti del manipolatore per l'esecuzione del task di posizionamento. Come si è visto nel paragrafo 1 e si vedrà più in dettaglio nel paragrafo 1 del capitolo 7, la scelta si riduce alle seguenti possibilità:

- Controllo in *coppia*;
- Controllo in *velocità*;
- Controllo in *posizione*.

Tramite il controllo in coppia è generalmente difficile pilotare un manipolatore complesso, perché esso prevede di fare affidamento sul calcolo della dinamica per stimare le coppie da imporre ai giunti per l'esecuzione di un certo movimento. Il controllo in posizione viene eseguito chiudendo due anelli di controllo successivi (velocità e posizione) attorno all'anello di controllo della coppia. Come si è già visto, questo approccio è quello seguito nel controllo in anello aperto. Per il controllo in asservimento visivo, invece, è il controllo in *velocità* quello più utilizzato. Infatti, scopo ultimo del visual servoing è quello di fornire una modalità per aggiornare continuamente il comando agli attuatori dei giunti durante l'esecuzione della traiettoria. L'aggiornamento incrementale del comando fa pensare

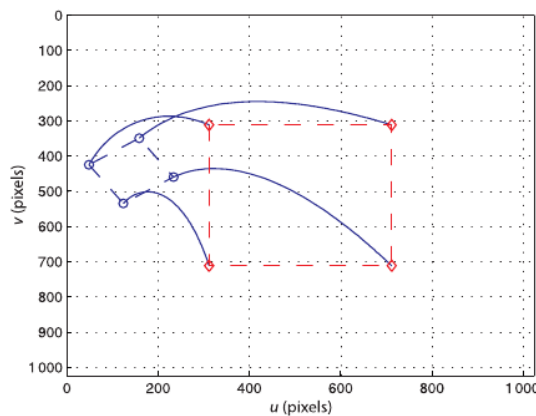


appunto alla definizione di velocità come derivata dello spostamento. Pertanto l'errore di posizionamento viene elaborato dal controllore in asservimento visivo per generare il *vettore della velocità*  $\mathbf{u}$  istantanea dell'end-effector nello spazio di lavoro. Il controllore del robot visto in precedenza viene modificato per cercare di seguire ad ogni istante il profilo di velocità dato dal controllore; la posizione e accelerazione istantanee dei giunti vengono utilizzate per il calcolo dello jacobiano necessario allo sviluppo della cinematica inversa. Detto  $\mathbf{x}_\Delta = [\Delta x, \Delta y, \Delta z, \Delta(\theta_x), \Delta(\theta_y), \Delta(\theta_z)]^T$  il vettore composto dalle traslazioni e gli angoli di rotazione attorno ai tre assi principali del sistema di riferimento generati dalla trasformazione  $\xi_{\Delta\mathbf{E}}$ , la legge di controllo PBVS è sintetizzata nell'equazione

$$\mathbf{u} = \mathbf{K} \cdot \mathbf{x}_\Delta . \quad (5.20)$$

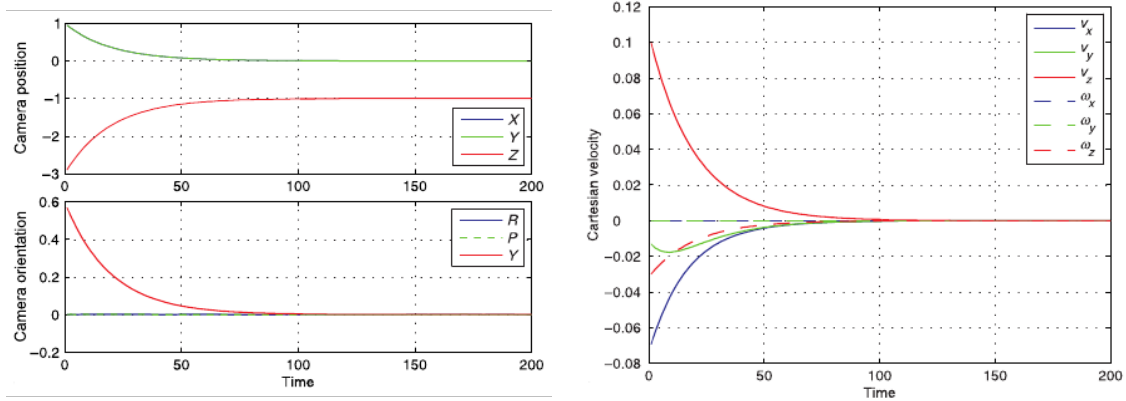
La matrice  $\mathbf{K}$  è una matrice diagonale i cui elementi della diagonale principale sono i controllori per ogni grado di libertà del sistema. I controllori possono essere tutti identici tra loro, oppure diversi a seconda della dinamica desiderata per ogni grado di libertà. La sintesi del controllo viene presentata nel capitolo 7 in cui si vedrà che un controllore proporzionale è spesso sufficiente a garantire la stabilità del sistema.

In figura 5.19 è riportato l'esempio di un task di posizionamento con controllo PBVS di cui vengono mostrate la posizione di partenza delle feature individuate e la posa desiderata.



**Figura 5.19:** Esempio PBVS: traiettoria nello spazio immagine

In figura 5.20 è riportata la velocità nello spazio di lavoro del robot da imporre per l'ottenimento della traiettoria desiderata e lo spostamento ottenuto per i diversi gradi di libertà del manipolatore.



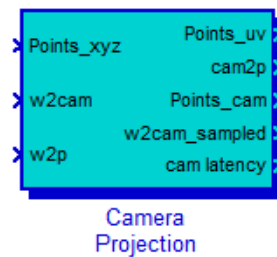
(a) Posizione e orientamento della telecamera.

(b) Velocità della telecamera nello spazio di lavoro.

**Figura 5.20:** Esempio di PBVS: traiettoria nello spazio di lavoro

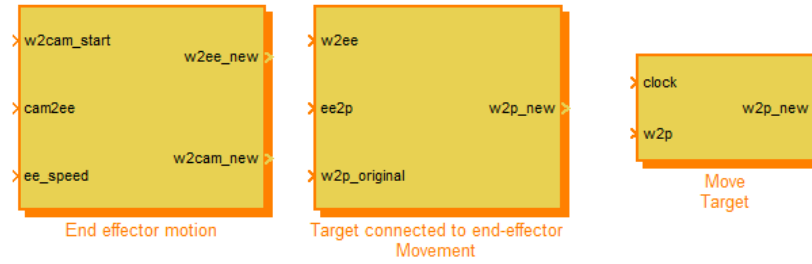
### 3.3 Schema Simulink

La realizzazione di uno schema Simulink per la simulazione di un controllore PBVS deve passare necessariamente dalla modellazione della trasformazione prospettica della telecamera e dalla simulazione del movimento dell'end-effector. In figura 5.21 è mostrato il blocco utilizzato per la modellazione della trasformazione prospettica dovuta alla telecamera: dato un vettore di feature dell'oggetto target, la loro posizione nel sistema di riferimento manipolatore e la posa della telecamera, si ottengono le coordinate in pixel assunte dalle feature sul piano immagine.

**Figura 5.21:** Blocco Simulink per la simulazione della telecamera

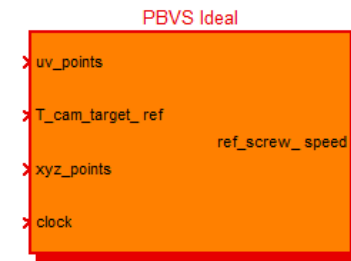
I blocchi in figura 5.22 permettono di spostare l'end-effector nello spazio di lavoro del robot. In particolare, il primo blocco calcola la posa dell'end-effector a partire dalla posa della telecamera, il secondo produce un movimento virtuale della scena opposto al movimento dell'end-effector, mentre il terzo permette di aggiungere una legge di moto desiderata all'oggetto.

In figura 5.23 è riportato lo schema a blocchi del controllore PBVS implementato e simu-

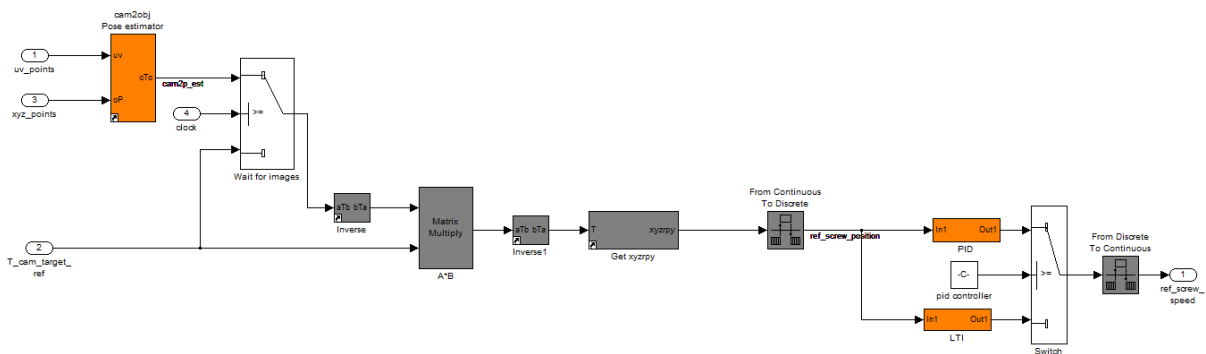


**Figura 5.22:** Blocco Simulink per la simulazione del movimento dell'end-effector

lato. Gli ingressi principali del blocco sono le coordinate in pixel delle feature individuate nell'immagine e la posa della telecamera desiderata. In uscita viene generato il vettore velocità da fornire al controllore del robot. Il blocco `cam2obj PoseEstimator` si occupa di stimare la posa della telecamera rispetto all'oggetto target individuato; la trasformazione geometrica ottenuta viene in seguito confrontata con la posa desiderata per ottenere l'errore di posizionamento `xyzrpy`; l'errore è il segnale di ingresso per il blocco regolatore che può essere costituito da un controllore PID o da un regolatore appositamente parametrizzato.



(a) Blocco del controllo PBVS.



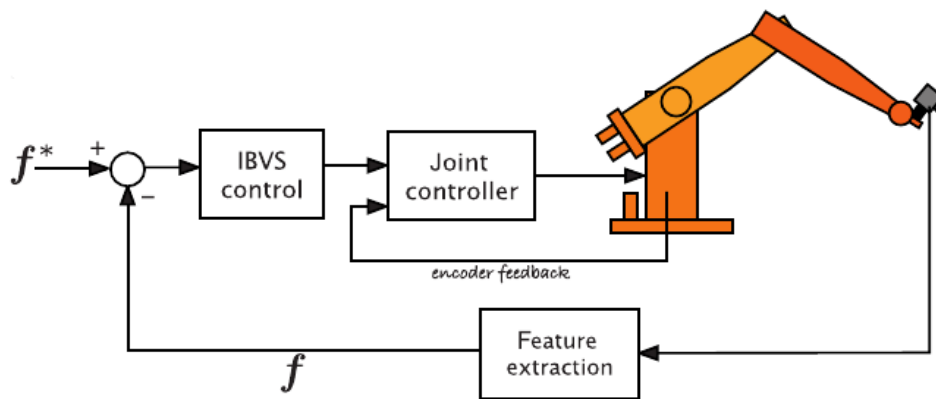
(b) Schema a blocchi del controllo.

**Figura 5.23:** Schema a blocchi Simulink del controllo PBVS

## 4 Image-Based Visual Servoing

Il controllo in asservimento visivo basato su immagini (*Image Based Visual Servoing*, *IBVS*) permette di elaborare la legge di controllo direttamente a partire dalle feature estratte dalle immagini, senza bisogno di conoscere la posizione dell'end-effector del robot e i parametri di calibrazione della telecamera.

Un task di asservimento visuale basato su immagini è infatti rappresentato da una funzione errore dell'immagine  $e : \mathcal{F} \rightarrow R^l$ , dove  $l \leq k$ , con  $k$  dimensione dello spazio dei parametri immagine. Questa definizione implica l'eliminazione della fase di stima della posa dell'oggetto target, come mostrato nello schema di figura 5.24.

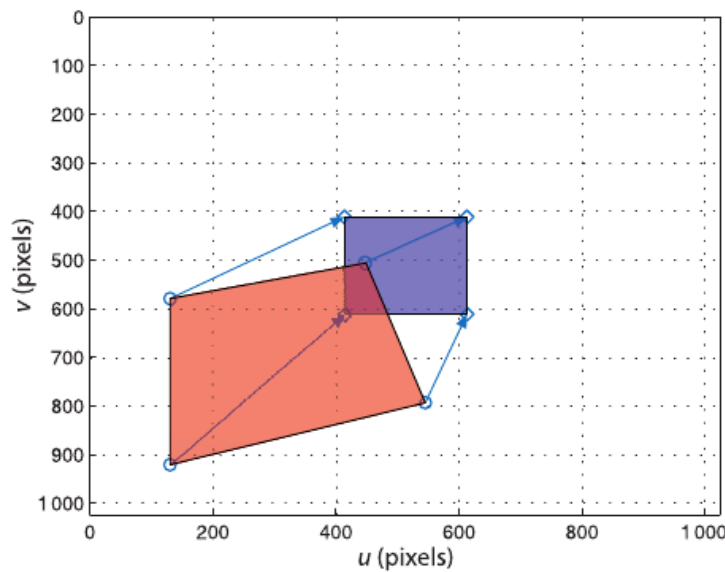


**Figura 5.24:** Schema del controllo in asservimento visivo Image-Based

La telecamera può essere in configurazione stand-alone o eye-in-hand. In entrambi i casi il movimento del manipolatore provoca una variazione nella posizione dell'oggetto target all'interno dell'immagine. Il primo passo da compiere è, come nel caso del controllo PBVS, la scelta della funzione errore, la quale deve avere un legame diretto con la variazione della scena inquadrata dalla telecamera. In particolare, rispetto al controllo basato su stima della posizione, cambia leggermente l'approccio alla scelta della posa desiderata. Infatti, se nel controllo PBVS era semplice rappresentare la posa in termini delle coordinate 3D del sistema di riferimento, con l'approccio IBVS si perde completamente il legame con la posa relativa telecamera-oggetto. Pertanto, il metodo più utilizzato per identificare la posizione target è il cosiddetto *teach by showing*, in cui l'end-effector viene spostato nella posizione desiderata e l'immagine corrispondente è utilizzata per il calcolo del vettore delle feature immagine desiderate  $\mathbf{f}_d$

Nel caso di sistema eye-in-hand, l'errore è definito in funzione della distanza, sul piano immagine, tra le feature nella posa corrente e le stesse feature nella posa desiderata. Il dover

considerare le stesse feature implica l'introduzione di qualche meccanismo per il calcolo delle corrispondenze e il loro successivo tracciamento all'interno dell'immagine. Alcune delle tecniche utilizzabili vengono trattate nel capitolo 6. Al contrario, nella configurazione eye-out-hand, l'errore può essere definito in funzione della distanza, sempre sul piano immagine, tra l'end-effector del manipolatore e il target. In figura 5.25 viene mostrato un esempio di definizione del task di posizionamento IBVS nello spazio immagine. In rosso è evidenziata la posizione delle feature di partenza, mentre quella rappresentata in blu è la posizione desiderata. Le frecce che collegano le feature nelle due pose rappresentano le corrispondenze tra i due vettori di feature.



**Figura 5.25:** Definizione del task di posizionamento IBVS nel piano immagine

Ad ogni modo, nonostante l'errore  $e$  venga definito nello spazio dei parametri dell'immagine, l'input al controllore del manipolatore deve essere definito nello spazio dei giunti o di lavoro del manipolatore, ad esempio sotto forma di vettore velocità come nel caso del controllo PBVS. È pertanto necessaria una modalità che permetta di effettuare il cambio di coordinate in modo trasparente, senza passare dalla stima della posa degli oggetti. Tale modalità è fornita dallo *Jacobiano immagine*.

## 4.1 Jacobiano immagine

Sia  $\mathbf{r}$  la rappresentazione delle coordinate dell'end-effector nello spazio di lavoro  $\mathcal{T}$ ;  $\dot{\mathbf{r}}$  rappresenta la corrispondente velocità dell'end-effector. Come si è già visto,  $\dot{\mathbf{r}}$  è noto come *vettore delle velocità* dell'end-effector nel suo spazio di lavoro e, nel tipico caso di

manipolatori a 6 gradi di libertà, è composto da una terna di velocità traslazionali e una terna di velocità rotazionali:

$$\dot{\mathbf{r}} = \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} . \quad (5.21)$$

Sia  $\mathbf{f}$  il vettore delle feature delle immagini, espresse, ad esempio, in funzione delle loro coordinate in pixel;  $\dot{\mathbf{f}}$  è quindi il corrispondente vettore delle variazioni della posizione delle feature oggetto nelle varie immagini. Il vettore  $\dot{\mathbf{f}}$  fornisce il significato profondo del controllo IBVS: un movimento relativo tra telecamera e scena produce un movimento all'interno delle immagini acquisite dal sistema di visione. Tale movimento è rappresentabile con un *vettore velocità nello spazio immagine*.

Lo *Jacobiano immagine*  $\mathbf{J}_v$  è quindi la trasformazione lineare che permette di passare dallo spazio di lavoro  $\mathcal{T}$  allo spazio immagine  $\mathcal{F}$ , mettendone in relazione le velocità, come espresso nella (5.22). Si può notare l'analogia con la definizione di matrice jacobiana per lo sviluppo della cinematica del manipolatore (vedi 1) in cui lo jacobiano permette di passare dalla velocità nello spazio di lavoro del robot alla velocità nello spazio dei giunti.

$$\dot{\mathbf{f}} = \mathbf{J}_v(\mathbf{r})\dot{\mathbf{r}} . \quad (5.22)$$

Lo jacobiano  $\mathbf{J}_v \in \mathcal{R}^{k \times m}$  è calcolato, come da definizione:

$$J_v(\mathbf{r}) = \left[ \frac{\partial \mathbf{f}}{\partial \mathbf{r}} \right] = \begin{bmatrix} \frac{\partial f_1(\mathbf{r})}{\partial r_1} & \cdots & \frac{\partial f_1(\mathbf{r})}{\partial r_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_k(\mathbf{r})}{\partial r_1} & \cdots & \frac{\partial f_k(\mathbf{r})}{\partial r_m} \end{bmatrix} . \quad (5.23)$$

Il numero di righe dello jacobiano dipende dalla dimensione dello spazio dei parametri dell'immagine, ovvero dal numero di feature considerate, mentre il numero di colonne è pari a  $m$ , cioè la dimensione dello spazio di lavoro del manipolatore, data dai suoi gradi di libertà.

Per il calcolo dello jacobiano immagine possono essere utilizzate diversi tipi di feature, quali (i) punti, (ii) linee e (iii) cerchi. Nel proseguo della trattazione viene considerato solo il caso di feature di tipo puntuale.

### 4.1.1 Calcolo dello jacobiano dato un pixel

Come si è visto, il numero di righe dello jacobiano immagine dipende dal numero di feature individuate e considerate all'interno dell'immagine. Più in dettaglio, il numero di righe viene influenzato anche dal numero di gradi di libertà posseduto da ogni feature. Un punto possiede 2 gradi di libertà, poiché all'interno del piano immagine esso è individuato e descritto da una coppia di coordinate  $(r, c)$  (riga e colonna del piano immagine). Pertanto, data una singola feature puntuale, vengono definite 2 righe della matrice jacobiana.

Considerando un manipolatore a 6 gradi di libertà, la velocità dell'end-effector risulta espressa da una componente rotazionale  $\mathbf{\Omega}_e = [\omega_x \ \omega_y \ \omega_z]$  e una traslazionale  $\mathbf{T}_e = [T_x \ T_y \ T_z]$ . Se è nota la trasformazione rigida  $\mathbf{P}$  che permette il cambio di coordinate dal sistema di riferimento dell'end-effector a quello della telecamera, l'equazione

$$\dot{\mathbf{P}}_c = \mathbf{\Omega}_e \times \mathbf{P} + \mathbf{T}_e . \quad (5.24)$$

fornisce la velocità traslazionale  $\dot{\mathbf{P}}_c = [\dot{x}, \dot{y}, \dot{z}]^T$  della telecamera.

Le coordinate  $(x, y, z)$  sono espresse nel sistema di riferimento della telecamera. Esplicitando le tre velocità dell'equazione (5.24) si ottiene il seguente sistema:

$$\begin{cases} \dot{x} = z\omega_y - y\omega_z + T_x \\ \dot{y} = x\omega_z - z\omega_x + T_y \\ \dot{z} = y\omega_x - x\omega_y + T_z \end{cases} . \quad (5.25)$$

La trasformazione prospettica della telecamera (vedi equazione (2.5)) permette di esprimere il sistema (5.25) direttamente in termini delle coordinate del piano immagine piuttosto che delle coordinate spaziali nel sistema di riferimento della telecamera. Il sistema ottenuto è quindi:

$$\begin{cases} \dot{x} = z\omega_y - \left[ (v - c_y) \cdot \frac{z}{f_y} \right] \omega_z + T_x \\ \dot{y} = \left[ (u - c_x) \cdot \frac{z}{f_x} \right] \omega_z - z\omega_x + T_y \\ \dot{z} = z \left[ \frac{v - c_y}{f_y} \omega_x - \frac{u - c_x}{f_x} \omega_y \right] + T_z \end{cases} . \quad (5.26)$$

Poiché lo scopo di questa trattazione è l'ottenimento di una relazione che permetta di esprimere la velocità dei singoli pixel nello spazio immagine, è possibile procedere alla derivazione delle coordinate dei singoli pixel. L'equazione

$$\dot{u} = \frac{d}{dt} \left( f_x \cdot \frac{X}{Z} + c_x \right) = f_x \frac{\dot{x}z - x\dot{z}}{z^2} \quad (5.27)$$

mostra il risultato della derivazione per la coordinata  $u$ .

Sostituendo nella (5.27) le relazioni (5.26), si ottiene la prima riga dello jacobiano immagine, esplicitata da:

$$\begin{aligned}
 \dot{u} &= \frac{f_x}{z^2} \{ z^2 \omega_y - z[(v - c_y) \frac{z}{f_y}] \omega_z + zT_x - zx[\frac{v - c_y}{f_y} \omega_x - \frac{u - c_x}{f_x} \omega_y] - xT_z \} \\
 &= \frac{f_x}{z} T_x + 0 \cdot T_y - \frac{xf_x}{z^2} T_z - \frac{f_x}{f_y} \cdot \frac{x(v - c_y)}{z} \omega_x + (f_x + \frac{u - c_x}{z} x) \omega_y - \frac{f_x}{f_y} (v - c_y) \omega_z \\
 &= \frac{f_x}{z} T_x + 0 \cdot T_y - \frac{u - c_x}{z} T_z \\
 &\quad - \frac{(u - c_x)(v - c_y)}{f_y} \omega_x + \frac{f_x^2 + (u - c_x)^2}{f_x} \omega_y - \frac{f_x}{f_y} (v - c_y) \omega_z
 \end{aligned} \tag{5.28}$$

Infine, ripetendo lo stesso ragionamento anche per la coordinata  $v$  e riorganizzando il tutto in forma matriciale si ottiene la matrice jacobiana di un singolo pixel:

$$\mathbf{J}_v = \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{u - c_x}{z} & -\frac{(u - c_x)(v - c_y)}{f_y} & \frac{f_x^2 + (u - c_x)^2}{f_x} & -\frac{f_x}{f_y} (v - c_y) \\ 0 & \frac{f_y}{z} & -\frac{v - c_y}{z} & -\frac{f_y^2 + (v - c_y)^2}{f_y} & \frac{(u - c_x)(v - c_y)}{f_x} & \frac{f_y}{f_x} (u - c_x) \end{bmatrix}. \tag{5.29}$$

Si può notare come nello jacobiano immagine siano ancora presenti i termini relativi ai parametri di calibrazione della telecamera e l'informazione relativa alla distanza  $z$  (profondità) del punto rispetto alla telecamera nel sistema di riferimento della telecamera stessa. In seguito viene messo in luce come la chiusura dell'anello di retroazione del controllo IBVS, cercando di azzerare l'errore di posizionamento delle feature all'interno dell'immagine, renda il sistema meno sensibile ad eventuali errori nella stima di tali parametri.

Per estendere il risultato (5.29) al caso in cui lo spazio delle feature sia composto da più di un pixel, è sufficiente procedere per *stacking*, impilando cioè più matrici jacobiane del singolo punto, come mostrato nella (5.30). In caso il sistema abbia meno di 6 gradi di libertà, si selezionano solo le colonne interessate.

$$\mathbf{J}_v(\mathbf{f}) = \begin{bmatrix} \mathbf{J}_v(f_1) \\ \mathbf{J}_v(f_2) \\ \vdots \\ \mathbf{J}_v(f_n) \end{bmatrix}. \tag{5.30}$$



### 4.1.2 Jacobiano inverso

I risultati ottenuti nel paragrafo precedente mostrano come mettere in relazione il movimento dell'end-effector con il moto percepito dalla telecamera (nel senso di spostamento dell'oggetto target nella successione di immagini). Tuttavia, come nel caso della cinematica dei manipolatori, l'asservimento visuale necessita della relazione opposta, ovvero il calcolo delle velocità da assegnare all'end-effector in modo da osservare un certo spostamento del target nella successione di immagini. Si possono considerare due casi a seconda delle dimensioni dello jacobiano:

- *Jacobiano quadrato,  $k = m$ .*

Se  $k = m$  e  $\mathbf{J}_v$  non è singolare, allora la matrice inversa  $\mathbf{J}_v^{-1}$  risulta definita e quindi la cinematica inversa dello spostamento dell'immagine è anch'essa definita:

$$\dot{\mathbf{r}} = \mathbf{J}_v^{-1} \dot{\mathbf{u}}. \quad (5.31)$$

In pratica, questa è la situazione in cui si osserva un numero appena sufficiente di caratteristiche per il calcolo del movimento. Ad esempio, se il sistema ha due gradi di libertà traslazionali è sufficiente un solo punto per il calcolo dell'errore. Tuttavia, quando si hanno movimenti più complessi che, ad esempio, prendono in considerazione anche rotazioni e/o avvicinamento all'oggetto target, si corre il rischio che lo jacobiano diventi singolare rendendo così impossibile la sua inversione. In questo caso bisogna quindi studiare quali sono i possibili punti di singolarità dello jacobiano immagine e prevedere che il controllore riesca a modificare le traiettorie in modo da limitare l'insorgenza di tali singolarità.

- *Jacobiano non quadrato,  $k \neq m$ .*

In questo caso la matrice inversa  $\mathbf{J}_v^{-1}$  non esiste e quindi la soluzione è data da un'approssimazione ai minimi quadrati che, in generale, è ottenuta tramite matrice pseudoinversa:

$$\dot{\mathbf{r}} = \mathbf{J}_v^+ \cdot \dot{\mathbf{f}} + (\mathbf{I} - \mathbf{J}_v^+ \mathbf{J}_v) \mathbf{b} \quad (5.32)$$

dove la matrice  $\mathbf{J}_v^+$  è la matrice *pseudoinversa* di  $\mathbf{J}_v$  e  $\mathbf{b}$  è un vettore arbitrario di dimensioni appropriate. La soluzione ai minimi quadrati fornisce un valore di  $\dot{\mathbf{r}}$  che minimizza la norma  $\|\dot{\mathbf{f}} - \mathbf{J}_v \dot{\mathbf{r}}\|$ . Questa soluzione è possibile solo se  $\mathbf{J}_v$  ha rango pieno, cioè  $\text{rank}(\mathbf{J}_v) = \min(k, m)$ .

In base al numero di feature osservate, si possono presentare due diversi sottocasi:

\*  $k > m$ .

Se si osserva un numero di feature maggiore del minimo necessario si hanno  $k - m$  feature *ridondanti*. Si ottiene un sistema di equazioni inconsistenti e la pseudoinversa è definita come

$$\mathbf{J}_v^+ = (\mathbf{J}_v^T \mathbf{J}_v)^{-1} \mathbf{J}_v^T . \quad (5.33)$$

Si ha che  $(\mathbf{I} - \mathbf{J}_v^+ \mathbf{J}_v) = 0$ , cioè il rango del kernel di  $\mathbf{J}_v$  è 0 (dal momento che  $m$  è il rango di  $\mathbf{J}_v$ ). Di conseguenza, la soluzione data dalla (5.32) si semplifica ulteriormente:

$$\dot{\mathbf{r}} = \mathbf{J}_v^+ \cdot \dot{\mathbf{f}} . \quad (5.34)$$

\*  $k < m$ .

Il sistema risulta sottovincolato; nell'asservimento visuale questo significa che non si stanno osservando abbastanza feature in modo da poter determinare un'unica soluzione convergente. La pseudoinversa può essere calcolata come:

$$\mathbf{J}_v^+ = \mathbf{J}_v^T (\mathbf{J}_v \mathbf{J}_v^T)^{-1} . \quad (5.35)$$

In generale, per  $k < m$ ,  $(\mathbf{I} - \mathbf{J}_v^+ \mathbf{J}_v \neq 0)$  e tutti i vettori della forma  $(\mathbf{I} - \mathbf{J}_v^+ \mathbf{J}_v) \mathbf{b}$  giacciono nello spazio nullo di  $\mathbf{J}_v$  e corrispondono a quelle componenti delle velocità dell'oggetto non osservabili. In questo caso, quindi, la soluzione è data ancora una volta dalla 5.32.

## 4.2 Generazione del controllo

Per lo sviluppo del controllore si può procedere allo stesso modo di quanto fatto per il controllo PBVS, utilizzando cioè l'approccio denominato *resolved rate motion control*. Per il controllo IBVS, lo scopo del task di controllo è l'ottenimento di una particolare vista dell'oggetto target, rappresentata dallo spazio dei parametri  $\mathbf{f}_d$  (feature desiderate). Se il controllore del robot accetta in input direttamente le velocità dell'end-effector ( $\mathbf{u} = \dot{\mathbf{r}}$ ), il controllore IBVS è dato dalla (5.36), dove la matrice  $\mathbf{K}$  è ancora una volta la matrice diagonale dei regolatori per ogni grado di libertà del manipolatore.

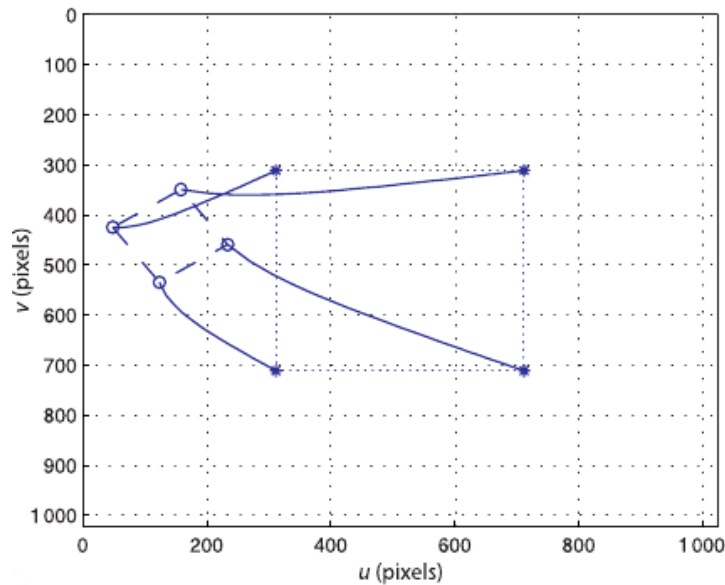
$$\mathbf{u} = \mathbf{K} \mathbf{J}_v^+ (\mathbf{r}) \mathbf{e}(\mathbf{f}) . \quad (5.36)$$

align L'equazione del controllo mette in evidenza come il controllore sia robusto nei confronti di errate stime della distanza  $Z$  utilizzata nel calcolo dello jacobiano. Infatti,

considerando regolatori proporzionali,  $Z$  ha un'influenza diretta sul valore del controllore proporzionale. Quindi, nel caso  $Z$  venisse sovrastimata, la dinamica del sistema ne risulterebbe rallentata, senza però influenzare la stabilità del sistema. Al contrario, i problemi potrebbero sorgere nel caso di  $Z$  sottostimata, perché il controllore potrebbe risultare instabile.

Allo stesso modo di quanto detto per il controllo PBVS,  $\mathbf{u}$  rappresenta il vettore velocità della telecamera. Pertanto, per il calcolo della velocità dell'end-effector del manipolatore è necessario considerare la trasformazione rigida di coordinate  ${}^C\xi_e$  al fine di riportare la velocità della telecamera alla velocità dell'end-effector che può essere fornita direttamente in ingresso al controllore del sistema meccanico.

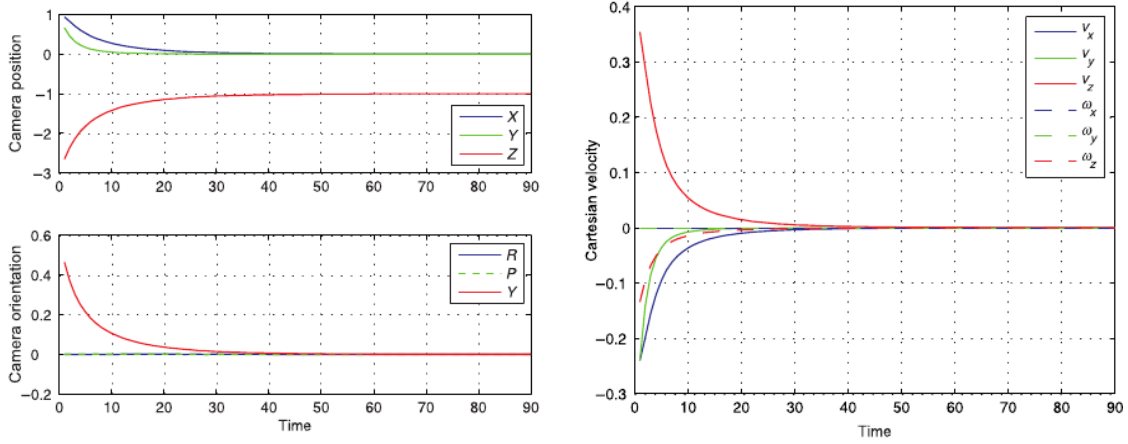
In figura 5.26 è mostrata la traiettoria nel piano immagine di un task di posizionamento IBVS nel caso la distanza  $Z$  venga stimata correttamente.



**Figura 5.26:** Esempio IBVS: traiettoria nello spazio immagine

Il vettore velocità della telecamera nello spazio di lavoro del manipolatore e lo spostamento provocato sono mostrati in figura 5.27.

Nel caso il sistema di elaborazione immagini non sia in grado di ottenere una stima della distanza  $Z$ , il sistema rimane comunque stabile qualora il valore rappresentante la distanza  $Z$  non vada ad intaccare il contributo proporzionale del regolatore andandone ad inficiarne la stabilità. In figura 5.28 vengono messe in evidenza le due situazioni tipiche: nel caso di  $Z$  sovrastimata il controllore rimane sicuramente stabile, ma con un calo delle prestazioni dinamiche del sistema; nel caso di  $Z$  sottostimata, invece, il sistema vede un



(a) Posizione e orientamento della telecamera. (b) Velocità della telecamera nello spazio di lavoro.

**Figura 5.27:** Esempio di IBVS: traiettoria nello spazio di lavoro

aumento delle prestazioni dinamiche, ma può avvicinarsi all'instabilità, allontanando le traiettorie calcolate dal caso lineare.

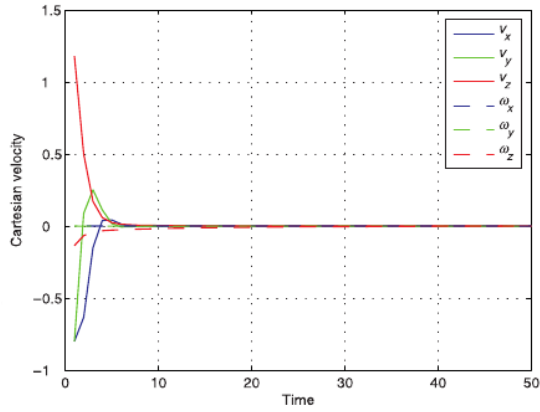
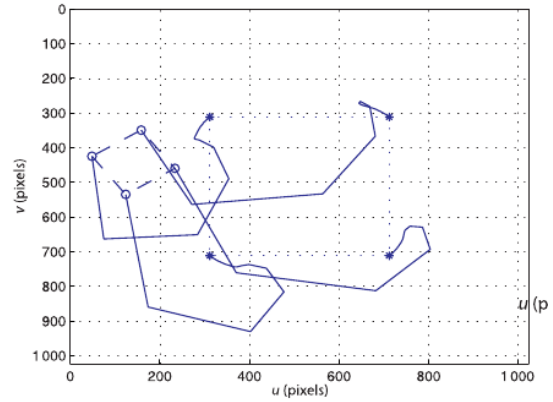
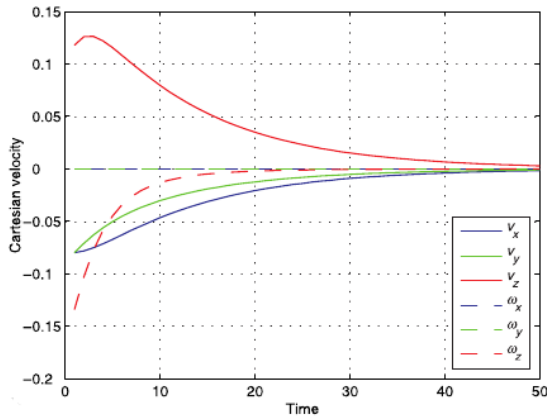
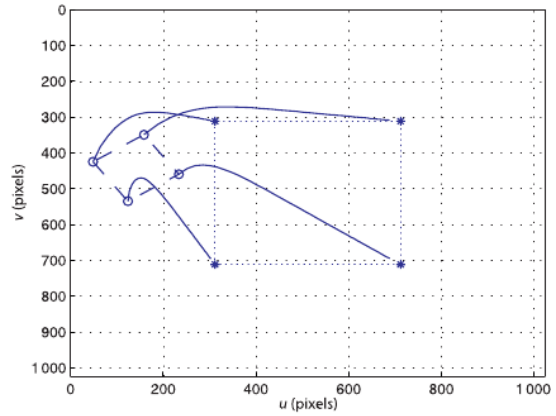
### 4.3 Punti di singolarità

Nel paragrafo 4.1.2 si sono analizzati i casi in cui è possibile l'inversione della matrice jacobiana, concludendo che è possibile trovare una soluzione ai minimi quadrati se lo jacobiano ha rango pieno. Se  $\mathbf{J}$  non ha rango pieno, la matrice  $(\mathbf{J}^T \mathbf{J})$  diventa singolare e non può quindi essere invertita. Di conseguenza non è più definita la matrice pseudoinversa. Le configurazioni in cui  $\mathbf{J}$  non ha rango pieno si chiamano *configurazioni singolari*. Tali situazioni non devono mai verificarsi durante l'asservimento poiché esse rendono impossibile la generazione dell'errore e, di conseguenza, del segnale di controllo in velocità del manipolatore.

Per analizzare i casi in cui possono verificarsi tali situazioni si può considerare lo jacobiano immagine semplificato della (5.37), ottenuto ponendo  $c_x = c_y = 0$  e  $f_x = f_y = f$ .

$$\mathbf{J} = \begin{bmatrix} \frac{f}{z} & 0 & -\frac{u}{z} & -\frac{uv}{f} & \frac{f^2 + u^2}{f} & -v \\ 0 & \frac{f}{z} & -\frac{v}{z} & -\frac{f^2 - v^2}{f} & \frac{uv}{f} & u \end{bmatrix}. \quad (5.37)$$

Si verifica una condizione di singolarità se:

(a) *Z sottostimata: traiettoria cartesiana.*(b) *Z sottostimata: traiettoria nello spazio immagine.*(c) *Z sovrastimata: traiettoria cartesiana.*(d) *Z sovrastimata: traiettoria nello spazio immagine.***Figura 5.28:** Esempio di IBVS: effetto della stima della distanza  $Z$ 

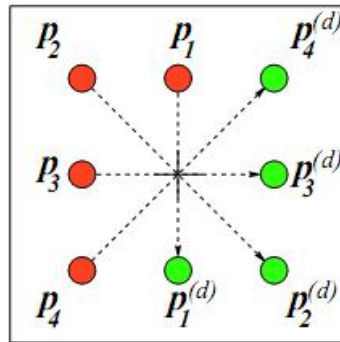
1. La telecamera si allontana troppo dal target, ovvero se:

$$\begin{cases} z \gg u \\ z \gg v \\ z \gg f \end{cases} .$$

Le prime tre colonne di  $J$  tendono a 0 e lo jacobiano non ha più rango pieno: l'oggetto tende a collassare in un punto e le traslazioni o rotazioni della telecamera non modificano praticamente la vista del target;

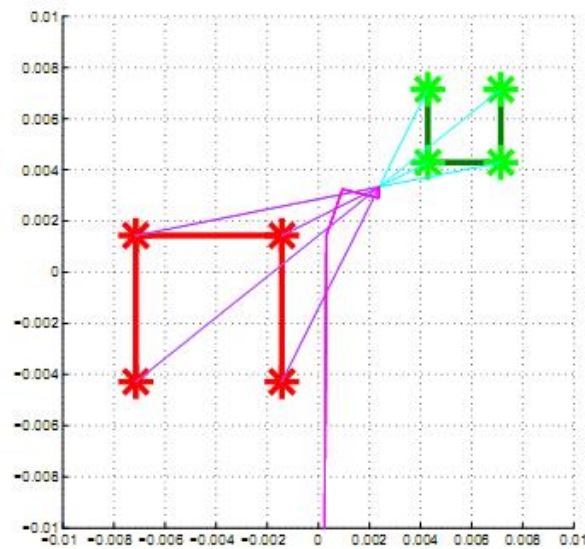
2. Si scelgono punti allineati (linearmente dipendenti). La legge di controllo proporzionale tende a passare per questa situazione perché la matrice pseudoinversa cerca di spingere ogni punto in linea retta verso la posizione desiderata trattando punti propri e impropri allo stesso modo.

Casi tipici di stallo del controllo per singolarità sono le *omotetie* con rotazione di  $180^\circ$ , mostrata in figura 5.29.



**Figura 5.29:** Esempio di omotetia a  $180^\circ$

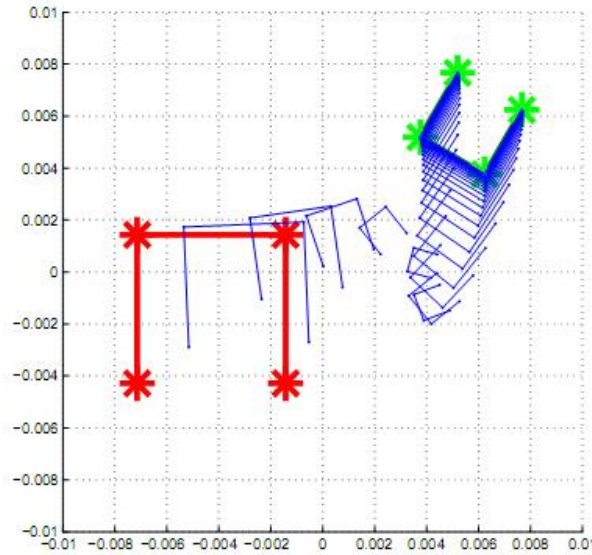
Il movimento corretto da eseguire è una sola rotazione attorno all'asse  $z$ . In questi casi, invece, l'asservimento tende a far passare la telecamera per il punto improprio dell'asse  $z$  del sistema di riferimento della telecamera allontanandosi all'infinito dall'oggetto, per poi riavvicinarsi dal lato opposto, come mostrato in figura 5.30. Ovviamente questo movimento non è realizzabile in pratica.



**Figura 5.30:** Controllo IBVS: passaggio da un punto singolare

Nella figura si mostrano la vista iniziale (in rosso) e la vista desiderata (in verde) assegnate all'asservimento. Le linee azzurre rappresentano le traiettorie desiderate per portare l'immagine a coincidere con quella desiderata; le linee magenta sono le traiettorie effettivamente seguite dalle feature dell'immagine durante la simulazione. In questo caso i punti immagine collassano in un punto per poi uscire dalla visuale; è stata riprodotta la

situazione descritta precedentemente: la traiettoria desiderata è irrealizzabile e passa per un punto singolare, quindi il controllo non agisce correttamente. Invece, quando non si passa esattamente per un punto singolare, ma comunque molto vicino ad esso, si osserva una brusca variazione nel controllo, come mostrato in figura 5.31.



**Figura 5.31:** Controllo IBVS: passaggio vicino ad un punto singolare

Una possibile soluzione al problema dei punti singolari si basa sull'estrazione della deformazione rigida (omografia totale  $\mathbf{H}_d$ ) tra posa attuale e posa desiderata del target per poi pianificare il moto del robot in due passi distinti:

1. Rototraslazione rigida nel piano immagine (telecamera) che porta la vista corrente vicino alla vista desiderata;
2. Aggiustamento della forma e delle dimensioni dell'oggetto (nella vista).

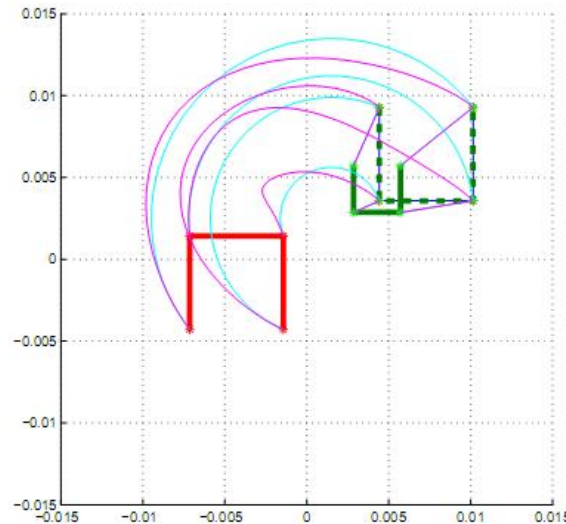
In termini di omografia, i passi comportano una scomposizione dell'omografia totale  $H_d$  in due omografie successive:

$$\mathbf{H}_d = \mathbf{H}'\mathbf{T}$$

$$\mathbf{H}' = \mathbf{H}_d\mathbf{T}^{-1}$$

dove  $\mathbf{T}$  è un'omografia che deforma l'immagine secondo una rotazione rigida attorno ad un punto (eventualmente improprio per includere anche le traslazioni pure) e  $\mathbf{H}'$  è il residuo dell'omografia necessario per portare la vista ottenuta per rototraslazione rigida su quella desiderata.

L'applicazione dell'algoritmo appena proposto al caso che ha portato alla singolarità in



**Figura 5.32:** Controllo IBVS: soluzione al passaggio vicino ad un punto singolare

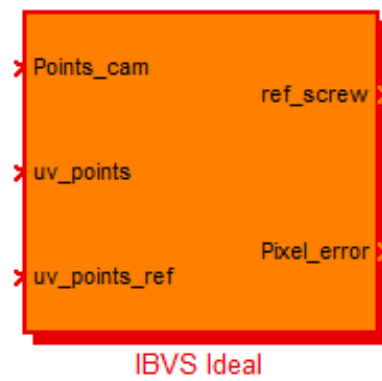
precedenza porta al risultato in figura 5.32.

Come si vede dall'immagine, il primo passo porta il target iniziale (in rosso) nella posizione intermedia (verde tratteggiata). Il secondo passo deve solo eseguire un'operazione di scalatura (allontanamento della telecamera) e traslazione delle feature.

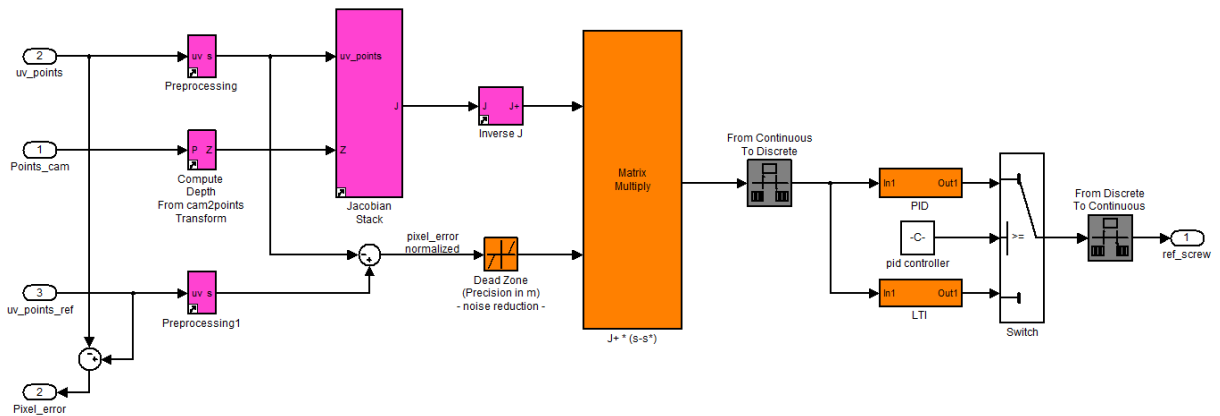
#### 4.4 Schema Simulink

In figura 5.33 è riportato lo schema a blocchi Simulink sviluppato per la simulazione del controllore IBVS. Il blocco riceve in ingresso le coordinate in pixel delle feature individuate e le coordinate della posizione desiderata delle stesse feature, sempre riferite al piano immagine. In uscita viene generato il vettore delle velocità cartesiane della telecamera. Più in dettaglio, le coordinate dei pixel vengono confrontate col modello dell'oggetto per la stima della coordinata  $Z$ , necessaria al calcolo dello jacobiano immagine; successivamente viene calcolata la matrice pseudoinversa dello jacobiano che, moltiplicata per il vettore dell'errore di posizionamento delle singole feature permette di ottenere il segnale di errore da sottoporre a controllo; come nel caso del PBVS, il regolatore può essere scelto tra un PID o un sistema LTI appositamente costruito.





(a) Blocco del controllo IBVS.



(b) Schema a blocchi del controllo.

**Figura 5.33:** Schema a blocchi Simulink del controllo IBVS

## 5 Conclusioni

Nel capitolo sono state trattate diverse tecniche di controllo per assolvere il compito della *guida robot*, ovvero elaborare l'informazione del mondo esterno acquisita sotto forma di immagini trasformandola in un segnale di controllo per un manipolatore. La trattazione è partita dalla guida robot più utilizzata in ambito industriale, cioè il controllo in anello aperto, per poi passare all'introduzione dei controlli in retroazione, i quali rappresentano il fronte ancora aperto della ricerca nell'ambito dell'integrazione tra sistemi di visione e sistemi meccanici.

Il controllo in anello aperto permette il completo disaccoppiamento delle problematiche relative alla visione da quelle del controllo del robot. Nell'ambito di questo lavoro, la presentazione di questo controllo ha fornito lo spunto per una rapida panoramica delle tematiche coinvolte nella modellazione e successivo controllo di un sistema meccanico. A lato della trattazione sono stati presentati gli schemi Simulink e le classi Matlab implementate per la successiva simulazione di controllori in asservimento visivo.

In seconda istanza, si è cercato di elencare una serie di possibili classificazioni per il controllo *vision in the loop*. Questo termine viene associato in generale a tutti i sistemi di guida robot che sfruttano la visione artificiale, ma in questo capitolo si è cercato di circoscrivere l'ambito di riferimento ad un insieme limitato di configurazioni. Di particolare interesse è la configurazione *eye-in-hand*, poiché essa prevede il montaggio della telecamera direttamente sull'end-effector del manipolatore. Questa configurazione, infatti, è l'emblema del visual servoing poiché la realizzazione di un sistema di questo tipo comporta la presenza di più parti in movimento: l'end-effector, la telecamera e, eventualmente, gli oggetti da inseguire.

Nei paragrafi successivi, infine, sono state presentate le due principali categorie di leggi di controllo, ovvero quelle basate su stima della posizione e quelle che elaborano la legge di controllo direttamente sulle caratteristiche delle immagini. Il primo tipo di controllo fornisce l'aggancio tra le tecniche di controllo in retroazione e quelle in anello aperto, poiché non si distaccano completamente dalla conoscenza della cinematica del robot e dei parametri di calibrazione della telecamera. L'altra tipologia di asservimento visivo, al contrario, è quella più estrema perché permette il controllo solo in funzione delle immagini acquisite e perfino senza necessità di calibrazione del dispositivo di visione.

Le tecniche di asservimento visivo permettono di incrementare le possibilità di controllo dei sistemi meccanici, ma la loro implementazione non è semplice perché apre delle

tematiche nuove rispetto al controllo in anello aperto. Tali tematiche hanno il *movimento* come filo conduttore. Come già detto, infatti, se nel controllo in anello aperto la tendenza più comune in ambito industriale prevede la costruzione di aree dedicate alla visione in cui gli oggetti da individuare vengono fatti stazionare, al contrario la base del controllo in asservimento visivo risiede nell'analisi del movimento della scena. Tale analisi ha influenza sia sullo sviluppo del sistema di elaborazione immagini che del controllo del manipolatore. Dal punto di vista dell'elaborazione immagini, le tecniche già presentate nei capitoli 3 e 4 devono essere necessariamente estese per poter affrontare il *tracciamento* di feature non ferme, ma in costante movimento. Il tracciamento, infatti, si deve scontrare con il problema del tempo: se la legge di moto del controllore deve essere aggiornata nel tempo, il sistema di visione deve essere in grado di fornire tali informazioni ad una frequenza costante e sufficientemente elevata, ovvero in tempo reale. Queste problematiche vengono approfondite nel capitolo 6.

Per quanto concerne lo sviluppo del controllore, infine, l'introduzione del sistema di visione non può essere considerata indolore, ma porta necessariamente all'introduzione di ritardi e anelli di controllo a frequenza diversa dalla frequenza di controllo principale. Pertanto, l'insorgere di queste tematiche deve portare all'analisi del comportamento *dinamico* del controllo in asservimento visivo, la quale analisi viene affrontata nel capitolo 7.



# Capitolo 6

## Metodologie per il Tracking Visivo

### Indice

---

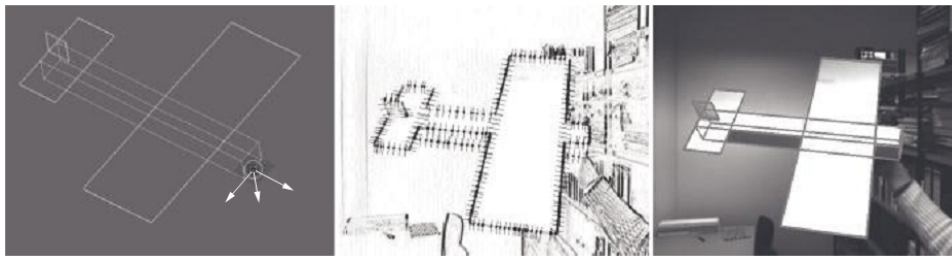
1	Classificazione e sequenza d'elaborazione . . . . .	<b>226</b>
1.1	Classificazione per algoritmi . . . . .	227
1.2	Classificazione per tematiche . . . . .	227
1.3	Sequenza d'elaborazione . . . . .	232
1.4	Linee guida per l'implementazione di un modulo di tracciamento	233
2	Algoritmi di tracciamento basati su feature . . . . .	<b>235</b>
2.1	Moving Edges Tracker . . . . .	236
2.2	Flusso ottico . . . . .	239
2.3	Tracciamento Mean Shift e Camshift . . . . .	246
3	Tracciamento basato su viste multiple di un modello . . . . .	<b>247</b>
3.1	Costruzione del descrittore geometrico . . . . .	249
3.2	Individuazione dell'oggetto in una scena . . . . .	255
3.3	Tracciamento . . . . .	260
3.4	Simulazioni . . . . .	262
4	Conclusioni . . . . .	<b>270</b>

---

Le tecniche presentate nei capitoli 3 e 4 permettono di ottenere alcune proprietà geometriche sugli oggetti inquadrati da una telecamera. L'utilizzo di queste informazioni è fondamentale per semplificare il contenuto di un'immagine e renderlo utilizzabile da un calcolatore. Il limite degli algoritmi fin qui presentati è dato dal fatto che vengono applicati ad una singola immagine e quindi permettono di estrarre un informazione istantanea relativa solamente alla scena inquadrata. Questo approccio è sufficiente, ad esempio, per

risolvere i classici problemi di manipolazione in cui è possibile predisporre un'area dedicata al sistema di visione in cui gli oggetti da manipolare sono fermi per tutto il tempo richiesto per il completamento del compito di manipolazione. In molte applicazioni, tuttavia, questo accorgimento è limitante o impossibile da mettere in pratica, pertanto nasce la necessità di estrarre informazioni su oggetti in movimento contemporaneamente all'esecuzione del task di manipolazione.

Il problema del *tracciamento* (*tracking*) consiste nel localizzare in tempo reale uno o più oggetti noti o meno, detti oggetti target, all'interno di una scena osservata da una telecamera. Il termine real-time è da intendersi alla maniera classica, ovvero con la necessità di avere a disposizione le informazioni in modo costante entro un limite di tempo prefissato. Nell'ambito dell'elaborazione di un flusso di immagini, questo implica l'estrazione di informazioni ad una frequenza la più vicina possibile alla frequenza d'acquisizione della telecamera. Un esempio di compito di tracciamento è mostrato in figura 6.1 in cui, dati un'immagine della scena e il modello tridimensionale dell'oggetto da individuare, questo è tracciato se ne viene ricostruita la posa nell'immagine corrente e nelle successive.



**Figura 6.1:** Esempio di tracking basato su modello

L'ampliamento di un algoritmo di elaborazione immagine ad un intero flusso video composto da più immagini porta all'insorgenza di nuove tematiche che introducono delle non-linearità nella ricerca di una soluzione univoca al problema del tracciamento:

- *Oggetti in movimento.* Si ha a che fare con continue variazioni della posa relativa oggetti-telecamera che può introdurre anche cambiamenti nell'apparenza degli oggetti. Il movimento può portare inoltre all'insorgere di un problema legato all'hardware utilizzato nel realizzare il sistema di tracciamento poiché la velocità del movimento influenza direttamente la frequenza di acquisizione ed elaborazione delle immagini richiesta: è possibile seguire oggetti solo se, nel tempo intercorrente tra l'elaborazione di due immagini successive, il movimento effettuato dall'oggetto è limitato. In altre parole, se la frequenza d'acquisizione ed elaborazione delle immagini è maggiore della frequenza cui si muove l'oggetto.

- *Occlusioni e distorsioni degli oggetti.* In alcune applicazioni è possibile che in alcune scene non sia visibile l'oggetto completo, o comunque una sua proiezione completa sul piano immagine, perché un altro oggetto potrebbe occludere parzialmente l'oggetto da tracciare. In questo caso è necessario applicare delle tecniche che permettano di riconoscere l'oggetto a partire dall'osservazione ed estrazione di informazioni parziali su di esso. La complessità del tracciamento dipende inoltre dalla tipologia degli oggetti, ovvero dalla loro rigidità e tridimensionalità. La tipica soluzione prevede di ricondurre l'oggetto ad un insieme di forme geometriche note e di introdurre nel sistema di visione artificiale un'informazione a priori relativa alla forma e apparenza degli oggetti, tipicamente sotto forma di modelli.
- *Apparenza degli oggetti e condizioni ambientali non costante.* La componente principale del tracking è il tempo. Le variazioni, infatti, non riguardano solo la posa degli oggetti, ma anche le condizioni ambientali, ovvero dello sfondo della scena inquadrata. Nel caso dell'elaborazione delle immagini si possono avere variazioni legate a come l'oggetto viene visto dalla telecamera, che portano tipicamente a cambiamenti negli algoritmi di segmentazione da utilizzare.

Spesso il compito di inseguire un oggetto nel tempo viene confuso con la banale iterazione delle tecniche di elaborazione base su più immagini. Dalle tematiche appena introdotte, risulta evidente che questa soluzione non è in pratica applicabile perché la variabilità degli scenari rende impossibile trattare le immagini singolarmente senza tenere conto della storia del tracciamento.

In letteratura sono descritti numerosi sistemi per il tracciamento visivo. Essi si differenziano tra di loro principalmente in base alla componentistica hardware e alle strategie adottate per il riconoscimento del movimento e della traiettoria degli oggetti target. Ognuno di questi sistemi descrive una particolare soluzione al problema del tracciamento. Date infatti le grandi differenze che ci possono essere tra un tipo di applicazione l'altro, raramente si sono visti sforzi di unificare il problema del tracciamento.

Le tematiche di tracciamento vengono tipicamente affrontate relativamente alla soluzione di problemi di sorveglianza o manipolazione con asservimento visivo ([33]).

Le principali alternative analizzate riguardano l'utilizzo di (i) punti d'interesse, o keypoint ([59], [3]), (ii) modelli tridimensionali completi ([15], [12]) e (iii) approcci ibridi basati su *feature fusion* ([52]). I diversi approcci sono riassunti in [45]. Infine, altra importante tematica inerente al problema del tracciamento è relativa alla stima del movimento ([23]).

Nello sviluppo di questa parte del lavoro di tesi, è stato considerato a fondo l'approccio basato sul riconoscimento di un modello. In particolare, il modello considerato è formato da un grafo di possibili pose e scene assunte da un oggetto, in modo del tutto simile a come trattato in [57] e con utilizzo di proprietà invarianti come in [24]. Le feature immagine prese in considerazione sono quindi di tipo geometrico e non statistico. Tale approccio è tutt'oggi in grado di offrire risultati soddisfacenti in ambito industriale ([68]) dove spesso si ha a che fare con oggetti non texturizzati, pertanto difficili da descrivere con caratteristiche locali.

All'interno della tesi, la tematica del tracciamento viene presentata subito dopo la classificazione del controllo in asservimento visivo poiché i due argomenti sono fortemente dipendenti uno dall'altro ([4]) poiché il movimento del robot porta a dover analizzare movimenti all'interno della scena, i quali a loro volta generano il moto del manipolatore. Per l'analisi del problema del tracciamento da un punto di vista generale, ovvero indipendente dall'applicazione, le principali fonti prese in considerazione sono (i) [48], in cui si presenta anche l'implementazione di una possibile architettura software rivolta specificatamente alle problematiche di tracciamento e (ii) [67], che tratta gli algoritmi da un punto di vista più generale.

In questo capitolo vengono approfondite le diverse problematiche del tracciamento di oggetti in una scena. Lo scopo della discussione è quello di individuare dei punti comuni a tutti gli algoritmi di tracking che vanno affrontati nell'elaborazione ed implementazione di una qualsivoglia procedura di tracciamento oggetti (1. Ad esemplificazione della validità della scomposizione del problema in più fasi successive, vengono presentati due approcci di tracciamento su immagini differenti: uno (2) basato sulla ricerca di particolari texture e l'altro (3) esclusivamente geometrico. Nel presentare quest'ultimo approccio, in particolare, viene introdotto un algoritmo originale di tracciamento, basato su una fusione di dati bidimensionali estratti dall'immagine e dati tridimensionali provenienti da un modello degli oggetti da tracciare. Nel paragrafo conclusivo (4) vengono tratte alcune considerazioni su quest'algoritmo e sulle altre tematiche presentate all'interno di questo capitolo.

## 1 Classificazione e sequenza d'elaborazione

Vengono di seguito proposti due criteri di classificazione del problema del tracciamento visivo in funzione (i) degli algoritmi utilizzati per l'individuazione degli oggetti nella scena



e (ii) delle tematiche da affrontare nell'implementazione della sequenza di operazioni da eseguire.

## 1.1 Classificazione per algoritmi

Tutti gli algoritmi di tracciamento possono essere suddivisi in un numero finito di sotto-classi:

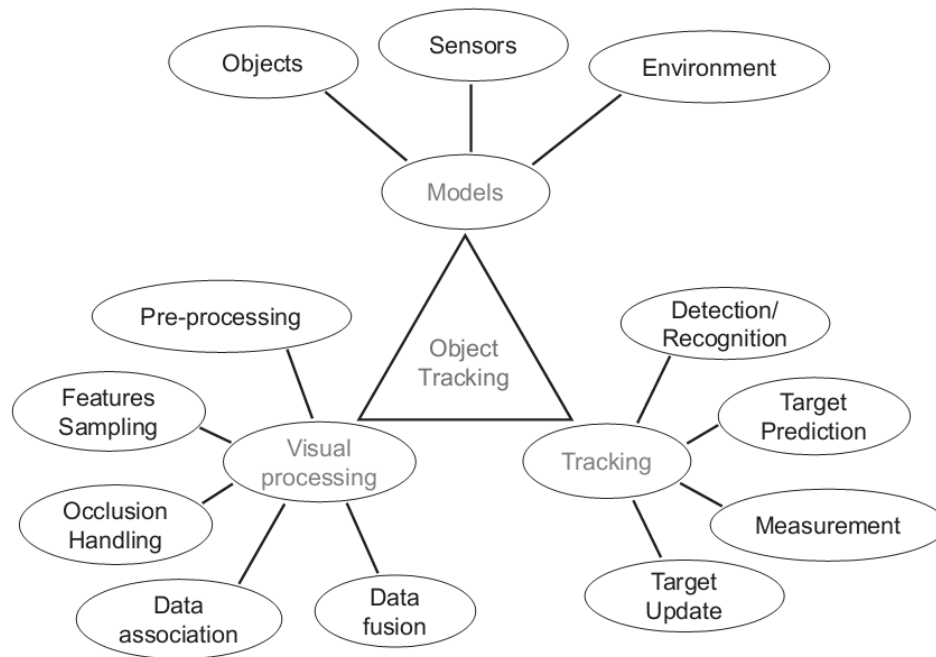
- i *blob tracker*, che definiscono l'oggetto di interesse in modo da poterlo isolare all'interno della scena, considerando il resto come sfondo. Tutti gli algoritmi appartenenti a questa categoria sfruttano il fatto che il modello dello sfondo sia noto e rimanga costante nel tempo. Un'estensione di questo concetto sono i *rilevatori di movimento* in cui lo sfondo può variare nel tempo, ma gli oggetti d'interesse sono l'unica parte in movimento tra un'immagine e l'altra. Vengono utilizzati soprattutto nell'elaborazione di immagini bidimensionali e senza conoscenza del modello degli oggetti da tracciare. Un tipico ambito di applicazione è quello della videosorveglianza;
- algoritmi basati sulla misurazione diretta della velocità dell'oggetto nell'immagine; questa tecnica è basata, ad esempio, sulla valutazione del flusso ottico e della correlazione tra immagini successive;
- algoritmi basati su opportuni *modelli* della scena. Questi cercano di riconoscere all'interno della scena delle feature presenti nel modello, le quali vengono utilizzate successivamente per aggiornare la conoscenza sull'oggetto da inseguire. Nel caso di oggetti rigidi vengono tipicamente utilizzati modelli CAD, mentre per oggetti deformabili si utilizzano delle strutture, dette *snake*, che permettono di descrivere i gradi di libertà delle deformazioni degli oggetti;
- algoritmi basati sul tracciamento di caratteristiche comuni ad una sequenza di immagini, che, in genere, consistono in punti o linee. Tali algoritmi vengono applicati ad oggetti difficili da descrivere geometricamente, ma caratterizzati dalla presenza di informazioni visive e texture facilmente identificabili da una telecamera.

## 1.2 Classificazione per tematiche

Dal punto di vista delle tematiche da affrontare, come prima approssimazione è possibile suddividere il problema del tracking in tre diversi sottoproblemi, riassunti in figura 6.2:

- Rappresentazione di *modelli* degli oggetti da inseguire, sensori e/o ambiente esterno;

- *Elaborazione delle immagini* per l'ottenimento di misure associate agli oggetti da identificare;
- *Tracciamento* vero e proprio degli oggetti nella sequenza di immagini attraverso un ciclo di predizione-misura-aggiornamento.

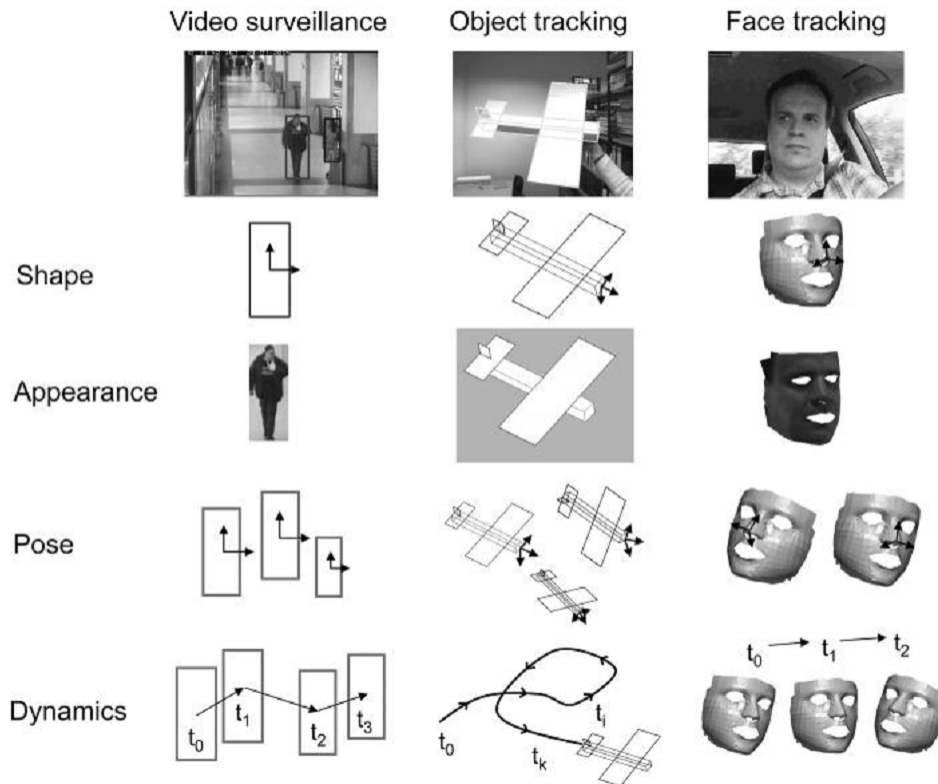


**Figura 6.2:** Macrotematiche relative al problema del tracciamento

### 1.2.1 Modelli

Alla base del *modello* di un oggetto c'è una più o meno specifica *conoscenza a priori* dell'apparenza dell'oggetto in un'immagine. I modelli possono essere sia di tipi statistico che geometrico e per la loro costruzione vengono utilizzati i descrittori presentati nei paragrafo 2, capitolo 3 e 3, capitolo 4. Il livello di tale conoscenza necessaria per il buon successo dell'individuazione dell'oggetto e del suo tracciamento dipende dal tipo di oggetto e dall'applicazione da sviluppare. In figura 6.3 sono riportati alcuni esempi di modelli basati sulla geometria, apparenza o movimento degli oggetti all'interno della scena.

Il modello di una persona per un'applicazione di video-sorveglianza può essere rappresentato da un insieme di informazioni relative alle proporzioni delle varie parti del corpo umano. Al contrario, se si ha a che fare con oggetti tridimensionali anche complessi, probabilmente è necessario avere a disposizione l'informazione complessiva sulla conformazione e disposizione delle superfici che costituiscono l'oggetto. Infine, se è necessario identificare un volto di una persona, l'informazione necessaria può essere relativa solamente al colore



**Figura 6.3:** Esempi di modellazione di oggetti per la costruzione di descrittori

da individuare nelle immagini, ovvero all'apparenza dell'oggetto target.

Se il problema non fa riferimento alla sola identificazione di un determinato set di oggetti all'interno delle immagini, ma anche al calcolo della sua posa relativamente al sistema di riferimento della telecamera, l'informazione che non può mancare all'interno del modello del target è costituita dai *gradi di libertà*. Questi definiscono in quali modi la forma di base può essere deformata e, in secondo luogo, di quante coordinate si ha bisogno per la ricostruzione della posa dell'oggetto all'interno della scena, ovvero la stima della trasformazione prospettica che porta l'oggetto ad essere visto in un determinato modo. Infine, anche la dinamica dell'oggetto può essere modellizzata.

Per poter stimare la posa dell'oggetto è necessario avere a disposizione anche un modello dei sensori, cioè delle telecamere. Questa informazione è contenuta nei parametri intrinseci ed estrinseci (vedi 2, capitolo 2) del modello prospettico della telecamera ed è ottenibile tramite le procedure di calibrazione (vedi 3, capitolo 2).

Infine, se disponibile e se l'applicazione lo permette, potrebbe essere utile avere anche informazioni a priori sull'ambiente in cui gli oggetti sono posizionati. Questa informazione riguarda quindi gli algoritmi e i loro parametri da utilizzare per la segmentazione delle immagini e, quindi, la separazione delle parti di immagine appartenenti al target da quelle dello sfondo.

### 1.2.2 Elaborazione delle immagini

La fase di elaborazione delle immagini ha come scopo l'individuazione delle caratteristiche geometriche o visive necessarie al confronto dell'osservazione con i modelli. Di seguito vengono riassunte le fasi in cui può essere scomposta una generica procedura di elaborazione delle immagini per il matching con un modello, fondamentale per tutti gli algoritmi di tracciamento:

1. *Pre-processing*. L'immagine di partenza viene preparata per mettere in rilievo le caratteristiche da estrarre per la descrizione degli oggetti;
2. *Estrazione feature*. Partendo dall'immagine elaborata nella fase precedente, si ottiene l'informazione basilare per il riconoscimento degli oggetti e per la stima della loro posa. A differenza della classica elaborazione delle immagini, nel tracking gli algoritmi di estrazione delle caratteristiche sono ottimizzati perché l'elaborazione non può essere computazionalmente troppo pesante. Solitamente si hanno due diversi algoritmi: (i) uno più accurato per individuare le caratteristiche su una qualsiasi immagine e (ii) uno che sfrutta i rilevamenti fatti al passo precedente per ottenere lo stesso risultato con meno sforzo.
3. *Confronto*. Le informazioni estratte vengono infine confrontate con le informazioni note a priori con lo scopo di segnalare o meno la presenza dell'oggetto target all'interno della scena. In generale, il problema del *matching* viene affrontato da un punto di vista statistico. Questa fase è spesso irrobustita unendo le misure di più sensori di tipo diverso (*data fusion*).

### 1.2.3 Tracciamento

Questa fase costituisce il vero nucleo del problema complessivo del tracciamento. L'informazione calcolata al passo d'elaborazione precedente viene utilizzata per stimare l'evoluzione del movimento dell'oggetto target nella scena in modo da ottenerne l'individuazione in tempo reale.

Un sistema completamente autonomo deve essere in grado di capire quando un target è stato perso e, quindi, terminare o riprendere il tracking nel modo più opportuno. Monitorare la qualità delle stime ottenute è un passo cruciale per individuare i target persi. Questo può essere fatto in molti modi, ad esempio tramite:

- *Misure statistiche.* Si può dichiarare una perdita di tracciamento quando lo stato delle statistiche calcolate presenta una grande incertezza in confronto alla dinamica attesa;
- *Misure residue.* Dopo un aggiornamento, il calcolo dei residui di una misura può essere utilizzato per definire la qualità del tracciamento: se i residui sono troppo alti, allora il tracking è stato perso.

Nella fase di tracking, misure di verosimiglianza vengono utilizzate per aggiornare la conoscenza riguardo allo stato dei target, rappresentato per ogni oggetto da una più o meno generica statistica a posteriori in un contesto di filtraggio bayesiano (predizione-correzione).

#### 1.2.4 Stima della velocità

Gli algoritmi di tracciamento trovano la loro perfetta applicazione in problemi in cui si ha a che fare con oggetti in movimento. Infatti, l'elaborazione su un flusso di immagini permette di avere a disposizione informazioni temporali per il calcolo della velocità con cui il target si sta muovendo. La misura della velocità è fondamentale per poter implementare tecniche di controllo che permettano di minimizzare l'errore di inseguimento del controllore di un sistema meccanico che debba seguire oggetti in movimento.

La stima della velocità si basa sulla conoscenza degli istanti d'acquisizione ( $T_1$  e  $T_2$ ) e delle coordinate sul piano immagine, quindi in pixel, dei punti corrispondenti ( $\mathbf{X}_1$  e  $\mathbf{X}_2$ ) di due immagini successive. Note queste quantità e applicando la definizione di rapporto incrementale, è sempre possibile stimare due tipi di velocità:

- *Stima della velocità nel sistema di riferimento oggetto.* Se è possibile ricavare, a partire dalla matrice  $\mathbf{X}$  di dimensione  $2 \times n$  contenente le coordinate di alcuni punti dell'oggetto sul piano immagine, la trasformazione geometrica  $\mathbf{H}$  che permette di proiettare tali punti nella matrice  $\mathbf{Y}$  di dimensione  $n \times 3$  di coordinate spaziali, il vettore velocità per ogni punto  $i$ -esimo è dato da

$$\dot{\mathbf{Y}}^i = \frac{\mathbf{Y}_1^i - \mathbf{Y}_2^i}{T_2 - T_1} = \frac{\mathbf{H}_1 \cdot \mathbf{X}_1^i - \mathbf{H}_2 \cdot \mathbf{X}_2^i}{T_2 - T_1}. \quad (6.1)$$

La precisione della stima dipende dalla precisione con cui vengono stimate le trasformazioni  $\mathbf{H}$ . Il calcolo del vettore velocità per ogni corrispondenza trovata è necessario nel caso si abbia a che fare con oggetti deformabili. Al contrario, nel caso di oggetti rigidi, a livello teorico si dovrebbe ottenere lo stesso vettore velocità per ogni punto

preso in considerazione; in realtà, a causa degli errori introdotti dalla stima della matrice  $\mathbf{H}$  e dalle distorsioni delle immagini, i vettori velocità possono essere leggermente diversi. Per ottenere un solo risultato si può applicare la minimizzazione ai minimi quadrati (capitolo 3).

- *Velocità nelle coordinate immagine.* Per evitare l'introduzione di errori di stima, in taluni casi e per taluni algoritmi di controllo è sufficiente stimare la velocità dell'oggetto target direttamente nel piano immagine. In questo caso si ottiene una matrice velocità di dimensione  $n \times 2$  direttamente "in pixel". La velocità si calcola allo stesso modo della velocità precedente:

$$\dot{\mathbf{X}}^i = \frac{\mathbf{X}_1^i - \mathbf{X}_2^i}{T_2 - T_1} . \quad (6.2)$$

A differenza della velocità nelle coordinate spaziali, nel caso della velocità nel piano immagine non ha senso cercare di ridurre la matrice velocità ad un singolo vettore. Infatti, la velocità di ogni singolo punto è fortemente influenzata dalla geometria e posa dell'oggetto tracciato.

Oltre che dalla precisione nell'identificazione dei punti nelle immagini, la bontà della stima è influenzata dalla velocità di acquisizione delle immagini dalla telecamera: l'errore di stima diminuisce all'aumentare della velocità di acquisizione. Per gli scopi di questa trattazione la stima della velocità basata sul semplice rapporto incrementale è più che sufficiente, ma tale stima può essere affinata sottoponendo i valori ottenuti ad ulteriore filtraggio.

### 1.3 Sequenza d'elaborazione

In generale, il problema del tracking può essere scomposto in una sequenza di passi (*pipeline*):

1. *Acquisizione dati.* L'informazione è incapsulata nelle immagini acquisite dalle telecamere. Dato che il tracking affronta il problema del riconoscimento e inseguimento di oggetti in una serie temporale di immagini (stream video), ad ogni immagine può essere associata una *timestamp*;
2. *Predizione dello stato.* Il tracker bayesiano genera una o più ipotesi riguardo allo stato dell'oggetto all'istante temporale corrispondente alla timestamp dell'immagine attualmente elaborata. La predizione è basata sulla precedente distribuzione dello stato e sulla dinamica del sistema;

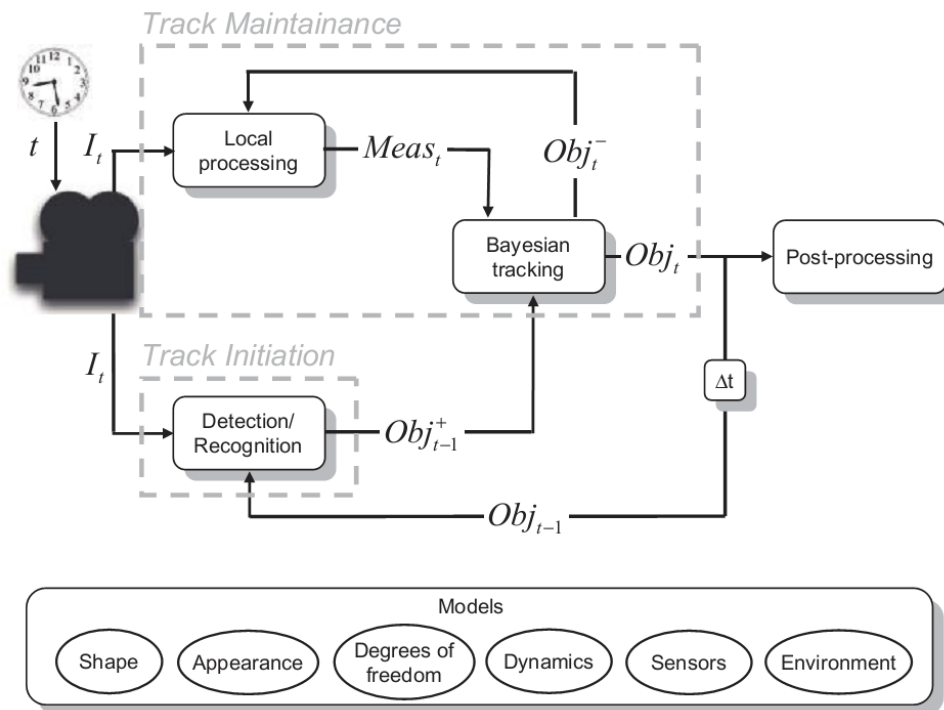
3. *Preprocessing*. Le immagini sono elaborate senza tenere conto del modello in modo da fornire un set di dati non ancora associati alle feature che compongono il template del target;
4. *Campionamento delle feature del modello*. Le ipotesi calcolate al punto 2 vengono usate per campionare dal set di feature identificate (punto 3) le feature migliori per tracciare il movimento dei target, ovvero le più robuste o, più in generale, quelle corrispondenti al template. Queste feature sono proiettate nello spazio del modello (*back-projection*) per effettuare il *matching*;
5. *Data association*. Le feature del modello sono confrontate con le feature estratte dalle immagini per produrre un insieme di misure associate al target;
6. *Data fusion*. I dati associati (punto 5) ottenuti da diverse telecamere e usando diverse feature sono combinati per ottenere un vettore di misure globali o una funzione di verosimiglianza globale;
7. *Aggiornamento dello stato*. La distribuzione a posteriori viene aggiornata a partire dalle misure effettuate. A volte può essere utile avere a disposizione una cifra di merito per valutare la qualità delle misure effettuate e, di conseguenza, del tracciamento;
8. *Aggiornamento delle feature online*. Lo stato in uscita (*output state*) è usato come informazione a priori per campionare le feature nelle immagini successive.

## 1.4 Linee guida per l'implementazione di un modulo di tracciamento

In figura 6.4 viene mostrata la sequenza di operazioni che permette di risolvere il problema del tracciamento di oggetti basandosi sulla conoscenza a priori di un modello dell'oggetto in questione.

I principali moduli del sistema sono:

- *Modelli*. Rappresentazione della conoscenza a-priori disponibile offline sugli oggetti, sensori e (possibilmente) sull'ambiente;
- *Inizializzazione/Finalizzazione*. Metodi d'identificazione e riconoscimento oggetti per inizializzare il tracking e decidere se e quando un oggetto è stato perso;
- *Tracciamento (mantenimento)*. Misurazioni, “data association & fusion”, postprocessing e visualizzazione dell'output.



**Figura 6.4:** Componenti e sequenza di operazioni di un modulo per il tracciamento di oggetti

Facendo ancora riferimento allo schema di figura 6.4,  $Obj$  rappresenta la distribuzione di probabilità multivariata che descrive l'oggetto target. Questa rappresentazione deve essere aggiornata nel tempo usando l'informazione proveniente dal sistema di visione ( $I_t$ ). In particolare, per l'inizializzazione del modulo di tracciamento, l'elaborazione delle immagini consiste nell'esecuzione di algoritmi di riconoscimento oggetti e di stima della loro posa per:

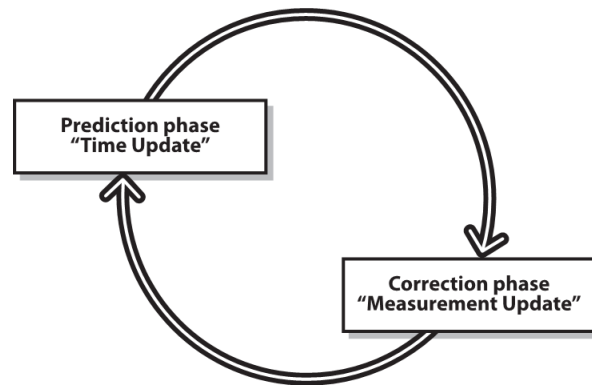
- Aggiungere nuovi target ( $Obj_{t-1}^+$ ) appena entrati in scena alla lista di oggetti da tracciare;
- Rimuovere gli oggetti per i quali il tracciamento è stato perso.

Dal punto di vista computazionale, gli algoritmi di inizializzazione sono generalmente più onerosi rispetto a quelli utilizzati per il tracking effettivo. Di conseguenza, il riconoscimento di oggetti deve essere limitato alla sola fase di inizializzazione ( $t = 0$ ) o, tutt'al più, ripetuto periodicamente ogni  $n$  immagini.

La parte superiore del sistema è costituita dai moduli per il mantenimento del tracking, dove i target attualmente tracciati sono sottoposti alle fasi di *predizione*, *misura* e *correzione* (figura 6.5), le quali modificano il loro stato (ad esempio i loro parametri di posa). Se il tracker è di tipo bayesiano, nella fase di predizione le informazioni relative a



$Obj_{t-1}$  vengono utilizzate per produrre la distribuzione a priori  $Obj_t^-$ . Questo stato viene quindi utilizzato per produrre misure  $Meas_t$  associate al target per l'aggiornamento bayesiano: l'informazione a priori  $Obj_t^-$  viene aggiornata in informazioni a posteriori ( $Obj_t$ ), cioè l'output del modulo di tracciamento.



**Figura 6.5:** Aggiornamento del modulo di tracciamento: predizione e aggiornamento

## 2 Algoritmi di tracciamento basati su feature

Quando gli oggetti da tracciare non sono facilmente descrivibili attraverso modelli geometrici, ma presentano delle caratteristiche visive facilmente individuabili, si parla di tracciamento basato su feature (*feature-based tracking*). Il tipico esempio è quello di oggetti con etichette per cui risulta facile individuarne le lettere, colori o anche semplicemente contorni e spigoli.

Il modello è costituito dalle stesse caratteristiche che vengono ricercate in ogni immagine. Questo, in unione con l'assenza di informazione geometrica nelle caratteristiche, permette di definire un semplice metodo di apprendimento dei modelli: l'oggetto da ricercare viene semplicemente mostrato al sistema di visione, il quale ne cataloga le caratteristiche che riesce ad individuare in più immagini.

Il principale vantaggio di questi algoritmi risiede nella mancanza di informazione geometrica nella caratterizzazione degli oggetti. Proprio per questo motivo tali algoritmi risultano efficaci anche con oggetti deformabili. Infatti, mentre le deformazioni hanno un effetto deleterio sulle caratteristiche completamente geometriche (ad esempio una retta che diventa un arco di circonferenza o un'altra curva), tale effetto non si ha su alcune caratteristiche (ad esempio i punti di un'immagine). Pertanto, è possibile continuare a tracciare l'oggetto anche in presenza di deformazioni.

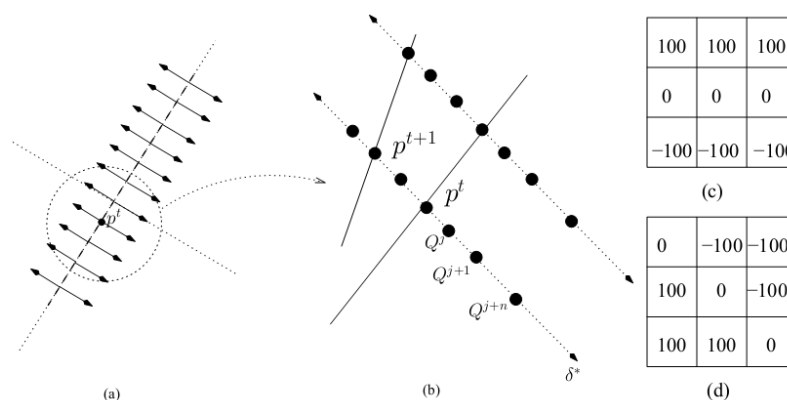
Il problema sorge quando, oltre all'individuazione dell'oggetto, è necessario calcolarne anche la posa. In questo caso, però, è sufficiente aggiungere al modello un'informazione relativa alla geometria dell'oggetto. Tale informazione non influenza la fase di ricerca dell'oggetto all'interno della scena, ma entra in gioco nella stima della trasformazione (rigida e non) che ha portato il modello ad essere proiettato in un certo modo all'interno della scena. Dei metodi per ottenere questa stima se ne è già parlato nei capitoli 3 e 4.

## 2.1 Moving Edges Tracker

I classici algoritmi di estrazione dei contorni, descritti nel paragrafo 1.2 del capitolo 3 campionano i contorni ad una distanza regolare, tipicamente corrispondente alla dimensione di un pixel. L'algoritmo descritto in questo paragrafo permette di individuare, all'interno di immagini di una scena scattate ad intervalli di tempo diversi, il *movimento* dei contorni a partire dalla loro posizione iniziale individuata attraverso una prima applicazione del rilevatore scelto.

Per ogni punto si esegue una ricerca 1-dimensionale (nell'intorno di 1 pixel) nella direzione normale al contorno. Questa viene eseguita usando diverse maschere di gradiente orientato (almeno una per ogni grado, quindi 180, sfruttando la simmetria del gradiente per angoli supplementari). Il carico computazionale e l'efficienza dell'algoritmo, pertanto, è pari a quelli di una tipica operazione di filtraggio su un'immagine.

L'algoritmo è schematizzato in figura 6.6. Nell'immagine  $I^{t+1}$  viene cercata la proie-



**Figura 6.6:** Schematizzazione dell'algoritmo Moving Edges Tracker

zione  $p^{t+1}$  del punto  $p^t$ ; la ricerca viene eseguita unicamente lungo la direzione normale ( $\delta$ ) al contorno  $c^t$ . Ad ogni punto candidato alla corrispondenza  $Q^j$  si applica una maschera

di convoluzione  $M_\delta$  per calcolare la radice quadrata della funzione di log-verosimiglianza  $\zeta_j$ . La nuova posizione del pixel  $p^{t+1}$  è ottenuta attraverso la corrispondenza ottimale

$$Q^j \star = \arg \max_{j \in [-J, J]} \zeta_j \quad (6.3)$$

con

$$\zeta_j = |I_{\nu(Q_j)}^{t+1} \star M_\delta + I_{\nu(p^t)}^t \star M_\delta| .$$

La funzione  $\nu(\cdot)$  rappresenta l'intorno del pixel considerato con dimensione al massimo  $7 \times 7$ . Maggiore è la dimensione dei kernel considerati, maggiore è la robustezza ottenuta, la quale viene però pagata in termini di tempi d'elaborazione.

### 2.1.1 Fitting di curve

Per far sì che la stima delle feature possa essere fatta anche in ambienti (scene) rumorosi è necessario avere a disposizione un metodo di stima particolarmente robusto. Il metodo dei minimi quadrati (vedi 3) può essere utilizzato per calcolare la curva a distanza minima in grado di descrivere al meglio la distribuzione dei punti individuati.

I due esempi tipici sono quelli che derivano dalla trasformata di Hough (paragrafo 3.2, capitolo 3):

- *Segmenti*: è il caso più semplice perché la direzione  $\theta$  è ottenuta direttamente dai parametri delle feature (è normale alla direzione di ricerca dei punti mobili) e la parametrizzazione è quindi data dall'equazione (6.4) della retta espressa in coordinate polari.

$$x \cos \theta + y \sin \theta - \rho = 0 . \quad (6.4)$$

- *Ellissi*: l'ellisse corrisponde al caso in cui  $K_2^2 < K_1$ . I parametri  $K_i$  vengono stimati direttamente dalla lista di punti tracciati utilizzando i minimi quadrati. La parametrizzazione utilizzata è la classica equazione dell'ellissi (6.5) derivante dalla geometria analitica.

$$x^2 + K_1 y^2 + 2K_2 xy + 2K_3 x + 2K_4 y + K_5 = 0 . \quad (6.5)$$

**Spline e NURBS.** Una *spline* è definita dall'equazione parametrica

$$Q(t) = \sum_{j=-d}^{n-1} \alpha_j B_j(t), \quad t \in [0, 1], \quad (6.6)$$

dove  $\alpha_j$  sono i punti di controllo della spline,  $d$  i gradi della spline ( $d = 3$  per una cubica) e  $B_j$  rappresenta la funzione base della spline stessa. Il grado della spline, invece, è definito come  $d + 1$ .

Dal momento che il numero di punti tracciati è generalmente superiore al numero  $n + d$  di punti di controllo desiderati, i minimi quadrati possono essere utilizzati tranquillamente per la stima dei parametri della curva.

Oppure, utilizzando la definizione delle funzioni base (basis function) e dei punti di controllo, si arriva alla seguente formulazione

$$Q(t) = \sum_{j=0}^k N_{j,n}(t) P_j, \quad t \in [0, 1], \quad (6.7)$$

dove  $P_i$  sono i punti di controllo della curva e  $N_i$  le funzioni base di grado  $n$  ( $n = p - 1$ ). I punti di controllo e le corrispondenti funzioni base sono  $k + 1$  ( $k + 1 = n + d$ ).

In particolare, la funzione di minimizzazione per le spline utilizza le *NURBS* (*Non-Uniform Rational Basis Spline*). Una NURBS è definita da:

1. un vettore di  $m$  nodi (*knot-vector*)  $U = u_0, \dots, u_m$  di valori crescenti strettamente ( $u_i < u_{i+1}$ ,  $i = 0, \dots, m$ ). Il numero di tratti che formano la spline è dato da  $m$ . Il primo e l'ultimo valore sono copiati  $p$  volte, dove  $p$  rappresenta il grado ( $n + 1$ ) della curva.
2. la funzione base (*basis function*) della B-Spline  $N_{i,p}$ , definita come

$$N_{i,0}(u) = \begin{cases} 1 & , \text{ se } u_i \leq u_{i+1} \\ 0 & , \text{ altrimenti} \end{cases},$$

da cui deriva la formulazione generale della funzione base della spline:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+1} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u). \quad (6.8)$$

3. il vettore dei punti di controllo  $P_i$  (nel caso dell'immagine definito da un insieme di pixel).
4. Il vettore dei pesi  $w_i$  associati ad ogni punto ( $w_i > 0$ ).

Di fatto, l'equazione della spline (6.6) si modifica leggermente, ottenendo una nuova definizione:

$$Q(t) = \frac{\sum_{j=0}^k N_{j,n}(t)w_jP_j}{\sum_{j=0}^k N_{j,n}(t)w_j} . \quad (6.9)$$

Una spline periodica di grado  $p$ , infine, è una curva che si chiude su se stessa. Questa richiede che i primi  $p$  e gli ultimi  $p$  punti di controllo coincidano e, quindi, che anche i primi e ultimi  $p$  intervalli di parametri abbiano la stessa lunghezza.

Utilizzare i minimi quadrati per stimare un set di punti tramite una spline significa calcolare le funzioni base della curva una volta fissato il numero di knot e di punti di controllo desiderati. Il numero di knot necessari è  $k + 1$ , pari alla somma dei punti di controllo totali e del grado di ogni singolo segmento della spline ( $n + d$ ). Visto che i punti sono  $n$ , la curva va divisa in  $n$  segmenti e, quindi, si hanno  $n$  knot. Relativamente al problema dei minimi quadrati, quindi, si ha:

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} N_{0,d}(t_1) & \dots & N_{k,p}(t_1) \\ \vdots & \dots & \vdots \\ N_{0,d}(t_n) & \dots & N_{k,p}(t_n) \end{bmatrix}$$

con  $y$  punti da tracciare e  $\mathbf{A}$  matrice delle funzioni base.

I parametri da stimare  $\mathbf{p}$  sono le coordinate (ascisse e ordinate nel piano immagine) dei punti di controllo e i pesi  $\mathbf{w}$ , quindi si applicano tre volte i minimi quadrati sulle corrispettive coordinate  $y$  (coordinate dei punti tracciati e pesi unitari) per il calcolo della spline desiderata.

## 2.2 Flusso ottico

Gli algoritmi di *Optical Flow* si basano sull'estrazione del moto tra due frame (o sequenze di frame) senza ogni altra conoscenza a-priori del contenuto di questi frame. Tipicamente, infatti, la presenza di movimento in una sequenza di immagini indica che qualcosa d'interessante sta succedendo all'interno della scena stessa.

Per modellare il movimento si può associare una formulazione di velocità dei pixel o, in modo equivalente, di spostamento relativo di un certo pixel da un frame all'altro. Questo tipo di formulazione è noto in letteratura col nome di *dense optical flow* nella quale ad ogni pixel è associato un vettore velocità. In pratica, tuttavia, non è semplice calcolare questa feature. Si consideri, ad esempio, il moto di un foglio bianco: è possibile

individuare il moto solo dei punti di contorno e, in particolare, solo dei punti il cui gradiente è parallelo alla direzione del moto. Il risultato è che le tecniche di *dense optical flow* richiedono l'interpolazione del movimento dei punti individuati per comprendere il moto assoluto della scena.

L'opzione alternativa è rappresentata da metodologie di *sparse optical flow* con le quali si traccia solo il moto di un set limitato di punti. Tali punti sono tipicamente quelli per cui il moto è più facilmente individuabile. Un tipico esempio è costituito dagli angoli, la cui identificazione è stata trattata nel paragrafo 1.3 del capitolo 3.

Limitandosi allo sviluppo di algoritmi di tracking real-time è possibile prendere in considerazione solo i metodi di *sparse optical flow* che garantiscono un carico computazionale inferiore rispetto alla ricerca del moto dell'intera scena.

### 2.2.1 Lucas-Kanade Tracker

L'algoritmo *Lucas-Kanade (LK)* è stato proposto nel 1981 come tentativo di produrre risultati densi senza però la complessità degli algoritmi di *dense optical flow*. Tuttavia, dal momento che esso lavora su un set limitato di punti e considerando solamente le loro proprietà locali, viene classificato tra le tecniche di *sparse optical flow*. Un esempio di applicazione dell'algoritmo è riportato in [22].

La restrizione dell'algoritmo all'osservazione di sole proprietà locali è ottenuta considerando una piccola finestra attorno al punto da tracciare; questo fatto rappresenta una limitazione nel momento in cui diventa necessario inseguire movimenti di oggetti troppo ampi. Per tale motivo, un'assunzione fondamentale dell'algoritmo si riferisce alla necessità che le immagini vengano riprese ad una piccola distanza temporale una dall'altra. Allo stesso modo, l'elaborazione deve essere rapida. Per quest'ultimo problema, una soluzione è data dall'implementazione *piramidale* dell'algoritmo LK che consiste nell'analizzare immagini di dimensioni diverse (piccole per seguire movimenti piccoli e immagini grandi per inseguire movimenti più ampi). In figura 6.7 è riportato un esempio di applicazione dell'algoritmo per l'identificazione del flusso ottico di una serie di punti dei contorni individuati nell'immagine.

**Assunzioni dell'algoritmo** Per una corretta esecuzione dell'algoritmo, è necessario che il flusso di immagini rispetti le tre seguenti assunzioni, riassunte in seguito in figura 6.8:



**Figura 6.7:** *Algoritmo LK: esempio*

1. *Luminosità costante.* L'aspetto di un pixel di un oggetto nella scena non cambia da frame a frame. Per aspetto si intende il valore della feature in analisi, la quale può essere la luminosità, il gradiente, il colore, ....
2. *Persistenza temporale di "piccoli movimenti".* Il movimento di una superficie cambia lentamente nel tempo. In pratica, questo significa che la distanza temporale tra un frame e il successivo è sufficientemente piccola per far sì che il relativo movimento sia piccolo. Questa assunzione è strettamente collegata alla capacità di elaborazione dell'hardware disponibile e, più in generale, alle frequenze in gioco nel sistema.
3. *Coerenza spaziale.* Punti vicini in una scena appartengono alla stessa superficie e hanno un moto simile.

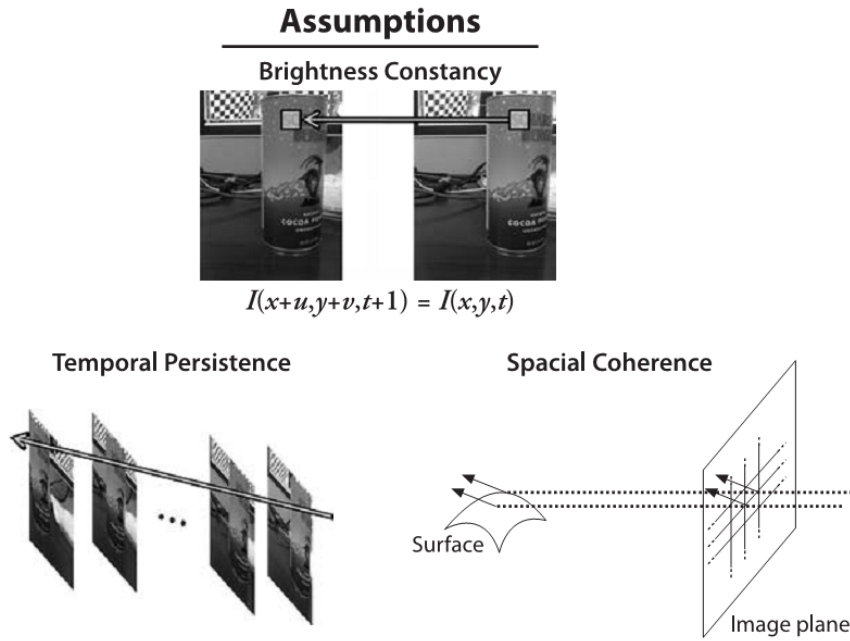
**Caso 1D: calcolo velocità.** La prima assunzione prevede che i pixel tracciati preservino il loro aspetto da un frame all'altro, ovvero, per una generica feature  $f$  legata all'intensità  $I$ :

$$f(x, t) = I(x(t), t) = I(x(t + dt), t + dt)$$

e quindi, che la derivata del valore della feature analizzata rispetto al tempo sia nulla.

$$\frac{\delta f(x)}{\delta t} = 0. \quad (6.10)$$

La persistenza temporale (cioè la presenza di piccoli movimenti) viene tradotta tramite un'approssimazione della derivata dell'intensità, quindi il valore della feature, rispetto al tempo. In altre parole, si afferma che il cambiamento da un frame al successivo è



**Figura 6.8:** Assunzioni dell'algoritmo Lucas-Kanade

differenzialmente piccolo.

In caso di movimento in una sola direzione, derivando l'intensità  $I$  rispetto alla direzione del moto e al tempo, tenendo conto della dipendenza dal tempo del moto, si ottiene l'equazione (6.11), che descrive il flusso ottico di un punto.

$$\frac{\delta I}{\delta x} \left( \frac{\delta x}{\delta t} \right) + \frac{\delta I}{\delta t} = 0 \quad (6.11)$$

con

$$\frac{\delta I}{\delta x} = I_x \quad , \quad \frac{\delta x}{\delta t} = \mathbf{v} \quad , \quad \frac{\delta I}{\delta t} = I_t \quad .$$

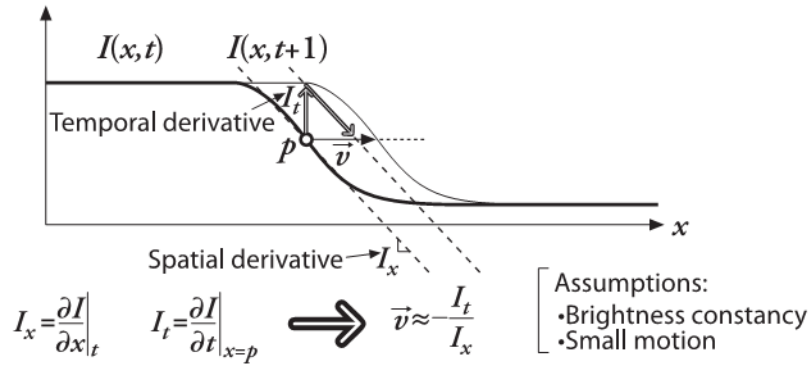
La variabile  $I_x$  rappresenta la derivata spaziale lungo la prima immagine,  $I_t$  è la derivata tra immagini in diversi istanti temporali e  $\mathbf{v}$  è il vettore velocità che si vuole ottenere. Risolvendo l'equazione (6.11) si ottiene la velocità (6.12) con cui i punti si sono mossi all'interno della scena.

$$\mathbf{v} = - \frac{I_t}{I_x} \quad . \quad (6.12)$$

La figura 6.9 mostra l'applicazione delle relazioni appena trovate ad un contorno in movimento da sinistra verso destra.

La stessa figura rivela un altro problema che può sorgere dalle assunzioni dell'algoritmo, cioè che il valore di intensità assunto del pixel non è veramente costante e il periodo di tempo tra un frame e il successivo non è sempre sufficientemente piccolo. Comunque,





**Figura 6.9:** Algoritmo LK ad 1 dimensione

queste limitazioni possono essere risolte tramite iterazioni successive dell'algoritmo: se si è abbastanza vicini alla soluzione si può usare la prima stima della velocità come punto di partenza per la successiva iterazione fino al raggiungimento di un valore asintotico della velocità.

**Dal caso 1D al 2D.** Relativamente all'analisi di immagini, la formulazione più generale dell'algoritmo si ha per il caso bidimensionale. La relazione generale è ottenuta aggiungendo la coordinata  $y$  all'equazione (6.11). Denominando  $u$  la componente della velocità lungo  $x$  e  $v$  la componente lungo  $y$ , si ottiene:

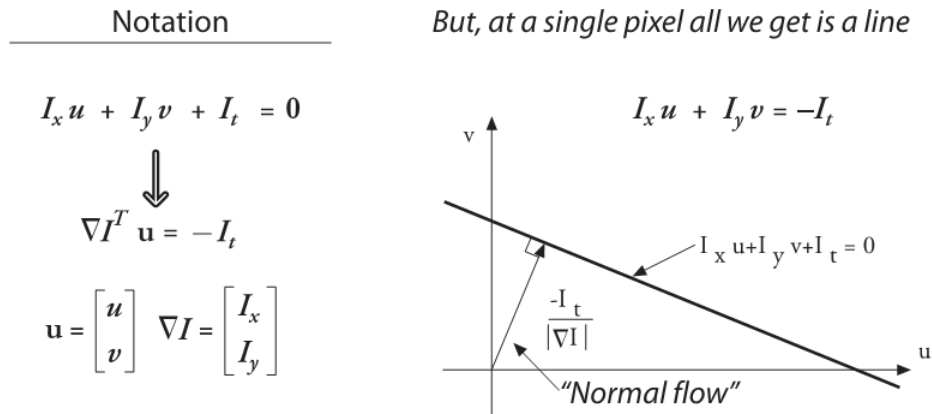
$$I_x u + I_y v + I_t = 0 . \quad (6.13)$$

In realtà, così facendo, si ottengono due incognite per ogni singolo pixel. Questo significa che le misure a livello del singolo pixel non sono più sufficienti per ottenere una soluzione unica per il moto bidimensionale di quel punto. Invece, come mostrato in figura 6.10, è possibile ottenere una soluzione unica se si considera solo il moto lungo la direzione perpendicolare (normale) ad una linea, analogamente a quanto fatto per il rilevatore del moto di contorni.

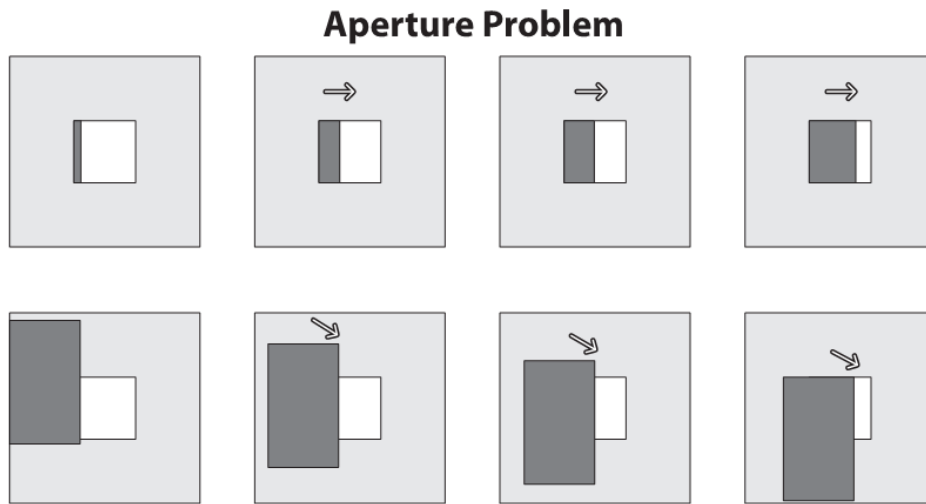
Un altro modo di vedere questo problema è quello di considerare il problema dell'*apertura*, relativo alla dimensione della finestra scelta per l'individuazione delle proprietà locali del punto d'interesse. Quando la finestra è troppo piccola si può cadere nella situazione in cui è possibile osservare solo il moto di un bordo e non di un angolo. Tuttavia, un singolo bordo è insufficiente per determinare la posizione esatta di un oggetto intero (figura 6.11).

Nel cercare una soluzione a questo problema viene in aiuto l'assunzione relativa alla *coerenza spaziale*: se il corpo si muove rigidamente è possibile utilizzare l'informazione di

### From 1D to 2D tracking



**Figura 6.10:** Algoritmo LK a 2 dimensioni



**Figura 6.11:** Algoritmo LK: problema dell'apertura

un intorno di pixel per ottenere, tramite interpolazione, il moto del pixel nascosto. Per esempio, utilizzando una finestra di  $5 \times 5$  pixel si ottiene un sistema di 25 equazioni:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} . \quad (6.14)$$

Quello appena descritto è un sistema sovrainvolto

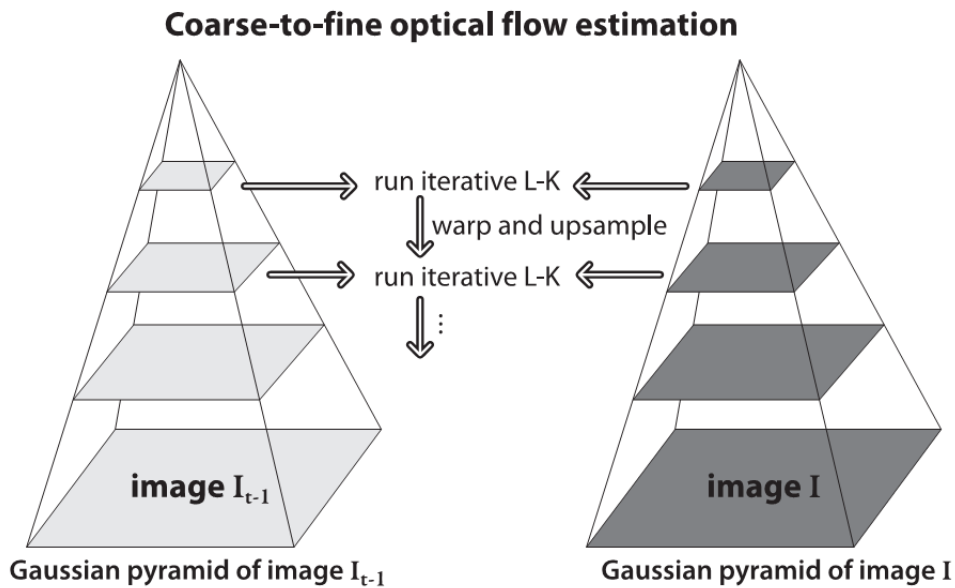
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} . \quad (6.15)$$

$$(\mathbf{A}^T \mathbf{A}) \mathbf{d} = \mathbf{A}^T \mathbf{b}$$

la cui soluzione è ottenuta ancora una volta tramite il metodo dei minimi quadrati (vedi 3):

$$\begin{bmatrix} u \\ v \end{bmatrix} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} . \quad (6.16)$$

**LK piramidale.** L'assunzione più problematica per la corretta applicazione dell'algoritmo LK è quella relativa alla limitazione a piccoli movimenti perché è quella direttamente influenzata dal framerate della telecamera a disposizione. L'onerosità computazionale dell'algoritmo aumenta all'aumentare della velocità dei movimenti da tracciare poiché per catturare movimenti ampi è necessario utilizzare finestre larghe e quindi lavorare con più pixel. Per prevenire questo problema e ottimizzare l'algoritmo è possibile implementare il tracker inizialmente su immagini a larga scala (top-layer) e raffinare i passi successivi lavorando sui livelli più bassi della *piramide gaussiana* (figura 6.12) dell'immagine fino ad arrivare ai pixel dell'immagine raw.



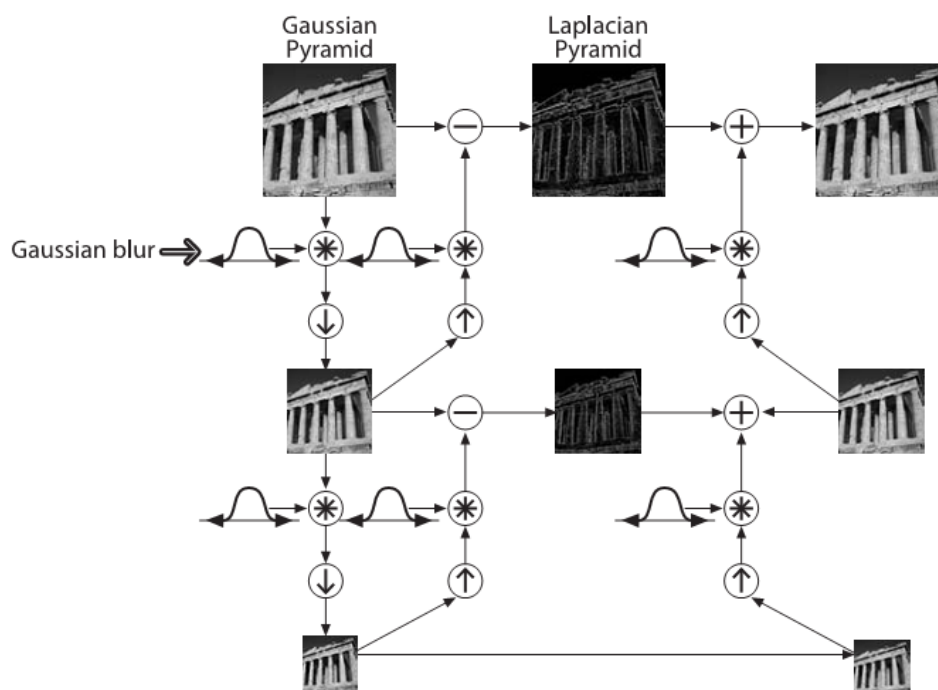
*Figura 6.12: Algoritmo LK piramidale*

La rappresentazione piramidale di un'immagine è utilizzata in molte applicazioni di visione artificiale. Ci sono due tipi di piramide: *gaussiana* e *laplaciana*. La prima è utilizzata per sottocampionare le immagini, mentre l'altra è l'operatore complementare che cerca di ricostruire l'immagine di partenza a partire da un'immagine a livello inferiore della piramide. Una piramide è essenzialmente un insieme di immagini derivanti dal sottocampionamento di una singola immagine sorgente fino al raggiungimento di un punto d'arresto. L'operazione di sottocampionamento corrisponde quindi ad una riduzione della risoluzione

dell'immagine.

Il livello  $(i+1)$  di una piramide gaussiana è ottenuto tramite un'operazione di convoluzione con un kernel gaussiano sul livello  $i$  e l'eliminazione delle righe e colonne pari dell'immagine così ottenuta. Da questo deriva che ogni immagine di livello inferiore ha dimensioni pari ad un quarto delle dimensioni delle immagini al livello superiore, con conseguente perdita d'informazione. Il punto d'arresto della costruzione della piramide deve essere scelto in modo da preservare il contenuto informativo delle feature immagine utilizzate per l'algoritmo LK.

La figura 6.13 riassume i concetti di piramidi gaussiana e laplaciana, i processi per il calcolo della piramide gaussiana e la possibile ricostruzione tramite piramide laplaciana.



**Figura 6.13:** Piramidi gaussiana e laplaciana

## 2.3 Tracciamento Mean Shift e Camshift

Due altre tecniche utilizzabili per il tracking sono l'algoritmo *mean shift* (presentato nel paragrafo 2.2, capitolo 3 come esempio di descrittore) e il suo derivato *camshift* (*continuously adaptive mean-shift*).

Nel 1998 Bradski ha introdotto l'idea di sfruttare la capacità di individuare i modi di una distribuzione di probabilità statistica per il tracciamento di feature in una sequenza video.

L'inizializzazione del tracciamento è ottenuta tramite la scelta della finestra di partenza che meglio contiene tutte le feature individuate nell'immagine. L'esecuzione dell'algoritmo mean shift sulla singola immagine permette di ottimizzare la posizione della finestra al fine di massimizzare il contenuto informativo dell'immagine stessa. In secondo luogo, passando all'elaborazione dei frame successivi, lo spostamento della scena provoca una variazione nella distribuzione di probabilità delle feature. L'iterazione dell'algoritmo mean shift permette di calcolare il vettore di spostamento medio delle feature e di centrare nuovamente la finestra delle informazioni. Lo spostamento della scena da un frame all'altro è dato dalla distanza tra il centro della finestra nel frame precedente e nel frame corrente.

L'algoritmo *camshift* costituisce un'estensione del mean shift poiché esso cerca di adattare non solo la posizione del centro della finestra delle feature, ma anche delle dimensioni della finestra stessa. Se le feature individuate sono sufficientemente dense, l'algoritmo camshift porta all'individuazione di un'area limitata cui esse appartengono e viene a coincidere con il rilevamento di feature locali e non puntuali.

### 3 Tracciamento basato su viste multiple di un modello

Gli algoritmi presentati nel paragrafo 2 non sono sempre applicabili. Ad esempio, nella tipica applicazione industriale di pick-and-place per svuotamento di cassoni, non si ha generalmente a che fare con oggetti individuabili a partire da un'etichetta o qualcosa di simile. Al contrario, tali oggetti hanno tipicamente una forma geometrica definita, nota a priori e di cui è generalmente disponibile un modello 3D, ad esempio da CAD, dell'oggetto da manipolare. In questo tipo di applicazioni, quindi, il modello diventa il centro focale dell'algoritmo di tracciamento, che prende quindi il nome di *model-based tracking*.

Come per gli altri approcci di tracciamento fin qui presentati, il problema può essere separato in due tematiche principali:

- *Individuazione delle caratteristiche geometriche.* Le caratteristiche geometriche da individuare sono quelle già presentate nel capitolo 3. Tipiche caratteristiche usate sono punti, linee e contorni.
- *Stima della posa dell'oggetto.* Fa riferimento al problema del calcolo della posa relativa oggetto-camera una volta noto il vettore delle corrispondenze tra lo spazio delle osservazioni e lo spazio del modello. Questo problema è già stato trattato nel capitolo 3, paragrafo 3.3.

Nello sviluppo di questo lavoro di tesi si è deciso di implementare un algoritmo di tracciamento che potesse trovare applicazione in ambito industriale, quindi sotto certe assunzioni:

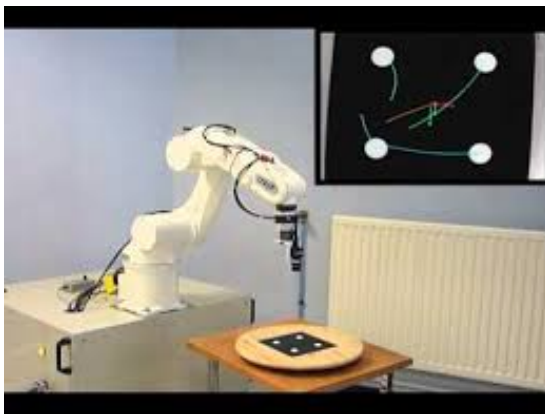
1. *Ambiente di lavoro strutturato.* Questa assunzione fa riferimento alla situazione tipica dell'ambito della visione industriale in cui gli oggetti da individuare sono situati in un ambiente strutturato appositamente per facilitarne l'individuazione da parte di un sistema di visione. Tale ambiente garantisce, ad esempio: (i) condizioni di illuminazione ottimale (oggetti scuri su sfondo chiaro); (ii) completa assenza di sovrapposizioni di oggetti; (iii) dimensione limitata dell'area e degli oggetti da individuare;
2. *Conoscenza degli oggetti da individuare.* Il sistema di visione è in grado di riconoscere qualsiasi oggetto venga proposto, ma solamente se ne possiede una conoscenza a priori. Pertanto, è necessaria una procedura di *apprendimento* per insegnare al sistema di visione artificiale quali sono gli oggetti che è chiamato ad individuare;
3. *Oggetti rigidi e scomponibili in forme geometriche.* Il tipo di approccio proposto deve essere fondamentalmente geometrico, in modo da differenziarsi nettamente dai metodi basati sull'individuazione di feature trattati nel paragrafo 2. Di conseguenza, l'individuazione degli oggetti target è indipendente dalla presenza di una forte componente texturizzata, ma essi devono necessariamente possedere una geometria ben definita, ovvero avere una forma geometrica unica o riconducibile ad una composizione di singole forme geometriche;
4. *Componente tridimensionale limitata.* L'algoritmo è in grado di identificare e tracciare anche oggetti tridimensionali, ma mostra i suoi punti di forza nel caso si abbia a che fare con oggetti dallo sviluppo tridimensionale ben definito (oggetti piatti o con superficie uniforme senza punti nascosti).

Esempi di applicazioni nell'ambito dell'automazione industriale e della robotica in cui questo approccio può essere utilizzato sono:

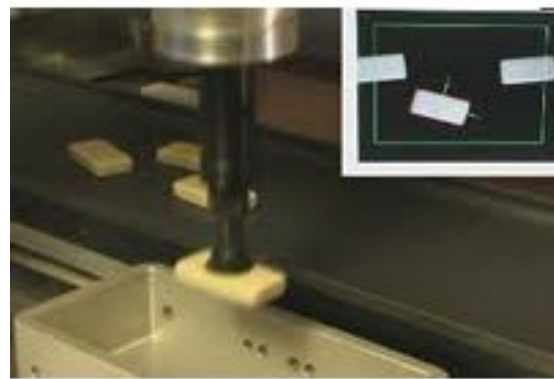
- Classiche applicazioni di pick-and-place con robot SCARA. La possibilità di tracciare il movimento degli oggetti permette di: (i) prelevare oggetti in movimento su un nastro trasportatore, (ii) prelevare oggetti piccoli o difficili da prendere montando un sistema di visione direttamente sull'end-effector del robot, configurazione di cui si è fatto cenno nel capitolo 5;

- Applicazioni di precisione (saldatura o avvitamento) con movimenti non programmati a priori dall'operatore, ma imposti dal sistema di visione;
- Posizionamento di oggetti su nastri trasportatori o tavole rotanti per esporre all'operatore un determinato strumento o pezzo su cui effettuare una lavorazione. Anche in questo caso si tratta di movimentazioni non programmate a priori, ma dipendenti dal tipo di situazione o necessità che si viene a creare sull'impianto.

Due esempi di applicazioni che rispettano questi requisiti sono riportate in figura 6.14. Nella prima si vede un robot che deve eseguire un compito di posizionamento su un oggetto ben identificabile (quattro cerchi bianchi su sfondo nero), mentre nella seconda un manipolatore deve prelevare oggetti, considerati planari, movimentati su un nastro trasportatore.



(a) Posizionamento dell'end-effector rispetto ad un piano di riferimento.



(b) Prelievo di oggetti da nastro trasportatore.

**Figura 6.14:** Esempi di applicazioni planari

La descrizione del metodo può essere suddivisa nelle sottofasi tipiche di ogni algoritmo di tracciamento: (i) costruzione del descrittore, (ii) individuazione e (iii) tracciamento degli oggetti.

### 3.1 Costruzione del descrittore geometrico

Date le assunzioni fatte in precedenza, il descrittore da utilizzarsi è fondamentalmente di tipo *geometrico* e si basa pertanto sugli strumenti presentati nel paragrafo 2.3 del capitolo 3. Basare un descrittore su semplici assunzioni geometriche potrebbe sembrare una scelta piuttosto semplicistica rispetto ad altri tipi di descrittori, ma, in realtà, nell'ambito dello sviluppo di un algoritmo di tracciamento essa risulta un buon compromesso tra *robustezza*

e velocità di elaborazione.

Solitamente, infatti, il punto debole degli algoritmi di tracciamento è la loro inizializzazione: l'inizializzazione richiede tipicamente un tempo pari a circa 100 volte quello richiesto per il tracciamento, rendendo quindi il sistema poco robusto alla perdita di tracciamento dell'oggetto. L'utilizzo di un descrittore semplice permette di diminuire questa differenza di complessità computazionale tra le fasi di tracciamento e inizializzazione; pertanto, se da un lato si rischia di perdere più spesso l'oggetto nella fase di tracciamento, dall'altro lato si ha la possibilità di recuperare immediatamente il tracciamento stesso reiterando la fase di inizializzazione. Come risulterà più evidente in seguito, in questo caso la fase di tracciamento è un sottocaso della fase di inizializzazione che introduce una serie di accorgimenti in grado di velocizzare ulteriormente l'esecuzione dell'algoritmo. Inoltre, l'avere a disposizione una procedura di inizializzazione poco complessa dal punto di vista computazionale potrebbe permettere, su alcune architetture, di non differenziare le fasi di tracciamento e inizializzazione in modo da avere sempre a disposizione la stima più precisa messa a disposizione dall'algoritmo in analisi.

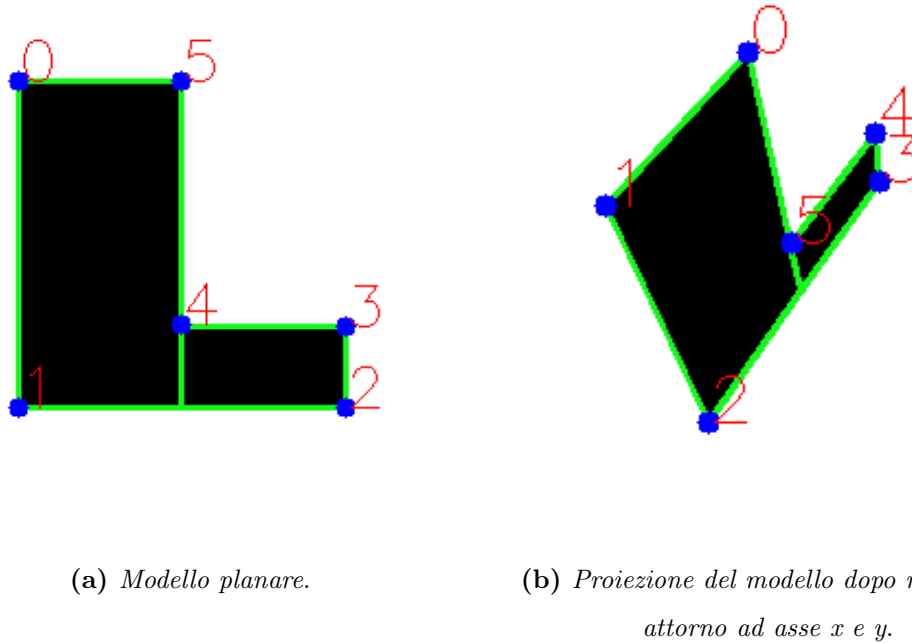
I descrittori geometrici utilizzati permettono di descrivere esclusivamente forme geometriche bidimensionali. In particolare, sono stati utilizzati i momenti invarianti (capitolo 3, paragrafo 2.3.3), vista la loro robustezza nel caso di rotazioni o variazioni di scala. Di conseguenza, il descrittore dell'oggetto viene costruito direttamente a partire dall'immagine acquisita e segmentata, senza trasformazioni di coordinate aggiuntive.

Nel caso di oggetti bidimensionali, il descrittore è già completo con i singoli momenti invarianti poiché questi permettono di riconoscere (senza stima della posa) l'oggetto all'interno di una scena indipendentemente dalla scala (zoom della telecamera), traslazioni e rotazioni dell'oggetto attorno all'asse  $Z$  (perpendicolare all'obiettivo) della telecamera. I problemi sorgono in presenza di (i) rotazioni più complesse che coinvolgono anche gli assi  $X$  e  $Y$  della telecamera e (ii) oggetti tridimensionali. Nel primo caso, infatti, la rotazione rende profondamente diversa la percezione dell'oggetto a causa dell'introduzione di distorsioni che si riflettono su variazioni del valore dei momenti geometrici che compongono il descrittore geometrico. Lo stesso effetto si ha con oggetti tridimensionali, ma questo caso è complicato dal fatto che le distorsioni compaiono per un numero maggiore di movimenti, come nel caso di traslazioni dell'oggetto lungo gli assi  $X$  e  $Y$  della telecamera.

In figura 6.15 è riportato un esempio di distorsione di un modello planare, nel dettaglio un oggetto a forma di "elle". Tale distorsione non viene introdotta da traslazioni, variazioni di scala o rotazioni attorno all'asse  $Z$ , ma unicamente da rotazioni attorno agli assi  $X$  e/o



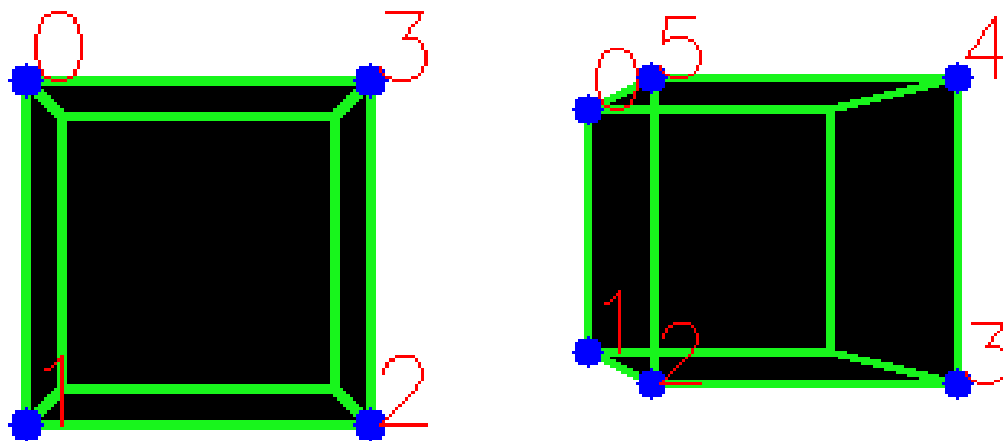
Y. La trattazione di questi casi particolari è dunque semplice perché limitata a due soli gradi di libertà di movimento della scena.



**Figura 6.15:** *Distorsione di un oggetto planare al variare della vista*

Al contrario, in presenza di oggetti tridimensionali, il problema si complica non poco. In figura 6.16 vengono mostrate le proiezioni di un cubo. La figura a sinistra mostra il modello del cubo visto dall'alto con l'asse della telecamera allineato con l'asse della faccia superiore del cubo. A destra si mette in evidenza come una semplice traslazione lungo l'asse  $X$  provoca una notevole variazione nelle feature osservate: oltre all'aumento dell'area occupata dalla proiezione, vengono infatti messi in vista più punti, in questo caso vertici, dell'oggetto. Per tali motivi, la modellazione dell'oggetto al variare dei punti di vista richiede un campionamento più fitto del numero di punti di vista e, quindi, porta alla creazione di un descrittore più complesso.

La soluzione scelta prende spunto dall'idea di rappresentare una scena tridimensionale come un insieme di viste bidimensionali ([61], capitolo 12) e dal descrittore utilizzato nella trasformata di Hough generalizzata (paragrafo 3.2.1, capitolo 3). Ammesso di poter conoscere in ogni istante (i) la posizione relativa della telecamera rispetto alla scena tridimensionale o specificatamente rispetto all'oggetto d'interesse  $\mathbf{P} = \{x, y, z, \theta_x, \theta_y, \theta_z\} = \{\mathbf{T}_{\mathbf{O}}^{\text{cam}}\}$  in un apposito sistema di riferimento e (ii) il set di momenti invarianti  $\mathbf{M} = \{m_1, m_2, \dots, m_n\}$ , il descrittore risulta costituito da un database di  $n$  coppie  $\mathbf{R}_i = \{\mathbf{P}_i, \mathbf{M}_i\}$ . Il set di momenti

(a) *Modello planare.*(b) *Proiezione del modello dopo rotazione  
attorno ad asse  $x$  e  $y$ .***Figura 6.16:** *Distorsione di un oggetto tridimensionale*

viene detto anche *descrittore base*.

La numerosità delle relazioni dipende dalla complessità e tipologia del problema: nel caso di oggetto planare e che può ruotare solo attorno all'asse della telecamera,  $n = 1$ ; la complessità aumenta in caso di oggetto sempre planare, ma rotazioni possibili anche attorno ad assi  $X$  e  $Y$ ; infine, il numero di relazioni necessarie a descrivere l'oggetto è massimo nel caso di oggetti tridimensionali con tutte le rotazioni e traslazioni possibili. Il database può essere normalizzato scegliendo un intervallo di confidenza per ogni componente del descrittore base in modo da raggruppare insieme pose aventi descrittori molto simili tra loro.

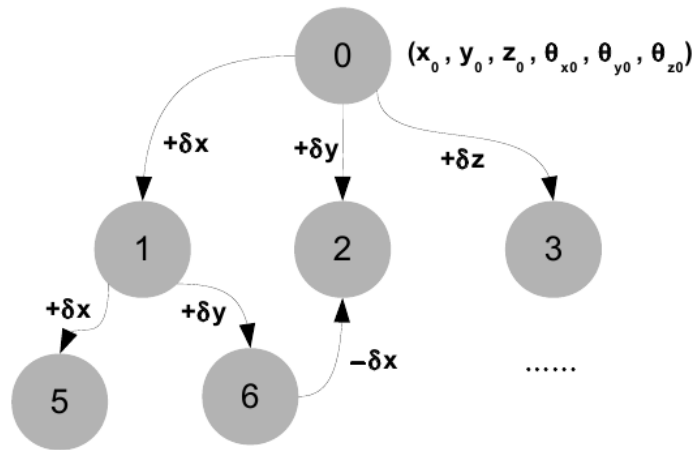
Il descrittore, infine, può essere completato da informazioni utili alla segmentazione delle scene in cui ricercare l'oggetto in questione, ovvero per separare l'oggetto da individuare dallo sfondo della scena. Per le assunzioni fatte, in particolar modo quella relativa all'avere a disposizione un ambiente strutturato, la segmentazione può essere tipicamente ridotta ad una semplice sogliatura dell'immagine (paragrafo 1.1.1, capitolo 3). In questo specifico caso il descrittore base viene completato dalle informazioni relative al piano colore su cui elaborare l'immagine (scala di grigi, RGB, HSV, ...) e alle soglie da applicare per separare l'oggetto dallo sfondo:  $\mathbf{M} = \{\text{piano colore}, T_1, T_2, T_3, m_1, m_2, \dots, m_n\}$ .

La formulazione del descrittore può essere riassunta dalla seguente:

$$\mathbf{D}_{\text{mvmbt}} = \{R_1, R_2, \dots, R_n\} = \{\{P_1, M_1\}, \{P_2, M_2\}, \dots, \{P_n, M_n\}\} . \quad (6.17)$$

Questo database può anche essere strutturato come un *grafo* i cui nodi corrispondono alle relazione posa-descrittore base mentre gli archi alla trasformazione geometrica da applicare all'oggetto telecamera per passare da una configurazione all'altra.

In figura 6.17 è riportato un estratto del grafo che costituisce il descrittore delle viste dell'oggetto. Il nodo radice è dato dal descrittore della vista dell'oggetto nella posa di partenza della telecamera. Ad ogni arco è associato lo spostamento della telecamera che provoca il passaggio ad un'altra vista. Il descrittore globale viene così costruito campionando tutte le posizioni assumibili da parte della telecamera. Si può notare come uno stesso nodo possa essere raggiunto seguendo percorsi differenti.



**Figura 6.17:** Grafo per la descrizione del modello 3D

Ora che il descrittore è stato definito, il passo successivo prevede la definizione di una procedura di *apprendimento* per la costruzione del modello da utilizzare per la ricerca dell'oggetto all'interno di una scena. L'apprendimento si basa sul mostrare al sistema di visione l'oggetto da tracciare da tutti i punti di vista d'interesse. Idealmente, quindi, avendo a disposizione l'oggetto e la telecamera, si dovrebbe muovere fisicamente la telecamera intorno all'oggetto e calcolare il descrittore base per ogni posa oggetto-telecamera desiderata. La difficoltà risiede nel fatto che per ogni punto di vista è necessario conoscere anche la posa relativa della telecamera rispetto al sistema di riferimento dell'oggetto, pertanto non è possibile eseguire l'operazione completamente "a mano". L'idea è però applicabile alle due situazioni seguenti:

- *Apprendimento online.* Telecamera montata sull'end-effector di un manipolatore in modo che la trasformazione geometrica necessaria a conoscere posizione e orientamento della telecamera nel sistema di riferimento del manipolatore è nota. In questo caso l'operazione di apprendimento viene eseguita situando l'oggetto in una specifica posizione dell'area di lavoro del manipolatore. L'end-effector del robot viene spostato attorno all'oggetto; la posizione della telecamera è calcolabile a partire dalla posizione dell'end-effector; il calcolo del descrittore base e l'associazione con la posa della telecamera vengono fatti online a partire direttamente dalle immagini acquisite;
- *Apprendimento offline da CAD.* È disponibile un modello CAD dell'oggetto da trattare. Il modello viene importato nel software per la creazione del descrittore. Si pone una *telecamera virtuale* in una posizione qualsiasi attorno all'oggetto e si applicano le trasformazioni base (2.5) per la stima della proiezione dell'oggetto sul piano immagine. L'immagine che si crea è un'immagine virtuale già segmentata e binarizzata che permette il calcolo del descrittore base a partire dall'oggetto proiettato. Si itera il procedimento ponendo la telecamera virtuale in un'altra posizione e così via per tutte le posizioni ritenute d'interesse. Per facilitare la proiezione dell'oggetto sulla scena, il modello deve contenere informazioni sui punti di interesse da proiettare, oltre che del volume. Ad esempio, nel caso di un solido geometrico, i punti d'interesse corrispondono ai vertici dell'oggetto. Il modello 3D dell'oggetto in questione deve quindi essere composto dalle facce che costituiscono la superficie esterna dell'oggetto geometrico e dai vertici che costituiscono le facce stesse. Questa caratteristica del modello favorisce l'utilizzo di file *CAD* perché essi memorizzano le informazioni riguardanti la geometria dell'oggetto sfruttando proprio il concetto di vertici e facce.

La procedura può essere quindi suddivisa nei seguenti passi:

1. Definizione di un sistema di riferimento globale per il calcolo della posa relativa tra telecamera e oggetto. È comodo porre l'origine del sistema di riferimento nel centro geometrico dell'oggetto da modellare;
2. Definizione di un set di pose d'interesse. Nel caso della procedura di apprendimento online le pose sono tutte quelle che può assumere il manipolatore durante lo svolgimento del task di posizionamento; nel caso della procedura di calibrazione virtuale, le pose vengono estratte definendo i limiti dell'area di lavoro della telecamera e i gradi di libertà con cui essa si può muovere. Il passo di campionamento tra una posa e la successiva influenza la numerosità del database e, di conseguenza, la complessità

computazionale e la precisione dell'algoritmo di tracciamento. La scelta del passo è il parametro da valutare con attenzione per ottenere i risultati di tracciamento desiderati. Gli unici parametri che non è possibile tarare con la procedura di posizionamento virtuale della telecamera sono quelli relativi alle soglie da applicare per la segmentazione delle immagini: l'unica soluzione è quella di acquisire un'immagine dal vivo della scena e tarare le soglie di conseguenza.

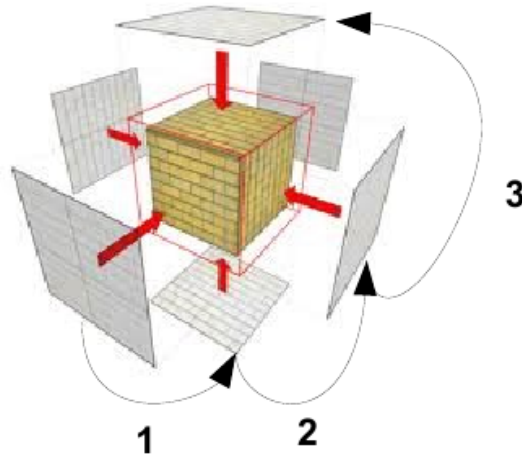
3. Iterazione del calcolo del descrittore base per tutte le pose scelte e memorizzazione della relazione posa-descrittore all'interno di un database.

In figura 6.18 viene esemplificata la modalità attraverso cui si arriva alla costruzione del descrittore a viste multiple. La prima immagine mostra che il modello è costruito spostando la telecamera attorno all'oggetto tridimensionale. Se il movimento della telecamera è virtuale, si sfrutta la conoscenza del modello tridimensionale dell'oggetto per ricostruirne le proiezioni sul piano immagine nelle varie viste. Il risultato del movimento della telecamera attorno all'oggetto è la costruzione di un grafo completo delle viste dell'oggetto stesso.

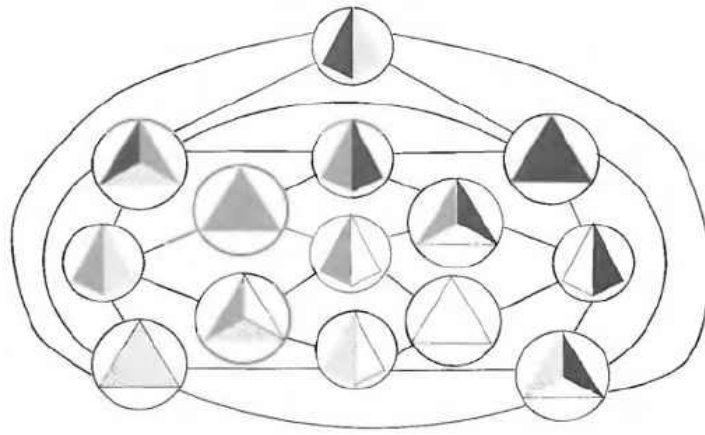
Fin qui si è parlato solo di descrittori per uno specifico oggetto, ma questo è più che sufficiente anche nel caso si debbano ricercare più oggetti all'interno di una stessa immagine. Infatti, è sufficiente costruire un descrittore per ogni classe di oggetti d'interesse ed iterare successivamente la procedura di ricerca per ogni classe di oggetti, tenendo conto che ogni classe di oggetti aggiunta va ad aumentare in modo importante la complessità computazionale dell'algoritmo di individuazione, ma in modo più lieve la complessità della fase di tracciamento.

### 3.2 Individuazione dell'oggetto in una scena

La fase di identificazione della presenza dell'oggetto desiderato all'interno di una scena e di calcolo della sua posa senza alcuna conoscenza a priori se non quella relativa al modello dell'oggetto da ricercare costituisce l'*inizializzazione* dell'algoritmo di tracking. Inoltre, come già accennato in precedenza, nel caso specifico di questo approccio questa procedura può essere utilizzata per recuperare il tracciamento di un oggetto nel caso di errori nella determinazione della posa nelle scene su cui viene applicata la procedura di tracciamento. Per come viene costruito il descrittore dell'oggetto, l'individuazione dell'oggetto target non permette di ottenere direttamente la posa dell'oggetto  $\mathbf{T}_{\text{obj}}$  all'interno del sistema di riferimento scelto, ma bensì la posa della telecamera rispetto all'oggetto  $\mathbf{T}_{\text{cam}}$ . L'informazione utile allo sviluppo di un algoritmo di controllo assistito da un sistema di visione



(a) Movimento della telecamera virtuale.



(b) Grafo delle viste dell'oggetto.

**Figura 6.18:** Costruzione del descrittore a viste multiple

è solitamente la posa dell'oggetto rispetto alla telecamera stessa. Se il modello è stato costruito facendo coincidere il sistema di riferimento del modello dell'oggetto col sistema di riferimento globale, la trasformazione è data dalla seguente equazione

$$T_{obj} = T_{cam}^{-1} \quad (6.18)$$

o in alternativa, nota la trasformazione tra sistema di riferimento globale e sistema di riferimento dell'oggetto nel modello  $\mathbf{T}_O^{obj_{mod}}$ , da:

$$T_{cam}^{obj} = (T_O^{cam})^{-1} \cdot T_O^{obj_{mod}}. \quad (6.19)$$

L'identificazione dell'oggetto all'interno di una scena tramite confronto con un modello segue i classici passi degli algoritmi di *pattern matching*:

1. *Segmentazione*. La separazione dell'oggetto dallo sfondo avviene sfruttando un semplice algoritmo di *blob search*. Le soglie utilizzate per la binarizzazione delle immagini sono

quelle ottenute tramite la procedura di apprendimento e memorizzate nel modello dell'oggetto da cercare;

2. *Individuazione presenza oggetto.* Prima di cercare di stimare la posa di un oggetto all'interno della scena, bisogna verificare se l'oggetto è presente o no nella scena. Subito dopo la determinazione dei blob presenti nell'immagine, un primo filtro prevede di eliminare blob con valore di area (in pixel) troppo piccolo. La comparsa di tali blob può essere infatti dovuta alla presenza di rumore nell'immagine. Nel caso in cui invece il blob scartato appartenga effettivamente all'oggetto da individuare, l'area troppo piccola potrebbe portare ad errori nella stima della posa dell'oggetto causata dalla povertà di informazioni a disposizione. Una volta scartati gli oggetti troppo piccoli, è necessario scorrere tutte le relazioni contenute nei descrittori degli oggetti da cercare (un descrittore per ogni oggetto, se si deve cercare più di una classe di oggetti all'interno della stessa immagine). Per ogni relazione e per ognuno degli  $n$  momenti invarianti contenuti nel descrittore si procede al calcolo di dell'errore di confronto tra il descrittore estratto dalla relazione contenuta nel modello  $\mathbf{M}^M$  e il descrittore ottenuto dall'immagine osservata  $\mathbf{M}^O$ :

$$\begin{cases} N_i^M = \|m_i^M\| \cdot \log(m_i^M) \\ N_i^O = \|m_i^O\| \cdot \log(m_i^O) \\ E_{match} = \sum_{i=1}^n \frac{N_i^M - N_i^O}{\max(N_i^M, N_i^O)} \end{cases} \quad (6.20)$$

Si noti l'utilizzo della scala logaritmica per far fronte alla presenza di momenti invarianti (dal quarto in poi) con valori dell'ordine di  $10^{-20}$ . A questo punto si marca l'oggetto come presente nella scena se tale errore è inferiore ad una certa soglia scelta a-priori  $T_{Em}$  (da scegliere inferiore a 1 per ottenere risultati soddisfacenti).

3. *Minimizzazione dell'errore algebrico.* La soluzione del problema algebrico rappresenta l'inizializzazione per l'algoritmo di risoluzione del problema geometrico. In questo caso, dato che l'informazione relativa ad una prima stima della posa relativa tra telecamera e oggetto è insita nel modello dell'oggetto, la soluzione è data dalla minimizzazione dell'errore di matching calcolato nella (6.20):

$$\mathbf{T}_O^{\text{cam}} \in \mathbf{D}_{\text{mvmbt}} : \min(E_{match}) \quad (6.21)$$

4. *Soluzione del problema geometrico.* La minimizzazione dell'errore algebrico ha permesso di trovare la trasformazione  $\mathbf{T}_O^{\text{cam}}$  che minimizza l'errore di matching, ma

tale soluzione appartiene comunque ad un set limitato di pose. La correttezza della soluzione dipende pertanto dal passo scelto nella costruzione del modello dell'oggetto. La soluzione del problema geometrico consiste nell'affinamento di tale risultato tramite due passi distinti: (i) proiezione del modello sull'immagine utilizzando la trasformazione ottenuta con la soluzione del problema algebrico e (ii) correzione della trasformazione geometrica grazie all'analisi delle corrispondenze tra punti del modello e punti osservati.

Per la proiezione dei punti del modello sull'immagine delle osservazioni è necessario conoscere i parametri intrinseci ed estrinseci (cfr. paragrafo 2, capitolo 2) sia della telecamera (virtuale o no)  $\mathbf{P}_I^M = \{f_x^M, f_y^M, c_x^M, c_y^M\}$  utilizzata per la costruzione del modello che della telecamera utilizzata per il tracciamento dell'oggetto sulla scena reale  $\mathbf{P}_I^O = \{f_x^O, f_y^O, c_x^O, c_y^O\}$ . Per snellire la trattazione si assume che i pixel  $(u, v)$  delle equazioni seguenti siano già stati privati delle distorsioni.

Considerando per il momento nullo l'errore geometrico, ovvero posa della telecamera del modello coincidente con la posa della telecamera delle osservazioni e quindi coordinate dei punti dell'oggetto coincidenti ( $\mathbf{X}^O = \mathbf{X}^M$ ;  $\mathbf{X} = [X; Y; Z]$ ), dall'equazione della trasformazione prospettica (2.5) da coordinate spaziali a coordinate immagini è possibile scrivere, per le coordinate  $u$ , il sistema

$$\begin{cases} u^M = f_x^M \cdot \frac{X}{Z} + c_x^M \\ u^O = f_x^O \cdot \frac{X}{Z} + c_x^O \end{cases} \quad (6.22)$$

estendibile allo stesso modo per la coordinata  $v$ . Ricavando  $\frac{X}{Z}$  dalla prima equazione e sostituendolo nella seconda, si ottiene la proiezione del modello sull'immagine delle osservazioni:

$$u^{M \rightarrow O} = \frac{f_x^O}{f_x^M} \cdot (u^M - c_x^M) + c_x^O. \quad (6.23)$$

La proiezione del modello nel sistema di riferimento delle osservazioni deve esplorare tutte le possibili pose che potrebbe assumere l'oggetto durante il tracciamento e deve quindi coprire tutte le trasformazioni geometriche che non vengono inserite direttamente nel descrittore, perché contenenti informazione già presente nel descrittore di altre pose. Tra queste trasformazioni si possono enumerare: (i) rotazioni  $\theta_z$  attorno al centro geometrico della proiezione; (ii) scalatura  $S$  del modello; (iii) riflessione  $Rf$  ( $Rf = 1$  o  $Rf = -1$ ) della proiezione attorno al suo asse di simmetria (rotazione di  $180^\circ$  attorno ad un asse perpendicolare all'asse della telecamera). Inoltre il modello



deve essere perfettamente sovrapposto all'oggetto osservato e quindi la proiezione deve essere traslata di una quantità pari al centro geometrico dell'oggetto osservato ( $\bar{\mathbf{U}} = (\bar{u}, \bar{v})$ ). La correzione delle coordinate immagine ( $u_p^M, v_p^M$ ) rappresentante la proiezione di un punto del modello sull'immagine delle osservazioni è data da

$$\begin{cases} u_p^M = \bar{u}^O + S \cdot Rf_u \cdot ((u^M - \bar{u}^M) \cdot \cos \theta - (v^M - \bar{v}^M) \cdot \sin \theta) \\ v_p^M = \bar{v}^O + S \cdot Rf_v \cdot ((u^M - \bar{u}^M) \cdot \sin \theta + (v^M - \bar{v}^M) \cdot \cos \theta) \end{cases} \quad (6.24)$$

Il fattore di scala è definito come radice quadrata del rapporto tra l'area del blob osservato e l'area del blob del modello. L'equazione (6.25) mostra come stimare la nuova distanza di un punto dall'obiettivo della telecamera utilizzando solo il fattore di scala.

$$Z^O = \frac{1}{S} \cdot \frac{f_x^O}{f_x^M} \cdot Z^M \quad , \quad S = \sqrt{\frac{A^O}{A^M}} \quad (6.25)$$

Stimato in questo modo il fattore di scala, rimangono da stimare il parametro di rotazione  $\theta$  e gli indici di riflessione  $Rf_x$  e  $Rf_y$ . Per la stima di tali parametri viene considerato un set limitato di  $n$  punti della proiezione del modello, ad esempio gli spigoli della proiezione del modello ( $\mathbf{P}$ ). Per ognuno di questi punti e per ognuno dei modelli proiettati con un diverso set di parametri, si cercano i punti corrispondenti sull'oggetto osservato ( $\mathbf{P}^O$ ) scegliendo i punti del contorno dell'oggetto che minimizzano la distanza tra punto proiettato e punto osservato. Il set di parametri che meglio riproietta il modello sull'osservazione è quello che minimizza la somma degli errori della distanza tra corrispondenze, ovvero

$$\theta, Rf : \min \sum_{i=1}^n |P_i^M - P_i^O| \quad (6.26)$$

Ricorrendo alle tecniche di ricostruzione 3D presentate nel paragrafo 3.3 del capitolo 3 si riesce a stimare la trasformazione omografica  $\mathbf{H}_M^O$  che permette di proiettare i punti del modello sul piano immagine dell'osservazione:

$$\mathbf{U}^O = \mathbf{H}_M^O \cdot \mathbf{U}^M \quad (6.27)$$

dove l'omografia è una matrice  $3 \times 3$  così composta:

$$\mathbf{H} = s \cdot \mathbf{M} \cdot \mathbf{R} \quad ; \quad \mathbf{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad ; \quad \mathbf{R} = [r_1 | r_2 | t]$$

mentre  $\mathbf{U}$  è il vettore  $3 \times 1$  delle coordinate di un punto nel piano immagine:

$$\mathbf{U} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}.$$

Riscrivendo il sistema di equazioni (2.5) in forma matriciale

$$\mathbf{U} = \frac{1}{Z} \cdot \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{X}$$

con

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

con le opportune sostituzioni si ottiene la soluzione al problema geometrico:

$$X^O = \left( \frac{Z^O}{Z^M} \cdot (M^O)^{-1} \cdot H_M^O \cdot M^M \right) \cdot X^M \quad (6.28)$$

dove la trasformazione geometrica corretta è data dalla (6.29). Si noti che le matrici dei parametri intrinseci  $\mathbf{M}^O$  e  $\mathbf{M}^M$  sono uguali dal momento che le corrispondenze modello-oggetto sono state calcolate col modello proiettato sullo stesso piano immagine su cui si è osservato l'oggetto.

$$T_{corr_{geom}} = \frac{Z^O}{Z^M} \cdot (M^O)^{-1} \cdot H_M^O \cdot M^M. \quad (6.29)$$

Infine, se si è interessati alla posa relativa dell'oggetto rispetto alla telecamera, questa è data dalla seguente:

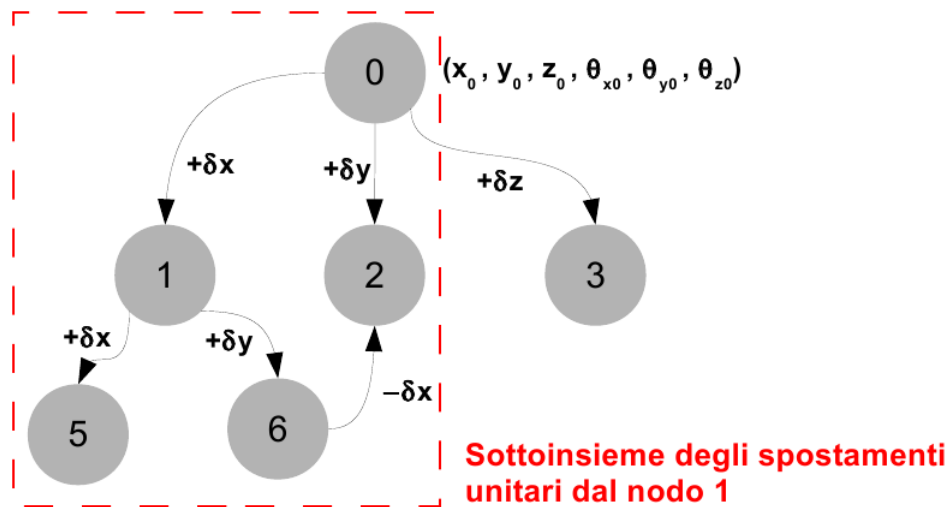
$$X^{camCorr} = T_{corr_{geom}} \cdot (H_{cam})^{-1} \cdot X^O. \quad (6.30)$$

### 3.3 Tracciamento

Come accennato in precedenza, questo tipo di approccio al problema del tracking prevede che la fase di tracciamento vera e propria sia solo un sottocaso del problema dell'identificazione. In questo caso particolare, infatti, il calcolo della posa passa esattamente per gli stessi passi da seguire per l'individuazione dell'oggetto senza informazioni a priori.

Ad ogni modo, come in ogni buon algoritmo di tracking che si rispetti, la fase di tracciamento puro deve essere resa più veloce e computazionalmente meno pesante sfruttando l'informazione calcolata al precedente passo d'acquisizione.

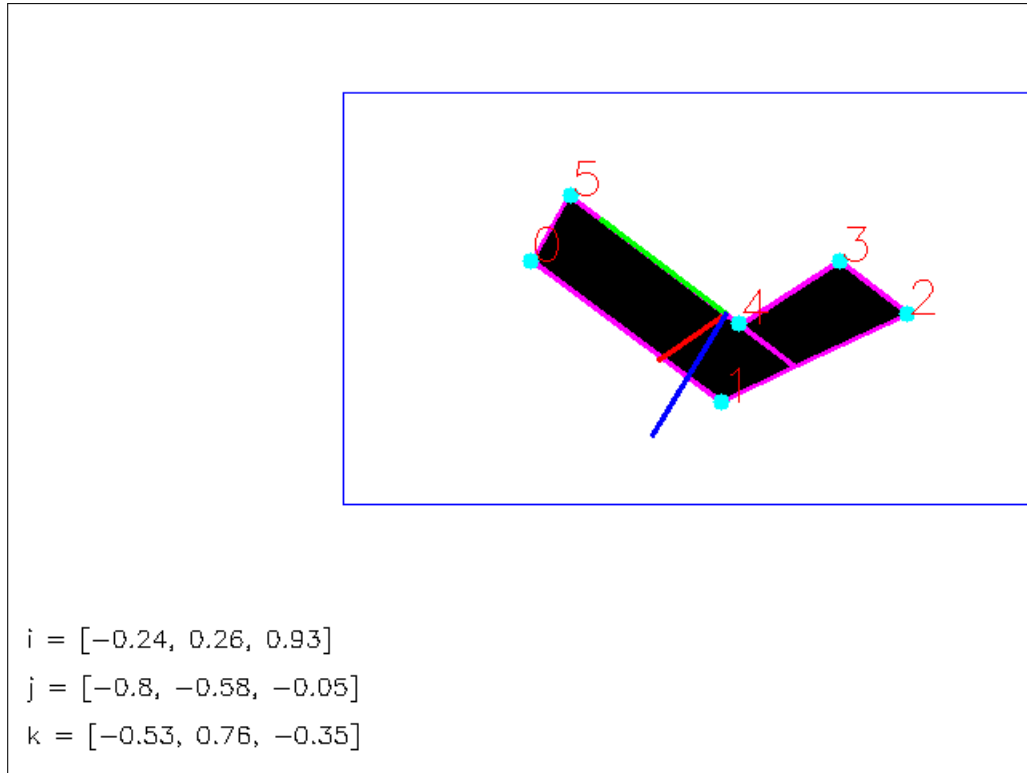
Per rendere meno onerosa l'elaborazione complessiva preservandone la robustezza, ci si basa su assunzioni simili a quanto detto per l'algoritmo di tracciamento Lukas-Kanade tracker (paragrafo 2.2). Considerando trascurabile o comunque bassa la distanza temporale tra l'acquisizione di due immagini successive, è possibile limitare l'ampiezza di ricerca sul grafo costruito a partire dal modello utilizzando l'informazione sulla posa calcolata al passo precedente. Come primo passo per la stima della posa, infatti, viene individuato il nodo che minimizza la distanza tra posa memorizzata nel database e posa stimata e limitare quindi la ricerca ai nodi raggiungibili attraverso un numero limitato di trasformazioni. In figura 6.19 viene mostrato come la ricerca parta da un nodo (nell'esempio il nodo 1) e interessi un numero limitato di nodi, ovvero i nodi del vicinato raggiungibili attraverso uno spostamento di un unico passo di campionamento.



*Figura 6.19: Sottoinsieme del grafo del descrittore considerato per il tracciamento dell'oggetto*

Sempre basandosi sull'assunzione di distanza temporale bassa tra due immagini successive, è possibile ridurre l'onere computazionale anche nella fase di segmentazione dell'immagine. Infatti, se l'oggetto si muove poco tra un frame e il successivo, è possibile ridurre la ricerca sull'immagine ad una finestra centrata sul centro geometrico dell'oggetto all'osservazione precedente e leggermente incrementata rispetto alla finestra che contiene tutte le feature dell'oggetto osservato. Questo approccio sfrutta le tecniche di finestramento già descritte nel capitolo 3, paragrafo 4.

La figura 6.20 mostra un'esemplificazione del concetto appena descritto: si ipotizza che nel prossimo frame l'oggetto subisca uno spostamento minimale e possa quindi essere individuato nella regione che circoscrive la proiezione corrente.



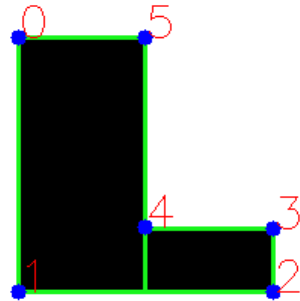
**Figura 6.20:** Limitazione della ricerca in una porzione dell'immagine

Infine, dalla posa degli oggetti in due frame successivi, è possibile calcolare di quanto si è spostato l'oggetto. L'omografia può essere scomposta nelle matrici di rotazione e traslazione tramite scomposizione ai valori singolari (paragrafo 3.3.2, capitolo 3) per ottenere il vettore degli spostamenti relativi  $\mathbf{X}_{rel} = [x_{rel}; y_{rel}; z_{rel}; \theta_{rel}^x; \theta_{rel}^y; \theta_{rel}^z]$ . La conoscenza dello spostamento dell'oggetto in ogni singola direzione permette di filtrare risultati derivati da errori di tracciamento o errori di stima della posa dell'oggetto. Per recuperare un errore di tracciamento è possibile ricorrere all'algoritmo di individuazione dell'oggetto con ricerca esaustiva sul database completo del descrittore.

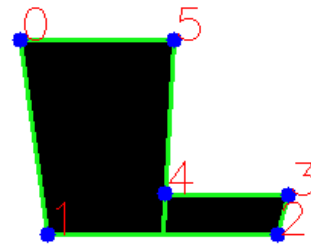
### 3.4 Simulazioni

Prima di passare all'applicazione dell'algoritmo in un caso reale, sono state eseguite alcune simulazioni al fine di validare l'approccio presentato. Come casi d'uso sono state considerati i due oggetti già introdotti nel corso di questo paragrafo: un oggetto a forma di "elle" per il caso planare e un cubo per il caso tridimensionale.

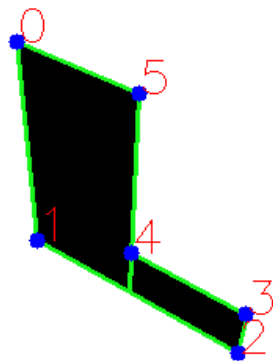
Per quanto riguarda l'oggetto planare, i modelli d'interesse sono quelli ottenuti a partire da rotazioni attorno agli assi  $X$  e  $Y$ , come mostrato in figura 6.21.



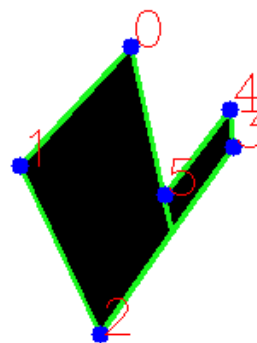
(a) *Modello di partenza.*



(b) *Modello sottoposto a rotazione attorno all'asse  $X$ .*



(c) *Modello sottoposto a rotazione attorno all'asse  $Y$ .*

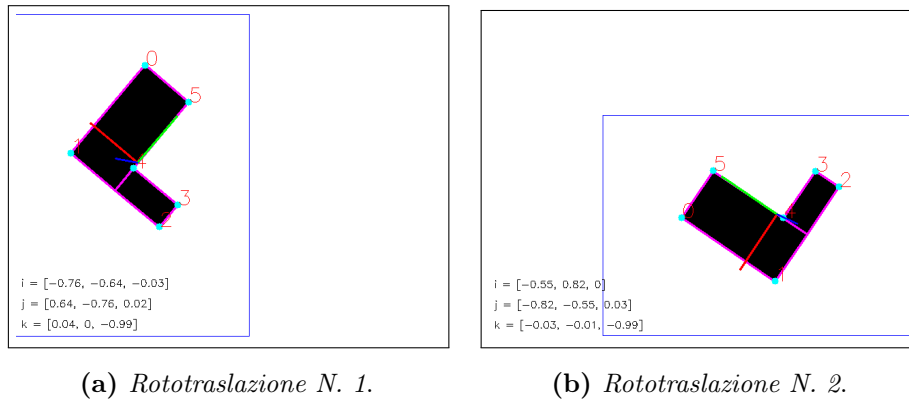


(d) *Modello sottoposto a rotazione attorno ai due assi.*

**Figura 6.21:** *Modelli per l'oggetto planare considerato*

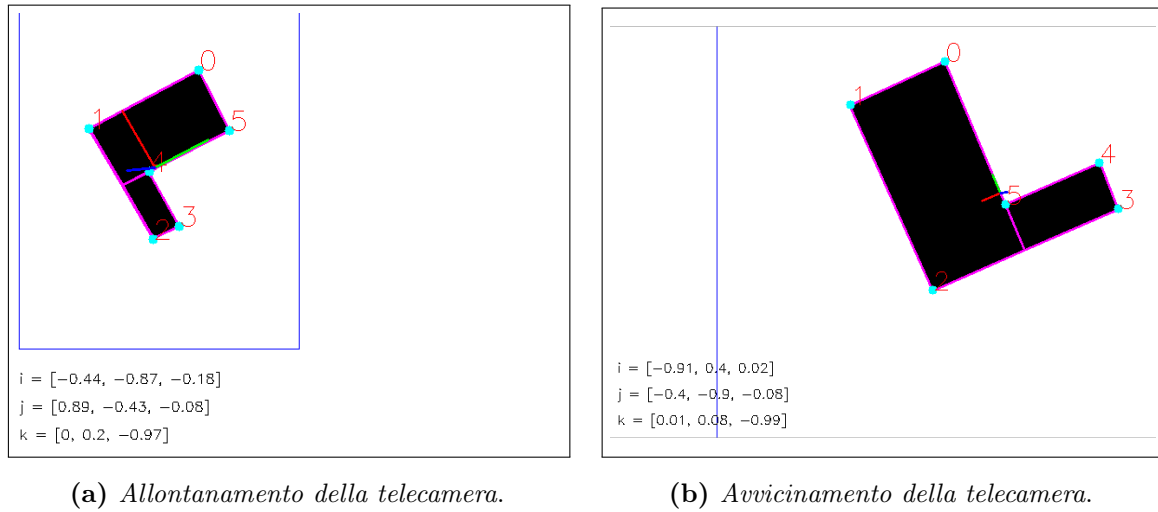
Questi, infatti, sono gli unici movimenti che portano a deformare la proiezione dell'oggetto. Si può notare che si tratta comunque di una deformazione limitata perché il numero di punti visualizzati non cambia. Qualora la deformazione sia tale da variare il numero di punti proiettati, il modello non risulterebbe più valido perché la proiezione sarebbe degenera, ovvero porterebbe tutti i punti ad essere proiettati verso un unico punto dell'immagine. Non è necessario analizzare e salvare nel modello le informazioni relative alle proiezioni ottenute con traslazioni (anche lungo l'asse  $Z$ ) o rotazioni attorno all'asse  $Z$  poiché, nel caso dell'oggetto planare, il riconoscimento può avvenire sfruttando l'invarianza dei momenti

geometrici a questo tipo di trasformazioni, come evidenziato dagli esempi riportati in figura 6.22. L'algoritmo di stima della scala  $S$  e della rotazione  $\theta_z$  è quindi sufficiente al calcolo della posa finale dell'oggetto. I numeri riportati a fianco dei vertici dell'oggetto indicano le corrispondenze trovate coi punti del modello, mentre il contorno magenta rappresenta la proiezione del contorno del modello di base sulla scena sottoposto alla trasformazione geometrica corrispondente alla posa stimata.



**Figura 6.22:** Invarianza a traslazioni e rotazioni attorno all'asse  $Z$  per il modello planare

In figura 6.23 è riportato il rilevamento dell'oggetto dopo traslazione della telecamera lungo l'asse  $Z$ .

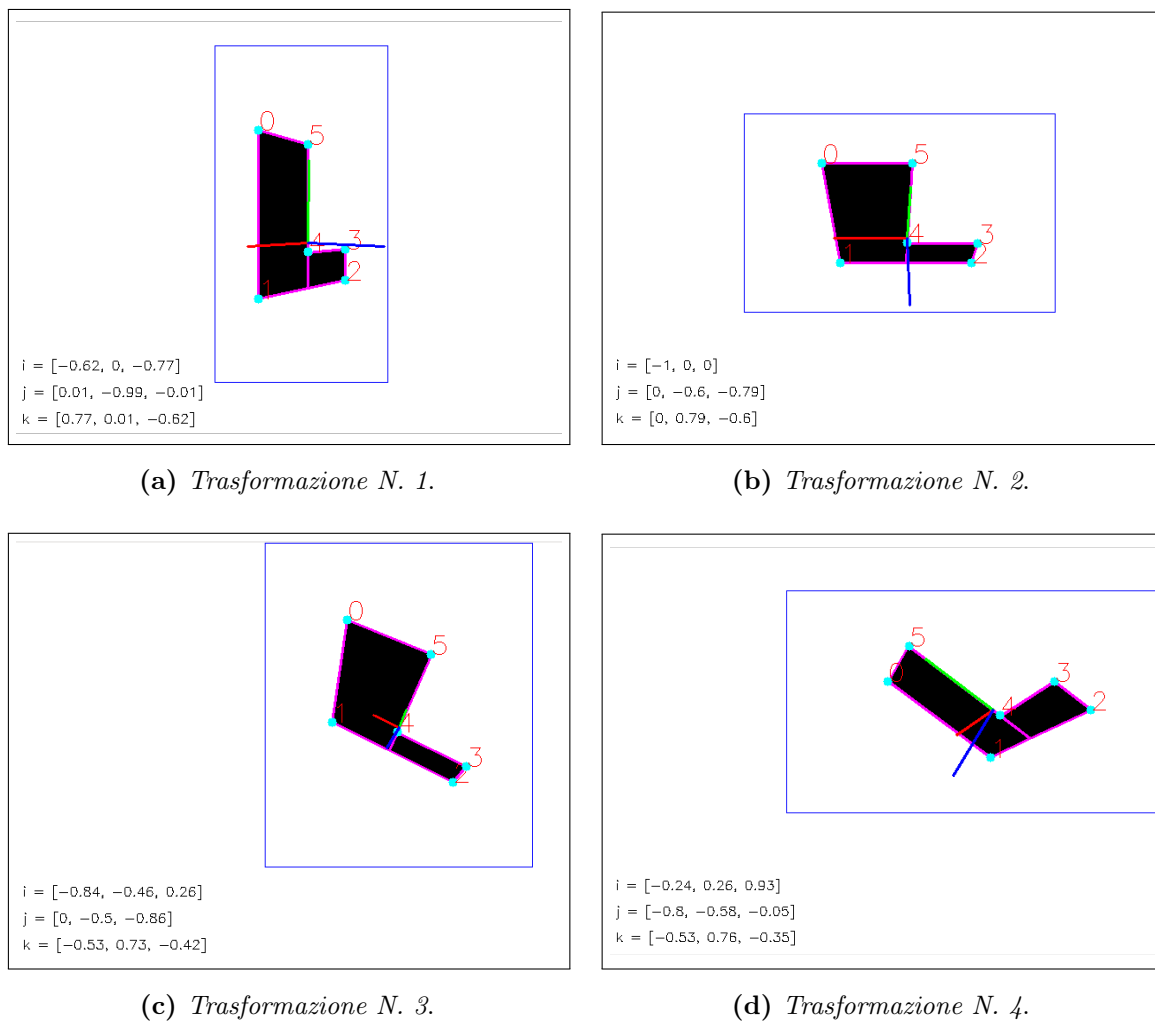


**Figura 6.23:** Riconoscimento dell'oggetto planare sottoposto a variazioni di scala

Si può notare come nel caso di allontanamento della telecamera il riconoscimento abbia successo senza troppi problemi; al contrario, nel caso dell'avvicinamento della telecamera, il riconoscimento soffre di un'inversione di alcune corrispondenze causato, nel caso specifico in questione, da un'inversione dei due lati della "elle". Tuttavia, l'errore viene mediato tramite la minimizzazione ai minimi quadrati utilizzata per il calcolo dell'omografia, facendo rientrare la stima della posa dell'oggetto (la linea rossa, corrispondente all'asse  $X$  è direzionata verso il lato lungo della "elle").

Infine, si può evidenziare come il riconoscimento della posa dell'oggetto sia robusto anche in presenza di trasformazioni geometriche più complesse, ovvero che coinvolgano anche rotazioni attorno agli assi  $X$  e  $Y$ , come evidenziato in figura 6.24.

L'algoritmo di riconoscimento dell'oggetto planare presenta la robustezza mostrata perché la fase di calcolo delle corrispondenze tra modello geometrico e proiezione osservata porta raramente all'insorgenza di errori. Ciò è dovuto al fatto che in tutte le viste modellate il numero di punti d'interesse proiettati non cambia. Inoltre, la dimensione del grafo costituente il modello non è particolarmente elevata, dal momento che non è necessario modellare le traslazioni perpendicolari all'asse ottico della telecamera e le rotazioni attorno a quest'asse. Per tali motivi, la fase di calcolo delle corrispondenze è sufficientemente immune ad errori poiché esso avviene in modo incrementale: ci si aspetta di trovare  $N$  corrispondenze dati  $N$  punti di modello; di conseguenza, una volta trovata una corrispondenza, si procede alla ricerca delle altre escludendo la corrispondenza già individuata.

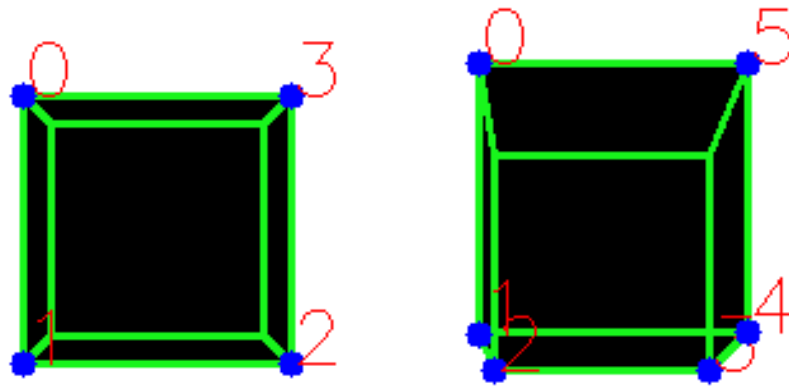


**Figura 6.24:** *Stima della posa dell'oggetto planare in presenza di trasformazioni geometriche più complesse*

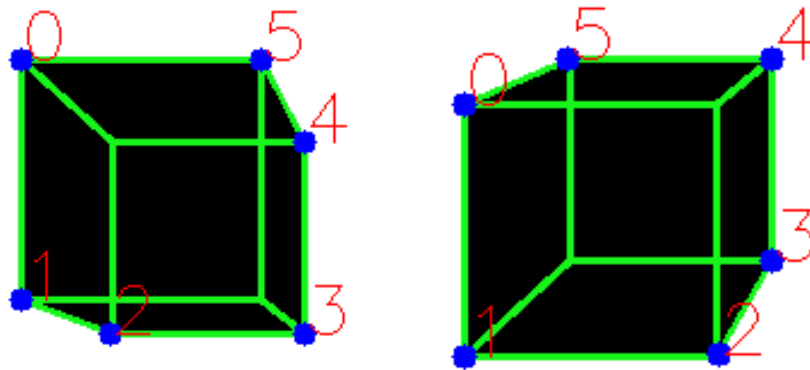
A causa del numero maggiore di deformazioni introducibile, queste considerazioni non si possono applicare al caso di riconoscimento di un oggetto tridimensionale. Questo concetto risulta subito evidente considerando la costruzione del modello delle proiezioni. In figura 6.25 viene mostrato come delle semplici traslazioni lungo gli assi  $X$  e  $Y$  del piano immagine possano portare a deformazioni importanti della geometria dell'oggetto. Non solo il numero di punti proiettati è variabile, ma anche le distanze tra i punti stessi e le relazioni geometriche tra gli spigoli del solido cambiano drasticamente da una proiezione all'altra.

L'alta variabilità e complessità del modello porta ad un aumento dell'onere computazionale dell'algoritmo e ad un aumento della percentuale di errore di stima della posa dell'oggetto. La scelta del cubo come oggetto di riferimento porta, inoltre, ad un'ulteriore considerazione legata alla simmetria del modello: è difficile distinguere una posa dalla stessa posa ruotata





(a) *Modello sottoposto a traslazione* N. 1. (b) *Modello sottoposto a traslazione* N. 2.

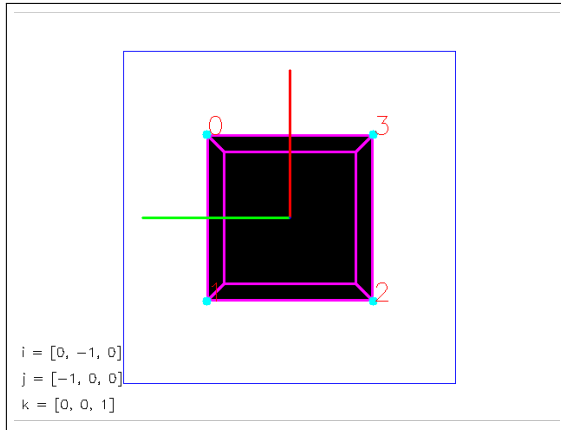


(c) *Modello sottoposto a traslazione* N. 3. (d) *Modello sottoposto a traslazione* N. 4.

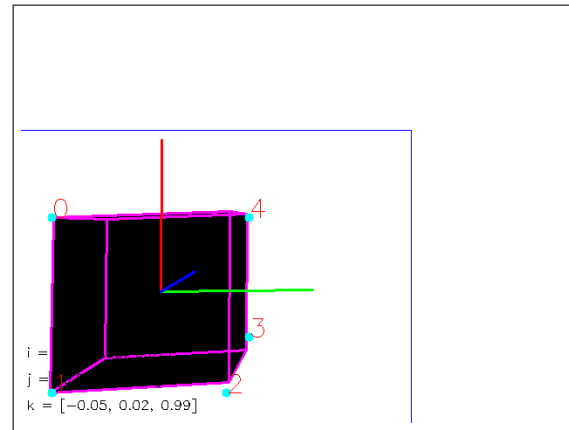
**Figura 6.25:** *Modelli per l'oggetto tridimensionale considerato*

di  $90^\circ$  o  $180^\circ$  e, pertanto, è necessario utilizzare ulteriori considerazioni per la stima della posa, ad esempio facendo affidamento sulla posa dell'oggetto al passo precedente. In figura 6.26 è riportato un esempio di questa situazione, evidenziata dal fatto che la terna del sistema di riferimento oggetto possa essere orientata in diversi modi nello spazio. La stessa figura mette in luce una modesta robustezza dell'algoritmo.

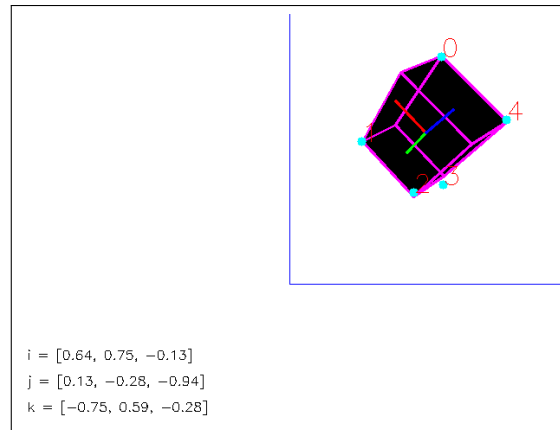
La figura 6.27 mette in evidenza l'insorgenza di possibili situazioni di errore nella stima della posa. In questo esempio, la posizione assunta dal cubo nello spazio non è stata correttamente modellata, portando al matching della proiezione con un modello ambiguo nel grafo del descrittore, in cui il cubo viene deformato da una serie di rotazioni e una scalatura dell'oggetto. L'errore è evidenziato dall'errato posizionamento dei punti del modello rispetto al contorno dell'oggetto.



(a) Riconoscimento dopo trasformazione geometrica N. 1.



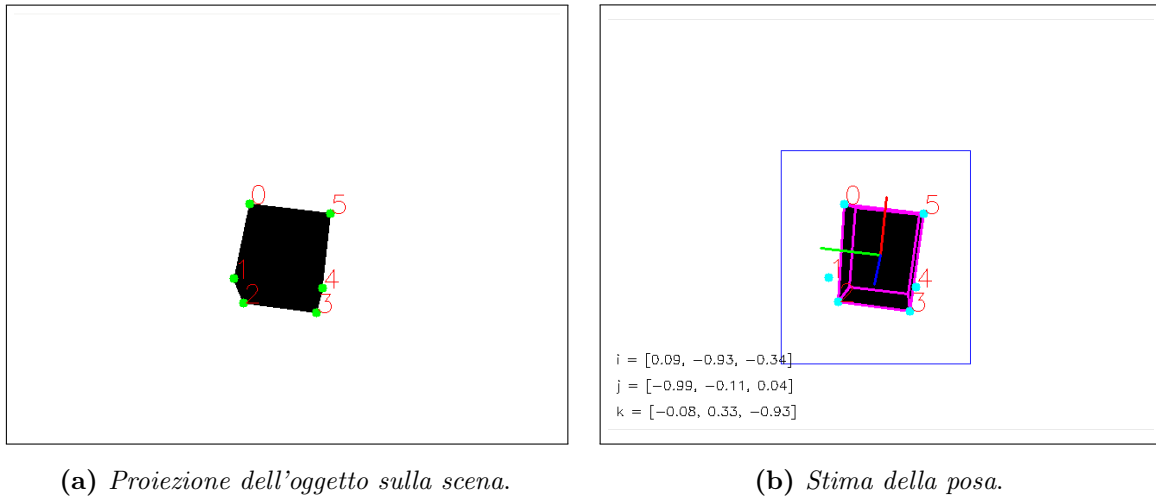
(b) Riconoscimento dopo trasformazione geometrica N. 2.



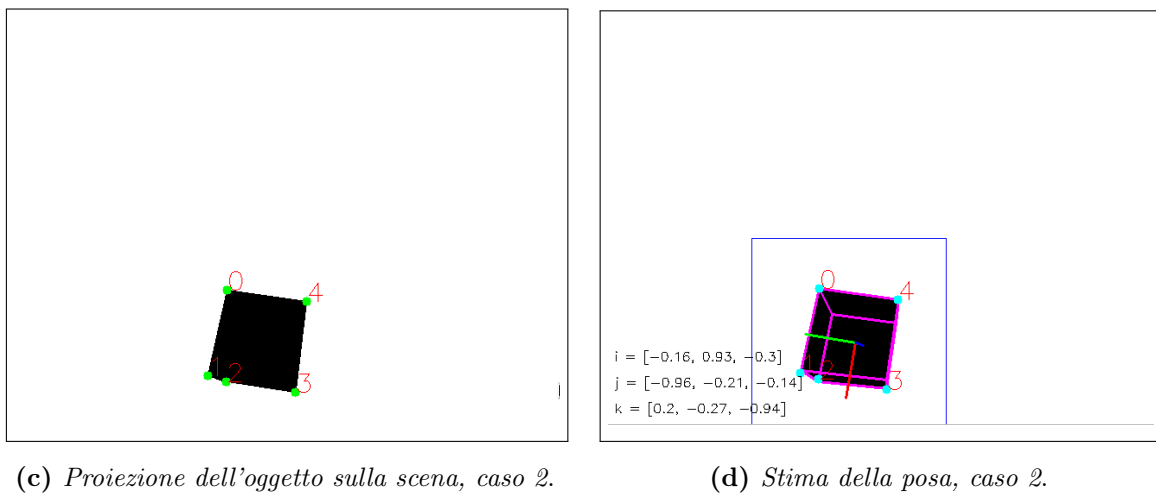
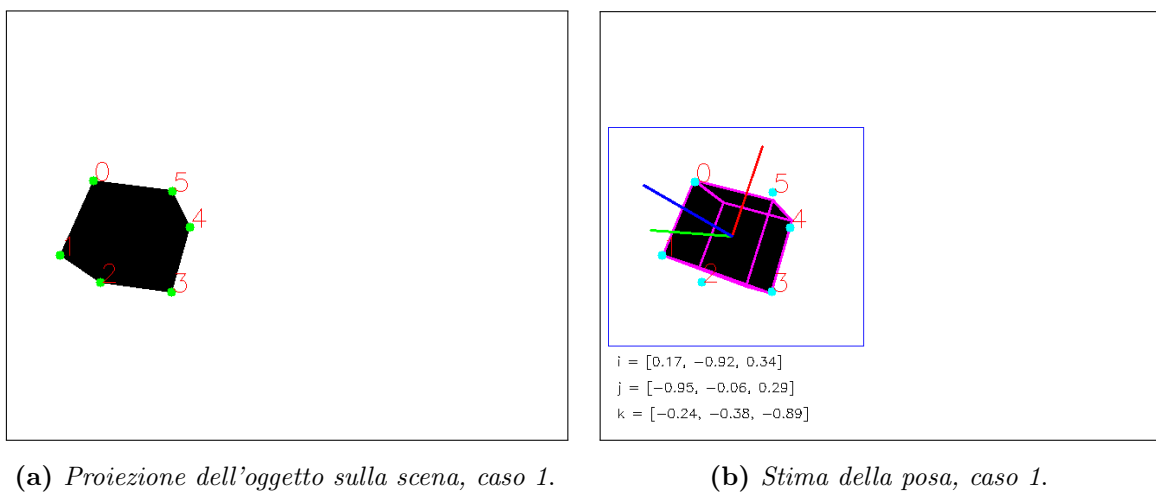
(c) Riconoscimento dopo trasformazione geometrica N. 3.

**Figura 6.26:** Riconoscimento dell'oggetto tridimensionale sottoposto a diverse trasformazioni e problema della simmetria

Quando l'oggetto tridimensionale torna ad assumere pose nello spazio vicine ad una posa correttamente inserita nel descrittore, l'algoritmo torna a dare risultati soddisfacenti, come mostrato in figura 6.28.



**Figura 6.27:** Errata stima della posa dell'oggetto tridimensionale



**Figura 6.28:** Stima corretta della posa per l'oggetto tridimensionale

Gli esempi di tracciamento appena mostrati mettono in evidenza come l'algoritmo sviluppato permetta di raggiungere risultati soddisfacenti per l'individuazione ed il tracciamento di oggetti planari o per i quali le deformazioni introdotte dal cambio del punto di vista non modificano in maniera eccessiva il numero di punti proiettati ed alcune relazioni geometriche tra i punti stessi. Questo avviene invece in presenza di oggetti tridimensionali, per i quali è quindi necessario introdurre ulteriori tecniche di filtraggio per riuscire a limitare o, nella migliore delle ipotesi, eliminare l'insorgenza di stime errate della posa dell'oggetto. Infine, risulta evidente che l'applicazione dell'algoritmo può essere favorita da alcune informazioni a priori riguardanti l'applicazione da sviluppare. Nell'analisi, infatti, si è cercato di verificare il comportamento dell'algoritmo in presenza di svariate tipologie di trasformazioni geometriche che hanno portato l'oggetto a posizionarsi in ogni modo possibile all'interno della scena. Tuttavia, per alcune applicazioni è possibile limitare il movimento della scena ad un set limitato di trasformazioni, rendendo quindi meno fitto il modello costruito, più facile la ricerca e limitata la possibilità di insorgenza di corrispondenze errate. Un esempio tipico nelle applicazioni del controllo in asservimento visivo è costituito dall'inseguimento di oggetti in movimento su un nastro trasportatore: in questo caso gli oggetti sono sottoposti unicamente a traslazioni lungo la direzione d'avanzamento del nastro con possibilità di rotazioni unicamente attorno all'asse perpendicolare al nastro. Ciò facilita enormemente l'applicazione dell'algoritmo, perché, come si è visto, le trasformazioni che creano maggiormente situazioni difficili sono quelle che coinvolgono rotazioni attorno agli assi  $X$  e  $Y$  del sistema di visione.

## 4 Conclusioni

Nel capitolo appena concluso sono state trattate le modalità e tecniche che permettono ad un sistema di visione di affrontare il problema del *moto degli oggetti* all'interno della scena oppure il *moto dell'intera scena* imputabile alla movimentazione del dispositivo di acquisizione immagini. Entrambi gli approcci trovano applicazione nell'implementazione di algoritmi di controllo in asservimento visivo, rispettivamente per la configurazione *eye-out-hand* o *eye-in-hand*.

L'obiettivo principale di tutti gli algoritmi presentati è quello di scomporre il compito di tracciamento in due fasi distinte: *inizializzazione* e *inseguimento*. La fase di inizializzazione fa uso delle tecniche di individuazione di caratteristiche della scena presentate nei capitoli 3 e 4. Tuttavia, tali algoritmi sono spesso onerosi dal punto di vista computazionale,

quindi inutilizzabili al fine di ottenere un'individuazione dell'oggetto e della sua posa in tempo reale. La fase di inseguimento prevede un'estensione di tali algoritmi, in modo da renderli utilizzabili in pratica, sotto certe assunzioni e condizioni, per un'individuazione degli oggetti più efficienti. La principale difficoltà nella scelta del corretto algoritmo di tracciamento risiede nella determinazione delle caratteristiche tracciabili e dalle condizioni dell'ambiente operativo, nonché dalla configurazione dell'hardware scelto (ad esempio il framerate del dispositivo d'acquisizione immagini).

Nell'ambito di questo lavoro di tesi si sono presentate due diverse opportunità: la prima soluzione consiste nell'individuazione di feature puntuali, mentre la seconda segue il filone degli algoritmi di ricerca basati sul confronto con modelli. Nell'ambito dell'utilizzo degli algoritmi di tracciamento per applicazioni di controllo *vision in the loop*, il metodo più comunemente utilizzato è una via di mezzo dei due approcci presentati, in cui l'individuazione e tracciamento delle feature viene utilizzato per il confronto con un modello e l'individuazione della posa dell'oggetto. Tuttavia, l'applicabilità di questo tipo di approccio dipende completamente dalla presenza di feature puntuali nell'oggetto da manipolare; tali feature sono spesso costituiti da punti della texture dell'oggetto, quali scritte e/o etichette. Al contrario, nell'ambito dell'automazione industriale si ha spesso a che fare con task di posizionamento basati sull'elaborazione di semplici oggetti bidimensionali, quali, ad esempio, fori. Pertanto, l'approccio portato avanti nella trattazione della tesi è il secondo, basato sull'analisi della geometria delle proiezioni degli oggetti inquadrati nella scena. Il metodo presentato si è mostrato robusto nel caso di oggetti bidimensionali, mentre nel caso di oggetti caratterizzati da una forte componente tridimensionale l'approccio ha presentato alcuni problemi di robustezza, non risolvibili senza prevedere un aumento della complessità computazionale dell'algoritmo presentato.



# Capitolo 7

## Approccio al Progetto di Regolatori in Asservimento Visivo

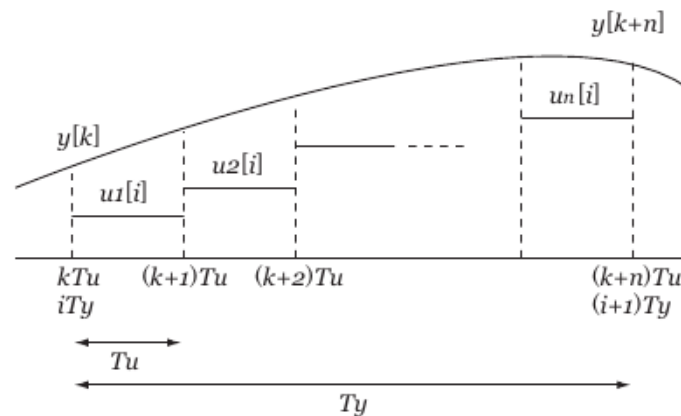
### Indice

---

1	Modello del controllo del motore DC . . . . .	<b>275</b>
1.1	Motore in corrente continua . . . . .	276
1.2	Controllo in cascata . . . . .	281
1.3	Controllo in coppia . . . . .	283
1.4	Controllo in velocità . . . . .	286
1.5	Controllo in posizione . . . . .	292
1.6	Schema Simulink del controllo del motore DC . . . . .	294
1.7	Simulazioni . . . . .	298
2	Dinamica del sistema di visione . . . . .	<b>302</b>
2.1	Ritardo d'elaborazione immagini . . . . .	303
2.2	Sistema digitale . . . . .	305
2.3	Altre non-linearità . . . . .	310
2.4	Funzione di trasferimento considerata . . . . .	312
3	Schemi di controllo in asservimento visivo . . . . .	<b>315</b>
3.1	Schema generale . . . . .	316
3.2	Funzione di posizionamento . . . . .	318
3.3	Funzione di inseguimento (tracking) . . . . .	319
3.4	Feedforward . . . . .	320
4	Progetto dei regolatori . . . . .	<b>323</b>
4.1	Funzione in anello aperto . . . . .	323
4.2	Controllo proporzionale . . . . .	324

4.3	Controllo proporzionale-derivativo . . . . .	325
4.4	Feedforward . . . . .	327
4.5	Funzioni in anello chiuso . . . . .	330
5	Simulazione asse lineare con controllo IBVS . . . . .	<b>331</b>
5.1	Posizionamento IBVS . . . . .	334
5.2	IBVS: inseguimento . . . . .	339
5.3	IBVS: inseguimento con feedforward . . . . .	342
6	Conclusioni . . . . .	<b>345</b>

Nel corso della trattazione di questo lavoro di tesi sono già state trattate alcune delle più importanti problematiche relative all'implementazione di controlli in asservimento visivo. In particolare, le due tematiche più note in letteratura riguardano la (i) scelta ed individuazione delle feature dell'immagine utilizzabili per la ricostruzione della posa della scena e la (ii) stabilità del controllo nel caso di controllori basati su immagini (IBVS). In realtà, la trattazione dei problemi di instabilità non può essere ritenuta completa senza un'analisi delle prestazioni dinamiche del sistema di controllo. Infatti, l'introduzione del dispositivo di visione nell'anello di retroazione del controllo origina tutta una serie di problematiche che possono essere riassunte sotto la denominazione di *controllo di un sistema multirate*. Questo termine viene utilizzato per classificare gli schemi di controllo in cui la frequenza operativa di alcune componenti del sistema è molto diversa da quella delle altre componenti. Nel caso dell'asservimento visivo, il sistema di elaborazione immagini introduce due componenti non lineari al controllo: un (i) ritardo nell'ottenimento dell'informazione e (ii) una frequenza di acquisizione ed elaborazione immagini che è generalmente un sottomultiplo della frequenza di elaborazione dei segnali acquisiti da altri sensori.



**Figura 7.1:** Frequenze di campionamento di un sistema multirate



Con riferimento alla figura 7.1, detto  $T_u$  il periodo di elaborazione dell'errore da parte del controllore e  $T_y$  il periodo di acquisizione delle immagini, con  $T_y > T_u$ , si può notare che esiste un periodo di tempo piuttosto ampio in cui il controllore si trova a dover generare il segnale di comando senza avere a disposizione nuove informazioni per il calcolo dell'errore. Questo porta inevitabilmente ad una diminuzione delle prestazioni dinamiche del sistema e all'introduzione di alcune limitazioni nella taratura dei parametri del controllore. Lo studio della dinamica del sistema di controllo permette di fornire gli strumenti per la definizione di tali limiti.

In letteratura, il problema del *controllo di sistemi multirate* è affrontato trattando l'intero sistema a tempo discreto fissato dalla frequenza operativa della telecamera ([18]) oppure utilizzando tecniche di controllo specifico ([47], [25] e [26]). Nel corso del lavoro di tesi si è cercato, invece, di modellare il sistema con le tecniche base per lo sviluppo dei controlli automatici ([5]). Dal punto di vista della sintesi del controllore in asservimento visivo sono stati analizzati diversi approcci, quali l'utilizzo di regolatori PID non lineari ([21]) o a struttura variabile ([42]), ma, per rimanere nell'ambito di una trattazione generale e non specifica, l'approccio seguito in questo lavoro è più simile a quello classico ([66]). L'analisi della dinamica del controllo in asservimento visivo è fondamentale per sviluppare un sistema in grado di inseguire il movimento di oggetti all'interno dell'area di lavoro del manipolatore ([56]).

In questo capitolo vengono presentati i dettagli relativi alla dinamica degli schemi di controllo in asservimento visivo. Innanzitutto, vengono completate le componenti, già accennate nel capitolo 5, relative al controllo del moto degli attuatori dei robot (1) e alla dinamica dei sistemi di acquisizione ed elaborazione immagini (2). In seguito viene mostrata l'integrazione di questi componenti nello schema di controllo in asservimento visivo generale, al fine di arrivare alla definizione delle funzioni di trasferimento analizzabili per lo studio e progettazione del controllo (3). Infine, il paragrafo 4 contiene alcune considerazioni relative alla scelta e parametrizzazione dei regolatori in visual servoing, che sfociano poi nella simulazione (5) del comportamento dinamico di un semplice sistema ad un grado di libertà. Come alla fine di ogni capitolo, si conclude con le considerazioni sulle tematiche svolte(6).

## 1 Modello del controllo del motore DC

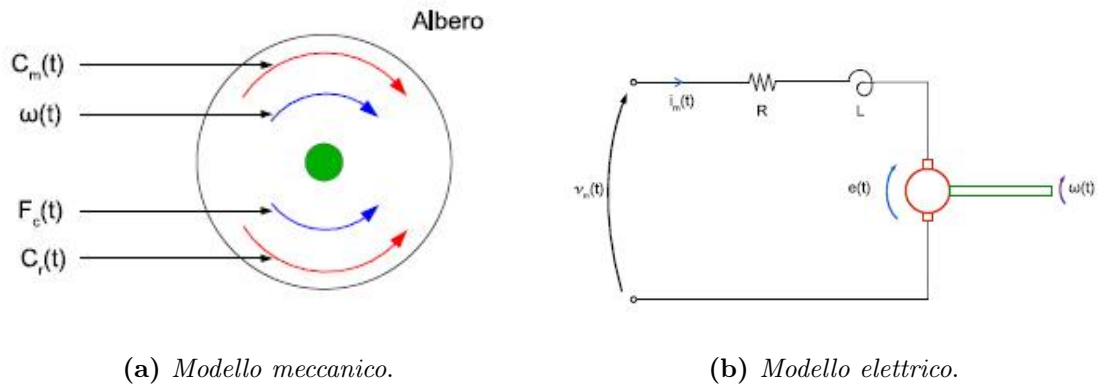
Il metodo di controllo dei robot a giunti indipendenti prevede che ognuno dei suoi giunti deve seguire una ben definita traiettoria per far sì che l'end-effector di un robot segua una

specifica traiettoria cartesiana

Il metodo di controllo dei motori in corrente continua descritto è noto in letteratura come *controllo in cascata* perché prevede la concatenazione di tre anelli di controllo diversi: l'anello più esterno si occupa di mantenere la posizione del motore, mentre gli anelli interni impostano la velocità e la coppia necessari al mantenimento o raggiungimento della posizione stessa. Il modello e le considerazioni presentati sono applicabili, con una certa approssimazione, anche al caso dei motori brushless.

## 1.1 Motore in corrente continua

Per analizzare il comportamento dinamico di un motore in corrente continua a magneti permanenti è possibile costruirne un modello in cui vengono messe in evidenza le relazioni esistenti tra le grandezze di comando del motore (*tensione e corrente d'alimentazione*) e le grandezze d'interesse (*velocità e coppia*). Per costruire tale modello è necessario analizzare i modelli elettrico e meccanico del motore stesso rappresentati in figura 7.2.



**Figura 7.2:** Modelli del motore in corrente continua

Le equazioni di bilancio elettrico e meccanico relative ai modelli in figura 7.2 costituiscono il sistema di equazioni (7.1);  $R$  e  $L$  sono la resistenza e l'induttanza concatenata,  $K_m$  è la costante di coppia (o costante meccanica del motore),  $K_e$  la costante elettrica,  $J$  l'inerzia del sistema ridotta all'albero del motore e  $V_{cem}$  è la cosiddetta forza contro elettromotrice, ovvero una tensione generata dal campo elettromagnetico derivante dalla rotazione del rotore attorno ai magneti del motore.

$$\begin{cases} V_a = i \cdot R + L \cdot \left( \frac{di}{dt} \right) + V_{cem} \\ V_{cem} = K_e \cdot \omega \\ C_m = J \cdot \dot{\omega} + C_r = K_m \cdot i \end{cases} \quad (7.1)$$

Per le simulazioni presentate in questo paragrafo viene preso come esempio il motore *SMB40-6000* prodotto da *Parker Motors* ([site11]), le cui caratteristiche sono elencate in tabella 7.1.

**Tabella 7.1:** Dati tecnici del motore *Parker SMB40-6000*

Coppia ad asse bloccato in S1 a 65°C	0.35	Nm
Coppia nominale in S1 a 65°C	0.21	Nm
Potenza nominale in S1 a 65°C	132	W
Velocità nominale	6000	rpm
Coppia di picco con duty cycle del 10% a 65°C	1.27	Nm
Costante di fem ( $\pm 10\%$ )	0.15	Vs
Costante di coppia ( $\pm 10\%$ )	0.26	Nm/A
Resistenza concatenata a 20°C ( $\pm 10\%$ )	8.8	$\Omega$
Induttanza concatenata media ( $\pm 10\%$ )	35	mH
Inerzia rotorica senza freno	0.0035	$10^{-3} \text{ kg m}^2$

La prima delle equazioni appena scritte contiene un termine differenziale, ovvero la corrente. Trasformando secondo Laplace queste stesse equazioni, si ottiene il sistema di equazioni lineari

$$\begin{cases} V_a(s) = R \cdot I(s) + s \cdot L \cdot I(s) + K_m \cdot W(s) \\ C_m(s) = s \cdot J \cdot W(s) + C_r(s) = K_m \cdot I(s) \end{cases} \quad (7.2)$$

da cui è possibile ricavare le funzioni di trasferimento tra velocità ottenibile e tensione d'armatura del motore.

Data la funzione di trasferimento tra tensione d'alimentazione e corrente

$$I = \frac{V_a - K_m W}{R + sL}$$

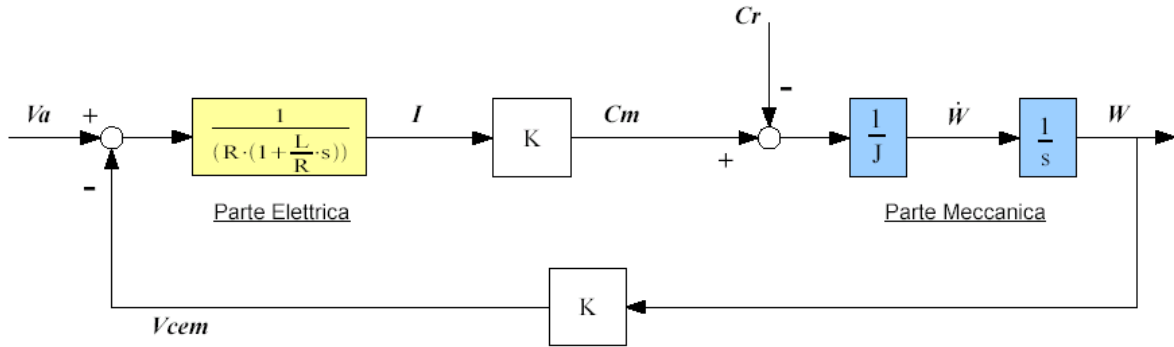
è possibile ricavare la trasformata della velocità  $W(s)$  ottenendo la relazione tra quest'ultima e la tensione di comando:

$$\begin{aligned} W &= \frac{C_m - C_r}{Js} = \frac{K_e I - C_r}{Js} = \frac{K_e}{sJ} \frac{V_a - K_m W}{R + sL} - \frac{C_r}{sJ} \\ &= \frac{K_e V_a - K_e K_m W}{sJ(R + sL)} - \frac{C_r}{sJ} \end{aligned} \quad (7.3)$$

e quindi

$$W \left( 1 + \frac{K_e K_m}{sJ(R + sL)} \right) = \frac{K_e}{sJ(R + sL)} V_a - \frac{C_r}{sJ} \quad (7.4)$$

Dalla (7.4) è possibile ricavare lo schema a blocchi della funzione di trasferimento del motore, come mostrato in figura 7.3.



**Figura 7.3:** Schema a blocchi della funzione di trasferimento del motore in corrente continua

Per studiare il comportamento dinamico del sistema così costruito, è necessario analizzare separatamente l'effetto dei due diversi ingressi: la tensione d'armatura  $V_a$  e la coppia resistente  $C_r$  sulla velocità di rotazione dell'albero del motore. In particolare, la tensione d'armatura agisce come segnale di comando, mentre la coppia resistente come disturbo.

### 1.1.1 Azione della tensione d'alimentazione

La funzione di trasferimento tra tensione d'alimentazione del motore e velocità in uscita è ottenuta trascurando l'effetto della coppia resistente nell'equazione (7.4):

$$\begin{aligned}
 \frac{W}{V_a} &= \frac{K_e}{sJ(R + sL)} \cdot \frac{sJ(R + sL)}{K_e K_m + sJ(R + sL)} \\
 &= \frac{K_e}{K_e K_m + sJ(R + sL)} \\
 &= \frac{\frac{1}{K_m}}{1 + \frac{sJ}{K_e K_m}(R + sL)}
 \end{aligned} \tag{7.5}$$

che è una funzione di trasferimento di un sistema del I ordine con due poli in

$$\begin{aligned}
 LJs^2 + RJs + K_e K_m &= 0 \\
 \Delta &= (RJ)^2 - 4LJK_e K_m = J(JR^2 - 4LK_e K_m) \\
 s_{1/2} &= \frac{-RJ \pm \sqrt{J(JR^2 - 4LK_e K_m)}}{2RJ}
 \end{aligned}$$

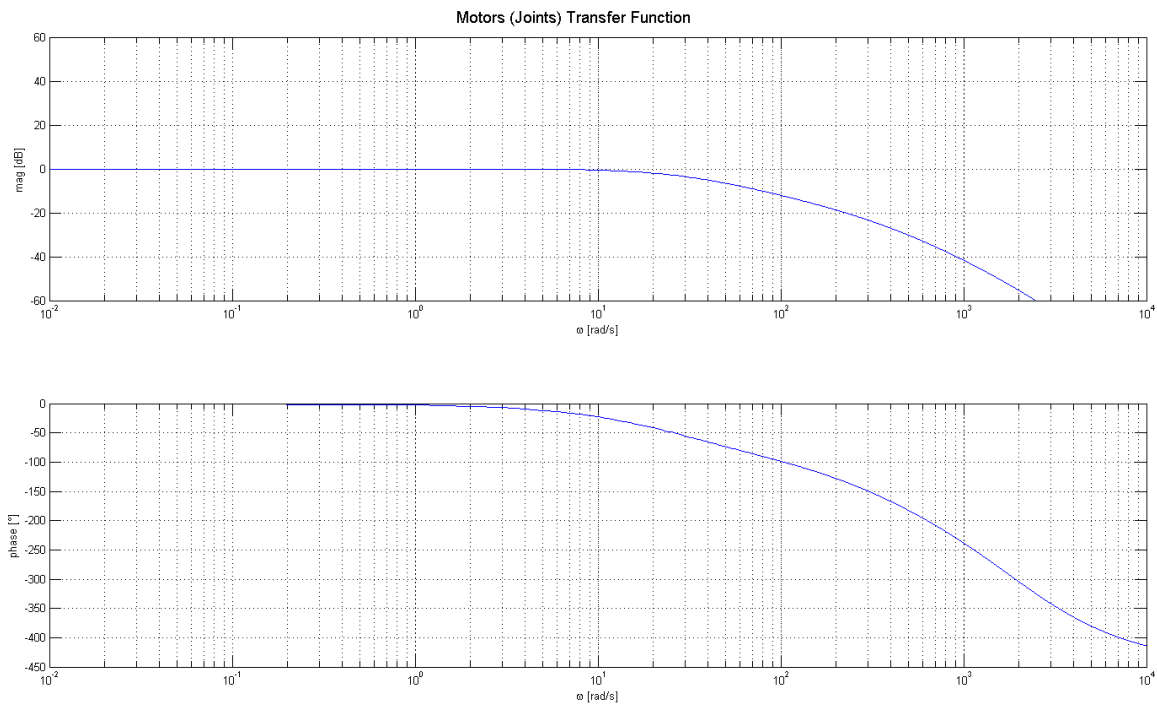
La costante di tempo fondamentale è pari all'inverso del polo dominante, quindi:

$$\tau_{mot} = \left| \frac{2RJ}{-RJ + \sqrt{J(JR^2 - 4LK_eK_m)}} \right|.$$

La risposta allo scalino ha guadagno  $\frac{1}{K_m}$  e presenta oscillazioni in presenza di poli complessi coniugati, ovvero

$$\Delta < 0 \quad J(R^2 - 4LK_eK_m) < 0 \quad \frac{L}{R} = \tau_e > \frac{1}{4} \cdot RJK_eK_m = \frac{1}{4}\tau_m \quad .$$

La risposta in frequenza è mostrata nel diagramma di Bode in figura 7.4.



**Figura 7.4:** Risposta in frequenza delle funzione di trasferimento del motore

### 1.1.2 Disturbo della coppia resistente

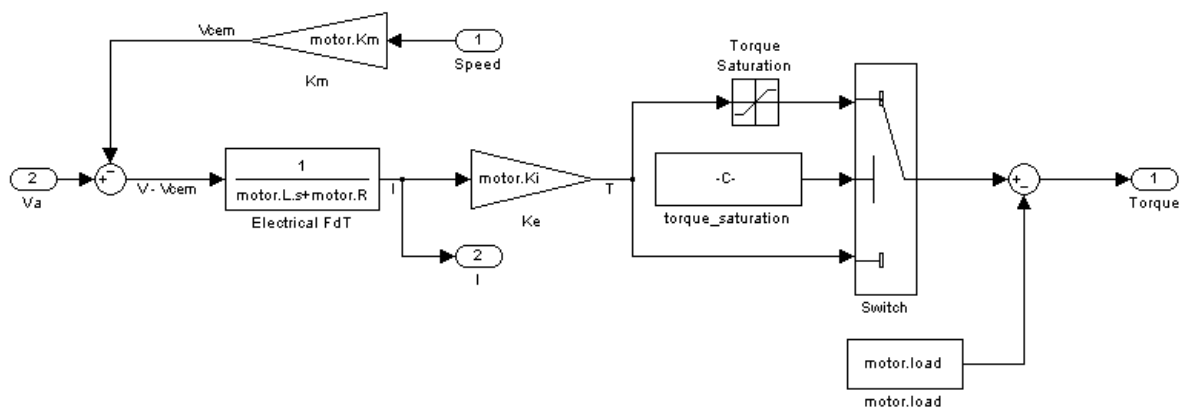
Analogamente a quanto fatto per la tensione d'alimentazione, per analizzare l'effetto del disturbo della coppia resistente è necessario porre  $V_a = 0$  e ricavare la funzione di trasferimento tra velocità e coppia resistente a partire dall'equazione (7.4):

$$\begin{aligned} \frac{W}{C_r} &= -\frac{1}{sJ} \cdot \frac{sJ(R + sL)}{K_eK_m + sJ(R + sL)} = -\frac{R + sL}{K_eK_m + sJ(R + sL)} \\ &= -\frac{\frac{R + sL}{K_eK_m}}{1 + \frac{sJ}{K_eK_m}(R + sL)} \end{aligned} \quad (7.6)$$

Tale funzione di trasferimento presenta gli stessi poli della funzione di trasferimento relativa all'analisi della tensione di alimentazione, con in aggiunta uno zero nella posizione  $s = -\frac{L}{R} = -\tau_e$ . La risposta allo scalino presenta la stessa dinamica della funzione di trasferimento analizzata precedentemente, con guadagno pari a  $\mu = \frac{R}{K_e K_m}$ .

### 1.1.3 Schema simulink

Il risultato delle considerazioni fin qui fatte è lo schema Simulink del motore in corrente continua, mostrato in figura 7.5.



**Figura 7.5:** Schema Simulink del motore a corrente continua

Nello schema Simulink del motore non viene modellata la funzione di trasferimento completa del motore (da tensione d'alimentazione a velocità), ma l'uscita è rappresentata dalla coppia del motore. Questa decisione è stata presa perché risulta più semplice modellare l'inerzia ridotta all'albero del motore all'esterno dello schema Simulink suddetto, ovvero nello schema dedicato al sistema meccanico completo.

La seconda nota riguarda la modellazione di un effetto non modellabile attraverso la rappresentazione tramite funzioni di trasferimento: la *saturation* della coppia in uscita. Infatti, il massimo valore di coppia erogabile dal motore è influenzato da limiti meccanici ed elettrici, di cui è possibile estrarre due componenti:

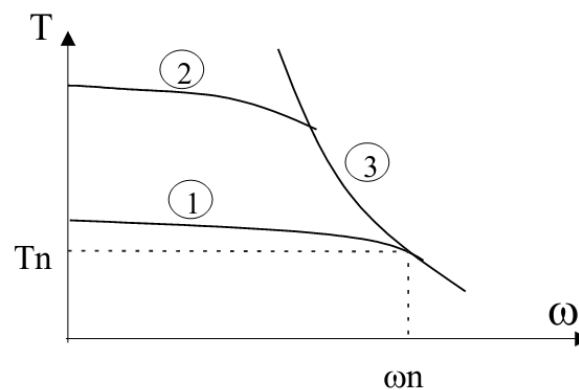
- La *coppia di picco*  $T_p$  è la massima coppia che può erogare il motore senza problemi, normalmente pari a  $2 \div 5$  volte la coppia in servizio continuo, cioè la coppia che il motore riesce ad erogare in modo continuativo mantenendo la temperatura entro valori accettabili e senza che il motore stesso si danneggi. Di solito la massima coppia di picco erogabile dal motore non viene mai utilizzata completamente, per non dover sovradimensionare il *driver*;

- La *coppia limite dovuta alla tensione massima di alimentazione dell'avvolgimento*.

Tale limite è imposto dalla forza controelettromotrice  $f_{cem} = K_m \omega$ .

Tale equazione mostra che la tensione indotta negli avvolgimenti dei motori è proporzionale alla velocità, quindi se aumenta  $\omega$  diminuisce la tensione a disposizione per imporre una data corrente e conseguentemente la relativa coppia, dato che parte della tensione disponibile deve compensare la tensione indotta  $f_{cem}$ .

I limiti di coppia sono facilmente visualizzabili nei grafici delle curve caratteristiche coppia-velocità dei motori, di cui la figura 7.6 mostra un esempio. In questa figura sono rappresentate, infatti, tre curve: la (1) coppia in servizio continuo, la (2) coppia di picco e il (3) limite di coppia dovuto alla tensione.



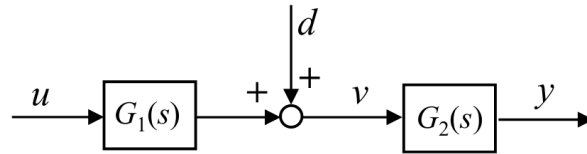
**Figura 7.6:** Curva caratteristica del motore

Tale non linearità verrà presa in considerazione successivamente per la progettazione dei regolatori del motore.

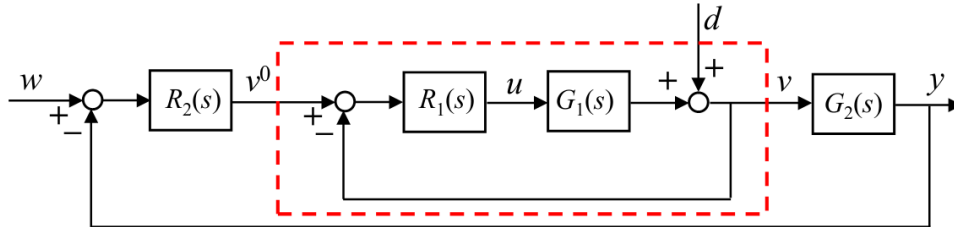
## 1.2 Controllo in cascata

La tecnica di *controllo in cascata* è molto utilizzata quando il sistema sotto controllo è costituito da due sottosistemi in serie descritti dalle funzioni di trasferimento  $G_1(s)$  e  $G_2(s)$ , come mostrato in figura 7.7. Si assuma poi che sull'uscita di  $G_1(s)$  agisca un disturbo  $d$  e che la variabile intermedia  $v$  sia misurabile. Si assuma infine che le prestazioni ottenibili nel progetto di un sistema di controllo solo per  $G_1(s)$  siano migliori di quelle raggiungibili con la sintesi di un regolatore per l'intero processo  $G_1(s)G_2(s)$ .

Sotto tali ipotesi è conveniente utilizzare lo schema di controllo in cascata di figura 7.8 perché consente di disaccoppiare il progetto del regolatore per il sottosistema più interno e veloce da quello del sottosistema più esterno e lento.



**Figura 7.7:** Processo composto da sottosistemi in serie

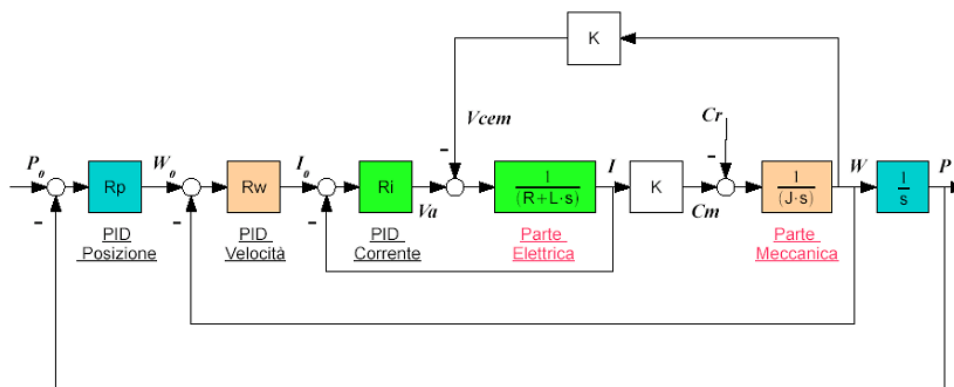


**Figura 7.8:** Controllo in cascata

Nel caso del motore in corrente continua ci si trova proprio nella situazione appena descritta. Infatti, il sistema motore può essere suddiviso in due parti distinte:

- *Parte elettrica.* Si occupa della “conversione” della tensione d’alimentazione in corrente e, conseguentemente, in coppia erogata dal motore;
- *Parte meccanica.* Converte la coppia in velocità.

La parte elettrica del motore è decisamente più veloce della sua controparte meccanica e, quindi, può rappresentare il sistema  $G_1(s)$ . La coppia resistente (dovuta al carico del motore) rappresenta il disturbo posto tra i due sistemi in serie. La coppia in uscita dalla parte elettrica, una volta sottratta la coppia dovuta al carico del motore, rappresenta l’ingresso per il secondo sistema in serie, ovvero la parte meccanica ( $G_2(s)$ ). Il tutto è riassunto in figura 7.9.



**Figura 7.9:** Schema a blocchi del controllo in cascata del motore in corrente continua

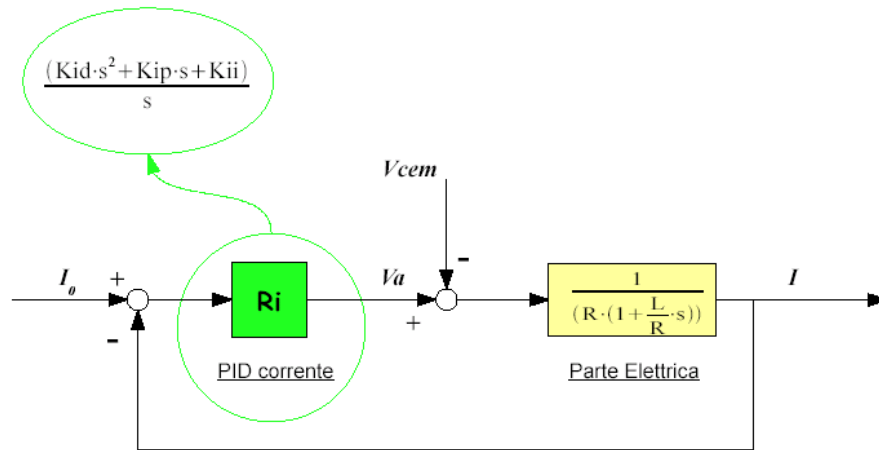


Solitamente, vista la semplicità delle funzioni di trasferimento coinvolte nel processo, vengono utilizzati controllori ad *azione PID* (*Proporzionale - Integrale - Derivativa*):

$$R_{PID}(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} . \quad (7.7)$$

### 1.3 Controllo in coppia

Il primo anello di retroazione da analizzare e progettare è quello più interno, cioè quello relativo al *controllo in corrente* (figura 7.10) o in coppia. Come si vede dallo schema a blocchi generale del controllo in cascata (figura 7.9), una caratteristica da tenere in considerazione per il progetto del controllore è sicuramente il disturbo provocato dalla forza controelettromotrice  $V_{cem}$ . Ovviamente, l'obiettivo del controllo è quello di minimizzare l'effetto di tale disturbo sulla grandezza da controllare, ovvero la corrente (o la coppia) del motore.



**Figura 7.10:** Schema a blocchi del controllo in coppia del motore in corrente continua

#### 1.3.1 Funzione di sensitività complementare

In un sistema di controllo in retroazione, la *funzione di sensitività complementare* mette in evidenza il rapporto tra il segnale di riferimento e quello d'uscita. Per progettare il regolatore, si studia la funzione di trasferimento in sola andata (senza regolatore):

$$\begin{aligned} L_I(s) &= \left( \frac{I}{I_0} \right)_{\text{anello}} = \frac{1}{R + sL} = \frac{1}{R \left( 1 + s \frac{L}{R} \right)} \\ L_T(s) &= \left( \frac{T}{I_0} \right)_{\text{anello}} = L_I(s) \cdot K_e = \frac{1}{R \left( 1 + s \frac{L}{R} \right)} \cdot K_e \end{aligned} \quad (7.8)$$

dove con  $I$  si indica la corrente e con  $T$  la coppia.

Nel caso del controllo in coppia si ha a che fare con un sistema del I ordine avente un polo in  $s = -\frac{R}{L}$ , sempre negativo, e guadagno  $\mu = \frac{1}{R}$ . Pertanto, il sistema risulta asintoticamente stabile.

Un buon progetto prevede la cancellazione del polo originario e l'inserimento di un nuovo polo nella posizione desiderata, per cui è sufficiente un regolatore PI, come mostrato nella (7.11).

$$\begin{aligned}
 L_I(s) &= R_I \cdot \frac{1}{R \left(1 + s \frac{L}{R}\right)} \\
 &= \frac{K_p^I s + K_i^I}{s} \cdot \frac{1}{R \left(1 + s \frac{L}{R}\right)} \\
 &= \frac{K_i^I \left(1 + s \frac{K_p^I}{K_i^I}\right)}{s} \cdot \frac{1}{R \left(1 + s \frac{L}{R}\right)} .
 \end{aligned} \tag{7.9}$$

Ponendo

$$\frac{K_p^I}{K_i^I} = \frac{L}{R}$$

tramite

$$\begin{cases} K_i^I = K^I \cdot R \\ K_p^I = K^I \cdot L \end{cases}$$

si ottengono le funzioni di trasferimento d'anello aperto:

$$\begin{aligned}
 L_I(s) &= \frac{K^I}{s} \\
 L_T(s) &= \frac{K^I \cdot K_e}{s} .
 \end{aligned} \tag{7.10}$$

Le funzioni in anello chiuso, invece, sono date dalle seguenti:

$$\begin{aligned}
 G_I(s) &= \frac{L_I(s)}{1 + L_I(s)} = \frac{K^I}{s + K^I} \\
 G_T(s) &= \frac{L_T(s)}{1 + L_T(s)} = \frac{K^I \cdot K_e}{s + K^I} .
 \end{aligned} \tag{7.11}$$

Si ottiene così un sistema del I ordine anche in retroazione, con guadagno unitario per quanto riguarda l'anello di corrente e guadagno pari alla costante elettrica del motore per

quanto riguarda il controllo della coppia. Il sistema è asintoticamente stabile se la seguente condizione è rispettata:

$$s = -K^I < 0 \quad , \quad \Leftrightarrow K^I > 0 \quad .$$

La scelta del guadagno del regolatore influisce direttamente sulla costante di tempo del sistema in anello chiuso, la quale, per un sistema del I ordine, è legata al tempo di risposta del sistema ad un ingresso a scalino dalla relazione:

$$T_{salita} = 5 \cdot \tau \quad .$$

Pertanto, relativamente alla coppia (la grandezza meccanica d'interesse) si ottiene un sistema con costante di tempo principale pari a

$$\tau^I = \frac{1}{K^I} \quad , \quad \Rightarrow K^I = \frac{1}{\tau^I} = \frac{5}{T_s^I} \quad . \quad (7.12)$$

Solitamente la dinamica di questo anello di controllo è molto elevata, con  $\tau$  dell'ordine della costante di tempo della funzione in anello aperto, ovvero della costante di tempo elettrica  $\frac{L}{R}$  (in genere dell'ordine delle decine di millisecondi).

### 1.3.2 Funzione di sensitività

L'analisi della *funzione di sensitività* permette di valutare la capacità del controllo di ridurre gli effetti dei disturbi sul controllo stesso, in questo caso della forza contro elettromotrice. La funzione di trasferimento della sensitività è:

$$\frac{I}{V_{cem}} = - \frac{\left( \frac{1}{R + sL} \right)}{\left( 1 + \frac{K^I}{s} \right)} \quad . \quad (7.13)$$

Quando si tratta di analizzare l'influenza di disturbi, è interessante valutare il guadagno del sistema, ovvero l'influenza del disturbo a regime. Nel caso della forza contro elettromotrice si ha

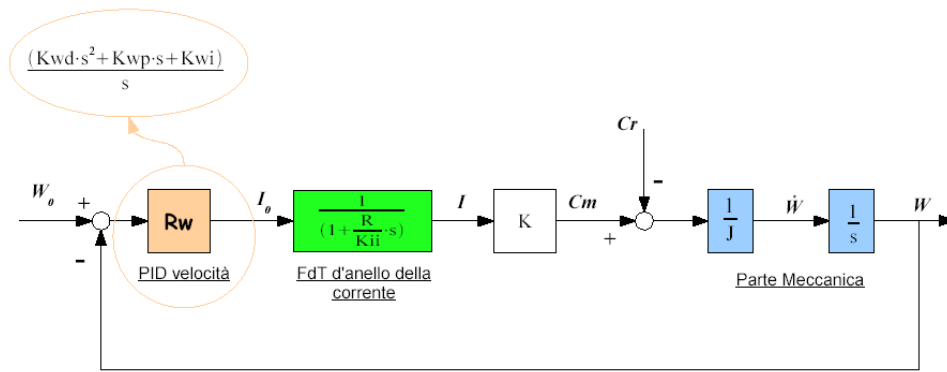
$$\mu_{V_{cem}} = \frac{1}{R K^I}$$

quindi maggiore è il guadagno del regolatore, minore è l'influenza della forza contro elettromotrice sul segnale in uscita.

## 1.4 Controllo in velocità

Attorno all'anello di controllo della corrente viene innestato l'anello di retroazione per il controllo in velocità del motore che risulta essere il controllo ideale per alcuni algoritmi di visual servoing.

Lo schema a blocchi utilizzato per le considerazioni relative al progetto del controllore è mostrato in figura 7.11. Per costruire tale schema, si sono dovute fare alcune semplificazioni e assunzioni; innanzitutto, l'anello interno di velocità è stato sostituito con la funzione in anello chiuso ottenuta precedentemente; in secondo luogo, non si è tenuto conto dell'effetto della forza controelettromotrice.



**Figura 7.11:** Schema a blocchi del controllo in velocità del motore in corrente continua

Come per il progetto dell'anello di corrente, si è iniziato considerando la funzione di trasferimento in anello aperto (7.14), dove è stato introdotto il coefficiente d'attrito viscoso  $B$  che va ad influire sulla conversione della coppia in velocità insieme all'inerzia  $J$ .

$$L_W(s) = \left( \frac{W}{W_0} \right)_{\text{anello}} = \frac{K^I}{s + K^I} \cdot \frac{K_e}{Js + B}. \quad (7.14)$$

Il sistema ha due poli in  $s_1 = -K^I$ ,  $s_2 = -\frac{B}{J}$  e guadagno  $\mu^W = \frac{K_e}{B}$ .

### 1.4.1 Regolatore PID: Saturazione della coppia

Tramite un regolatore PID è possibile cancellare entrambi i poli della funzione in anello aperto della corrente, cioè sia quello dovuto all'anello di corrente sia quello della meccanica.

Sia dato il regolatore

$$R_W(s) = \frac{K_d^W s^2 + K_p^W s + K_i^W}{s} = K^W \frac{(1 + sK_1^W)(1 + sK_2^W)}{s}$$

con

$$\begin{cases} K_d^W = K^W \cdot K_1^W \cdot K_2^W \\ K_i^W = K^W \\ K_p^W = K_1^W + K_2^W \end{cases}.$$

Ponendo

$$\begin{cases} K_1^W = \frac{1}{K^I} \\ K_2^W = \frac{J}{B} \end{cases}$$

si ottiene la funzione di trasferimento in anello aperto ideale

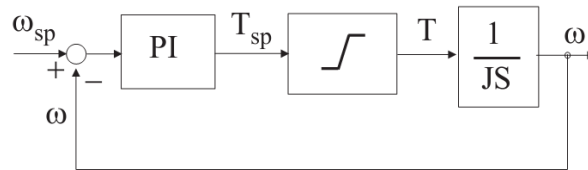
$$L_W^{PID}(s) = \frac{K^W K_e}{Bs} \quad (7.15)$$

da cui si ottiene la funzione di trasferimento in anello chiuso di un sistema del I ordine:

$$G_W^{PID} = \frac{K^W K_e}{Bs + K^W K_e}. \quad (7.16)$$

Il sistema è asintoticamente stabile se  $K^W > 0$  e la costante di tempo del sistema è pari a  $\tau^W = \frac{B}{K^W K_e}$ .

Le considerazioni, quindi, potrebbero essere uguali a quelle già fatte per l'anello di controllo della corrente, se non fosse che in questo caso subentra la non linearità legata alla saturazione della coppia erogabile dal motore (vedi paragrafo 1.1.3). Sotto l'effetto combinato della saturazione di coppia e del regolatore PI o PID (vedi figura 7.12) può verificarsi un ben noto funzionamento anomalo del sistema che va sotto il nome di *wind-up*.



**Figura 7.12:** Controllo in velocità: *wind-up*

Poniamo infatti che il riferimento di velocità  $\omega_{sp}$  presenti un gradino di ampiezza elevata, tale che il controllo dell'azionamento sia portato a richiedere, per ottenere l'inseguimento, un valore di coppia maggiore del massimo erogabile (saturazione). In tali condizioni l'errore di velocità  $e(t) = \omega_{sp} - \omega(t)$  rimane non nullo (e positivo) per un periodo piuttosto lungo, "caricando" la componente integrale del regolatore. Allo stesso modo, la presenza del derivatore porta il sistema, nello stato iniziale, a richiedere un contributo di coppia

molto elevato (proporzionale alla derivata dell'errore). Queste considerazioni rendono di fatto inutilizzabile il controllore PID e preferibile il semplice controllore PI con l'utilizzo di tecniche, dette di *anti windup* per la limitazione del contributo dato dall'azione integrale.

### 1.4.2 Regolatore PI

Per “cancellare” il polo dominante della funzione in anello aperto, imputabile alla parte meccanica è necessario porre

$$\frac{K_p^W}{K_i^W} = \frac{J}{B}$$

ovvero

$$\begin{cases} K_p^W = K^W \cdot \frac{J}{K_e} \\ K_i^W = K^W \cdot \frac{B}{K_e} \end{cases}$$

ottenendo così la funzione in anello aperto:

$$L_W^{PI} = \frac{K^W}{s} \cdot \frac{K^I}{s + K^I} \quad (7.17)$$

La relativa funzione in anello chiuso è data da:

$$G_W^{PI} = \frac{K^I K^W}{s^2 + K^I s + K^I K^W} \quad (7.18)$$

Quest'ultima rappresenta un sistema del secondo ordine la cui dinamica dipende dalla dinamica imposta all'anello di controllo in corrente  $K^I$  e in velocità  $K^W$ . Il guadagno del controllore della corrente contiene il contributo della resistenza ( $R$ ) concatenata, mentre il guadagno del controllo in velocità “nasconde” il contributo della meccanica del sistema (attrito viscoso, inerzia, costante di coppia del motore).

In un sistema del secondo ordine è particolarmente importante valutare le condizioni per cui possono insorgere oscillazioni nella risposta allo scalino, ovvero in presenza di poli complessi coniugati. Dall'analisi del denominatore della funzione di trasferimento (7.18), si ottiene

$$\Delta = (K^I)^2 - 4K^W K^I = K^I(K^I - 4K^W)$$

da cui si ricava il valore limite di  $K^W$  per cui si ottengono le prestazioni (in termini di velocità) massime senza la presenza di oscillazioni indesiderate

$$\Delta = K^I(K^I - 4K^W) \geq 0 \quad \Leftrightarrow \quad K^W \leq \frac{K^I}{4} \quad .$$

Per quanto riguarda l'analisi di stabilità, è necessario esplicitare i valori dei poli:

$$s_{1/2} = \frac{-K^I \pm \sqrt{\Delta}}{2} .$$

Si possono verificare diverse situazioni a seconda del valore assunto da  $\Delta$ , cioè a seconda del guadagno del regolatore di velocità:

- $\Delta < 0$ . Il sistema presenta oscillazioni, ma è asintoticamente stabile poiché la parte reale dei poli è  $-\frac{K^I}{2}$ , con  $K^I > 0$ .
- $\Delta = 0$ . Il sistema è sempre asintoticamente stabile e presenta due poli reali coincidenti, quindi si ottengono le massime prestazioni.
- $\Delta > 0$ . In questo caso, la stabilità è data dal segno del polo  $s_1 = \frac{-K^I + \sqrt{\Delta}}{2}$ . L'asintotica stabilità è garantita per  $\Delta \leq (K^I)^2$ , espressione sempre vera se  $K^W \geq 0$ .

Trattandosi di un sistema di secondo ordine, è conveniente caratterizzare il sistema tramite le caratteristiche di *frequenza propria*  $\omega_N$  e *smorzamento*  $\xi$  (equazione (7.19)).

$$\begin{aligned} \omega_N &= \sqrt{K^I \cdot K^W} \\ \xi &= \frac{K^I}{2\omega_N} = \frac{K^I}{2\sqrt{K^I \cdot K^W}} = \frac{1}{2} \cdot \sqrt{\frac{K^I}{K^W}} . \end{aligned} \quad (7.19)$$

Qualora la specifica di progetto non fosse l'abbattimento delle oscillazioni (o smorzamento unitario), ma l'impostazione di una specifica costante di tempo, è possibile analizzare più in dettaglio i poli del sistema e il determinante del denominatore della funzione di trasferimento in termini dello smorzamento e della frequenza naturale appena ricavati:

$$\Delta = 4\omega_N^2\xi^2 - 4\omega_N^2 = 4\omega_N^2(\xi^2 - 1)$$

da cui derivano i poli

$$s_{1/2} = \frac{-2\omega_N\xi \pm 2\omega_N\sqrt{\xi^2 - 1}}{2} = -\omega_N \cdot (\xi \mp \sqrt{\xi^2 - 1}) .$$

Si possono considerare, come al solito, due casi:

- $\xi \leq 1$ , che corrisponde alla massima prestazione (in termini di velocità) ottenibile.

$$\begin{aligned} \tau &= \left| \frac{1}{s_1} \right| = \frac{1}{\omega_N\xi} \\ &= \frac{1}{0.5\sqrt{K^I K^W} \sqrt{\frac{K^I}{K^W}}} = \frac{2}{K^I} = 2 \cdot \tau^I . \end{aligned}$$

- $\xi > 1$ .

Il polo dominante è  $s = -\omega_N(\xi - \sqrt{\xi^2 - 1})$ . Pertanto la costante di tempo del sistema è pari a

$$\begin{aligned}
 \tau &= \frac{1}{\omega_N \cdot (\xi - \sqrt{\xi^2 - 1})} \\
 &= \frac{1}{\sqrt{K^I K^W}} \cdot \frac{1}{0.5 \sqrt{\frac{K^I}{K^W}} - \sqrt{\frac{1}{4} \frac{K^I}{K^W} - 1}} \\
 &= \frac{1}{\sqrt{K^I K^W}} \cdot \frac{2}{\sqrt{\frac{K^I}{K^W}} - \sqrt{\frac{K^I - 4K^W}{K^W}}} \\
 &= \frac{2}{K^I - \sqrt{K^I(K^I - 4K^W)}}
 \end{aligned} \tag{7.20}$$

Volendo fissare  $\tau^W$ , si può procedere nel seguente modo:

$$\begin{aligned}
 K^I - \sqrt{K^I(K^I - 4K^W)} &= \frac{2}{\tau^W} \\
 \sqrt{K^I(K^I - 4K^W)} &= K^I - \frac{2}{\tau^W} \\
 K^I(K^I - 4K^W) &= \left(K^I - \frac{2}{\tau^W}\right)^2 \\
 K^W &= \frac{1}{4} \cdot \left[ K^I - \frac{\left(K^I - \frac{2}{\tau^W}\right)^2}{K^I} \right]
 \end{aligned}$$

Valutata l'influenza del guadagno del controllore di velocità  $K^W$  sulle prestazioni e sulla stabilità del controllo in velocità stesso, rimane da valutare l'influenza di tale parametro sulla coppia richiesta al motore, al fine di porre un ulteriore limite superiore per evitare che la coppia venga saturata.

Un limite superiore da imporre a  $K^W$  per evitare il più possibile la saturazione della coppia può essere ottenuto valutando la coppia richiesta all'istante 0 (zero), ovvero quando l'azione integrale è nulla ed è presente solo l'azione proporzionale. Infatti, nella maggior parte delle situazioni avviene la saturazione della coppia del motore quando l'errore di velocità è massimo, ovvero quando si richiede il movimento più rapido del sistema. La funzione di trasferimento da velocità a coppia è la seguente:

$$\left( \frac{T}{W_0} \right) = \frac{K^W(Js + B)}{s} \cdot \frac{K^I}{s + K^I} \cdot K_e \tag{7.21}$$

Se il termine relativo all'attrito  $B$  non è trascurabile, allora il valore di coppia che potrebbe saturare il sistema è:

$$\mu_T^{sat} = K^W \cdot B \cdot K_e \cdot \omega_{max}$$



Se invece il termine relativo all'attrito è trascurabile rispetto al termine inerziale ( $J$ ), allora l'equazione da usare prevede la presenza di  $J$  al posto di  $B$ :

$$\mu_T^{sat} = K^W \cdot J \cdot K_e \cdot \omega_{max} .$$

In generale, chiamato  $\psi$  il termine ( $J$  o  $B$ ) da cui dipende  $T^{sat}$ , il limite superiore è dato dalla relazione

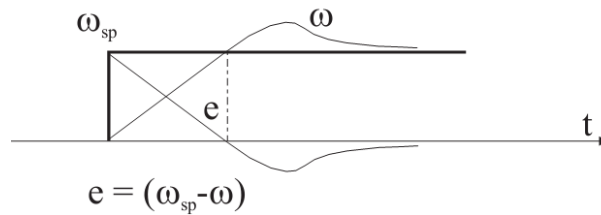
$$K^W \leq \frac{1}{K_e \psi} \frac{1}{T^{sat} \omega_{max}} . \quad (7.22)$$

La costante di tempo del sistema si può ricavare dalla relazione (7.20).

Qualora non si voglia imporre un limite troppo stringente alle prestazioni del sistema, bisogna tenere conto del fatto che si possano verificare le condizioni tali per cui l'azionamento satura la coppia. Facendo riferimento alla figura 7.12, la coppia di riferimento  $T_{sp}$  che il regolatore PI richiede al controllo ha la forma:

$$T_{sp} = K_p e(t) + K_i \int_0^t e(t) dt .$$

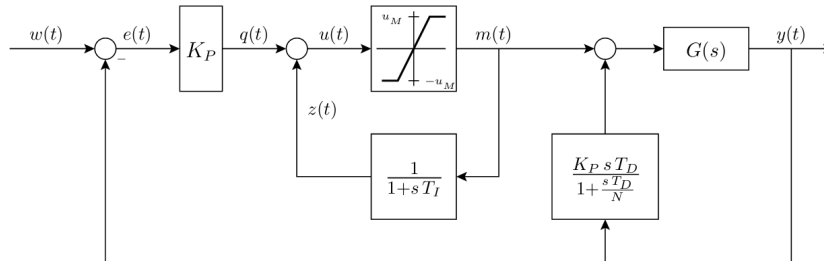
Il termine integrale, se caricato eccessivamente dal perdurare dell'errore a livelli alti (in funzione del fatto che la coppia erogata dal motore non è quella richiesta dal controllo), comporta un andamento di velocità come quello indicato in figura 7.13.



**Figura 7.13:** Saturazione della coppia

Fino a che il valore della velocità non raggiunge il riferimento, la sua crescita è di tipo lineare con pendenza massima pari a  $\frac{T_{max}}{J}$  in quanto il motore eroga per tutto il tempo la coppia massima. Questa situazione non è desiderabile perché provoca un'assenza totale di controllo. Nell'istante di annullamento dell'errore solo il termine proporzionale si azzerava, mentre quello integrale continua ad imporre un riferimento positivo di coppia e conseguentemente il motore continuerà ad accelerare il carico; a questo punto l'errore diviene negativo ( $\omega > \omega_{sp}$ ) e l'azione proporzionale si oppone a quella integrale. Lo scaricamento del termine integrale richiede un certo tempo, durante il quale si ha una sovraelongazione della velocità, effetto che potrebbe risultare indesiderato.

Occorre quindi limitare l'azione integrale con tecniche di *anti windup* in modo da evitare che la saturazione di coppia perduri anche in prossimità dell'annullamento dell'errore. In figura 7.14 viene mostrata una possibile realizzazione di un regolatore PID con desaturazione dell'azione integrale.



**Figura 7.14:** Desaturazione dell'azione integrale per un regolatore PI

Si noti, innanzitutto, che poiché usualmente l'azione derivativa è esercitata solo sull'uscita  $y$ , lo schema di desaturazione riguarda unicamente le azioni proporzionale e integrale. Si osservi inoltre che all'interno del regolatore viene replicata la caratteristica non lineare dell'attuatore. Se si opera in zona di linearità, la funzione di trasferimento complessiva tra l'errore  $e$  e la variabile  $m$  coincide con quella di un PI. Infatti

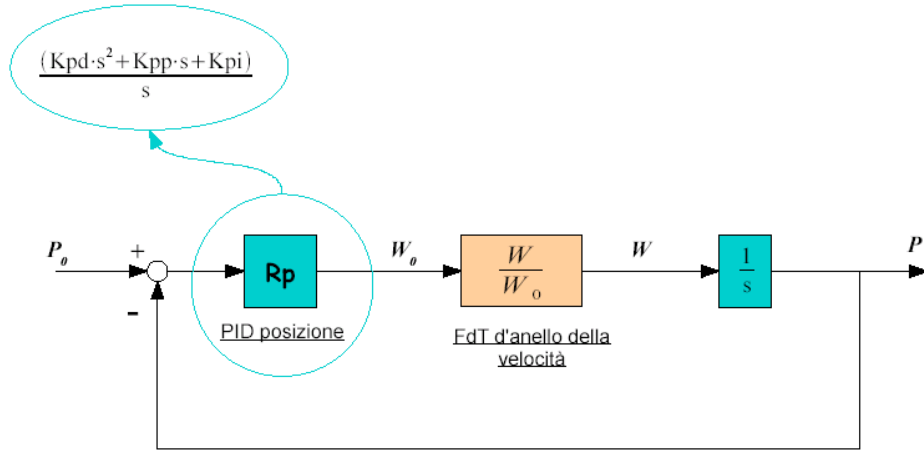
$$\begin{aligned} M(s) &= \frac{1}{1 - \frac{1}{1+sT_I}} Q(s) \\ &= \frac{1+sT_I}{sT_I} Q(s) \\ &= K_P \frac{1+sT_I}{sT_I} E(s) \end{aligned}$$

Si supponga ora  $K_P > 0$ , che l'errore si mantenga positivo per un certo periodo di tempo e che la variabile  $u$  sia saturata dal valore  $u_M$ ; allora anche  $m$  risulta uguale a  $u_M$  e la variabile  $z$  tende al valore costante  $u_M$  con la dinamica di un sistema del primo ordine con costante di tempo  $T_I$ . Se poi  $e$  cambia di segno, anche  $q$  assume segno negativo e la variabile  $u = q + z$  diventa inferiore al limite di saturazione  $u_M$ , cioè il sistema torna a comportarsi linearmente.

## 1.5 Controllo in posizione

A differenza degli altri due anelli del controllo in cascata, nessun disturbo influisce direttamente sul controllo in posizione. Pertanto, se i regolatori di velocità e corrente sono stati progettati correttamente, è possibile trascurare definitivamente l'effetto della forza contro-elettromotrice  $V_{cem}$  e della coppia resistente  $C_r$  dovuta al carico. La funzione d'anello

della velocità considerata è quella ricavata considerando un regolatore di tipo PI (vedi paragrafo 1.4.2) tarato in modo da ottenere le massime prestazioni possibili ( $\tau^W = 2\tau^I$ ) senza oscillazioni, ovvero in presenza di poli reali coincidenti. L'anello di posizione è mostrato in figura 7.15.



**Figura 7.15:** Schema a blocchi del controllo in posizione del motore in corrente continua

Introducendo la funzione di trasferimento derivante dal controllo in velocità (equazione (7.18)), si ottiene la seguente funzione di trasferimento in anello aperto:

$$L_P(s) = \left( \frac{P}{P_0} \right)_{anella} = R_p \cdot \frac{K^I K^W}{\left( s + \frac{K^I}{2} \right)^2} . \quad (7.23)$$

Dato che i problemi di saturazione della coppia sono stati affrontati nel progetto del regolatore di velocità, è possibile progettare un regolatore PD per abbassare l'ordine del sistema:

$$R_p = K_p^P + K_d^P s = K^P \left( s + \frac{K^I}{2} \right) .$$

I parametri del regolatore vengono tarati fissando:

$$K_p^P = K^P \frac{K^I}{2} , \quad K_d^P = K^P .$$

Utilizzando il suddetto regolatore, infine, si ottiene la funzione in anello aperto

$$L_P^{PD}(s) = \frac{K^I K^W K^P}{s \left( s + \frac{K^I}{2} \right)} \quad (7.24)$$

e la corrispondente funzione d'anello chiuso

$$G_P^{PD}(s) = \frac{K^I K^W K^P}{s \left( s + \frac{K^I}{2} \right) + K^I K^W K^P} . \quad (7.25)$$

Quindi il sistema rimane del secondo ordine. Analogamente a quanto fatto per l'anello di velocità, si possono analizzare le condizioni limite per cui non si verificano oscillazioni:

$$\Delta = \frac{(K^I)^2}{4} - 4K^P K^W K^I$$

$$K^P \leq \frac{K^I}{16K^W}$$

da cui, mettendosi esattamente nella condizione limite si ottengono due poli reali negativi (quindi il sistema è asintoticamente stabile) coincidenti:

$$s_{1/2} = -\frac{K^I}{4}$$

e la costante di tempo del sistema è

$$\tau^P = \frac{4}{K^I} = 4\tau^I .$$

Come già detto per l'anello di velocità, è possibile descrivere il sistema in termini di smorzamento e frequenza propria:

$$\omega_N = \sqrt{K^I K^W K^P}$$

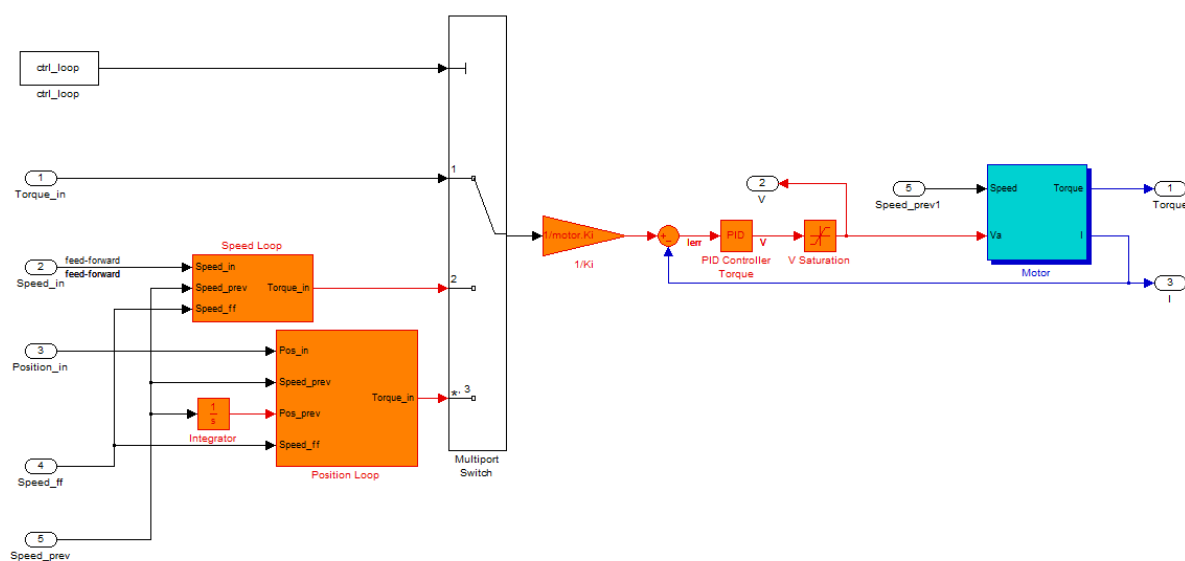
$$\xi = \frac{K^I}{2\sqrt{K^I K^W K^P}}$$

da cui è infine possibile ricavare un'espressione di  $K^P$  per ottenere un sistema con costante di tempo desiderata:

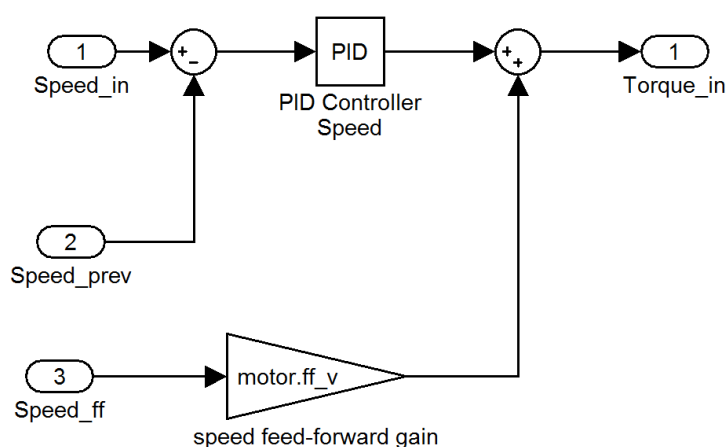
$$K^P = \frac{1}{16K^I K^W} \cdot \left[ (K^I)^2 - 4 \left( \frac{K^I}{2} - \frac{2}{\tau_{des}^P} \right)^2 \right] .$$

## 1.6 Schema Simulink del controllo del motore DC

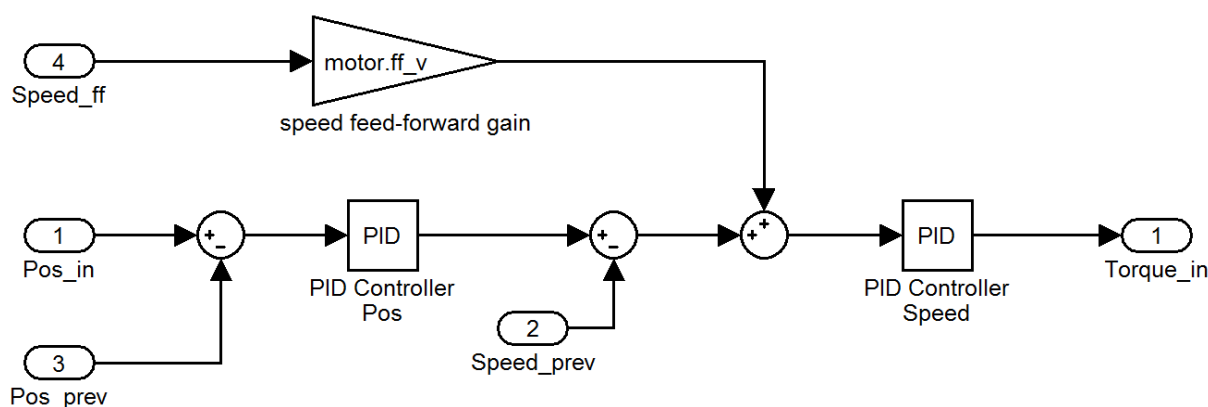
In figura 7.16 viene mostrato lo schema a blocchi Simulink implementato per la simulazione del controllo del motore DC. In fondo allo schema è inserito il modello del motore già mostrato in figura 7.5. Il modello del motore è inserito nell'anello di controllo in coppia, mentre gli altri due anelli sono implementati nei blocchi denominati **Speed Loop** e **Position Loop** (figura 7.17). Il tipo di controllo è selezionabile tramite uno switch che permette di passare dal controllo in coppia a quello di posizione o velocità.



*Figura 7.16: Schema Simulink per l'implementazione del controllo del motore DC*



*(a) Anello di controllo in velocità.*



*(b) Anello di controllo in posizione.*

*Figura 7.17: Implementazione Simulink degli anelli di controllo in posizione e velocità*

Il listato 7.1 mostra invece l'implementazione della classe motore utilizzata per inizializzare nel modo corretto i blocchi dello schema Simulink appena presentato.

---

*Codice 7.1: Classe Matlab per l'inizializzazione del controllo del motore DC*

---

```

1  classdef Simulink_Motor < handle
2      properties (SetAccess = private)
3          % Motor Parameters
4          L;
5          R;
6          Ki;
7          Km;
8          Cmax;
9          J;          % rotor inertia
10         B;          % friction coefficient
11         V_az;
12
13         %
14         tau_red;
15
16         % Simulation Parameters
17         torque_saturation;
18
19         % Motor Regulator
20         K_i;
21         Kp_i;
22         Ki_i;
23         Kd_i;
24         K_v;
25         Kp_v;
26         Ki_v;
27         Kd_v;
28         ff_v;
29         K_p;
30         Kp_p;
31         Ki_p;
32         Kd_p;
33     end
34
35     properties (SetAccess = public)
36         % Motor connected parameters

```

---

```

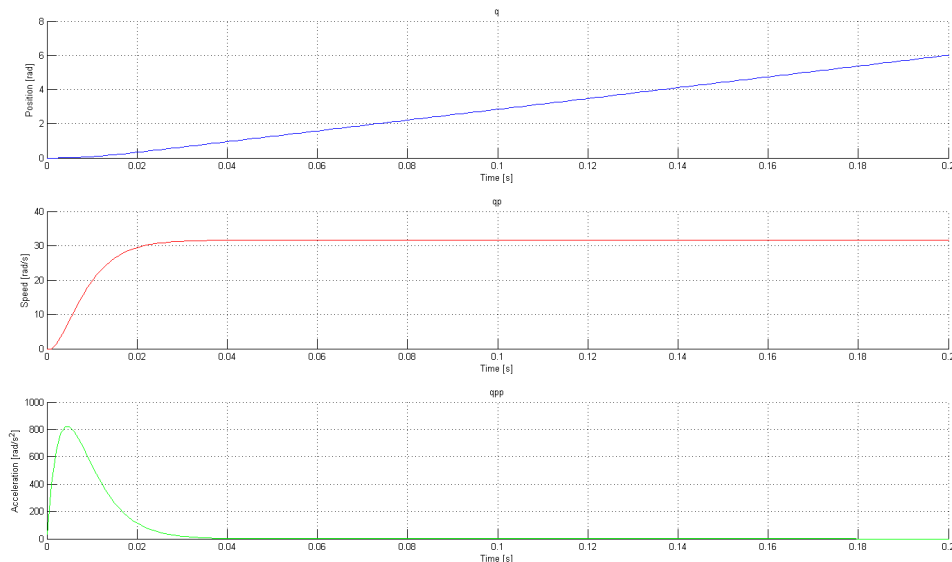
37     load;
38 end
39
40 methods
41     function M = Simulink_Motor(L, R, Ki, Km, Cmax, J, B, V_az,
42         tau_red,
43         torque_saturation)
44         % Here the initialization of class fields...
45
46         function set_current_ctrl(M, tau_current)
47             M.K_i = 1/tau_current;
48             M.Kp_i = M.L * M.K_i;
49             M.Ki_i = M.R * M.K_i;
50             M.Kd_i = 0;
51         end
52
53         function set_position_ctrl_after_PI_damping (M, damping)
54             M.K_p = M.K_i / (16 * damping^2 * M.K_v) ;
55             M.Kp_p = M.K_p * M.K_i / 2;
56             M.Ki_p = 0;
57             M.Kd_p = M.K_p ;
58         end
59
60         function set_position_ctrl_after_PI_tau (M, tau)
61             if tau >= 4/M.K_i
62                 M.K_p = 1 / (16*M.K_v*M.K_i) * (M.K_i^2 - 4*(M.K_i/2 -
63                     2/tau)^2) ;
64             else
65                 disp('error: tau not reachable');
66             end
67
68             M.Kp_p = M.K_p * M.K_i / 2;
69             M.Ki_p = 0;
70             M.Kd_p = M.K_p ;
71         end
72     end
73 end

```

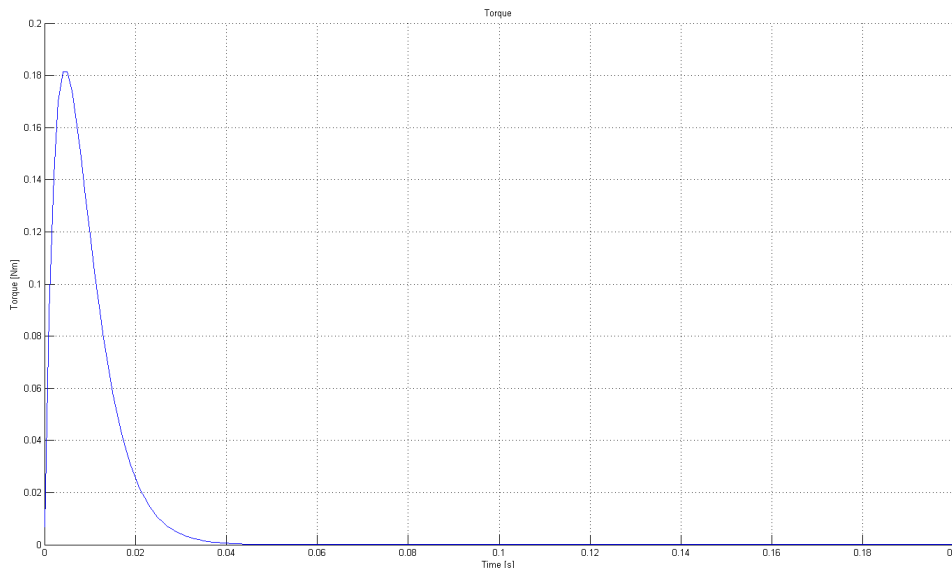
---

## 1.7 Simulazioni

Di seguito si mostrano gli andamenti di coppia, accelerazione e velocità della risposta allo scalino del controllo in velocità progettato, nel caso delle massime prestazioni (figura 7.18), cioè  $\tau = 0.005$  e di un controllo 4 volte più lento (figura 7.19).



(a) Posizione, velocità e accelerazione.

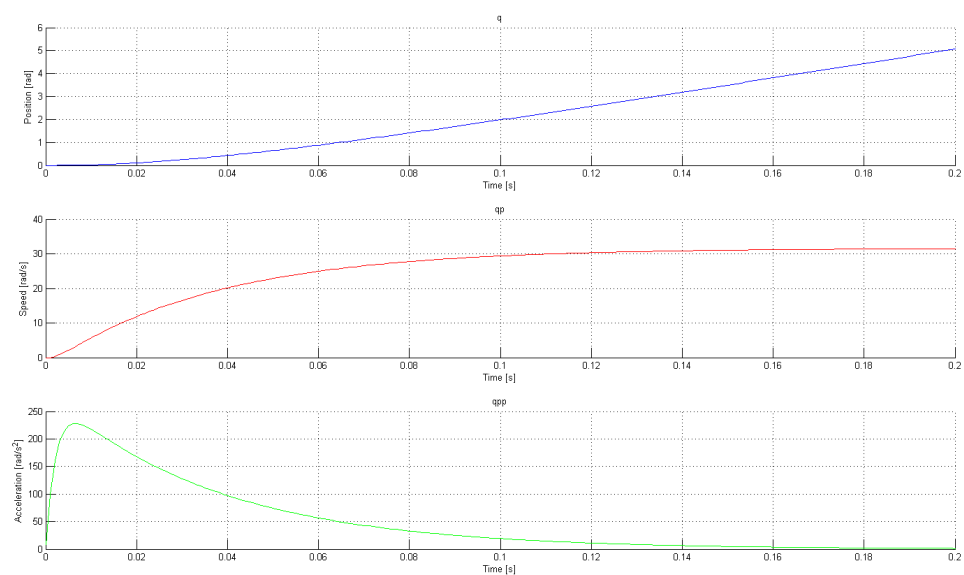


(b) Coppia.

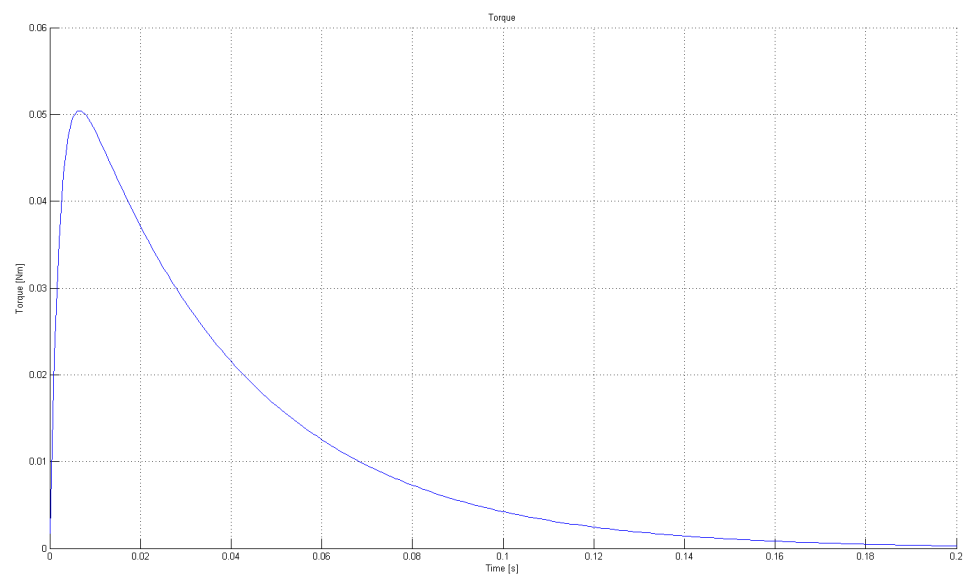
**Figura 7.18:** Controllo in velocità del motore DC, massime prestazioni

Come per il controllo in velocità, si riportano le risposte allo scalino per il controllo in posizione alla massima velocità ( $\tau = 0.01$ , figura 7.20) e uno a velocità ridotta, ma più lineare ( $\tau = 0.15$ , figura 7.21).



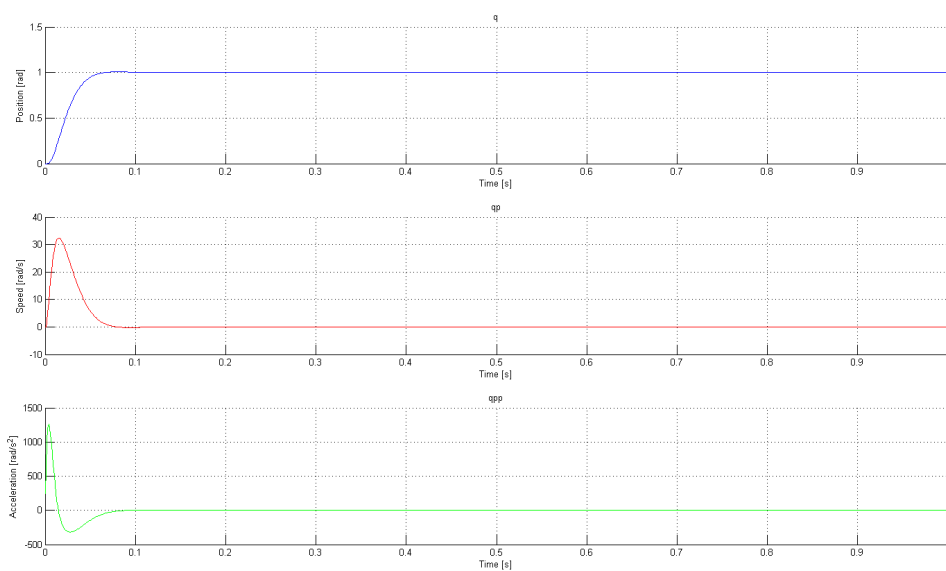


(a) Posizione, velocità e accelerazione.

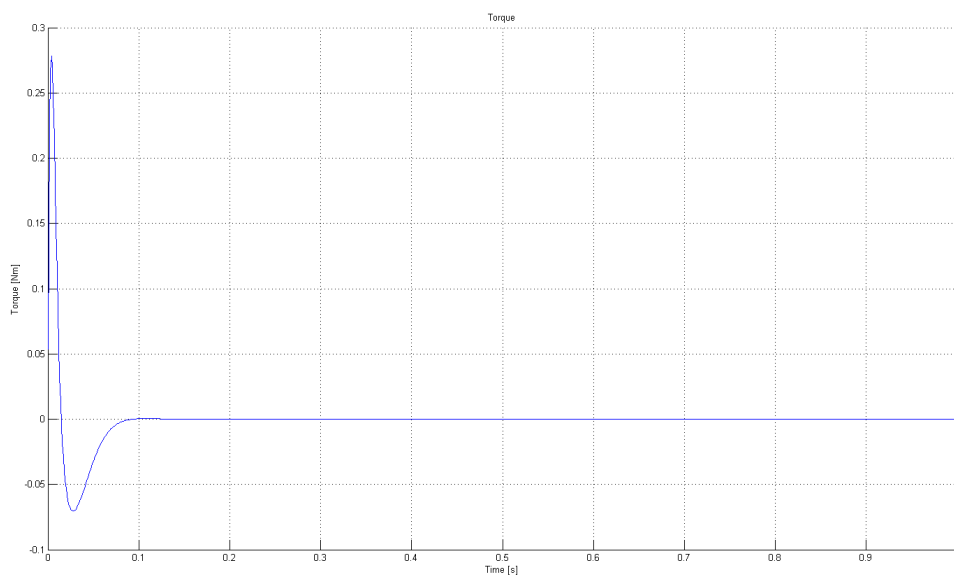


(b) Coppia.

**Figura 7.19:** Controllo in velocità del motore DC, prestazioni contenute

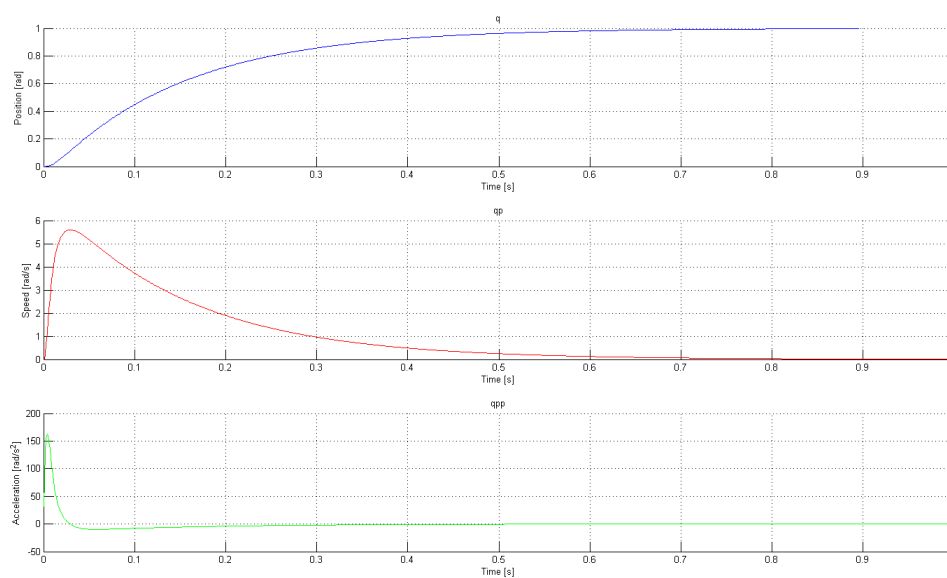


(a) Posizione, velocità e accelerazione.

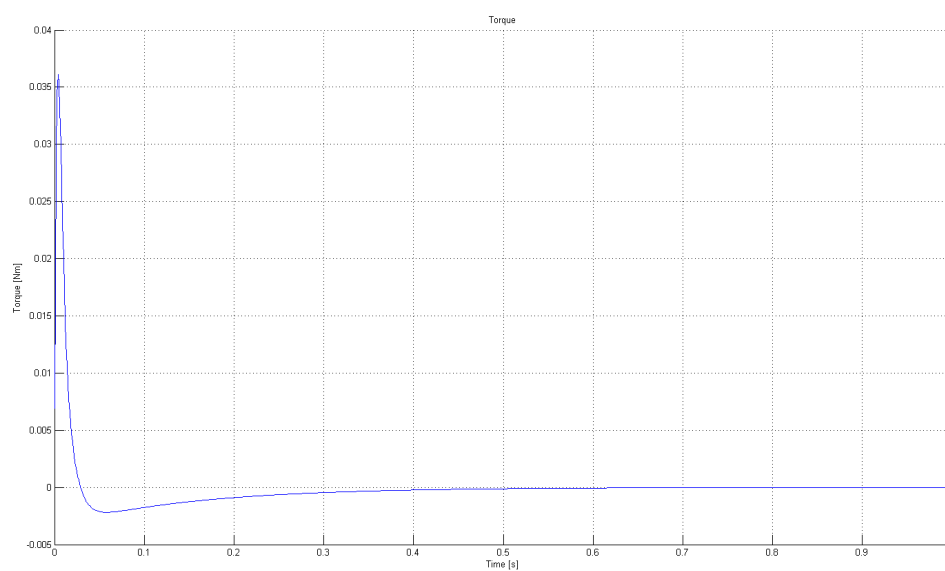


(b) Coppia.

**Figura 7.20:** Controllo in posizione del motore DC, massime prestazioni



(a) Posizione, velocità e accelerazione.



(b) Coppia.

**Figura 7.21:** Controllo in posizione del motore DC, prestazioni contenute

## 2 Dinamica del sistema di visione

Gli aspetti che determinano la dinamica del controllo con sistema di visione in retroazione derivano principalmente dall'interazione tra il sensore (sistema di visione) e il controllore del robot. Infatti, quest'ultima porta alle due problematiche fondamentali che bisogna affrontare nel progettare un controllo in asservimento visivo:

- *Latenza del sistema di visione*: generalmente il sistema di visione richiede molto tempo per l'elaborazione delle immagini, la quale corrisponde, nel controllo tramite asservimento visivo, alla generazione del set-point di velocità, posizione o coppia da fornire al controllore del motore. Questo tempo è fondamentalmente un tempo morto per il controllore del motore perché questo non può comandare gli attuatori alla frequenza che vorrebbe. Ad oggi il tempo di elaborazione, nel caso degli algoritmi più veloci ed efficienti, è circa 10 volte più elevato del periodo di tempo dedicato per il controllo del moto (1  $ms$  con un controllore avente frequenza 1  $kHz$  contro qualche decina di  $ms$  da spendere per l'elaborazione immagine);
- *Frequenza di campionamento del sistema di visione*: il periodo di campionamento di un sensore di visione (ad esempio una camera CCD) è anch'esso più lungo del periodo di tempo dedicato al controllo degli attuatori. In questo caso si può arrivare (nel caso delle risoluzioni più basse) a 100 o 200  $Hz$ .

In particolare, è questo secondo aspetto che caratterizza maggiormente la dinamica del visual servoing. In pratica, si ha a che fare con due anelli di controllo innestati, ovvero visione e controllo giunti, operanti a frequenze diverse. Pertanto, il sistema formato da sensore di visione e controllore può essere considerato a pieno titolo un *sistema multirate*. In generale, molti degli approcci alla progettazione di regolatori e anelli di retroazione sono validi solamente se i sistemi sono lineari e a tempo invariante. Per questo motivo, l'approccio scelto per l'analisi dinamica del controllo è stato quello di linearizzare il sistema, in modo da poter ottenere una buona approssimazione della funzione di trasferimento effettiva. Questa approssimazione, come sarà evidenziato, non è penalizzante perché comporta una buona modellizzazione a frequenze inferiori alla banda passante messa a disposizione dal sistema di visione, cioè l'unica a cui si può operare.

## 2.1 Ritardo d'elaborazione immagini

Il ritardo di tempo, descritto come

$$y(t) = u(t - \tau) \quad (7.26)$$

è un sistema lineare e stazionario.

Applicando la trasformata di Laplace ad ambo i membri dell'equazione (7.26), si ha

$$Y(s) = e^{-\tau s} U(s)$$

per cui la funzione di trasferimento del ritardo è data da:

$$G(s) = e^{-\tau s} . \quad (7.27)$$

Per linearizzare il ritardo di tempo vengono normalmente impiegate le cosiddette *approssimanti di Padé*, derivate come descritto di seguito. Si definisca in primo luogo la struttura della funzione approssimante, razionale in  $s$ , che si desidera considerare e il valore  $\hat{s}$  della variabile complessa  $s$  per cui si vuole ottenere l'approssimante. Il valore dei  $q$  parametri liberi della funzione approssimante viene quindi determinato facendo coincidere i primi  $q$  termini del suo sviluppo in serie attorno a  $\hat{s}$  con i corrispondenti valori dello sviluppo in serie di  $e^{-\tau s}$  attorno a  $\hat{s}$ . Poiché normalmente, come anche in questo caso, si è interessati alle approssimazioni a bassa frequenza, si può scegliere  $\hat{s} = 0$ .

Per gli scopi di questa analisi, in cui interessa il comportamento solo in bassa frequenza, si può scegliere di ottenere un approssimante del primo ordine, avente struttura

$$G_{delay}^I(s) = \mu \cdot \frac{1 + as}{1 + bs} . \quad (7.28)$$

Poiché i parametri liberi della (7.28) sono  $\mu$ ,  $a$ ,  $b$ , è possibile far coincidere i primi tre coefficienti dello sviluppo di McLaurin di

$$e^{-\tau s} \approx 1 - \tau s + \frac{\tau^2 s^2}{2} + \dots$$

con i corrispondenti termini dello sviluppo della (7.28):

$$\mu \cdot \frac{1 + as}{1 + bs} = \mu + \mu(a - b)s + \mu b(b - a)s^2 + \dots$$

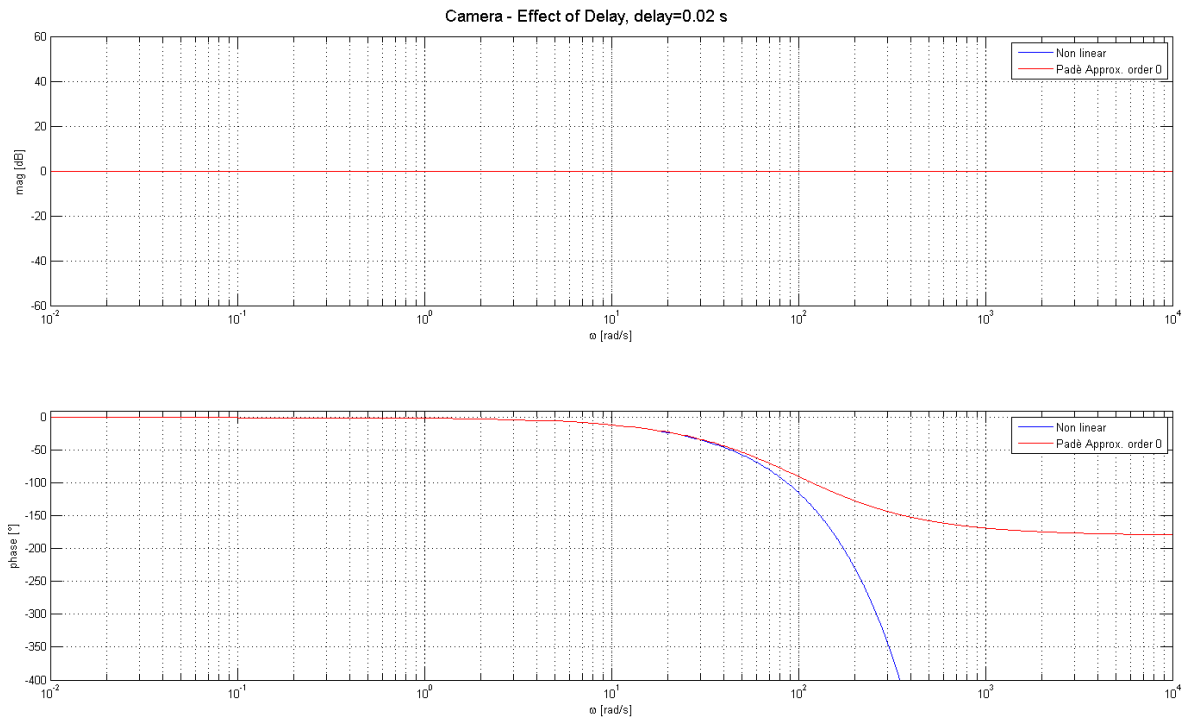
Risulta pertanto:

$$\begin{cases} \mu = 1 \\ b - a = \tau \\ b(b - a) = \tau^2/2 \end{cases}$$

da cui si ricava la funzione di trasferimento relativa all'elaborazione delle immagini:

$$G_{delay}^I(s) = \frac{1 - 0.5\tau s}{1 + 0.5\tau s} . \quad (7.29)$$

Poiché tale funzione di trasferimento ha un polo e uno zero di stesso modulo, ma segno diverso, si può concludere che l'effetto introdotto dal ritardo è pari a quello di uno *sfasatore puro* asintoticamente stabile, con risposta in frequenza di modulo unitario e fase decrescente all'aumentare della frequenza. Relativamente al problema del controllo tramite visual servoing, questo porta ad una considerazione lapalissiana: il ritardo di elaborazione pone il primo limite superiore alla banda passante del sistema. In parole povere, più il sistema di elaborazione immagini è pronto ed efficiente, più prestante sarà il controllo risultante. Tuttavia, la velocità di elaborazione si deve scontrare anche col problema della frequenza di acquisizione delle immagini, il quale verrà affrontato nel paragrafo 2.2.



**Figura 7.22:** Diagramma di Bode del ritardo puro e dell'approssimante di Padé di primo grado

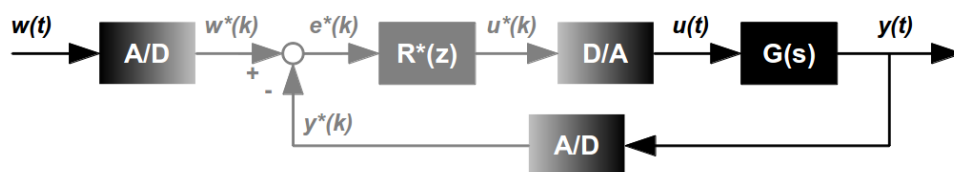
La figura 7.22 mostra il confronto dei diagrammi di Bode delle funzioni di trasferimento di un ritardo puro e del suo approssimante di Padé per un ritardo di 20 ms, corrispondente al periodo medio di elaborazione immagini per un sistema di acquisizione a 50 Hz. Tale valore è coerente con un ampio range di algoritmi di estrazione delle feature da immagini aventi differenti risoluzioni. Come si può vedere, l'approssimazione è decisamente buona

per frequenze inferiori ai 50  $Hz$ , ovvero alla banda passante permessa dal sistema di elaborazione immagini.

## 2.2 Sistema digitale

Il sistema di visione è difficilmente modellabile come un sistema a tempo continuo non solo per l'introduzione del ritardo di elaborazione, ma anche perché le telecamere, così come qualsiasi controllore real-time, sono per natura un sistema a tempo *discreto*. Infatti, una delle caratteristiche di cui tenere conto nella scelta di una telecamera industriale è il *framerate*, ovvero la massima frequenza di acquisizione delle singole immagini. L'inverso del framerate è il *periodo di acquisizione*, cioè l'unità di tempo base del sistema a tempo discreto.

Nel campo dei controlli automatici, la teoria dei sistemi discreti viene utilizzata per progettare i controllori in un mondo “più vicino” a quello reale, ovvero nel mondo digitale. Poiché il *regolatore digitale* può ricevere in ingresso soltanto sequenze di numeri, è necessario prevedere innanzitutto la presenza del *convertitore analogico/digitale A/D* che trasformi il segnale a tempo continuo  $e(t)$  nel segnale a tempo discreto  $e^*(k)$ ; tale blocco è detto anche *campionatore*. Analogamente, è necessario il *convertitore digitale/analogico D/A*, detto anche *mantenitore*, che trasforma il segnale a tempo discreto  $u^*(k)$  nel segnale a tempo continuo  $u(t)$ . In figura 7.23 è rappresentato lo schema di controllo digitale a campionamento dell'uscita e del riferimento.



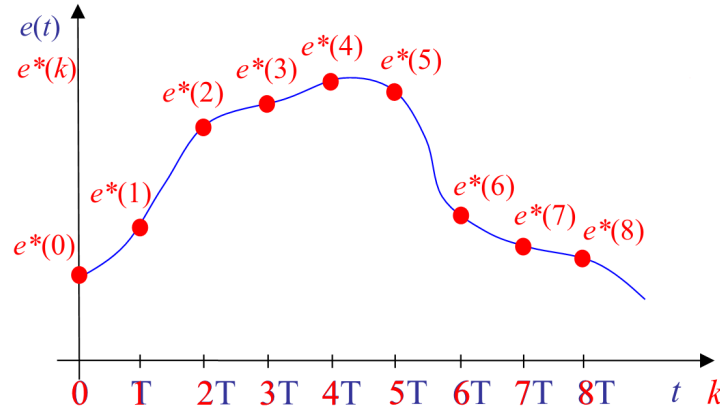
**Figura 7.23:** Schema di controllo digitale a campionamento dell'uscita e del riferimento

### 2.2.1 Campionatore

Il campionatore ideale e periodico di figura 7.24 è un convertitore che, come già detto, dato in ingresso il segnale a tempo continuo  $e(t)$ , produce in uscita il segnale a tempo discreto dato dalla equazione (7.30), dove  $T_s$  è il *periodo di campionamento* che rappresenta

la durata dell'intervallo di tempo continuo che intercorre tra gli istanti in cui vengono prelevati due campioni successivi.

$$e^*(k) = e(kT_s + \tau_s) . \quad (7.30)$$



**Figura 7.24:** Campionatore periodico

A partire da  $T_s$ , è possibile definire anche la frequenza  $f_s$  e la pulsazione di campionamento:

$$f_s = \frac{1}{T_s} , \quad \omega_s = \frac{2\pi}{T_s} .$$

### 2.2.2 Mantentore

Il *mantentore* può essere visto come l'elemento complementare del campionatore. Infatti, ha il compito di convertire un segnale a tempo discreto  $u^*(k)$  nel segnale a tempo continuo  $u(t)$ . Nel caso più semplice si ha a che fare con un *mantentore di ordine zero* (ZOH, *Zero Order Holder*) che mantiene per  $u(t)$  un andamento polinomiale di ordine zero, quindi costante, compatibile con l'ultimo valore di  $u^*(k)$ :

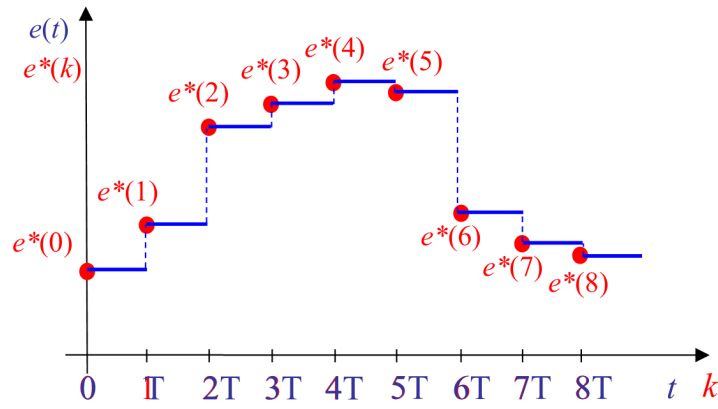
$$u(t) = u^*(k) , \quad kT_m + \tau_m \leq t < (k+1)T_m + \tau_m . \quad (7.31)$$

Il funzionamento dello ZOH per  $\tau_m = 0$  è rappresentato in figura 7.25.

Il mantentore di ordine zero, oltre ad essere il più semplice, è il più adatto a rappresentare il sistema di visione poiché le informazioni provenienti dal sistema di elaborazione immagini rimangono disponibili fino a quando delle informazioni più nuove non vengono messe a disposizione, ovvero fino a quando una nuova immagine non è stata elaborata.

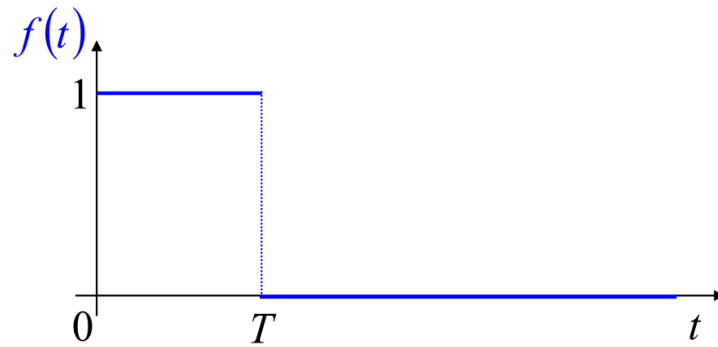
Per l'analisi successiva è utile determinare la relazione che intercorre, in termini di trasformate, tra il segnale  $f^*(k)$  di ingresso allo ZOH e il corrispondente segnale d'uscita





**Figura 7.25:** *Mantenitore di ordine zero*

$f(t)$ . Anziché condurre l'analisi con riferimento alle corrispondenti trasformate di Fourier, è conveniente analizzare inizialmente la relazione fra la trasformata Zeta  $F^*(z)$  di  $f^*(k)$  e la trasformata di Laplace  $F(s)$  di  $f(t)$ . A questo scopo, si assuma dapprima che il segnale d'ingresso sia un impulso all'istante  $k = 0$ , cioè  $f^*(k) = \text{imp}^*(k)$  e  $F^*(z) = 1$ .



**Figura 7.26:** *Risposta all'impulso dello ZOH*

L'uscita dello ZOH corrisponde all'impulso rettangolare in figura 7.26

$$f(t) = h_0(t) = \text{sca}(t - \text{sca}(t - T))$$

la cui trasformata di Laplace è

$$H_0(s) = \frac{1 - e^{sT}}{s} . \quad (7.32)$$

In generale, dato il segnale di ingresso  $f^*(k)$  con trasformata

$$F^*(z) = \sum_{k=0}^{\infty} f^*(k) z^{-k}$$

il corrispondente segnale d'uscita è

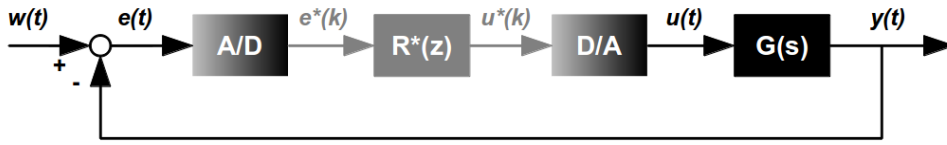
$$f(t) = \sum_{k=0}^{\infty} f^*(k) h_0(t - kT)$$

con trasformata di Laplace del mantentore ZOH data da

$$F(s) = \sum_{k=0}^{\infty} f^*(k) e^{-skT} H_0(s) = H_0(s) F^*(e^{sT}) . \quad (7.33)$$

### 2.2.3 Sistema ibrido

Per rendere il sistema continuo, si può trattare il sistema sotto controllo come un sistema ibrido (figura 7.27), in cui il sensore (telecamera) è a tempo discreto e, tutto quello che c'è a valle verrà poi considerato a tempo continuo. Lo studio dei sistemi di controllo ibridi viene condotto interpretando la serie costituita da campionatore, regolatore digitale e mantentore come un unico sistema il cui ingresso e la cui uscita sono segnali a tempo continuo. In quest'ottica, è necessario determinare le relazioni che intercorrono tra le trasformate di Fourier di questi segnali, cioè ricavare la relativa “risposta in frequenza”.



*Figura 7.27: Schema di controllo ibrido*

La relazione tra le trasformate di Fourier di  $e(t)$  ed  $e^*(k)$  è data dal teorema del campionamento:

$$E^*(e^{j\omega T}) = \frac{1}{T} E_s(j\omega) , \quad (7.34)$$

dove

$$E_s(j\omega) = \sum_{h=-\infty}^{+\infty} E(j(\omega + h\omega_s)) , \quad \omega_s = \frac{2\pi}{T} .$$

D'altra parte, dato il regolatore digitale  $R^*(z)$ , la funzione di trasferimento completa è data da:

$$U^*(e^{j\omega T}) = R^*(e^{j\omega T}) E^*(e^{j\omega T})$$

mentre, per l'equazione (7.33) si ottiene la relazione

$$U(j\omega) = H_0(j\omega) U^*(e^{j\omega T}) . \quad (7.35)$$

Unendo le equazioni (7.34) e (7.35), si ottiene

$$U(j\omega) = \frac{1}{T} H_0(j\omega) R^*(e^{j\omega T}) E_s(j\omega) . \quad (7.36)$$

Per ricavare un legame diretto tra  $U(j\omega)$  e  $E(j\omega)$  si può notare che:

- se  $e$  è un segnale a banda limitata con pulsazione massima  $\omega_{max} < \omega_N$ , per il teorema del campionamento nell'intervallo  $[0, \omega_N]$  lo spettro  $E_s(j\omega)$  coincide con  $E(j\omega)$ ;
- come già osservato,  $H_0(s)$  può essere interpretata come la funzione di trasferimento di un filtro passa-basso con guadagno pari a  $T$  e banda passante  $[0, \omega_N]$ .

Pertanto, in base a queste considerazioni, si può ritenere

$$H_0(j\omega)E_s(j\omega) \approx H_0(j\omega)E(j\omega)$$

e, quindi, dalla relazione (7.36) risulta

$$U(j\omega) \approx \frac{1}{T}H_0(j\omega)R^*(e^{j\omega T})E(j\omega) .$$

Quest'ultima relazione mostra come il sistema a tempo continuo costituito da campionatore, regolatore e mantenitore possa essere approssimativamente rappresentato dalla funzione di trasferimento (7.37).

$$R(s) = \frac{1}{T}H_0(s)R^*(e^{sT}) . \quad (7.37)$$

Dato che in questo paragrafo si sta cercando di modellare il comportamento di un sensore (il sistema di visione), e non di un regolatore, si può considerare

$$R(z) = 1 .$$

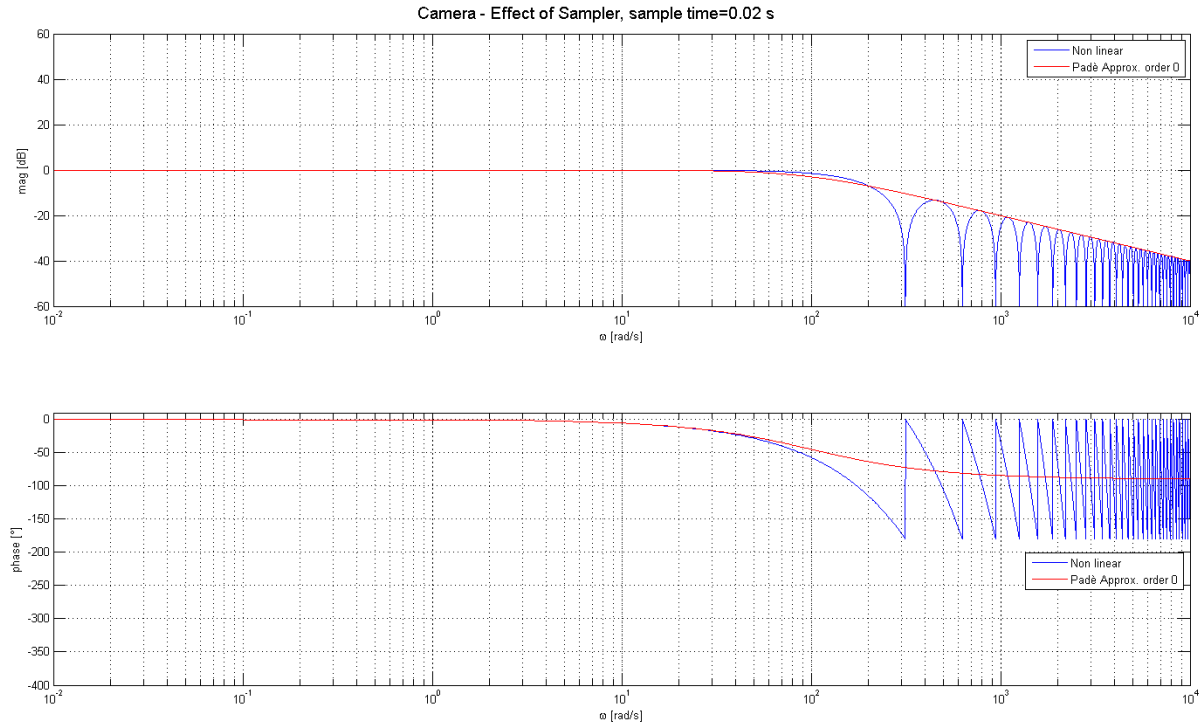
Quindi, dalla relazione (7.32) e dato l'approssimante di Padé (7.29), si ottiene la funzione di trasferimento completa del sistema di visione, definita come

$$\begin{aligned} R_v(s) &= \frac{1}{\tau} \frac{1 - e^{-s\tau}}{s} 1 = \frac{1}{\tau} \frac{1 - \frac{1-0.5\tau s}{1+0.5\tau s}}{s} \\ &= \frac{1}{\tau} \frac{\tau s}{s(1 + 0.5\tau s)} = \frac{1}{(1 + 0.5\tau s)} . \end{aligned} \quad (7.38)$$

La figura 7.28 mostra come, allo stesso modo che per il ritardo, l'approssimazione sia buona per frequenze basse.

Per facilitare il calcolo della risposta in frequenza, si può procedere alla seguente riscrittura dell'equazione (7.32):

$$\begin{aligned} H_0(j\omega) &= \frac{e^{-j\omega T}}{j\omega} \\ &= e^{-j\omega T/2} \frac{e^{j\omega T/2} - e^{-j\omega T/2}}{2j} \frac{2j}{j\omega} = T e^{-j\omega T/2} \frac{\sin(\omega T/2)}{\omega T/2} \end{aligned}$$



**Figura 7.28:** Diagramma di Bode del gruppo campionatore-mantenitore puro e dell'approssimante di Padé di primo grado

da cui risulta

$$\begin{cases} |H_0(j\omega)| = T \left| \frac{\sin(\omega T/2)}{\omega T/2} \right| \\ \angle(H_0(j\omega)) = -\frac{\omega T}{2} + \angle(\sin(\frac{\omega T}{2})) \end{cases}.$$

Dai diagrammi in figura 7.28 si può evincere che nell'intervallo di pulsazioni  $[0, \omega_N]$  il modulo di  $H_o(j\omega)$  si mantiene all'incirca costante per poi diminuire rapidamente, mentre la sua fase è identica a quella che sarebbe fornita da un ritardo di tempo pari a  $T/2$ . Pertanto, almeno in prima approssimazione, lo ZOH può considerarsi come un filtro reale passa-basso con banda passante  $[0, \omega_N]$ , come tra l'altro è l'approssimante di Padé.

Il mantenitore, quindi, influenza sia il modulo che la fase della risposta in frequenza e, nel caso della telecamera, il suo effetto va ad aggiungersi a quello del ritardo dovuto all'elaborazione delle immagini.

## 2.3 Altre non-linearità

In questo paragrafo, si fa cenno ad altre non-linearità di cui non si è potuto tenere conto nella funzione di trasferimento utilizzata per l'analisi della dinamica del sistema di controllo. Tuttavia, il loro effetto è stato studiato in fase di simulazione.

- *Quantizzazione.* Il campionatore descritto nel paragrafo 2.2 è ideale, in quanto si suppone che negli istanti di campionamento il valore  $e^*$  coincida perfettamente col valore di  $e$ . In realtà, nei sistemi digitali i valori reali  $e^*$  sono rappresentati con parole di lunghezza finita, cioè con un numero finito di bit. Il campionamento implica quindi inevitabilmente un'approssimazione, detta *quantizzazione*, dovuta alla codifica del segnale. Nel caso del sistema di visione, si vede l'effetto della quantizzazione nella formazione della matrice di pixel che formano l'immagine. Questa, infatti, viene suddivisa in  $n$  righe e  $m$  colonne. Il risultato dell'elaborazione dell'immagine è generalmente un insieme di pixel che rappresentano i punti d'interesse dell'oggetto individuato. Successivamente, tramite gli algoritmi di visual servoing, l'informazione viene convertita da pixel a coordinate nello spazio reale (ad esempio tramite triangolazione). Tuttavia, la misura è meno corretta tanto maggiore è la risoluzione spaziale di un singolo pixel. La quantizzazione, quindi, può portare ad effetti imprevedibili dovuti all'ottenimento di comandi di moto troppo grossolani, soprattutto nel caso di immagini sgranate, ovvero poco risolte o prese da distanze troppo piccole.
- *Rumore* derivante dall'elaborazione delle immagini. Le informazioni ricavabili tramite l'elaborazione delle immagini derivano generalmente da informazioni relative ai pixel appartenenti alla feature geometrica individuata (linea, punto, ...). Tale individuazione è affetta da errori, tanto che l'applicazione dello stesso algoritmo su due immagini della stessa scena prese ad istanti temporali diversi può dare risultati leggermente diversi. Infatti, l'individuazione dei pixel delle feature è influenzata da diversi fattori, tra cui la qualità dell'immagine di partenza (presenza o meno di rumore elettronico, presenza di ombre, qualità del contrasto, ...) e i parametri con cui l'algoritmo di individuazione è stato inizializzato (livelli di sogliatura e numero di iterazioni). La situazione è anche peggiore se la telecamera riprende la stessa scena (fondamentalmente l'oggetto da inseguire) da punti di osservazioni diversi, il che porta inevitabilmente ad avere immagini con diverse condizioni. Relativamente al modello, la presenza di rumore comporta l'aggiunta di un disturbo al termine della proiezione dei punti d'interesse sul piano immagine. Per modellare tale rumore, si è considerato il caso peggiore, in cui cioè la misura di tutte le feature viene influenzata da un rumore bianco (rumore gaussiano) di media nulla e deviazione standard proporzionale alla distanza media tra il pixel corretto e il pixel individuato (un valore coerente può essere un errore pari a 1 o 2 % della risoluzione massima). Pertanto, ad ogni pixel viene aggiunta una quantità indipendente da quella aggiunta per gli

altri pixel.

- *Variabilità del tempo di elaborazione.* Il tempo richiesto per l'elaborazione delle immagini non è costante poiché esso dipende dalla qualità delle immagini di partenza e dal numero di iterazioni richiesto dagli algoritmi scelti per convergere ad un risultato singolo. Questo comporta che la funzione di trasferimento del ritardo deve essere modellizzata con un ritardo  $\tau$  pari al tempo di elaborazione medio.

## 2.4 Funzione di trasferimento considerata

Il comportamento dinamico della telecamera è modellabile tramite i seguenti blocchi (rappresentati nello schema Simulink in figura 7.29):

- *Proiezione di punti in coordinate 3D sul piano immagine.* Tale blocco richiede in ingresso la posizione dei punti da proiettare(`points_xyz`) e le matrici di rototraslazione per identificare la posizione dei punti(`w2p`) e della telecamera(`w2cam`) rispetto all'origine del sistema di riferimento. In uscita al blocco si hanno le coordinate dei punti proiettati (in pixel).
- *Rumore gaussiano.* Ai pixel viene aggiunto il rumore gaussiano, calcolato come variabile casuale gaussiana (7.2, dove `SC` è un oggetto della classe che rappresenta la telecamera).

---

### Codice 7.2: Calcolo del rumore gaussiano

---

```
y(i,j) = random('norm', SC.noise.mu, SC.noise.sigma);
```

---

- *Ritardo d'elaborazione.* Di seguito si aggiunge la latenza dovuta al tempo di elaborazione. Come nel caso del rumore, il tempo di elaborazione è fissato di base come l'inverso della frequenza d'elaborazione fissata (ad esempio  $50\text{ Hz}$ ). Questo valore di base viene incrementato tramite un'ulteriore variabile gaussiana (positiva) in modo da modellizzare la variabilità del tempo di elaborazione, come mostrato dal codice 7.3. Al massimo il tempo di elaborazione risulta il doppio del tempo di base.

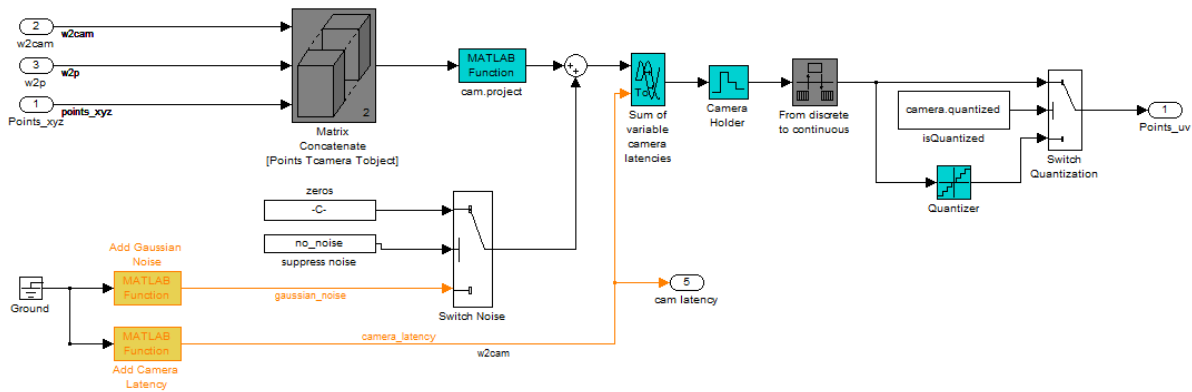
---

### Codice 7.3: Variabilità del ritardo di elaborazione

---

```
latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5);
```

---



**Figura 7.29:** Schema Simulink del modello dinamico del sensore di visione

- *Campionatore e mantenitore.* Il blocco successivo è quello costituito dal mantenitore e dal convertitore da digitale a continuo (DAC).
- *Quantizzatore.* Infine, i risultati ottenuti sono quantizzati per simulare il fatto di avere a che fare con immagini matriciali (le coordinate dei pixel devono essere intere).

Il listato 7.4 presenta la classe Matlab utilizzata per modellizzare la dinamica del sistema di visione all'interno dello schema Simulink. Si può notare come tra i campi della classe non vi siano solo i parametri intrinseci della telecamera (riassunti dall'oggetto `cam`), ma anche tutti i parametri relativi alla caratterizzazione dei ritardi, del campionamento e della quantizzazione del dispositivo di visione.

**Codice 7.4:** Classe Matlab per la modellazione del sistema di visione

```

1  classdef Simulink_Camera < handle
2      properties(SetAccess = private)
3          cam; % CentralCamera Object
4          noise;
5          latency;
6          sampleTime;
7          quantized;
8          n_viewed_features;
9      end
10
11     methods
12         function SC = Simulink_Camera(f_mm, pixel_mm, res_x, res_y,
13                                     cam_name, ...
14                                     noise_mu, noise_sigma, latency, sampleTime, quantized
15                                     )
16             SC.cam = CentralCamera('focal', f_mm, 'pixel', pixel_mm,
17                                   ...

```

---

```

15         'resolution', [res_x, res_y], 'centre', [res_x/2, res_y
16             /2], ...
17         'name', cam_name);
18     % Here the initialization of other fields
19 end
20
21 function y = add_gaussian_noise(SC)
22     y = zeros(2, SC.n_viewed_features);
23     for i=1:2
24         for j=1:SC.n_viewed_features
25             y(i,j) = random('norm', SC.noise.mu, SC.noise.
26                 sigma);
27         end
28     end
29
30 function y = add_camera_latency(SC)
31     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
32     ;
33     y = ones(2, SC.n_viewed_features) * latency;
34 end
35
36 function y = add_transform_latency(SC)
37     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
38     ;
39     y = latency;
40 end
41
42 function y = add_transform_latency(SC)
43     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
44     ;
45     y = latency;
46 end
47
48 function y = add_transform_latency(SC)
49     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
50     ;
51     y = latency;
52 end
53
54 function y = add_transform_latency(SC)
55     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
56     ;
57     y = latency;
58 end
59
60 function y = add_transform_latency(SC)
61     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
62     ;
63     y = latency;
64 end
65
66 function y = add_transform_latency(SC)
67     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
68     ;
69     y = latency;
70 end
71
72 function y = add_transform_latency(SC)
73     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
74     ;
75     y = latency;
76 end
77
78 function y = add_transform_latency(SC)
79     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
80     ;
81     y = latency;
82 end
83
84 function y = add_transform_latency(SC)
85     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
86     ;
87     y = latency;
88 end
89
90 function y = add_transform_latency(SC)
91     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
92     ;
93     y = latency;
94 end
95
96 function y = add_transform_latency(SC)
97     latency = SC.latency + SC.latency/2 * floor(rand(1)+0.5)
98     ;
99     y = latency;
100 end

```

---

Si riporta, infine, la funzione di trasferimento completa del sistema di visione (7.39), la quale è stata utilizzata per le considerazioni riguardanti il progetto dei regolatori e le derivanti analisi di stabilità e prestazioni dinamiche del sistema.

$$L_{camera}(s) = \alpha \frac{1 - 0.5\tau_e}{(1 + 0.5\tau_e)(1 + 0.5\tau_{ZOH})} \cdot \quad (7.39)$$

Il parametro  $\alpha$  è un guadagno relativo alla proiezione dei punti sul piano immagine, mentre  $\tau_e$  e  $\tau_{ZOH}$  sono rispettivamente il tempo di elaborazione medio e il periodo di campionamento del sensore di visione (l'inverso del framerate).



### 3 Schemi di controllo in asservimento visivo

Il problema dell'instabilità del controllo di sistemi in *asservimento visivo* è stato messo in luce per la prima volta all'inizio degli anni '80 e attribuito principalmente all'alto tempo di latenza del sistema di acquisizione immagini. Tuttavia, anche ai giorni nostri alcuni sistemi di controllo *vision in the loop* presentano problemi di *lieve instabilità*, a dimostrazione del fatto che l'alta latenza del sistema di visione non è l'unica responsabile di tale instabilità. Infatti, una delle principali difficoltà nel progettare un sistema di controllo in visual servoing risiede nel fatto che il processo non è facilmente modellizzabile e il segnale di controllo (soprattutto se il sistema è basato sul calcolo dello jacobiano immagini) è spesso il risultato di algoritmi di minimizzazione o predizione della traiettoria degli oggetti da inseguire. Un'altra osservazione che di solito viene fatta a tale tipo di sistemi è infatti un ritardo evidente nell'inseguire gli oggetti che si muovono nell'area di lavoro del manipolatore. Per tali motivi, si rende necessario costruire un modello generico di un sistema di controllo in asservimento visivo e, in particolar modo, della sua *dinamica*, in modo che il progetto e sintesi del regolatore vengano fatti in modo da rispettare le specifiche del sistema stesso.

In molti aspetti, le tematiche relative al progetto di controllori per il controllo di robot assistito da sistemi di visione sono molto simili all'ormai ben noto problema del *controllo in forza*: negli anni '60 tali controllori non raggiungevano prestazioni elevate perché non erano disponibili modelli accurati degli attuatori. Nel corso degli anni la ricerca ha portato allo sviluppo di accurati modelli dinamici sia degli attuatori che dei sensori, portando quindi alla realizzazione di controllori più performanti e stabili. Al contrario, attualmente gli algoritmi di controllo in asservimento visivo sono molto semplici e la stabilità è garantita semplicemente dalla convergenza del problema di minimizzazione dell'errore a discapito però della velocità e prestazioni dinamiche ottenute.

Uno dei vantaggi delle tecniche di visual servoing risiede, invece, nella possibilità di definire una linea di demarcazione tra gli aspetti cinematici e gli aspetti dinamici: non importa come vengono estratte le feature e come viene generata la traiettoria da imporre al manipolatore, poiché gli aspetti dinamici sono i medesimi per ogni tipo di approccio. Questo rende molto più facile la modellazione di tali aspetti, dal momento che per l'analisi delle prestazioni dinamiche è sufficiente prendere in considerazione anche un semplice sistema ad un singolo grado di libertà e, successivamente, riportare i risultati ad un sistema molto più complesso.

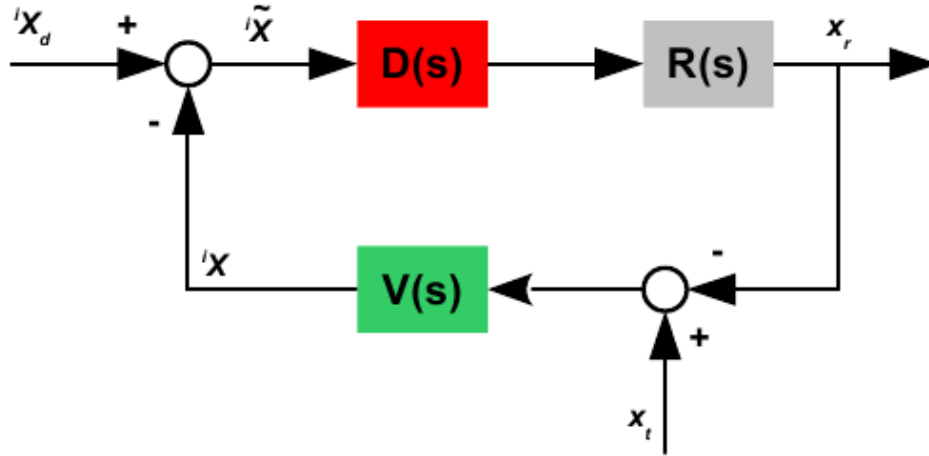
In questa sezione vengono presentati gli schemi di controllo in asservimento visivo che verranno utilizzati nella sezione 4 per l'analisi delle prestazioni del controllo per mezzo del sensore di visione. I sistemi sono suddivisibili in due sottosistemi aventi dinamica ben distinta: il *sensore di visione* e il *controllore del manipolatore* (o meglio dei giunti del manipolatore), la cui dinamica è già stata affrontata rispettivamente nelle sezioni 2 e 1. Vengono presentate due varianti dello stesso schema di controllo che prevedono uno spostamento del punto di vista: in un caso ci si concentra sul target (paragrafo 3.3), mentre nell'altro sul riferimento relativo alle feature del piano immagine (paragrafo 3.2). Infine (paragrafo 3.4) viene introdotto uno schema adatto all'inseguimento di target in movimento tramite l'utilizzo di tecniche di predizione del moto del target stesso.

### 3.1 Schema generale

In generale, si può ritenere che tutti i problemi di visual servoing cerchino di assolvere lo stesso task: l'anello di controllo col sistema di visione in retroazione si sforza di mantenere il centro del target nella posizione desiderata sul piano immagine. Di conseguenza, il set-point è costituito dalla posizione del centroide sul piano immagine, o, meglio, dalla posizione (in pixel) dell'elenco delle feature necessarie a descrivere la posizione del centroide del target. In aggiunta, non è detto che il target sia fermo, pertanto l'eventuale movimento del target deve essere trattato come un disturbo non misurabile. Infine, l'uscita del sistema è il moto del manipolatore. Tutte queste considerazioni danno origine allo schema mostrato in figura 7.30, dove sono evidenziati i singoli sottosistemi che costituiscono il controllo:

- **R**: il controllore del robot che riceve in ingresso i comandi per movimentare l'end-effector in termini di velocità, posizione o coppia.
- **V**: il sensore di visione che permette di "convertire" il movimento del robot in un'informazione comparabile con il riferimento fornito, quindi in pixel (IBVS) o una stima della posizione del target (PBVS). In questo blocco viene considerata anche la dinamica del sistema di visione.
- **D**: il controllore che implementa il visual servoing. Vista la complessità di modellazione dell'intero sistema, si tratta generalmente di controllori PID con la sola componente proporzionale, o, tutt'al più, proporzionale-integrativa.

Considerando un sistema ad un singolo grado di libertà (ad esempio una guida lineare oppure una singola rotazione del polso di un manipolatore), la funzione di trasferimento del



**Figura 7.30:** Visual Servoing: schema di base

robot risulta modellata con la funzione di trasferimento che descrive la risposta dinamica del motore utilizzato per il controllo del suo giunto. Si può notare come il guadagno  $K_{kininv}$  dato dalla soluzione della cinematica inversa, necessaria per il passaggio da coordinate nello spazio di lavoro a coordinate nello spazio dei giunti, non influisca sulla dinamica complessiva del sistema poiché è seguito subito dopo dal guadagno della cinematica diretta ( $K_{kindir}$ ) che è la soluzione al problema inverso del precedente. Ad esempio, se l'algoritmo di controllo richiede di movimentare l'end-effector con set-point di velocità, caso tipico dei controllori in asservimento visivo, si ha un sistema del second'ordine (vedi paragrafo 1.4), riassunto da (7.40).

$$R_W(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n \cdot s + \omega_n^2} \cdot \quad (7.40)$$

Per quanto riguarda la parte di visione, la funzione di trasferimento include i contributi delle non-linearità, ovvero del ritardo d'acquisizione/trasferimento delle immagini e del blocco campionatore/mantenitore, come già descritto nel paragrafo 2) e riassunto dalla relazione

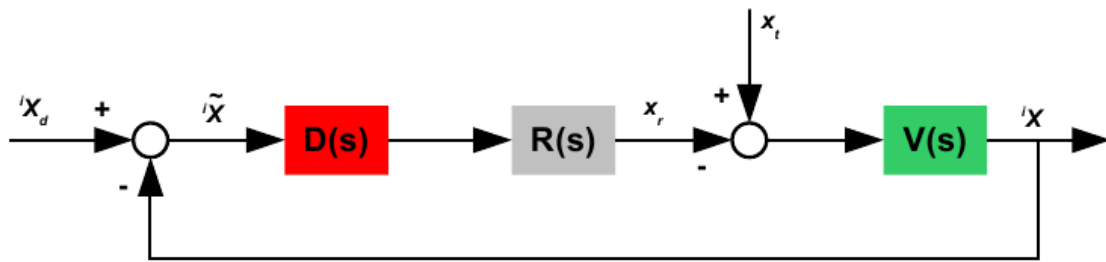
$$V(s) = \alpha \frac{1 - 0.5\tau_e}{(1 + 0.5\tau_e)(1 + 0.5\tau_{ZOH})} \cdot \quad (7.41)$$

Lo schema di controllo appena presentato è utile per introdurre il problema del controllo con asservimento visivo, ma può essere ricombinato in modo da mettere in evidenza il riferimento e l'uscita effettivi del controllore. In particolare, si possono distinguere due diverse funzioni di trasferimento:

- *Funzione di posizionamento.* Evidenzia il legame tra la scena ripresa dalla telecamera e la scena di riferimento (quella che la telecamera dovrebbe osservare a task completato);
- *Funzione di inseguimento.* Mette in evidenza la posizione finale dell'end-effector espressa in termini della scena ripresa dalla telecamera in funzione del riferimento espresso rispetto al sistema di riferimento assoluto (quello dello spazio di lavoro del manipolatore).

### 3.2 Funzione di posizionamento

Lo schema di figura 7.31 è la ricombinazione dello schema in figura 7.30 che permette di mettere in evidenza la funzione di trasferimento per il problema di *posizionamento*.



**Figura 7.31:** Definizione del task di posizionamento delle immagini

La funzione  $\frac{{}^iX(s)}{{}^iX_d(s)}$  descrive la relazione tra la posizione delle feature nel piano immagine durante lo svolgimento del task e la posizione desiderata delle stesse feature. Questa funzione di trasferimento è quella da considerare per problemi in cui la definizione del set-point di riferimento avviene direttamente nel piano immagine (ad esempio se il manipolatore viene “spostato” manualmente nella posizione di riferimento o se semplicemente bisogna mantenere il centro dell’oggetto tracciato nel centro dell’immagine).

Se  ${}^iX_d$  è costante, si può esprimere la relazione tra i due piani immagine, data dalla funzione di trasferimento

$$\frac{{}^iX(s)}{{}^iX_d} = \frac{V(s)R(s)D(s)}{1 + V(s)R(s)D(s)} . \quad (7.42)$$

Per i problemi di visual servoing non esistono delle metriche ben definite per la valutazione delle prestazioni di un controllo. Generalmente, infatti, si tende a dare una valutazione qualitativa dell’assolvimento del compito di posizionamento, del tipo: *l’end-effector ha raggiunto la posizione stabilita in un tempo sufficientemente basso.*

Tradizionalmente le prestazioni di un sistema di controllo in retroazione sono valutate in termini di *banda passante* del controllo. Tuttavia, in un sistema di asservimento visivo la banda passante è fortemente limitata dalla frequenza con cui le immagini vengono elaborate, rappresentando un limite piuttosto che una specifica: al massimo si può arrivare ad una frequenza pari alla metà della frequenza di elaborazione delle immagini. A questo va aggiunto che la scelta di implementazione di un sistema di asservimento visivo “puro” può essere dovuta principalmente a due cause:

- *Inseguimento* di oggetti in moto, dove serve una grande interazione e continuo scambio di messaggi tra sistema di visione e controllore del sistema meccanico;
- *Posizionamento* preciso da raggiungere senza particolari limiti di tempo. In questo caso si utilizzano delle tecniche che mirano all’annullamento (o quanto meno alla diminuzione) degli errori dovuti all’utilizzo di tecniche per la stima della posizione di un oggetto nello spazio di lavoro del robot.

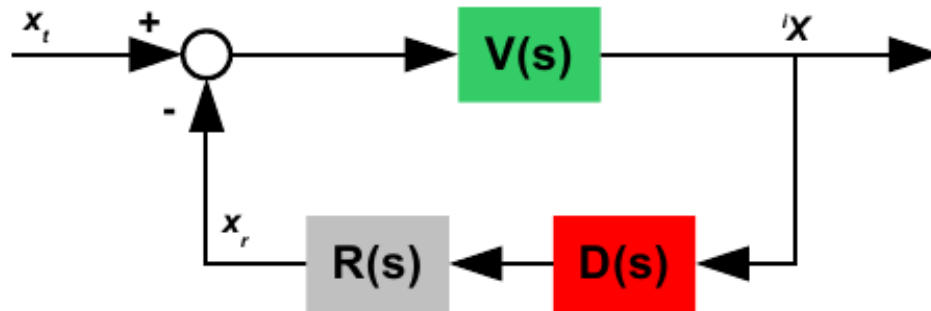
Pertanto, fatte queste considerazioni, per tale classe di problemi risulta più vantaggioso usare una metrica basata sull’*errore di posizionamento* nel piano immagine (valore picco o RMS).

### 3.3 Funzione di inseguimento (tracking)

Nello schema di figura 7.31 il moto del target  $x_t$  viene modellato come un disturbo perché, dal punto di vista del compito di posizionamento, esso agisce come tale. Se, invece, il compito principale del task di controllo consiste nell’inseguimento (*tracking*) di un oggetto in moto, ovvero facendo in modo che l’end-effector del manipolatore si muova come l’oggetto, è meglio considerare una funzione di trasferimento diversa (*funzione di tracciamento*) che permetta di mettere in evidenza la risposta del sistema al moto dell’oggetto. Lo schema a blocchi in figura 7.32 mette in evidenza le funzioni di trasferimento definite da questo task di controllo.

La funzione di trasferimento  $\frac{{}^iX}{x_t}$  descrive l’andamento dell’errore di posizionamento in coordinate del piano immagine in funzione del movimento del target, come specificato dalla relazione

$$\frac{{}^i\tilde{X}(s)}{x_t(s)} = \frac{V(s)}{1 + V(s) \cdot [R(s)D(s)]} , \quad (7.43)$$



**Figura 7.32:** Definizione del compito di inseguimento

dove  $\tilde{X}(s) = {}^i X_d - {}^i X$  è l'errore nel piano immagine.

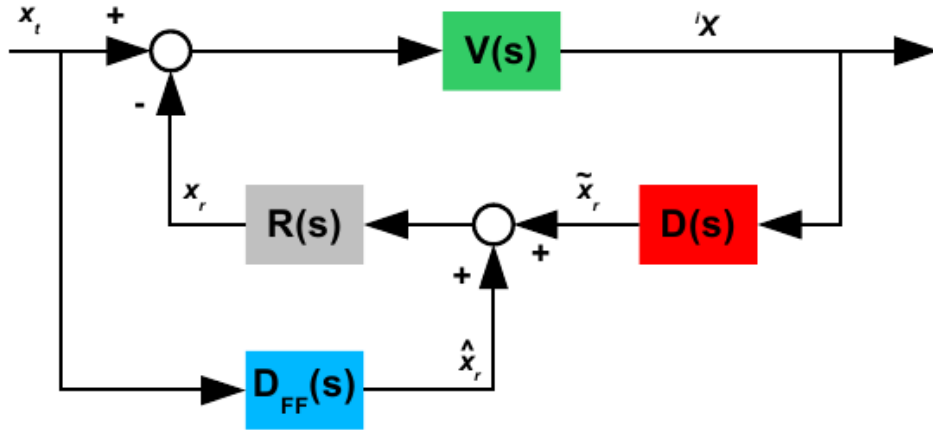
Idealmente, avendo a disposizione un controllore ottimo,  $\left| \frac{{}^i \tilde{X}(s)}{x_t(s)} \right|$  dovrebbe tendere a 0 (zero), ma, in realtà, a causa della minor reattività della dinamica del sistema di visione, essa tende ad aumentare all'aumentare della frequenza. Quindi una prima *misura di prestazione* del controllore è il modulo della funzione di trasferimento dell'errore. Alcuni approcci classici per diminuire l'errore sono:

1. Aumentare il guadagno del controllore  $D(s)$ . Tuttavia, all'aumentare del guadagno il sistema tende rapidamente all'instabilità;
2. Aumentare l'ordine del sistema con l'aggiunta di componenti puramente integrali. Sistemi del primo e second'ordine hanno risposte ad errore nullo rispettivamente ad ingressi a scalino e rampa;
3. Introdurre una componente di *feedforward* sul segnale da tracciare. Comporta una complicazione nell'algoritmo di visione perché prevede l'implementazione di tecniche per la ricostruzione della posa 3D del target e filtri per la stima della velocità.

### 3.4 Feedforward

In letteratura il termine *visual servoing* ha largamente rimpiazzato il termine *visual feedback*. Infatti, sebbene quest'ultimo termine evidenzia ulteriormente il fatto che si tratti di un controllo in retroazione con la componente visuale del sistema posta nel pieno del ramo di retroazione, esso esclude completamente altri utilizzi del sensore di visione. Al giorno d'oggi quasi tutti i sistemi di asservimento visivo rimangono vincolati alla sola chiusura dell'anello di retroazione, ma questi presentano limiti notevoli nei confronti dei task di inseguimento di target in movimento. Tali limiti, inoltre, possono essere parzialmente compensati, ma non eliminati del tutto, tramite la scelta di controllori più performanti

e posizionamento dei poli nelle posizioni desiderate. Tuttavia, quando anche questi accorgimenti non sono sufficienti perché il movimento del target segue traiettorie troppo complesse e imprevedibili, la soluzione risiede nell'aggiunta di un ramo di *feedforward*, come mostrato in figura 7.33. La funzione di trasferimento in anello chiuso (riferita al



**Figura 7.33:** *FeedForward sul comando di moto*

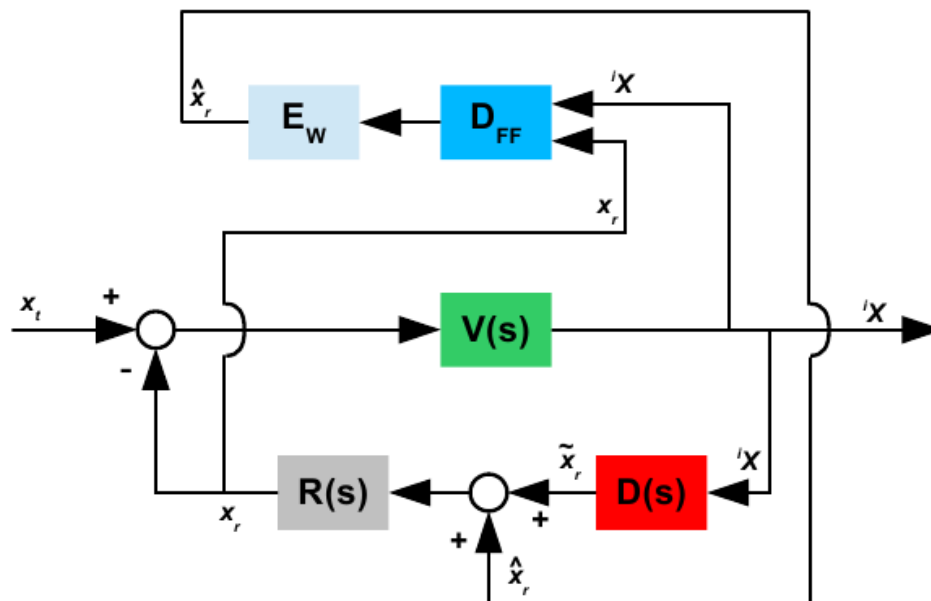
problema dell'inseguimento) risulta:

$$\frac{\hat{x}}{x_t} = \frac{V(s) \cdot [R(s)D_F(s)]}{1 + V(s) \cdot [R(s)D(s)]} \quad (7.44)$$

Che ha lo stesso denominatore (e quindi stessi poli) della funzione di trasferimento relativa al problema dell'inseguimento, con un fattore aggiuntivo al numeratore, dovuto per l'appunto all'aggiunta del compensatore del ramo di feedforward. L'errore tenderebbe a 0 (zero) con  $D_F(s) = R^{-1}(s)$ , ma essendo la funzione di trasferimento del controllore del manipolatore un sistema del second'ordine, tale funzione prevederebbe una doppia derivazione del segnale in ingresso (posizione del target) e, quindi, porterebbe all'incremento del segnale di pilotaggio dei motori (generalmente la tensione), con la conseguenza che il sistema non risulta realizzabile a causa della saturazione delle coppie erogabili dagli attuatori elettrici. Una buona soluzione nel caso il robot venga controllato in velocità consiste nell'utilizzare un singolo derivatore con un guadagno proporzionale alla prontezza che si desidera ottenere (sempre per evitare problemi di saturazione degli attuatori è preferibile scegliere  $\mu \leq 1$ ).

Il problema principale nella realizzazione di tale struttura di controllo risiede nel fatto che, per calcolare il contributo del ramo di feedforward, è necessario conoscere la posizione del target, la quale, ovviamente, fa parte dello stato non osservabile del sistema. La soluzione, pertanto, risiede nello stimare la posizione del target tramite il sistema di

visione (figura 7.34), prevedendone così un ulteriore utilizzo oltre al già affrontato uso nell'anello di retroazione (e giustificando il termine visual servoing). Il blocco che si occupa



**Figura 7.34:** Feedforward sul comando di moto con stima della velocità del target

di stimare la posizione del target deve ricevere in ingresso la posizione della telecamera, un'immagine della scena ripresa e l'informazione della posa relativa telecamera-robot. Date queste informazioni è possibile applicare una tecnica per la ricostruzione 3D della posa dell'oggetto nello spazio di lavoro e utilizzare questo valore stimato come ingresso del compensatore in feedforward. Due delle tecniche che potrebbero venire utilizzate e già descritte nei capitoli 2 e 3:

- *Direct Linear Transform.* Ricostruzione della posa 3D dell'oggetto a partire dall'individuazione di alcune feature specifiche in una scena ripresa da posizione della telecamera nota e successivo confronto con lo stesso set di feature rappresentato in un modello 3D.
- *Visione Stereoscopica.* Utilizzo di due o più telecamere per la stima della profondità di un punto. Il problema di stima è risolvibile tramite il *metodo dei minimi quadrati*.

Come detto in precedenza, il compensatore di feedforward potrebbe presentare solamente una componente derivativa per il calcolo della velocità, ma nel suo calcolo subentrano pesantemente gli errori di quantizzazione dell'immagine proveniente dalla telecamera e le imprecisioni dovute alla mancanza di condizioni ideali per la corretta esecuzione dell'algoritmo di elaborazione delle immagini prescelto. Per questi motivi, il compensatore di feedforward è generalmente costituito da uno stimatore di velocità come i *filtri di Kalman* e  $\alpha - \beta$ .



## 4 Progetto dei regolatori

In generale, il migliore approccio per il progetto di un controllore per un sistema di asservimento visivo consiste nel ridurre il più possibile la latenza del sistema (quindi utilizzare l'algoritmo di estrazione delle feature più efficiente tra tutti quelli adatti alla situazione in esame, migliorare la comunicazione con il controllore del robot, ...), per poi sintetizzare un controllore con le caratteristiche dinamiche desiderate.

### 4.1 Funzione in anello aperto

La funzione d'anello aperto del controllo in asservimento visivo è composta dalla somma dei contributi dinamici dovuti alle componenti del sistema, ovvero:

- Dinamica del sistema di visione, data dalla presenza di un ritardo nella generazione delle informazioni e del loro mantenimento per un certo periodo di tempo. In prima approssimazione, si può ritenere che ritardo e tempo di campionamento coincidano con la *latenza*  $T_l$  del sistema di visione, quindi

$$F_{cam}(s) = \underbrace{\frac{1 - s\frac{T_l}{2}}{1 + s\frac{T_l}{2}}}_{\text{Ritardo}} \cdot \underbrace{\frac{1}{1 + s\frac{T_l}{2}}}_{\text{ZOH}} .$$

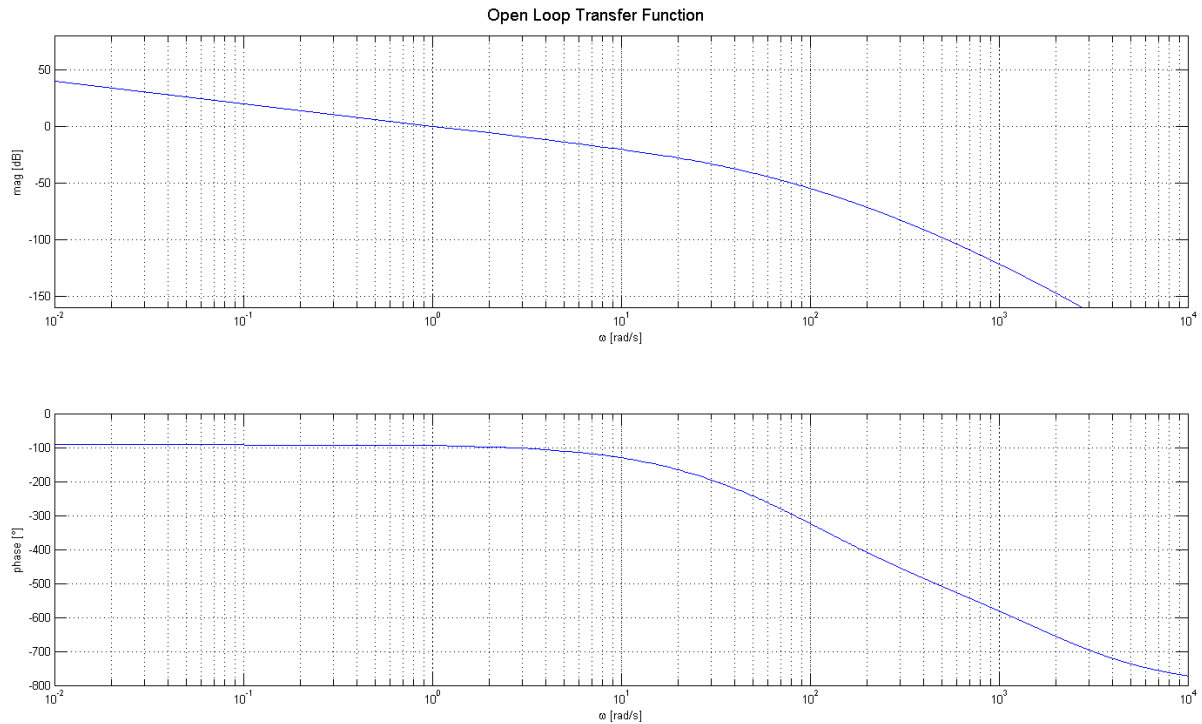
- Dinamica dell'attuatore del moto. In questo caso si possono considerare due termini dati dalla chiusura degli anelli di controllo in coppia e velocità con un doppio controllore PI e da un tempo  $T_c$  di aggiornamento e ritardo di propagazione del segnale di controllo:

$$F_{mot}(s) = \underbrace{\frac{K_v K_i}{s^2 + K_i s + K_v K_i}}_{\text{Controllo}} \underbrace{\frac{1 - s\frac{T_l}{2}}{1 + s\frac{T_l}{2}} \cdot \frac{1}{1 + s\frac{T_l}{2}}}_{\text{Propagazione segnale}} .$$

- Integrazione ( $\frac{1}{s}$ ) del segnale di velocità per ottenere l'informazione dello spostamento subito dalla scena.

La funzione di trasferimento completa è data dall'equazione (7.45), di cui la figura 7.35 mostra i diagrammi di Bode del modulo e della fase, ottenuti ponendo a 1  $kHz$  la frequenza del controllo degli attuatori e 100  $Hz$  la frequenza d'acquisizione del sistema di visione.

$$L_{vs}(s) = F_{cam}(s) \cdot F_{mot}(s) \cdot \frac{1}{s} . \quad (7.45)$$



**Figura 7.35:** Funzione in anello aperto dell'asservimento visivo

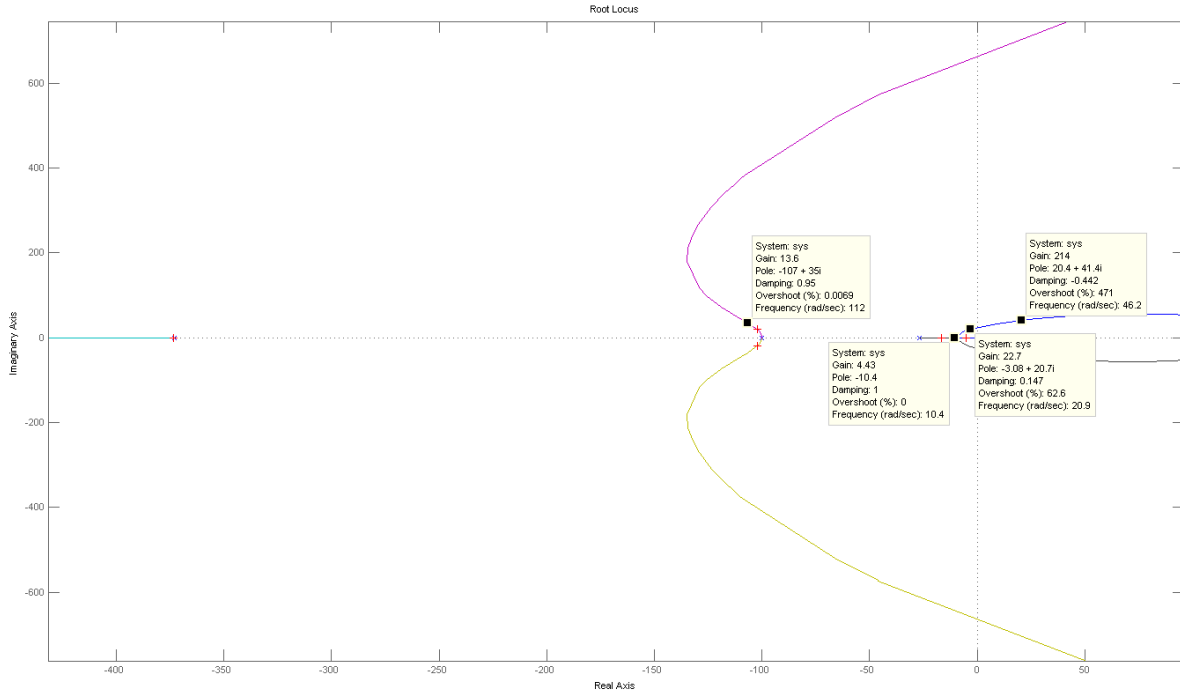
Come si vede, lo sfasamento del sistema è sempre al di sotto di  $-90^\circ$  e diverge appena superata la frequenza di acquisizione delle immagini da parte del sistema di visione, che quindi è dimostrato essere un limite superiore invalicabile per la banda passante del sistema.

## 4.2 Controllo proporzionale

Il primo regolatore preso in considerazione è un semplice controllore proporzionale  $Kp_{vs}$ :

$$L_{vs}^I(s) = Kp_{vs} \cdot F_{cam}(s) \cdot F_{mot}(s) \cdot \frac{1}{s}.$$

Per l'analisi della stabilità del sistema, si è deciso di avvalersi del *luogo delle radici*, mostrato in figura 7.36, dove sono evidenziate le posizioni dei poli e zeri per  $Kp_{vs} = 1$  e le frequenze operative del sistema di visione e del controllo assi sono state poste rispettivamente a  $100\text{ Hz}$  e  $1\text{ kHz}$ . Come si può vedere, per  $Kp_{vs} = 1$  il sistema è stabile, ma già presenta dei poli complessi coniugati che generano piccole sovraelongazioni nella risposta allo scalino. I poli a parte reale negativa con modulo minore, ovvero quelli più vicini all'asse delle ordinate, sono quelli imputabili alla dinamica del sistema di visione. Infatti, utilizzando un sistema di visione più lento l'effetto che si andrebbe ad osservare sarebbe un ulteriore



**Figura 7.36:** Controllore Proporzionale: luogo delle radici

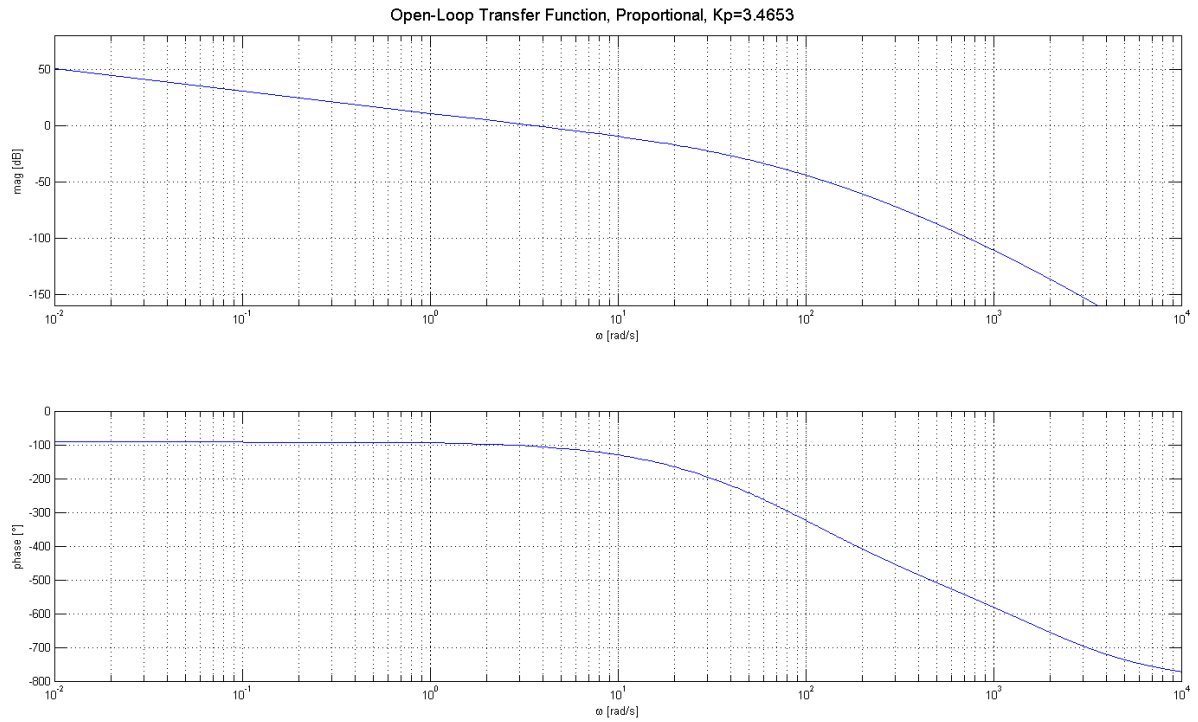
avvicinamento all'asse delle ordinate, il quale andrebbe a ridurre ulteriormente la banda passante del controllo. L'altro problema della componente dinamica dovuta al sistema di visione è data dal fatto che, all'aumentare del guadagno proporzionale del controllo, i poli diventano molto velocemente complessi coniugati, per poi diventare a parte reale positiva, rendendo quindi il sistema instabile. Il limite di guadagno oltre cui il sistema preso in considerazione è  $Kp_{vs} = 20$ .

Infine, si mostra il diagramma di Bode della funzione in anello aperto con il controllore in serie ( $Kp_{vs} \approx 3$ ). La scelta di tale controllore ha permesso di alzare la frequenza propria del sistema al massimo valore possibile e che permetta di mantenere un margine di fase accettabile.

### 4.3 Controllo proporzionale-derivativo

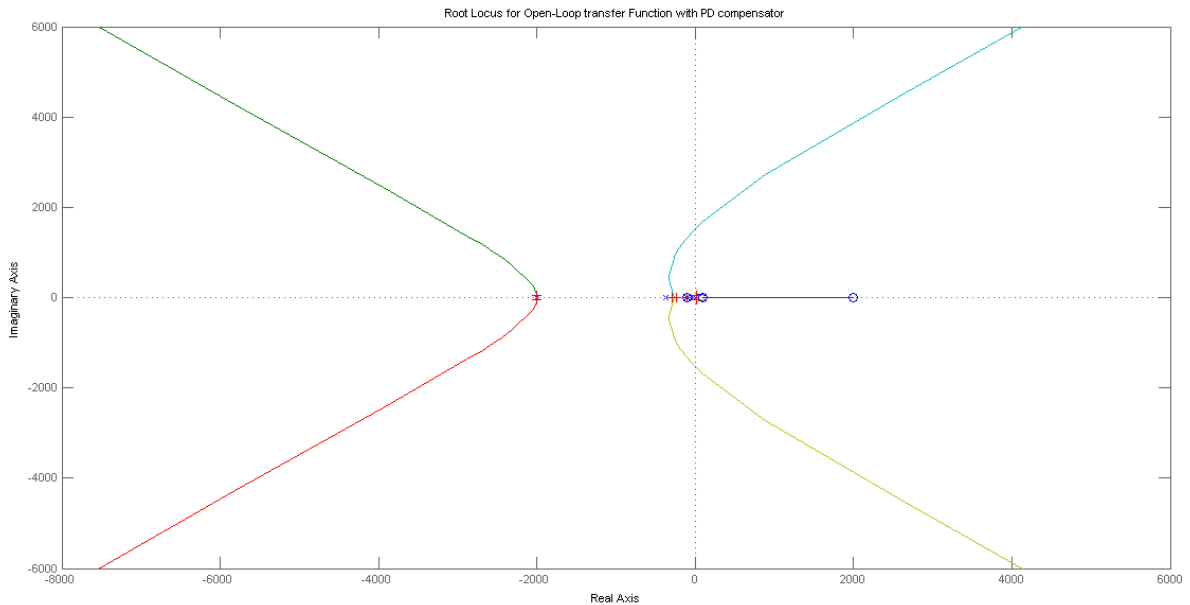
L'unico problema individuato nello schema di controllo con regolatore proporzionale riguarda l'alto indice di instabilità del polo dovuto ai ritardi introdotti dal sistema di visione, ovvero dal dispositivo più lento introdotto nel controllo. Questo effetto può essere ridotto introducendo una componente derivativa nel controllo:

$$L_{vs}^I(s) = (Kp_{vs} + s \cdot Kd_{vs}) \cdot F_{cam}(s) \cdot F_{mot}(s) \cdot \frac{1}{s}.$$



**Figura 7.37:** *Controllore proporzionale: funzione in anello aperto*

In figura 7.38 viene mostrato come si modifica il luogo delle radici introducendo un opportuno contributo derivativo. La parte derivativa ha lo scopo di ridurre gli effetti del

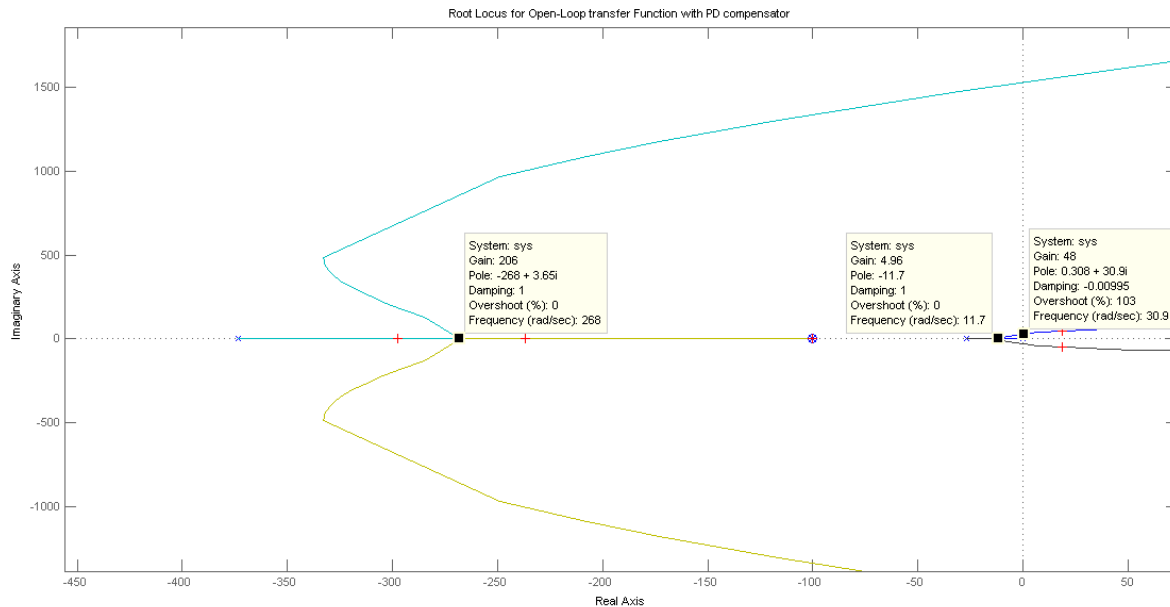


**Figura 7.38:** *Controllore Proporzionale-Derivativo: luogo delle radici*

ritardo e mantenimento dei segnali generati dal sistema di visione. A tale scopo, si è posto

$$\begin{cases} Kp_{vs} = 1 \\ Kd_{vs} = Kp_{vs} \cdot \frac{T_l}{2} \end{cases}$$

in modo da andare ad eliminare il polo con dinamica più lenta. La dinamica relativa alle componenti in frequenza più elevate rimane analoga a quella ottenuta col controllore proporzionale. L'effetto del controllo derivativo si nota, invece, sulla dinamica dell'altro polo. In figura 7.39 è riportato il dettaglio del luogo della radice per quanto concerne il polo a bassa frequenza.



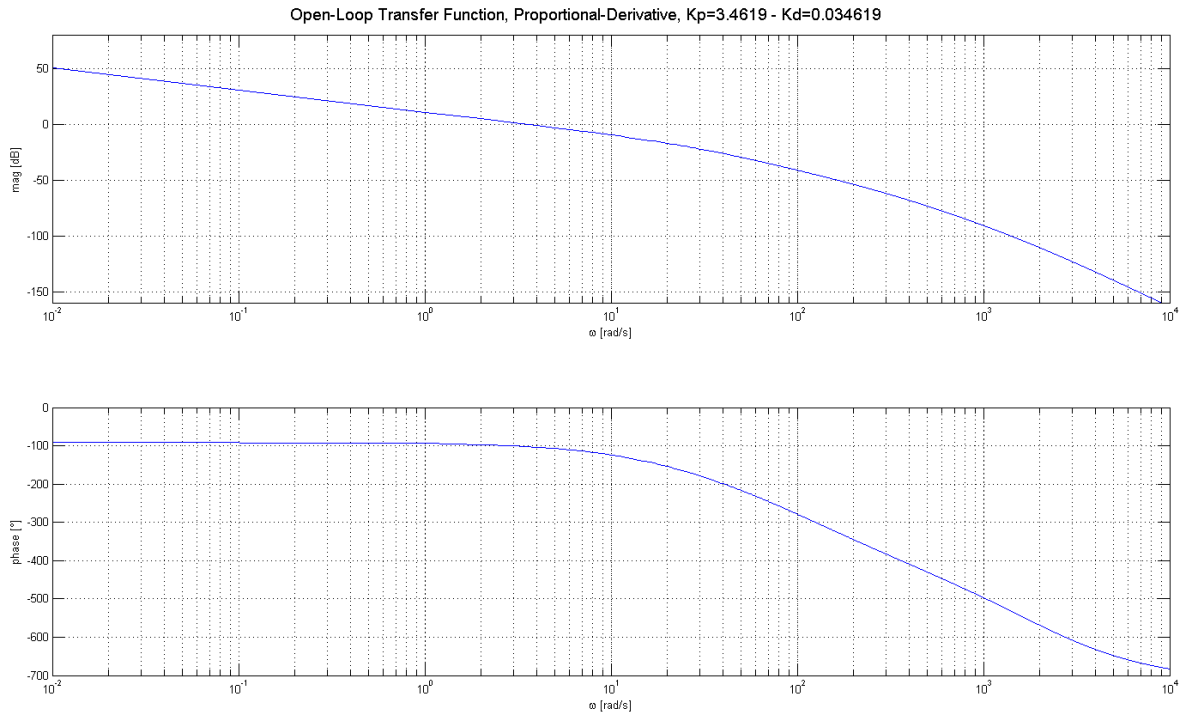
**Figura 7.39:** Controllore Proporzionale-Derivativo: luogo delle radici, dettaglio

La componente derivativa introduce un polo e uno zero. La parte relativa al polo non influisce sul controllo, poiché esso rimane puramente reale mantenendo il sistema stabile per guadagni proporzionali anche elevati. Al contrario, sulla componente dovuta alla latenza del sistema di visione vengono mantenuti poli puramente reali per guadagni simili al caso del controllore proporzionale, ma, all'aumentare del guadagno il sistema rimane stabile più a lungo. Di conseguenza, questo permette di raggiungere prestazioni dinamiche più elevate.

Per completezza, in figura 7.40 è riportato il diagramma di Bode della funzione di trasferimento in anello aperto ottenuta. Le considerazioni fatte in precedenza relativamente al controllo proporzionale sono ancora valide, con la frequenza propria del sistema in grado di essere impostata ad un valore leggermente più alto.

## 4.4 Feedforward

L'analisi del controllo proporzionale-derivativo permette di fare alcune considerazioni sul motivo alla base di questo effetto. Infatti, l'aggiunta della componente derivativa al



**Figura 7.40:** *Controllore proporzionale-derivativo: funzione in anello aperto*

controllore equivale ad introdurre un controllo sulla derivata del segnale di riferimento; nel caso del controllo in asservimento visivo, questo è uno spostamento, per cui il calcolo della derivata permette di ottenere un'informazione sulla velocità di movimento della scena. Ciò permette di trarre la conclusione che un modo per aumentare le prestazioni dinamiche del sistema è dato dall'introduzione di una componente di feedforward ottenuta attraverso la stima, da parte del sistema di visione, della velocità di movimento della scena.

#### 4.4.1 Filtraggio: stima della velocità

Le tecniche di asservimento visivo che si basano sull'utilizzo di jacobiani immagine (IBVS) permettono di ottenere i profili di velocità da applicare all'end-effector del manipolatore. Pertanto, per applicare un controllo in feedforward non è sufficiente avere informazioni riguardo alla posizione dell'oggetto d'interesse rispetto alla telecamera, ma bisogna provvedere a stimare la velocità dell'oggetto stesso. Di seguito vengono descritti alcuni filtri utilizzabili per ottenere tale stima a partire dall'informazione relativa alla posa dell'oggetto identificato nelle immagini.

- *Derivazione.*

L'approccio più semplice per la stima della velocità è dato dalla semplice operazione

di derivazione. Tale operazione è esprimibile per mezzo della seguente funzione di trasferimento a tempo discreto:

$$\frac{\hat{V}(z)}{Y(z)} = \frac{1}{T} \frac{z-1}{z} .$$

Fondamentalmente anche la stima del flusso ottico da parte del sistema di visione è basata sulla differenza del primo ordine tra l'intensità dell'immagine in due frame successivi.

- *Filtri  $\alpha$ - $\beta$  e  $\alpha$ - $\beta$ - $\gamma$ .*

Il filtro  $\alpha - \beta$  di base è basato sull'assunzione che l'accelerazione dell'oggetto di cui calcolare la velocità sia nulla. In questo caso, le relazioni del filtro sono date da:

$$\begin{cases} \hat{x}_{p_{k+1}} = \hat{x}_k + T\hat{v}_k \\ \hat{x}_{k+1} = \hat{x}_{p_{k+1}} + \alpha [y_{k+1} - \hat{x}_{p_{k+1}}] \\ \hat{v}_{k+1} = \hat{v}_k + \frac{\beta}{T} [y_{k+1} - \hat{x}_{p_{k+1}}] \end{cases} .$$

Manipolando in modo opportuno queste equazioni, è possibile ottenere la forma in funzione di trasferimento del filtro:

$$\frac{\hat{V}(z)}{Y(z)} = \frac{\beta}{T} \frac{z(z-1)}{z^2 + (\alpha + \beta - 2)z + 1 - \alpha}$$

da cui si ottiene che la velocità è ottenuta con una differenza del primo ordine filtrata da un sistema del second'ordine parametrizzato con  $\alpha$  e  $\beta$ . Di solito un solo parametro viene lasciato libero, mentre l'altro dipende da esso, secondo:

$$\beta = 2 - \alpha - 2\sqrt{1 - \alpha} .$$

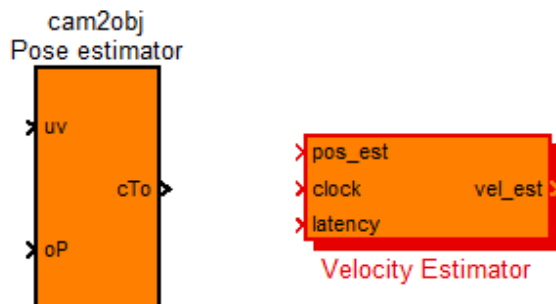
- *Filtro di Kalman.*

In questo paragrafo non si ha intenzione di presentare in dettaglio le equazioni del filtro di Kalman, ma solamente di utilizzarne la funzione di trasferimento ottenuta considerando la forma semplificata del filtro. Questa è data da:

$$\frac{\hat{V}(z)}{Y(z)} = \frac{k_2 z(z-1)}{z^2 + (k_1 - 2)z + (k_2 - k_1 + 1)}$$

dove  $k_1$  e  $k_2$  sono gli elementi della matrice di guadagno  $\mathbf{K}$  del filtro. La forma della funzione di trasferimento è la stessa di quella ottenuta col filtro  $\alpha - \beta$ , con l'unica differenza che il filtraggio definito dai parametri  $k_1$  e  $k_2$  non è costante, ma varia nel tempo.

In figura 7.41 viene mostrato il blocco Simulink utilizzato per inserire la componente di feedforward negli schemi presentati nel corso della trattazione.



*Figura 7.41: Blocco Simulink per la stima della velocità degli oggetti target*

## 4.5 Funzioni in anello chiuso

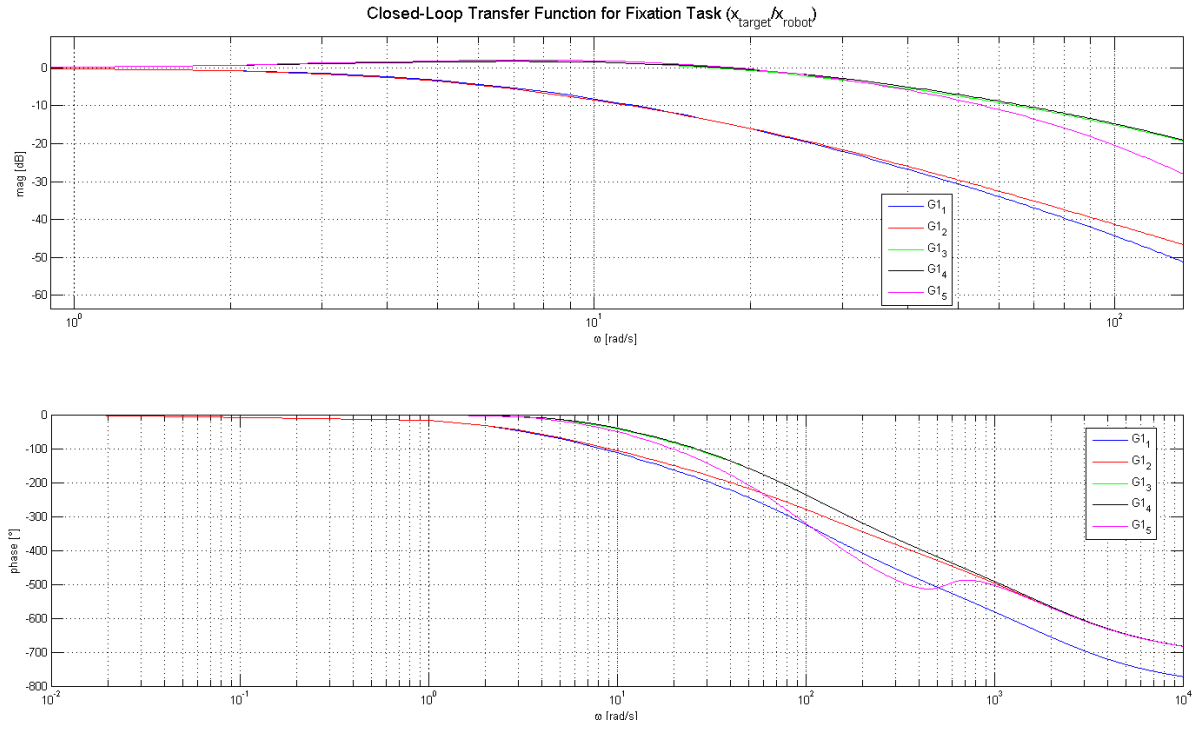
Prima di passare a descrivere le simulazioni effettuate su un sistema da controllare in asservimento visivo, si procede al confronto delle varie funzioni in anello chiuso ottenute seguendo i differenti approcci di controllo presentati nel corso della trattazione. In particolare, ciò viene fatto mostrando i diagrammi di Bode delle differenti funzioni di trasferimento ottenute. Per il task di *posizionamento*  $G_1$ , si considera:

- $G_1^1$ . Controllo proporzionale, senza feedforward.
- $G_1^2$ . Controllo proporzionale-derivativo, senza feedforward.
- $G_1^3$ . Controllo proporzionale, con feedforward differenziale.
- $G_1^4$ . Controllo proporzionale-derivativo, con feedforward differenziale.
- $G_1^5$ . Controllo proporzionale-derivativo, con feedforward realizzato tramite filtro  $\alpha - \beta$ .

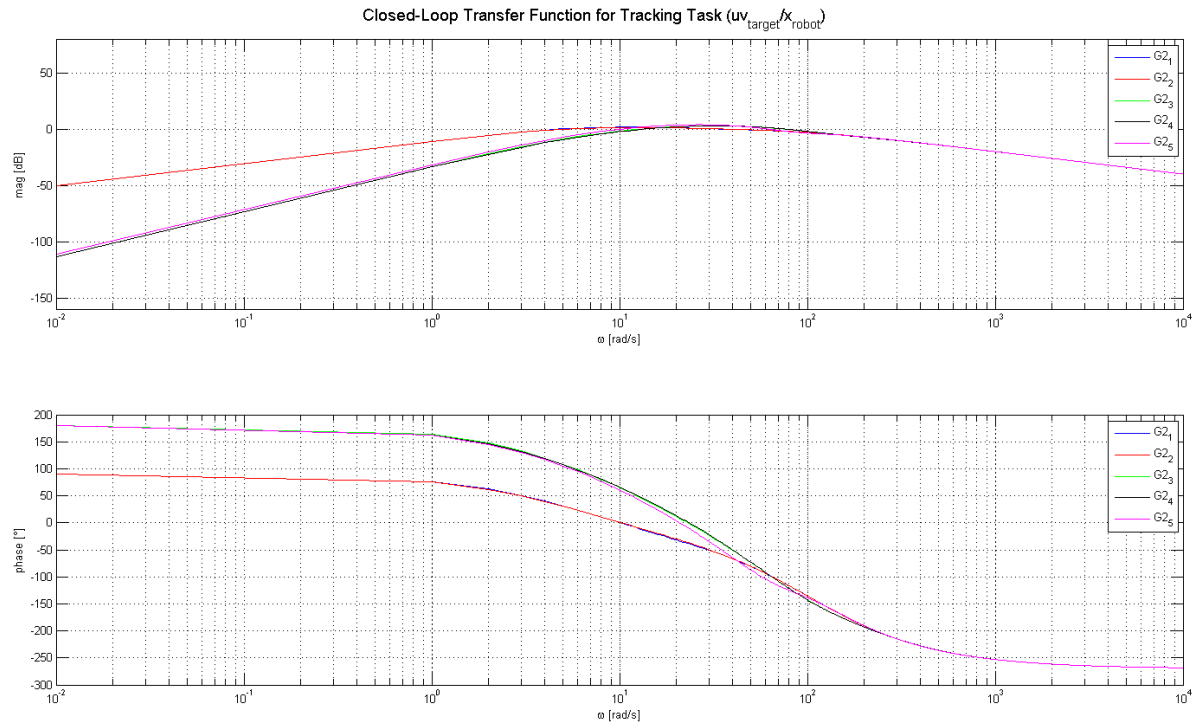
Il diagramma di Bode delle funzioni di trasferimento appena descritte è mostrato in figura 7.42. Si può notare come l'introduzione della componente di feedforward porta benefici all'aumentare della frequenza. Relativamente al problema dell'asservimento visivo, non è interessante analizzare il comportamento a tutte le frequenze, ma solo al di sotto delle massime frequenze operative dei sistemi di visione, quindi circa  $100\text{ Hz}$ . Detto questo, la scelta della tecnica di derivazione non influisce in particolar modo sulla dinamica del sistema per il task di posizionamento.

Gli stessi schemi di controllo utilizzati per il calcolo delle funzioni di trasferimento del task di posizionamento sono state utilizzate per estendere il ragionamento al task di *tracciamento/inseguimento*  $G_2$ , i cui diagrammi di Bode sono mostrati in figura 7.43. In questo caso si nota come l'introduzione del ramo di feedforward permette di ottenere un sistema più pronto a rispondere ad eventuali spostamenti dell'oggetto da inseguire.





*Figura 7.42: Funzione di trasferimento in anello chiuso: posizionamento*



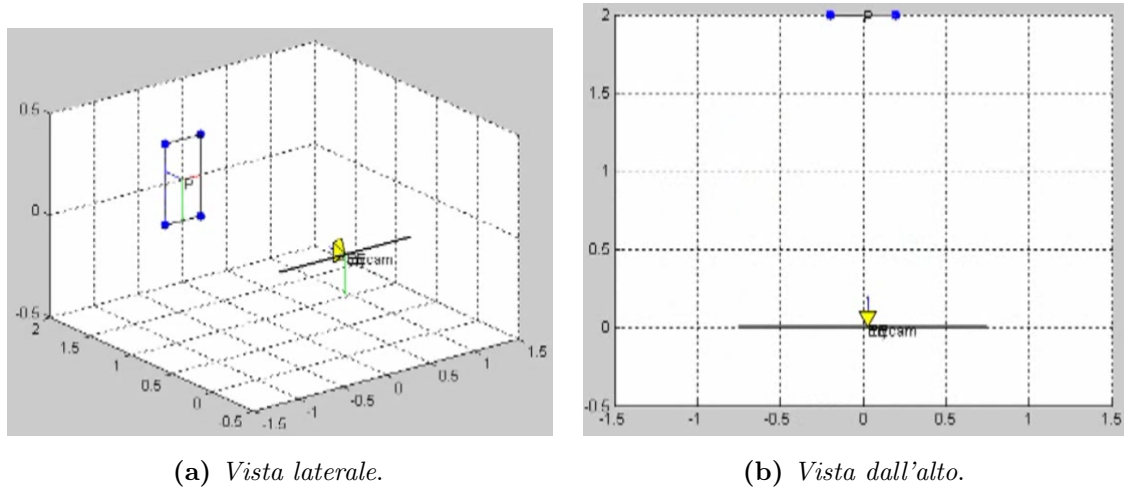
*Figura 7.43: Funzione di trasferimento in anello chiuso: inseguimento*

## 5 Simulazione asse lineare con controllo IBVS

Per poter fornire una prima validazione degli schemi di controllo e relativi regolatori presentati è stata approntata la simulazione dello schema di controllo di un semplice

sistema ad un grado di libertà. La scelta di un sistema di questo tipo permette, infatti, di separare le tematiche del controllo in asservimento visivo da quelle del controllo del sistema meccanico stesso.

La figura 7.44 mostra la schematizzazione del sistema visualizzato così come visualizzato dallo strumento di simulazione utilizzato.



**Figura 7.44:** Sistema ad un grado di libertà simulato

Il sistema meccanico è costituito semplicemente da un carrello montato su una guida lineare. Il moto è trasmesso dal motore elettrico al carrello per mezzo di una trasmissione a cinghia. Pertanto, detto  $R$  il raggio della puleggia su cui la cinghia è installata, la cinematica del sistema risulta definita dalle seguenti relazioni:

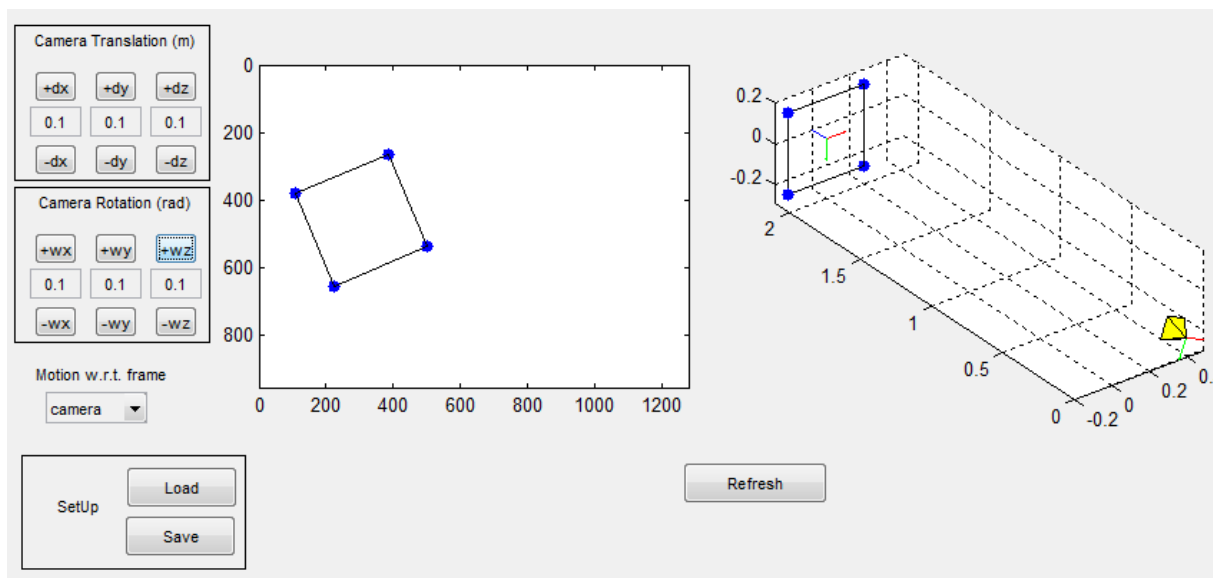
$$\begin{cases} x_c = 2\pi R q_m \\ \dot{x}_c = 2\pi R \dot{q}_m \\ \ddot{x}_c = 2\pi R \ddot{q}_m \end{cases} .$$

Il controllo in velocità del carrello viene quindi raggiunto controllando in velocità l'unico motore coinvolto nella generazione del moto.

La telecamera viene montata sul carrello, il quale può essere considerato a tutti gli effetti come l'end-effector del robot da muovere. Il task di controllo in asservimento visivo è definito nel modo seguente: dato un oggetto posto di fronte alla telecamera e mosso nella stessa direzione, il carrello deve essere movimentato per fare in modo che il centro geometrico dell'oggetto sia sempre posizionato nel centro dell'immagine ripresa dalla telecamera. In figura 7.44 e in simulazione l'oggetto è rappresentato e completamente descritto da quattro punti per permetterne l'estensione e utilizzo per un sistema di controllo a più gradi di libertà, ma in realtà l'informazione del solo centro geometrico dell'oggetto è

più che sufficiente a definire l'errore di posizionamento dei task di posizionamento ad un solo grado di libertà.

Avendo introdotto la tipologia di oggetto utilizzata per le simulazioni, in figura 7.45 è mostrato lo strumento sviluppato per l'inizializzazione del task di posizionamento o inseguimento. Tramite esso, infatti, è possibile definire la posa relativa tra oggetto e telecamera da utilizzare come configurazione iniziale e come posizione di riferimento desiderata.



**Figura 7.45:** Interfaccia per l'inizializzazione della simulazione del task di asservimento visivo

Mantenendo e portando avanti l'analogia tra il carrello e l'end-effector del manipolatore, si può concludere che il controllo in asservimento visivo realizzato rientra nella categoria dei controlli basati su immagine (*IBVS*) *eye-in-hand*. I task analizzati sono i principali compiti realizzabili con un sistema di questo tipo:

- *Posizionamento*. L'oggetto viene mantenuto fermo nel tempo; il carrello è posizionato inizialmente fuori asse rispetto al centro geometrico dell'oggetto; il carrello viene allineato con l'oggetto.
- *Inseguimento di oggetti in movimento*. A differenza del caso precedente, l'oggetto viene messo in movimento. Il carrello deve essere pilotato in modo che la telecamera insegua l'oggetto. Questa situazione fa riferimento alla tipica applicazione in cui un manipolatore debba prelevare gli oggetti movimentati da una nastro trasportatore. Solamente ai fini della simulazione, avendo la guida lunghezza finita, è stato imposto all'oggetto un movimento periodico. Questo ha permesso innanzitutto di evidenziare

la differenza tra il primo raggiungimento dell'oggetto e il successivo mantenimento della traiettoria. In secondo luogo, sono stati imposti movimenti periodici di tipo differente, tali da mettere in risalto il comportamento del sistema alle variazioni istantanee del verso del moto. Infatti, sono le variazioni di direzione del movimento a causare l'errore di inseguimento più grande, poiché i filtri per il calcolo della velocità non sono in grado di prevedere l'inversione del moto.

## 5.1 Posizionamento IBVS

Il semplice task di posizionamento non richiede la realizzazione di un controllore con prestazioni dinamiche elevate. Pertanto, è stato utilizzato esclusivamente il regolatore proporzionale. Il task di posizionamento di quest'applicazione permette di valutare la risposta allo scalino del sistema. La condizione iniziale del sistema è stata individuata spostando il carrello a sinistra di  $0.6\text{ m}$  rispetto all'oggetto target.

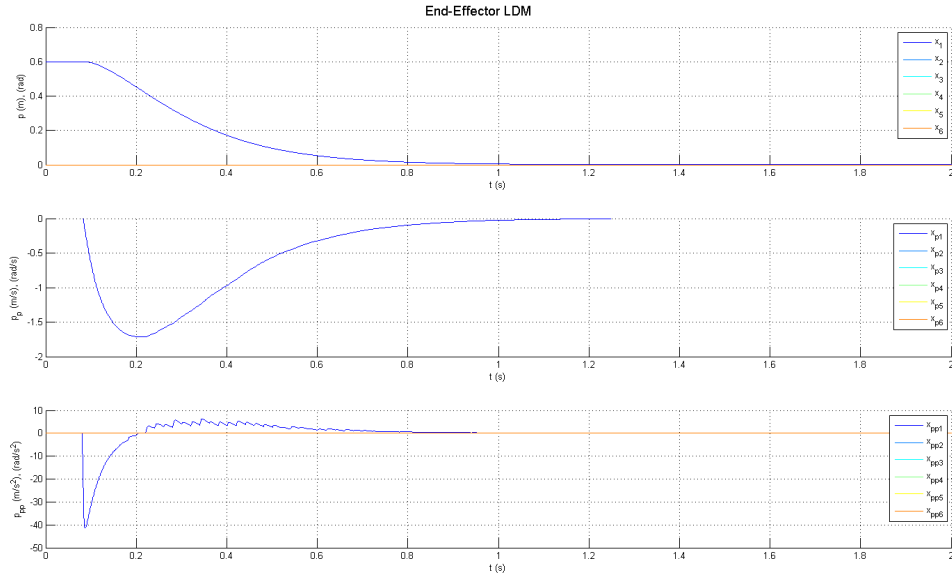
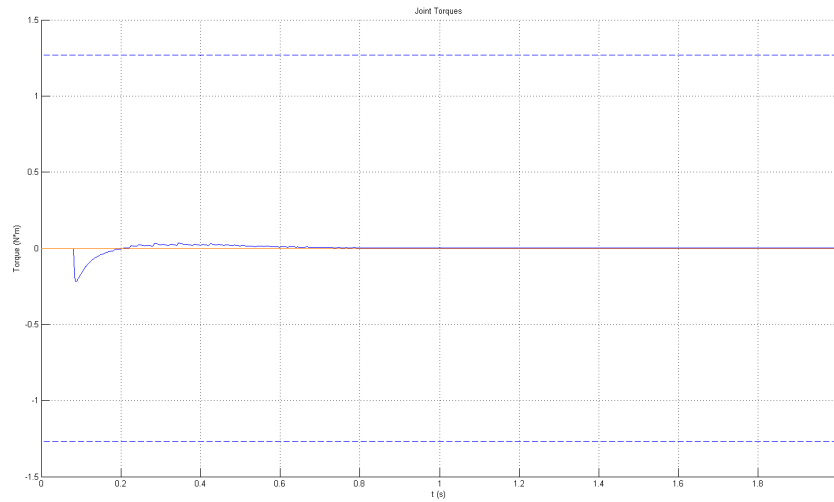
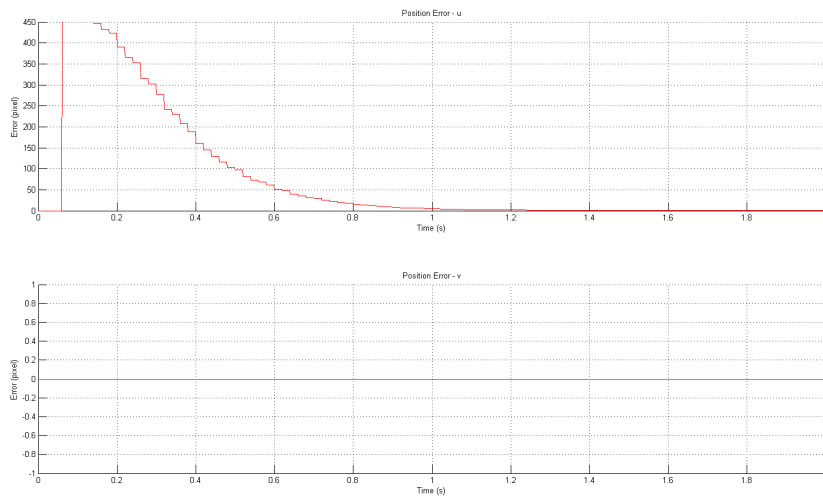
In figura 7.46 è visualizzata la risposta allo scalino del sistema controllato con guadagno  $\mu = 3$ , ovvero tale da garantire l'assenza di poli complessi coniugati. La risposta allo scalino è pulita, non presenta sovraelongazioni e l'errore viene portato ad essere nullo all'incirca in 60 frame. La visualizzazione della coppia permette di mostrare che la legge di moto risultante è ottenibile col motore selezionato.

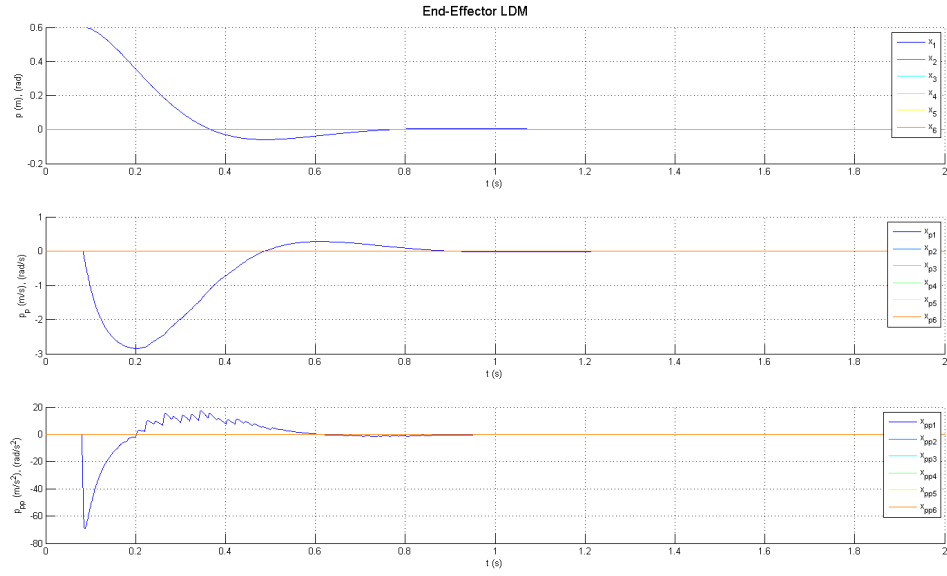
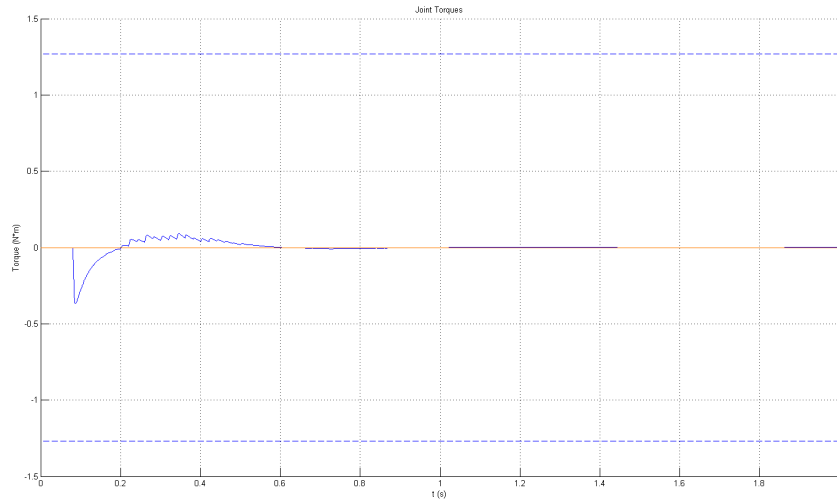
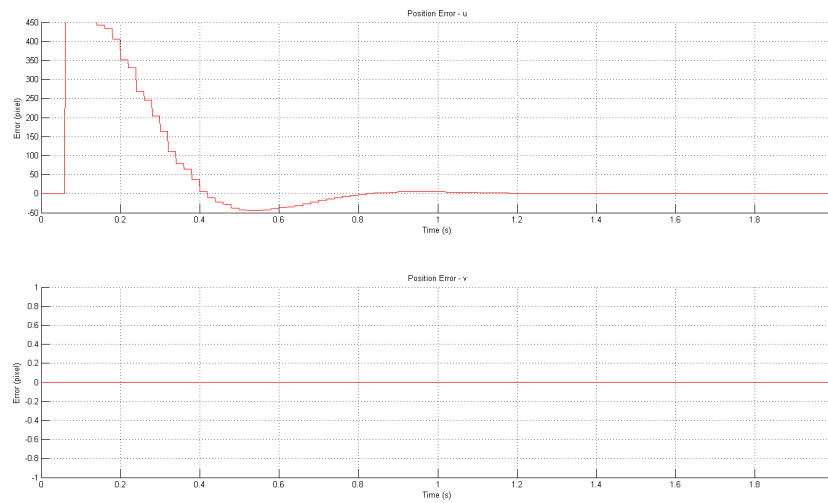
Alzando il guadagno del controllore, ad esempio  $\mu = 5$  in figura 7.47, il sistema acquisisce prestazioni dinamiche più elevate, a discapito della pulizia della traiettoria seguita. Infatti, si è superato il limite superiore di guadagno che garantisce l'assenza di poli complessi coniugati. La risposta allo scalino inizia a presentare oscillazioni. In pratica, il carrello tende a superare l'oggetto, per poi tornare a centrarsi su di esso. Questo non porta a particolari problemi se c'è ancora sufficiente spazio a permettere il movimento della guida lineare, mentre in assenza di spazio sufficiente porterebbe ad un comportamento anomalo. Questa situazione si verifica quando l'oggetto target viene posizionato in corrispondenza degli estremi della guida lineare: con questo controllo il carrello va inevitabilmente a sbattere contro i finecorsa.

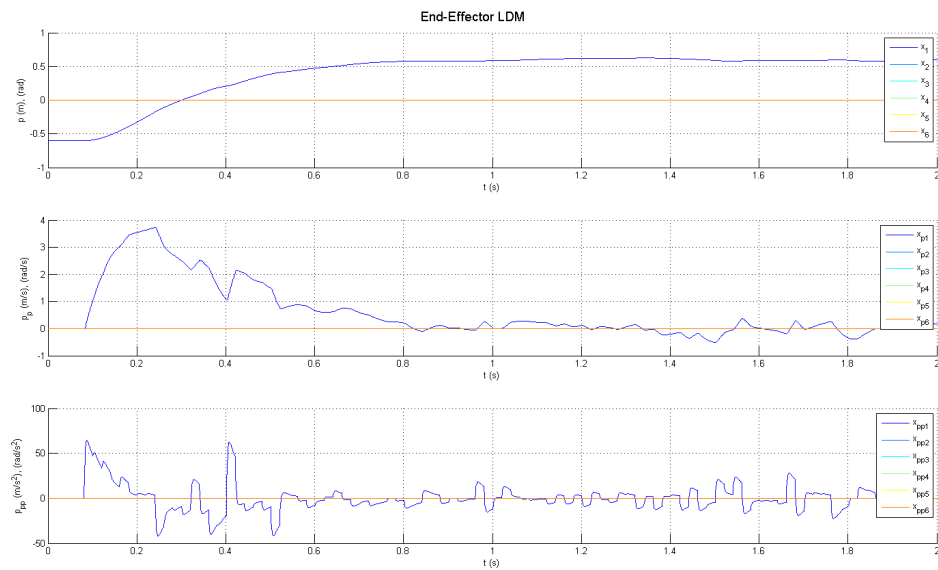
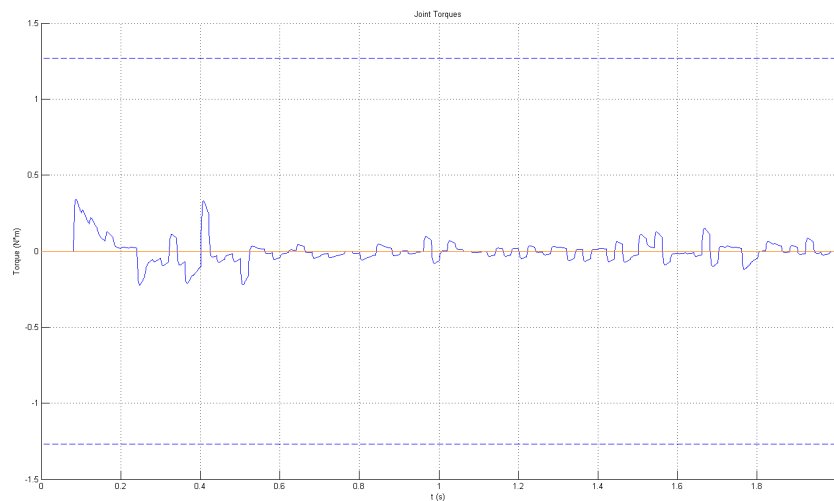
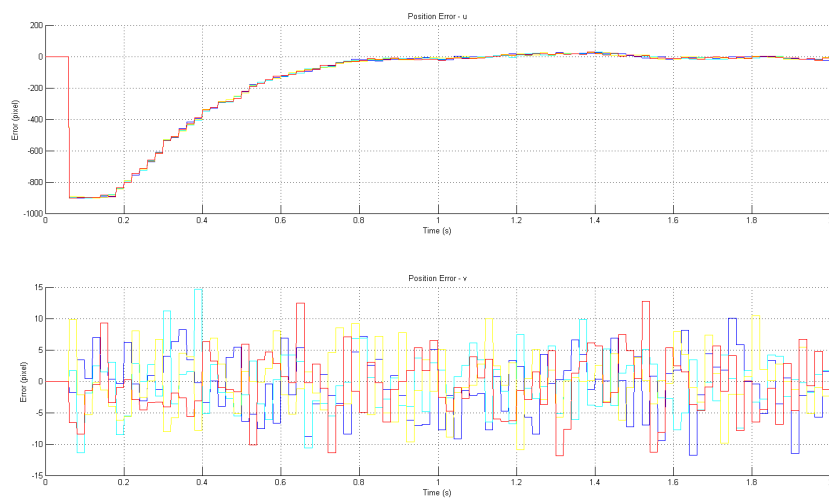
Osservato il corretto funzionamento del sistema, si è cercato di valutarne il comportamento in presenza dell'altra componente non-lineare introdotta dal sistema di visione: la presenza di rumore nelle misure. Introducendo un apposito rumore gaussiano, come in figura 7.48, si arriva a dimostrare che il controllo opera ancora correttamente, ma esso rimane un po' più nervoso. Per tale motivo, nel caso l'algoritmo di elaborazione immagine

---

introduca sempre un po' di errore nelle misure, è necessario studiare delle tecniche per limitarne l'effetto sulla legge di moto da imporre all'attuatore.

(a) *Profilo di velocità.*(b) *Coppia.*(c) *Errore di posizionamento.***Figura 7.46:** Guida lineare: risposta scalino algoritmo IBVS ,  $\mu = 3$

(a) *Profilo di velocità.*(b) *Coppia.*(c) *Errore di posizionamento.***Figura 7.47:** Guida lineare: risposta scalino algoritmo IBVS ,  $\mu = 5$

(a) *Profilo di velocità.*(b) *Coppia.*(c) *Errore di posizionamento.***Figura 7.48:** Guida lineare: risposta scalino algoritmo IBVS ,  $\mu = 3$  , con rumore

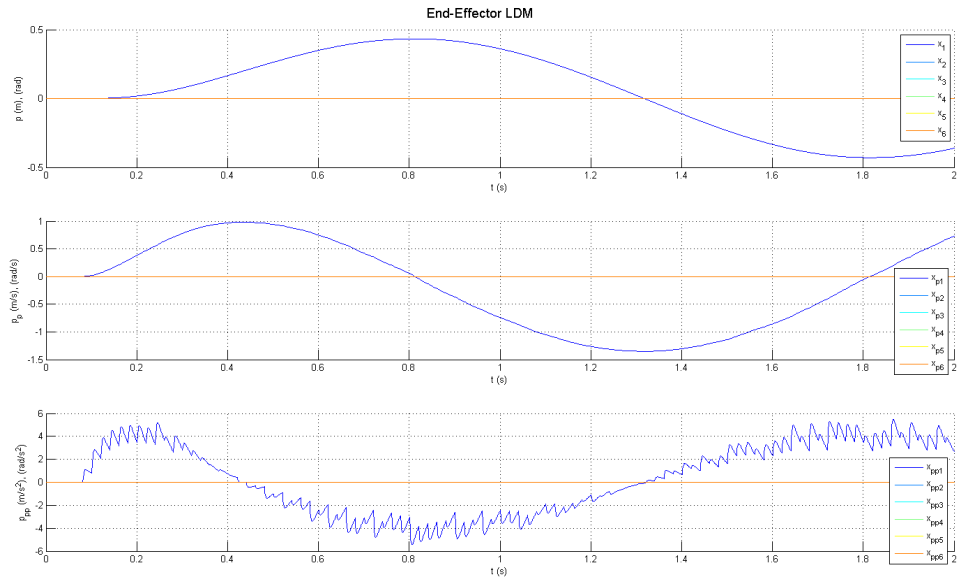
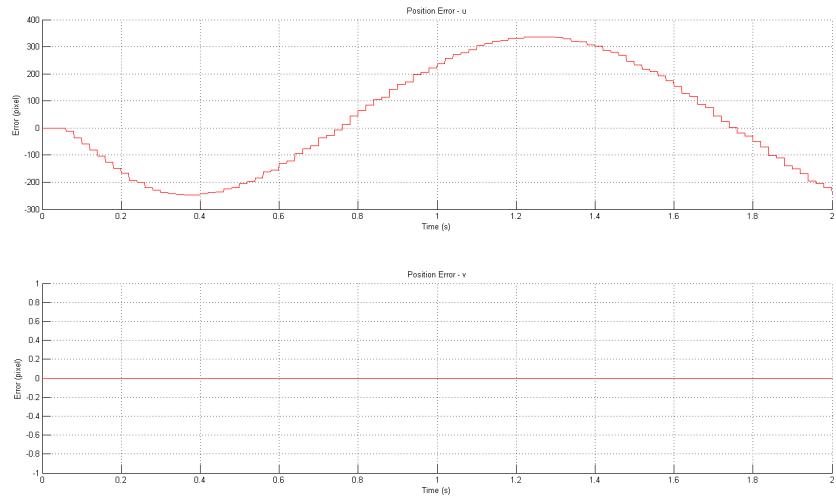
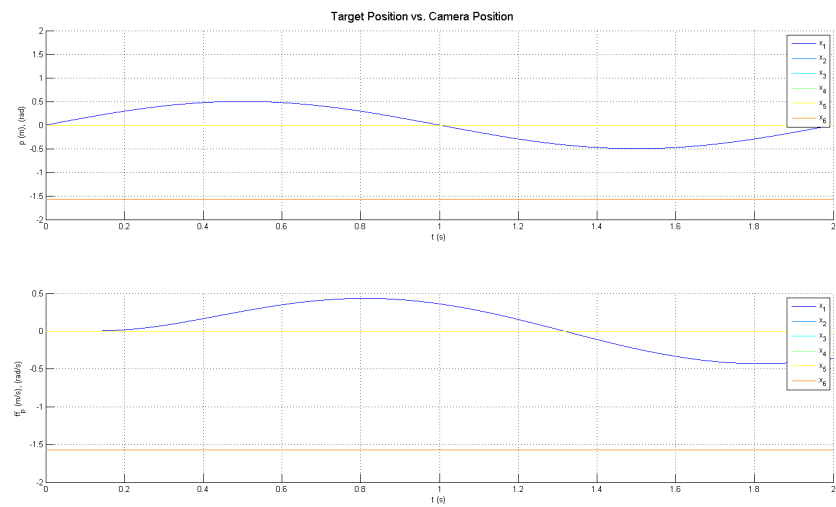


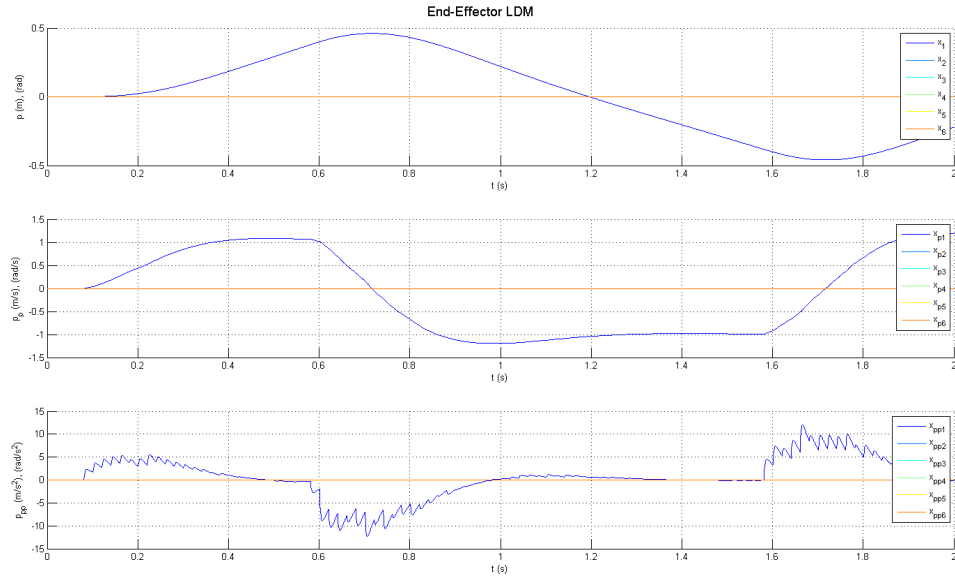
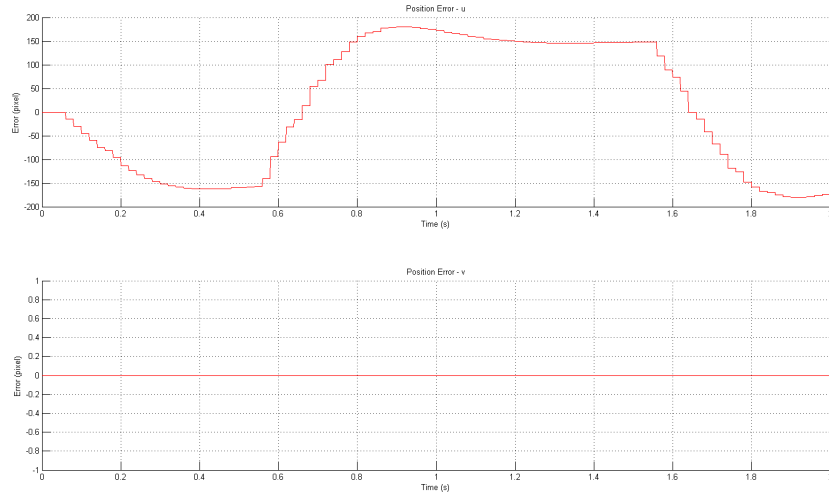
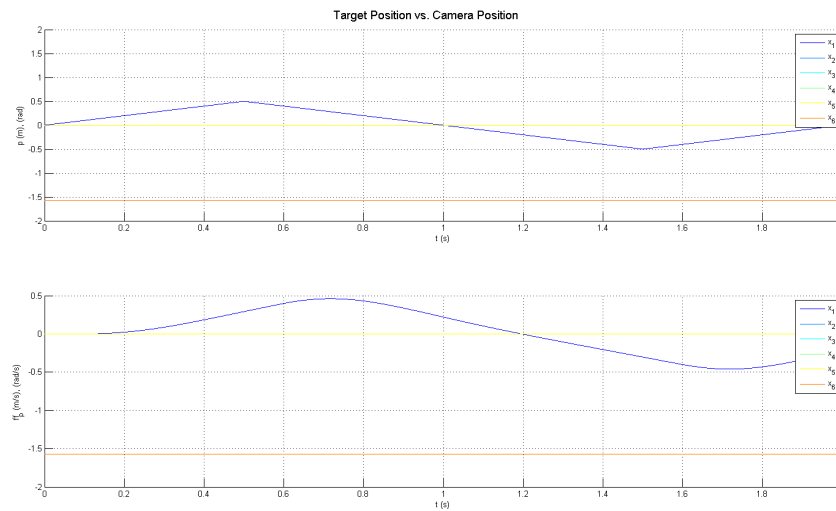
## 5.2 IBVS: inseguimento

Per mettere alla prova il controllo implementato, il passo successivo ha previsto di studiarne il comportamento in presenza di un moto dell'oggetto. In particolare, si sono presi in considerazione due tipologie di moto: un profilo sinusoidale e uno triangolare.

In figura 7.49 è mostrata la risposta all'inseguimento di una traiettoria sinusoidale. L'effetto che balza subito all'occhio dall'analisi dell'andamento dell'errore è il fatto che il carrello è sempre in ritardo. L'errore diventa nullo per un istante nel tratto intermedio dello spostamento da un lato all'altro della guida, ma torna subito ad essere rilevante dopo l'inversione del moto. L'inseguimento non è quindi perfetto, ma la traiettoria seguita dalla telecamera è vagamente sinusoidale.

La situazione peggiora ancora con l'applicazione di un profilo di moto triangolare, come mostrato in figura 7.50. In questo caso il carrello è sempre all'inseguimento dell'oggetto nel tratto rettilineo, poiché l'errore di inseguimento non è nullo. I cambi di traiettoria sono evidenziati dal cambio di segno dell'errore. Come nel caso del moto sinusoidale, il carrello segue lo stesso andamento del moto dell'oggetto, semplicemente in ritardo.

(a) *Profilo di velocità.*(b) *Errore di Posizionamento.*(c) *Comparazione.***Figura 7.49:** Guida lineare: inseguimento, movimento sinusoidale

(a) *Profilo di velocità.*(b) *Errore di Posizionamento.*(c) *Comparazione.***Figura 7.50:** Guida lineare: inseguimento, movimento triangolare

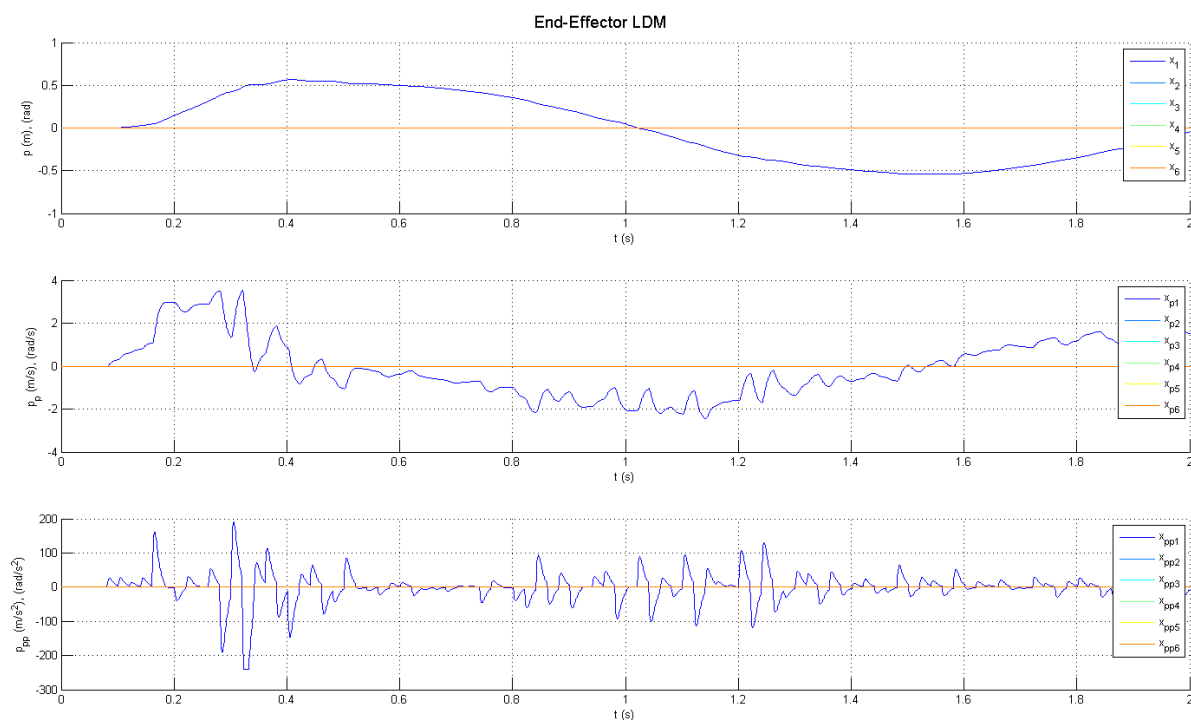
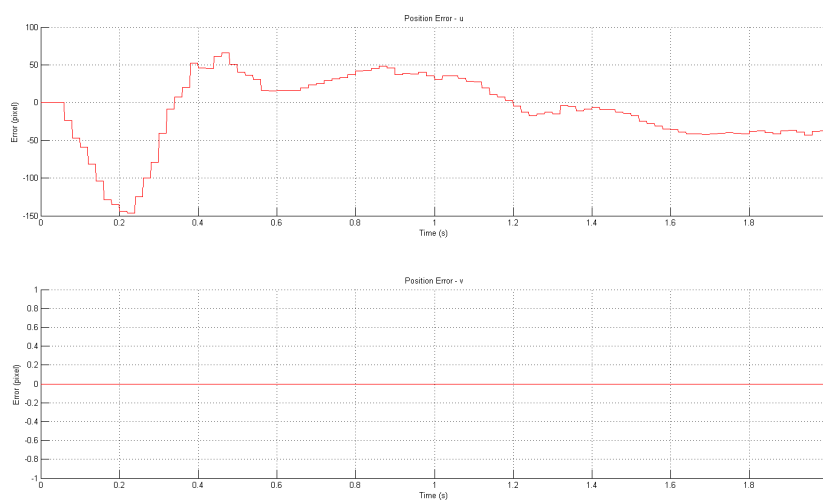
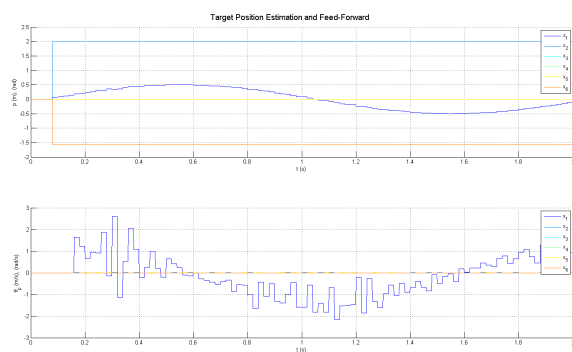
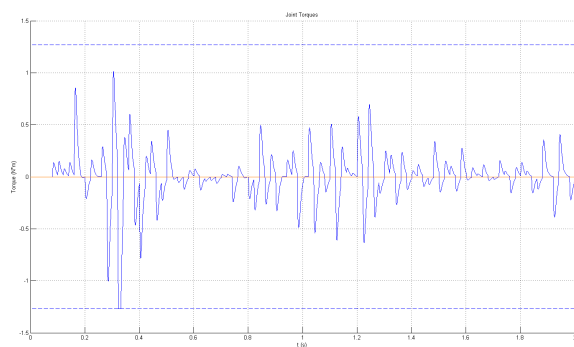
### 5.3 IBVS: inseguimento con feedforward

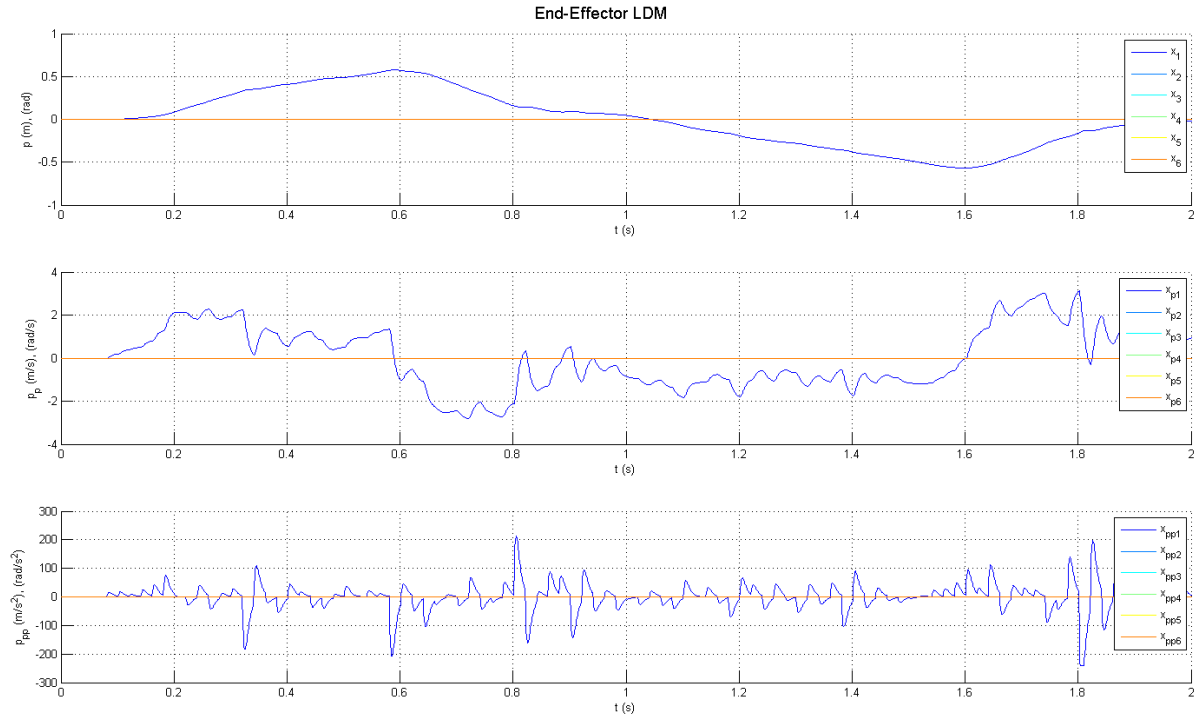
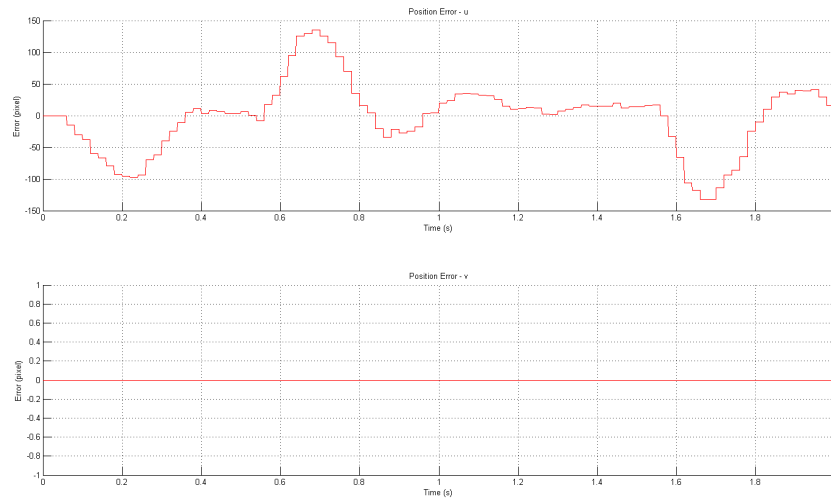
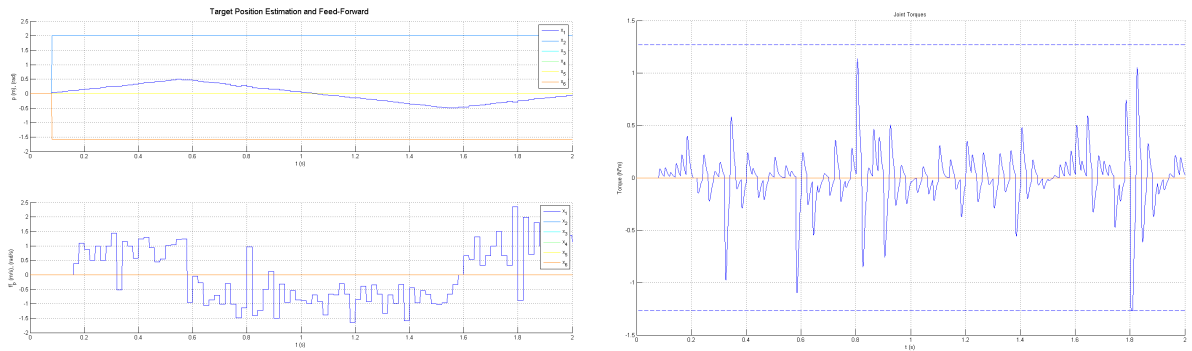
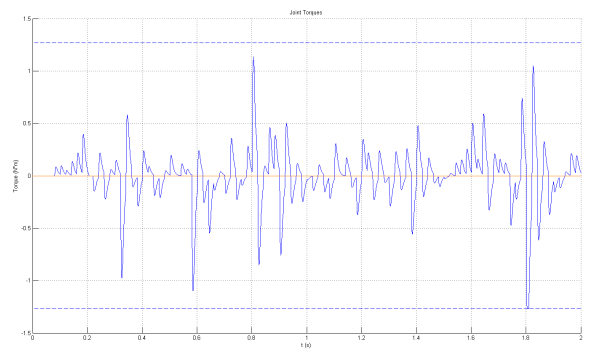
Come si è visto nel corso della trattazione della tesi, per rendere più performante l'inseguimento del moto della scena è necessario implementare una componente di controllo in feedforward.

In figura 7.51 viene mostrata la risposta del sistema alla stessa perturbazione sinusoidale di cui si è discusso in precedenza. La stima della velocità permette di migliorare l'inseguimento del moto. In particolare, si riescono a distinguere due fasi dell'inseguimento. Nei primi 400 *ms* il carrello subisce accelerazioni importanti perché si deve allineare all'oggetto. Durante la seconda fase, invece, interviene in modo prevalente la stima del moto dell'oggetto osservato, la quale stima permette di minimizzare l'errore di posizionamento, che viene ridotto del 75 % rispetto al controllo senza feedforward e rimane pertanto nell'intorno dello zero. I cambiamenti di segno sono gli effetti dei cambi di posizione, ma la loro influenza sull'errore di inseguimento è decisamente ridotta. L'andamento del segnale di feedforward è lo stesso del movimento applicato all'oggetto.

Infine, il controllo in feedforward è stato messo alla prova col profilo di moto triangolare, come mostrato in figura 7.52. Come si è già messo in evidenza trattando l'inseguimento del profilo di moto triangolare nel controllo senza feedforward, questo tipo di moto è più difficile da inseguire rispetto al profilo sinusoidale. Tuttavia, anche in questo caso l'inserimento del ramo di feedforward nello schema di controllo porta notevoli benefici all'inseguimento. In questo caso, si possono distinguere tre fasi: (i) l'inizializzazione, (ii) l'inseguimento nel tratto rettilineo del moto in cui l'errore di posizionamento viene annullato e (iii) la correzione del posizionamento dopo ogni cambio di direzione dell'oggetto. Quest'ultima fase è una sorta di nuova inizializzazione, provocata dal fatto che il cambio di direzione fa “perdere” il tracciamento al predittore del moto dell'oggetto.

Infine, analizzando il profilo di coppia richiesta al motore, si può notare come il feedforward porti il motore ad essere più nervoso. È per questo motivo che l'implementazione di un buon controllo per l'inseguimento, con annessa scelta del motore, è più complesso rispetto al caso del posizionamento.

(a) *Profilo di velocità.*(b) *Errore di Posizionamento.*(c) *Feedforward.*(d) *Coppia.***Figura 7.51:** Guida lineare: tracking, movimento sinusoidale, feedforward

(a) *Profilo di velocità.*(b) *Errore di Posizionamento.*(c) *Feedforward.*(d) *Coppia.***Figura 7.52:** Guida lineare: inseguimento con feedforward, movimento triangolare

## 6 Conclusioni

Questo capitolo ha riguardato la presentazione delle problematiche da affrontare per il progetto e la valutazione di controllori per un sistema di controllo in asservimento visivo. La trattazione ha riguardato, in primo luogo, lo studio dell'analisi dei controllori implementati da un azionamento di un attuatore elettrico. In particolare, la scelta è ricaduta sulla presentazione del controllo di un motore in corrente continua, data la semplicità con cui tale sistema può essere modellato. Questa assunzione non è limitante, poiché essa ha comunque permesso di presentare i fondamenti del controllo in cascata che permettono la differenziazione e integrazione dei controlli in coppia, velocità e posizione. Questo modello, inoltre, sotto particolari assunzioni può essere applicato anche allo studio del comportamento dinamico di un motore brushless.

Se lo studio della dinamica di un azionamento elettrico è un argomento ben noto e ampiamente trattato in letteratura, non si può dire la stessa cosa per quanto concerne l'analisi della dinamica di un sistema di elaborazione immagini. In questo caso, l'analisi dinamica riguarda la modellazione di tutti i possibili ritardi introdotti dall'elaborazione immagine. La componente introdotta da questi ritardi porta al principale limite di banda imposto dal controllo *vision in the loop*. In letteratura gli aspetti dinamici sono spesso trattati con tecniche di controllo *multirate*, afferenti al problema dell'integrazione di anelli di controllo operanti a frequenze molto diverse tra loro. Tuttavia, anche se queste tecniche permettono di affrontare il problema in tutta la sua complessità, esse rendono difficile l'individuazione di una relazione stretta tra la legge di controllo sviluppata e il risultato atteso. Al contrario, trattare il problema abbassando la frequenza operativa dal sistema alla minima frequenza in gioco porta ad una semplificazione eccessiva della dinamica del sistema. La soluzione intrapresa prevede la modellazione a tempo continuo delle componenti a tempo discreto del controllo. Tale soluzione rappresenta una via di mezzo tra i due approcci noti in letteratura poiché da un lato permette l'utilizzo delle classiche tecniche di controllo automatico, non trascurando nello stesso tempo le non-linearità introdotte.

Un'ulteriore difficoltà nella trattazione del controllo in asservimento visivo è data dalla mancanza di una funzione di sensitività ben definita, ovvero non c'è una relazione diretta tra il segnale di riferimento e il segnale generato in uscita. Infatti, sia il riferimento che l'errore sono calcolati in termine di posa e differenza di posa tra l'immagine desiderata e l'immagine corrente. È stato pertanto necessario definire le funzioni di trasferimento utilizzabili per l'analisi delle prestazioni del sistema controllato. Inoltre, spesso i sistemi

di controllo in asservimento visivo hanno a che fare con oggetti in movimento, pertanto l'introduzione di una componente di feedforward è la soluzione più utilizzata per permettere la corretta esecuzione del task di inseguimento.

Infine, tutte le considerazioni hanno permesso lo sviluppo di uno strumento di simulazione specifico per i task di controllo in visual servoing. Tale strumento, basandosi su considerazioni di tipo generale, permette l'analisi di differenti tipi di sistemi, dai più semplici meccanismi ad un grado di libertà fino a robot più complessi. La simulazione ha riguardato l'analisi delle prestazioni del controllo di un semplice sistema ad un grado di libertà, poiché in questo modo è possibile separare le componenti di controllo dovute all'asservimento visivo dalle tematiche del controllo di sistemi meccanici più complessi.



# Capitolo 8

## Criteri di Sviluppo di Componenti Software per l'Asservimento Visivo

### Indice

---

1	Acquisizione immagini . . . . .	<b>351</b>
1.1	IPC: Inter-Process-Communication . . . . .	354
1.2	Protocollo Firewire: libdc1394 . . . . .	361
2	Elaborazione immagini . . . . .	<b>367</b>
2.1	Costruzione del descrittore . . . . .	368
2.2	Individuazione e tracciamento dell'oggetto . . . . .	377
2.3	Libreria per visione artificiale: openCv . . . . .	380
3	Controllore RTAI . . . . .	<b>381</b>
3.1	Linux RTAI . . . . .	382
3.2	IPC: FIFO . . . . .	389
3.3	ComediLib: comunicazione con le periferiche . . . . .	397
4	Conclusioni . . . . .	<b>403</b>

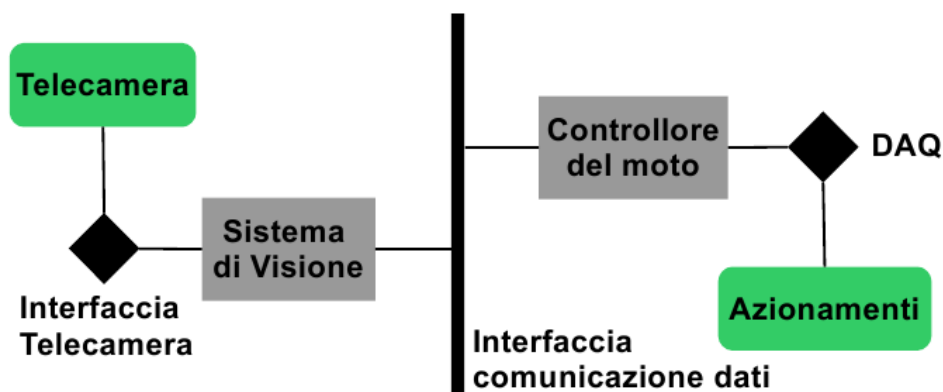
---

Nel corso della trattazione sono stati presentati i diversi aspetti riguardanti il controllo in asservimento visivo. In particolare, è stato seguito uno schema logico volto a suddividere il sistema completo in due sottosistemi distinti: il *sistema di visione* e il *supervisore del moto* del manipolatore. Come si è visto nel capitolo 5, lo sviluppo di controlli in puro asservimento visivo prevede una profonda integrazione tra le due componenti. Per favorire lo sviluppo di un'integrazione funzionale, è necessario analizzare le problematiche relative alla progettazione e implementazione software delle componenti fondamentali del sistema.

Una prima scomposizione delle tematiche da affrontare può essere ottenuta considerando la sequenza di operazioni da compiere per arrivare all'esecuzione del controllo in asservimento visivo:

1. *Acquisizione delle immagini;*
2. *Elaborazione delle immagini;*
3. *Calcolo della legge di moto;*
4. *Applicazione dei comandi agli azionamenti degli attuatori.*

Questa scomposizione del problema in una serie di sottoelementi permette di mettere in luce la natura *distribuita* del sistema che si intende implementare. La separazione tra i compiti del sistema di visione e del controllore del robot è data direttamente dall'hardware con cui essi hanno a che fare: il sistema di visione si occupa di gestire i dispositivi di acquisizione immagine, mentre il controllore del manipolatore gestisce gli azionamenti degli attuatori. La scomposizione delle componenti del controllo in funzione dell'hardware è schematizzata in figura 8.1: il sistema di visione comunica con la telecamera per mezzo della sua interfaccia di comunicazione; il controllore del robot comunica con gli azionamenti per mezzo dell'elettronica di acquisizione e generazione dei segnali digitali e analogici (DAQ); i due componenti si interfacciano per mezzo di un canale di comunicazione dedicato.



**Figura 8.1:** Definizione dei task in funzione dell'hardware

Le due fasi intermedie, invece, si trovano a cavallo tra i due componenti. Nell'ottica del controllo in anello aperto (vedi 1) l'interconnessione tra i due componenti è molto debole, pertanto l'elaborazione delle immagini è tipicamente svolta dal sistema di visione e la legge di moto è calcolata dal controllore del manipolatore a partire dalle informazioni estratte dalle immagini. Al contrario, nell'asservimento visivo le immagini sono poste al centro del calcolo della legge di moto del manipolatore; pertanto, è necessario introdurre un terzo



- *Architettura distribuita.* Prevede la corrispondenza uno-a-uno tra componente logica e componente fisica. Questa configurazione porta ad una completa indipendenza tra le componenti, nel senso che diventa possibile scegliere hardware dedicato per ogni singola funzione e la sincronizzazione dello scambio dati è implementata direttamente via hardware dal canale di comunicazione scelto per l'interconnessione dei dispositivi.
- *Architettura semi-distribuita.* In questa configurazione si concentrano due task in un solo dispositivo. La scelta tipica prevede lo spostamento del task di elaborazione delle immagini e generazione della legge di controllo in serie all'acquisizione immagini. Pertanto, l'architettura software del sistema di visione deve essere complicata leggermente attraverso l'introduzione di elementi di sincronizzazione tra il dato acquisito e quello da elaborare.
- *Architettura concentrata.* In questo caso, tutti i task logici vengono raccolti ed eseguiti da un unico dispositivo fisico. Questo rende necessario l'utilizzo di hardware generico e non dedicato ad un singolo compito, poiché esso deve garantire sia le potenzialità di elaborazione e gestione di un gran numero di dati (l'immagine), sia le funzionalità di un vero e proprio sistema di controllo real-time per l'acquisizione e generazione di segnali analogici e/o digitali.

La terza configurazione è quella presa in considerazione per lo sviluppo di questo lavoro di tesi, poiché è, a mio avviso, la configurazione più difficile da implementare e quella che ha permesso di mettere a dura prova i meccanismi di gestione e scambio dati analizzati.

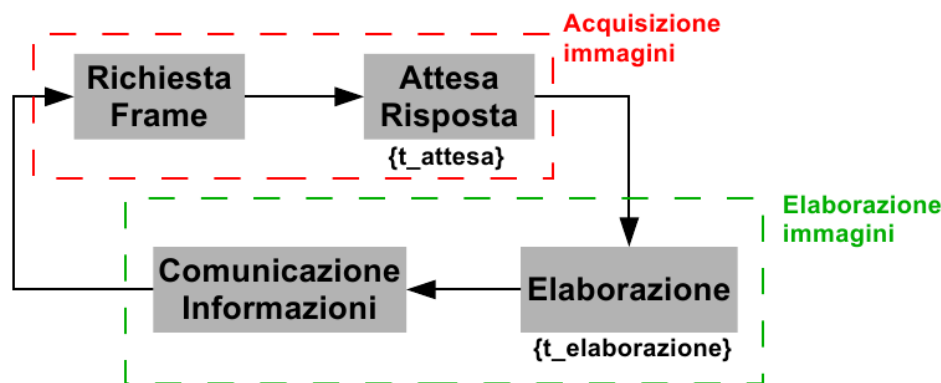
La complessità d'implementazione delle strutture e dei processi del task di asservimento visivo è testimoniato dalla presenza di architettura software dedicate a questo specifico compito, come ad esempio la piattaforma *VISP (Visual Servoing Platform)* ([site16],[44]). L'approccio seguito in questo capitolo prevede l'analisi di alcune tematiche relative allo sviluppo del completo controllo in asservimento visivo, al fine di porre le basi per la realizzazione di un'architettura per la programmazione di un generico task di posizionamento, come in [34]. Sempre mantenendo un punto di vista generale, alcune di queste tematiche si prestano bene all'utilizzo di formalismi astratti per la loro modellazione ([40], [41]).

In questo capitolo, la progettazione delle componenti software necessarie per l'implementazione del sistema visivo viene scomposta nelle tre componenti logiche principali: l'acquisizione delle immagini (1), la successiva elaborazione (2) e, infine, l'utilizzo delle informazioni per l'applicazione del controllo (3). In particolare, per ogni tema si presentano i meccanismi di base per la gestione e lo scambio dei dati indipendentemente dal

linguaggio di programmazione utilizzato per la loro implementazione; in seguito si entra nel dettaglio dell'implementazione in *C++* di tali elementi, presentando nel contempo anche le librerie software utilizzate per favorirne l'implementazione. Al termine, si presentano le considerazioni finali sul lavoro presentato (4).

## 1 Acquisizione immagini

Nella progettazione ed implementazione di un sistema di controllo in asservimento visivo, il problema dell'acquisizione immagini è spesso sottovalutato. Tipicamente, infatti, la telecamera è vista come un dispositivo in grado di vivere di vita propria e che debba unicamente rispondere alle richieste di frame da parte del sistema di elaborazione immagini. Infatti, le operazioni eseguite dal software del sistema di visione, sono generalmente due: la *richiesta di un frame* al dispositivo di acquisizione e, una volta che quest'ultimo ha fornito la risposta desiderata, l'*elaborazione delle immagini*, scomponibile a sua volta in un numero finito di operazioni, ma con l'unico scopo di ottenere le informazioni desiderate. L'esecuzione di queste operazioni avviene quindi sequenzialmente, poiché l'output dell'una costituisce l'input dell'altra. La sequenza acquisizione-elaborazione è schematizzata in figura 8.3.



**Figura 8.3:** Sequenza di acquisizione ed elaborazione immagini

Ad ogni modo, come si è visto nel paragrafo 2 del capitolo 7, i ritardi di tempo coinvolti nella formazione e acquisizione dell'immagine, seppur trascurabili, hanno il loro peso sulla dinamica complessiva del controllo. L'esecuzione sequenziale delle operazioni di acquisizione ed elaborazione delle immagini porta a sommare i ritardi della prima al tempo necessario per l'ottenimento delle informazioni da parte della seconda. Se il tempo d'elaborazione dipende fondamentalmente dalla complessità computazionale degli algoritmi scelti per arrivare all'estrazione delle caratteristiche dalle immagini, il ritardo di trasferimento dell'immagine dal dispositivo al calcolatore è dato da diversi fattori, tra cui:

- Ritardo di arrivo del comando per la cattura del frame. Questo tempo dipende dal comportamento della telecamera all'arrivo della richiesta, da parte del supervisore dell'acquisizione, di un frame. Nel caso di telecamera *sincrone* questo ritardo è nullo, poiché la loro modalità di funzionamento prevede l'acquisizione continua delle immagini e la loro memorizzazione in una struttura dati interna; la richiesta di un frame porta alla trasmissione dell'ultima immagine acquisita. Al contrario, le telecamere *asincrone* non prevedono l'acquisizione e digitalizzazione continua delle immagini: il processo comincia solo su richiesta da parte del sistema di acquisizione. In questo caso, quindi, il tempo di arrivo della richiesta alla telecamera non può essere trascurato.
- Tempo di *formazione* dell'immagine. Tale ritardo rappresenta il tempo richiesto dall'elettronica del dispositivo d'acquisizione per la digitalizzazione dei segnali letti dal sensore d'acquisizione. Come è stato detto nel paragrafo 1 del capitolo 2, questo è influenzato dai parametri di acquisizione scelti, quali lo *shutter* e il *guadagno*. Questo tempo non è mai trascurabile e porta sempre alla generazione di uno sfasamento temporale tra il vero istante di generazione dell'informazione e quello in cui tale informazione è resa disponibile. Questo ritardo, non essendo eliminabile, viene in generale gestito da alcune telecamere, le quali, insieme alle immagini, sono in grado di memorizzare e comunicare anche l'informazione relativa all'istante temporale (*timestamp*) d'acquisizione dell'immagine.
- Tempo di *trasmissione* del dato sul canale di comunicazione fisico. Questo dipende dalla banda passante dell'interfaccia di comunicazione scelta e può essere semplicemente sommato al ritardo di formazione delle immagini.

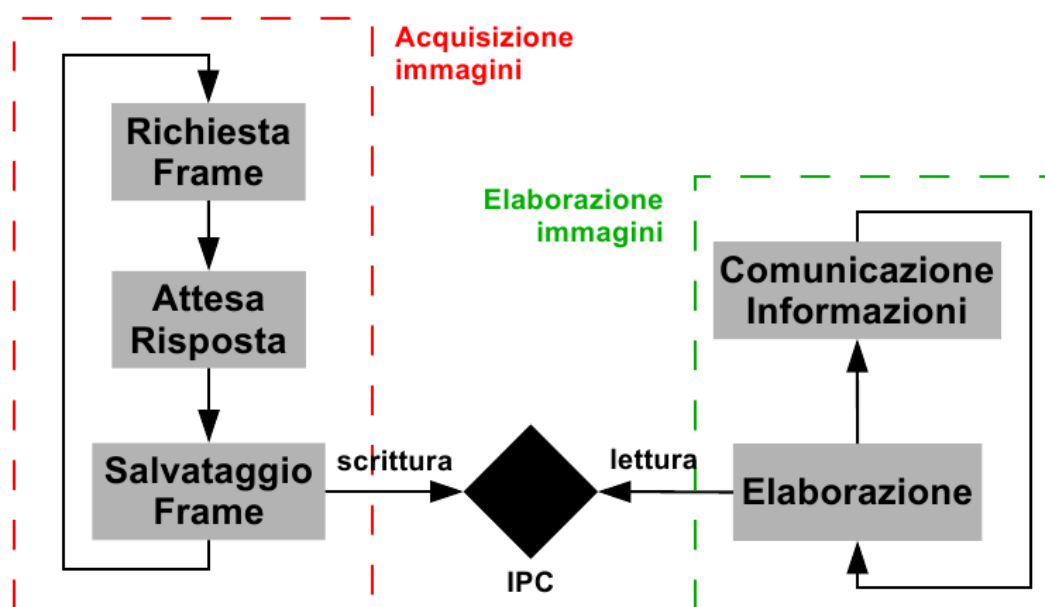
Come si evince dall'analisi di questi punti, le telecamere *sincrone* presentano notevoli vantaggi rispetto alle telecamere *asincrone*, poiché permettono di (i) annullare il ritardo dovuto alla ricezione della richiesta dell'immagine e di (ii) stimare l'istante reale di acquisizione dell'immagine stessa associandole la sua *timestamp*.

L'utilizzo delle telecamere *asincrone* con la semplice sequenza di acquisizione ed elaborazione, invece, tende a complicare lo sviluppo del controllo in asservimento visivo perché il ritardo di ricezione della richiesta dell'immagine porta all'introduzione, nell'anello di controllo, di un ritardo variabile. La soluzione a questo problema consiste nell'introduzione di un processo di *sincronizzazione* dell'acquisizione. Tale processo sfrutta il fatto che, tipicamente, l'elaborazione dell'immagine è computazionalmente più onerosa della semplice

acquisizione. Pertanto, il ritardo dovuto alla ricezione delle richieste da parte del dispositivo d'acquisizione viene eliminato *parallelizzando* l'invio delle richieste. In questo modo l'informazione risulta essere immediatamente a disposizione della legge di controllo, per l'implementazione della quale è sufficiente analizzare il ritardo di formazione e successiva elaborazione dell'immagine. Il flusso di operazioni richieste dal processo di *acquisizione sincrona* delle immagini risulta quindi essere composto da:

1. Invio della richiesta di un frame;
2. Attesa del frame;
3. Memorizzazione del frame in un'apposita struttura dati.

In figura 8.4 è mostrata la scomposizione dell'acquisizione e dell'elaborazione in due processi paralleli.



**Figura 8.4:** *Acquisizione ed elaborazione immagini in parallelo*

Questo modo di operare, inoltre, tende a limitare, se non ad annullare, l'effetto di ritardi eccessivi nella risposta da parte del dispositivo di acquisizione. Tali ritardi, infatti, vengono mascherati dalla gestione dell'acquisizione: il processo di elaborazione immagini percepisce solamente il fatto di trovarsi ad elaborare un'immagine molto vecchia; questi può decidere se continuare con l'elaborazione o attendere un'immagine più nuova, mentre, nel caso dell'acquisizione sequenziale, l'unica scelta possibile è attendere.

La difficoltà principale nell'implementazione di questo meccanismo sincrono di scambio dati (*IPC, Inter-Process-Communication*) consiste nell'implementazione della struttura dati per

la memorizzazione delle immagini, la quale deve garantire un accesso *mutuo-esclusivo* alle immagini: il processo di elaborazione non deve leggere le informazioni mentre il processo di acquisizione si sta occupando del loro aggiornamento. Una perfetta analisi e descrizione delle componenti per lo scambio dati e gestione di processi real-time è fornita da [9].

## 1.1 IPC: Inter-Process-Communication

Il task rappresentato in figura 8.4 appartiene alla classe di problemi modellati col pattern *Produttore-Consumatore*. Infatti, uno dei due task (l'acquisizione immagini) si occupa di produrre i dati, facendo accesso alla memoria solo in scrittura, mentre l'altro (l'elaborazione immagini) utilizza la struttura dati solo in modalità di lettura.

In generale, il progetto di una struttura per la comunicazione di dati tra processi (*Inter Process Communication*) prevede l'individuazione di un oggetto (composto da dati e metodi), denominato *mediatore* che si deve occupare di far passare i dati da un oggetto A ad un altro oggetto B. Nel caso del problema del Produttore-Consumatore il mediatore permette di instaurare la comunicazione (i) senza che l'oggetto A conosca alcunché dell'oggetto B e (ii) senza preoccuparsi dei dettagli implementativi che permettono il passaggio del messaggio da A e B. Pertanto, il mediatore è tipicamente costituito da un'area di memoria ad *accesso protetto*, ovvero con accesso possibile esclusivamente tramite la chiamata a metodi di lettura ("*read*") e scrittura ("*write*") messi a disposizione dal mediatore e protetti da *meccanismi di mutua esclusione*, solitamente semafori e mutex.

Tuttavia, anche tenuto conto di queste caratteristiche, nel progetto del meccanismo di IPC bisogna tener conto della frequenza di accesso dei task Produttore e Consumatore. In particolare, si possono prendere in considerazione le seguenti tipologie di sincronizzazione:

- *Memoria condivisa*. Nel caso in cui produttore e consumatore cerchino di accedere ai dati con la stessa frequenza, è sufficiente condividere un'area di memoria. La sincronizzazione è data da un semplice semaforo: dato che la frequenza di accesso è la stessa, il consumatore rimane bloccato in attesa di lettura mentre il consumatore sta aggiornando i dati. In seguito, avviene il contrario, col produttore bloccato in scrittura in attesa che il consumatore termini di leggere i dati.
- *Buffer circolare*. Rappresenta un'estensione del concetto di memoria condivisa in cui non viene salvata solo l'ultima informazione, ma un vettore contenente un certo numero di informazioni.



- *Swinging buffer*. È l'estensione del concetto di buffer circolare in cui vengono utilizzate due locazioni di memoria: una per l'accesso in scrittura e l'altra per la lettura. Questa struttura dati permette di risolvere i problemi derivanti dalla differenza di frequenza operativa tra il processo Produttore e il Consumatore.

### 1.1.1 Memoria Condivisa

Utilizzando un'area di *memoria condivisa* due o più task riescono a comunicare scrivendo e leggendo dati sulle stesse celle di memoria. Il principale vantaggio di questo meccanismo è che lo scambio dati avviene senza bisogno di effettuare copie dei dati. Ad esempio, un ambito in cui si sfrutta la memoria condivisa è nella comunicazione con le periferiche. Infatti, un caso particolare di memoria condivisa è il *DMA* (*Direct Memory Access*), in cui il dato viene letto direttamente dalla memoria della periferica.

Lo svantaggio della memoria condivisa consiste nel limite al numero di canali IPC inizializzabili, dato dal limite di memoria RAM disponibile. La memoria condivisa viene allocata e riservata direttamente dal sistema operativo, che però non mette a disposizione dei meccanismi per conoscere il tempo intercorso tra la scrittura e la lettura. Questo viene quindi stimato ricorrendo all'associazione di un'informazione temporale (timestamp) all'informazione desiderata.

La memoria condivisa non permette la gestione della perdita di dati ed è poco efficiente dal punto di vista dell'ottimizzazione dei periodi di tempo in cui i processi hanno accesso alla memoria. Infatti, se il processo produttore fornisce dati ad una frequenza più elevata di quella sopportabile dal consumatore, la memoria finisce per essere sovrascritta più volte, mentre viceversa il consumatore leggerebbe lo stesso dato per più volte. Inoltre, l'accesso alla memoria deve essere mantenuto per tutto il tempo necessario alla scrittura o lettura dei dati. Per tali motivi, la memoria condivisa non è adatta all'utilizzo come mediatore per l'acquisizione delle immagini.

Il codice 8.1 mostra una possibile implementazione C++ di un oggetto per la gestione dell'accesso mutuo-esclusivo a particolari sezioni del programma.

---

**Codice 8.1:** Definizione della classe per la gestione della mutua-esclusione

---

```
1  class Lock_MultipleReadersSingleWriter {
2  public:
3      bool reader_lock();
4      bool reader_unlock();
5      bool writer_lock();
```

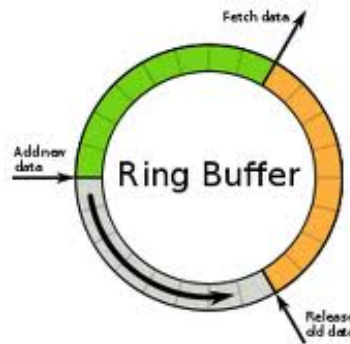
```
6    bool writer_unlock();
7 };
8
9 class LockedData_MultipleReadersSingleWriter{
10 private:
11     Lock_MultipleReadersSingleWriter lock;
12
13 public:
14     LockedData_MultipleReadersSingleWriter();
15
16     bool lock_reader(){
17         return lock.reader_lock();
18     }
19
20     bool unlock_reader();{
21         return lock.reader_unlock();
22     }
23
24     bool lock_writer(){
25         return lock.writer_lock();
26     }
27
28     bool unlock_writer(){
29         return lock.writer_unlock();
30     }
31 };
```

---

### 1.1.2 Buffer Circolare

Un *buffer circolare* ha le stesse caratteristiche della memoria condivisa, eccetto il fatto che ha dimensione maggiore di uno. Il buffer viene implementato come un vettore di singole strutture dati per la comunicazione con in più l'indicazione della posizione della cella scritta e letta. Quando uno di questi indici raggiunge la fine del vettore, viene riazzerato e l'operazione associata ricomincia dall'inizio del vettore. In figura 8.5 la rappresentazione del buffer circolare.

Avviene perdita di dati se il produttore completa il giro del vettore più velocemente del consumatore. Viceversa, ovvero se il consumatore è più veloce del produttore, non si ha perdita di dati, ma l'ultimo dato scritto viene letto più volte. Per garantire accesso mutuo



**Figura 8.5:** Struttura di un buffer circolare

esclusivo si utilizzano dei mutex, definibili a livello delle singole celle o a livello dell'intero vettore. Esistono due principali tipi di buffer circolari:

- a *memoria fissa*. La memoria utilizzata per il buffer è indirizzata completamente all'interno dei limiti della RAM disponibile;
- *gestione della memoria con interrupt*. Il consumatore o il produttore emettono un segnale di interrupt se il buffer è riempito per più o meno della metà. Infatti, nel primo caso significa che il consumatore legge i dati troppo lentamente, mentre nel secondo caso è il produttore che carica un numero inferiore di dati rispetto a quello che il consumatore potrebbe gestire.

Relativamente al problema dell'acquisizione immagini, si ha a che fare con produttore e consumatore operanti a frequenze molto diverse. Il produttore si occupa esclusivamente dell'acquisizione delle immagini, mentre il consumatore è tipicamente seguito dall'elaborazione delle immagini. Assumendo che questa operazione sia molto più onerosa dell'acquisizione delle immagini, l'utilizzo del buffer circolare porterebbe ad avere un vettore sempre pieno, col vantaggio che il consumatore potrebbe avere a disposizione una storia delle immagini acquisite.

### 1.1.3 Swinging Buffer

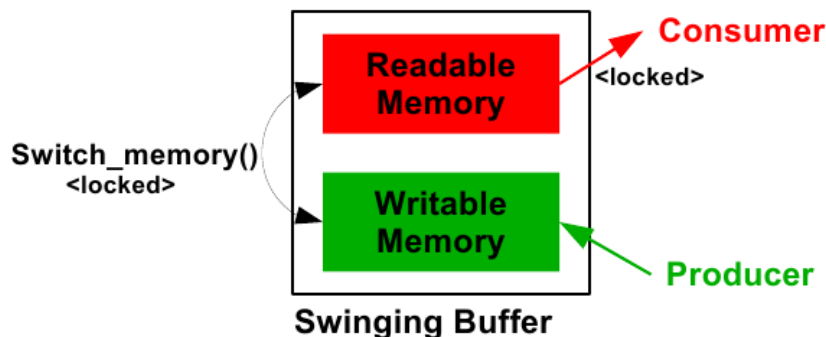
Lo *swinging buffer* è una struttura dati per la sincronizzazione dei task utilizzata tipicamente nei casi di *letture frequenti e scritture infrequenti*. Tale struttura può essere definita, in modi diversi, come:

- *buffer circolare avanzato*, che utilizza due vettori di memoria condivisa invece che un solo array. Il produttore riempie uno dei vettori, mentre il consumatore ne svuota

un'altra. Ogni volta che uno dei due processi raggiunge la fine del vettore su cui sta operando, ricomincia l'esecuzione sull'area di memoria non utilizzata dall'altro processo;

- un *meccanismo di sincronizzazione senza deadlock*. Entrambi i processi operano su strutture dati diverse, in modo tale che l'utilizzo di semafori per limitarne l'accesso non è necessario o è sfruttato in un modo del tutto particolare, come viene descritto in seguito.

Per tali motivi la struttura a swinging buffer è non bloccante, ma comunque non immune da perdita di dati, poiché il consumatore può leggere per più volte la stessa informazione. A causa del modo in cui lo swinging buffer funziona, questo approccio è noto in letteratura col termine *RCU (Read-Copy-Update)* poiché il consumatore tipicamente legge i dati, copiandone il valore, e successivamente preparare la struttura dati per la prossima lettura sull'altra area di memoria. In figura 8.6 sono evidenziate le caratteristiche salienti dello swinging buffer: produttore e consumatore lavorano su aree di memoria differenti, alternabili tramite la chiamata di un apposito metodo.



*Figura 8.6: Struttura di uno swinging circolare*

Nell'ambito dell'acquisizione delle immagini l'approccio dello swinging buffer può essere adattato invertendo consumatore e produttore: il dispositivo di visione genera dati ad una frequenza maggiore della frequenza con cui essi vengono elaborati. Questo porta ad una differenziazione del flusso di operazioni da eseguire per accedere alla memoria. Il processo più veloce agisce opera sullo swinging buffer esattamente come se si trattasse di una memoria condivisa, ovvero:

1. acquisisce il *lock* dello swinging buffer, in modo che la cella di memoria marcata come scrivibile rimanga scrivibile fino al termine della scrittura;

2. scrive i dati;
3. rilascia il lock;
4. aspetta la successiva acquisizione per aggiornare la memoria.

Al contrario, il consumatore può operare per tutto il tempo necessario sull'altra cella di memoria, seguendo questo flusso di operazioni:

1. lettura ed elaborazione del dato;
2. acquisizione del lock dello swinging buffer;
3. inversione della funzione della memoria: la cella in scrittura viene marcata come memoria da leggere, mentre quella in scrittura diventa leggibile;
4. rilascio del lock dello swinging buffer.

L'implementazione C++ delle operazioni sullo swinging buffer è fornita in 8.2.

---

**Codice 8.2:** Definizione della classe per la gestione di uno *Swinging Buffer*

---

```
1  template <class T>
2  class PSwingingBuffer : public
        LockedData_MultipleReadersSingleWriter
3  private:
4      T data[2];
5      int indexRead;
6      int indexWrite;
7
8  public:
9      PSwingingBuffer(T init) : LockedData_MultipleReadersSingleWriter()
        {
10         indexRead = -1;
11         indexWrite = 0;
12         data[0] = init;
13         data[1] = init;
14     }
15
16     T push (T object){
17         T res = data[indexWrite];
18         data[indexWrite] = object;
19         if (indexRead < 0){
20             update_indexes();
21         }
```

```
22     return res;
23 }
24
25 bool pull (T &pulled){
26     if (indexRead <0){
27         return false;
28     }
29     pulled = data[indexRead];
30     return true;
31 }
32
33 void update_indexes(){
34     indexRead = (indexRead + 1) % 2;
35     indexWrite = (indexWrite + 1) % 2;
36 }
37 };
```

---

Come si può notare dal flusso delle operazioni, lo swinging buffer permette di ridurre al minimo i tempi di latenza della comunicazione dei dati poiché il processo più lento limita l'acquisizione dell'accesso alla memoria solo per il tempo necessario alla fase di aggiornamento della funzione assegnata alle strutture dati. Il codice 8.3 riporta l'esempio di applicazione dello swinging buffer all'acquisizione e lettura delle immagini.

---

**Codice 8.3:** *Esempio d'utilizzo dello Swinging Buffer*

---

```
1  class SwingingBufferCapture{
2  private:
3      PSwingingBuffer<char*> memory;
4  public:
5      bool writeImage(){
6          char *image;
7          // Image acquisition into char *image ...
8
9          if (memory.lock_writer()){
10             memory.push(image);
11             memory.unlock_writer();
12         }
13     }
14
15     bool readImage(char* &result){
16         bool res = memory.pull(result);
```

```
17     if (memory.lock_writer()) {
18         memory.update_indexes();
19         memory.unlock_writer();
20     }
21     return res;
22 }
23 };
```

---

## 1.2 Protocollo Firewire: libdc1394

L'implementazione del protocollo firewire per la comunicazione con le telecamere è stata fornita dalla libreria *libdc1394* ([site4]). Questa fornisce un'interfaccia di programmazione (API) di alto livello per il controllo di telecamere basate sullo standard IEEE1394. La libreria è attualmente disponibile per i sistemi operativi più usati (Linux, Mac OSX e Windows) ed è completamente open source.

Tra le principali funzionalità della API si possono elencare:

- individuazione del dispositivo;
- comandi di broadcast;
- controllo delle risorse del bus;
- controllo delle proprietà della telecamera;
- gestione dei canali di memoria della telecamera;
- supporto per segnali di trigger interni ed esterni;
- supporto di diverse modalità video.

Il lavoro svolto per questa tesi ha portato alla progettazione e successiva implementazione di alcuni oggetti per la gestione e controllo dei dispositivi d'acquisizione immagine. I due principali oggetti implementati sono (i) un'*interfaccia* per la gestione dei comandi di basso livello da inviare alla telecamera (principalmente l'inizializzazione e l'acquisizione) e un (ii) oggetto per la *gestione del processo d'acquisizione* come descritto sopra. La definizione degli oggetti in C++ è mostrata rispettivamente in 8.4 e 8.5.

---

### *Codice 8.4: Interfaccia di comunicazione con la telecamera*

---

```
1  class Camera_Application_Interface
2  {
3  public:
4      Camera_Application_Interface();
```

---

```

5   virtual bool acquireImage (IplImage* &acq, bool undistort);
6   virtual bool getCameraIsRunning();
7   virtual void initCameraAndParameters();
8   virtual void destroyCameraAndParameters();
9 };

```

---

*Codice 8.5: Processo per la gestione della telecamera*

---

```

1  class Camera_Application_Thread
2  {
3  public:
4      Camera_Application_Thread(Camera_Application_Interface *interface)
5      {
6          this->interface = interface;
7      }
8
9  protected:
10     virtual void *run();
11
12 private:
13     Camera_Application_Interface *interface;
14 };

```

---

In figura 8.7 è schematizzata la suddivisione dei compiti per la gestione della telecamera. Si può notare che la gestione del dispositivo fisico è demandata all'oggetto interfaccia e al thread dedicato al controllo dell'acquisizione. Per accedere dall'esterno alle immagini acquisite è sufficiente chiamare il metodo di lettura dell'ultima immagine caricata sullo swinging buffer.

Per quanto riguarda il controllo delle telecamere firewire, tutti i metodi utili sono stati inglobati in un unico file, mostrato in 8.6.

---

*Codice 8.6: Metodi della libdc1394*

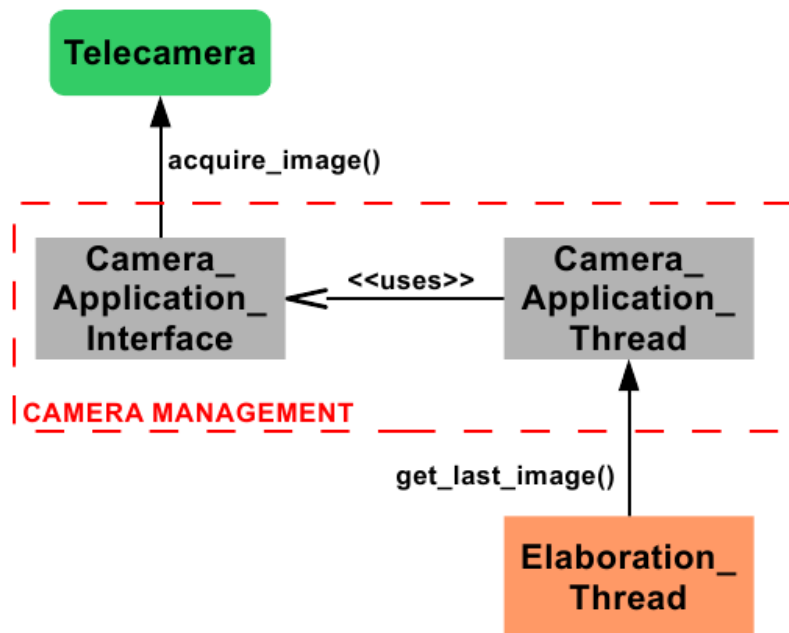
---

```

1  struct rawImageData
2  {
3      dc1394video_frame_t* frame;    //captured frame
4      char* imageData;
5      int width, height, depth;
6      bool released;
7  };
8

```





*Figura 8.7: Schema degli oggetti software per la gestione dell'acquisizione*

```

9  bool dc1394_control_initCamera(dc1394camera_t*& camera, dc1394_t *d
10      , int cameraId, dc1394video_mode_t video_mode ,
11      int dmaNumber, dc1394speed_t speed,
12      dc1394framerate_t &framerate_res);
13
14  bool dc1394_control_stopCamera(dc1394camera_t* &camera);
15
16  // Read a frame from the dma
17  bool dc1394_control_getFrame(dc1394camera_t* camera, rawImageData*&
18      rawData, dc1394capture_policy_t policy);
19
20  // Free a frame into the dma
21  bool dc1394_control_releaseFrame(dc1394camera_t* camera,
22      rawImageData*& rawData);
23
24  // Set camera feature mode to AUTO or MANUAL
25  int dc1394_control_setFeatureMode(dc1394camera_t* camera,
26      dc1394feature_t feature, dc1394feature_mode_t mode);
27
28  // Set camera feature value
29  int dc1394_control_setFeatureValue(dc1394camera_t* camera,
30      dc1394feature_t feature, unsigned int value);

```

In 8.7 è riportato lo pseudocodice C++ che mostra l'implementazione degli oggetti telecamera per la comunicazione con telecamere firewire. Operando in questo modo, la telecamera è utilizzabile semplicemente avviando un processo di tipo *dc1394\_Application\_Thread* che si occupa dell'acquisizione in continuo delle immagini e della loro memorizzazione nello swinging buffer, della gestione degli errori, . . . . Il processo che deve utilizzare le immagini per l'elaborazione si interfaccia col processo di controllo tramite la chiamata del metodo `getLastImage()` che permette di estrarre l'ultima immagine memorizzata nello swinging buffer.

---

**Codice 8.7:** *Implementazione per le libdc1394*

---

```

1  class dc1394_Application_Interface : public
    Camera_Application_Interface
2  {
3  private:
4      dc1394camera_t* camera;
5
6  public:
7      dc1394_Application_Interface() : Camera_Application_Interface(){
8          // ...
9      }
10
11     void initCameraAndParameters(){
12         // Initialization of dc1394camera_t *camera struct
13     }
14
15     bool acquireImage (IplImage* &acq, bool undistort){
16         // Initialize image to be acquired
17         // ...
18         // Acquire Image
19         rawData rawData; // filled by dc1394_control_getFrame
20         bool dc1394_control_getFrame(camera, &rawData,
            DC1394_CAPTURE_POLICY_WAIT);
21         // Convert rawData to IplImage*
22         memcpy(acq->data, rawData->imageData, sizeof(rawData->imageData)
            );
23         // Return
24         return true;
25     }
26 };

```

```
27
28 class dc1394_Application_Thread : public Camera_Application_Thread
29 {
30 private:
31     bool undistort;
32     PSwingingBuffer<IplImage*> memory;
33
34 public:
35     dc1394_Application_Thread(dc1394_Application_Interface *interface)
36     : Camera_Application_Thread(interface){
37     }
38
39     bool getLastImage(IplImage* &image){
40         bool res = memory.pull(image);
41         if (memory.lock_writer()){
42             memory.update_indexes();
43             memory.unlock_writer();
44         }
45         return res;
46     }
47
48 protected:
49     void* run(){
50         while (running){
51             IplImage *image = NULL;
52             interface->acquireImage(image, undistort);
53             if (memory.lock_writer()){
54                 memory.push(image);
55                 memory.unlock_writer();
56             }
57             usleep(2*1000);
58         }
59     }
60 };
```

---

Le informazioni relative alla parametrizzazione della telecamera, necessarie per la sua corretta inizializzazione, sono salvate in un apposito file XML, di cui un estratto è mostrato in 8.8. Le informazioni riguardano la modalità video (risoluzione e profondità colore), la velocità di trasmissione dati e tutti i parametri impostabili (shutter, guadagno,

...). Pertanto, nel metodo di inizializzazione della telecamera viene invocato un parser XML per la lettura delle informazioni da impostare successivamente.

---

*Codice 8.8: File xml dei parametri telecamera*

---

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2  <camera>
3    <camera-info colorConversion="49" dmaSize="3" framerateIndex="37"
      guid0="19260842" guid1="134235664" isoSpeed="2" modelId="0"
      name="SONY_524358 - XCD-V60CR_0" vendorId="524358"
      videoModeIndex="69"/>
4  <features>
5    <feature featureId="416" featureIsOn="1" featureMode="736"
      featureValue="1024"/>
6    <feature featureId="417" featureIsOn="1" featureMode="736"
      featureValue="512"/>
7    <feature featureId="418" featureIsOn="0" featureMode="736"
      featureValue="0"/>
8    <feature featureId="420" featureIsOn="1" featureMode="736"
      featureValue="2352"/>
9    <feature featureId="421" featureIsOn="1" featureMode="736"
      featureValue="64"/>
10   <feature featureId="422" featureIsOn="1" featureMode="736"
      featureValue="0"/>
11   <feature featureId="423" featureIsOn="1" featureMode="736"
      featureValue="300"/>
12   <feature featureId="424" featureIsOn="1" featureMode="736"
      featureValue="0"/>
13   <feature featureId="425" featureIsOn="0" featureMode="736"
      featureValue="0"/>
14   <feature featureId="426" featureIsOn="0" featureMode="736"
      featureValue="0"/>
15   <feature featureId="427" featureIsOn="0" featureMode="736"
      featureValue="0"/>
16   <feature featureId="428" featureIsOn="0" featureMode="736"
      featureValue="14904163"/>
17   <feature featureId="429" featureIsOn="0" featureMode="736"
      featureValue="0"/>
18   <feature featureId="430" featureIsOn="0" featureMode="736"
      featureValue="0"/>

```

```
19     <feature featureId="431" featureIsOn="0" featureMode="736"  
      featureValue="0"/>  
20     <feature featureId="432" featureIsOn="0" featureMode="736"  
      featureValue="0"/>  
21     <feature featureId="433" featureIsOn="1" featureMode="736"  
      featureValue="0"/>  
22     <feature featureId="434" featureIsOn="1" featureMode="736"  
      featureValue="6"/>  
23     <feature featureId="435" featureIsOn="1" featureMode="736"  
      featureValue="0"/>  
24     <feature featureId="436" featureIsOn="0" featureMode="736"  
      featureValue="0"/>  
25     <feature featureId="437" featureIsOn="0" featureMode="736"  
      featureValue="0"/>  
26 </features>  
27 </camera>
```

---

## 2 Elaborazione immagini

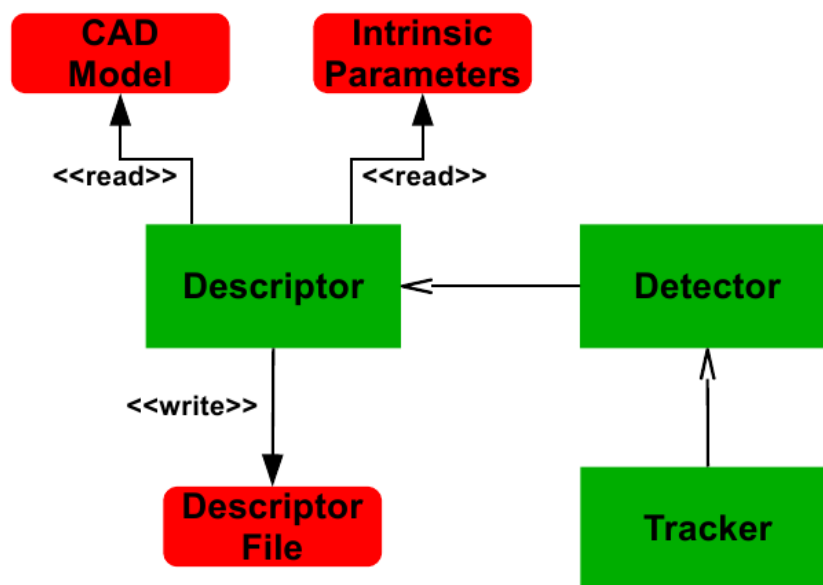
Nei capitoli dedicati alla descrizione degli algoritmi per la visione artificiale e, in particolar modo, per gli algoritmi di tracciamento, si è raramente fatto riferimento a considerazioni relative alla tipologia di hardware da utilizzare per l'esecuzione del software di visione. Questo fatto è dovuto al semplice motivo che tipicamente la scelta dell'hardware è strettamente collegata al tipo d'applicazione da sviluppare. Alcune caratteristiche dell'applicazione che possono fare da ago della bilancia sono ad esempio il numero di oggetti da seguire, la velocità di movimento della scena, la complessità dei modelli tridimensionali scelti per la descrizione degli oggetti, ...

Se quindi non è possibile a priori dare delle linee guida generali per la scelta e configurazione dell'hardware, la stessa cosa non si può generalmente dire per lo sviluppo e progettazione dell'architettura software. Nel corso di questa trattazione è stato evidenziato come sia necessario, ai fini della realizzazione di un controllo *vision in the loop*, implementare algoritmi robusti che permettano il tracciamento del movimento della scena. Pertanto, lo studio si è concentrato sulla progettazione di oggetti software che permettano di facilitare tutti i compiti di un sistema di tracciamento. In particolare, sono state analizzate e progettate le tre componenti fondamentali di un sistema di tracciamento:

- *Costruttore del descrittore geometrico (Descriptor);*

- Sistema di *individuazione dell'oggetto* (*Detector*);
- Modulo per il *tracciamento* del movimento dell'oggetto (*Tracker*).

Le tre componenti appena descritte sono schematizzate in figura 8.8, dove si nota come il descrittore è incaricato di acquisire da file le informazioni utili alla costruzione del descrittore delle feature dell'immagine. Successivamente, il descrittore si interfaccia direttamente col componente incaricato del matching col modello. Lo stesso componente è infine utilizzato dal *tracker* vero e proprio per il mantenimento del tracciamento della posa dell'oggetto nel tempo per tutta la sequenza di immagini analizzate.



**Figura 8.8:** Schema delle componenti del software di tracciamento implementato

## 2.1 Costruzione del descrittore

L'oggetto dedicato alla costruzione, salvataggio e gestione del descrittore geometrico dell'oggetto da individuare all'interno della scena è composto a sua volta da due componenti distinte:

- *ShapeModel*: descrittore del modello 3D (CAD) dell'oggetto;
- *ShapeDescriptor*: contenitore delle feature immagine dell'oggetto.

L'oggetto *ShapeModel* contiene essenzialmente un vettore di coordinate di punti e facce (mesh) la cui unione costituisce l'oggetto tridimensionale. Nel caso dell'algoritmo di tracciamento descritto nel paragrafo 3 del capitolo 6, tale oggetto viene esteso per

contenere anche l'informazione relativa alle facce e punti visibili in una certa vista. Le facce e i punti sono espressi sia in coordinate tridimensionali che in coordinate del piano immagine su cui sono proiettati. L'informazione essenziale per l'esecuzione dell'algoritmo è rappresentata dalla posa relativa telecamera-oggetto che ha generato il modello. Nel caso dell'algoritmo trattato, il matching dei modelli viene fatto a partire dalla semplice analisi dei momenti e, pertanto, il modello di una singola vista contiene anche l'informazione relativa ai momenti geometrici della forma individuata dalla proiezione dell'oggetto sul piano immagine. I campi della classe *ShapeModel* sono visualizzati nel listato 8.9.

**Codice 8.9:** Campi della classe *ShapeModel*

---

```

1  class ShapeModel {
2  public:
3      Camera *camera;
4      ReferencePoint *cosineDirector_i;
5      ReferencePoint *cosineDirector_j;
6      ReferencePoint *cosineDirector_k;
7      vector<FaceMesh*> faceVertexes;
8      vector<ReferencePoint*> referencePoints;
9      vector<FaceMesh*> faceVertexes_onCamera;
10     vector<ReferencePoint*> referencePoints_onCamera;
11     vector<float*> vertexes_onCamera_visible;
12     BlobDescriptor_Moments *momentsDescriptor;
13 };

```

---

Il modello base, per quanto riguarda la generazione del vettore di punti e facce dell'oggetto, viene ottenuto leggendo un file CAD. Per lo scopo di questo lavoro di tesi, il formato di file analizzato è il *COLLADA* (*COLLABorative Design Activity*), il quale memorizza le informazioni sulla geometria 3D di un oggetto in formato XML. Per lo scopo della costruzione del descrittore, infatti, il file COLLADA, a differenza di altri formati CAD quali STL, permette di memorizzare informazioni aggiuntive sugli oggetti che circondano l'oggetto 3D principale. Tra le caratteristiche memorizzabili si possono citare:

- Importazione di geometrie mesh;
- Gerarchie di trasformazione;
- Materiali e texture;
- Shader;
- Telecamere virtuali e luci;

- Skin e Morphing;
- Animazioni;
- Simulazioni fisiche;
- Istanze;

Al fine di muovere la telecamera attorno all'oggetto è sembrato opportuno, quindi, sfruttare la possibilità di aggiungere la posa iniziale della telecamera, rispetto a cui tutte le coordinate dei punti e delle facce vengono definite. Infine, un altro vantaggio risiede nel fatto che il formato COLLADA è oggi supportato da molti ambienti per il disegno 3D, come:

- 3D Studio Max;
- Autodesk Maya;
- Poser;
- LightWave 3D;
- Maxon Cinema 4D;
- Softimage XSI;
- SketchUp;
- Sweet home 3D;
- Blender;
- Bryce;
- Luxology MODO.

In 8.10 è riportato un estratto del file COLLADA utilizzato per costruire il modello dell'oggetto a "elle" utilizzato nelle simulazioni del paragrafo 3 del capitolo 6.

---

**Codice 8.10:** Estratto del file COLLADA per la descrizione di un oggetto a elle

---

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <COLLADA xmlns="http://www.collada.org/2005/11/COLLADASchema"
    version="1.4.1">
3    <asset>
4      <contributor>
5        <author>Blender User</author>
6        <authoring_tool>Blender 2.62.0 r44136</authoring_tool>
7      </contributor>
8    <created>2012-04-26T10:49:51</created>
9    <modified>2012-04-26T10:49:51</modified>
```



```

10     <unit name="meter" meter="1"/>
11     <up_axis>Z_UP</up_axis>
12 </asset>
13 <library_geometries>
14     <geometry id="Plane-mesh" name="Plane">
15         <mesh>
16             <source id="Plane-mesh-positions">
17                 <float_array id="Plane-mesh-positions-array" count="21"
18                     >1.263158 -0.8421053 0 -0.7368423 -0.8421052 0
19                     -0.7368418 1.157895 0 0.2631578 -0.8421053 0 0.263158
20                     1.157895 0 1.263158 -0.3421053 0 0.2631579 -0.3421053
21                     0</float_array>
22             <technique_common>
23                 <accessor source="#Plane-mesh-positions-array" count="7"
24                     stride="3">
25                     <param name="X" type="float"/>
26                     <param name="Y" type="float"/>
27                     <param name="Z" type="float"/>
28                 </accessor>
29             </technique_common>
30         </source>
31         <source id="Plane-mesh-normals">
32             <float_array id="Plane-mesh-normals-array" count="6">0 0 1
33                 0 0 1</float_array>
34             <technique_common>
35                 <accessor source="#Plane-mesh-normals-array" count="2"
36                     stride="3">
37                     <param name="X" type="float"/>
38                     <param name="Y" type="float"/>
39                     <param name="Z" type="float"/>
40                 </accessor>
41             </technique_common>
42         </source>
43         <vertices id="Plane-mesh-vertices">
44             <input semantic="POSITION" source="#Plane-mesh-positions"
45                 />
46         </vertices>
47         <polylist count="2">
48             <input semantic="VERTEX" source="#Plane-mesh-vertices"
49                 offset="0"/>

```

```

41         <input semantic="NORMAL" source="#Plane-mesh-normals"
           offset="1"/>
42     <vcount>4 4 </vcount>
43     <p>1 0 3 0 4 0 2 0 0 1 5 1 6 1 3 1</p>
44 </polylist>
45 </mesh>
46 <extra><technique profile="MAYA"><double_sided>1</double_sided
    ></technique></extra>
47 </geometry>
48 </library_geometries>
49 <library_visual_scenes>
50     <visual_scene id="Scene" name="Scene">
51         <node id="camera-01" type="NODE">
52             <translate sid="location">0 0 5</translate>
53             <rotate sid="rotationZ">0 0 1 0</rotate>
54             <rotate sid="rotationY">0 1 0 0</rotate>
55             <rotate sid="rotationX">1 0 0 180.0001</rotate>
56             <scale sid="scale">1 1 1</scale>
57         </node>
58         <node id="Plane" type="NODE">
59             <translate sid="location">0 0 0</translate>
60             <rotate sid="rotationZ">0 0 1 0</rotate>
61             <rotate sid="rotationY">0 1 0 0</rotate>
62             <rotate sid="rotationX">1 0 0 0</rotate>
63             <scale sid="scale">1 1 1</scale>
64             <instance_geometry url="#Plane-mesh"/>
65         </node>
66     </visual_scene>
67 </library_visual_scenes>
68 <scene>
69     <instance_visual_scene url="#Scene"/>
70 </scene>
71 </COLLADA>

```

Oltre al file CAD, sempre al fine di simulare il movimento della telecamera attorno all'oggetto e la sua successiva proiezione sul piano immagine, è necessario fornire al sistema un file contenente i *parametri intrinseci* della telecamera, necessari per sfruttare le ben note trasformazioni prospettiche da coordinate spaziali a coordinate immagine. Il file 8.11 rappresenta il formato di file XML utilizzato dalle *openCv* per il salvataggio dei parametri di calibrazione delle telecamere.

---

**Codice 8.11:** File XML per il salvataggio dei parametri di calibrazione di una telecamera

---

```

1  <?xml version="1.0"?>
2  <opencv_storage>
3  <Intrinsics type_id="opencv-matrix">
4    <rows>3</rows>
5    <cols>3</cols>
6    <dt>f</dt>
7    <data>
8      4.21618500e+002 0. 3.46250763e+002 0. 4.21618500e+002
9      2.01068481e+002 0. 0. 1.</data></Intrinsics>
10 </opencv_storage>

```

---

Nel caso dell'algoritmo di tracciamento a viste multiple, il descrittore è costruito muovendo la telecamera (fisicamente o virtualmente) attorno all'oggetto da descrivere. Pertanto il descrittore *ShapeDescriptor* complessivo contiene (i) il modello di base e (ii) un grafo di modelli costruiti a partire dai cambiamenti di vista.

Un estratto della classe *ShapeDescriptor* è mostrato in 8.12. Si può notare che ogni descrittore contiene il grafo dei modelli che lo costituiscono, mentre i metodi utilizzati per il matching delle osservazioni con i modelli sono implementati a livello dei descrittori specifici, nell'esempio il descrittore basato sui momenti geometrici.

---

**Codice 8.12:** Estratto della classe *ShapeDescriptor*

---

```

1  class ShapeDescriptor {
2  public:
3      ShapeModel *modelBase;
4      NDimensionalMatrix<ShapeModel> models;
5  };
6
7  class ShapeDescriptor_Moments : public ShapeDescriptor{
8  public:
9      /**
10     * @param descriptor Descrittore delle osservazioni
11     * @param camera Parametri telecamera utilizzata per le
12         osservazioni
13     * @param selectedModel modello confrontato con successo
14     * @param errorValue errore compiuto nel matching
15     */
16     void match (BlobDescriptor_Moments *descriptor, Camera *camera,
17         ShapeModel* &selectedModel, double &errorValue); };

```

La costruzione dei modelli del descrittore richiede la definizione di un set limitato di pose assunte dalla telecamera attorno all'oggetto. In generale, la definizione delle varie pose parte dalla posizione base della telecamera, scritta nel file COLLADA, per poi definire i limiti esterni del volume in cui si può muovere la telecamera. Dato che le variazioni di scala e rotazioni attorno all'asse ottico della telecamera vengono individuate indipendentemente dal matching con un modello, non è generalmente necessario definire degli spostamenti lungo questi assi. Le linee di codice seguenti mostrano l'inizializzazione dei limiti e i passi di campionamento del volume di lavoro della telecamera attraverso la definizione delle traslazioni lungo gli assi  $x$ ,  $y$  e le rotazioni attorno agli stessi assi.

---

```

1  double limits[4][2] = {{-3, 3} , {-3, 3} , {-M_PI/2, M_PI/2} ,
2    {-M_PI/2, M_PI/2}};
3  int steps[4] = {6, 6, 20 ,20};

```

---

A questo punto, alla telecamera vengono fatte assumere tutte le pose desiderate e, utilizzando le informazioni derivanti dal modello 3D e dai parametri intrinseci della telecamera, vengono generate tutte le immagini contenenti le proiezioni dell'oggetto. Infine, a partire dall'elaborazione delle immagini così ottenute, si estraggono tutte le feature immagine d'interesse, utilizzate successivamente nella fase di matching.

Il risultato del processo di costruzione del descrittore viene memorizzato in un file XML contenente (i) la descrizione della telecamera virtuale utilizzata per il calcolo dei modelli, (ii) la descrizione in termini di facce dell'oggetto completo e (iii) l'elenco dei modelli aggiunti al grafo di descrizione completo. Ad ogni modello del grafo è associata l'informazione immagine d'interesse; nell'esempio descritto tale informazione è costituita dalle caratteristiche del blob (centro geometrico, angolo principale, area, ...) e dai momenti invarianti di Hu. L'altra informazione utile invece al calcolo delle corrispondenze è relativa alle coordinate in pixel dei punti visibili nella proiezione dell'oggetto da un particolare punto di vista, anch'esso inserito nel file descrittore sia in termini dello spostamento della telecamera, a partire dalla sua posizione di base, che ha generato la proiezione, sia in termini di livello di profondità del grafo. Tutte queste informazioni sono rintracciabili nell'estratto del file XML 8.13 rappresentante il descrittore dell'oggetto a "elle" di cui si è già presentato il COLLADA.

---

**Codice 8.13:** File XML contenente il descrittore dell'oggetto a elle

---

```

1  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>

```

```

2  <blenderTest_flat.xml>
3    <common>
4      <camera>
5        <location x="0" y="0" z="5"/>
6        <orientation xRot="-3.14159" yRot="0" zRot="0"/>
7        <f fx="421.618" fy="421.618"/>
8        <resolution height="480" width="640"/>
9      </camera>
10     <face id="0" object_id="0">
11       <vertex x="-0.736842" y="-0.842105" z="0"/>
12       <vertex x="0.263158" y="-0.842105" z="0"/>
13       <vertex x="0.263158" y="1.15789" z="0"/>
14       <vertex x="-0.736842" y="1.15789" z="0"/>
15     </face>
16     <face id="1" object_id="0">
17       <vertex x="1.26316" y="-0.842105" z="0"/>
18       <vertex x="1.26316" y="-0.342105" z="0"/>
19       <vertex x="0.263158" y="-0.342105" z="0"/>
20       <vertex x="0.263158" y="-0.842105" z="0"/>
21     </face>
22     <refpoints/>
23     <levels>
24       <l lend="3" lid="0" lstart="-3" lstep="6"/>
25       <l lend="3" lid="1" lstart="-3" lstep="6"/>
26       <l lend="1.5708" lid="2" lstart="-1.5708" lstep="20"/>
27       <l lend="1.5708" lid="3" lstart="-1.5708" lstep="20"/>
28       <l lend="0" lid="4" lstart="0" lstep="1"/>
29     </levels>
30   </common>
31
32   <matching_graph area="202">
33     <model id="0">
34       <camera>
35         <location x="-3" y="-2" z="5"/>
36         <orientation xRot="-3.76991" yRot="-0.785398" zRot="0"/>
37         <f fx="421.618" fy="421.618"/>
38         <resolution height="480" width="640"/>
39       </camera>
40       <matching_tuple hu0="2.90889" hu1="8.42353" hu2="7.55228" hu3=
          "7.48945" hu4="56.3267" hu5="21.7334" hu6="-0.00516623"/>

```

```

41     <blob_parameters angle="-2.6214" angle360="2.07244" area="400"
        cog_x="70.1996" cog_y="65.005"/>
42     <visible_vertexes>
43         <vertex x="93" y="27"/>
44         <vertex x="101" y="8"/>
45         <vertex x="84" y="35"/>
46         <vertex x="19" y="153"/>
47     </visible_vertexes>
48     <levels l1="0" l2="1" l3="6" l4="5" l5="0"/>
49 </model>
50 <model id="1">
51     <camera>
52         <location x="-3" y="-2" z="5"/>
53         <orientation xRot="-3.76991" yRot="-0.628319" zRot="0"/>
54         <f fx="421.618" fy="421.618"/>
55         <resolution height="480" width="640"/>
56     </camera>
57     <matching_tuple hu0="0.655138" hu1="0.374422" hu2="0.0246681"
        hu3="0.0126772" hu4="0.000223792" hu5="0.00768919" hu6="
        1.32347e-005"/>
58     <blob_parameters angle="-2.45021" angle360="2.47925" area="
        2040" cog_x="67.0176" cog_y="74.9819"/>
59     <visible_vertexes>
60         <vertex x="108" y="10"/>
61         <vertex x="68" y="53"/>
62         <vertex x="8" y="148"/>
63         <vertex x="107" y="49"/>
64         <vertex x="121" y="23"/>
65         <vertex x="73" y="72"/>
66     </visible_vertexes>
67     <levels l1="0" l2="1" l3="6" l4="6" l5="0"/>
68 </model>
69 <model id="2">
70     <camera>
71         <location x="-3" y="-1" z="5"/>
72         <orientation xRot="-4.71239" yRot="-1.5708" zRot="0"/>
73         <f fx="421.618" fy="421.618"/>
74         <resolution height="480" width="640"/>
75     </camera>

```

```
76      <matching_tuple hu0="0.316195" hu1="0.0503146" hu2="0.0183719"
      hu3="0.00727515" hu4="8.26472e-005" hu5="0.00151173" hu6="
      -1.56095e-005"/>
77      <blob_parameters angle="2.81055" angle360="-1.20775" area="
      11435" cog_x="54.077" cog_y="146.57"/>
78      <visible_vertexes>
79          <vertex x="16" y="11"/>
80          <vertex x="16" y="213"/>
81          <vertex x="115" y="222"/>
82          <vertex x="115" y="154"/>
83          <vertex x="48" y="125"/>
84          <vertex x="48" y="35"/>
85      </visible_vertexes>
86      <levels l1="0" l2="2" l3="0" l4="0" l5="0"/>
87  </model>
88  <!-- ....>
89  </blenderTest_flat.xml>
```

---

## 2.2 Individuazione e tracciamento dell'oggetto

La fase di individuazione dell'oggetto può essere scomposta in diverse sottofasi:

1. Estrazione delle feature immagine utilizzate nella costruzione del descrittore. Nel caso dell'algoritmo in esame, le caratteristiche d'interesse sono i momenti geometrici della proiezione;
2. Confronto delle feature estratte con gli insiemi di feature memorizzate nei singoli modelli che compongono il descrittore, al fine di individuare il modello più vicino all'osservazione;
3. Affinamento dei modelli. Utilizzando l'approccio della retro-proiezione del modello sull'immagine viene calcolato il vettore delle corrispondenze, in modo da poter effettuare una stima della trasformazione omografica che proietta i punti del modello individuato precedentemente sull'osservazione in analisi;
4. La trasformazione omografica viene scomposta in modo da ottenere la trasformazione che permette di passare dalla posa della telecamera che ha generato il particolare modello trovato alla posa della telecamera che ha generato l'osservazione;

5. Nota la posizione della telecamera nel modello base, la posizione della telecamera nel modello trovato e la stima della trasformazione che permette di passare da quest'ultima posa alla posa dell'osservazione, risulta nota la stima della posizione della telecamera.
6. Dato che nel modello l'oggetto non viene mai spostato, la posa dell'oggetto è costante e nota a priori. Pertanto, la posa relativa tra telecamera ed oggetto è costruita a partire dalla posa assoluta della telecamera.

L'estratto dell'oggetto Detector 8.14 mostra le strutture dati ottenute tramite l'esecuzione dell'algoritmo di identificazione dell'oggetto all'interno della scena.

---

*Codice 8.14: Oggetto ShapeDetector*

---

```

1  class Match_struct{
2  public:
3      // Found blobs
4      CBlob *blob;
5
6      // Set of visible points
7      vector<float*> projectedPoints;
8
9      // Matched model
10     ShapeModel *model;
11
12     // Pose estimation and z correction
13     ReferenceSystem *correctionMatrix;
14     double dz;
15 };
16
17 class ShapeDetector {
18 private:
19     // Vector of blobs detected into the image
20     CBlobResult *detectedBlobs;
21     // Parameters of the camera used for observations
22     Camera *camera;
23     // Descriptor
24     ShapeDescriptor *descriptor;
25     // Threshold to discard wrong matches
26     double matchThreshold;
27

```



---

```

28 public:
29     // Result of detection
30     vector<Match_struct*> matches;
31
32     // Search a matching model into descriptor set
33     int matchModel(IplImage *&img, int levels[2][4]);
34
35     // Correct model transform with homography
36     void correctModel(BlobDescriptor_Moments *singleBlobDescriptor,
37         ShapeModel *model, Match_struct *currentMatch); };

```

---

Il tracciamento segue lo stesso flusso di operazioni della fase di individuazione. L'unica differenza è data dal fatto che l'oggetto per il tracciamento (*Tracker*) contiene informazioni sulla storia delle stime delle pose individuate nel flusso di immagini analizzato. Per quanto riguarda le informazioni utili a rendere l'elaborazione delle immagini meno onerosa ad ogni passo d'esecuzione dell'algoritmo, queste riguardano principalmente la finestra dell'immagine in cui limitare la successiva ricerca delle feature e la posa oggetto-telecamera stimata al passo precedente. La classe *ShapeTracker* (8.15) contiene il vettore dei limiti del sottografo del descrittore in cui ricercare il modello da confrontare e, relativamente all'elaborazione delle immagini, i limiti della porzione d'immagine utilizzata come finestra.

---

**Codice 8.15:** Classe *ShapeTracker*

---

```

1  class ShapeTracker {
2  private:
3      // Detector and Descriptor
4      ShapeDetector *detector;
5      ShapeDescriptor *descriptor;
6
7      // Subgraph levels and Region-Of-Interest
8      int levelsToBeScannedForMatching[2][4];
9      CvRect imageRoi;
10
11     // Model matched at previous step
12     ShapeModel *modelToTrack;
13
14 public:
15     /**
16     * @brief Tracker initialization
17     * @param img Observation

```

```
18  * @param res Result of initialization
19  * @return Number of matching models found
20  */
21  int init(IplImage* &img, ShapeModel* &res);
22
23  /**
24  * @param img Observation
25  * @param res Result of tracking
26  * @return TRUE if the object tracking is maintained
27  */
28  bool track(IplImage* &img, ShapeModel* &res);
29  };
```

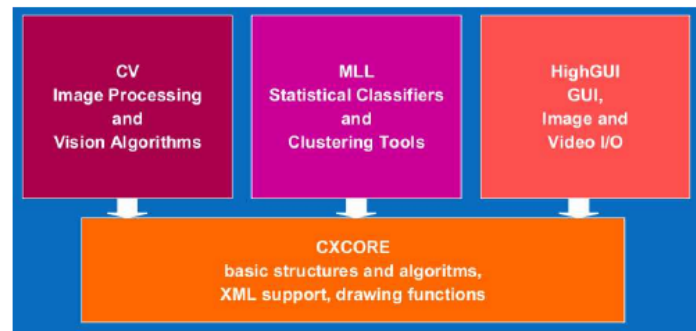
---

## 2.3 Libreria per visione artificiale: openCv

Per l'implementazione degli algoritmi strettamente legati alle problematiche di visione artificiale, ci si è avvalsi del supporto della libreria software *openCv* ([site10]). Essa mette a disposizione una serie di strumenti per l'elaborazione delle immagini, a partire dalle operazioni a livello dei singoli pixel fino ad arrivare allo sviluppo di applicazioni di riconoscimento complesso. La libreria è completamente open source e, come nel caso della libreria *libdc1394*, è disponibile sulle tre piattaforme principali (Linux, Windows, MacOSX). I moduli sono scritti principalmente in C/C++, ma attualmente sono in fase di sviluppo i suoi porting in altri linguaggi di programmazione, quali Python, Ruby e Matlab.

L'utilizzo delle *openCv* è particolarmente indicato per lo sviluppo di applicazioni di visione real-time poiché il codice è stato particolarmente ottimizzato proprio per queste applicazioni e per sfruttare l'elaborazione multi-core o via gpu. Le funzioni di visione artificiale messe a disposizione dell'utente sono più di 500 e coprono diversi ambiti operativi, dal controllo qualità alle applicazioni in ambito medico, dalla robotica alla visione stereoscopica, ...

Come mostrato in figura 8.9, la libreria è strutturata in cinque sottolibrerie (dette anche moduli) ciascuna con funzionalità specifiche. Il modulo *CXCORE* contiene le strutture dati di base con le rispettive funzioni di inizializzazione, le funzioni matematiche, le funzioni di lettura, scrittura e memorizzazione dati, le funzioni di sistema e di gestione degli errori. Il modulo *CV* fornisce le funzioni relative all'immagine processing, le funzioni di analisi strutturale e del moto, di object detection e ricostruzione 3D. *HIGHGUI* mette a disposizione le funzioni GUI e quelle di salvataggio e caricamento immagini, nonché le funzioni di acquisizione video e di gestione delle telecamere. Il modulo *ML* (Machine



**Figura 8.9:** Struttura della libreria software openCv

Learning) contiene classi e funzioni relative all'implementazione e gestione di reti neurali, in particolare di tipo multilayer perceptrons (MPL), di classificazione statistica e clustering di dati. Infine vi è il modulo *CVAUX* che offre sia le funzioni basate su algoritmi ancora in fase di sperimentazione, il cui futuro è quello di migrare nel modulo CV, e sia le funzioni considerate obsolete e quindi non più supportate. Le funzionalità di tale modulo sono rivolte in particolar modo alla corrispondenza stereo, al tracking 3D, al riconoscimento degli oggetti.

In questo lavoro di tesi ci si è limitati all'utilizzo delle openCv per quanto riguarda le sole elaborazioni locali e puntuali sui pixel (ad esempio gli operatori di filtraggio), senza utilizzare le implementazioni già messe a disposizione per quanto riguarda la fase di descrizione e riconoscimento degli oggetti di una scena.

### 3 Controllore RTAI

Per lo sviluppo del controllore in tempo reale, dove per controllore si intende il software per la lettura, elaborazione e generazione dei segnali di controllo, si è optato per una soluzione *pc-based*. La ricerca ha riguardato la scelta e utilizzo di un sistema operativo real-time, ovvero in grado di schedare i processi in funzione della loro priorità e, quindi, garantendo il rispetto di certi vincoli temporali predefiniti.

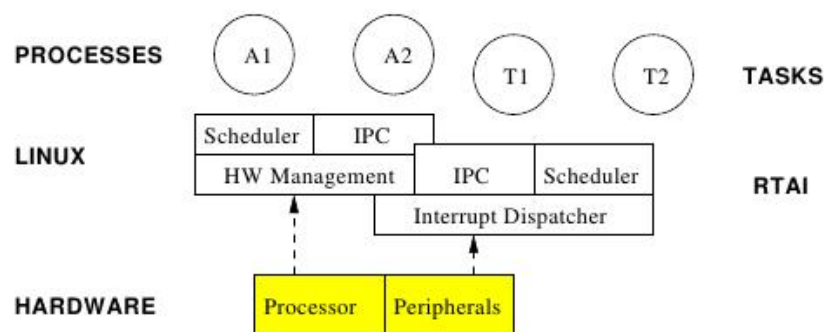
Tra i vari sistemi operativi real-time a disposizione, la scelta è ricaduta su *Linux RTAI* per i seguenti motivi principali:

- Compatibilità con le strutture POSIX;
- Distinzione tra servizi in spazio utente e spazio kernel;
- Possibilità di installazione su più piattaforme.

Come già visto per lo scambio dati relativo all'acquisizione di immagini, [9] descrive anche tutti gli aspetti e funzionalità dei sistemi operativi real-time.

### 3.1 Linux RTAI

*Linux RTAI* ([site5], [site6]) costituisce un'estensione del kernel dei sistemi operativi Unix che permette di modificare la gestione delle politiche di scheduling e interrupt dei task. Tali modifiche riguardano tre macrogruppi: l'astrazione hardware (*RTHAL*), i meccanismi di schedulazione dei task e l'estensione dello spazio utente all'esecuzione dei task real-time (*LX-RT*). Queste componenti sono riassunte in figura 8.10.



**Figura 8.10:** Componenti di *Linux RTAI*

Il livello di astrazione dell'hardware (*RTHAL*) è ovviamente dipendente dalla piattaforma su cui RTAI deve essere installato. Questa componente è codificata in linguaggio assembler che va a costituire il livello base per la compatibilità con le altre componenti del sistema operativo e i servizi real-time nello spazio utente. Questo livello viene attivato attraverso una patch del kernel base di Linux e contiene quindi tutti i dati e le funzioni temporalmente critiche del kernel, quali:

- gestione delle funzionalità base del kernel (interrupt e chiamate di sistema) con politiche real-time;
- utilizzo dei puntatori *RTHAL*;
- accesso prioritario ai dispositivi con kernel RTAI.

I moduli di RTAI sono sviluppati per operare accanto ai normali moduli Linux, la cui gestione viene lasciata alle componenti a bassa priorità (non real-time). Tali moduli gestiscono, ad esempio, la rete, le interfacce utente, gli IO ... Tuttavia, la struttura di questi componenti non è adatta per la gestione di processi real-time ed è pertanto *RTHAL* che fornisce una nuova implementazione di queste strutture più adatta.

Lo schedulatore del sistema operativo real-time è implementato come modulo kernel perché è tipicamente uno dei processi a priorità maggiore in esecuzione. Quando dei processi

real-time vengono creati, lo scheduler si occupa di assegnar loro priorità maggiore persino dello stesso kernel Linux. La principale funzione dello scheduler consiste nel decidere quale task deve essere messo correntemente in esecuzione e, per farlo, mette a disposizione servizi sui processi quali:

- *sospensione*;
- *messa in esecuzione*;
- *esecuzione periodica*;
- *messa in stato di attesa*.

Lo scheduler più utilizzato nei sistemi operativi real-time, e RTAI non fa eccezione, è lo scheduler FIFO perché esso permette l'esecuzione dei processi in funzione della loro priorità.

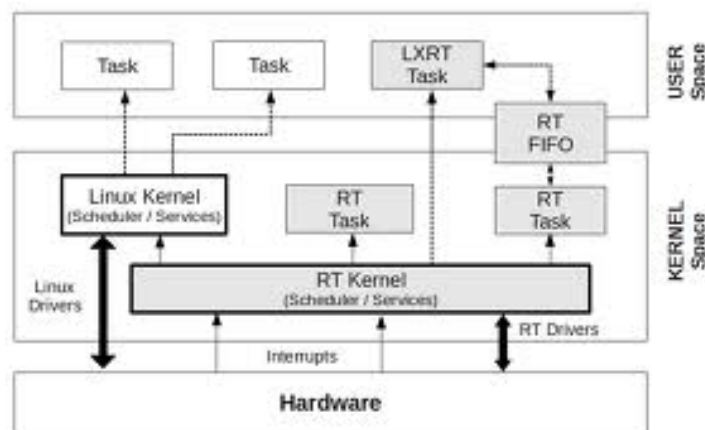
Il cuore del sistema operativo real-time è costituito dalle funzioni di *gestione del tempo* e della *memoria*. Per quanto riguarda il tempo, esiste un singolo clock di sistema in grado di tenere traccia e contare il tempo d'esecuzione di ogni singolo task, in modo da poter stabilire a quale processo assegnare la priorità massima nell'elenco dei processi da tenere in esecuzione. Questa gestione del tempo, inoltre, torna utile nello sviluppo di applicazioni di controllo, poiché l'informazione relativa al periodo d'esecuzione è resa disponibile ai processi della singola applicazione. Per quanto riguarda la memoria, invece, l'esecuzione in tempo reale è garantita dall'allocazione statica di tutte le pagine di memoria richieste da un particolare processo, sia quelle attualmente richieste, che quelle che verranno richieste nel corso dell'esecuzione del processo.

### 3.1.1 Linux LXRT

Il tipico approccio per la programmazione di task in tempo reale in Linux RTAI prevede l'implementazione di moduli, generalmente scritti in linguaggio C, eseguibili nello spazio kernel, ovvero in quell'area del sistema operativo gestita a basso livello. Nel corso di questo lavoro di tesi si è invece mirato ad indagare la programmazione di task real-time direttamente nello spazio utente, messo a disposizione dal sistema operativo senza la necessità di attaccarsi a moduli di basso livello. Questa scelta porta numerosi vantaggi, tra cui la possibilità (i) di programmare ad alto livello (C++), quindi sfruttando i principi della programmazione ad oggetti e (ii) di avere un maggiore controllo sull'esecuzione del task senza però rinunciare a nessuno dei requisiti di un task real time. Lo svantaggio è senza

dubbio costituito dall'impossibilità di spingere il sistema alle sue massime prestazioni, ma, nel caso di una soluzione *pc-based* questo problema non si pone poiché un pc industriale è in grado di garantire elevate frequenze operative (nell'ordine di qualche *kHz*) anche nello spazio utente.

La dicitura *LX/RT* ([site7]) sta per "Linux/real-time" e indica il complesso di moduli software che permette di offrire funzionalità *soft/hard* real-time ad un qualsiasi processo Linux eseguito nello spazio utente del sistema operativo. Per quanto riguarda RTAI, la differenza tra hard e soft real-time è relativa a quanto stringenti siano alcuni limiti posti al software, come ad esempio la necessità di determinare il tempo d'esecuzione di ogni singola linea di codice o la possibilità di avere lock sulla memoria a tempo indeterminato, ... A livello pratico, la differenza che si nota riguarda principalmente la stabilità dei segnali generati dai task: nel caso di un'applicazione *soft* real-time mancare una deadline è sì un episodio negativo e indice di errata programmazione, ma essa non provoca il blocco totale dell'applicazione, cosa che invece accade per la stessa applicazione configurata come *hard* real-time. Generalmente, il rispetto delle deadline è garantito dall'eliminazione di segnali di interrupt da parte di processi nello spazio utente che possono bloccare l'esecuzione dei processi in memoria. In questo modo, i moduli eseguiti dal kernel possono interrompere l'esecuzione del processo real-time nello spazio utente, mentre non possono farlo i normali processi del sistema operativo. In figura 8.11 viene mostrato come si integra un task LXRT nello spazio utente nell'architettura di RTAI. Come si vede, il task LXRT si pone ad un livello più alto dei task nello spazio Kernel, mantenendo tuttavia la comunicazione con i moduli Kernel. L'*IPC* (*Inter Process Communication*) per i task real-time viene trattata in seguito.



**Figura 8.11:** Spazio utente LX-RT nell'architettura RTAI

L'esecuzione di processi real-time nello spazio utente è resa possibile dalla presenza di un

apposito modulo real-time eseguito nello spazio kernel e con cui i processi utenti real-time si interfacciano per avere accesso alla memoria allocata e al segnale di tempo. Un task real-time allocato nello spazio utente può essere schedulato solo con lo *scheduler FIFO* e con la memoria completamente preallocata alla sua massima estensione possibile. Il preambolo necessario all'inizializzazione di ogni processo real-time è costituito dal codice 8.16.

---

**Codice 8.16:** *Inizializzazione di un task LXRT*

---

```

1  // Task initialization
2  RT_TASK *task;
3  task = rt_task_init_schmod(nam2num("MYTASK"), 9, 0, 0, SCHED_FIFO, 0
        xF);
4  // Memory allocation
5  mlockall(MCL_CURRENT | MCL_FUTURE);
6  // RTAI permissions (optional)
7  rt_allow_nonroot_hrt();
8  // Start realtime timer and scheduler
9  rt_set_oneshot_mode();
10 start_rt_timer(0);
11
12 // ...
13 // Main Task code
14 // ...
15
16 // Clean Stuff and stop timer
17 rt_task_delete(task);
18 stop_rt_timer();

```

---

Un oggetto della struttura *RT\_TASK* viene allocato in memoria assegnandogli un nome, lo scheduler FIFO, la dimensione dello stack e il numero di CPU da utilizzare. Successivamente, viene allocata la memoria richiesta all'esecuzione e avviato il contatore del tempo d'esecuzione del task stesso.

### 3.1.2 Task di controllo

Lo scopo finale di Linux LXRT in questo lavoro di tesi è quello di arrivare a definire gli oggetti software necessari all'inizializzazione di più processi real-time. La necessità primaria è la definizione dell'interfaccia di un generico controllore, costituito da:

- *inizializzazione* del processo. Permette di definire la frequenza e la modalità operativa del task.
- *inizializzazione dei segnali*. Consiste nel configurare le interfacce di lettura e scrittura dei segnali analogici e/o digitali.
- *lettura dei segnali*.
- *elaborazione e generazione del controllo*.
- *scrittura dei segnali sulle periferiche*.
- *terminazione del controllore*.

L'implementazione dell'oggetto per la creazione di thread real-time è mostrata nel codice 8.17.

---

**Codice 8.17:** *Inizializzazione di un task LXRT*

---

```

1  class Rtai_ControllerThread_Base : public PThread {
2  public:
3      enum RT_MODE{
4          RT_HARD = 0, ///< RT_HARD
5          RT_SOFT,    ///< RT_SOFT
6      };
7
8  private:
9      unsigned int control_period_ns;
10     Rtai_ControllerThread_Base::RT_MODE mode;
11     StringUnicode task_name;
12     bool initialized;
13
14 public:
15     Rtai_ControllerThread_Base(StringUnicode task_name, double
16         frequency,
17     Rtai_ControllerThread_Base::RT_MODE mode)
18     {
19         // Control Frequency and RT mode
20         this->control_period_ns = (1.0f / frequency) * 1000 * 1000 *
21             1000;
22         this->mode = mode;
23         this->task_name = task_name;
24
25         // Variables
26         initialized = false;

```



```
25
26     // Schedule Policy
27     this->setSchedPolicy(SCHED_FIFO);
28 }
29
30 protected:
31     void* run()
32     {
33         try{
34             // Make Thread Real-Time
35             RT_TASK *task;
36             task = rt_task_init_schmod(nam2num(task_name.toString()).c_str
37                                     (), 0, 0, 0,
38             SCHED_FIFO, 0xF); mlockall(MCL_CURRENT | MCL_FUTURE);
39
40             // Soft or Hard
41             switch (mode){
42                 case RT_SOFT:
43                     rt_make_soft_real_time();
44                     break;
45                 case RT_HARD:
46                     rt_make_hard_real_time();
47                     break;
48                 default:
49                     break;
50             }
51
52             // Init System
53             configureController();
54             initController();
55
56             // Periodic Thread
57             rt_task_make_periodic(task, rt_get_time() + nano2count(
58                                     control_period_ns),
59             nano2count(control_period_ns));
60
61             // Main Cycle
62             while (running){
63                 // Signals Elaboration
64                 acquireSignals();
```

---

```

63         elaborateSignals();
64         sendSignals();
65
66         // Wait for next period
67         rt_task_wait_period();
68     }
69
70     // Finalize Controller
71     endController();
72
73     // Clean Stuff
74     rt_task_delete(task);
75 }
76 catch (exception &e){
77     runtime_error error (e.what());
78     this->setRuntimeError(error);
79 }
80 return NULL;
81 }
82
83 virtual void initSignalParameters();
84
85 virtual bool configureController();
86
87 virtual bool initController();
88
89 virtual bool endController();
90
91 virtual bool acquireSignals();
92
93 virtual bool elaborateSignals();
94
95 virtual bool sendSignals();

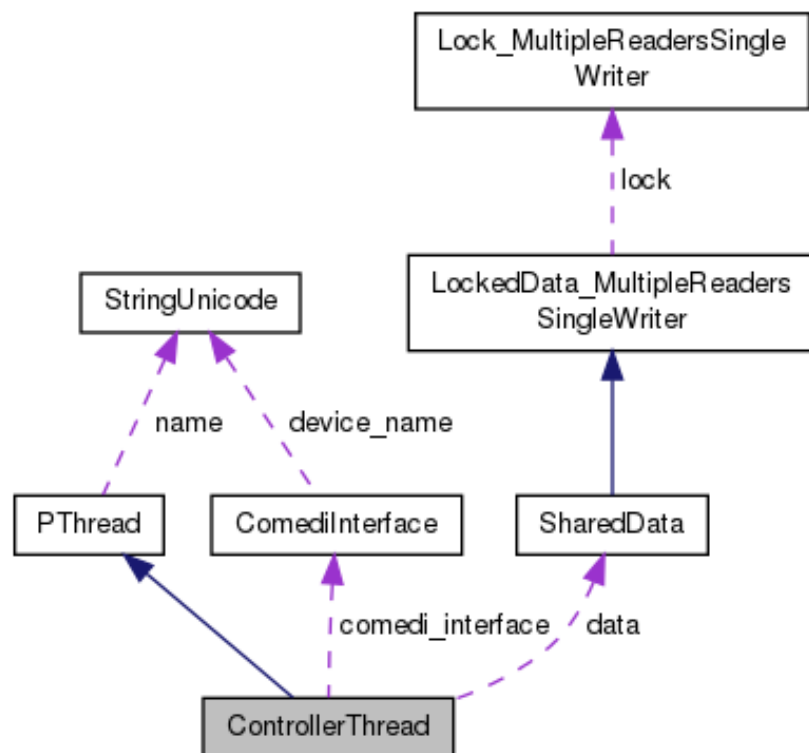
```

---

Rispetto all'inizializzazione del task real-time mostrata in precedenza, è stata introdotta la specifica della priorità real-time (*rt\_make\_soft\_real\_time()* e *rt\_make\_hard\_real\_time()*) e la frequenza di esecuzione del processo (*rt\_task\_make\_periodic(...)*) garantita dall'attesa del periodo di tempo corretto al termine dell'esecuzione di ogni ciclo del programma principale (*rt\_task\_wait\_period()*).

In figura 8.12 è riportato lo schema delle classi coinvolte nello sviluppo del thread control-

lore. Oltre all'oggetto controllore, viene messo in evidenza l'oggetto *SharedData*, utilizzato per la comunicazione tra differenti processi della stessa applicazione real-time.



**Figura 8.12:** Schema delle classi per l'oggetto *ControllerThread*

## 3.2 IPC: FIFO

La comunicazione tra processi real-time e non real-time è una delle componenti più delicate del sistema di controllo. Infatti, le frequenze operative in gioco sono talmente differenti (almeno 1 *KHz* per la chiusura degli anelli di controllo e nell'ordine di decine di *Hz* per i task ausiliari) che ci si ritrova ad affrontare le stesse problematiche già viste per il software di acquisizione immagini (vedi paragrafo 1 del capitolo 2). Per tali motivi, l'utilizzo della memoria condivisa non è più applicabile e si rende quindi necessario prendere in considerazione una struttura dati che permetta di disaccoppiare completamente i task. La struttura dati scelta è una coda di dati *FIFO* (*First In First Out*).

La memoria condivisa permette ai processi di accedere ai dati in un qualsiasi ordine, mentre le *FIFO* sono un meccanismo di comunicazione tra processi che costringe l'accesso sequenziale alle informazioni. Il produttore scrive i dati al termine della coda, mentre il consumatore preleva i dati dall'inizio della coda, nell'ordine in cui essi sono stati inseriti. Nell'implementazione delle *FIFO* nell'ambiente Linux RTAI, la posizione della coda in

memoria è completamente definita dal descrittore del file scelto per la comunicazione. Sia le code real-time che le classiche code per i processi non in tempo reale devono fornire le stesse procedure, che permettano di garantire:

- l'*inizializzazione* della coda, subordinata alla scelta del descrittore a livello di numero identificativo delle code riservate o di percorso del file di sistema (tipicamente a partire da `/dev/rxf0`);
- la *scrittura* di dati su una FIFO con la generazione di eventi di sistema quando la coda è piena;
- la *lettura* dei dati scritti sulla FIFO.

La differenza tra le due tipologie di FIFO risiede principalmente nella gestione del blocco del thread chiamante dopo le chiamate dei metodi di lettura e scrittura: nel caso non real-time, il processo chiamante si può bloccare in attesa di ricezione dei dati o della possibilità di scrivere sulla coda, mentre nell'altro caso il processo non si può mai bloccare, poiché questo comporterebbe il mancato rispetto dei vincoli temporali di esecuzione (deadline). Gli estratti di codice 8.18 e 8.19 mostrano rispettivamente l'implementazione di una coda lato non Real-Time e lato Real-Time.

---

**Codice 8.18:** Implementazione di una FIFO non RT

---

```
1  class FifoChannel
2  {
3  private:
4      int fifo_id;
5      string fifo_path;
6      bool opened;
7      bool non_blocking;
8
9  public:
10     enum Fifo_Open_Mode
11     {
12         READ_ONLY,          //!< READ_ONLY
13         WRITE_ONLY,         //!< WRITE_ONLY
14         READ_AND_WRITE,     //!< READ_AND_WRITE
15     };
16
17 public:
18     FifoChannel(string fifo_path)
```

```
19     {
20         fifo_id = -1;
21         this->fifo_path = fifo_path;
22         opened = false;
23         non_blocking = false;
24     }
25
26     int createFifo(mode_t permission)
27     {
28         int res = mkfifo(fifo_path.c_str(), permission);
29
30         if (res < 0){
31             switch (errno){
32                 case 0:
33                 case EEXIST: // Fifo already created
34                     break;
35                 default: // Generic Error
36                     return -1;
37             }
38         }
39
40         return res;
41     }
42
43     void openFifo(FifoChannel::Fifo_Open_Mode mode, bool non_blocking)
44     {
45         if (opened){
46             return;
47         }
48
49         this->non_blocking = non_blocking;
50         int flags = 0;
51
52         switch (mode){
53             case READ_ONLY:
54                 flags |= O_RDONLY;
55                 break;
56             case WRITE_ONLY:
57                 flags |= O_WRONLY;
58                 break;
```

```
59     case READ_AND_WRITE:
60         flags |= O_RDWR;
61         break;
62     default:
63         break;
64 }
65
66 if (non_blocking){
67     flags |= O_NONBLOCK;
68 }
69
70 fifo_id = open(fifo_path.c_str(), flags);
71
72 // Throw exception
73 if (fifo_id < 0){
74     StringUnicode error_string = "";
75     error_string = error_string + "FifoChannel error on opening
76         fifo at "
77         + fifo_path + " : " + strerror(errno);
78     throw runtime_error(error_string.toString());
79 }
80 else{
81     opened = true;
82 }
83
84 void closeFifo()
85 {
86     if (!opened){
87         return;
88     }
89
90     int res = close (fifo_id);
91
92     // Throw exception
93     if (res < 0){
94         StringUnicode error_string = "";
95         error_string = error_string + "FifoChannel error on closing
96             fifo at "
97             + fifo_path + " : " + strerror(errno);
```

```
97         throw runtime_error(error_string.toString());
98     }
99     else{
100         opened = false;
101     }
102 }
103
104 int writeOnFifo(const void *buf, size_t count)
105 {
106     int wr = write(fifo_id, buf, count);
107
108     // Throw Exception
109     if (wr < 0){
110         StringUnicode error_string = "";
111         error_string = error_string + "FifoChannel error on writing on
            fifo at "
112             + fifo_path + " : " + strerror(errno);
113         throw runtime_error(error_string.toString());
114     }
115
116     return wr;
117 }
118
119 int readFromFifo(void *buf, size_t count){
120     int rd = read(fifo_id, buf, count);
121
122     // Throw Exception
123     if (rd < 0 && errno != EAGAIN){
124         StringUnicode error_string = "";
125         error_string = error_string + "FifoChannel error on reading on
            fifo at "
126             + fifo_path + " : " + strerror(errno);
127         throw runtime_error(error_string.toString());
128     }
129
130     return rd;
131 }
132 };
```

---

---

*Codice 8.19: Implementazione di una FIFO RT*

---

```
1
2  class FifoChannel_RT
3  {
4  private:
5      int fifo_id;
6      bool created;
7
8  public:
9      FifoChannel_RT(int fifo_rt_id)
10     {
11         this->fifo_id = fifo_rt_id;
12         created = false;
13     }
14
15     void createFifo(int initial_size)
16     {
17         if (created){
18             return ;
19         }
20
21         int res = rtf_create(fifo_id, initial_size);
22
23         // Exception
24         switch (res){
25             case 0:
26                 // All ok...
27                 created = true;
28                 break;
29             case ENODEV:
30             {
31                 StringUnicode error_string = "";
32                 error_string = error_string + "FifoChannel_RT error on
33                     creating RT fifo "
34                     + fifo_id + " : fifo is greater than or equal to RTF_NO";
35                 throw runtime_error(error_string.toString());
36             }
37             break;
38             case ENOMEM:
```



```

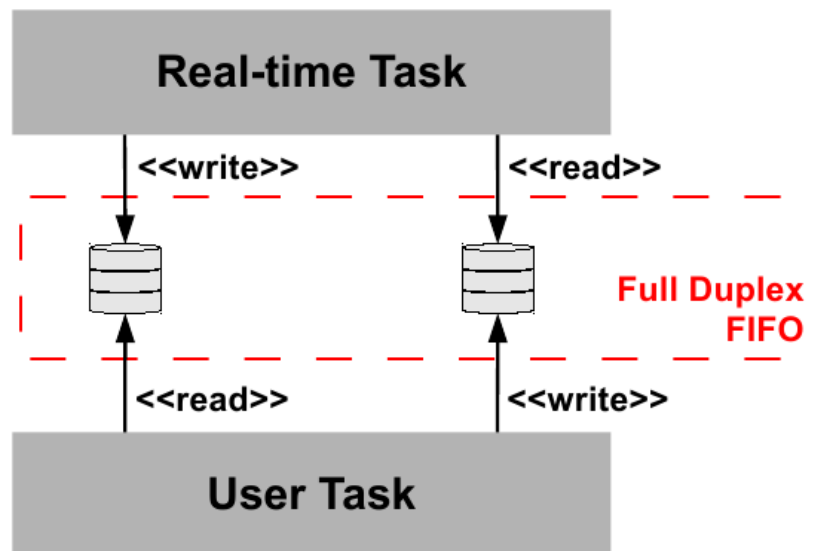
39     StringUnicode error_string = "";
40     error_string = error_string + "FifoChannel_RT error on
        creating RT fifo "
41         + fifo_id + " : the necessary size could not be allocated
            for the RT-FIFO";
42     throw runtime_error(error_string.toString());
43 }
44 break;
45 default:
46     break;
47 }
48 }
49
50 int writeOnFifo(const void *buf, size_t count)
51 {
52     int res = rtf_put (fifo_id, buf, count);
53
54     switch (res){
55     case 0:
56         // All ok
57         break;
58     case ENODEV:
59     {
60         StringUnicode error_string = "";
61         error_string = error_string + "FifoChannel_RT error on
            creating RT fifo "
62             + fifo_id + " : fifo is greater than or equal to RTF_NO";
63         throw runtime_error(error_string.toString());
64     }
65     break;
66     case EINVAL:
67     {
68         StringUnicode error_string = "";
69         error_string = error_string + "FifoChannel_RT error on
            creating RT fifo "
70             + fifo_id + " : fifo refers to a not opened fifo";
71         throw runtime_error(error_string.toString());
72     }
73     break;
74     default:

```

```
75         break;
76     }
77
78     return res;
79 }
80
81 int readFromFifo(void *buf, size_t count)
82 {
83     int res = rtf_get (fifo_id, buf, count);
84
85     switch (res){
86     case 0:
87         // All ok
88         break;
89     case ENODEV:
90     {
91         StringUnicode error_string = "";
92         error_string = error_string + "FifoChannel_RT error on
          creating RT fifo "
93             + fifo_id + " : fifo is greater than or equal to RTF_NO";
94         throw runtime_error(error_string.toString());
95     }
96     break;
97     case EINVAL:
98     {
99         StringUnicode error_string = "";
100        error_string = error_string + "FifoChannel_RT error on
          creating RT fifo "
101            + fifo_id + " : fifo refers to a not opened fifo";
102        throw runtime_error(error_string.toString());
103    }
104    break;
105    default:
106        break;
107    }
108
109    return res;
110 }
111 };
```

---

Tipicamente vengono create code monodirezionali (*half duplex*), ovvero aperte in sola scrittura da un task ed in sola lettura dall'altro task. Per gestire una comunicazione bidirezionale (*full duplex*), una soluzione tipica prevede la creazione di due code, come mostrato in figura 8.13.



*Figura 8.13: Implementazione di una FIFO full duplex*

### 3.3 ComediLib: comunicazione con le periferiche

L'ultima componente analizzata per la creazione del task di controllo è costituita dai moduli necessari all'inizializzazione della comunicazione con le periferiche, ovvero per permettere la lettura e scrittura dei segnali digitali e/o analogici. Tali componenti vengono forniti dalle librerie *Comedi* ([site1]). Queste sono un progetto open source che sviluppa driver e strumenti per la comunicazione con un grande numero di schede DAQ.

Il modulo *Comedi*, eseguito nello spazio kernel rende possibile la comunicazione anche dallo spazio utente allo stesso modo della scrittura/lettura delle FIFO, ovvero interagendo con l'apposito file creato nella cartella `/dev/comedi`. Il codice riportato in 8.20 evidenzia i metodi e strutture implementati per gestire la comunicazione con le periferiche.

*Codice 8.20: Interfaccia Comedi per la comunicazione con le periferiche*

```
1 class ComediInterface {
2 public:
3     enum DIO_MODE{
4         DIO_OUTPUT = 0, //!< DIO_OUTPUT
5         DIO_INPUT ,    //!< DIO_INPUT
```

```
6     };
7
8     private:
9         comedi_t *device;
10        StringUnicode device_name;
11        bool opened;
12
13    public:
14        ComediInterface(StringUnicode device_name)
15        {
16            this->device_name = device_name;
17            this->device = NULL;
18            this->opened = false;
19        }
20
21        // Opening and closing
22        void openDevice()
23        {
24            if (!opened){
25                device = comedi_open(device_name.toString().c_str());
26                if(device == NULL){
27                    StringUnicode msg = "";
28                    msg = msg + "Error on comedi_open for device " + device_name
29                        ;
30                    throw ComediException(msg.toString().c_str());
31                }
32                opened = true;
33            }
34        }
35
36        void closeDevice()
37        {
38            if (opened){
39                int retval = comedi_close(device);
40                if(retval < 0){
41                    StringUnicode msg = "";
42                    msg = msg + "Error on comedi_close for device " +
43                        device_name + " with error " + retval;
44                    throw ComediException(msg.toString().c_str());
45                }
46            }
47        }
48    };
49 }
```

```
44     }
45
46     device = NULL;
47     opened = false;
48 }
49 }
50
51 // Digital Input / Output
52 void configureDIO(unsigned int subdevice, unsigned int channel,
53     ComediInterface::DIO_MODE mode)
54 {
55     unsigned int dir = COMEDI_INPUT;
56     switch(mode){
57     case DIO_INPUT:
58         dir = COMEDI_INPUT;
59         break;
60     case DIO_OUTPUT:
61         dir = COMEDI_OUTPUT;
62         break;
63     default:
64         dir = COMEDI_INPUT;
65         break;
66     }
67
68     if (device){
69         int retval = comedi_dio_config(device, subdevice, channel, dir
70             );
71         if(retval < 0){
72             StringUnicode msg = "";
73             msg = msg + "Error on comedi_dio_config for device " +
74                 device_name +
75                 + ", subdevice = " + (int)subdevice + ", channel = " + (
76                     int)channel +
77                     ", with error " + retval;
78             throw ComediException(msg.toString().c_str());
79         }
80     }
81     else{
82         StringUnicode msg = "";
83         msg = msg + "Error on comedi_dio_config for device " +
```

```
        device_name +
80         + ", subdevice = " + (int)subdevice + ", channel = " +
81         (int)channel + ", DEVICE NOT INITIALIZED";
82         throw ComediException(msg.toString().c_str());
83     }
84 }
85
86 bool writeDO(unsigned int subdevice, unsigned int channel, bool
    bit)
87 {
88     if (!device){
89         return false;
90     }
91
92     unsigned int bit_u = bit ? 1 : 0;
93     int retval = comedi_dio_write(device, subdevice, channel, bit_u)
        ;
94     if(retval < 0){
95         return false;
96     }
97     return true;
98 }
99
100 bool readDI(unsigned int subdevice, unsigned int channel, bool &
    bit)
101 {
102     unsigned int bit_u = 0;
103     bit = false;
104
105     if (!device){
106         return false;
107     }
108
109     int retval = comedi_dio_read(device, subdevice, channel, &bit_u)
        ;
110
111     if (retval < 0){
112         return false;
113     }
114 }
```

```
115     bit = bit_u;
116     return true;
117 }
118
119 // Analog Input / Output
120 bool writeAO(unsigned int subdevice, unsigned int channel,
               unsigned int selRange, double value, unsigned int aref =
               AREF_GROUND)
121 {
122     if (!device){
123         return false;
124     }
125
126     // Get range and maxData
127     comedi_range *range = NULL;
128     range = comedi_get_range(device, subdevice, channel, selRange);
129     lsampl_t maxData = comedi_get_maxdata(device, subdevice, channel
        );
130
131     // Compute Output Value
132     lsampl_t output_sample = 0;
133     output_sample = comedi_from_phys(value, range, maxData);
134
135     // Send Output Value to device
136     int retval = comedi_data_write(device, subdevice, channel,
        selRange, aref , output_sample);
137     if (retval < 0){
138         return false;
139     }
140
141     return true;
142 }
143
144 bool readAI(unsigned int subdevice, unsigned int channel, unsigned
               int selRange, double &value, unsigned int aref = AREF_GROUND)
145 {
146     value = 0;
147
148     if (!device){
149         return false;
```

```
150     }
151
152     // Get range and maxData
153     comedi_range *range = NULL;
154     range = comedi_get_range(device, subdevice, channel, selRange);
155
156     lsampl_t maxData = comedi_get_maxdata(device, subdevice, channel
157                                         );
158
159     // Compute Output Value
160     lsampl_t input_sample = 0;
161
162     // Send Output Value to device
163     int retval = comedi_data_read (device, subdevice, channel,
164                                   selRange, aref, &input_sample);
165     if (retval < 0){
166         return false;
167     }
168
169     // Compute Read Value
170     value = comedi_to_phys(input_sample, range, maxData);
171
172     return true;
173 }
174
175 // Counters (for National Instruments DAQ)
176 void configureCounter_NI(unsigned int subdevice, vector<
177                         EncoderChannel>
178                         encoder_channels, lsampl_t counter_mode, unsigned int
179                         counter_channel)
180 {
181     //... Encoder Channels Configuration
182 }
183
184 bool readCounter(unsigned int subdevice, unsigned int channel,
185                 unsigned int selRange, lsampl_t &data, unsigned int aref =
186                 AREF_GROUND)
187 {
188     data = 0;
```



```
184     if (!device){
185         return false;
186     }
187
188     // Compute Output Value
189
190     // Send Output Value to device
191     int retval = comedi_data_read (device, subdevice, channel,
192                                   selRange, aref, &data);
193     if (retval < 0){
194         return false;
195     }
196     return true;
197 }
```

---

## 4 Conclusioni

Nel capitolo sono state presentate una serie di considerazioni utili alla risoluzione delle principali tematiche software da affrontare nell'implementazione di un generico software di controllo di un sistema in asservimento visivo. Gli estratti di codice o pseudo-codice presentati non intendono fornire la soluzione al problema nel suo complesso, ma si concentrano sulla soluzione di alcune specifiche parti del problema stesso. Esse pertanto vogliono essere una linea guida all'implementazione del software nel suo complesso.

Per quanto riguarda la parte di acquisizione immagine, non ci si è concentrati sull'elencare le funzioni messe a disposizione dalla libreria per l'interfacciamento con le telecamere firewire, ma si è invece posto l'accento sulle tematiche di comunicazione tra i processi coinvolti nell'acquisizione. Le funzionalità della libreria sono state analizzate con attenzione per lo sviluppo del software nella sua completezza, ma in questo caso non è sembrato utile riportare intere parti del codice implementato.

Relativamente allo sviluppo dell'algoritmo di visione, la trattazione poteva riguardare uno qualsiasi degli algoritmi descritti nei capitoli 3, 4 e 6. In realtà, si è deciso di concentrarsi sull'algoritmo di tracciamento basato su viste multiple perché esso ha permesso di evidenziare il modo in cui si sono affrontati i vari problemi di un tipico algoritmo di visione: (i) la creazione e salvataggio di un descrittore di un oggetto a partire dalla lettura

del suo modello 3D, (ii) la successiva individuazione dell'oggetto all'interno della scena e (iii) il mantenimento del tracciamento dell'oggetto precedentemente individuato.

Per quanto riguarda invece lo sviluppo delle componenti real-time dei controllori, si è riportata quasi interamente l'implementazione degli oggetti per la gestione dei thread real-time, della comunicazione tra processi e della gestione delle periferiche. Infatti, mentre nel caso dell'acquisizione immagini la gestione della comunicazione con la periferica è dipendente dall'interfaccia di comunicazione e prevede solamente la chiamata delle funzioni per inviare la richiesta di acquisizione di un frame, in questo caso lo sviluppo ha portato alla progettazione degli elementi di base di ogni sistema di controllo. Questi componenti possono infatti costituire il punto di partenza per lo sviluppo di altre applicazioni in ambiente real-time.

Infine, per tutte le componenti analizzate si è fornita una breve descrizione delle librerie software utilizzate, nel dettaglio le librerie *libdc1394*, *openCv*, *Linux RTAI* e le *Comedi*. In questo caso lo scopo non è stato elencare le funzioni messe a disposizione, ma evidenziare i motivi della scelta di una particolare libreria e perché essa si adatta al lavoro svolto.

I concetti e gli estratti di codice presentati troveranno applicazione nello sviluppo del software utilizzato per le prove sperimentali descritte nei capitoli seguenti e, più in generale, rappresentano la base per l'implementazione di un generico controllore industriale o di un sistema di visione.

## Parte III

### Validazione sperimentale



# Capitolo 9

## Posizionamento di una Tavola Rotante

### Indice

---

1	Architettura del sistema . . . . .	<b>409</b>
1.1	Disco . . . . .	410
1.2	Telecamera . . . . .	410
1.3	Motore e azionamento . . . . .	413
1.4	PC e schede DAQ . . . . .	415
2	Elaborazione immagini . . . . .	<b>422</b>
2.1	Costruzione del descrittore . . . . .	423
2.2	Individuazione dell'oggetto . . . . .	424
2.3	Tracciamento . . . . .	427
3	Sviluppo del controllo . . . . .	<b>428</b>
3.1	Controllo IBVS . . . . .	430
3.2	Controllo PBVS . . . . .	432
4	Prove sperimentali . . . . .	<b>435</b>
5	Conclusioni . . . . .	<b>443</b>

---

La prima applicazione sviluppata per validare la ricerca effettuata sul controllo in asservimento visivo ha avuto come fine ultimo la realizzazione di un sistema simile a quello presentato per la simulazione nel paragrafo 5 del capitolo 7. Di conseguenza, per quanto già detto nel paragrafo appena citato, si è pensato di realizzare ed implementare il controllo di un sistema ad un grado di libertà.

In particolare, non è stato preso in considerazione direttamente il sistema costituito dalla guida lineare a causa delle limitazioni imposte dalla guida stessa: la valutazione delle

prestazioni dinamiche del controllo agli estremi della guida è condizionata dalla presenza dei finecorsa che limitano il movimento del carrello. La soluzione adottata ha previsto la trasformazione del moto rettilineo in un moto rotatorio. Pertanto, come sistema da controllare è stato scelto un *disco*, in modo da non avere limiti nella rotazione dello stesso. L'introduzione della parte di asservimento visivo, nel caso della guida lineare, prevede il montaggio della telecamera sulla parte movimentata (il carrello), in modo da permettere l'implementazione di un controllore *IBVS eye-in-hand*. Nel caso del disco, al contrario, vista l'impossibilità di riprodurre tale configurazione, si è optato per l'introduzione di un controllo *eye-out-hand*: la telecamera è posta di fronte al disco, il quale presenta dei disegni (oggetti) in sovraimpressione; lo scopo del controllo è posizionare il disco in un certo modo, in funzione dell'oggetto scelto come riferimento. Rispetto al caso della guida lineare la tipologia di controllo è leggermente differente. La posizione della telecamera e, quindi, la scena osservata, non indica direttamente il movimento da imporre all'end-effector, ma essa deve individuare la posizione dell'end-effector. Pertanto, il disco può essere visto come l'end-effector del manipolatore.

In figura 9.1 è mostrata una possibile realizzazione del banco prova, in cui si possono distinguere tutte le componenti fondamentali: il disco la cui posizione è controllata da un attuatore elettrico pilotato a partire dalle informazioni sul moto della scena osservate dalla telecamera posta di fronte al disco.



**Figura 9.1:** Rappresentazione del banco prova realizzato

Il limite dato dall'utilizzo di un sistema ad un singolo grado di libertà è relativo, poiché anche compiti più complessi possono essere scomposti in più fasi in cui il movimento del sistema avviene su 1 o 2 gradi di libertà. Sono due esempi (i) [60], in cui vengono tracciati

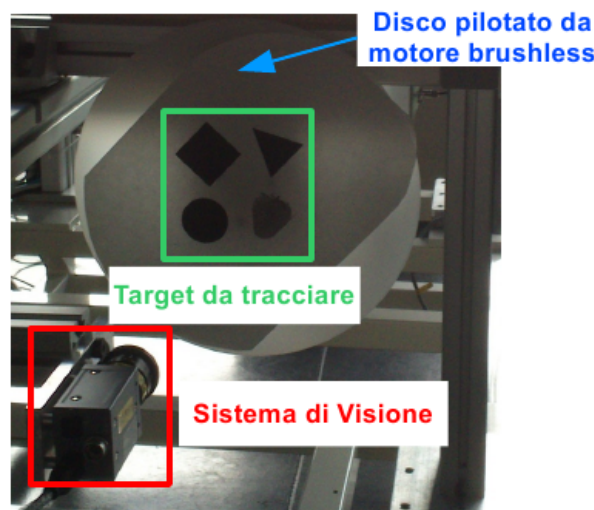
degli oggetti movimentati da un nastro trasportatore e (ii) [51], in cui si utilizza un sistema a 2 gradi di libertà per la validazione del metodo presentato.

In questo capitolo viene presentata nel dettaglio la realizzazione del banco prova. Innanzitutto si mostra la realizzazione del sistema dal punto di vista dei componenti scelti e della configurazione delle architetture hardware e software(1). In seguito vengono trattate l'implementazione dell'algoritmo di elaborazione immagini (2) e la sintesi del controllo (3), per la quale vengono mostrati i risultati della simulazione del sistema. Infine, vengono presentati i risultati delle prove sperimentali (4) e si traggono le conclusioni dell'esperienza effettuata (5).

## 1 Architettura del sistema

Il sistema risulta composto da diverse componenti, alcune delle quali sono mostrate in figura 9.2:

- Parte da movimentare *disco*;
- *Dispositivo di acquisizione immagini* (telecamera);
- *Attuatore elettrico* e relativo *azionamento*;
- *Calcolatore* (PC) per l'elaborazione dei segnali e la generazione del controllo.



*Figura 9.2: Componenti principali del sistema*

Dal punto di vista dell'architettura software progettata, la soluzione è basata sull'utilizzo di un unico calcolatore per l'espletazione delle varie funzioni. In particolare, l'applicazione è suddivisa in due componenti principali: (i) interfaccia utente e software di elaborazione

immagini e (ii) software di controllo. Il software di elaborazione immagini si occupa della comunicazione col dispositivo di acquisizione, mentre il controllore gestisce esclusivamente i segnali da leggere e/o scrivere sull'azionamento. L'integrazione del controllo in asservimento visivo è garantita dall'infrastruttura dedicata alla comunicazione, per la quale viene utilizzato il classico meccanismo a code FIFO presentato nel paragrafo 3 del capitolo 8. Pertanto, si può concludere che il sistema di controllo realizzato appartiene alla categoria dei controlli in asservimento visivo diretti (*direct visual servo*), i cui vantaggi e svantaggi sono già stati trattati a loro volta nel capitolo 8, paragrafo 2.1.

Nel corso del paragrafo vengono presentati i vari componenti prendendoli singolarmente, ovvero senza considerare la loro integrazione, la quale viene invece introdotta nel corso della trattazione.

## 1.1 Disco

Il *disco* costituisce la parte da movimentare del sistema. Questo è costruito in materiale sintetico che permetta di garantire la rigidità del sistema mantenendo al contempo contenuto il peso. Per quanto riguarda le dimensioni, il diametro esterno del disco è di 30 cm, ma l'area utile dipende esclusivamente dalla distanza della telecamera e dall'ottica montata su quest'ultima.

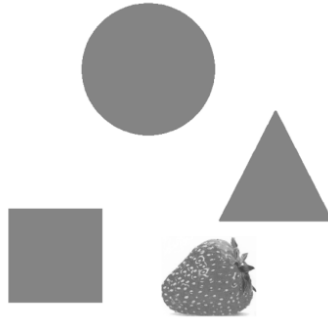
La caratteristica peculiare del disco è però data dalle immagini sovrainpresse. Poiché lo scopo ultimo del sistema è l'analisi delle prestazioni dell'algoritmo di controllo in asservimento visivo, sul disco sono state scelte figure semplici da riconoscere, ma che permettano, nel contempo, di valutare l'efficacia e robustezza degli algoritmi di elaborazione immagine scelti e implementati. In particolare, la scelta è ricaduta su tre forme geometriche differenti (cerchio, triangolo e quadrato) e su un oggetto di forma indefinita (fragola), come mostrato in figura 9.3.

## 1.2 Telecamera

Il componente che introduce la maggiore limitazione della banda passante del controllo in asservimento visivo è il dispositivo d'acquisizione immagini. Per questo banco sperimentale si è scelto di utilizzare la telecamera *Sony XCD V60-CR* ([site13]), mostrata in figura 9.4.

Questa telecamera costituisce una giusta via di mezzo tra qualità delle immagini acquisite





**Figura 9.3:** *Figure sovrainpresse sul disco*



**Figura 9.4:** *Telecamera Sony XCD V60-CR*

e framerate d'acquisizione. La qualità delle immagini è garantita dal sensore CCD Sony che permette l'acquisizione di immagini a colore in formato Bayer, alla massima risoluzione VGA di  $640 \times 480$  pixel. Questa risoluzione è piuttosto contenuta, ma la risoluzione spaziale ottenuta è più che sufficiente per gli scopi dell'applicazione che si intende sviluppare: supponendo di riuscire ad inquadrare unicamente il disco, si ottiene una risoluzione spaziale inferiore al mm, ovvero  $\frac{300}{480} \frac{\text{mm}}{\text{pixel}} = 0.625 \text{ mm/pixel}$ . L'utilizzo di una risoluzione contenuta garantisce l'ottenimento di un framerate piuttosto sostenuto, che, per la telecamera in oggetto arriva a 90 fps, grazie anche all'utilizzo dell'interfaccia firewire più prestante. Il protocollo di comunicazione con la telecamera è conforme allo standard *IEEE1394*, che ne permette anche la taratura dei parametri che influiscono direttamente la qualità dell'immagine (guadagno, saturazione, shutter, ...). In tabella 9.1 sono riportati tutti i dati tecnici della telecamera, come riportate sul datasheet della serie cui la *Sony XCD V60-CR* appartiene. Per permettere il montaggio della telecamera a distanza ridotta dal disco da inquadrare, si è deciso di montare un obiettivo *Tamron* a distanza focale corta, ovvero 4.8 mm (figura 9.5). Tale distanza focale è la minima ammessa senza invadere l'ambito degli obiettivi fish-eye, non particolarmente adatti all'applicazione da sviluppare a causa dell'alto grado di distorsione che essi introducono nelle immagini acquisite. Dato che la telecamera monta un sensore a  $1/3''$  e applicando le relazioni per la celta

**Tabella 9.1:** *Telecamera Sony XCD V60-CR, dati tecnici*

Sensore	CCD 1/3", scansione progressiva
Risoluzione	640 × 480 (VGA)
Dimensione pixel	7.4 × 7.4 $\mu m$
Interfaccia digitale	IEEE1394b-2002 x 2
Velocità di trasferimento dati	800/400/200/100 Mb/s
Framerate massimo	90 fps
Velocità shutter	1/100,000 ÷ 16 s

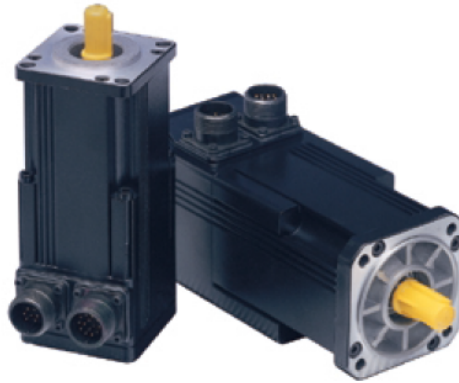
**Figura 9.5:** *Obiettivo Tamron, 4.8 mm*

degli obiettivi, ne risulta che per inquadrare l'intero disco la minima distanza operativa è 50 mm. In realtà, il supporto utilizzato per la telecamera, ne permette il montaggio ad una distanza inferiore, ovvero circa 30 mm. Questo significa che le figure geometriche da tracciare possono estendersi solamente per  $\frac{1}{3}$  della lunghezza del disco, occupando l'area di un cerchio di diametro 10 mm.

Tale riduzione dell'area utile per il tracciamento delle figura non è limitante per gli scopi dell'applicazione. Infatti, lo scopo del controllo in asservimento visivo è quello di ottenere posizionamenti precisi. In pratica, questo significa che l'assolvimento di un generico task di posizionamento può essere suddiviso in due fasi distinte: in primo luogo si fa in modo che la telecamera riesca ad inquadrare la scena desiderata posizionandola in un intorno della posa desiderata; nella seconda fase, invece, entra in gioco il controllo in asservimento visivo vero e proprio con il quale si correggono gli errori compiuti nel primo posizionamento. Data la dimensione ridotta dell'area inquadrata, questo banco di prova permette l'analisi della seconda fase del posizionamento, ovvero quella più delicata.

### 1.3 Motore e azionamento

Il motore messo a disposizione per la movimentazione del disco è il *Mavilor BLS-055* ([site8]) di figura 9.6.



**Figura 9.6:** Motore brushless Mavilor BLS-055

Questo è un motore elettrico brushless a prestazioni abbastanza elevate. Poiché il disco è decisamente leggero e piccolo (il peso complessivo è inferiore al  $Kg$ ), la coppia necessaria alla sua movimentazione è trascurabile ed è ampiamente al di sotto della coppia massima erogabile dal motore. Per tale motivo, il disco è innestato direttamente sull'albero motore dell'attuatore, senza bisogno di utilizzare riduttori meccanici. Pertanto, la massima velocità di rotazione del disco corrisponde alla massima velocità di rotazione del motore, ovvero  $10000\ rpm$ . Per completezza, in tabella 9.2 vengono mostrati tutti i dati tecnici del motore utilizzato.

**Tabella 9.2:** Motore brushless Mavilor BLS-055, dati tecnici

Velocità di rotazione massima	10,000	$rpm$
Coppia di stallo ( $\pm 10\%$ )	0.7	$Nm$
Coppia di picco ( $\pm 10\%$ )	2.8	$Nm$
Costante fem ( $\pm 5\%$ )	0.29	$Vs/rad$
Costante di coppia ( $\pm 5\%$ )	0.5	$Nm/A$
Resistenza concatenata ( $\pm 5\%$ )	14.7	$\Omega$
Induttanza concatenata ( $\pm 5\%$ )	18.6	$mH$
Inerzia del rotore	0.017	$Kg\ m^2\ 10^{-3}$

Il motore è fornito con un encoder incrementale con risoluzione inferiore al grado, di cui in

tabella 9.3 sono riportati i relativi dati tecnici. Per questa applicazione non è necessario leggere la posizione del motore attraverso l'encoder, poiché il controllo in posizione è completamente retroazionato dal sistema di visione. Uno dei vantaggi del controllo in asservimento visivo è infatti dato dalla possibilità di rendere stabili e performanti anche il controllo di sistemi non completamente calibrati. Tuttavia, l'encoder può essere letto per tenere sotto controllo il funzionamento del sistema e, per di più, deve essere necessariamente collegato all'azionamento del motore per permettere il controllo in velocità dell'attuatore elettrico. L'azionamento utilizzato per il controllo del motore è l'*Infranor XtrapulsPac 230*

**Tabella 9.3:** Motore brushless Mavilor BLS-055, encoder

Forma d'onda	Onda quadra
Numero di linee	500 ppr
Canali	A, B, Z Open Collector
Frequenza di risposta	100kHz

V ([site3]). Questo mette a disposizione diverse modalità di funzionamento e interfacce di comunicazione. Le modalità di controllo utilizzabili sono relative ai classici anelli del controllo in cascata degli attuatori elettrici: coppia, velocità e posizione. Il modulo può funzionare in modalità stand-alone dopo opportuna configurazione col suo software, oppure come slave, attraverso la comunicazione tramite interfaccia seriale RS-232, CanOpen o EtherCAT. In figura 9.7 l'azionamento in oggetto.



**Figura 9.7:** Azionamento Infranor XtrapulsPac 230 V

Per l'asservimento visivo il motore deve essere controllato in velocità. L'azionamento Infranor permette la chiusura dell'anello di controllo alla frequenza di  $10\text{ kHz}$ . Il modulo è stato configurato per il funzionamento stand-alone con controllo in velocità. A questo scopo, i collegamenti minimi necessari hanno riguardato (i) il freno del motore, tramite la morsettiera X8 e (ii) la fase di potenza, tramite morsettiera X9. Il pinout delle due morsettiere è mostrato rispettivamente in tabella 9.4 e 9.5.

**Tabella 9.4:** Azionamento Infranor XtrapulsPac 230 V, morsettiera freno (X8)

Pin	Segnale	I/O	Funzione
1	Brake-	O	Uscita freno motore
2	Brake+	O	Uscita freno motore
3	Brake In	I	Segnale per il relais del freno (opzionale)
4	24V	I	-
5	0V	I	-

Il controllo vero e proprio del motore è reso possibile dalla morsettiera X2, il cui pinout è mostrato in tabella 9.6. Sugli input son configurati il segnale di abilitazione (*enable*) del motore e di inibizione (*inhibit*) collegato ad un fungo di emergenza. Il segnale analogico differenziale (*ANA1+* e *ANA1-*) può essere configurato in modo da ricevere il segnale di riferimento in velocità:  $10\text{ V}$  corrispondono alla massima rotazione di velocità configurata ( $10000\text{ rpm}$  al massimo), mentre i segnali negativi permettono l'inversione del verso di rotazione.

## 1.4 PC e schede DAQ

Come accennato nell'introduzione del paragrafo, la soluzione adottata per il controllo dei vari dispositivi del sistema è *pc-based*. Infatti, un unico calcolatore si occupa della gestione di tutti i dispositivi, sia per l'acquisizione delle immagini che per la gestione dei segnali dell'azionamento.

Ad oggi, l'approccio tipico per la scelta di hardware dedicato all'elaborazione immagini prevede l'utilizzo di componenti con elevata capacità computazionale. Nel caso del banco di prova presentato, tuttavia, la sua semplicità non necessita di hardware particolarmente potente. Inoltre, la limitazione nella potenza computazionale dell'hardware permette di

**Tabella 9.5:** Azionamento Infranor XtrapulsPac 230 V, morsettiera potenza (X9)

Pin	Segnale	I/O	Funzione
1	U	O	Fase motore U
2	V	O	Fase motore V
3	W	O	Fase motore W
4	DC-	I/O	-
5	DC+	I/O	-
6	Rint	O	Resistenza freno interna 100 $\Omega$ / 35 W
7	DR	O	Transistor uscita freno
8	L1	I	Alimentazione 230 Vac principale
9	L2	I	Alimentazione 230 Vac principale
10	GND	-	Riferimento di massa per l'alimentazione principale

ottenere i risultati ottenuti con questa piattaforma sperimentale al caso del controllo di sistemi più complessi con hardware più prestante. In definitiva, il PC utilizzato è un PC industriale dotato di processore *Intel Pentium 4* a 3.0 GHz e 1 GB di memoria RAM. Inoltre, l'assenza di una scheda video dedicata costituisce un ulteriore collo di bottiglia per l'esecuzione degli algoritmi di elaborazione delle immagini, poiché la loro esecuzione è completamente a carico del processore.

Dal punto di vista software, il PC è equipaggiato con un sistema operativo Linux a 32 bit. Il kernel del sistema operativo è stato successivamente patchato con Linux-RTAI. Il sistema è stato completato con l'installazione delle librerie software (descritte nel capitolo 8) necessarie, ovvero la libreria *libdc1394* per la comunicazione con la telecamera, le *openCv* per l'elaborazione immagini e la *Comedi* per la comunicazione con la scheda d'acquisizione dati.

Come detto, il PC si deve interfacciare con tutte le periferiche del sistema, ovvero gli azionamenti e la telecamera. La comunicazione è resa possibile dall'estensione del PC con

**Tabella 9.6:** Azionamento Infranor XtrapulsPac 230 V, IO

Pin	Segnale	I/O	Funzione
1	ANA1+	I	Input analogico N. 1
10	ANA1-	I	differenziale
2	GND	-	Input analogico N. 2
11	ANA2	I	non differenziale
13	GND	I	Alimentazione per sensore
3	24 Vdc	I	ad effetto Hall
16	Encoder Output Z-	O	
7	Encoder Output Z+	O	
17	Encoder Output B-	O	
8	Encoder Output B+	O	
18	Encoder Output A-	O	
9	Encoder Output A+	O	

schede *PCI* dedicate.

Per quanto riguarda la gestione della telecamera, fornendo essa l'interfaccia di comunicazione IEEE1394, la gestione è possibile tramite la scheda d'espansione *Delock 89167* (figura 9.8) che fornisce il supporto per due bus firewire-B. Tra le sue caratteristiche principali si può elencare:

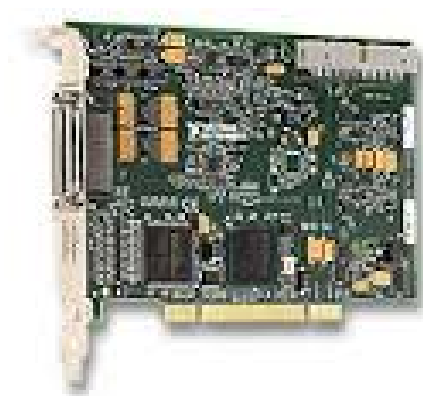
- IEEE 1394b to PCI Controller: Texas Instruments TSB82AA2;
- IEEE 1394b Transceiver/Arbiter: Texas Instruments TSB81BA3;
- Trasferimento fino a 800 Mbps, come definito dallo standard IEEE1394B;
- Bus PCI a 64 bit, retrocompatibile con PCI a 32 bit;
- Supporto per Linux 2.4.x.

Per quanto concerne l'acquisizione e gestione dei segnali, un altro bus PCI è stato occupato con la scheda *National Instruments PCI-6229* ([site9]), la quale rappresenta una soluzione entry-level per l'acquisizione dei dati. Questa fornisce l'interfaccia di comunicazione tra il PC e la relativa morsettiera su cui vengono effettuati i collegamenti con le varie periferiche. Scheda e morsettiera sono mostrate in figura 9.9.

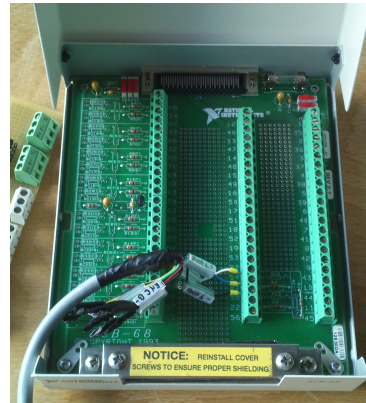
La scheda DAQ permette di gestire svariati tipi di segnali tra cui:



*Figura 9.8: Scheda d'espansione per il bus firewire*



(a) Scheda.



(b) Morsettiera.

*Figura 9.9: National Instruments PCI-6229*

- 32 input analogici “single-ended” o 16 differenziali. Range disponibili:  $\pm 10\text{ V}$ ,  $\pm 5\text{ V}$ ,  $\pm 1\text{ V}$ ,  $\pm 0.2\text{ V}$ ;
- 4 uscite analogiche  $\pm 10\text{ V}$ ;
- 48 I/O digitali;
- 2 contatori a 32 bit, utilizzabili per la lettura dei segnali provenienti da encoder.

A causa dell'alto valore di corrente richiesto dal segnale digitale per lo sblocco dei freni del motore, essi vengono sbloccati tramite l'opportuna interfaccia fornita dall'azionamento. Di conseguenza, per il controllo dell'azionamento del motore del disco è sufficiente utilizzare (i) un segnale digitale per l'abilitazione dell'azionamento, (ii) il segnale analogico per l'impostazione della velocità del motore e (iii) un contatore per la lettura dell'encoder virtuale dell'azionamento.

Facendo riferimento al pinout di figura 9.10, la scelta è ricaduta su:

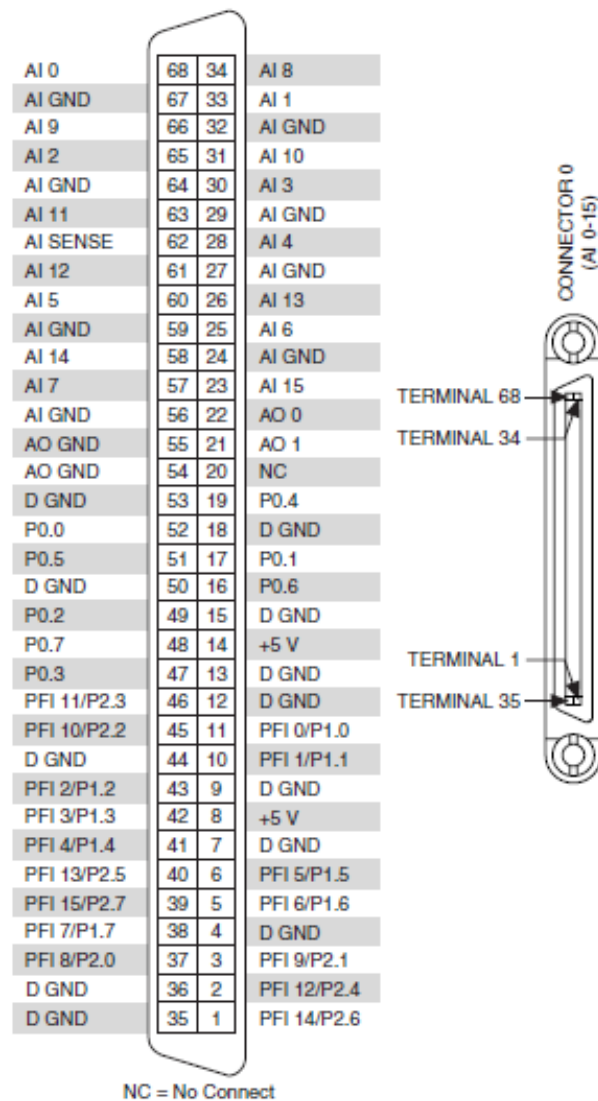
- Pin 21, uscita analogica per riferimento velocità;
- Pin 54, massa uscita analogica;
- Pin 42, canale A encoder;



- Pin 46, canale B encoder;
- Pin 17, abilitazione azionamento, tramite optoisolatore;
- Pin 18, massa segnale digitale.

Si può notare come vengono utilizzati solo due segnali per la lettura del segnale dell'encoder. Tuttavia, l'encoder virtuale fornito dal motore fornisce quattro segnali:  $A+$ ,  $A-$ ,  $B+$  e  $B-$ . Infatti, per ogni segnale viene fornito sia il segnale in fase che quello invertito. Per ottenere solamente due segnali utilizzabili, tra l'encoder e la scheda di acquisizione dati viene frapposto un *line-receiver*, ovvero un circuito integrato che permette di ottenere in uscita la differenza di due onde quadre. Applicando questa operazione ai segnali dell'encoder, si ottiene un'amplificazione del livello di segnale "alto", riducendo l'impatto del rumore sulla lettura della posizione dell'albero motore. L'integrato utilizzato è il *Texas Instruments AM26LS32A*, il cui funzionamento è mostrato in figura 9.11.

Infine, lo schema a blocchi di figura 9.12 mostra lo schema elettrico realizzato per l'acquisizione dei segnali e gestione dell'azionamento.



(a) Pinout.

Counter/Timer Signal	Default Connector 0 Pin Number (Name)
CTR 0 SRC	37 (PFI 8)
CTR 0 GATE	3 (PFI 9)
CTR 0 AUX	45 (PFI 10)
CTR 0 OUT	2 (PFI 12)
CTR 0 A	37 (PFI 8)
CTR 0 Z	3 (PFI 9)
CTR 0 B	45 (PFI 10)
CTR 1 SRC	42 (PFI 3)
CTR 1 GATE	41 (PFI 4)
CTR 1 AUX	46 (PFI 11)
CTR 1 OUT	40 (PFI 13)
CTR 1 A	42 (PFI 3)
CTR 1 Z	41 (PFI 4)
CTR 1 B	46 (PFI 11)
FREQ OUT	1 (PFI 14)

(b) Configurazione dei contatori.

Figura 9.10: Pinout della scheda NI PCI-6229

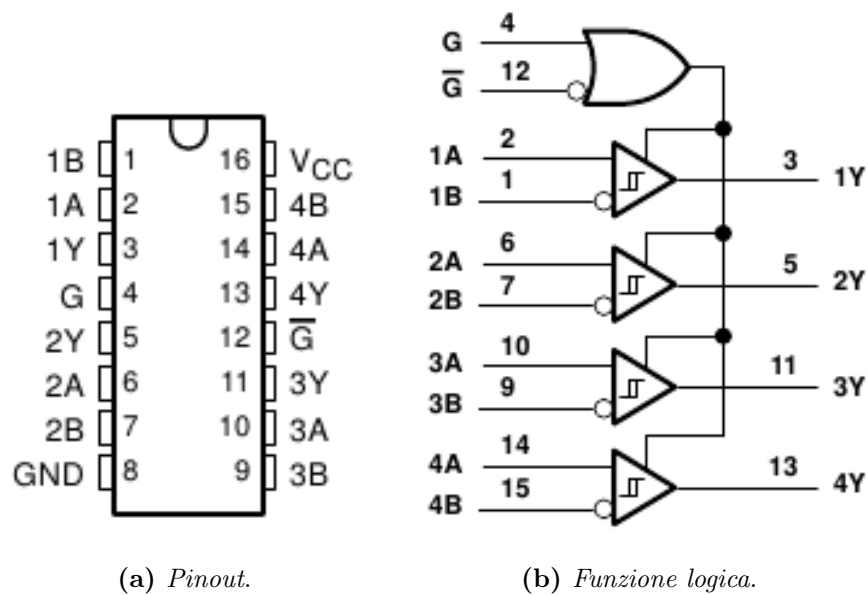


Figura 9.11: Line Receiver

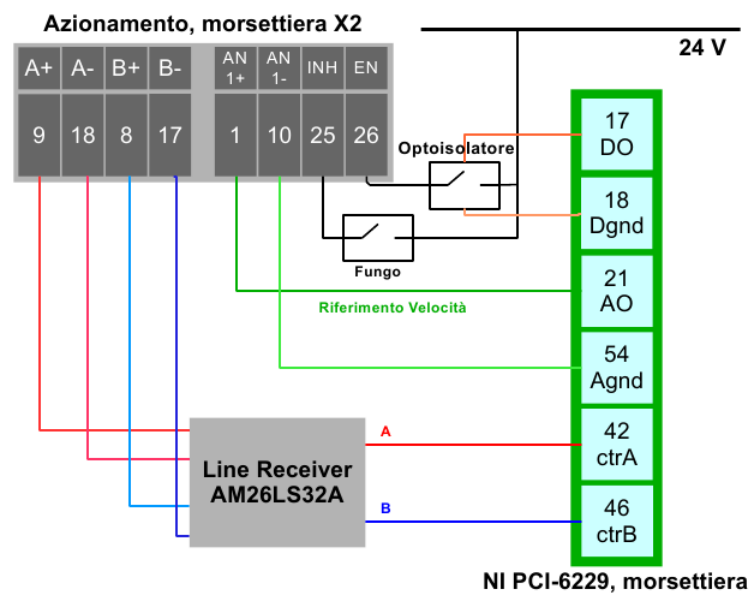


Figura 9.12: Schema dei segnali

## 2 Elaborazione immagini

Lo scopo dell'elaborazione delle immagini è il tracciamento di una qualsiasi delle figure sovrainpresse sul disco nel flusso delle immagini acquisite dalla telecamera durante l'esecuzione del task di posizionamento del disco. Pertanto, la spiegazione e implementazione dell'algoritmo utilizzato può attraversare le varie fasi di un classico algoritmo di tracciamento, ovvero:

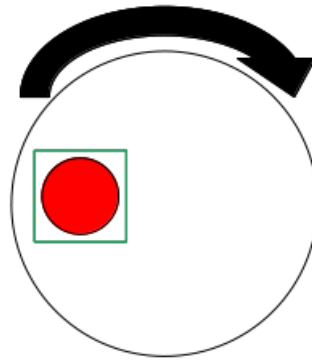
- Costruzione di un descrittore o modello dell'oggetto;
- Filtraggio e segmentazione della scena, al fine di utilizzare il modello completo per l'individuazione dell'oggetto d'interesse in una scena;
- Utilizzo di tecniche di finestramento e simili per irrobustire e velocizzare l'esecuzione dell'algoritmo di individuazione dell'oggetto nelle immagini successive.

L'algoritmo implementato costituisce un'applicazione della classe di algoritmi di tracciamento basati sulla costruzione di un modello a vista multipla degli oggetti. Rispetto all'algoritmo completo presentato nel capitolo 6, paragrafo 3, il caso in esame è uno dei più semplici immaginabili: gli oggetti da modellare sono puramente bidimensionali e gli unici spostamenti della scena possibili sono quelli dovuti alla movimentazione del disco, pertanto una rotazione attorno al suo asse. Ipotizzando di non conoscere la posizione del centro del disco sul piano immagine, il movimento degli oggetti può essere modellato con rotazioni attorno all'asse  $Z$  e traslazioni lungo gli assi  $X$  e  $Y$  della telecamera. Per questo motivo non sono considerati gli spostamenti che causano le maggiori distorsioni delle proiezioni degli oggetti sul piano immagine, ovvero le rotazioni attorno agli assi  $X$  e  $Y$ . Sempre nel paragrafo 3 del capitolo 6, è già stata dimostrata la robustezza dell'algoritmo relativamente al tracciamento di oggetti bidimensionali, ma l'applicazione al caso reale permette di evidenziare alcune problematiche non analizzate in fase di simulazione, tra cui:

- Effetti di distorsione dovuti al fenomeno del *blurring*, causato dall'utilizzo di un sensore CCD e dalla presenza di movimenti della scena a velocità variabile.
- Influenza dell'illuminazione e delle condizioni ambientali.

## 2.1 Costruzione del descrittore

Come già detto, la scena non viene sottoposta a movimenti che causano l'insorgenza di distorsioni particolari. Per tale motivo, il descrittore è costituito da un singolo set di *momenti invarianti di Hu* (vedi paragrafo 2.3, capitolo 3). Ad ogni modo, per irrobustire ulteriormente l'algoritmo di ricerca, i momenti di Hu non invarianti alle rotazioni vengono comunque calcolati imponendo alla singola proiezione di un oggetto rotazioni di passo  $10^\circ$ , ottenendo per ogni momento geometrico un insieme di valori  $\mathbf{M}_i$ , come mostrato in figura 9.13. Per fare in modo che il matching venga eseguito su un solo set di valori, per ogni



**Figura 9.13:** Modalità di costruzione del modello

insieme di momenti vengono calcolati media e deviazione standard. Pertanto, il descrittore di ogni oggetto è costituito da una serie di tuple  $M = \{m_\mu^i, m_\sigma^i\}$ :

$$\mathbf{D} = \{ \{m_\mu^1, m_\sigma^1\}, \{m_\mu^2, m_\sigma^2\}, \dots, \{m_\mu^n, m_\sigma^n\} \} .$$

In figura 9.14 è mostrata l'immagine acquisita dalla telecamera e due possibili rotazioni del disco utilizzate per la costruzione del modello.

In 9.1 è mostrato lo pseudocodice della fase di costruzione del descrittore.

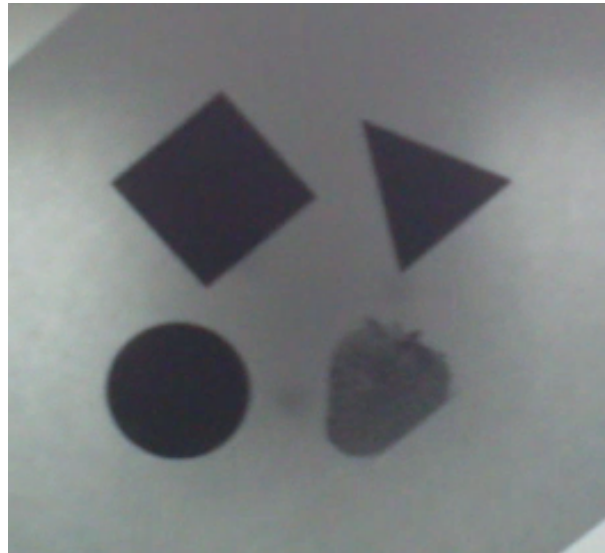
---

### Codice 9.1: Pseudocodice per il calcolo del descrittore

---

```

1  Array<Modello> models;
2  Map<Modello,Descrittore> descriptors;
3  for each Modello m in models
4      Array<Moments> moments;
5      ctr=0;
6      for angle=0:10:360
7          Modello m_rotated = rotate_modello(m, angle);
8          moments[ctr] = compute_moments(m_rotated);
```

(a) *Immagine originale.*(b) *Immagine ruotata N. 1.*(c) *Immagine ruotata N. 2.***Figura 9.14:** *Rotazioni dell'immagine del disco*

```

9      ctr++;
10     end
11     descriptors.at(m).second() = compute_mean_and_std(moments);
12 end

```

---

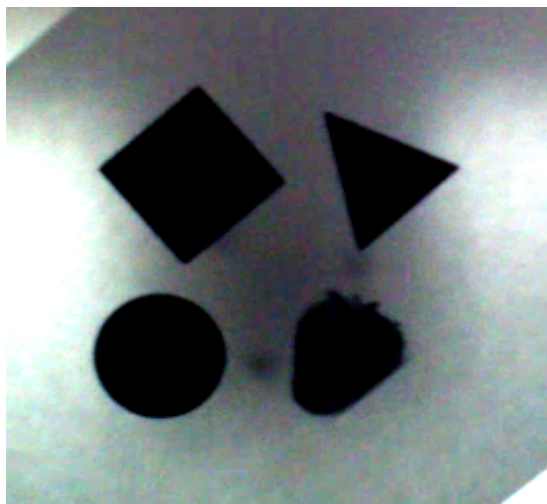
## 2.2 Individuazione dell'oggetto

Essendo il sistema da controllare un sistema ad un solo grado di libertà, è sufficiente individuare un solo punto dell'oggetto per calcolare la legge di controllo in asservimento

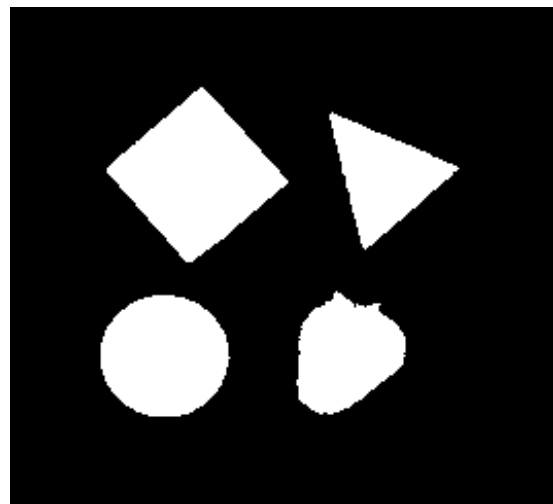
visivo. Il punto più comodo è dato dal centroide della figura individuata.

La fase di individuazione dell'oggetto è suddivisa in più sottofasi:

1. Conversione dell'immagine in scala di grigi. La telecamera permette di acquisire immagini a colore, ma l'elaborazione non necessita dell'informazione di colore. L'immagine di partenza è la prima mostrata in figura 9.14;
2. Sogliatura dell'immagine. I parametri della telecamera vengono tarati in modo da massimizzare il contrasto dell'immagine ottenuta. Infatti, gli oggetti rappresentati sul disco sono già in formato binario: gli oggetti sono scuri, mentre lo sfondo è chiaro. Pertanto la segmentazione è ottenuta sogliando l'immagine con un valore di soglia costante e prossimo allo zero. Il risultato dell'operazione, mostrato in figura 9.15 insieme all'immagine con alto contrasto, è un'immagine binaria in cui lo sfondo è nero e gli oggetti bianchi;



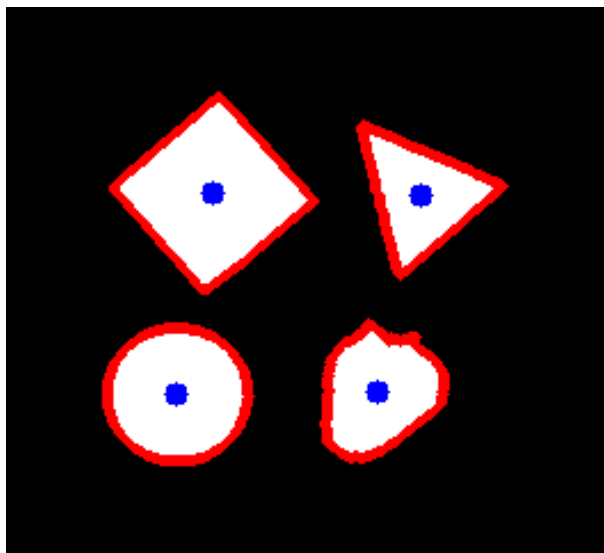
(a) Immagine con contrasto aumentato.



(b) Immagine segmentata.

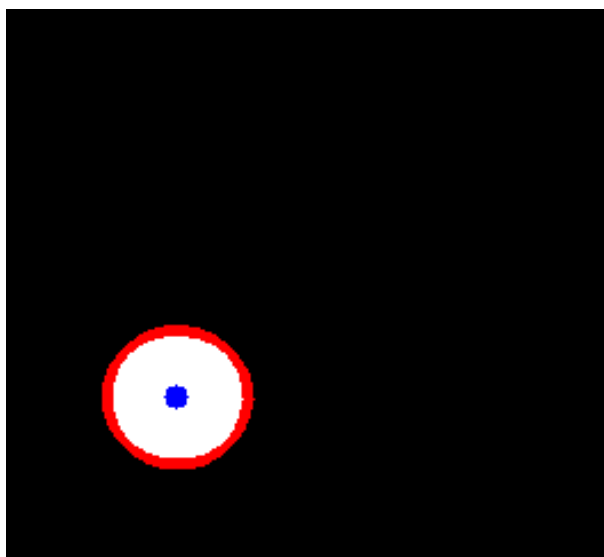
**Figura 9.15:** Segmentazione dell'immagine

3. Individuazione dei blob di colore presenti nell'immagine. Questa operazione può essere eseguita identificando i contorni della scena. Idealmente viene individuato un contorno per ogni oggetto raffigurato, come mostrato in figura 9.16, dove vengono evidenziati anche i centri geometrici degli oggetti;
4. Scansione dei contorni individuati e confronto col modello dell'oggetto ricercato. Per ogni contorno si calcolano i momenti invarianti di Hu e li si confronta con quelli inseriti calcolati nella costruzione del descrittore: se tutti i momenti identificati si trovano all'interno dell'intervallo di confidenza scelto nel descrittore, l'oggetto viene



*Figura 9.16: Individuazione dei blob e dei centri geometrici*

marcato come individuato. In figura 9.17 è mostrato il risultato della ricerca del cerchio;



*Figura 9.17: Individuazione dell'oggetto*

5. il contorno dell'oggetto individuato viene utilizzato per il calcolo dei momenti centrali, in modo da ottenere le coordinate, in pixel, del centro geometrico dell'oggetto, le quali costituiscono il punto di partenza della legge di controllo in asservimento visivo.

Lo pseudocodice dell'algoritmo di individuazione è mostrato in 9.2.

---

**Codice 9.2:** Pseudocodice per l'individuazione di un oggetto

---

```

1  Blob detect_object(Array<Blob> blobs, Modello m_to_be_searched,
    Descrittore

```



---

```

2  descriptor, int distance_thr)
3      Blob res = NULL;
4      int distance_min = distance_thr;
5      for each Blob b in blobs
6          Moments m = compute_moments(b);
7          int distance = match_moments(m, descriptor.moments);
8          if (distance < distance_min)
9              res = b;
10         end
11     end
12     return res;
13 end

```

---

Ai fini del controllo, potrebbe essere necessario calcolare l'*orientamento* del centro geometrico dell'oggetto. Questo può essere ottenuto facilmente una volta note le coordinate in pixel del centro del disco  $(x_c, y_c)$ . Queste ultime rappresentano quindi un ulteriore parametro di calibrazione del sistema. L'orientamento è quindi dato dalla relazione (9.1).

$$\theta_{obj} = \arctan\left(\frac{y_{obj} - y_c}{x_{obj} - x_c}\right). \quad (9.1)$$

## 2.3 Tracciamento

Relativamente alla fase di tracciamento, sono state considerate due possibili implementazioni:

- Reiterazione dell'algoritmo di individuazione dell'oggetto sull'intera immagine;
- Applicazione dell'algoritmo di identificazione su una regione dell'immagine ridotta. L'area di ricerca dell'oggetto è individuata tramite tecniche di finestramento (vedi capitolo 3, paragrafo 4).

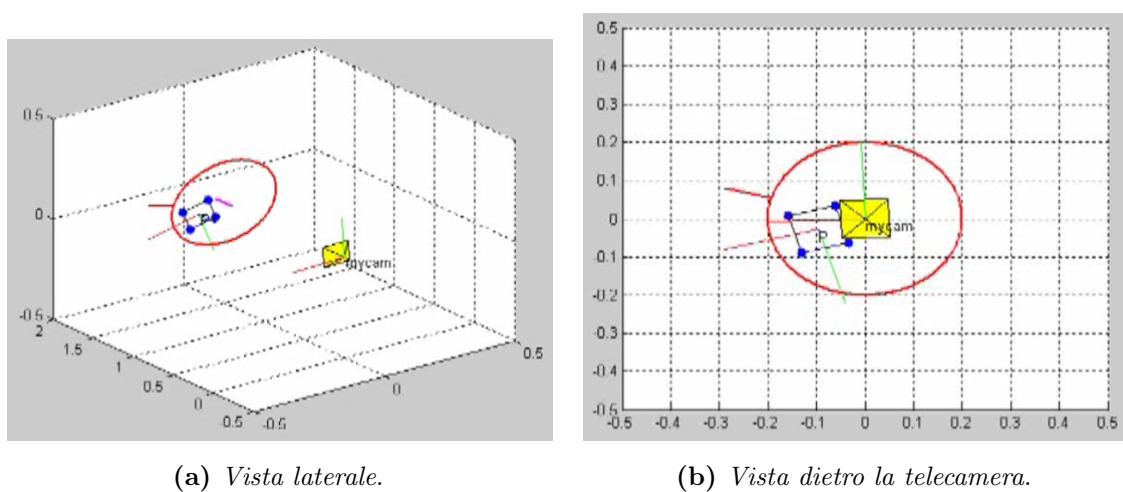
La prima implementazione è la più facile da realizzare perché prevede una semplice iterazione dell'algoritmo su ogni immagine acquisita. Da un certo punto di vista è anche la soluzione più robusta, poiché la correttezza dell'esecuzione dell'algoritmo dipende unicamente dalla robustezza del descrittore costruito e dall'ampiezza dell'intervallo di confidenza scelto. Tuttavia, questa implementazione è la più onerosa dal punto di vista computazionale.

La seconda possibilità, al contrario, permette di ridurre il tempo e le risorse dedicati all'esecuzione dell'algoritmo di individuazione della posa dell'oggetto. Tuttavia, la correttezza

del risultato dell'algoritmo scelto dipende dalla scelta della finestra di ricerca dell'oggetto nelle immagini successive, soprattutto dalla scelta della dimensione di tale finestra. La regola empirica utilizzata relaziona la dimensione della finestra scelta alla velocità di rotazione del disco, ovvero al guadagno proporzionale del controllore scelto.

### 3 Sviluppo del controllo

Per la sintesi del controllo in asservimento visivo si sono seguiti gli approcci presentati nei capitoli 5 e 7. In questo paragrafo vengono presentate le considerazioni fatte per la taratura di un controllore visual servoing proporzionale, il quale si è dimostrato più che sufficiente per rispettare i criteri di stabilità e comportamento dinamico desiderati. In figura 9.18 è riportato il sistema sottoposto a simulazione, in cui si nota la telecamera posta di fronte al disco, sul quale è sovrainpresso il solito oggetto target composto da quattro punti.



**Figura 9.18:** Rappresentazione del sistema in simulazione

Per la realizzazione della simulazione è stato necessario completare gli oggetti *Robot* e *Camera* con i loro dati tecnici, presentati in 1.

Per quanto riguarda la telecamera, si è supposto un tempo di elaborazione di circa 30 ms, ovvero metà del framerate d'acquisizione. Tale stima del tempo d'elaborazione è compatibile con le caratteristiche del calcolatore utilizzato, considerando che lo stesso deve garantire l'esecuzione real-time del task di controllo e gestire contemporaneamente l'acquisizione delle immagini.

Per il completamento delle parti relative al robot, composto in questo caso dall'accoppiamento tra motore e disco, invece, è stato necessario formulare le relazioni cinematiche e dinamiche. Come detto nel paragrafo di presentazione del sistema 1, l'albero motore è innestato direttamente sul disco, pertanto non è necessario considerare nessun rapporto di riduzione tra le due componenti. Detto ciò, la cinematica è molto semplice, poiché le rotazioni imposte al motore ( $q$ ) sono pari alle rotazioni ottenute sul disco ( $\theta$ ), come evidenziato dalle equazioni (9.2).

$$\begin{cases} \theta = q \\ \dot{\theta} = \dot{q} \\ \ddot{\theta} = \ddot{q} \end{cases} . \quad (9.2)$$

Per quanto concerne la dinamica del sistema, la coppia da imporre al motore ( $C_{mot}$ ) per ottenere una certa accelerazione  $\ddot{q}$  dipende dall'inerzia del motore stesso ( $J_{mot}$ ) e dall'inerzia del disco ( $J_d = \frac{1}{2}M_d R^2$ ), come esplicitato dalla (9.3).

$$C_{mot} = (J_{mot} + J_d)\ddot{q} . \quad (9.3)$$

Infine, al fine di muovere il target sulla superficie del disco in fase di simulazione, sono state scritte le relazioni per la rotazione di un punto qualsiasi del disco. In particolare, poiché l'asse  $Z$  del disco è parallelo all'asse  $Z$  della telecamera, la rotazione dei punti del disco è modellata semplicemente come una rotazione attorno all'asse  $Z$ . Detta  $\theta$  una rotazione imposta al disco, le nuove coordinate  $\mathbf{X}_1 = [x_1 \ y_1 \ z_1]^T$  del punto  $\mathbf{X} = [x \ y \ z]^T$  sono date dalla (9.4).

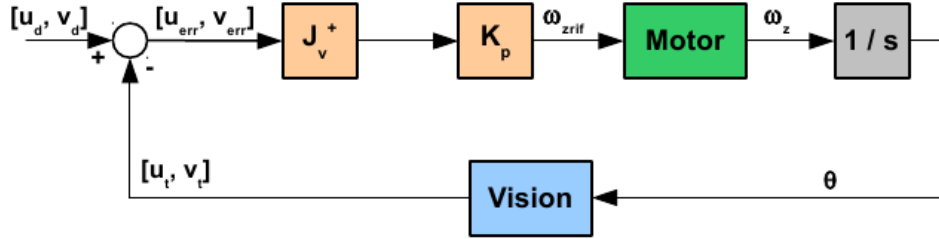
$$\mathbf{X}_1 = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \mathbf{X} . \quad (9.4)$$

Le coordinate  $\mathbf{X} = [x \ y \ z]^T$  nello spazio di lavoro sono ottenute a partire dalle coordinate  $\mathbf{U} = [u \ v]^T$  nello spazio immagine, secondo la ben nota relazione (2.5) ed essendo nota la distanza  $D_d^T$  tra disco e telecamera lungo l'asse  $Z$ . Il risultato è dato dalle relazioni (9.5).

$$\begin{cases} x = \frac{u - c_x}{D_d^T} \\ y = \frac{v - c_y}{D_d^T} \\ z = D_d^T \end{cases} . \quad (9.5)$$

### 3.1 Controllo IBVS

Il primo schema di controllo analizzato è il controllo *Image-Based Visual Servoing*, il cui schema a blocchi è riportato in figura 9.19. Questo controllo prevede l'utilizzo dello



*Figura 9.19: Schema di controllo del disco IBVS*

jacobiano immagine per riportare l'errore nello spazio immagine in un errore nello spazio di lavoro. Ipotizzando che il movimento da imporre sia solo una rotazione attorno all'asse  $Z$ , lo jacobiano immagine del singolo pixel si semplifica notevolmente poiché si rende necessario considerare unicamente l'ultima colonna della matrice. Lo jacobiano risultante è dato dalla (9.6), dove  $\frac{f_x}{f_y} = \frac{4}{3}$  poiché la risoluzione della telecamera è  $640 \times 480$ .

$$\mathbf{J}_v = \begin{bmatrix} -\frac{4}{3}(v - 240) \\ \frac{3}{4}(u - 320) \end{bmatrix}. \quad (9.6)$$

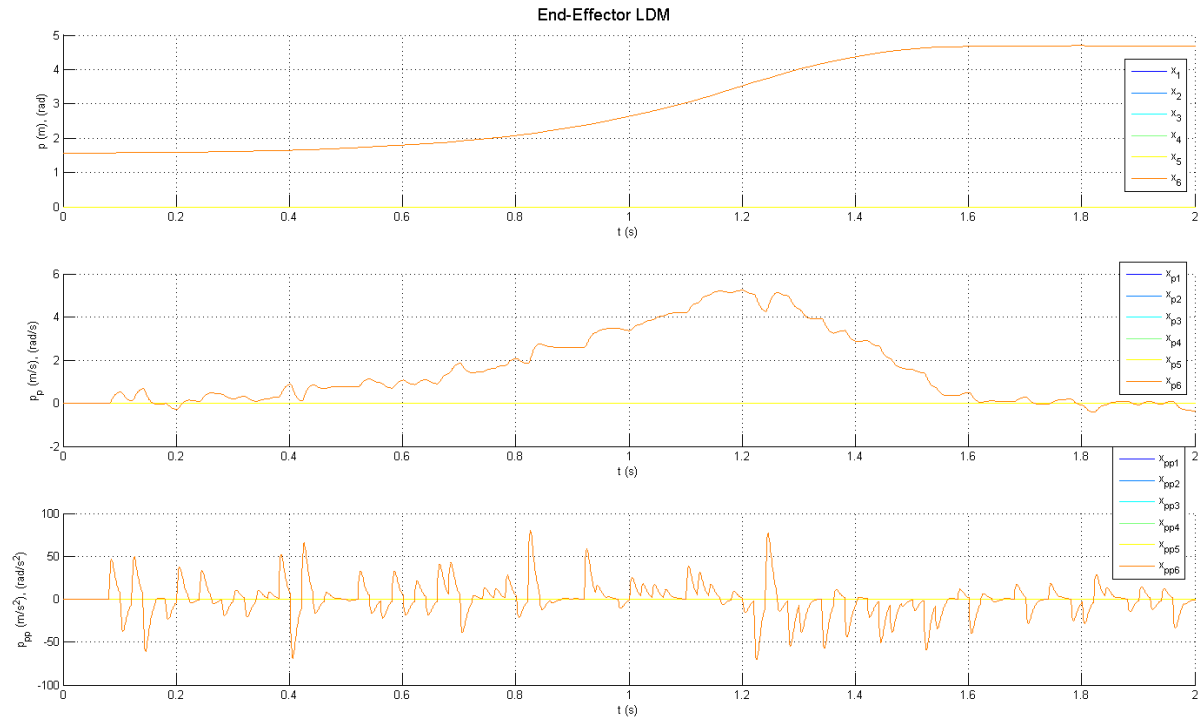
Di conseguenza, data la posizione desiderata del target direttamente in coordinate del piano immagine  $(u_d, v_d)$ , si può ottenere la velocità di rotazione da imporre al disco ricorrendo alla matrice pseudoinversa dello jacobiano, come esplicitato dalla (9.7), dove  $K_p$  è il guadagno proporzionale imposto dal regolatore.

$$\omega_z = K_p \cdot \mathbf{J}^+ \begin{bmatrix} u_d \\ v_d \end{bmatrix}. \quad (9.7)$$

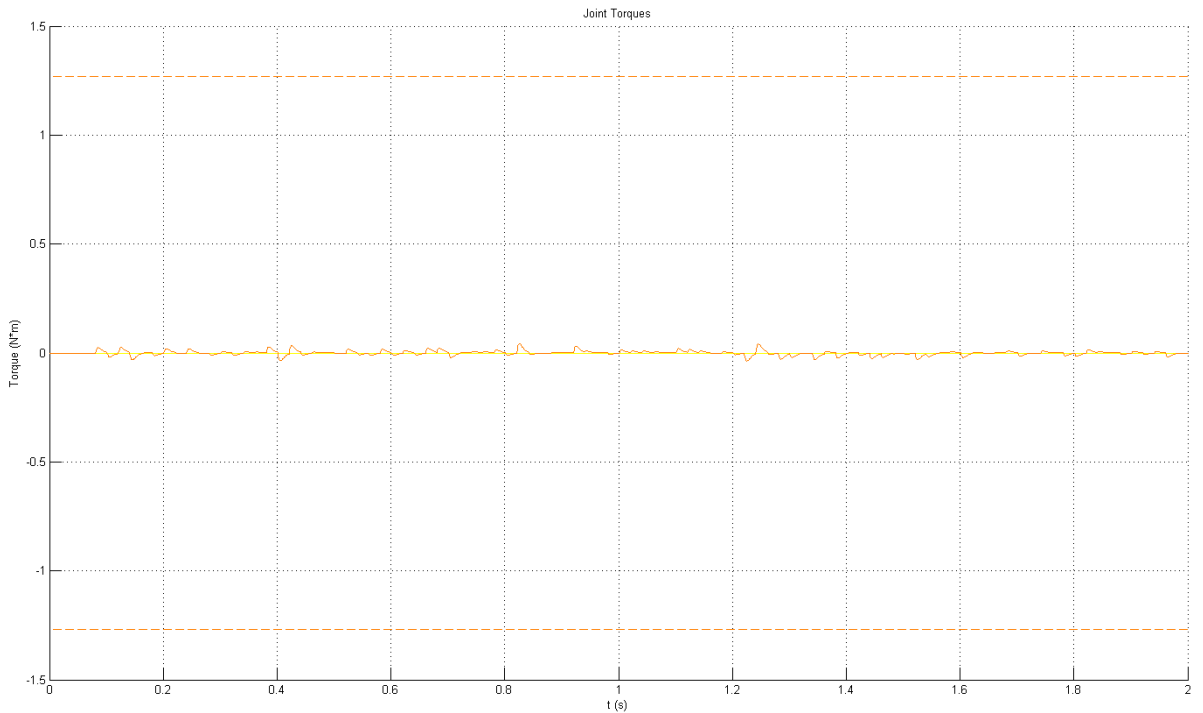
In figura 9.20 è mostrata la risposta del sistema ad uno scalino di circa  $180^\circ$  con  $K_p = 4$ . Come si vede, la traiettoria imposta non è lineare: il disco parte lentamente per poi accelerare solamente verso la metà della traiettoria. Questo è dovuto al fatto che l'utilizzo dello jacobiano immagine non ha come scopo l'ottimizzazione delle traiettorie, ma solamente l'ottenimento di una traiettoria che permetta la corretta esecuzione del task. Nel caso specifico, non si riesce a trovare una relazione diretta tra la posizione corrente del target e la rotazione da imporre al disco per raggiungere la posa desiderata. Tuttavia, viene impostata una piccola componente di rotazione e man mano che il sistema avanza,

la relazione tra posa corrente e posa desiderata si delinea sempre di più, fino ad ottenere la rotazione desiderata.

Infine, osservando il diagramma della coppia, si può notare che il contributo richiesto è estremamente basso.



(a) Profilo di velocità.

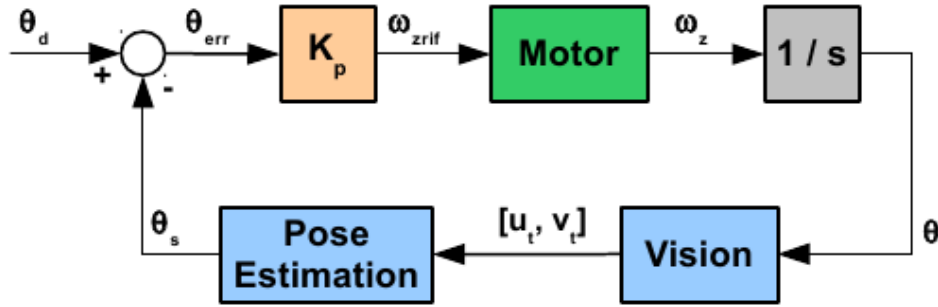


(b) Coppia.

Figura 9.20: Disco, controllo IBVS, risposta allo scalino

### 3.2 Controllo PBVS

In un'applicazione come quella in esame, dove la stima della posa dell'oggetto non è particolarmente complessa, è possibile utilizzare un controllo in asservimento visivo basato sulla posizione (*Position-Based Visual Servoing*) per ottenere una traiettoria più lineare. Lo schema a blocchi del controllo è mostrato in figura 9.21. Nel caso specifico, la proprietà



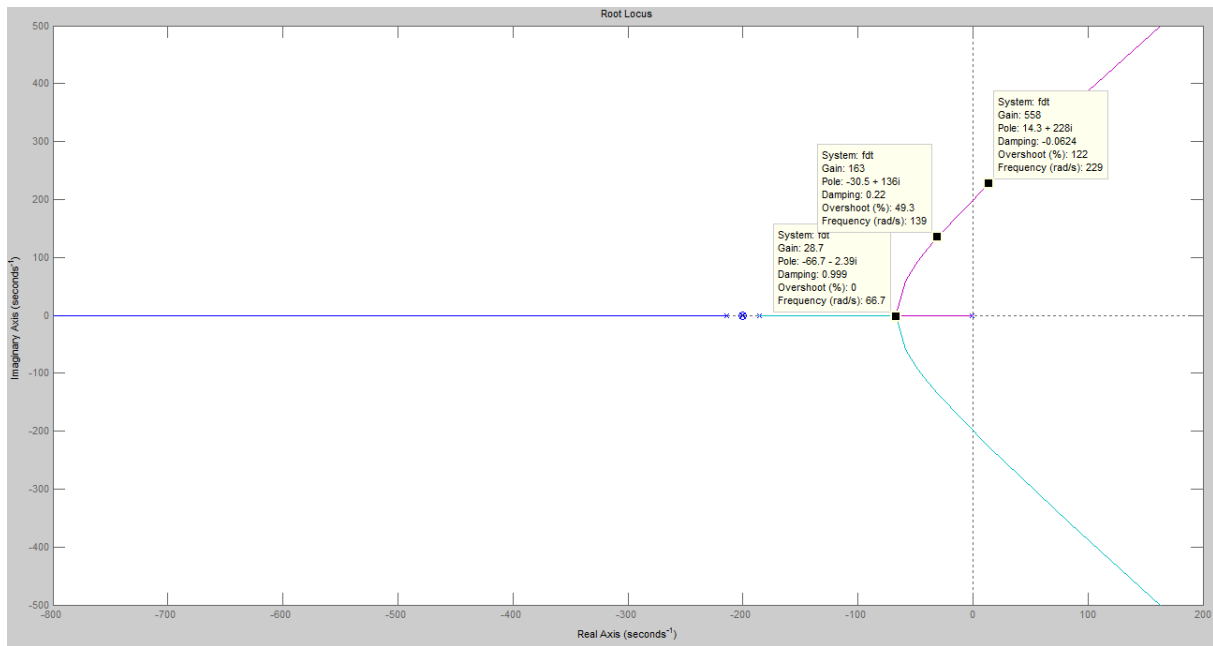
*Figura 9.21: Schema di controllo del disco PBVS*

che permette di stimare la posa dell'oggetto target e, di rimando, del disco, è l'orientamento del target rispetto al centro del disco. La relazione che permette di stimare l'orientamento a partire dalle coordinate in pixel del centro geometrico dell'oggetto è già stata mostrata nel paragrafo 2. In questo caso, la posa desiderata da fornire come riferimento al sistema di controllo è direttamente l'orientamento desiderato ( $\theta_d$ ) dell'oggetto target. Data la stima  $\theta_s$ , la legge di controllo con regolatore proporzionale è data dalla relazione (9.8).

$$\omega_z = K_p \cdot (\theta_d - \theta_s) . \quad (9.8)$$

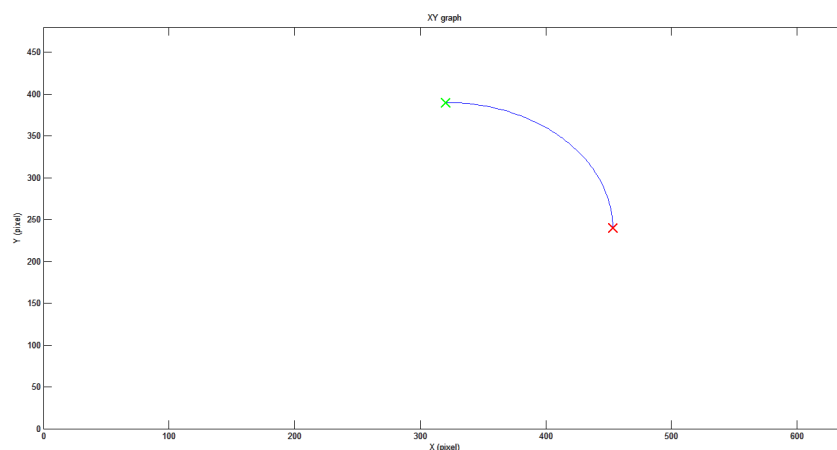
Come si vede dalla relazione appena scritta, nel caso di un sistema ad un solo grado di libertà, il controllo PBVS prevede la completa sostituzione di un sensore di posizione, come l'encoder del motore, col sensore di visione. Pertanto, il controllo PBVS sostituisce la chiusura dell'ultimo anello del controllo in cascata degli attuatori, ovvero quello di posizione.

Per la taratura del guadagno proporzionale  $K_p$  del regolatore, si è deciso di valutare il luogo delle radici del sistema completamente a tempo continuo, ovvero valutando la dinamica del motore, ma senza prendere in considerazione la dinamica del sistema di visione. In figura 9.22 viene mostrato il luogo delle radici di suddetto sistema. Come si vede, il sistema rimane stabile anche per valori elevati del guadagno. Come nel caso dei task IBVS trattati nel paragrafo 4 del capitolo 7, esiste un limite di guadagno per cui gli



**Figura 9.22:** Luogo delle radici del sistema a tempo continuo

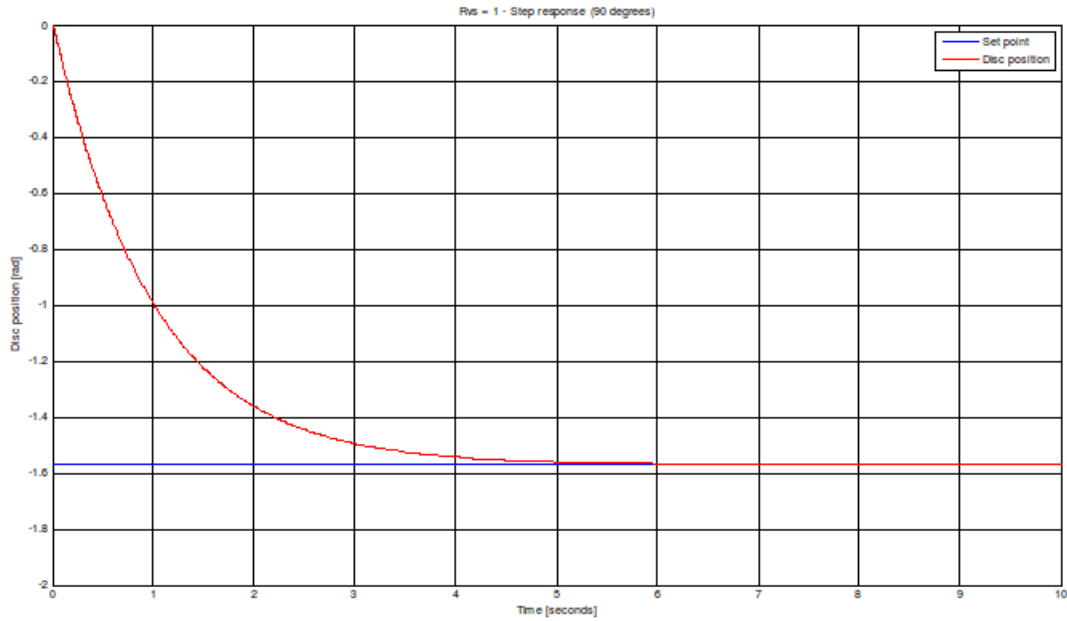
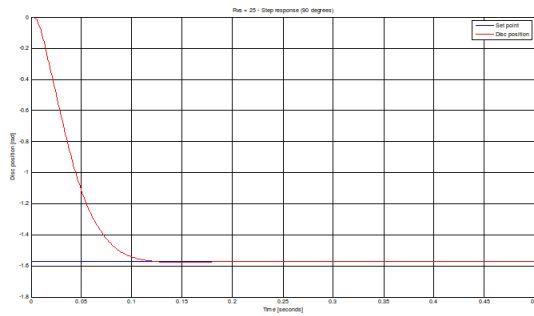
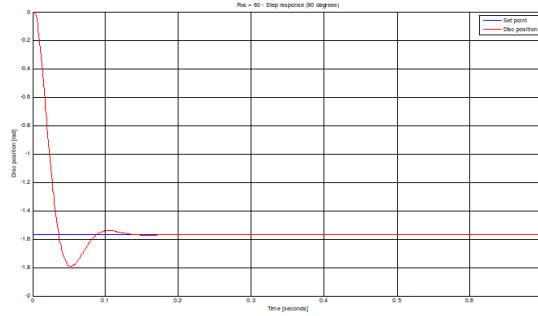
autovalori diventano complessi coniugati e la risposta allo scalino presenta sovraelongazioni. Dal punto di vista della scena osservata dalla telecamera, la risposta allo scalino si presenta come variazione della stima della posa dell'oggetto target. In figura 9.23 è mostrata la traiettoria seguita dall'oggetto target sul piano immagine, a partire dalla posizione indicata dalla croce verde per arrivare alla posizione data dalla croce rossa.



**Figura 9.23:** Posizionamento del disco visto sul piano immagine

In figura 9.24 viene mostrata la risposta ad uno scalino ideale di  $-180^\circ$  nel caso di (a) risposta stabile e lenta,  $K_p = 1$ , (b) risposta più veloce senza sovraelongazioni,  $K_p = 25$  e (c) risposta con sovraelongazioni una volta superato il limite,  $K_p = 60$ .

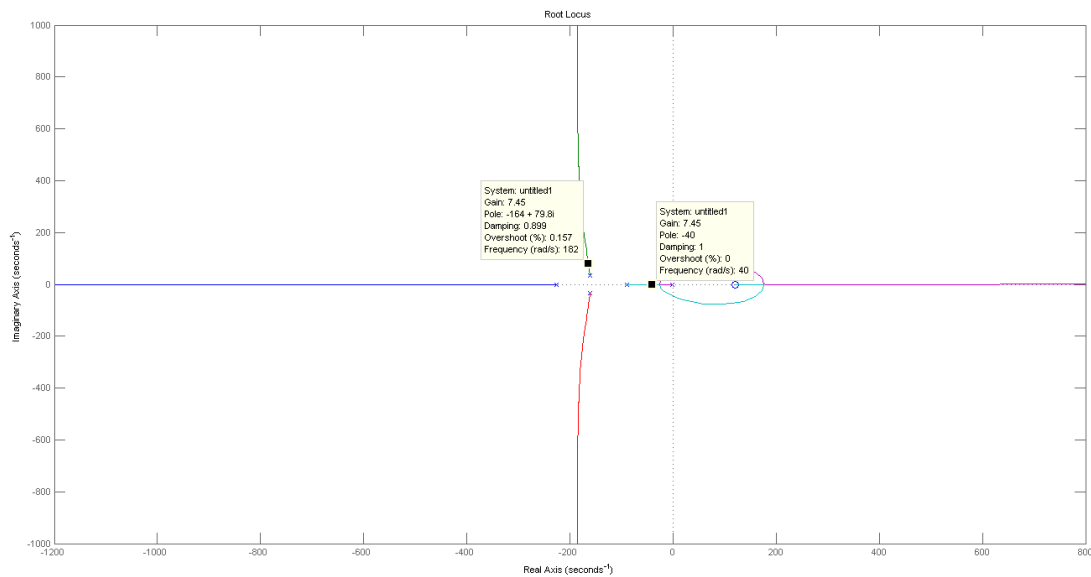
Introducendo nel sistema da controllare anche la dinamica dovuta al sistema di visione, il luogo delle radici subisce le modifiche già analizzate per la guida lineare nel paragrafo 4

(a)  $K_p = 1$ .(b)  $K_p = 25$ .(c)  $K_p = 60$ .**Figura 9.24:** Disco, controllo PBVS ideale, risposta allo scalino

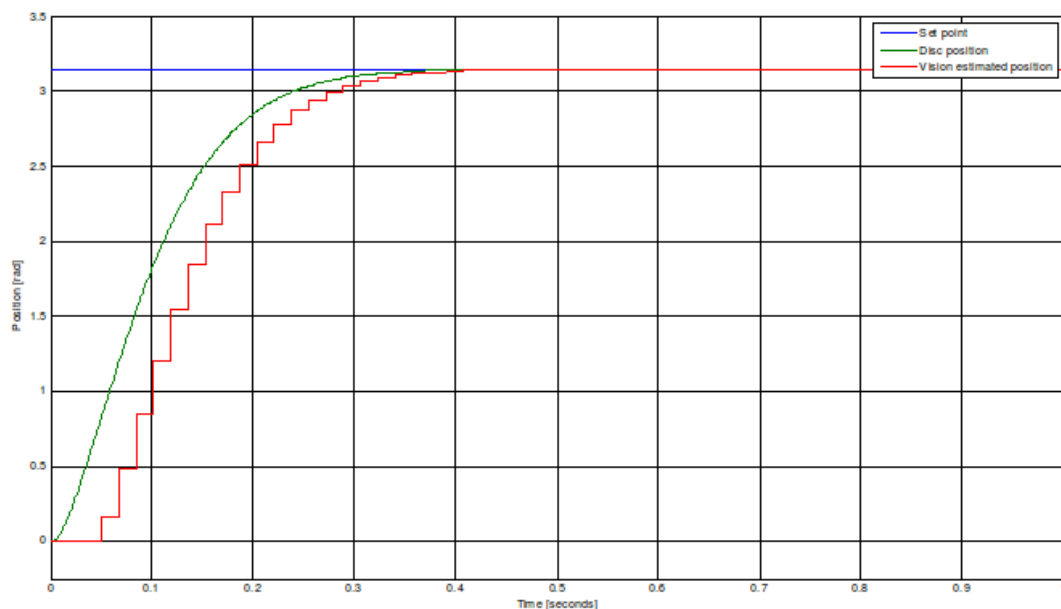
del capitolo 7. Il luogo delle radici ottenuto è mostrato in figura 9.25. L'introduzione del polo a dinamica lenta dovuto al sistema di visione porta ad una riduzione delle prestazioni dinamiche raggiungibili dal controllo. Infatti, per ottenere una traiettoria pulita senza sovraelongazioni il guadagno proporzionale del regolatore deve essere tarato, ad esempio, come  $K_p = 8$ . La risposta allo scalino del sistema così controllato è mostrata in figura 9.26. La curva blu è lo scalino, la curva verde rappresenta la traiettoria seguita dal disco, mentre la curva rossa è la stima della posizione del disco. Come si vede, la stima è campionata ed in ritardo rispetto alla posizione assunta dal disco.

Infine, l'ultima non linearità analizzata è la presenza di rumore statistico nell'individuazione del centroide del target. Come si vede in figura 9.27, il rumore influisce sulla stima della posizione dell'oggetto target, ma la chiusura dell'anello di controllo in retroazione garantisce il corretto completamento del task di posizionamento.





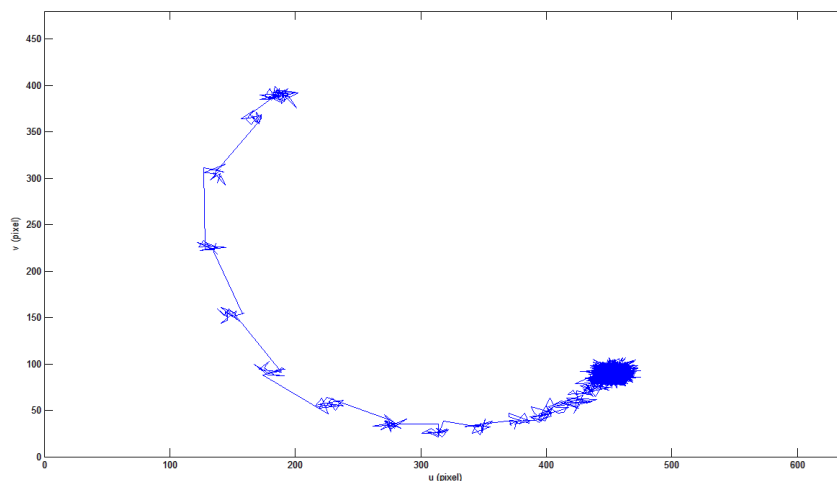
*Figura 9.25: Luogo delle radici del sistema a tempo discreto*



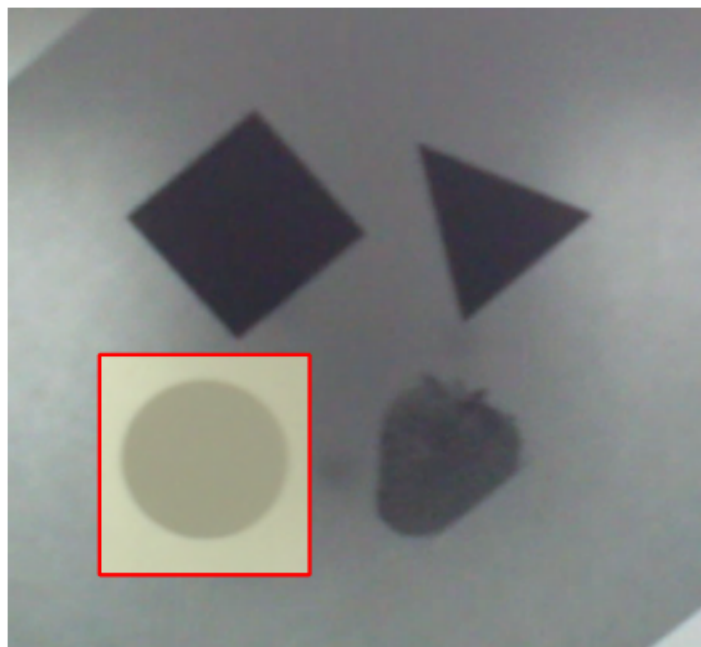
*Figura 9.26: Disco, controllo PBVS reale, risposta allo scalino*

## 4 Prove sperimentali

Prima di effettuare le prove sperimentali sul sistema, si è proceduto alla costruzione dei descrittori delle varie figure da sottoporre a tracciamento. La selezione del target da posizionare viene effettuata direttamente sull'immagine ripresa dalla telecamera, selezionando l'area dell'immagine occupata dalla figura target, come mostrato in figura 9.28, dove viene selezionato il cerchio. Invece, per quanto riguarda la scelta della posa desiderata, si è deciso di spostare l'oggetto a  $90^\circ$ , ovvero a ore 12 nell'immagine. Nell'esecuzione delle prove, bisogna far notare che i task di posizionamento sono stati portati a compimento senza una



**Figura 9.27:** Posizionamento del disco visto sul piano immagine con l'introduzione di rumore

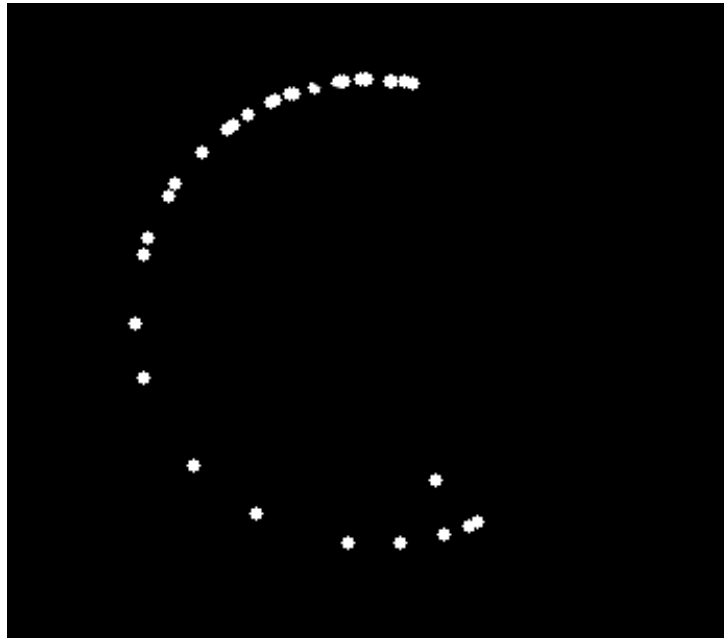


**Figura 9.28:** Selezione del target da tracciare per il controllo

taratura precisa dell'azionamento dell'attuatore. Questo non è stato dovuto a negligenza, ma all'intenzione di voler mettere in evidenza la robustezza dell'algoritmo di controllo anche in presenza di dispositivi scarsamente calibrati. Un'altra difficoltà presentatasi è relativa alla difficoltà nel tarare i parametri del motore praticamente in assenza di carico, dato che il disco ha un inerzia così ridotta da poter essere trascurata. A causa della mancanza di calibrazione del motore, si è osservata una variazione di  $\pm 5\%$  della velocità fornita come riferimento all'azionamento.

Inoltre, anche i parametri della telecamera non sono stati tarati mediante un'opportuna procedura di calibrazione, ma stimati a partire dai dati tecnici della telecamera stessa.

Le prime prove sperimentali hanno riguardato l'applicazione del controllo con individuazione dell'oggetto target tramite l'algoritmo di tracciamento con successivo finestramento dell'immagine. Ponendo  $K_p = 7$ , il compito di tracciamento viene portato a termine con successo, come mostrato in figura 9.29. Il framerate di elaborazione ottenuto è di circa

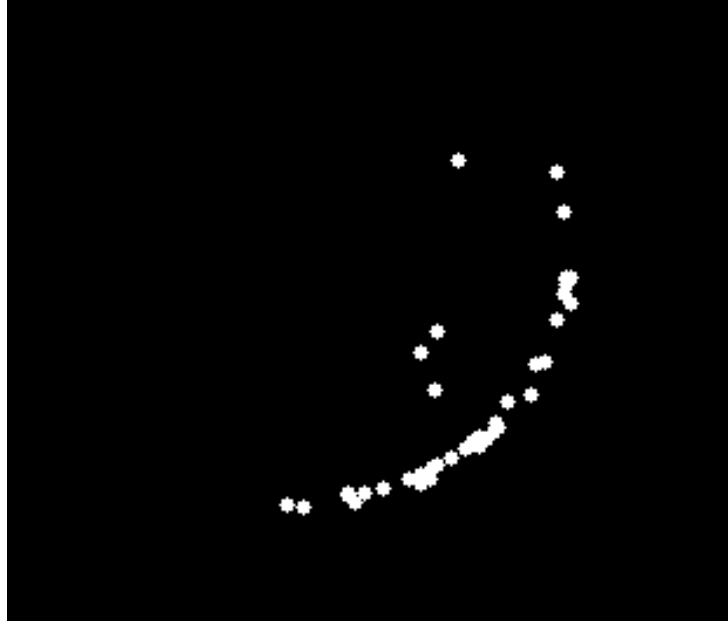


**Figura 9.29:** Posizionamento con tracking,  $K_p = 7$

40 *fps*. In figura i punti bianchi indicano la posizione del centro geometrico del target nei vari frame acquisiti dalla telecamera. La velocità di rotazione del disco è inversamente proporzionale alla densità dei punti nell'immagine: nel tratto iniziale (in basso) la velocità del disco è bassa, aumenta nel tratto intermedio (sulla sinistra) dove pochi punti vengono individuati e diminuisce nel tratto di posizionamento finale (in alto), dove il disco deve rallentare per permettere il posizionamento preciso del target.

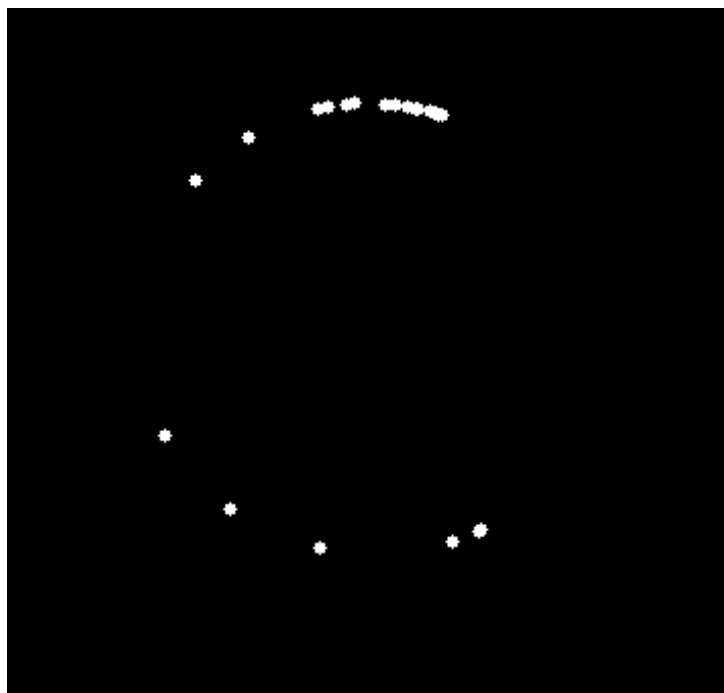
L'algoritmo di tracciamento può soffrire nella fase intermedia in cui la velocità di rotazione del disco è superiore e quindi l'individuazione del target è più sensibile alla scelta della dimensione della regione dell'immagine selezionata per la ricerca nell'immagine seguente. Infatti, in figura 9.30 viene mostrato il risultato del tracciamento per controllo con  $K_p = 10$ . Questa volta la scena ruota in senso antiorario, ma il tracciamento viene perso a metà della traiettoria, nel momento in cui il disco ha velocità maggiore. La densità dei punti nel tratto iniziale indica il fatto che il disco ruota così velocemente che il target viene recuperato ad ogni giro, ma non in maniera sufficiente a portare a termine il task di posizionamento.

Osservato tale comportamento, si è deciso di rinunciare al finestramento per provare a



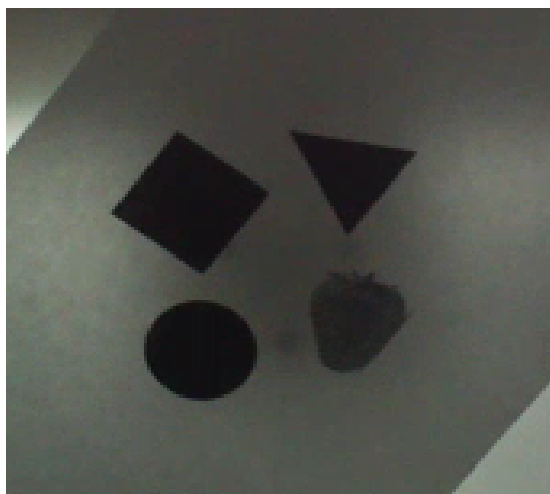
**Figura 9.30:** Posizionamento con tracking,  $K_p = 10$

raggiungere prestazioni dinamiche maggiori. Questo porta ad una riduzione del framerate di elaborazione raggiungibile, ora pari a 30 *fps* e, quindi ad una riduzione delle massime prestazioni raggiungibili, il cui collo di bottiglia è ora rappresentato dalla dinamica del sistema di visione e non da lacune nell'algoritmo di tracciamento. In figura 9.31 sono mostrati i risultati ottenuti per valori del guadagno pari a  $K_p = 15$  e  $K_p = 20$ . Si nota che il disco viene ora posizionato rispettivamente in 30 e 20 frame, che corrispondono a circa 1 e 0.67 secondi. Provando ad aumentare nuovamente il guadagno ( $K_p = 30$ ), il sistema diventa instabile, sia a causa del raggiungimento dei limiti imposti dalla dinamica del sistema di visione, sia per l'insorgere di problemi nel riconoscimento degli oggetti a causa di fenomeni di blurring.

(a)  $K_p = 15$ .(b)  $K_p = 20$ .*Figura 9.31: Posizionamento senza finestramento*

In figura 9.32 vengono riportati alcuni frame acquisiti da un'altra telecamera durante l'esecuzione del task di posizionamento. La sequenza fa riferimento al task eseguito con guadagno del controllore basso. Si può notare come in tutte le immagini riprese le figure non siano sottoposte a particolari deformazioni.

Per concludere, in figura 9.33 è mostrata la sequenza di un task di posizionamento eseguito con guadagno maggiore. L'esempio riportato, in particolare, fa riferimento ad una prova in cui si è verificata la perdita del tracciamento durante il task di posizionamento. Tale perdita del tracciamento è da imputare principalmente all'insorgere di fenomeni di sfumatura (blurring) dei contorni delle figura, evidenti a partire dal frame n. 2. L'algoritmo di controllo è robusto anche a queste situazioni, infatti verso la fine della sequenza l'oggetto viene nuovamente individuato e il motore effettua l'inversione del senso di rotazione necessaria a completare il posizionamento. Il tracciamento viene recuperato nel frame n. 6.



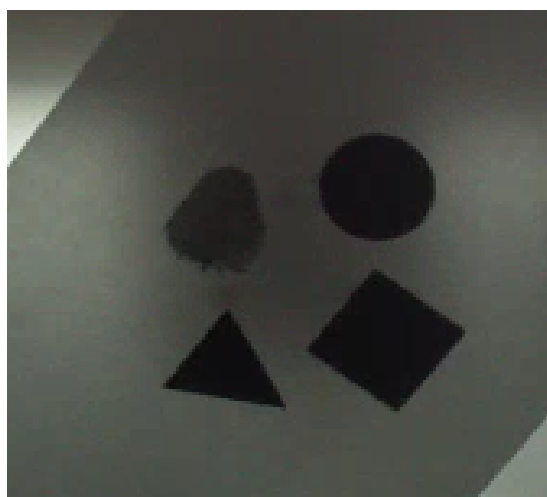
(a) *Condizione iniziale.*



(b) *Frame 1.*



(c) *Frame 2.*



(d) *Frame 3.*



(e) *Frame 4.*



(f) *Task completato.*

**Figura 9.32:** Sequenza di frame del task di posizionamento



(a) *Condizione iniziale.*



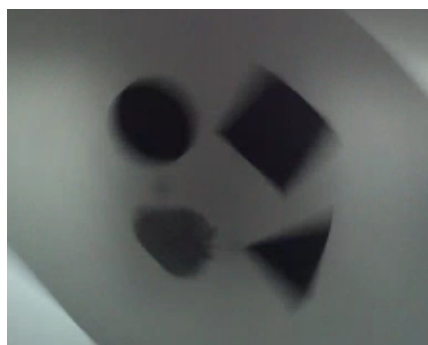
(b) *Frame 1.*



(c) *Frame 2.*



(d) *Frame 3.*



(e) *Frame 4.*



(f) *Frame 5.*



(g) *Frame 6.*



(h) *Posizionamento completato.*

**Figura 9.33:** Sequenza di frame del task di posizionamento con perdita temporanea del tracciamento



## 5 Conclusioni

In questo capitolo si è mostrato come la ricerca e l'analisi delle tematiche del visual servoing abbiano trovato la loro applicazione nella realizzazione di un semplice, ma al contempo completo, sistema di controllo in asservimento visivo.

Dal punto di vista della progettazione dell'architettura software del controllo si è visto come la scelta di una soluzione pc-based abbia permesso di implementare tutte le componenti software (interfaccia utente, elaborazione immagini e controllo) su un solo dispositivo. Questa configurazione non è la più adatta alla realizzazione di un sistema distribuito poiché non permette l'utilizzo di hardware dedicato al controllo (PLC) o all'elaborazione immagini. Per di più, l'esecuzione in parallelo su uno stesso calcolatore del task in tempo reale dedicato al controllo macchina e del processo di acquisizione ed elaborazione immagini porta ad un notevole decremento delle prestazioni raggiunte da entrambi i processi. D'altro canto questa è la configurazione che permette di minimizzare i tempi di comunicazione tra i vari processi.

In secondo luogo, per quanto concerne l'acquisizione e l'elaborazione immagini, le considerazioni, gli algoritmi e gli oggetti software presentati nel capitolo 8 hanno trovato realizzazione ed utilizzo, dimostrando la validità di quanto realizzato. Come si è visto, infatti, l'algoritmo di tracciamento implementato fornisce ottimi risultati per il tracciamento di oggetti o e figura bidimensionali. Le prestazioni, in termini di onerosità computazionale, sono in linea con quanto ci si può attendere dall'hardware utilizzato.

La sintesi del controllo ha dimostrato come le considerazioni fatte nel capitolo 7 possono essere estese anche a sistemi e configurazioni di controllo in asservimento visivo differenti. In particolar modo, il sistema da sottoporre a controllo trattato in questo capitolo non presenta limiti di movimento o di prestazione degli attuatori e questo ha permesso di poter sperimentare in libertà il controllo in asservimento visivo.

Lo sviluppo delle componenti di elaborazione immagini e controllo della meccanica del sistema è confluito successivamente nel controllo del sistema reale. Questo ha permesso di mettere a punto il sistema operativo real-time in tutte le sue componenti ottimizzando la generazione dei segnali di controllo per gli azionamenti del motore utilizzato. L'integrazione delle componenti ha permesso la buona riuscita del controllo del sistema, per il quale si sono ottenute prestazioni del tutto simili a quelle previste in fase di simulazione.

Per gli scopi del lavoro di tesi, questo banco di prova è stato sviluppato con il principale obiettivo di costruire una piattaforma per la sperimentazione e la validazione degli algoritmi di controllo in asservimento visivo. Tuttavia, le componenti implementate sono facilmente riutilizzabili per l'esportazione dell'applicazione in ambito industriale. Un esempio di una possibile applicazione di questo tipo è mostrato in figura 9.34, in cui viene mostrata una tavola rotante. Il controllo in asservimento visivo potrebbe venire utilizzato per il corretto posizionamento della tavola al fine di completare un qualche tipo di lavorazione meccanica.



**Figura 9.34:** *Tavola rotante industriale*

Infine, occorre sottolineare ancora una volta che il risultato complessivo ottenuto è il controllo di un sistema ad un solo grado di libertà, ma i risultati parziali ottenuti a livello di acquisizione ed elaborazione immagini, modellazione, simulazione e sviluppo del controllo real-time forniscono le basi per l'estensione dell'approccio a sistemi più complessi a più gradi di libertà.

# Capitolo 10

## Controllo di un Pendolo Inverso con Asservimento Visivo

### Indice

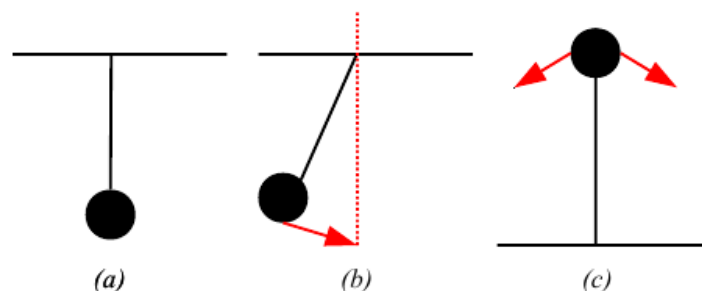
---

1	Descrizione del banco . . . . .	<b>448</b>
1.1	Telecamera . . . . .	449
1.2	Componenti meccaniche . . . . .	450
1.3	Motore e Azionamento . . . . .	453
2	Controllo classico . . . . .	<b>454</b>
2.1	Modellazione . . . . .	456
2.2	Controllo . . . . .	461
2.3	Simulazione . . . . .	464
2.4	Prove sperimentali . . . . .	465
3	Controllo in asservimento visivo . . . . .	<b>466</b>
3.1	Sottocampionamento . . . . .	467
3.2	Controllo Vision-In-The-Loop . . . . .	471
3.3	Risultati sperimentali . . . . .	473
4	Elaborazione delle immagini . . . . .	<b>477</b>
4.1	Segmentazione . . . . .	479
4.2	Estrazione delle feature . . . . .	483
4.3	Problema di tracciamento . . . . .	484
4.4	Prova sperimentale . . . . .	487
5	Risultati sperimentali . . . . .	<b>487</b>
6	Conclusioni . . . . .	<b>494</b>

---

La seconda applicazione sviluppata si discosta leggermente dai classici problemi affrontati tramite le tecniche di controllo in asservimento visivo. In letteratura il termine *visual servoing* è sempre associato a sistemi di controllo di robot, in particolare manipolatori, con lo scopo di posizionare l'end-effector in un determinato modo al fine di compiere il task di manipolazione prefissato. In questo lavoro di tesi, tuttavia, si è inteso il termine nel senso più generale di *controllo vision-in-the-loop*. In questo modo, l'analisi effettuata in questa trattazione è estendibile a qualsiasi sistema che preveda l'utilizzo dei sistemi di visione per la chiusura dell'anello di controllo più esterno.

In generale, uno dei banchi sperimentali più utilizzati per studiare la progettazione, sintesi e implementazioni di controllori a sistemi instabili è costituito dal cosiddetto *pendolo inverso*. La descrizione del problema è molto semplice: un pendolo viene incernierato su un carrello libero di muoversi su una guida lineare nella direzione delle oscillazioni del pendolo. Il pendolo è un semplice sistema ad un grado di libertà rotazionale utilizzato molto spesso per approcciarsi a concetti propri della disciplina dei controlli automatici, quali stabilità, asintotica stabilità e instabilità. Infatti, si possono individuare tre configurazioni del pendolo: (i) nella posizione di riposo il pendolo è in equilibrio stabile, poiché da essa non si muove a meno di non applicare una forza (figura 10.1.a); (ii) in tutti gli altri punti dello spazio la configurazione è instabile, perché il pendolo tende a tornare nel suo punto di equilibrio (figura 10.1.b); (iii) un altro punto di equilibrio è la posizione opposta al punto di riposo, ovvero con la parte terminale del pendolo rivolta verso l'alto (figura 10.1.c). Questa posizione è però un punto di equilibrio instabile, poiché l'applicazione di una forza mette in movimento il pendolo che tende a tornare nella sua posizione di riposo.

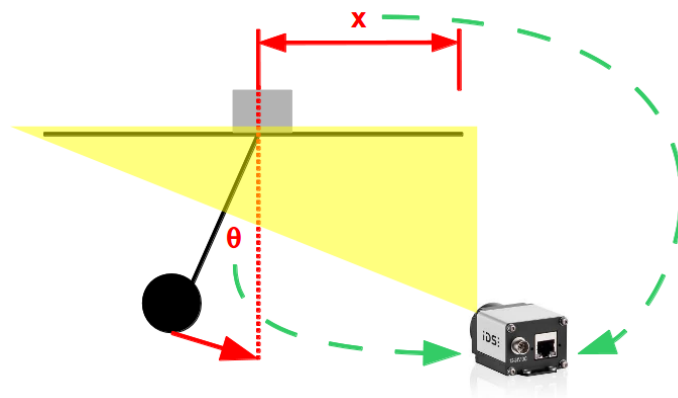


**Figura 10.1:** Punti d'equilibrio del pendolo

L'obiettivo del classico esercizio del pendolo inverso consiste nel mantenere il pendolo

stesso nella sua posizione di equilibrio instabile, tramite la semplice applicazione di una forza sul carrello. Di conseguenza, il controllo riesce a stabilizzare un sistema di per sé instabile. Inoltre, la modellazione del pendolo inverso porta alla scrittura di relazioni non lineari, le quali, secondo uno degli approcci di controllo possibili, devono essere linearizzate per permettere la sintesi del controllo stesso. Sono queste due caratteristiche, ovvero l'instabilità e la non linearità del sistema da controllare a rendere il pendolo inverso l'esercizio tipico per la sperimentazione e validazione di controlli particolari.

Nello scopo di questo lavoro di tesi, si è cercato di portare l'esercizio del pendolo inverso ad un livello di difficoltà superiore. Infatti, come si vedrà successivamente, le informazioni necessarie al regolatore per il completamento del suo task sono la posizione del carrello sulla guida e la posizione angolare dell'asta del pendolo. Per quanto riguarda il banco di prova descritto nel capitolo 9, si è già visto come nel posizionamento della tavola rotante il sistema di visione venga utilizzato in completa sostituzione dell'encoder del motore per la chiusura dell'anello di controllo in posizione dell'attuatore. Di conseguenza, si è deciso di provare a ripetere l'esperienza col controllo del pendolo inverso, andando a sostituire i sensori per la lettura della posizione di carrello e asta con una singola telecamera, come schematizzato in figura 10.2.



**Figura 10.2:** Rappresentazione schematica del sistema

Come è stato ripetuto più volte nel corso della trattazione, questa semplice sostituzione non è esente da problematiche da affrontare, individuabili come al solito tra i due ambiti del controllo *vision-in-the-loop*: l'elaborazione immagini e il controllo multirate. Dal punto di vista dell'elaborazione delle immagini, questo è un ulteriore problema di tracciamento del moto di un oggetto all'interno di una scena. Invece, per quanto concerne il controllo vero e proprio, si verifica l'insorgenza delle ben note tematiche del controllo multirate, in

questo caso ancora più critiche rispetto al controllo della piattaforma rotante, poiché il pendolo inverso è un sistema con una dinamica maggiore.

In questo capitolo si presentano le prove sperimentali e le considerazioni sul *controllo visivo del pendolo inverso*. Innanzitutto, vengono presentate le componenti del banco sperimentale (1), alcune delle quali sono già state descritte nell'applicazione del posizionamento della tavola rotante. In secondo luogo, si comincia ad affrontare il problema dal punto di vista del controllo, presentando prima lo schema di controllo classico con lettura dei segnali degli encoder (2), per poi analizzare la possibilità di sostituire o estendere tali sensori con l'utilizzo di un sistema di visione (3). In seguito, al fine di implementare tale controllo visivo, si presenta la tecnica di tracciamento implementata per l'inseguimento del pendolo (4) e si mostrano i risultati sperimentali ottenuti (5). Infine, in chiusura del capitolo, si traggono le conclusioni sull'esperienza effettuata.

## 1 Descrizione del banco

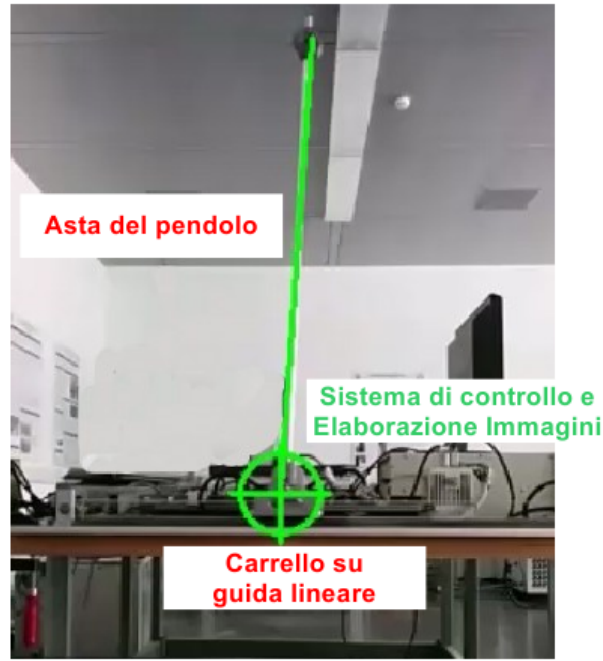
Il banco sperimentale del pendolo inverso può essere suddiviso in diversi componenti:

- *Parte meccanica*, composta essenzialmente da guida lineare, pendolo, attuatore elettrico e relativi sensori;
- *Quadro elettrico*, costituito dall'azionamento del motore e dalle componenti necessarie alla gestione dei segnali elettrici;
- *Schede DAQ* per l'acquisizione dati;
- *Telecamera* per l'acquisizione delle immagini del pendolo;
- *Calcolatore* (PC) per la gestione del banco.

Per quanto riguarda la descrizione dell'architettura hardware del controllo, del PC utilizzato e delle schede d'acquisizione dati, si rimanda il lettore al paragrafo 1.4 del capitolo 9 poiché le componenti utilizzate sono le medesime utilizzate per l'applicazione del posizionamento del disco, ovvero:

- PC industriale con *Intel Pentium 4* a 3.0 GHz e 1 GB di RAM;
- Scheda d'acquisizione dati *National Instruments PCI-6229*;
- Scheda d'espansione firewire *Delock 89167*.

In figura 10.3 è mostrata l'inquadratura del sistema da parte della telecamera utilizzata per la stabilizzazione del pendolo e permette di individuare alcune componenti del sistema.



*Figura 10.3: Pendolo inverso inquadrato dalla telecamera di controllo*

## 1.1 Telecamera

Anche nel caso della telecamera si è scelto di utilizzare lo stesso dispositivo già descritto nel capitolo 9, paragrafo 1.2 per quanto riguarda l'applicazione della tavola rotante, ovvero la telecamera *Sony XCD V60-CR*. Le motivazioni principali sono le stesse già viste in precedenza e relative al giusto compromesso tra qualità dell'immagine e framerate massimo raggiungibile.

A differenza del caso del disco, tuttavia, questa applicazione si discosta notevolmente dal classico utilizzo delle telecamere in ambito industriale. Infatti, l'area da inquadrare è all'incirca un quadrato di  $1.5\text{ m}$  di lato, quindi molto vasta. Per tale motivo, la scelta dell'obiettivo da montare si è rivelata fondamentale per stabilire la distanza di lavoro della telecamera stessa. Dovendo inquadrare una scena così ampia, si è optato per lo stesso obiettivo utilizzato per il disco, cioè il *Tamron* a focale  $4.8\text{ mm}$ . Anche in questo caso l'obiettivo è stato scelto con l'intenzione di minimizzare la distanza di lavoro. In particolare, data la dimensione del sensore della telecamera ( $w_s \times h_s = 4.736 \times 3.552\text{ mm}$ ) e la distanza focale ( $f = 4.8\text{ mm}$ ), è possibile stimare la distanza necessaria per ottenere il field of view desiderato:

$$\begin{cases} D_w = \frac{FOV_w \cdot f}{w_s} = \frac{1500 \cdot 4.8}{4.736} \text{ mm} = 1520 \text{ mm} \\ D_h = \frac{FOV_h \cdot f}{h_s} = \frac{1500 \cdot 4.8}{3.552} \text{ mm} = 2027 \text{ mm} \end{cases}$$

da cui risulta una distanza operativa di 2 m. Tale distanza è stata ridotta rinunciando ad inquadrare la parte inferiore del pendolo, cioè quando esso si trova nella sua posizione di riposo. Infatti, inquadrando solo la parte superiore del sistema, a partire dal carrello, si è ottenuta la distanza operativa finale:

$$D = 1500 \text{ mm} . \quad (10.1)$$

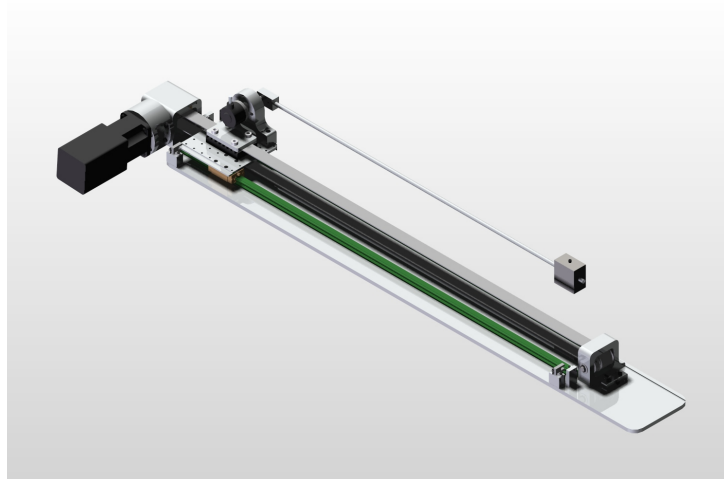
Come si vedrà in seguito nel paragrafo dedicato all'elaborazione delle immagini per l'estrazione del pendolo, l'ampiezza dell'area di lavoro porta alla complicazione dell'algoritmo di tracciamento a causa di una maggiore variabilità della scena e una grande probabilità di presenza di componenti di rumore nell'immagine.

## 1.2 Componenti meccaniche

Per quanto riguarda la costruzione meccanica del banco del pendolo, essa è suddivisa in diversi elementi costituenti:

- Pendolo;
- Carrello;
- Guide lineari;
- Cinghia di trasmissione;
- Puleggia folle e puleggia motrice.

In figura 10.4 è mostrata una vista del progetto meccanico completo del banco sperimentale del pendolo inverso.

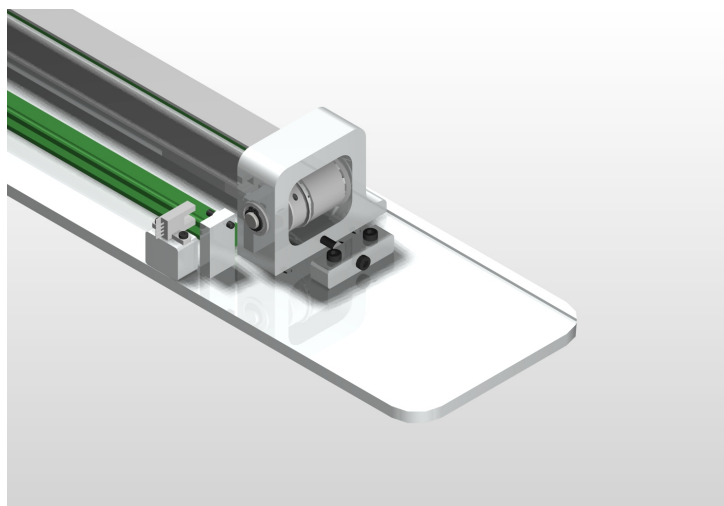


*Figura 10.4: Vista del banco pendolo inverso completo*



L'asta del pendolo è costituita da un tubo cilindrico d'alluminio su cui è fissata una massa, anch'essa metallica e di forma discoidale, che può essere posizionata lungo l'asta ad altezza diverse (ciò permette di modificare il momento d'inerzia al fine, per esempio, di verificare la robustezza di un controllore al variare del parametro meccanico). L'estremità inferiore dell'asta è collegata ad un albero a cui è calettato il *sensore di posizione angolare* dell'asta. Questo è realizzato attraverso un encoder a potenziometro: la rotazione dell'asta provoca una variazione della resistenza del sensore che un opportuno circuito di condizionamento converte in un segnale elettrico proporzionale allo spostamento angolare. Tale circuito prevede l'utilizzo del *line receiver*, come già spiegato nell'applicazione del disco.

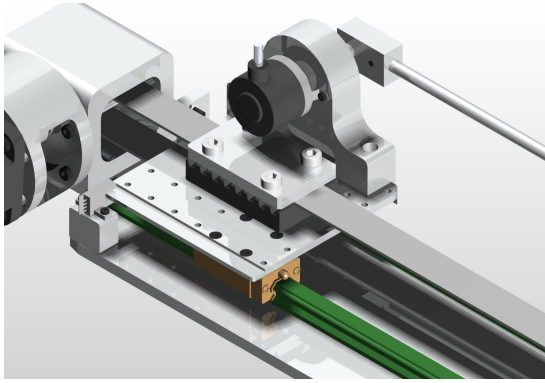
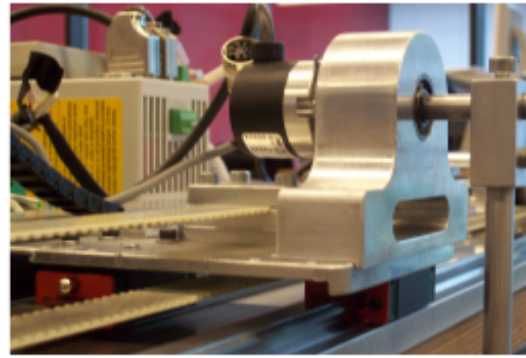
Le guide sono costituite da due barre parallele d'acciaio di sezione cilindrica. La lunghezza delle guide è di 1000 *mm* e permettono una corsa massima di 800 *mm*. Infatti, vicino alle loro estremità sono collocati due microinterruttori di fine corsa: la chiusura di uno dei due segnala che il carrello ha raggiunto la massima escursione ammissibile e disattiva l'azionamento del motore. Per maggiore sicurezza, oltre a questi fine corsa sono presenti altri due fine corsa meccanici, che servono a fermare il carrello in caso di guasto del microinterruttore. In figura 10.5 è mostrato il particolare di progetto relativo alla puleggia folle e ai finecorsa meccanico ed elettrico.



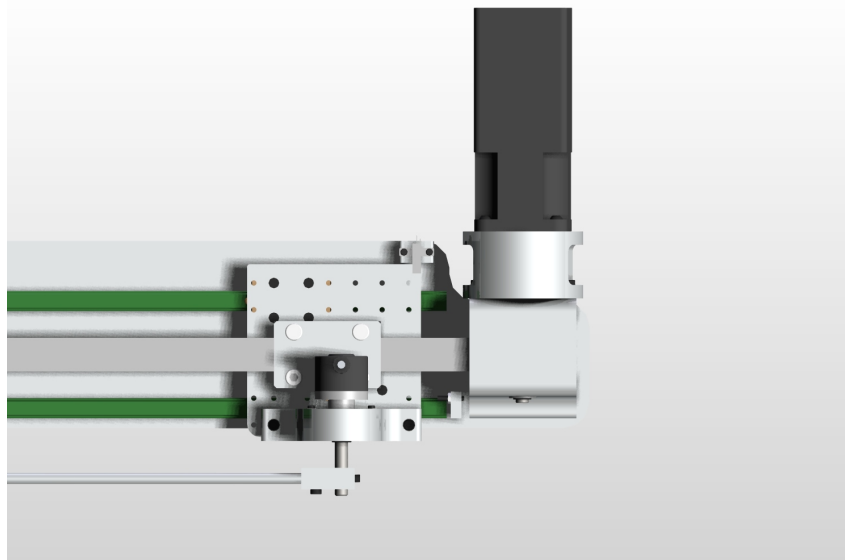
**Figura 10.5:** Banco pendolo inverso, particolare dei finecorsa

Il carrello è costituito da un corpo d'alluminio che può scorrere sulle guide in acciaio. Una vista del progetto del carrello e la sua realizzazione sono mostrate in figura 10.6.

Sulla sommità del carrello è sistemata la sede dell'albero a cui il pendolo è collegato. Sulla parte inferiore del carrello è fissata la cinghia di trasmissione che trasferisce il moto dal motore al carrello stesso. La posizione del carrello è misurata a partire dalla lettura del

(a) *Vista del progetto.*(b) *Carrello realizzato.***Figura 10.6:** *Banco pendolo inverso, carrello*

*resolver* del motore, descritto in seguito. L'albero motore è innestato, tramite apposito giunto, sulla puleggia motrice, che permette la conversione del moto rotatorio del motore nel moto lineare del carrello. Il particolare del blocco motore e relativo innesto alla puleggia motrice sono mostrati in figura 10.7.

**Figura 10.7:** *Banco pendolo inverso, particolare del blocco motore*

Questo tipo di trasmissione porta all'insorgenza di tre effetti che possono influire sul controllo del sistema: l'elasticità della cinghia, il gioco tra gli ingranaggi e l'attrito di primo distacco. Tutti e tre gli effetti introducono delle non-linearità che non vengono generalmente trattate nella modellazione del pendolo inverso, poiché il sistema viene considerato rigido e senza attrito. Questo significa che, nel caso ideale, la coppia del motore viene passata tutta al carrello senza dispersioni e senza oscillazioni. Nel caso reale, invece, è necessario un contributo di coppia aggiuntivo per vincere l'attrito, soprattutto

durante le inversioni del senso di marcia e la possibile presenza di oscillazioni può rendere il controllo meno stabile che in simulazione.

Per completezza, in tabella 10.1 sono elencate le dimensioni delle componenti del banco.

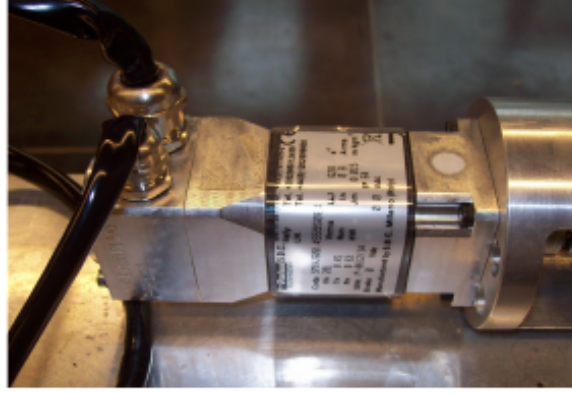
**Tabella 10.1:** *Banco pendolo inverso, dimensioni*

Caratteristica	Simbolo	Valore	Unità di misura
Massa Slitta	$m_s$	1.304	$kg$
Massa Snodo	$m_{sn}$	0.035	$kg$
Momento d'inerzia Snodo	$J_{sn}$	$11.62 \cdot 10^{-6}$	$kg \ m^2$
Massa concentrata	$m_c$	0.365	$kg$
Momento d'inerzia concentrato nel baricentro del carico	$J_c$	$116 \cdot 10^{-6}$	$kg \ m^2$
Massa Asta	$m_a$	0.17	$kg$
Momento d'inerzia concentrato nel baricentro dell'asta	$J_a$	$9010.92 \cdot 10^{-6}$	$kg \ m^2$
Momento d'inerzia della puleggia folle	$J_f$	$9.048 \cdot 10^{-6}$	$kg \ m^2$
Massa d'inerzia della puleggia motrice con motore	$J_m$	$(12.134 + 15) \cdot 10^{-6}$	$kg \ m^2$
Diametro pulegge	$D_p$	$29.40 \cdot 10^{-3}$	$m$
Lunghezza Asta	$L_a$	$400 \cdot 10^{-3}$	$m$
Posizione Massa Concentrata lungo l'asta	$L_c$	$200 \cdot 10^{-3}$	$m$
Rapporto di trasmissione	$\tau$	$D_p/2$	$m$

### 1.3 Motore e Azionamento

Per lo spostamento del carrello si è utilizzato il motore brushless *Parker SMB 42-60-035* ([site11]), mostrato in figura 10.8.

La coppia nominale raggiungibile è di  $0.15 \ Nm$  e in tabella 10.2 vengono riportati tutti i dati tecnici del motore.



**Figura 10.8:** Motore Parker SMB 42-60

**Tabella 10.2:** Motore Parker SMB 42-60, dati tecnici

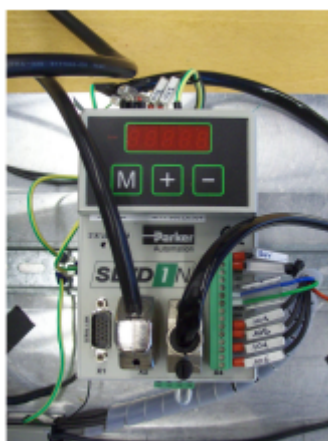
Coppia di stallo	0.35	$Nm$
Corrente di stallo	0.78	$A$
Coppia nominale	0.15	$Nm$
Velocità nominale	6000	$rpm$
Corrente nominale	0.38	$A$
Coppia massima	0.9	$Nm$
Inerzia senza freno	$13 \cdot 10^{-6}$	$kg \ m^2$
Costante elettrica	0.29	$Vs$
Costante di coppia	0.46	$Nm/A_{rms}$

La chiusura degli anelli di controllo in velocità e posizione è resa possibile dal resolver integrato, che, in questa applicazione viene utilizzato unicamente per leggere la posizione assunta dal carrello lungo la guida lineare. Infatti, per la stabilizzazione del pendolo il motore deve essere controllato in coppia. Questo tipo di controllo è reso possibile dall'accoppiamento del motore con l'azionamento *Parker SLVD1N* di figura 10.9.

Il funzionamento è analogo a quello dell'azionamento descritto nell'applicazione del disco e in figura 10.3 vengono riportati i dati tecnici.

## 2 Controllo classico

Prima di passare all'analisi della possibilità utilizzare un controllo in asservimento visivo per la stabilizzazione del pendolo inverso, è necessario analizzare e sviluppare il classico



**Figura 10.9:** Azionamento Parker SLVD1N

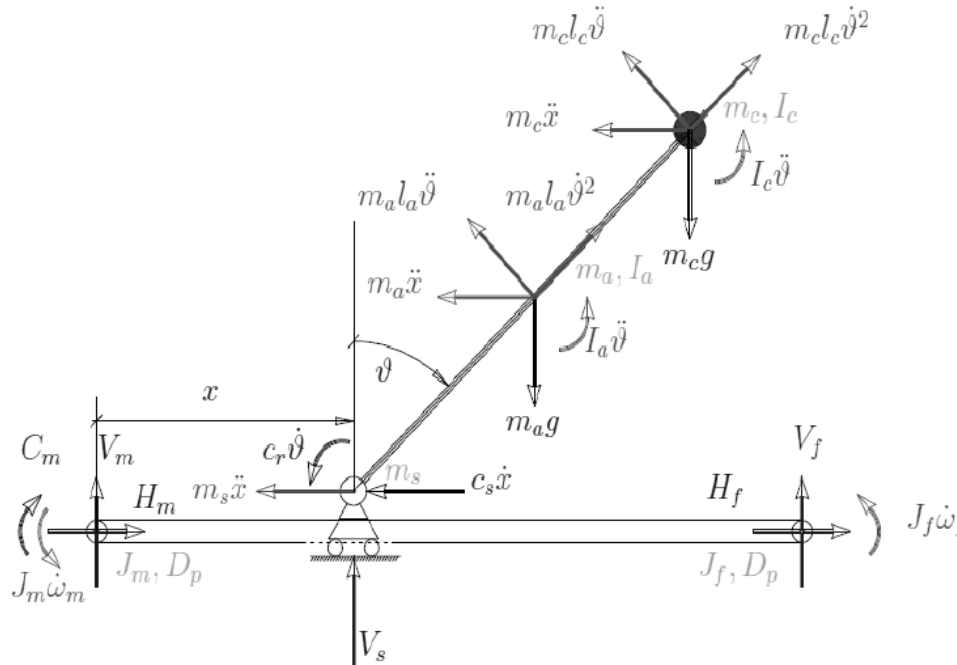
**Tabella 10.3:** Azionamento Parker SLVD1N, dati tecnici

Corrente di uscita nominale	1.25	$A_{rms}$
Corrente di uscita massima (picco)	2.5	$A_{rms}$
Potenza resa all'albero	0.345	$kW$
Frequenza di commutazione	8	$kHz$
<b>Retroazione Motore</b>		
Uscita simulazione encoder	4...65536	$step/rev$
Frequenza massima	160	$kHz$
<b>Ingresso analogico differenziale</b>		
Tensione	$\pm 10$	$V$
Risoluzione	10	$bit$
Frequenza Massima	500	$Hz$

controllo con l'utilizzo di encoder per la lettura della posizione del carrello e inclinazione dell'asta. In questo paragrafo si mostra la modellazione del pendolo inverso per arrivare alla definizione delle equazioni di stato del sistema. In seguito viene presentato lo sviluppo del controllo, implementato direttamente a partire dal sistema di stato senza passare esplicitamente per la scrittura della funzione di trasferimento del sistema. Infine, vengono mostrati i risultati raggiunti in fase di simulazione e di valutazione sperimentale.

## 2.1 Modellazione

In figura 10.10 viene mostrato il modello dinamico utilizzato per la modellazione del sistema complessivo.



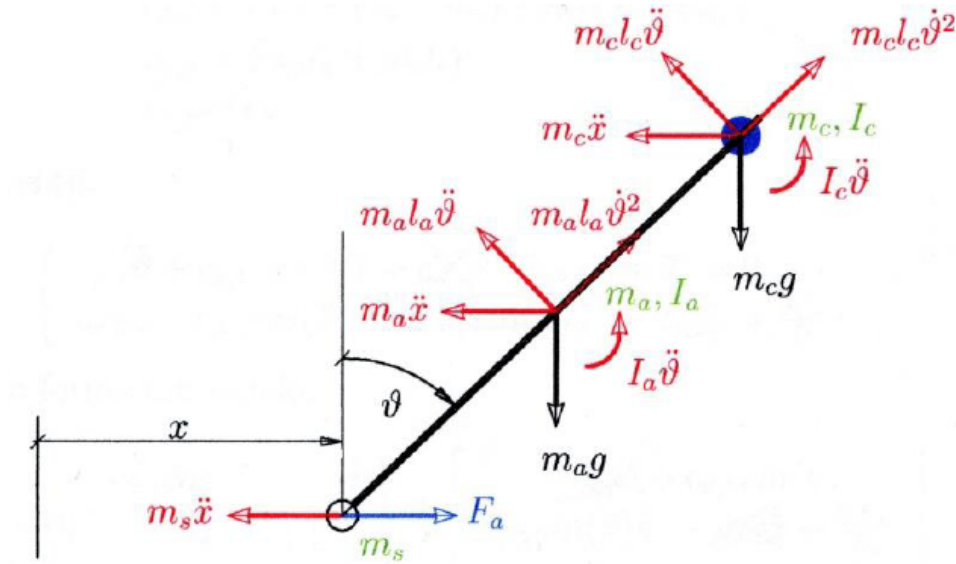
**Figura 10.10:** Modello e forze del banco per il controllo pendolo inverso

Il comportamento dinamico del sistema può essere descritto con un'equazione lineare del quarto ordine. Le variabili di stato del sistema sono la posizione e velocità del carrello e dell'asta del pendolo. L'unico ingresso è dato dalla forza orizzontale impressa al carrello tramite il motore. Per la scrittura delle equazioni di stato in forma finita, si considerano le seguenti assunzioni:

- Le masse del carrello, del carico posto all'estremità dell'asta e dell'asta stessa sono concentrate nei rispettivi baricentri;
- L'asta del pendolo è idealmente rigida;
- Gli attriti sono trascurabili;
- La cinghia di trasmissione è anch'essa idealmente rigida.

Il sistema di riferimento principale ha origine nel punto intermedio della guida lineare e le coordinate dell'asse  $X$  sono positive a destra del carrello. La posizione angolare dell'asta è nulla nella posizione di equilibrio instabile, ovvero col pendolo rivolto verso l'alto. La rotazione dell'asta è positiva muovendo il pendolo in senso orario, ovvero a destra del sistema di riferimento e negativa al contrario.

Per la scrittura delle equazioni della dinamica si è utilizzato il bilancio delle potenze del sistema. A questo fine, il modello completo di figura 10.10 è stato semplificato nel modello di figura 10.11 “spezzando” la cinghia all’altezza del carrello e considerando la forza  $F_a$  applicata al carrello nel punto di taglio.



**Figura 10.11:** Modello semplificato del pendolo inverso

Il bilancio di potenze relative alla rotazione dell’asta del pendolo è dato dall’equazione

$$\begin{aligned}
 & m_c l_c^2 \ddot{\theta} + m_c l_c \cos(\theta) \ddot{x} + J_c \ddot{\theta} \\
 & - m_c l_c g \sin(\theta) + m_a l_a^2 \ddot{\theta} + m_a l_a \cos(\theta) \ddot{x} \\
 & + J_a \ddot{\theta} - m_a l_a g \sin(\theta) = 0
 \end{aligned} \tag{10.2}$$

che può essere riscritta, con opportuno raccoglimento di termini, per evidenziare le variabili di stato del sistema:

$$\begin{aligned}
 & (m_c l_c^2 + J_c + m_a l_a^2 + J_a) \ddot{\theta} \\
 & + (m_c l_c + m_a l_a) \cos(\theta) \ddot{x} \\
 & - (m_c l_c + m_a l_a) \sin(\theta) g = 0 .
 \end{aligned} \tag{10.3}$$

In modo analogo, si ottiene il bilancio di potenze relativo alla traslazione del carro,

$$\begin{aligned}
 & m_c l_c \sin(\theta) \dot{\theta}^2 - m_c l_c \cos(\theta) \ddot{\theta} \\
 & - m_c \ddot{x} + m_a l_a \sin(\theta) \dot{\theta}^2 - m_a l_a \cos(\theta) \ddot{\theta} - m_a \ddot{x} \\
 & - m_s \ddot{x} + F_a = 0
 \end{aligned} \tag{10.4}$$

che, dopo opportuno raccoglimento, può essere riscritta come

$$\begin{aligned} & (m_c l_c + m_a l_a) \cos(\theta) \ddot{\theta} \\ & + (m_c + m_a + m_s) \ddot{x} \\ & = F_a + (m_c l_c + m_a l_a) \sin(\theta) \dot{\theta}^2 . \end{aligned} \quad (10.5)$$

Per ottenere una relazione in forma chiusa di  $F_a$  si può passare dalla scrittura del bilancio di potenze sul carro:

$$F_a \dot{x} + J_f \dot{\omega}_m \omega_m + J_m \dot{\omega}_m \omega_m = C_m \omega_m . \quad (10.6)$$

Dato che

$$\dot{x} = \omega_m \tau ,$$

sostituendo il valore di  $\omega_m$  nell'equazione (10.6), si ottiene

$$F_a \dot{x} + J_f \ddot{x} \dot{x} \frac{1}{\tau^2} + J_m \ddot{x} \dot{x} \frac{1}{\tau^2} = C_m \dot{x} \frac{1}{\tau} \quad (10.7)$$

che permette di esprimere il bilancio di potenze del carro in funzione della sua posizione  $x$  e velocità  $\dot{x}$ . Dalla relazione (10.7) si ottiene la relazione della forza applicata al carro:

$$F_a = \frac{C_m}{\tau} - (J_f + J_m) \ddot{x} \frac{1}{\tau^2} . \quad (10.8)$$

Sostituendo la relazione (10.8) nel bilancio di potenze del movimento traslazionale (10.5) si ottiene anche per questo bilancio un'equazione nelle sole variabili di stato:

$$\begin{aligned} & (m_c l_c + m_a l_a) \cos(\theta) \ddot{\theta} \\ & + (m_c + m_a + m_s + \frac{J_f + J_m}{\tau^2}) \ddot{x} \\ & = \frac{C_m}{\tau} + (m_c l_c + m_a l_a) \sin(\theta) \dot{\theta}^2 . \end{aligned} \quad (10.9)$$

L'equazione (10.9) insieme alla (10.3) formano il sistema di equazioni di stato del sistema da controllare:

$$\begin{cases} (m_c l_c^2 + J_c + m_a l_a^2 + J_a) \ddot{\theta} + (m_c l_c + m_a l_a) \cos(\theta) \ddot{x} - (m_c l_c + m_a l_a) \sin(\theta) g = 0 \\ (m_c l_c + m_a l_a) \cos(\theta) \ddot{\theta} + (m_c + m_a + m_s + \frac{J_f + J_m}{\tau^2}) \ddot{x} = \frac{C_m}{\tau} + (m_c l_c + m_a l_a) \sin(\theta) \dot{\theta}^2 \end{cases} . \quad (10.10)$$

Col sistema appena descritto si riesce a trovare una motivazione dei punti d'equilibrio del sistema. Ad accelerazioni nulle, il sistema è risolto per  $\theta = i \cdot \pi$ , ovvero per posizione angolari di  $0^\circ$  e  $180^\circ$ .



Questo è un sistema di equazioni non lineari in  $\theta$ , poiché esse presentano termini in seni e coseni. Per permettere lo sviluppo del controllo, il sistema viene linearizzato nell'intorno del punto d'equilibrio  $\theta = 0$ , per cui si ha  $\sin(\theta) = 0$  e  $\cos(\theta) = 1$ . Il sistema (10.10) può essere riscritto come:

$$\begin{cases} (m_c l_c^2 + J_c + m_a l_a^2 + J_a) \ddot{\theta} + (m_c l_c + m_a l_a) \ddot{x} = 0 \\ (m_c l_c + m_a l_a) \ddot{\theta} + (m_c + m_a + m_s + \frac{J_f + J_m}{\tau^2}) \ddot{x} = \frac{C_m}{\tau} \end{cases} \quad (10.11)$$

Per semplificare la scrittura delle equazioni, di seguito si utilizza la seguente dicitura:

- *Momento d'inerzia totale.*

$$J_i = m_c l_c^2 + J_c + m_a l_a^2 + J_a .$$

- *Momento d'inerzia del pendolo.*

$$J_p = m_c l_c + m_a l_a .$$

- *Massa totale ridotta al carrello.*

$$m_T = m_c + m_a + m_s + \frac{J_f + J_m}{\tau^2} .$$

Considerando il vettore degli spostamenti  $\mathbf{X}$  e le sue derivate

$$\mathbf{X} = \begin{bmatrix} x \\ \theta \end{bmatrix} , \quad \dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} , \quad \ddot{\mathbf{X}} = \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix}$$

e le notazioni appena descritte, il sistema (10.11) può essere riscritto in formulazione matriciale come segue:

$$\mathbf{M}\ddot{\mathbf{X}}(t) + \mathbf{C}_f\dot{\mathbf{X}}(t) + \mathbf{K}\mathbf{X}(t) = \mathbf{B}_f\mathbf{U}(t)$$

$$\begin{bmatrix} J_p & J_i \\ m_T & J_p \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{\tau} \end{bmatrix} C_m . \quad (10.12)$$

Dalla riscrittura dell'equazione del modello in forma matriciale risulta subito evidente come il modello non consideri l'attrito e componenti dovute alla cedevolezza della cinghia di trasmissione. Infine, si vuole ottenere la rappresentazione del sistema attraverso le classiche equazioni di stato

$$\begin{cases} \dot{z} = \mathbf{A}z + \mathbf{B}u \\ y = \mathbf{C}z + \mathbf{D}u \end{cases} .$$

Considerando il seguente vettore di stato e la sue derivata prima:

$$\mathbf{z} = \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix}, \quad \dot{\mathbf{z}} = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} \quad (10.13)$$

ed elaborando il sistema (10.12) si ottiene la rappresentazione di stato del modello scelto:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ | -\mathbf{M}^{-1}\mathbf{K} | & | \mathbf{M}^{-1}\mathbf{C}_f | \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ | \mathbf{M}^{-1}\mathbf{B}_f | \end{bmatrix} \\ \mathbf{C} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \quad (10.14)$$

Per permettere la scrittura delle matrici per la rappresentazione dello stato del sistema, si è posto  $y(t) = z(t)$ , ovvero l'uscita del sistema è lo stato del sistema stesso. Al fine dello sviluppo del controllo, è opportuno notare come il sistema possa essere ricondotto alla rappresentazione in forma canonica di raggiungibilità tramite una semplice inversione di variabili di stato. Infatti, ponendo

$$\mathbf{z} = \begin{bmatrix} x \\ \dot{x} \\ \dot{\theta} \\ \theta \end{bmatrix}, \quad \dot{\mathbf{z}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \ddot{\theta} \\ \dot{\theta} \end{bmatrix} \quad (10.15)$$

e operando uno spostamento di colonne della matrice  $\mathbf{A}$  (spostamento della seconda colonna in fondo alla matrice), si ottiene la forma canonica di raggiungibilità. La matrice  $\mathbf{B}$  non subisce modifiche perché l'unico termine non nullo si riferisce alla relazione ottenuta dall'ultima riga della matrice  $\mathbf{A}$ , mentre  $\mathbf{C}$  e  $\mathbf{D}$  permettono esclusivamente di mantenere valida la relazione  $y(t) = z(t)$ . Per completezza, si riporta la rappresentazione del sistema

in forma canonica di raggiungibilità:

$$\begin{aligned} \mathbf{A}_{\text{ragg}} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ | -\mathbf{M}^{-1}\mathbf{K}|_{c1} & |\mathbf{M}^{-1}\mathbf{C}_f|_{c1} & |\mathbf{M}^{-1}\mathbf{C}_f|_{c2} & | -\mathbf{M}^{-1}\mathbf{K}|_{c2} \end{bmatrix} \\ \mathbf{B}_{\text{ragg}} &= \begin{bmatrix} 0 \\ 0 \\ |\mathbf{M}^{-1}\mathbf{B}_f| \end{bmatrix}, \quad \mathbf{C}_{\text{ragg}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{D}_{\text{ragg}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \quad (10.16)$$

## 2.2 Controllo

Prima di passare alla sintesi del controllo è stata analizzata la dinamica del sistema non controllato. Come si vede in figura 10.12, i poli del sistema si trovano a  $0 \text{ Hz}$ ,  $-4.6 \text{ Hz}$  e  $4.6 \text{ Hz}$ . Di conseguenza, a causa del polo a parte reale positiva, il sistema è instabile e, in risposta ad uno scalino di coppia del motore il pendolo tende a cadere per raggiungere l'altra posizione d'equilibrio.

Nel paragrafo 2.1 si è visto come il modello del sistema meccanico possa essere rappresentato in forma canonica di raggiungibilità. Pertanto, è possibile utilizzare il metodo di *assegnamento degli autovalori* (vedi 3) per sintetizzare il controllo direttamente a partire dalla rappresentazione dello stato del sistema. In particolare, vengono posizionate due coppie di poli complessi coniugati. In un sistema del second'ordine, il polinomio caratteristico può essere descritto in termini di frequenza propria  $\omega_n$  e smorzamento  $\xi$ , per cui i relativi poli assumono la forma:

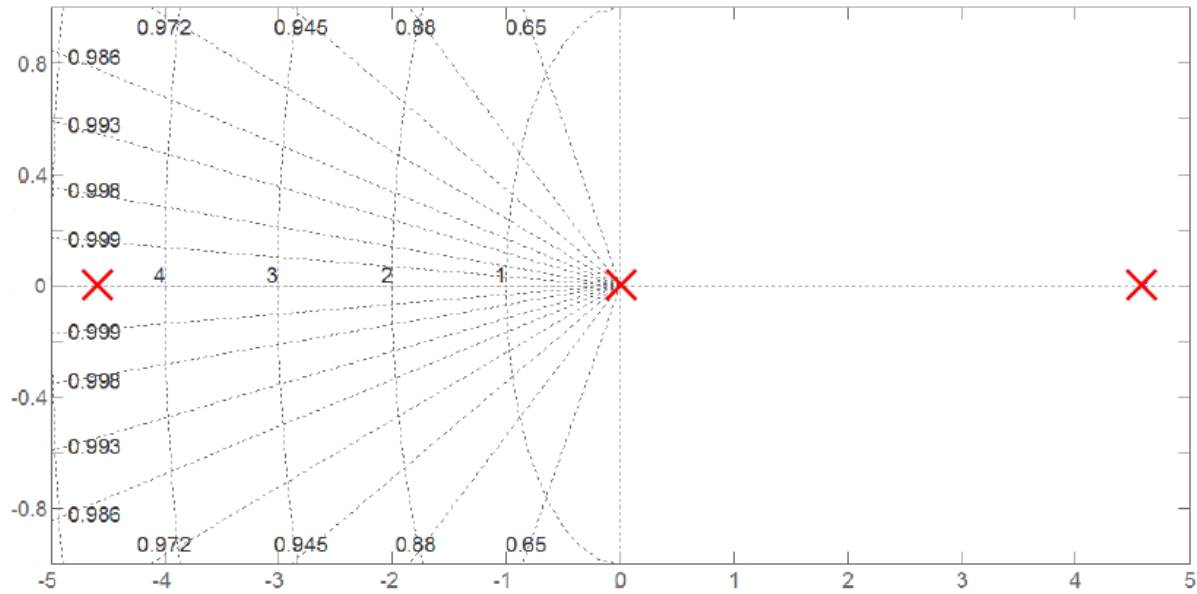
$$\begin{cases} p_1 = \omega_n \cdot (-\xi + j \cdot \sqrt{1 - \xi^2}) \\ p_2 = \omega_n \cdot (-\xi - j \cdot \sqrt{1 - \xi^2}) \end{cases}. \quad (10.17)$$

In figura 10.13 è mostrato il luogo delle radici ottenuto posizionando i poli dati dalle equazioni (10.17) con  $\omega_1 = 4$ ,  $\omega_2 = 5$  e  $\xi = 0.8$ .

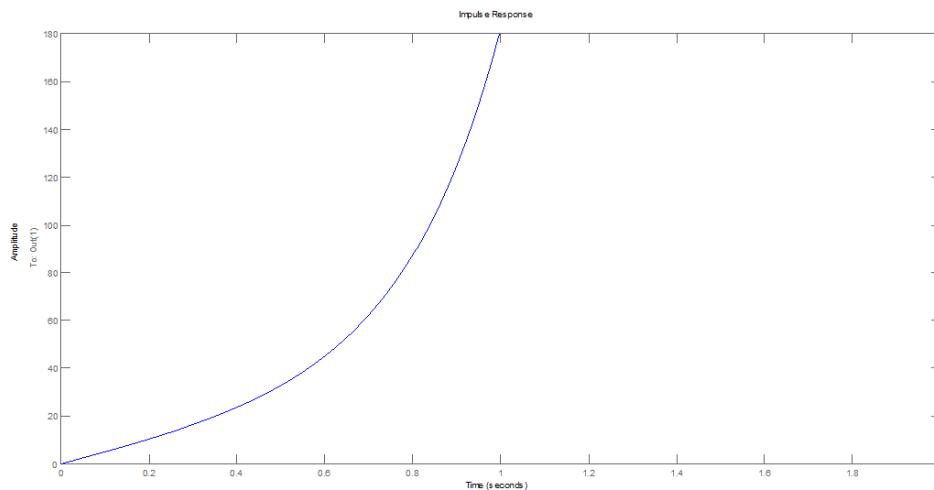
Il risultato dell'assegnamento degli autovalori è la matrice dei guadagni  $\mathbf{G}$ , avente come elementi della diagonale i guadagni proporzionali  $\text{diag}(\mathbf{G}) = [G_1, G_2, G_3, G_4]$  da applicare alle variabili di stato  $z = [x, \theta, \dot{x}, \dot{\theta}]$ .

In figura 10.14 è riportato lo schema di controllo implementato in Simulink per simulare il comportamento dinamico del controllore sviluppato.

Lo schema può essere suddiviso nelle seguenti componenti fondamentali:



(a) Mappa dei poli e zeri.

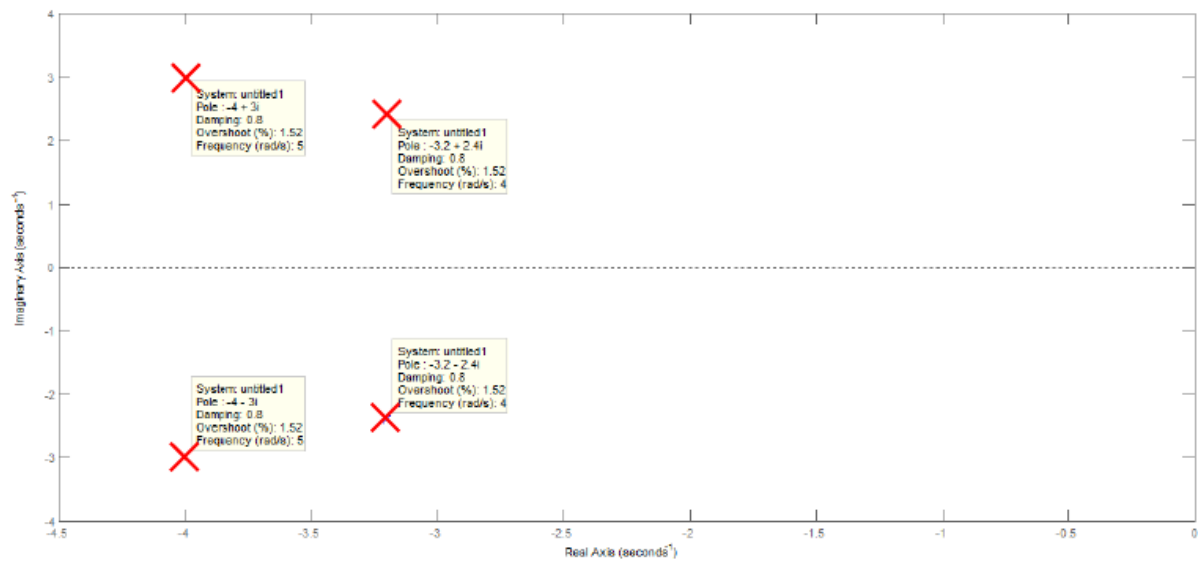


(b) Risposta allo scalino di coppia.

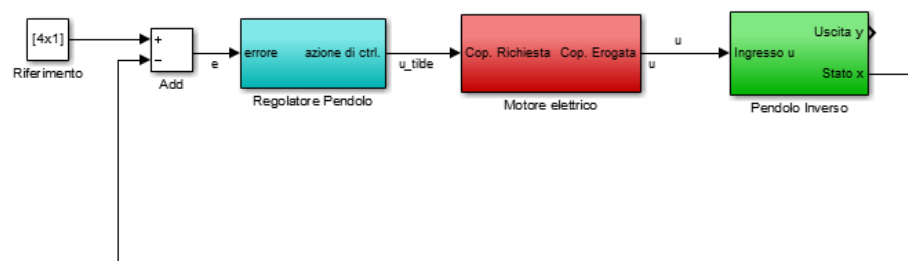
**Figura 10.12:** Analisi della dinamica del pendolo in anello aperto

- *Controllore*, costituito dalla matrice dei guadagni  $\mathbf{G}$ ;
- *Motore elettrico DC*, modellato come descritto nel paragrafo 1 del capitolo 7;
- *Modello del pendolo inverso*, come descritto in 2.1;
- *Retroazione negativa dello stato*;
- *Riferimento*, costituito dallo stato  $z_{rif} = [0, 0, 0, 0]$ , corrispondente al pendolo in posizione d'equilibrio instabile.

Per terminare la trattazione della sintesi del controllo, occorre aggiungere che la lettura delle variabili di stato, ovvero posizione e velocità di carrello e asta, sono state opportunamente filtrate per togliere eventuali componenti in alta frequenza. In figura

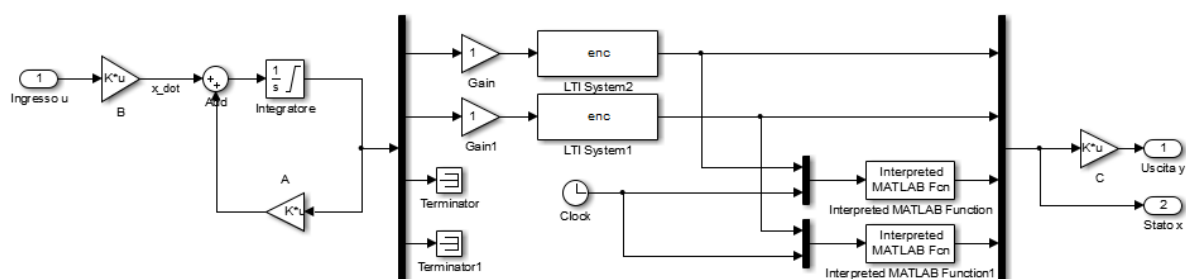


**Figura 10.13:** Controllo del pendolo inverso con posizionamento dei poli



**Figura 10.14:** Schema di controllo del pendolo inverso

10.15 è mostrato l'inserimento dei filtri per encoder e resolver nel sistema del pendolo inverso.



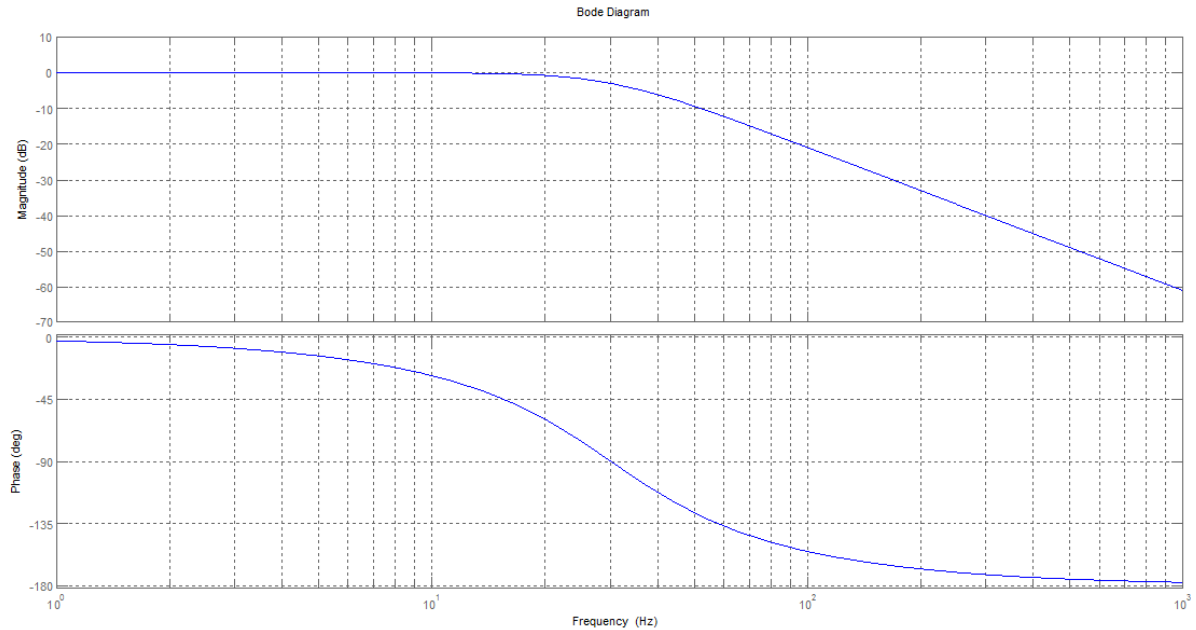
**Figura 10.15:** Introduzione dei filtri di posizione e velocità nello schema di controllo del pendolo inverso

In particolare, sono stati progettati dei filtri del secondo ordine con funzione di trasferi-

mento della forma

$$F_{filter}^{II}(s) = \frac{\omega_N^2}{s^2 + 2\xi\omega_N s + \omega_N^2} . \quad (10.18)$$

Per il controllo del pendolo inverso è stato scelto un filtro a smorzamento  $\xi = 0.8$  e frequenza  $f_N = 30 \text{ Hz}$  ( $\omega_N = 2\pi f_N$ ) di cui in figura 10.16 vengono riportati i diagrammi di Bode.



**Figura 10.16:** Diagrammi di Bode dei filtri di posizione e velocità del pendolo

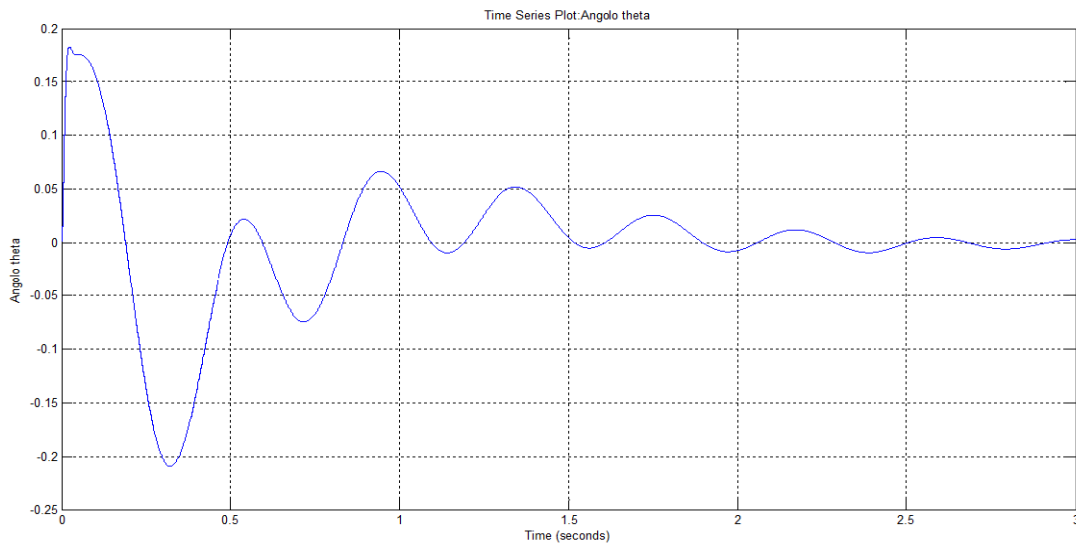
## 2.3 Simulazione

Nella fase di simulazione è stato studiato il comportamento del controllo a frequenze di controllo elevate. Infatti, il PC real-time utilizzato permette l'acquisizione dei segnali di controllo e la generazione del segnale di coppia del motore alla frequenza operativa di  $1 \text{ kHz}$ .

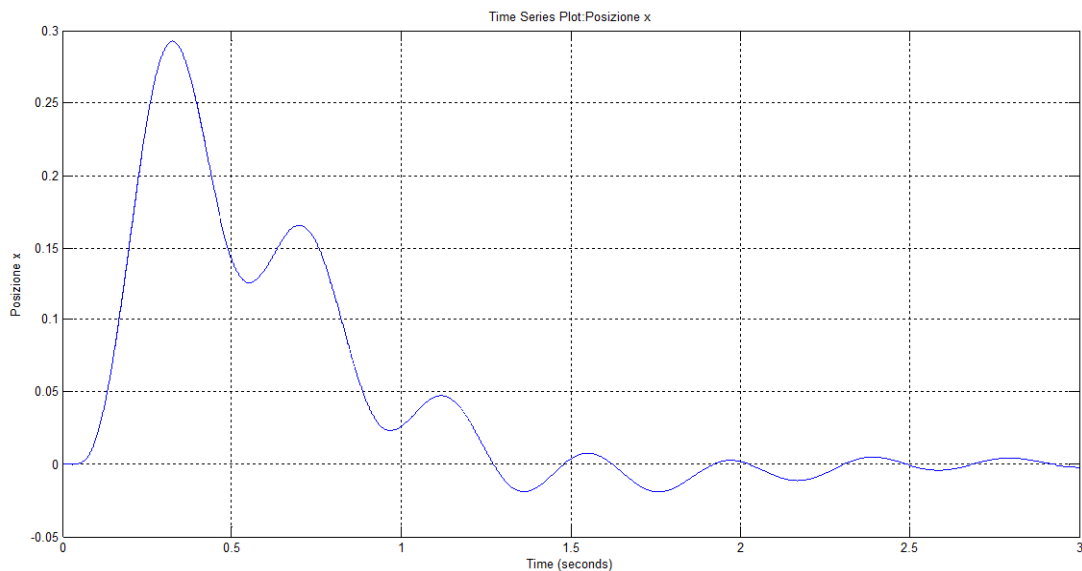
Nei diagrammi mostrati di seguito, l'asta del pendolo viene inizialmente posizionata a  $10^\circ$  e successivamente avviato il controllo per la sua stabilizzazione.

In figura 10.17 è mostrato l'andamento dell'asta del pendolo. La stabilizzazione nell'intorno dello 0 passa per l'inversione dell'angolo dell'asta e la completa stabilizzazione è raggiunta in circa  $2 \text{ s}$ .

L'oscillazione dell'asta appena presentata viene ottenuta attraverso la traiettoria del carrello mostrata in figura 10.18. Come si può notare, per invertire l'iniziale posizione dell'asta



*Figura 10.17: Pendolo inverso: angolo dell'asta in simulazione*



*Figura 10.18: Pendolo inverso: posizione del carrello in simulazione*

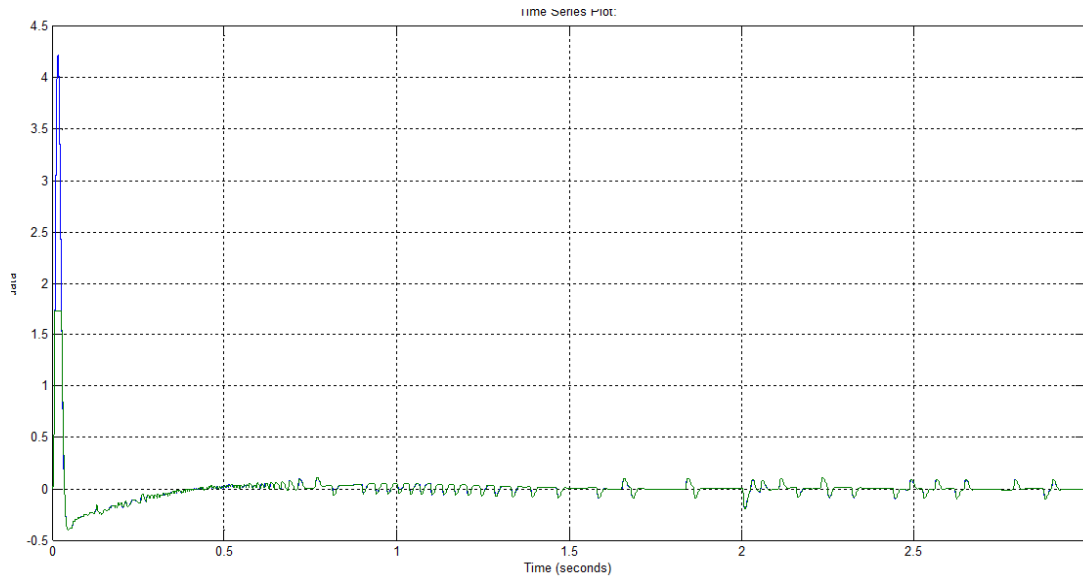
del pendolo è necessaria una certa corsa.

Per imporre questo movimento al carrello è necessario imporre una determinata coppia al motore, mostrata in figura 10.19.

## 2.4 Prove sperimentali

Come si è visto, in fase di modellazione non si è tenuto conto delle non linearità del sistema, come l'attrito e la cedevolezza della trasmissione. Di conseguenza, il sistema reale non si comporta allo stesso modo del sistema simulato e, per stabilizzare il sistema, è stato necessario modificare la posizione dei poli.

Nel primo tentativo, il controllore assegna i poli del sistema in modo da avere  $\omega_1 = 4 \text{ rad/s}$ ,



**Figura 10.19:** *Pendolo inverso: coppia motrice in simulazione*

$\omega_2 = 10 \text{ rad/s}$  e  $\xi = 0.9$ . In figura 10.20 vengono riportati i risultati ottenuti.

Una volta raggiunta la stabilità, le oscillazioni del carrello sono costanti perché è stata aggiunta una piccola componente di coppia per spostare il pendolo dalla sua posizione di equilibrio e permetterne una nuova stabilizzazione.

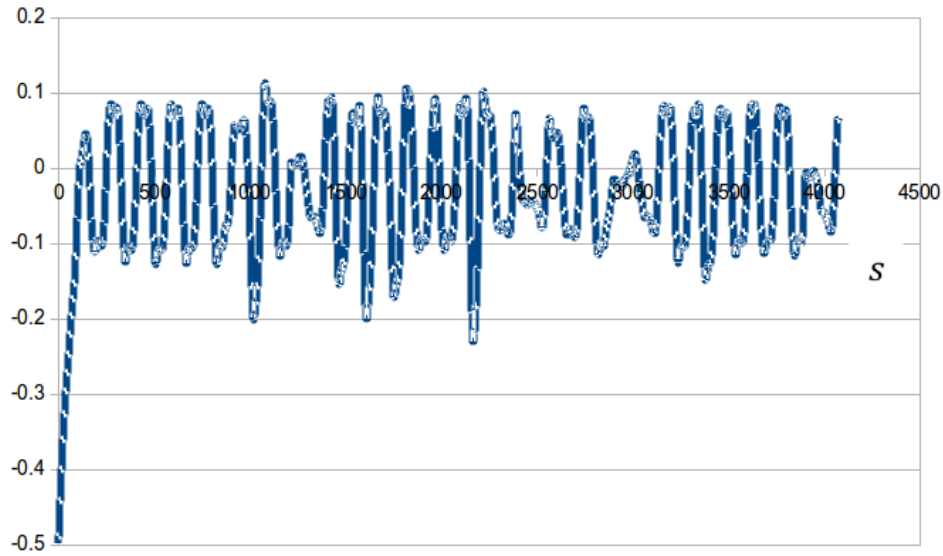
Il sistema controllato si è dimostrato stabile anche di fronte a perturbazioni del controllo, introdotte, ad esempio, dall'applicazione di piccole forze sulla massa del pendolo.

Maggiori prestazioni possono essere raggiunte aumentando la frequenza dei poli del sistema, ma ad un certo punto questo va a discapito della stabilità del controllo.

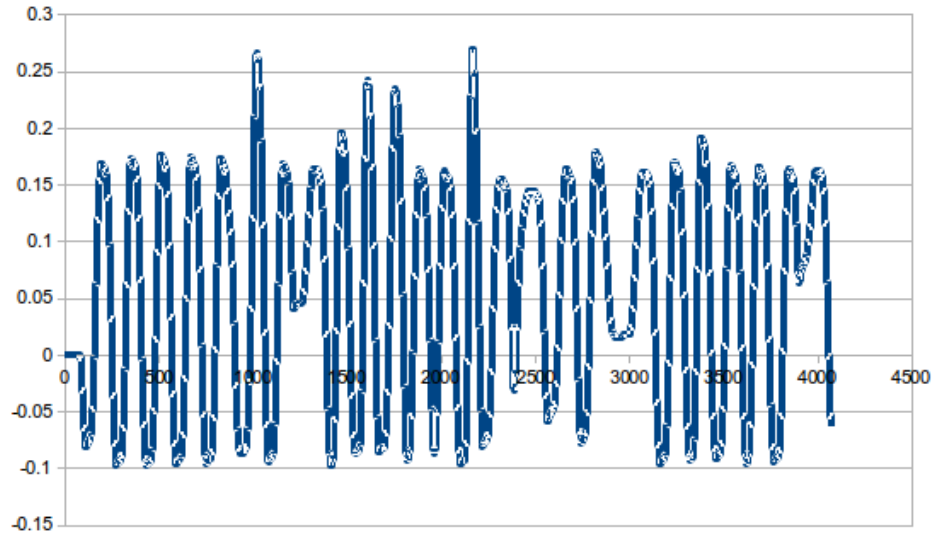
### 3 Controllo in asservimento visivo

Essendo il pendolo inverso un sistema caratterizzato da un'alta dinamica, il principale problema da affrontare per valutare la fattibilità dell'applicazione di un controllo in asservimento visivo è legato alla problematica del *multirate*. Si suppone infatti di riuscire a raggiungere, nello sviluppo del sistema di visione, un framerate d'elaborazione di  $50 \text{ Hz}$ , ovvero una frequenza 20 volte inferiore alla frequenza d'acquisizione delle informazioni di posizione dall'encoder dell'asta del pendolo e dal resolver del motore. Di conseguenza, prima di affrontare il problema dal punto di vista dell'algoritmo di elaborazione immagini da sviluppare per estrarre le informazioni del pendolo, è necessario soffermarsi sulla sintesi del controllo con assegnamento degli autovalori. L'analisi è stata effettuata in due fasi successive, provando prima a sottocampionare la lettura dei sensori di posizione e studiando successivamente lo schema di controllo vision-in-the-loop complessivo.





(a) Posizione angolare dell'asta in radianti.



(b) Posizione del carrello in metri.

**Figura 10.20:** Risultati sperimentali del controllo classico del pendolo inverso,

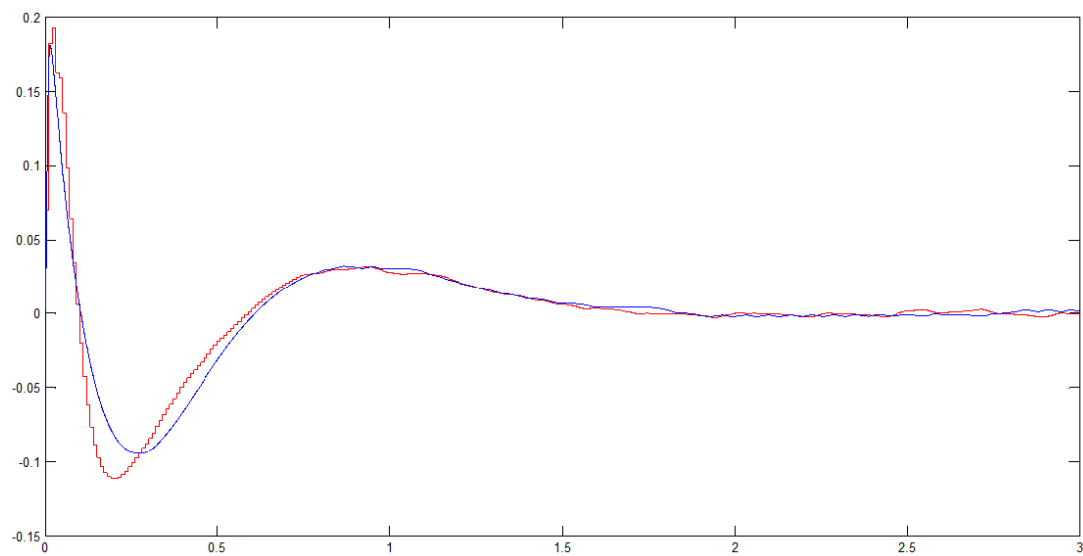
$$\omega_1 = 4 \text{ rad/s}, \omega_2 = 10 \text{ rad/s} \text{ e } \xi = 0.9$$

### 3.1 Sottocampionamento

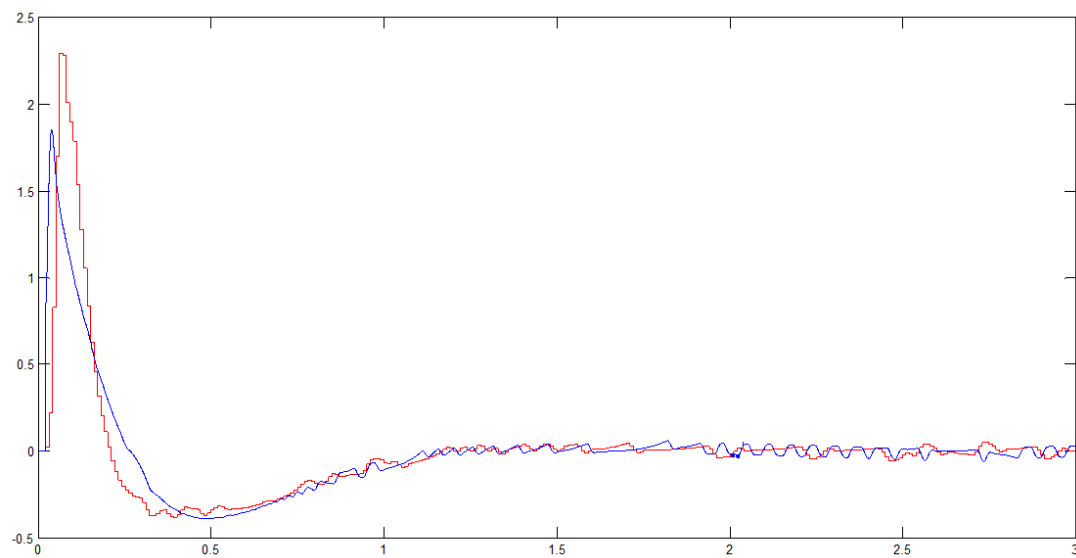
La prima prova effettuata esclusivamente in fase di simulazione è costituita da un sottocampionamento di entrambi i segnali di posizione, cioè sia quello proveniente dall'encoder che quello del resolver.

In figura 10.21 è riportato in blu l'andamento della posizione e velocità angolare del sistema con campionamento a  $1 \text{ kHz}$ , mentre in rosso si osserva il comportamento a  $100 \text{ Hz}$ . Come si vede, il sistema è leggermente più lento, ma il campionamento a  $100 \text{ Hz}$

è più che sufficiente a tracciare la dinamica del sistema e, quindi, portare a compimento il compito di stabilizzazione del pendolo.



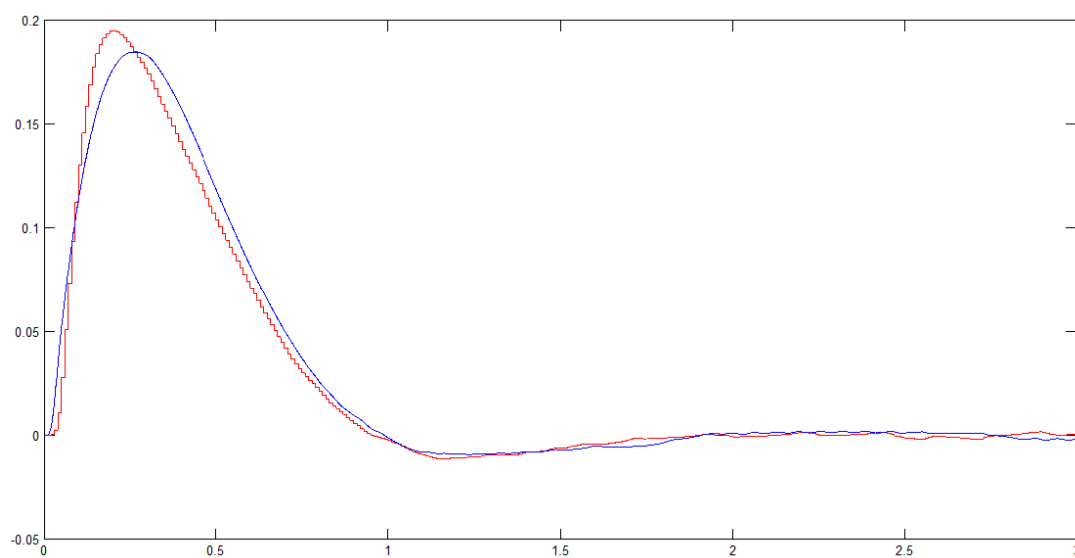
(a) *Posizione angolare.*



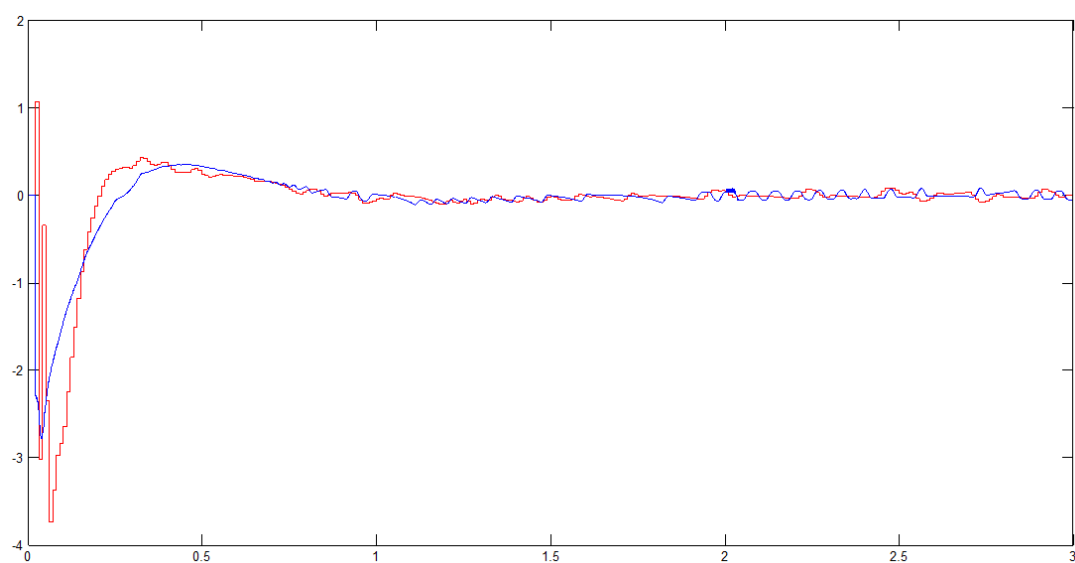
(b) *Velocità angolare.*

**Figura 10.21:** *Pendolo Inverso, simulazione: posizione e velocità angolare con sottocampionamento a 100 Hz*

In figura 10.22 vengono mostrate la relativa posizione e velocità traslazionali del carrello.



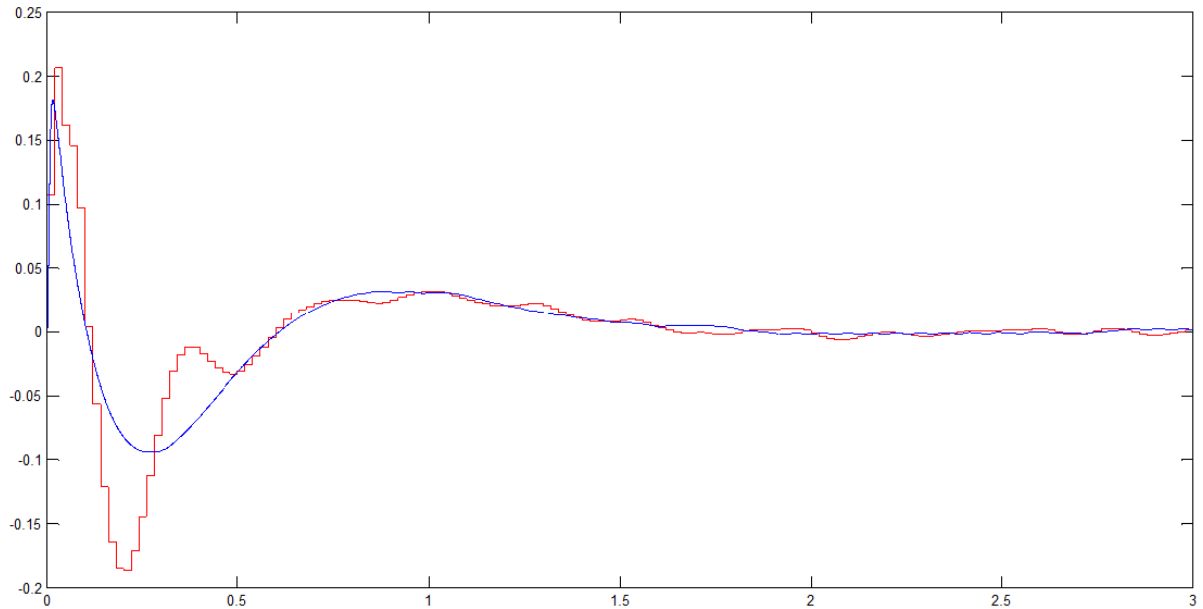
(a) Posizione del carrello.



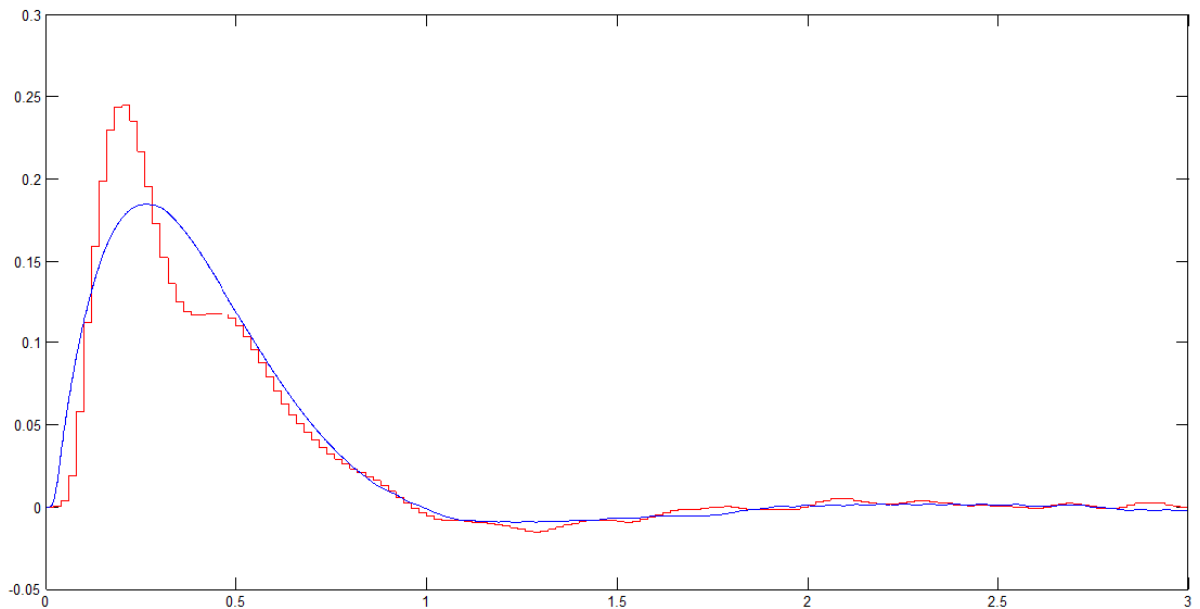
(b) Velocità del carrello.

**Figura 10.22:** Pendolo Inverso, simulazione: posizione e velocità del carrello con sottocampionamento a 100 Hz

Sottocampionando ulteriormente a  $50\text{ Hz}$  la lettura delle posizioni di carrello e asta, il sistema controllato rimane ancora stabile, ma la risposta dinamica è ancora più lenta. In figura 10.23 l'andamento di posizione del carrello e dell'asta ottenute dalla simulazione.



(a) Posizione angolare dell'asta.



(b) Posizione del carrello.

**Figura 10.23:** Pendolo Inverso, simulazione: posizione di carrello e asta con sottocampionamento a  $50\text{ Hz}$

Dall'analisi effettuata si può concludere che il controllo del sistema con acquisizione dello stato a frequenza  $50\text{ Hz}$  permette ancora la stabilizzazione del sistema. Tuttavia, tale

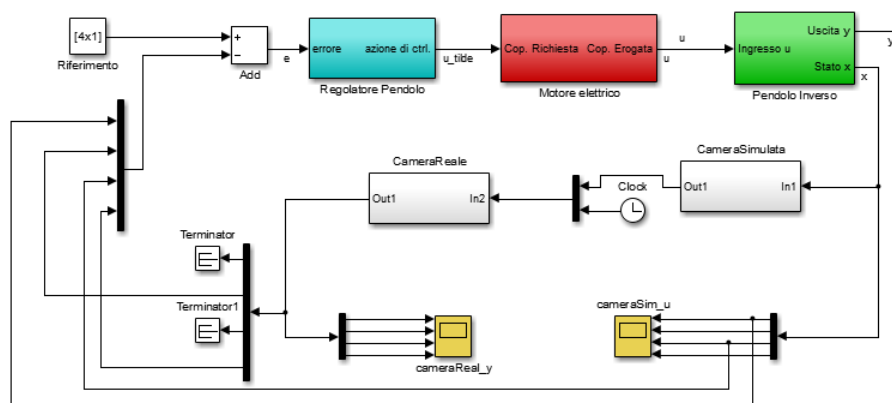
controllo è sottoposto alla presenza di alcuni requisiti fondamentali:

- Calibrazione della telecamera per stima precisa della posizione del carrello. La posizione non può essere definita nello spazio immagine.
- Ottenimento di una frequenza di elaborazione immagini costante a  $50\text{ Hz}$ , ovvero molto simile alla frequenza d'acquisizione delle immagini.

Per tali motivi, si è deciso di applicare il controllo in asservimento visivo unicamente alle variabili di stato associate alla posizione angolare del pendolo, per le quali è possibile effettuare le misure direttamente nello spazio delle immagini. Al contrario, la lettura della posizione del carrello continua ad avvenire per mezzo della lettura dei segnali provenienti dal resolver del motore. Tale lettura può essere effettuata alla massima frequenza operativa, cioè  $1\text{ kHz}$ , oppure sottocampionata ad una frequenza a scelta.

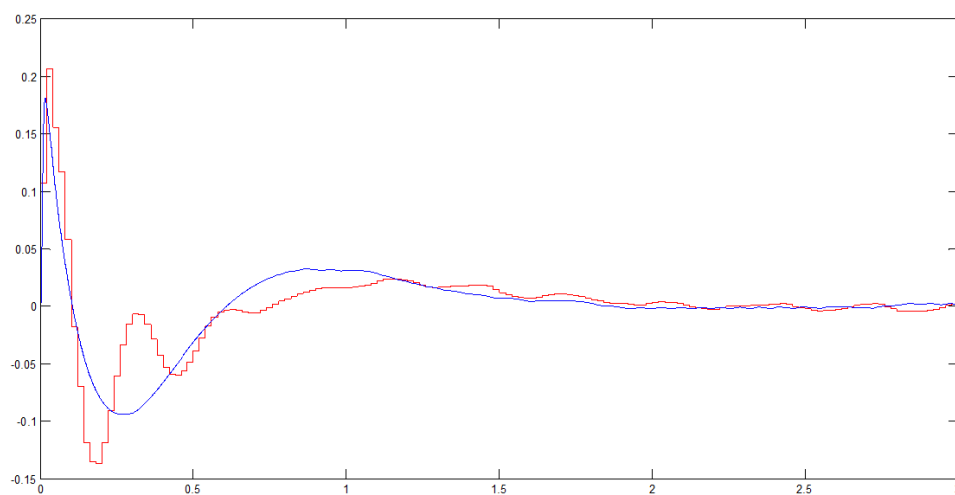
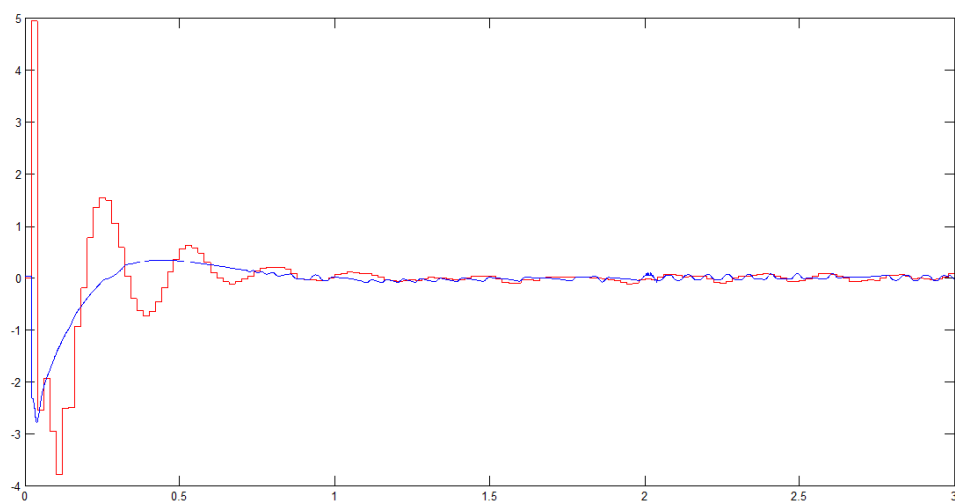
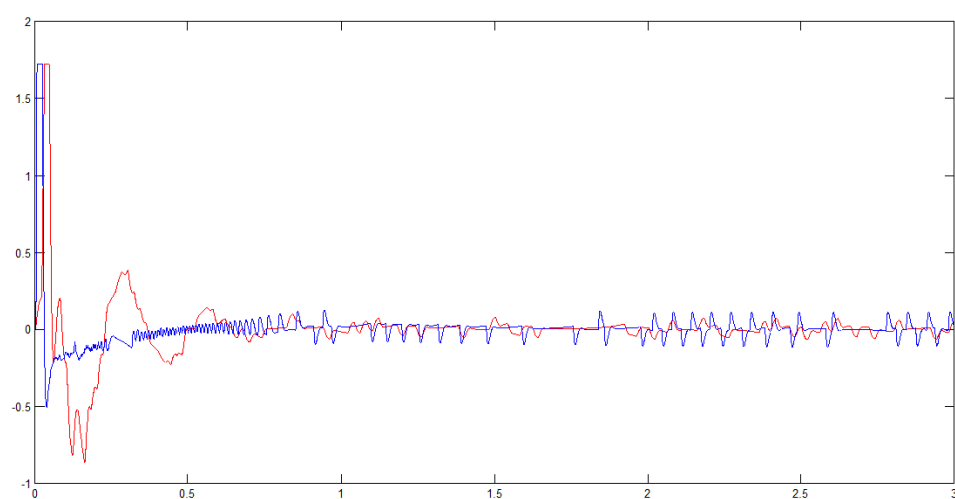
### 3.2 Controllo Vision-In-The-Loop

Per l'introduzione della componente di visione nello schema di controllo, lo schema di figura 10.14 viene esteso con l'introduzione del sistema di visione nell'anello di retroazione del controllo, come mostrato in figura 10.24.



**Figura 10.24:** Schema di controllo vision-in-the-loop del pendolo inverso

In figura 10.25 è riportata la risposta ottenuta dal sistema sottoposto a controllo in asservimento visivo a  $50\text{ Hz}$  relativamente alle variabili di stato legate alla posizione angolare dell'asta del pendolo.

(a) *Posizione angolare dell'asta.*(b) *Velocità angolare dell'asta.*(c) *Coppia motrice.*

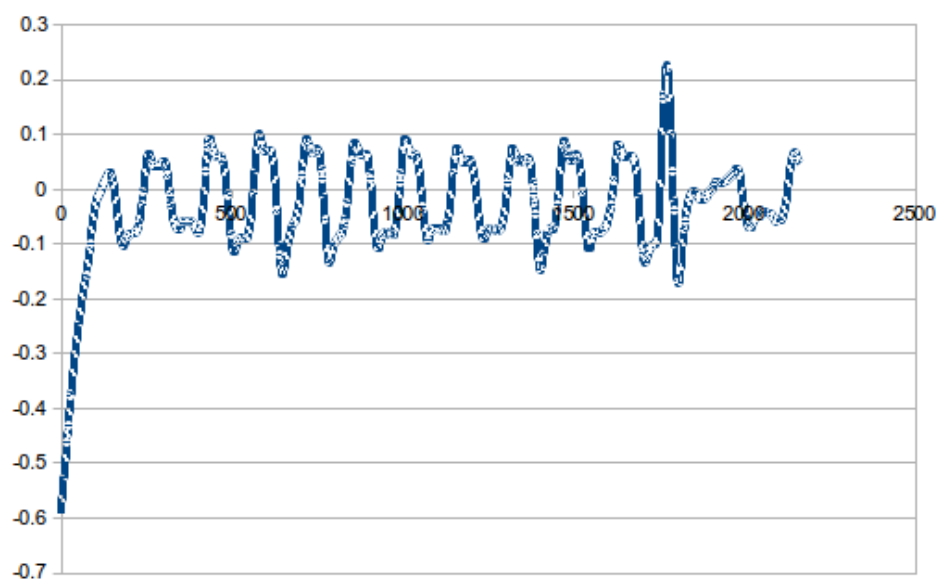
**Figura 10.25:** *Pendolo inverso, simulazione: controllo del pendolo inverso con asservimento visivo a 50 Hz*

### 3.3 Risultati sperimentali

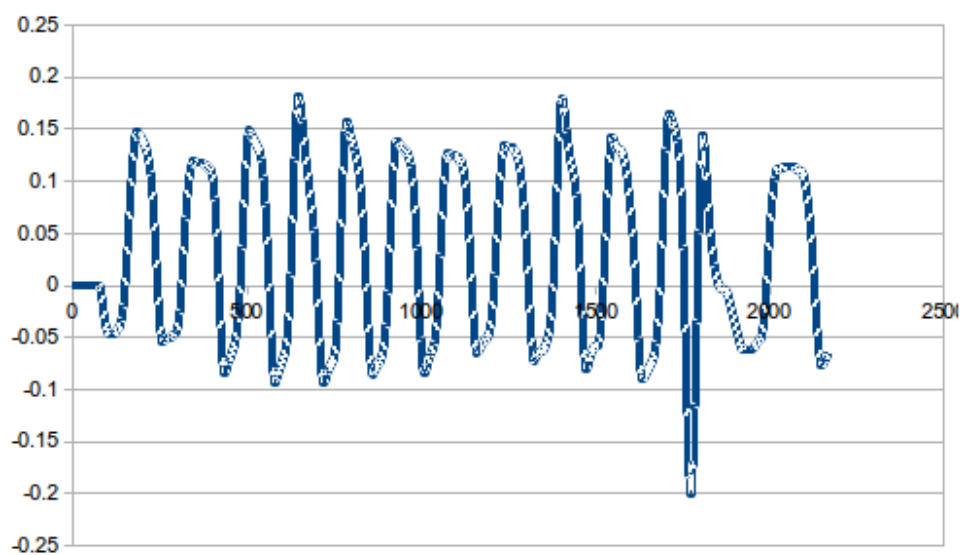
In figura 10.26 vengono riportati i risultati ottenuti sottocampionando la lettura del segnale di posizione angolare dell'asta. Le considerazioni che si possono trarre dall'osservazione degli andamenti sono le medesime già fatte in precedenza. Per quanto riguarda il passaggio dal sistema simulato al sistema reale, si ottiene un peggioramento delle prestazioni a causa della presenza degli attriti e delle cedevolezze non considerate nella realizzazione del modello. Per quanto riguarda invece il sottocampionamento, questo porta ad una riduzione della velocità di risposta del sistema, reso evidente, nel caso del pendolo, dall'aumento dell'ampiezza delle oscillazioni di carrello e asta.

Il peggioramento della prontezza di risposta del sistema è ancora più evidente se si confrontano i risultati ottenuti imponendo due frequenze di sottocampionamento differenti. In figura 10.27 vengono visualizzati i diagrammi relativi all'andamento delle oscillazioni del pendolo nel caso di sottocampionamento a  $50\text{ Hz}$  e  $200\text{ Hz}$ . Sull'asse delle ascisse sono indicati i campioni, mentre sulle ordinate vi sono i gradi di inclinazione dell'asta del pendolo. Nel caso del campionamento a frequenza più elevata, l'ampiezza dell'oscillazione è inferiore di 1 o 2 gradi.

Per concludere, si è provato a spingere oltre il controllo, sottocampionando i segnali a  $25\text{ Hz}$ . I risultati ottenuti sono mostrati in figura 10.28. L'acquisizione dei segnali a questa frequenza non è più sufficiente a mappare la dinamica del sistema e la prontezza del controllo risulta talmente ridotta che il sistema controllato non è più stabile. In particolare, nel tentativo di rincorrere la stabilizzazione dell'asta in posizione verticale, il carrello del pendolo va a sbattere contro i finecorsa software e meccanici.



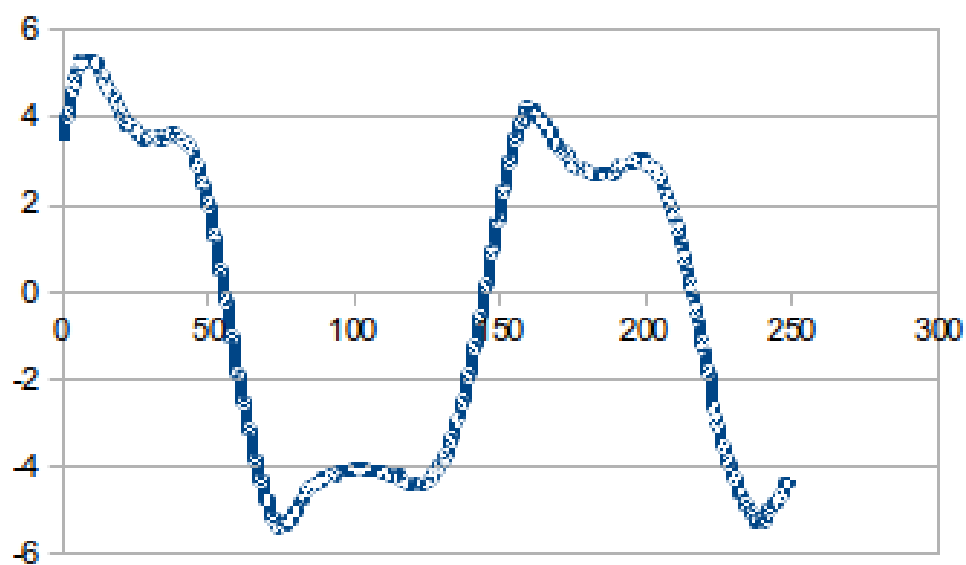
(a) Posizione angolare dell'asta.



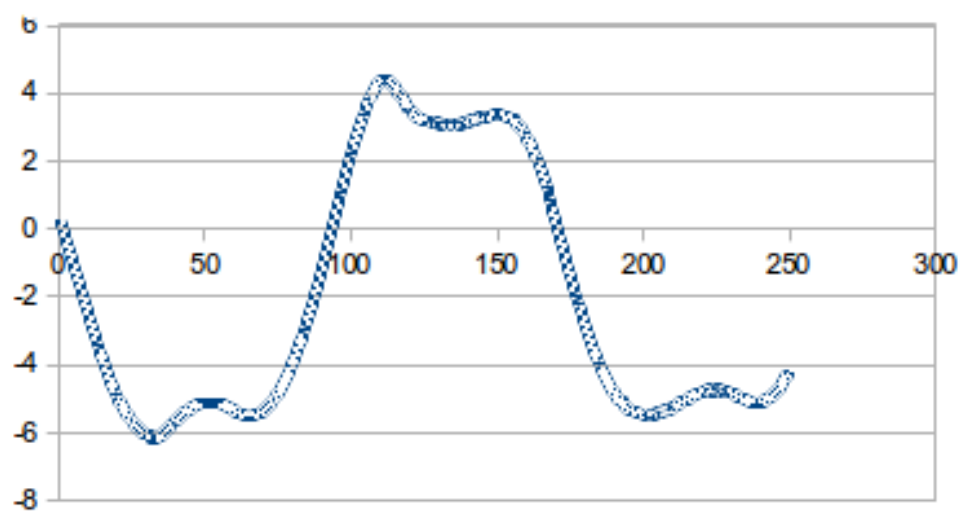
(b) Posizione del carrello.

**Figura 10.26:** Pendolo inverso, prova sperimentale: sottocampionamento a 50 Hz



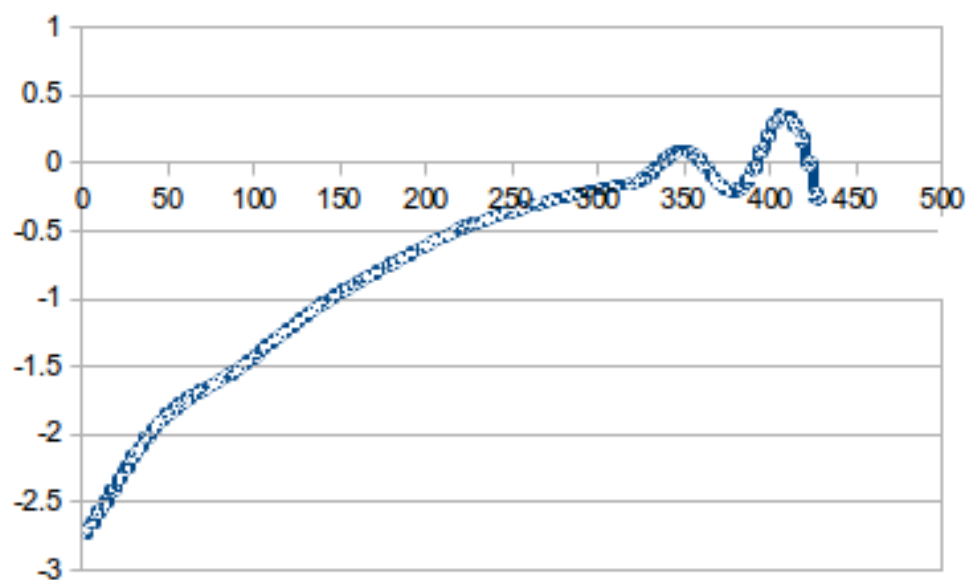


(a) Posizione angolare asta per  $f_1 = 50$  Hz.

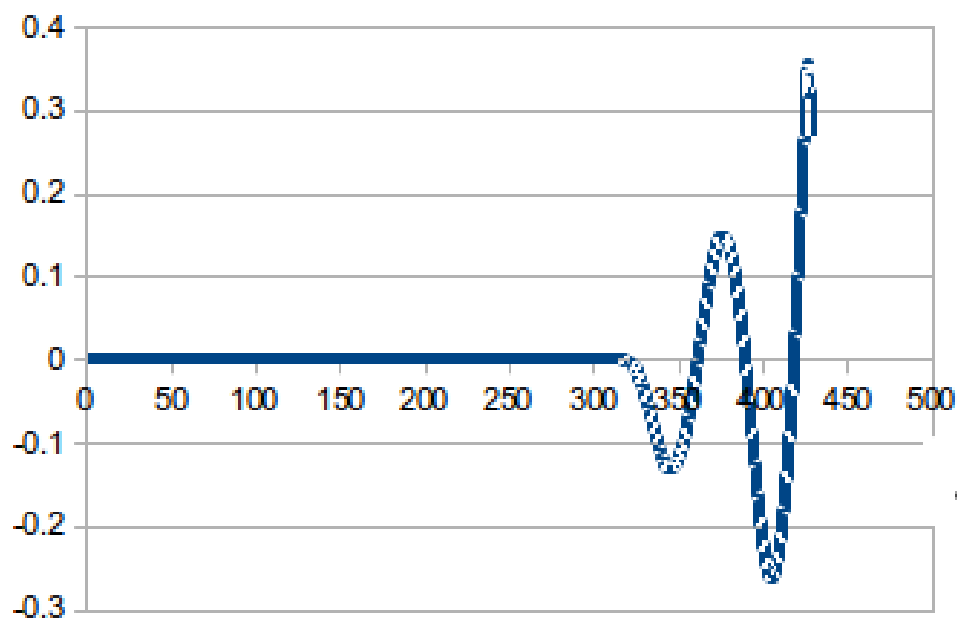


(b) Posizione angolare asta per  $f_2 = 200$  Hz.

**Figura 10.27:** Pendolo inverso, prova sperimentale: confronto tra sottocampionamento a  $f_1 = 50$  Hz e  $f_2 = 200$  Hz



(a) Posizione angolare dell'asta.



(b) Posizione del carrello.

*Figura 10.28: Pendolo inverso, prova sperimentale: sottocampionamento a 25 Hz*

## 4 Elaborazione delle immagini

Per lo sviluppo del controllo in asservimento del pendolo, il sistema di visione non interviene direttamente nella generazione della legge di controllo. Tuttavia, esso deve fornire al controllore le informazioni relative alle variabili di stato del sistema, utilizzate successivamente dal controllore per il calcolo dell'errore di stabilizzazione dell'asta del pendolo. In questo senso, l'asservimento visivo del pendolo inverso può essere visto come un sistema *dynamic look and move*, al contrario dell'applicazione relativa al posizionamento del disco in cui il sistema di visione influisce direttamente sul controllo.

Come descritto nel paragrafo 1.1, la telecamera è posta esattamente di fronte alla guida lineare su cui scorre il carrello. Inoltre, si suppone che l'asse  $X$  della terna di riferimento della telecamera sia perfettamente parallelo all'asse  $X$  del sistema di riferimento del pendolo (direzionato come la guida lineare). Infine, si suppongono note le distanze (i)  $x_{cam}^{carr} = x_{cam} - x_{carr}$ , rappresentante l'offset di posizionamento lungo l'asse  $X$  della telecamera rispetto al centro della guida lineare e (ii) la distanza operativa della telecamera  $z_{cam}^{carr}$ .

L'obiettivo del sistema di elaborazione delle immagini è il tracciamento della posizione del pendolo nelle varie immagini appartenenti al flusso video acquisito tramite il dispositivo di acquisizione immagini. In particolare, in ogni immagine è necessario individuare le coordinate in pixel  $(u_c, v_c)$  del carrello e dell'appendice finale dell'asta del pendolo stesso  $(u_p, v_p)$ . Dati questi due punti è possibile ottenere le misure delle variabili di stato desiderate:

- *Posizione angolare dell'asta.*

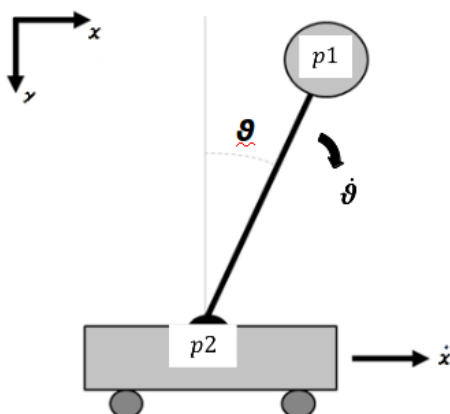
$$\theta = \arctan \left( \frac{|v_p - v_c|}{u_p - u_c} \right) . \quad (10.19)$$

- *Posizione del carrello.* Questa variabile di stato deve essere calcolata relativamente al sistema di riferimento utilizzato per il controllo, ovvero con terna di riferimento centrata sulla guida lineare. Inoltre, la stima è subordinata alla conoscenza dei parametri intrinseci della telecamera  $(f_x, f_y, c_u, c_v)$ , ottenibile tramite calibrazione.

$$x_c = \frac{z_{cam}^{carr}}{f_x} (u_c - c_u) + x_{cam}^{carr} . \quad (10.20)$$

- *Velocità.* Le problematiche relative alla stima della velocità nel tracciamento sono descritte nel paragrafo 1.2.4 del capitolo 6. Oltre all'operazione di derivazione delle posizioni, è stato necessario un opportuno filtraggio, come descritto nel paragrafo relativo al controllore, per evitare l'insorgenza di valori di stima troppo elevati.

In figura 10.29 è schematizzato il modello delle feature del pendolo da estrarre dalle immagini.



**Figura 10.29:** Schematizzazione delle feature del pendolo

In figura 10.30 è riportata l'immagine tipo del sistema ripreso dal dispositivo di acquisizione immagine. Si può notare come lo sfondo della scena non sia costante e l'asta del pendolo non spicchi in modo netto da esso.



**Figura 10.30:** Immagine del pendolo inverso ripresa dal sistema di visione

Ai fini dell'ottenimento del controllo in asservimento visivo del pendolo inverso, il sistema di elaborazione delle immagini deve rispettare alcuni vincoli prestazionali, tra cui:

- Tempo d'elaborazione del ciclo di tracciamento inferiore a 40 ms;
- Robustezza rispetto a variazioni di luminosità e della scena inquadrata;
- Evitare o minimizzare la fase di inizializzazione e calibrazione del sistema.

L'algoritmo di tracciamento può essere suddiviso in due fasi distinte: (i) la segmentazione dell'immagine per identificare i pixel che possono appartenere al pendolo e (ii) l'analisi di questo sottoinsieme di pixel per l'estrazione delle due feature d'interesse. Lo sviluppo di entrambe queste fasi è descritto nel paragrafo 4.1. Invece, per quanto riguarda l'algoritmo di tracciamento vero e proprio, come verrà descritto nel paragrafo 4.2, l'informazione relativa alla posizione del pendolo nel frame precedente viene utilizzata per discriminare false letture dall'individuazione della posa corretta. Gli algoritmi utilizzati in seguito sono alcuni degli approcci presentati nel capitolo 3.

## 4.1 Segmentazione

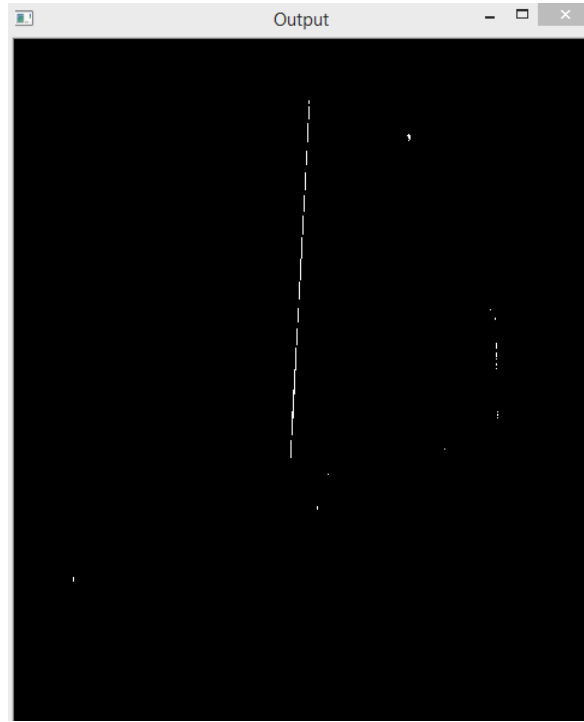
Come già detto, l'obiettivo della segmentazione dell'immagine è l'estrazione dei singoli pixel appartenenti al carrello e all'asta del pendolo.

La tecnica di segmentazione dell'immagine più semplice e, di conseguenza, facilmente applicabile al tracciamento real-time di oggetti, è la semplice *sogliatura dell'immagine*. Poiché le immagini a disposizione sono a colori, è possibile applicare la trasformazione dal piano colore RGB al piano HSV (Hue-Saturation-Value), più adatto per individuare i corretti valori di soglia del pendolo. Da un'analisi di immagini campione simili a quella in figura 10.30 si individuano i seguenti valori di soglia:

$$\begin{cases} 0 \leq H_{thr} \leq 179 \\ S_{thr} = 0 \\ 243 \leq V_{thr} \leq 255 \end{cases} .$$

Sogliando l'immagine di partenza con questi valori, si ottiene il risultato mostrato in figura 10.31, in cui si può osservare che i pixel appartenenti all'asta del pendolo vengono individuati quasi completamente. Inoltre, è presente poco rumore di fondo, la cui quasi completa assenza garantisce la robustezza dell'algoritmo.

Ad ogni modo, modificando leggermente i valori di soglia, ad esempio ampliando il range dei valori di saturazione ammessi ( $0 \leq S_{thr} \leq 8$ ), si osserva un consistente aumento del rumore di fondo. L'immagine ottenuta è mostrata in figura 10.32. Con questi nuovi valori di soglia vengono marcati come appartenenti al pendolo alcuni elementi appartenenti



**Figura 10.31:** Sogliatura dell'immagine del pendolo inverso con  $0 \leq H_{thr} \leq 179$ ,

$$S_{thr} = 0, 243 \leq V_{thr} \leq 255$$



**Figura 10.32:** Sogliatura dell'immagine del pendolo inverso con  $0 \leq H_{thr} \leq 179$ ,

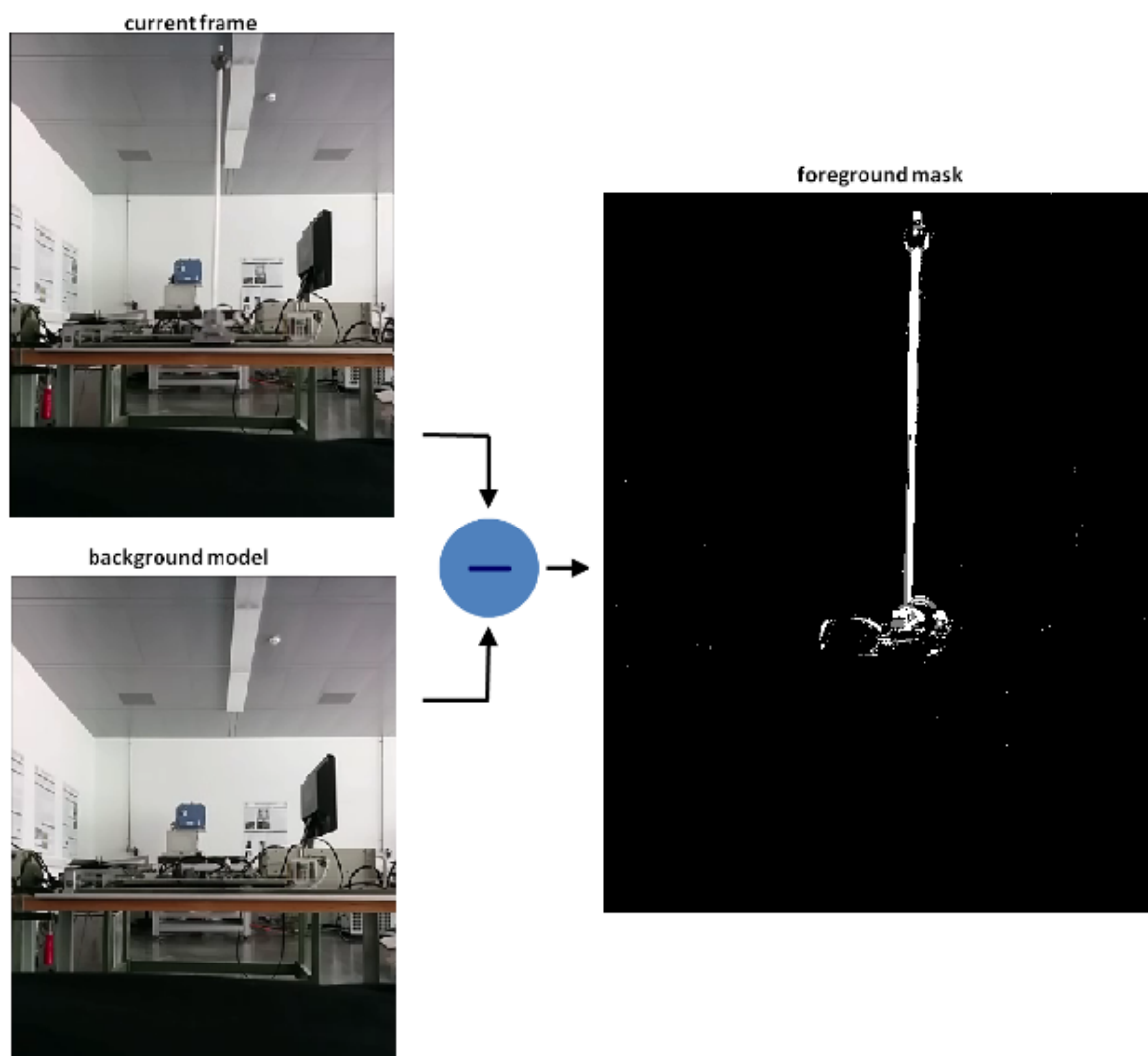
$$0 \leq S_{thr} \leq 8, 243 \leq V_{thr} \leq 255$$

invece allo sfondo della scena inquadrata.

Per tale motivo, nonostante l'efficienza computazionale garantita dalla tecnica di sogliatura

dell'immagine, essa non garantisce la robustezza richiesta per il controllo in asservimento visivo. Infatti, variazioni delle condizioni di luminosità ambientali della scena possono portare a variazioni della soglia utile ad estrarre le caratteristiche del pendolo.

Per migliorare la segmentazione dell'immagine si è pensato di fornire al sistema di elaborazione delle immagini una conoscenza a priori sulla scena da trattare, ovvero lo sfondo della scena. Sottraendo da ogni immagine uno sfondo assoluto, è possibile eliminare il rumore di fondo introdotto dalla procedura di sogliatura dell'immagine. In figura 10.33 vengono mostrate l'immagine acquisita, l'immagine di sfondo con cui il sistema di elaborazione è stato addestrato e il risultato della sottrazione.



**Figura 10.33:** Segmentazione del pendolo per sottrazione dello sfondo

Come si può vedere, la procedura porta all'ottenimento di un'immagine binarizzata quasi perfetta, con poco rumore di fondo e con l'oggetto individuabile facilmente. Inoltre, questo approccio non è particolarmente oneroso, poiché si tratta unicamente di eseguire una

semplice operazione su ogni pixel dell'immagine. Di conseguenza, la complessità dell'algoritmo dipende esclusivamente dalla dimensione dell'immagine analizzata. D'altro canto, ancora una volta il problema è costituito dal comportamento dell'algoritmo in presenza di condizioni di luminosità differenti. In questo caso, infatti, anche l'intensità dei pixel dello sfondo cambia e fa in modo che la sottrazione dello sfondo non riesca ad evitare l'aggiunta di pixel appartenenti allo sfondo all'insieme dei pixel del pendolo.

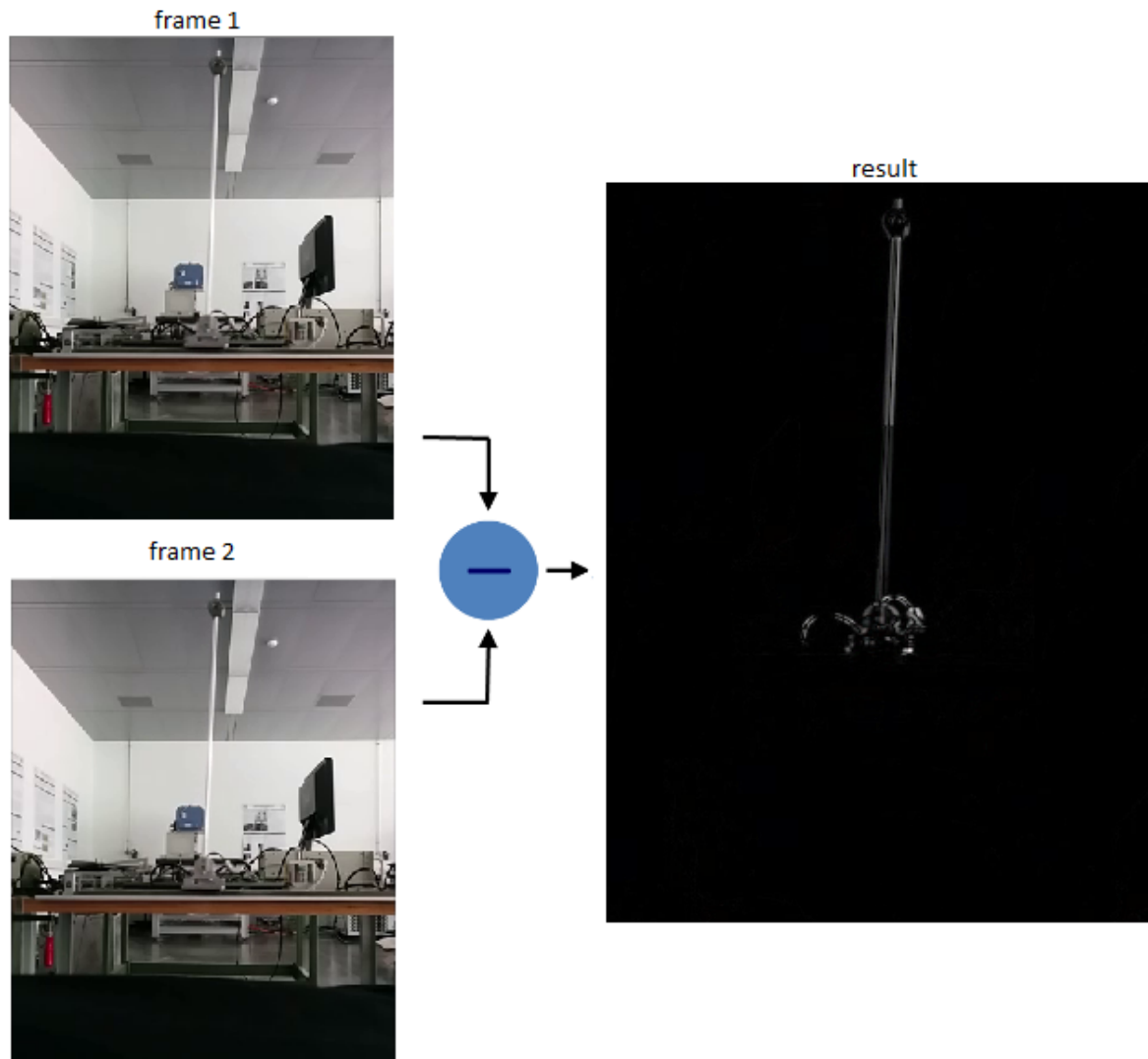
Un ulteriore irrobustimento del tracciamento delle caratteristiche delle immagini è fornito da una semplice considerazione riguardante il compito da portare a termine: l'oggetto da individuare è l'unico elemento della scena in movimento. Pertanto, è possibile utilizzare delle tecniche di analisi del *flusso ottico* per permettere l'individuazione delle sole componenti della scena in movimento. In questo modo viene introdotta una componente appartenente alla classe delle tecniche afferenti al problema del tracciamento del movimento della scena.

In particolare, l'approccio scelto è quello di aggiornare continuamente l'immagine di sfondo, utilizzando l'immagine acquisita all'istante  $t - 1$  come immagine di sfondo per l'immagine acquisita all'istante  $t$ . In questo modo viene evidenziata unicamente la parte in movimento, eliminando sempre lo sfondo e aggiornandone allo stesso tempo le condizioni di luminosità. In figura 10.34 è riportato il risultato dell'operazione.

Se il pendolo rimane fermo, il risultato della segmentazione è un'immagine completamente nera, nessuna feature del pendolo viene estratta e lo stato del sistema non viene aggiornato. Infine, tale approccio rispetta tutti i requisiti funzionali desiderati: (i) l'implementazione è semplice e non troppo onerosa, (ii) la tecnica permette di tracciare le componenti dinamiche della scena e (iii) non c'è bisogno di inizializzare il sistema poiché l'identificazione può cominciare a partire dalla seconda immagine acquisita.

Il passo finale della segmentazione consiste nel mettere ancora più in evidenza i pixel appartenenti al pendolo. Infatti, il valore di luminosità dei pixel del pendolo ottenuto attraverso la semplice differenza tra immagini consecutive non è massimizzato, ma è un valore in scala di grigi. Di conseguenza, viene effettuata una sogliatura dell'immagine su valori di luminosità non nulli ( $V_{thr} \geq 20$ ), in modo che i pixel del pendolo appaiano bianchi su sfondo nero. Infine, per rendere più semplice la successiva elaborazione, l'immagine viene sottoposta a dilatazione, in modo che eventuali pixel appartenenti all'oggetto, ma separati da esso dalla procedura di differenza delle immagini, tornino ad essere connessi all'oggetto. Il risultato ottenuto è mostrato in figura 10.35.





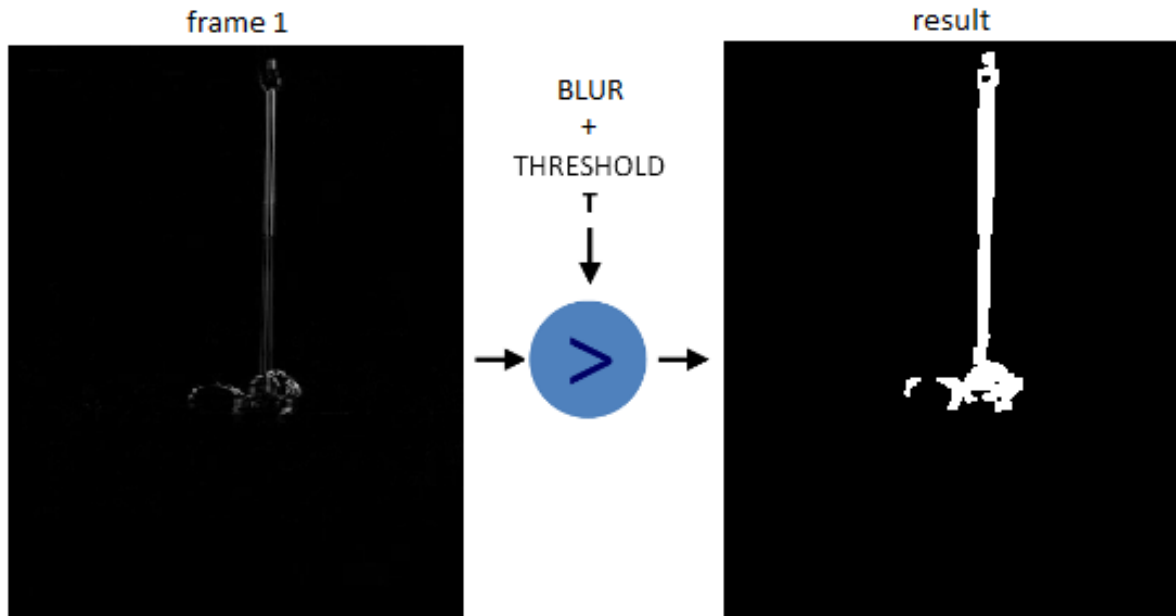
*Figura 10.34: Segmentazione del pendolo per sottrazione di immagini consecutive*

L'immagine ottenuta è finalmente adatta ad essere elaborata successivamente per l'estrazione delle feature del pendolo.

## 4.2 Estrazione delle feature

Lo scopo della fase di estrazione delle feature ha come scopo l'individuazione delle coordinate del carrello e dell'appendice del pendolo. Dall'immagine di figura 10.35 è evidente che tali caratteristiche costituiscono gli estremi del segmento costituito dai pixel bianchi dell'immagine.

L'individuazione di rette all'interno di un'immagine è un compito tipico di un sistema di elaborazione delle immagini, una cui possibile soluzione viene fornita dalla trasformata di Hough, descritta nel paragrafo 3.2 del capitolo 3. In particolare, le librerie openCv met-



**Figura 10.35:** Filtraggio finale dell'immagine differenza del pendolo

tono a disposizione una funzione in grado di fornire tutte le coppie di punti che formano segmenti all'interno dell'immagine analizzata, il cui prototipo è mostrato in 10.1.

---

**Codice 10.1:** Funzione OpenCv per la trasformata di Hough

---

```
1 void HoughLinesP (InputArray image, OutputArray lines, double rho,
    double theta,
2 int threshold, double minLineLength, double maxLineGap);
```

---

I parametri da cui dipende la corretta esecuzione della trasformata di Hough sono: (i) il passo `rho` di moduli da considerare, (ii) il passo di angoli `theta` da ricercare, (iii) il valore `threshold` dell'accumulatore della trasformata per affermare che i pixel appartengono effettivamente ad una retta, (iv) la lunghezza minima `minLineLength` del segmento per eliminare la residua presenza di falsi segmenti dovuti al rumore e (v) la distanza massima `maxLineGap` tra due punti perché essi possano appartenere alla stessa linea.

Il risultato dell'applicazione della trasformata di Hough alla figura 10.35 è riportato in figura 10.36.

Per filtrare ancora di più il risultato è sufficiente, in generale, prendere in considerazione solamente il segmento più lungo.

### 4.3 Problema di tracciamento

L'applicazione dell'algoritmo appena descritto per l'elaborazione delle immagini in tempo reale può portare all'insorgenza di un problema di tracciamento. Infatti, la differenza tra



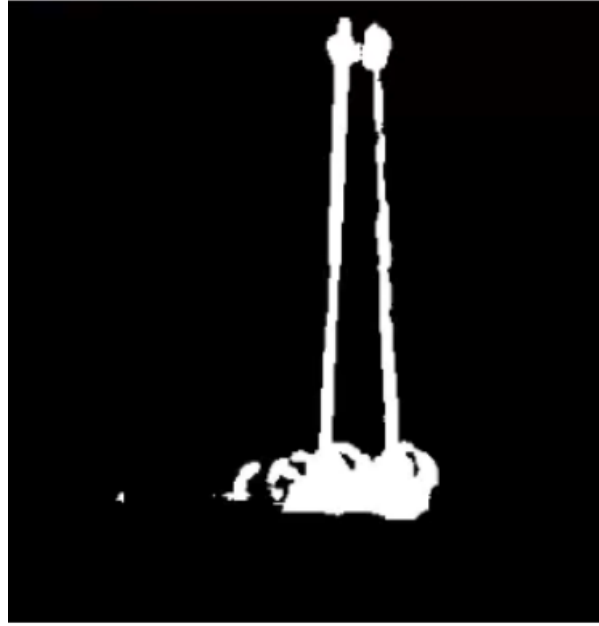
**Figura 10.36:** Risultato dell'applicazione della trasformata di Hough per la rilevazione del pendolo

due frame successivi porta alla rilevazione di due pendoli. Infatti, nell'immagine vi sono due differenze principali, date dalla posizione precedente del pendolo e dalla posizione attuale. Nella presentazione dell'algoritmo questo problema non è stato messo in evidenza poiché le due immagini analizzate sono state acquisite ad istanti temporali talmente vicini tra loro che il successivo filtraggio ha portato i due pendoli a convergere in uno solo. In realtà, se il tempo intercorrente tra l'acquisizione di due immagini successive è troppo elevato, i due pendoli potrebbero essere troppo distanti tra loro e il filtraggio non porta all'individuazione di un solo pendolo. Questa situazione è mostrata in figura 10.37.

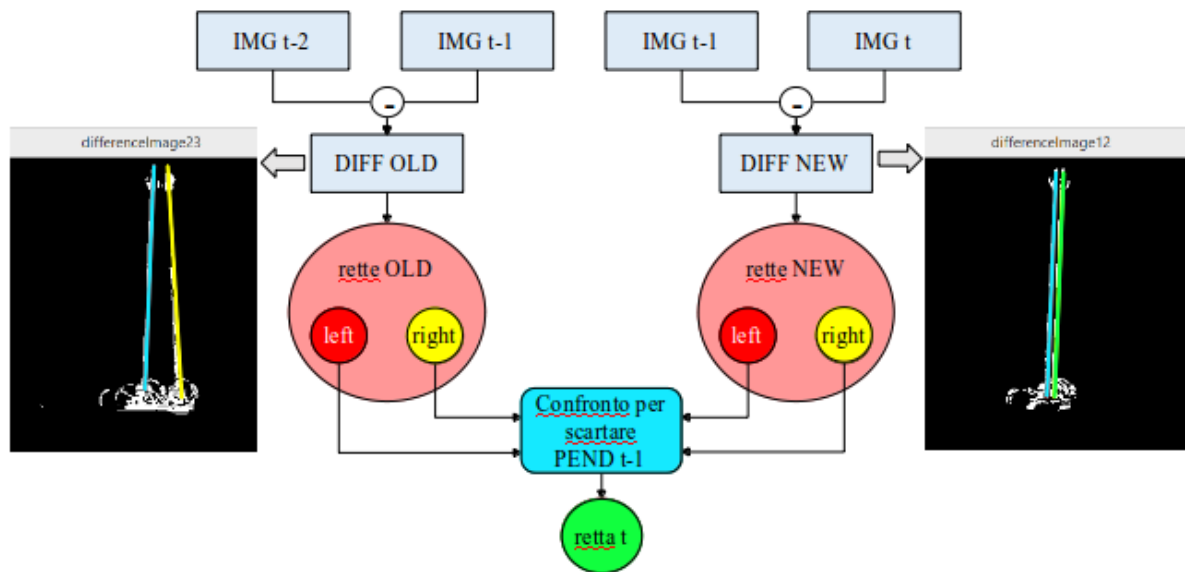
Per riuscire ad identificare la posizione corretta del pendolo, è necessario approfondire l'analisi. In particolare, l'informazione richiesta per discriminare il pendolo vero da quello "virtuale" è contenuta nella storia delle differenze tra frame ottenute nel corso dell'elaborazione. Salvando ad ogni passo la differenza ottenuta, è possibile applicare la procedura descritta dallo schema in figura 10.38.

La procedura può essere così riassunta:

1. All'istante  $t - 1$  viene trovata la differenza  $ID_{t-1}$  tra le immagini  $I_{t-2}$  e  $I_{t-1}$ .
2. Tramite trasformata di Hough vengono individuate due rette e la loro parametrizzazione è salvata in memoria insieme all'immagine  $I_{t-1}$ .
3. All'istante  $t$  viene ottenuta la differenza  $ID_t$  tra le immagini  $I_t$  e  $I_{t-1}$ .
4. Elaborando l'immagine  $ID_t$  si estraggono i parametri delle due rette individuate.



*Figura 10.37: Problema dell'individuazione di due pendoli*



*Figura 10.38: Algoritmo per la discriminazione di pendolo reale e virtuale nella differenza tra immagini consecutive*

5. Le rette dell'immagine  $ID_t$  vengono confrontate con quelle individuate nell'immagine  $ID_{t-1}$ .
6. La retta comune alle immagini  $ID_t$  e  $ID_{t-1}$  è quella relativa al pendolo virtuale. La retta rimanente in  $ID_t$  è la posa corrente del pendolo.
7. All'istante  $t - 1$  la procedura viene reiterata a partire dal punto 3.

La procedura completa di individuazione del pendolo, visualizzazione compresa, impiega tra i 30 e 50  $ms$ , quindi al limite della frequenza d'elaborazione necessaria per la realizzazione di un controllo completamente stabile.

## 4.4 Prova sperimentale

Per il test dell'algoritmo di visione presentato si è deciso di eseguire il controllo classico del pendolo inverso e, contemporaneamente, eseguire l'algoritmo di tracciamento del pendolo. Il confronto dei risultati ottenuti è mostrato in figura 10.39, dove la lettura della posizione tramite encoder e resolver è rappresentata in blu, mentre i risultati del dispositivo di visione è visualizzato con la curva verde.

Come si può vedere, la lettura effettuata col sistema di visione segue abbastanza fedelmente i risultati ottenuti in precedenza con la lettura dei sensori di posizione. Inoltre, come ci si aspettava, la lettura della posizione da parte della telecamera risulta in ritardo di alcuni campioni e l'errore è maggiore in corrispondenza di brusche variazioni di velocità del pendolo, ovvero a seguito delle inversioni del senso di marcia del carrello.

La stima della posizione del carrello è più precisa rispetto alla stima della posizione angolare dell'asta del pendolo, poiché per la stima della prima variabile di stato è sufficiente estrarre una sola feature dall'immagine (le coordinate del centro geometrico del carrello), mentre la stima della seconda è influenzata da due feature visuali (le coordinate del centro geometrico del carrello e quelle del centro dell'appendice finale dell'asta del pendolo).

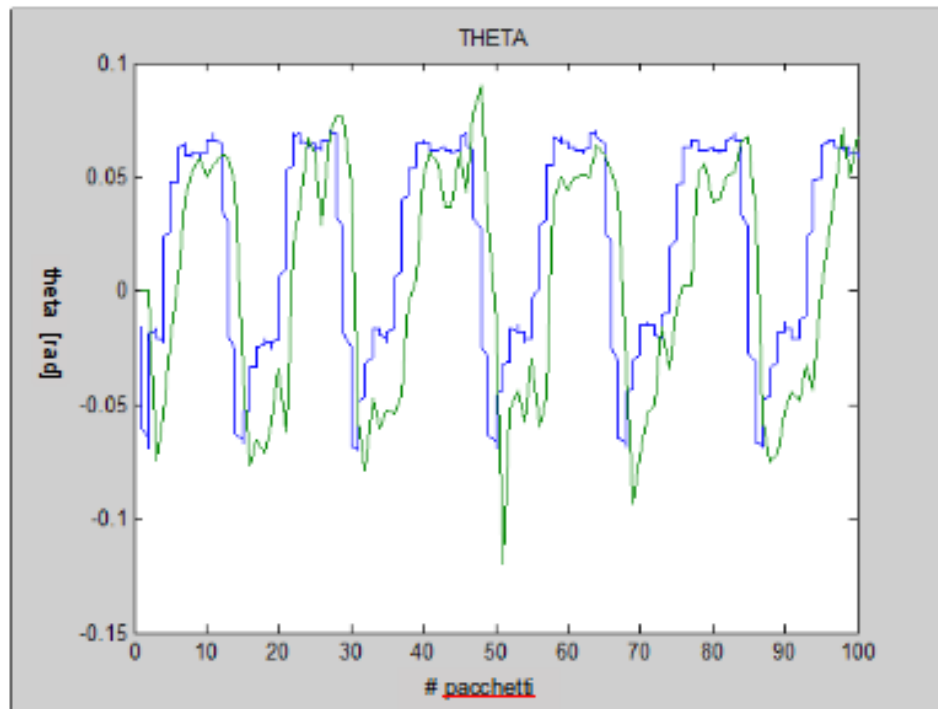
In figura 10.40 è riportata, a titolo d'esempio, una sequenza di tracciamento del pendolo inverso.

Ad ogni modo, visti i risultati, la sostituzione almeno parziale del sensore di posizione dell'asta col dispositivo di visione sembra percorribile. È quindi stato possibile procedere con la valutazione sperimentale dell'intero controllo in asservimento visivo.

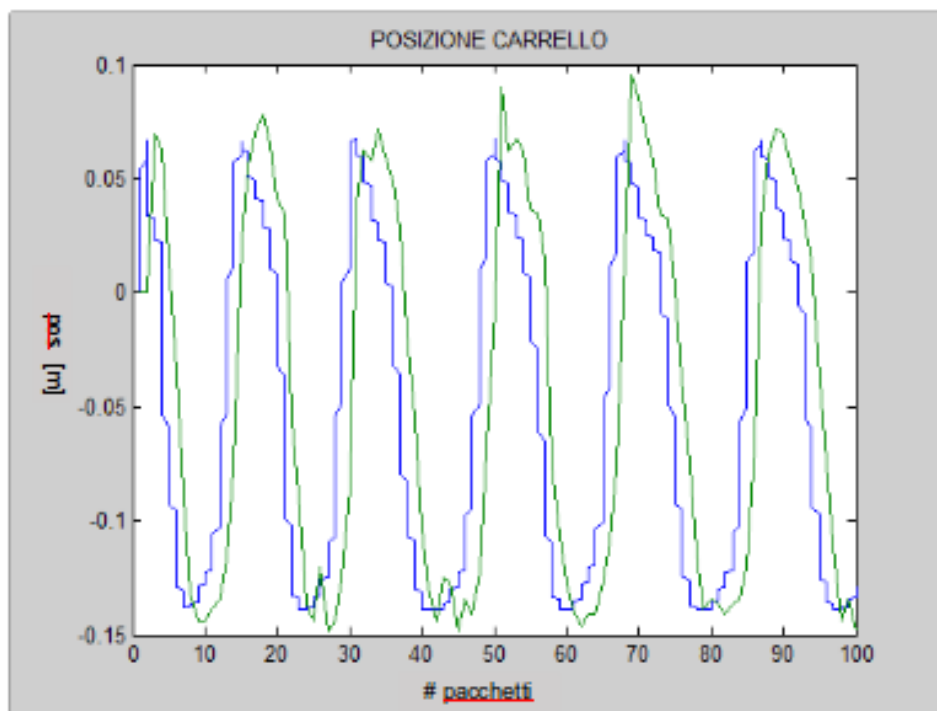
## 5 Risultati sperimentali

Per l'esecuzione delle prove sperimentali, il controllo del pendolo inverso utilizzato è un misto del controllo classico presentato nel paragrafo 2 e del controllo in asservimento visivo realizzato tramite l'elaborazione delle immagini presentata nel paragrafo 4.

In particolare, l'esecuzione di ogni prova ha seguito i seguenti passi:



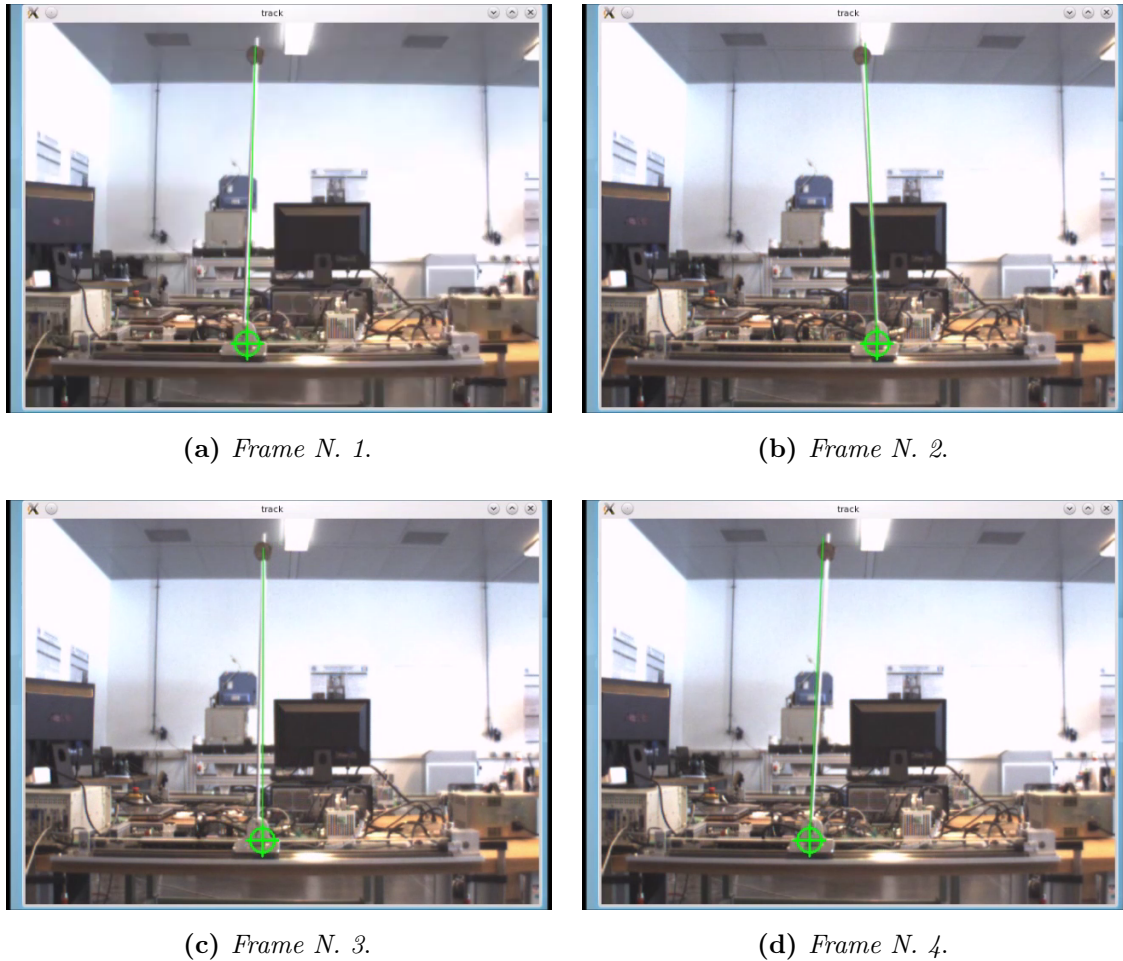
(a) Posizione angolare dell'asta.



(b) Posizione del carrello.

**Figura 10.39:** Confronto tra le letture delle variabili di stato dei sensori di posizione e del sistema di visione

1. A sistema fermo viene impostato lo zero dell'encoder per la lettura della posizione angolare dell'asta del pendolo e del resolver per la posizione del carrello.



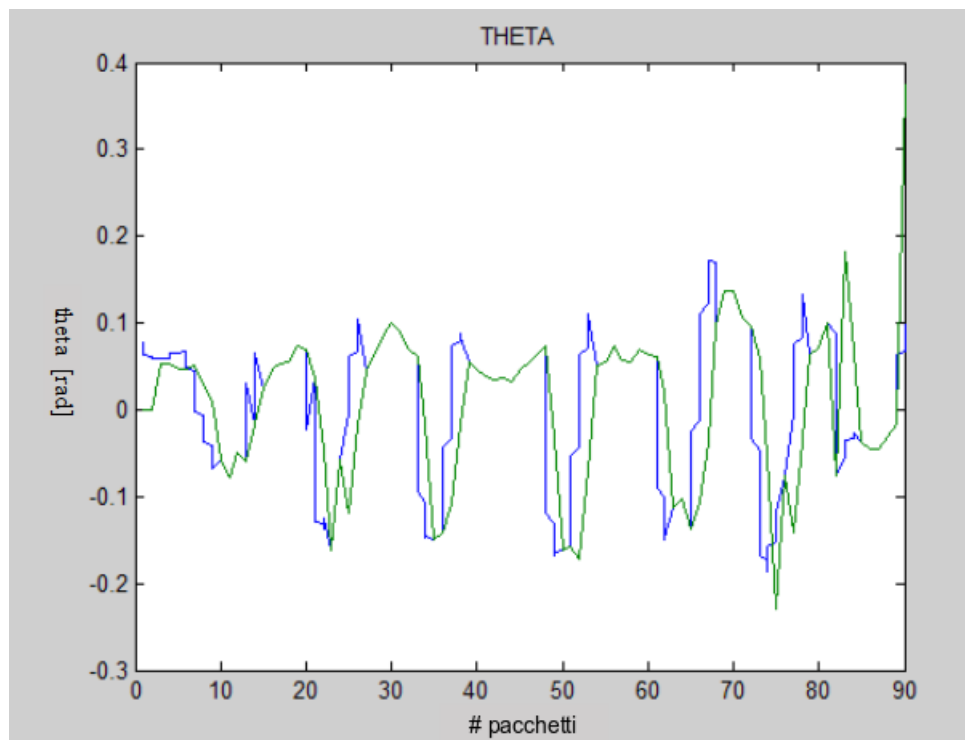
**Figura 10.40:** Sequenza di tracciamento del pendolo inverso

2. Il pendolo viene sollevato a mano nella sua posizione verticale.
3. Una volta che l'encoder montato alla base dell'asta del pendolo misura un angolo di  $\pm 5^\circ$ , il pendolo viene stabilizzato per mezzo del controllo classico.
4. Viene avviato il software di acquisizione ed elaborazione immagini ed inizializzato il canale di comunicazione col controllore.
5. Una volta acquisito un certo numero di immagini, il pendolo inizia ad essere stabilizzato per mezzo del controllo in asservimento visivo.
6. Durante l'esecuzione del controllo vision-in-the-loop, se si presenta un errore troppo elevato tra la lettura della posizione angolare dell'asta eseguita per mezzo dell'encoder e la stessa lettura effettuata dal sistema di visione, il controllo è effettuato per mezzo della lettura dell'encoder, per cui la stabilità del sistema è già stata verificata.

Nei diagrammi mostrati in seguito, vengono mostrati i risultati ottenuti dal solo controllo in asservimento visivo, ovvero a partire dal punto 5. In blu sono rappresentate le letture

effettuate coi sensori di posizione, mentre in verde quelle del sistema di visione. Per il controllo è stato utilizzato esclusivamente la lettura della posizione angolare dell'asta.

In figura 10.41 è riportato l'andamento della stima della posizione angolare dell'asta. L'inseguimento è analogo a quanto già analizzato nel caso del tracciamento senza controllo. Come ci si aspetta, il controllo del pendolo è più "brusco", nel senso che l'ampiezza dell'oscillazione del pendolo è maggiore rispetto a quella ottenuta col controllo classico. Questo risultato è dato dal fatto che la lettura della telecamera arriva in ritardo e quindi il sistema si accorge dopo del fatto che il pendolo sta cadendo ed è necessario spostare il carrello per raddrizzarne la posizione. Inoltre, come già visto nell'analisi dell'algoritmo di tracciamento, il ritardo si fa sentire maggiormente nel momento in cui il carrello cambia direzione di marcia.

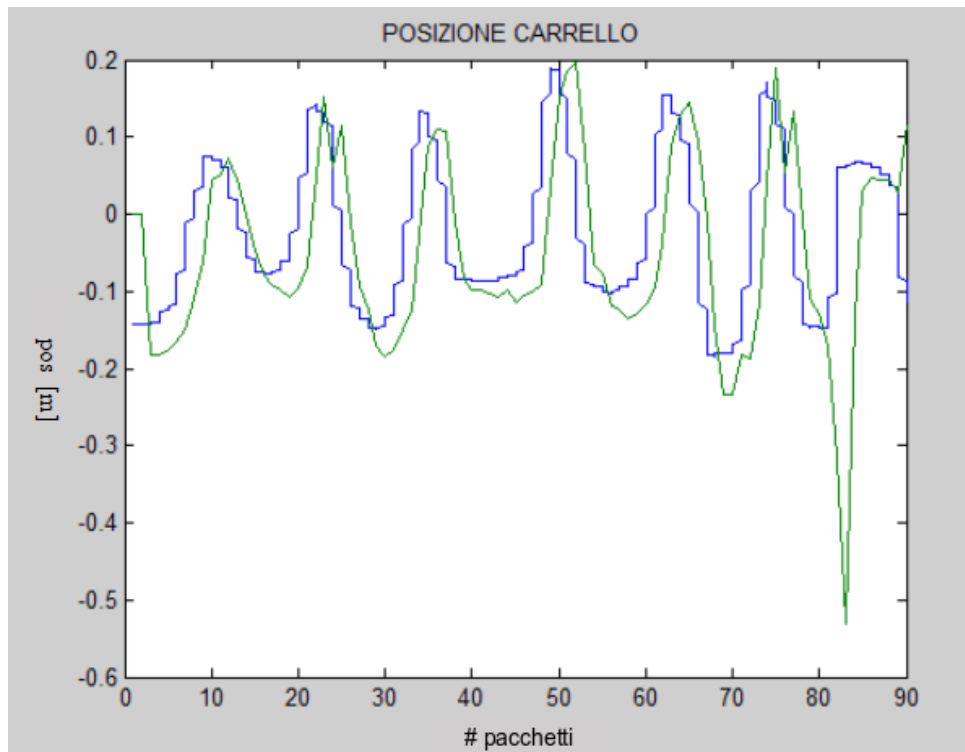


**Figura 10.41:** Stima della posizione angolare dell'asta per l'asservimento visivo

In figura 10.42 si riporta l'andamento della posizione del carrello, sia come stimata dal sistema di visione che come letta dal resolver del motore. La stima da parte del sistema di visione presenta alcuni errori a causa della mancata calibrazione della telecamera. Inoltre, dall'analisi delle letture del resolver si nota come l'oscillazione del carrello risulti meno ripetibile rispetto al caso del controllo classico e l'ampiezza dell'oscillazione stessa sia maggiore (circa 400 mm). Nel caso del sistema realizzato, questo costituisce il primo problema nella realizzazione dell'asservimento visivo: il sistema è al limite della stabilità e l'ampia



oscillazione del carrello porta lo stesso ad andare a sbattere contro i finecorsa della guida lineare.

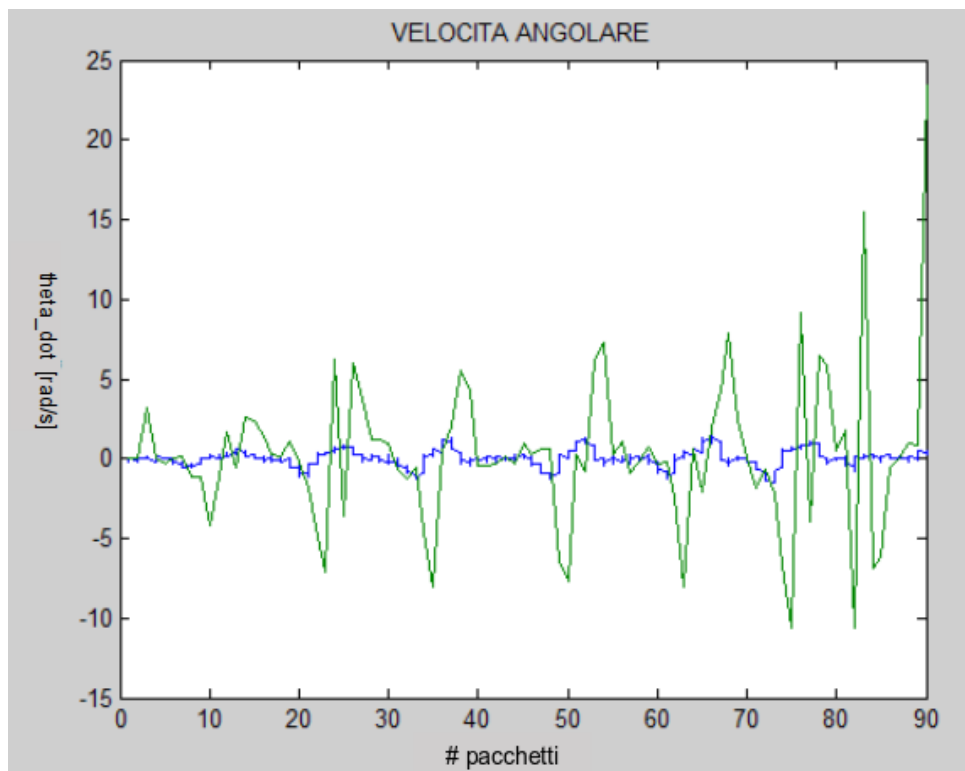


**Figura 10.42:** Stima della posizione del carrello per l'asservimento visivo

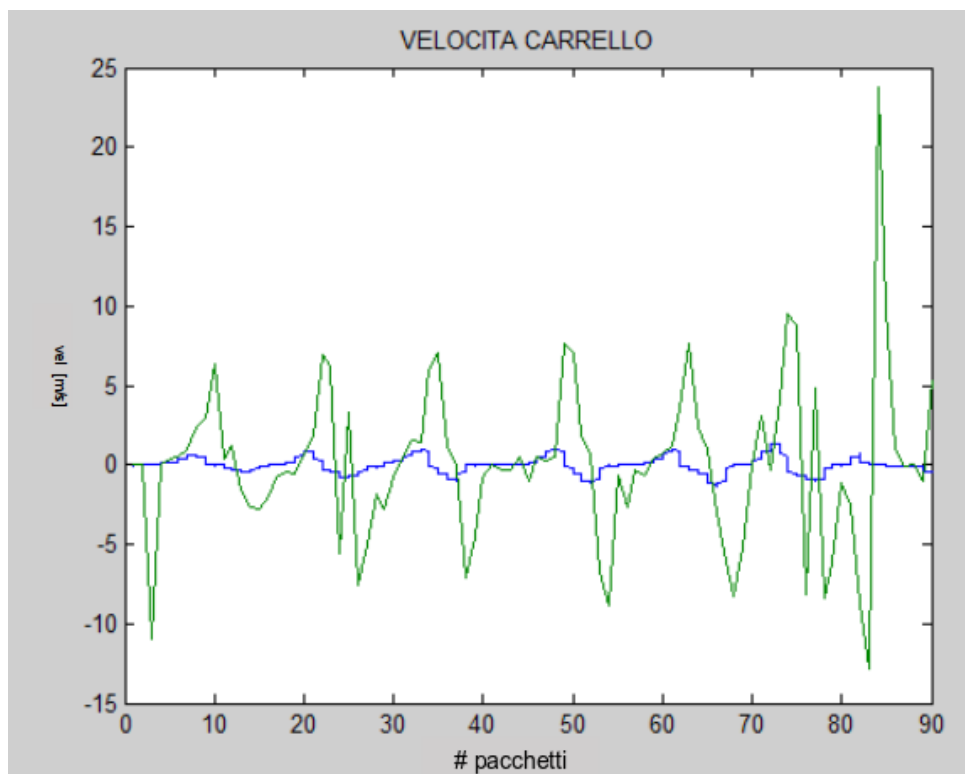
Per completezza si riportano in figura 10.43 l'andamento delle velocità del carrello e dell'asta. Le stime da parte del sistema di visione non sono direttamente utilizzabili dal sistema di controllo a causa della grande differenza tra il tempo di campionamento dall'encoder e dal sistema di visione. Il loro utilizzo è possibile solo dopo un opportuno filtraggio, che però non è l'obiettivo che si è voluto raggiungere con la realizzazione di questo banco prova.

In conclusione, si può dire che il controllo sviluppato è stabile sotto certe assunzioni, ad esempio con l'utilizzo parziale del sistema di visione. Tuttavia, si è raggiunto il limite di prestazione superiore delle componenti utilizzate e non risulta possibile ottenere di meglio senza modificare il banco prova. Alcune di queste modifiche possono riguardare:

- Utilizzo di un dispositivo di acquisizione immagini con framerate più elevato.
- Utilizzo di un PC con hardware dedicato e più performante per l'elaborazione delle immagini.
- Realizzazione di un sistema di controllo ad architettura distribuita, in modo da esentare il PC dedicato all'elaborazione delle immagini dall'esecuzione del task di controllo real-time.



(a) Velocità angolare dell'asta.



(b) Velocità del carrello.

**Figura 10.43:** Stima delle velocità del carrello e dell'asta effettuata dal sistema di visione durante il controllo in asservimento visivo

- Riduzione della dinamica del pendolo inverso, ad esempio modificando la massa e/o la posizione dell'appendice del pendolo.
- Aumento della corsa massima del carrello, in modo da rendere realizzabile l'oscillazione del carrello richiesta per la stabilizzazione del pendolo con il controllo visivo.

## 6 Conclusioni

In questo capitolo è stata presentata un'applicazione particolare delle tecniche di controllo *vision in the loop*. Questo ha permesso di verificare l'efficacia e validità delle tecniche di controllo analizzate approfondendo diversi aspetti:

- utilizzo del sistema di visione nell'anello di retroazione di uno schema a controllo dello stato;
- controllo di un sistema caratterizzato da un comportamento dinamico in alta frequenza;
- valutazione della capacità del sistema di visione di sostituire altri tipi di sensori.

Dal punto di vista del controllo, in questo caso la chiusura dell'anello di retroazione ha dimostrato i limiti dell'utilizzo del dispositivo di acquisizione immagine e del relativo calcolatore utilizzati: il pendolo inverso opera a frequenze superiori rispetto a quelle utilizzabili da parte del sistema di visione. Pertanto, è stato necessario utilizzare la telecamera solamente per una sostituzione parziale dei sensori di posizione del pendolo. I risultati della simulazione, tuttavia, mostrano come sia possibile ottenere prestazioni migliori riducendo solo leggermente il tempo d'elaborazione delle immagini.

Nelle prove sperimentali si è provato a sostituire la telecamera senza cambiare i parametri del controllore. Questo ha permesso di mettere in luce le problematiche relative all'inserimento del sistema di visione nell'anello di controllo principale, ma non rappresenta certamente la scelta di sintesi del regolatore migliore. Un possibile sviluppo della parte relativa alla taratura del controllo potrebbe riguardare, quindi, l'ottimizzazione dei parametri del regolatore per adattarsi meglio alle frequenze messe a disposizione dalla componente più lenta del sistema, ovvero il sistema di visione.

Per quanto riguarda l'algoritmo di tracciamento del pendolo, questo ha dato risultati più che sufficienti. Inoltre, si ritiene di aver già raggiunto la massima ottimizzazione dell'algoritmo possibile senza cambiare l'approccio scelto. Di conseguenza, per diminuire il tempo d'esecuzione necessario all'elaborazione delle immagini e, di conseguenza, ottenere le prestazioni auspiccate dal controllore, è necessario operare un cambiamento a livello dell'architettura hardware del sistema di controllo. In particolar modo, uno sviluppo possibile potrebbe riguardare la separazione dei compiti di controllo real-time ed elaborazione immagini su due calcolatori differenti, in modo da massimizzare le prestazioni di entrambi i processi.

# Capitolo 11

## Conclusioni e Sviluppi Futuri

### Indice

---

1	Risultati principali . . . . .	<b>497</b>
2	Tematiche aperte . . . . .	<b>500</b>
3	Sviluppi futuri . . . . .	<b>501</b>

---

Questo lavoro di tesi è incentrato sulle tematiche relative all'utilizzo dei sistemi di visione in applicazioni industriali. In particolare, sono stati seguiti due diversi filoni di ricerca.

Il primo settore d'indagine ha riguardato l'utilizzo del sistema di visione nel modo classico, ovvero come un sensore più flessibile di altri, in cui l'informazione estratta dalle immagini viene passata al controllore del sistema meccanico senza incidere minimamente sul calcolo della legge di moto del sistema. Questo insieme di attività ha portato allo sviluppo di conoscenze spendibili per lo sviluppo di applicazioni industriali vere e proprie.

Al contrario, l'altro ambito di ricerca ha fatto riferimento all'analisi dettagliata delle tematiche relative allo sviluppo di leggi di controllo col sistema di visione inserito direttamente nell'anello di retroazione del controllore. Questo tipo di controllo è noto in letteratura col termine di *asservimento visivo* (*visual servoing*, *vision in the loop*) ed è molto spesso affiancato dalla problematica del *controllo multirate*, causato dall'introduzione di un sistema a dinamica lenta (in questo caso il sistema di visione) nel controllo.

Le attività di ricerca vera e propria hanno riguardato le problematiche da affrontare nello sviluppo di sistemi di controllo in asservimento visivo, poiché essi non sono ancora particolarmente diffusi nell'ambito industriale e la loro analisi permette di ottenere spunti

di ricerca più interessanti. La difficoltà principale incontrata nel percorso del dottorato di ricerca e nella stesura di questa tesi ha riguardato la grande complessità del problema. Infatti, nella progettazione di un tale sistema di controllo è necessario indagare diverse aree tematiche, afferenti solitamente a discipline differenti: dalla meccanica all'ingegneria del software, per arrivare alla sintesi dei controllori e al trattamento dei segnali.

A causa di questa commistione di discipline, la letteratura reperibile sul tema dell'asservimento visivo è molto vasta. Tuttavia, quasi la totalità dei riferimenti affronta l'argomento solo da un punto di vista specifico e raramente il problema è affrontato nella sua totale complessità. In particolar modo, si possono individuare tre macroaree di ricerca sull'asservimento visivo:

- *Elaborazione delle immagini;*
- *Sintesi dei controllori;*
- Progettazione ed implementazione di *architetture hardware/software* per il completamento di task di posizionamento e/o inseguimento basato su immagini.

Nell'affrontare questo problema, l'approccio classico è quindi costituito da separare nettamente le tematiche appena presentate. In tal senso, questo lavoro di tesi si sarebbe dovuto concentrare solamente su una delle problematiche proposte. Al contrario, l'approccio della *meccatronica* alla soluzione di problemi complessi prevede l'analisi e l'integrazione di tutte le singole componenti del problema. Pertanto, questo lavoro di tesi offre una panoramica completa su tutte le tematiche dell'asservimento visivo.

Il capitolo 2 si occupa di indagare i meccanismi che regolano l'acquisizione e la formazione delle immagini, mentre il capitolo 3 presenta un estratto dei classici algoritmi di elaborazione delle immagini. Già in questi due capitoli si riesce ad intravedere il lavoro svolto. La tematica dell'elaborazione delle immagini è talmente complessa che affrontarla nella sua totalità avrebbe richiesto un libro intero. Pertanto, ci si è concentrati sul fornire le basi necessarie ad affrontare il problema, presentando gli algoritmi più utili allo scopo prefissato: arrivare a sviluppare un task di asservimento visivo completo. Per finire la parte dedicata alla visione, il capitolo 4 rappresenta un'estensione dei classici algoritmi di elaborazione delle immagini. Tale estensione è resa possibile dai miglioramenti dei dispositivi atti all'acquisizione delle immagini, che ad oggi permettono di ottenere informazioni tridimensionali complete delle scene inquadrare.

Una volta presentate le tematiche relative all'estrazione delle informazioni dalle immagini, col capitolo 5 si entra nel vivo del controllo in asservimento visivo, presentando le diverse

configurazioni realizzabili e le basi del controllo stesso. In seguito, dato che il controllo richiede l'analisi di scene in movimento, nel capitolo 6 si è reso necessario analizzare le modalità e le tecniche attraverso cui è possibile definire ed analizzare tale movimento. Infine, per chiudere la parte relativa alla sintesi dei controllori, il capitolo 7 presenta un'analisi dettagliata delle componenti dinamiche coinvolte nel controllo.

La parte finale della tesi riguarda invece la realizzazione di applicazioni atte alla validazione dell'attività di ricerca svolta. Innanzitutto, nel capitolo 8 viene affrontato il problema dal punto di vista della realizzazione del software di controllo. Anche in questo caso il problema è stato affrontato in modo generale, non presentando una particolare implementazione, ma fornendo gli spunti per la soluzione dei principali problemi da affrontare nello sviluppo di un qualsiasi sistema di controllo in asservimento visivo. Per finire sono state presentate due applicazioni sviluppate appositamente per validare i risultati presentati. La prima applicazione, descritta nel capitolo 9 è un classico task di posizionamento assistito da un sistema di visione, dove il sistema da controllare ha un solo grado di libertà, in modo da permettere l'evidenziazione delle tematiche di controllo. La seconda applicazione e il capitolo 10 mostrano invece come lo studio delle classiche tematiche legate all'asservimento visivo possa essere applicato allo sviluppo di controllo di sistemi che poco hanno a che fare con i classici task di posizionamento di manipolatori robotici.

## 1 Risultati principali

Tramite questo lavoro di tesi, si è cercato di rispondere alla domanda: *quanto è difficile integrare un sistema di visione nel controllo di un generico sistema meccanico? E cosa comporta questo compito?* L'avanzamento della tecnologia a disposizione rende sempre più fattibile l'integrazione di sistemi di visione in svariati tipi di applicazioni. Inoltre, man mano che si prendeva confidenza con i dispositivi d'acquisizione immagine e si sviluppavano i primi sistemi di visione, sono state comprese del tutto le potenzialità di questi tipi di sensore. Infatti nella domanda si è parlato di un *generico* sistema meccanico poiché, a differenza di altri tipi di sensori, una telecamera permette l'assolvimento di diverse tipologie di compiti in svariati tipi di configurazioni. In realtà, il tutto non è poi così semplice come sembra. Infatti, i principali risultati ottenuti possono essere suddivisi in diverse categorie:

- Analisi delle caratteristiche comuni a tutte le problematiche di controllo assistito da sistemi di visione;

- Individuazione dei vantaggi e delle limitazioni di tali sistemi;
- Realizzazione e utilizzazione di strumenti atti allo studio, progettazione ed implementazioni di sistemi di controllo in asservimento visivo.

Dal punto di vista dell'elaborazione immagini, l'obiettivo è sempre lo stesso: determinare la presenza dell'oggetto d'interesse nell'immagine e, in seguito, fornire una stima della posa dell'oggetto rispetto al sistema meccanico da controllare. Tuttavia, se il problema è ben noto, la soluzione da applicare per il raggiungimento dell'obiettivo prefissato non è nota. Infatti, si suole dire che la visione artificiale è ancora un'arte più che una scienza ingegneristica. Mentre il trattamento e filtraggio del segnale delle immagini si basa su una forte formulazione matematica, la giusta integrazione degli algoritmi per arrivare al risultato desiderato si basa ancora sull'esperienza e su un approccio del tipo *trial and error*. Inoltre, il tutto è complicato dall'elevato numero di algoritmi di elaborazione immagini noti in letteratura. In questo senso, la tesi riesce a scremare questo numero di possibilità, valutando un numero limitato di soluzioni applicabili a differenti tipi di situazioni. Inoltre, svariati riferimenti in letteratura trattano gli algoritmi di elaborazione immagine solo dal punto di vista della loro funzionalità, ovvero del tipo di problema che risolvono e come lo risolvono, senza però preoccuparsi della scalabilità dell'algoritmo stesso. Infatti, quando ci si trova a che fare con il problema dell'identificazione del movimento della scena, l'elaborazione delle immagini non deve solamente essere robusta, ma fornire risultati in tempi limitati. A questo scopo le soluzioni possibili sono due: (i) aumentare la capacità computazionale dell'hardware oppure (ii) modificare la sequenza di algoritmi in modo da implementare una soluzione computazionalmente meno onerosa. In questo lavoro di tesi si è preferito mostrare algoritmi di elaborazione delle immagini poco onerosi, in modo che essi siano scalabili su differenti tipi di architetture.

Dal punto di vista delle tecniche di controllo in asservimento visivo, il principale contributo del lavoro di tesi consiste nel fornire le basi per l'implementazione del controllo nelle diverse configurazioni disponibili. In particolare, in letteratura vengono spesso trattate in dettaglio solamente le tematiche relative alla *cinematica* del controllo, senza analizzare in dettaglio la *dinamica* del sistema da controllare. La problematica principale è relativa al *controllo multirate*, in cui una componente del sistema presenta caratteristiche dinamiche di gran lunga inferiori a quelle degli altri componenti. Nel caso dell'asservimento visivo è la telecamera ad introdurre la componente multirate. Anche in questo caso l'approccio è simile a quanto già visto per gli algoritmi di elaborazione delle immagini:



si progetta il controllo considerando il sistema ideale, se in seguito l'implementazione del controllore reale pone limiti ulteriori, o il problema risiede nell'hardware scelto, oppure si riducono le prestazioni dinamiche fino ad arrivare nuovamente alla stabilità. In questo caso, quindi, non viene effettuata una vera e propria analisi di questi limiti, ma ci si limita a verificarne la presenza. Al contrario, altre pubblicazioni puntano in maniera spinta sull'analisi del controllo multirate, proponendo approcci e tecniche dedicate alla sintesi di controllori complessi. Tuttavia, questi controllori risultano a volte troppo complessi, soprattutto in relazione alle capacità di elaborazione delle immagini fornita dal sistema di visione. L'approccio seguito in questo lavoro di tesi ha portato alla modellazione ed analisi del problema del multirate utilizzando le classiche tecniche date dai fondamenti del controllo automatico. In questo modo, si ha la certezza che tutti i controllori progettati siano effettivamente implementabili e realizzabili, mantenendo al contempo una profondità d'analisi sufficiente per la sintesi del controllore e lo studio delle prestazioni dinamiche del sistema. In particolare, si sono individuate le principali componenti del sistema di controllo, ovvero manipolatore, sistema di visione e controllore del sistema, ponendo l'attenzione maggiore sulla dinamica del sistema di visione, le cui prestazioni dinamiche sono quelle meno analizzate in letteratura. Alla fine del lavoro, si è arrivati alla definizione di alcuni schemi di controllo tipici e indici prestazionali da utilizzare per la valutazione dell'efficienza del controllo.

Infine, lo sviluppo delle applicazioni utilizzate per la validazione dei risultati ha permesso di dimostrare che l'approccio seguito nella trattazione delle diverse tematiche è compatibile e favorisce lo sviluppo di applicazioni di controllo in asservimento visivo. In particolare, gli strumenti di simulazione sviluppati per l'analisi della dinamica del sistema di controllo sono stati applicati alla progettazione e sintesi di controllori di sistemi reali. Allo stesso modo, sono state fornite delle considerazioni sulla progettazione di alcuni componenti software critici nell'implementazione di ogni sistema di controllo in asservimento visivo. Anche queste assunzioni sono sfociate nell'implementazione dell'architettura software di controllo e la corretta esecuzione dei task ne dimostra la validità. Infine, è necessario fornire un cenno particolare all'applicazione del controllo del pendolo inverso. Quest'ultima, infatti, a differenza dell'applicazione di posizionamento del disco, si differenzia dal classico ambito d'applicazione del visual servoing. Di fatto, essa non appartiene del tutto all'ambito del classico controllo in asservimento visivo, ma costituisce l'ennesima conferma della versatilità dell'utilizzo di sensori di visione per le più svariate applicazioni. Inoltre, il controllo del pendolo tramite visione prevede l'inserimento della visione nell'anello di

retroazione del controllo del pendolo e, pertanto, può benissimo essere classificato come problema di *vision in the loop*.

## 2 Tematiche aperte

Nonostante l'ampiezza della trattazione e del lavoro prodotti, non si può negare che si sia toccata solamente la scorza esterna della problematica del controllo in asservimento visivo. Infatti, la tesi pone ulteriori domande e rimangono ancora delle tematiche aperte su tutti i fronti di ricerca associabili al problema generale dell'asservimento visivo.

Dal punto di vista dell'elaborazione delle immagini, l'algoritmo di tracking principalmente presentato nel lavoro di tesi deve essere irrobustito e ottimizzato per il trattamento di oggetti fortemente tridimensionali. A questo scopo, si ritiene di essere arrivati al limite di prestazioni raggiungibili tramite l'elaborazione delle classiche immagini bidimensionali. Pertanto, è necessario utilizzare i dispositivi di acquisizione di immagini tridimensionali e le tecniche di elaborazione annesse per estendere l'algoritmo alla gestione di casi più complessi.

Inoltre, sempre dal punto di vista dell'elaborazione delle immagini, rimane da testare l'algoritmo in presenza di oggetti reali e non solo su figure costruite "ad hoc" per la realizzazione di una piattaforma sperimentale.

Per quanto concerne le tecniche di controllo in asservimento vero e proprio, la ricerca dovrebbe estendere la trattazione del controllo basato sulle feature delle immagini (*Image Based Visual Servoing*). Infatti, tale tipo di controllo apre una serie di tematiche sull'influenza dello Jacobiano immagine nella generazione delle traiettorie da seguire. Nell'applicazione relativa al posizionamento della piattaforma rotante si è avuto un piccolo assaggio di questa problematica: col controllo IBVS il disco partiva "in ritardo", nel senso che lo Jacobiano immagine ha bisogno di iniziare a mettere in rotazione il disco per rendersi conto della traiettoria che minimizza lo spostamento richiesto dal task di posizionamento. Inoltre, in sistemi a più gradi di libertà, il controllo IBVS porta all'insorgenza di traiettorie che possono avvicinarsi a punti di singolarità del sistema controllato. Per evitare il verificarsi di queste situazioni è quindi necessario studiare e mettere a punto ulteriori tecniche che permettano un'analisi più approfondita dello Jacobiano immagine. Questi approcci sono solitamente basati sul disaccoppiamento delle varie componenti dello Jacobiano, ad esempio delle componenti traslazionali da quelle rotazionali, in modo da ottenere un controllo maggiore sulle traiettorie generate.

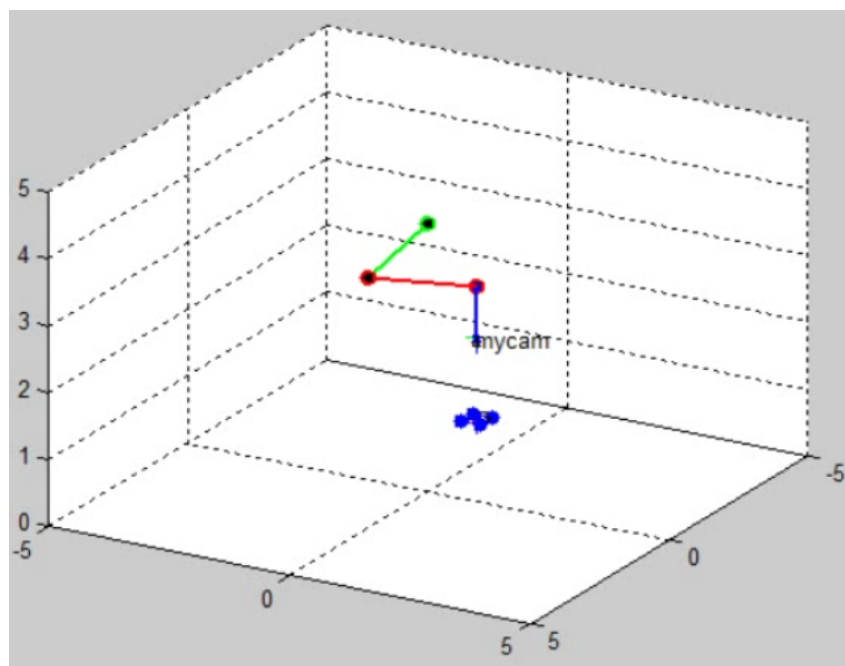
Per finire, dal punto di vista dell'architettura software, entrambe le applicazioni sono state sviluppate in un'ottica "pc-based", con un singolo calcolatore dedicato all'assolvimento di tutti i compiti, dall'elaborazione delle immagini alla generazione del controllo. Rimangono pertanto aperte le tematiche di sviluppo di un software distribuito su più calcolatori. Come si è visto nel corso della tesi, un'architettura di questo tipo permette una separazione ulteriore dei compiti da assolvere per il compimento del task di asservimento. Tuttavia, l'implementazione necessita un approfondimento ulteriore sui meccanismi di comunicazione tra le differenti applicazioni, poiché la comunicazione dei dati su rete introduce un ulteriore ritardo nel controllo.

Infine, un'ultima tematica aperta relativa all'implementazione software del sistema di controllo riguarda la scalabilità delle applicazioni sviluppate. In questo senso, si potrebbe valutare la possibilità di portare le applicazioni sviluppate su PC per questo lavoro di tesi su piattaforme diverse, ad esempio su PC embedded.

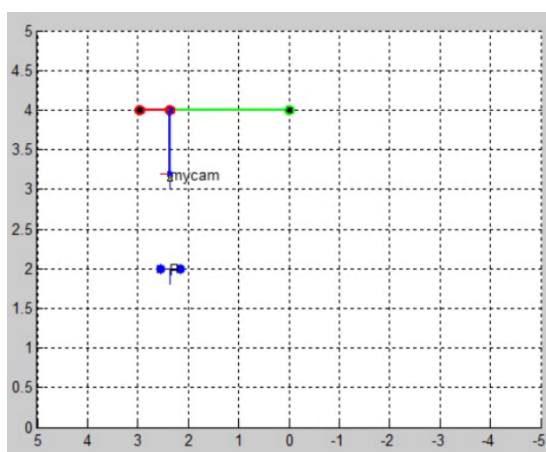
### 3 Sviluppi futuri

Gli sviluppi futuri di questo lavoro di tesi prevedono essenzialmente l'estensione dell'analisi al controllo di sistemi meccanici e robot a più gradi di libertà. Infatti, nel corso del lavoro non sono stati trattati esclusivamente sistemi ad un solo grado di libertà per diversi motivi. Innanzitutto, per l'analisi dell'efficacia e delle prestazioni del controllo in asservimento visivo un sistema ad un solo grado di libertà si presta meglio poiché il movimento della scena si ripercuote su un solo attuatore. Pertanto, è stato possibile individuare una relazione diretta tra i limiti del sistema di visione e la loro influenza sui limiti complessivi del sistema controllato. In secondo luogo, al momento della stesura di questa tesi, nel *Laboratorio di Meccatronica e Sistemi Dinamici* dell'università degli studi di Bergamo non è ancora disponibile un manipolatore completamente funzionante da dedicare esclusivamente alla valutazione delle prestazioni di un sistema di controllo in asservimento visivo. Infatti, il tipico approccio seguito in altri lavori simili prevede l'utilizzo di un sistema di visione e di un manipolatore preconfezionati, mentre il lavoro originale si incentra sull'integrazione dei due sistemi stand-alone. Tuttavia, questo non è l'approccio che si intende seguire nella meccatronica, in cui l'integrazione tra i due principali componenti del sistema di controllo deve essere più profonda e costruita durante tutte le fasi di progettazione del sistema automatico.

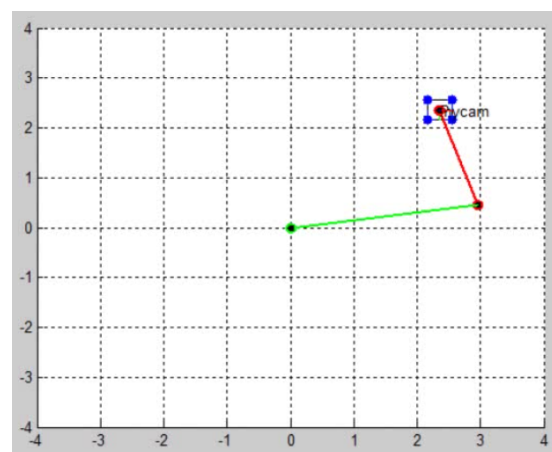
Ad ogni modo, la valutazione dell'applicabilità del lavoro svolto a sistemi a più gradi di libertà è già stato approcciato. Il primo caso che si intende studiare in futuro è relativo all'esecuzione di task di posizionamento precisi da parte di un *robot SCARA*, da applicare nella fase finale di posizionamento per correggere gli errori di posizionamento dovuti alle piccole differenze tra il sistema utilizzato per la sintesi cinematica e dinamica e il comportamento del sistema reale. In figura 11.1 è mostrata la ricostruzione del sistema utilizzato per la fase di simulazione.



(a) Vista 3D.



(b) Vista laterale.

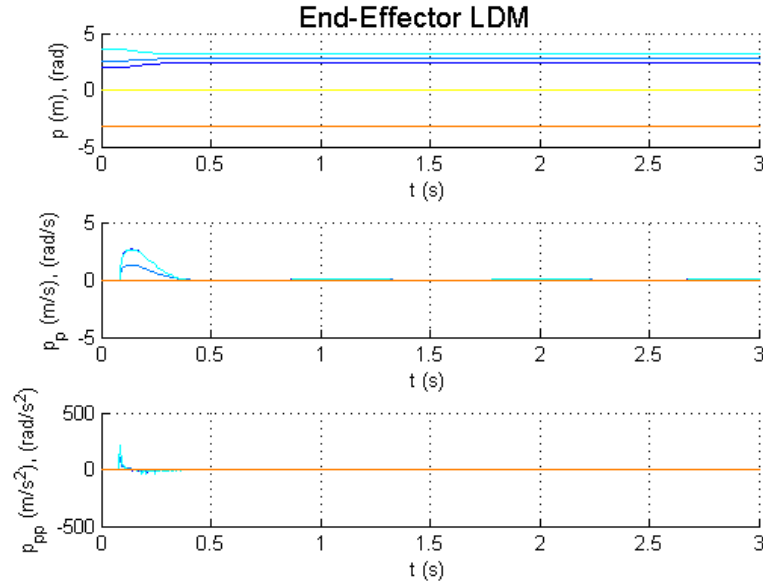


(c) Vista dall'alto.

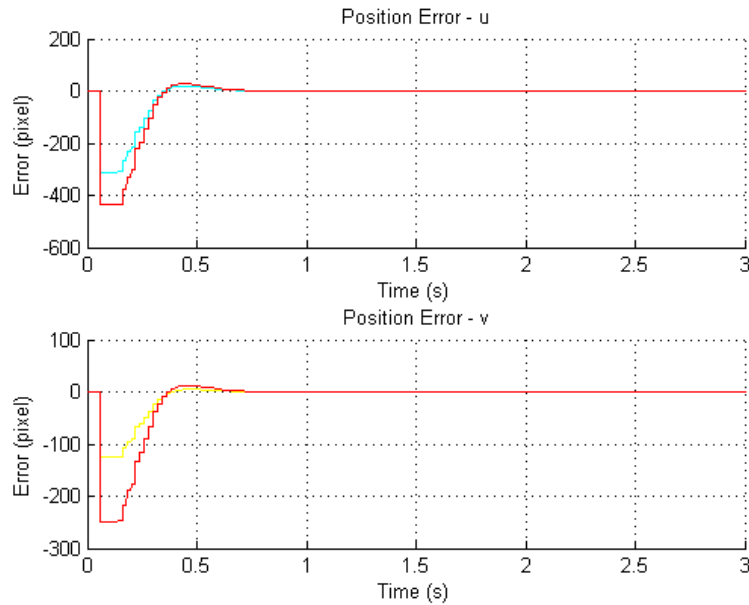
**Figura 11.1:** Simulazione di un robot SCARA

I primi movimenti che si intendono studiare sono quelli che non comprendono le rotazioni dell'end-effector, ma permettono lo spostamento lungo le tre direzioni dello

spazio, ovvero lungo gli assi  $X$ ,  $Y$  e  $Z$  del sistema di riferimento del robot. In figura 11.2 e 11.3 vengono riportati, a titolo esemplificativo, l'esecuzione di una traiettoria lungo tutte e tre le direzioni considerate (le linee costanti sono relative ai restanti 3 gradi di libertà) e il relativo errore osservato nel piano immagine.



**Figura 11.2:** Legge di moto ottenuta per il task di posizionamento del robot SCARA



**Figura 11.3:** Errore nel piano immagine per il task di posizionamento del robot SCARA



# Appendice A

## Metodo dei minimi quadrati

Il metodo dei *minimi quadrati ordinari* (*OLS*, *Ordinary Least Squares*) è una tecnica di ottimizzazione che permette di trovare una funzione, detta *curva di regressione*, che si avvicini il più possibile ad un insieme di dati (tipicamente punti del piano). In particolare la funzione trovata deve essere quella che minimizza la somma dei quadrati delle distanze tra i dati osservati e quelli della curva che rappresenta la funzione stessa. Questo metodo va distinto da quelli per l'interpolazione, dove si richiede che la curva relativa alla funzione contenga i punti dati.

Siano  $(x_i, y_i)$  con  $i = 1, 2, \dots, n$  i punti rappresentanti i dati in ingresso. Si vuole trovare una funzione  $f$  che approssimi la successione di punti data. Questa può essere determinata minimizzando la distanza (euclidea) tra le due successioni  $y_i$  e  $f(x_i)$ , ovvero la quantità

$$S = \sum_{i=1}^n (y_i - f(x_i))^2 \quad (\text{A.1})$$

da cui il nome *minimi quadrati*. Nei casi pratici in genere  $f(x)$  è definita in modo parametrico, così che il problema si riduce a determinare i parametri che minimizzano la distanza dei punti dalla curva. Naturalmente per ottenere un'unica curva ottimizzata e non un fascio, è necessario un numero di punti sperimentali maggiore del numero di parametri da cui dipende la curva in modo da poter infine ridurre il problema alla soluzione di un sistema sovradeterminato.

Nella formulazione dei *minimi quadrati lineari*  $f(x)$  è una funzione lineare rispetto ai parametri

$$f(x) = p_1 f_1(x) + p_2 f_2(x) + \dots + p_k f_k(x) \quad (\text{A.2})$$

dove  $p_i$  sono  $k$  parametri da utilizzare per descrivere la distribuzione di  $n$  punti noti ( $k < n$ ). Si può quindi riorganizzare il problema attraverso il sistema lineare sovradimensionato

$$\mathbf{A}p = y \quad (\text{A.3})$$

dove

$$\mathbf{A} = \begin{bmatrix} f_1(x_1) & \dots & f_k(x_1) \\ \vdots & \dots & \vdots \\ f_1(x_n) & \dots & f_k(x_n) \end{bmatrix}, \quad p = \begin{bmatrix} p_1 \\ \vdots \\ p_k \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Il problema di minimizzare la funzione  $S$  in equazione (A.1) si riconduce dunque a minimizzare la norma del residuo

$$\begin{aligned} \|r\| &= \|\mathbf{A}p - y\| \\ \|r\|^2 &= \|\mathbf{A}p - y\|^2 = ([\mathbf{A}p]_1 - y_1)^2 + \dots + ([\mathbf{A}p]_n - y_n)^2 = S \end{aligned}$$

dove con  $[\mathbf{A}p]_i$  si intende l' $i$ -sima componente del vettore prodotto tra  $\mathbf{A}$  e  $p$ . Si può minimizzare  $\|r\|$  derivando  $\|r\|^2$  rispetto a ciascun punto  $p$  ed eguagliando a 0 la derivata, ovvero

$$\frac{d\|r\|^2}{dp_m} = \sum_{i=1}^n 2 \left( \sum_{j=1}^k a_{ij} p_j - y_i \right) a_{im} = 0 \quad (\text{A.4})$$

equivalente al sistema

$$\begin{aligned} (\mathbf{A}p - y)^T \mathbf{A} &= 0 \\ \mathbf{A}^T \mathbf{A} p &= \mathbf{A}^T y \end{aligned} \quad (\text{A.5})$$

Quest'ultima equazione è detta *equazione normale*. Se il rango di  $\mathbf{A}$  è completo, allora  $\mathbf{A}^T \mathbf{A}$  è invertibile e dunque la soluzione è data da:

$$p = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T y. \quad (\text{A.6})$$

La matrice  $(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$  è detta *matrice pseudoinversa*.

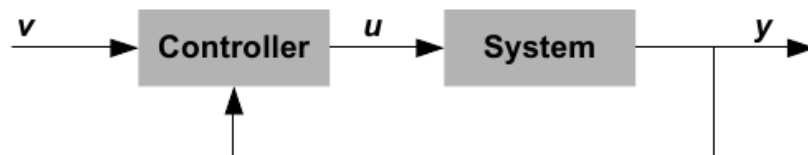


## Appendice B

### Assegnamento degli autovalori

Il metodo di sintesi del controllo per *assegnamento degli autovalori* è una tecnica di sintesi nello spazio di stato perché permette di tarare i guadagni del regolatore senza avere a disposizione la funzione di trasferimento del sistema, ma solamente il suo modello nel dominio del tempo. In particolare, l'approccio mira a far sì che il sistema retroazionato abbia un comportamento dinamico stabilito, assegnando ad esso dei poli in posizione ben definite.

Gli schemi di controllo più usati prevedono la sola retroazione dell'uscita, come raffigurato in figura B.1.

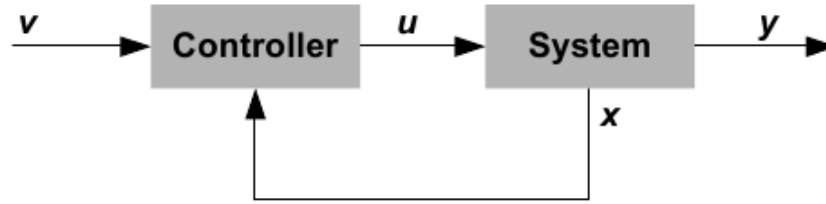


**Figura B.1:** Schema di controllo con retroazione dell'uscita

Il regolatore genera il controllo del sistema a partire dal riferimento  $v$  e dalla lettura dell'uscita  $y$ . Anche per questo schema di controllo sono disponibili delle tecniche che permettano la valutazione della posizione dei poli del sistema retroazionato. Un tipico esempio è il luogo delle radici in cui la posizione dei poli varia al variare del guadagno d'anello.

Il motivo per il quale lo schema di controllo a retroazione dell'uscita è il più utilizzato risiede nel fatto che spesso l'unico segnale letto è proprio l'uscita del sistema controllato. Se, invece, il regolatore disponesse di un'informazione completa sulle variabili di stato del processo, come mostrato in figura B.2, si potrebbe utilizzare anche tale informazione per progettare il regolatore in modo che la funzione di trasferimento in anello chiuso presenti

i poli nelle posizioni desiderate. Questo è l'approccio del metodo di *assegnamento degli autovalori* (o *posizionamento dei poli*).



**Figura B.2:** Schema di controllo con retroazione dello stato

Si consideri il sistema di ordine  $n$  descritto dall'equazione di stato

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \quad (\text{B.1})$$

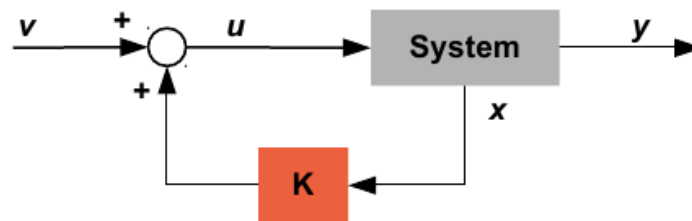
dove si suppone che l'ingresso  $u$  sia scalare. La legge di controllo che si vuole utilizzare è

$$u(t) = \mathbf{K}x(t) + v(t) \quad (\text{B.2})$$

in cui  $v(t)$  è il segnale di riferimento del regolatore e, per le ipotesi fatte,

$$\mathbf{K} = [k_0 \quad k_1 \quad \dots \quad k_{n-1}] \quad (\text{B.3})$$

è un vettore riga, detto *matrice dei guadagni*. Lo schema di controllo risultante è raffigurato in figura B.3.



**Figura B.3:** Schema per l'assegnamento degli autovalori con stato misurabile

Sostituendo la legge di controllo (B.2) nell'equazione di stato (B.1) si ottiene l'equazione di stato estesa

$$\dot{x}(t) = (\mathbf{A} + \mathbf{BK})x(t) + \mathbf{B}v(t) = \mathbf{F}x(t) + \mathbf{B}v(t) \quad (\text{B.4})$$

avendo definito  $\mathbf{F} = \mathbf{A} + \mathbf{BK}$ .

Il problema dell'assegnamento degli autovalori si riduce alla scelta di una matrice di guadagni  $\mathbf{K}$  tale per cui la matrice  $\mathbf{F}$ , rappresentante la dinamica del sistema retroazionato,

abbia autovalori arbitrari. In questo senso, condizione necessaria e sufficiente per cui tale matrice  $\mathbf{K}$  esista e faccia in modo che gli autovalori di  $\mathbf{F}$  siano influenzati solamente da essa è che la coppia  $(\mathbf{A}, \mathbf{B})$  sia completamente raggiungibile. Infatti, se una parte del sistema non fosse raggiungibile, i suoi autovalori non potrebbero essere influenzati in nessun modo da una qualunque retroazione che agisca su  $u$ .

Si supponga che il sistema (B.1) sia completamente raggiungibile e sia in *forma canonica di raggiungibilità*, ovvero che le matrici  $\mathbf{A}$  e  $\mathbf{B}$  abbiano la seguente forma:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -\alpha_0 & -\alpha_1 & -\alpha_2 & \dots & -\alpha_{n-1} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \quad (\text{B.5})$$

In tal caso, il polinomio caratteristico della matrice  $\mathbf{A}$  è dato da:

$$\phi(s) = s^n + \alpha_{n-1}s^{n-1} + \dots + \alpha_1s + \alpha_0. \quad (\text{B.6})$$

Inoltre, la matrice  $\mathbf{F} = \mathbf{A} + \mathbf{BK}$  conserva la struttura di  $\mathbf{A}$ . In base alle equazioni (B.3) e (B.5), risulta

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -\alpha_0 + k_0 & -\alpha_1 + k_1 & -\alpha_2 + k_2 & \dots & -\alpha_{n-1} + k_{n-1} \end{bmatrix}. \quad (\text{B.7})$$

Il polinomio caratteristico in anello chiuso è quindi

$$\phi_F(s) = s^n + (\alpha_{n-1} - k_{n-1})s^{n-1} + \dots + (\alpha_1 - k_1)s + (\alpha_0 - k_0). \quad (\text{B.8})$$

Da ciò consegue che assegnando gli  $n$  parametri liberi  $k_i$  si può rendere il polinomio (B.8) identico ad un qualunque polinomio preassegnato di grado  $n$  e, in altre parole, si possono assegnare arbitrariamente gli autovalori del sistema in anello chiuso.

Più precisamente, si indichino con  $\lambda_i^O$  gli autovalori che si vogliono assegnare e si costruisca il polinomio caratteristico desiderato

$$\begin{aligned} \phi^O(s) &= (s - \lambda_1^O)(s - \lambda_2^O) \dots (s - \lambda_n^O) \\ &= s^n + \beta_{n-1}s^{n-1} + \dots + \beta_1s + \beta_0. \end{aligned} \quad (\text{B.9})$$

Per fare in modo che  $\mathbf{F}$  abbia gli autovalori desiderati, è necessario porre

$$k_i = \alpha_i - \beta_i \quad , \quad i = 0, \dots, n-1$$

e, infine, la matrice dei guadagni che risolve il problema è data dalla (B.10).

$$\mathbf{K} = \begin{bmatrix} (\alpha_0 - \beta_0) & (\alpha_1 - \beta_1) & \dots & (\alpha_{n-1} - \beta_{n-1}) \end{bmatrix} . \quad (\text{B.10})$$

Il caso più generale è quello in cui la coppia  $(\mathbf{A}, \mathbf{B})$  è sì completamente raggiungibile, ma non è invece espressa nella forma canonica di raggiungibilità. Per poter applicare la soluzione (B.10), occorre innanzitutto applicare una trasformazione preliminare di variabili per porre il sistema in forma canonica. La soluzione consiste nel trovare una matrice  $\mathbf{T}$  tale che, operando le trasformazioni  $\hat{\mathbf{A}} = \mathbf{TAT}^{-1}$  e  $\hat{\mathbf{B}} = \mathbf{TB}$ , la coppia  $(\hat{\mathbf{A}}, \hat{\mathbf{B}})$  si presenti nella forma canonica di raggiungibilità espressa dalla (B.5).

Procedendo in questo modo, si arriva ad ottenere una soluzione generale al problema dell'assegnamento degli autovalori con stato completamente misurabile. La matrice dei guadagni è data da

$$\mathbf{K} = \hat{\mathbf{K}}\hat{\mathbf{M}}_{\mathbf{r}}\mathbf{M}_{\mathbf{r}}^{-1} \quad (\text{B.11})$$

dove

$$\begin{cases} \mathbf{M}_{\mathbf{r}} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \dots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix} \\ \hat{\mathbf{M}}_{\mathbf{r}} = \begin{bmatrix} \hat{\mathbf{B}} & \hat{\mathbf{A}}\hat{\mathbf{B}} & \dots & \hat{\mathbf{A}}^{n-1}\hat{\mathbf{B}} \end{bmatrix} \end{cases}$$

sono rispettivamente le matrici di raggiungibilità del sistema originario e del sistema in forma canonica. Infine, la soluzione è data da

$$\hat{\mathbf{K}} = \begin{bmatrix} (\alpha_0 - \beta_0) & (\alpha_1 - \beta_1) & \dots & (\alpha_{n-1} - \beta_{n-1}) \end{bmatrix} \quad (\text{B.12})$$

dove  $\alpha$  e  $\beta$  fanno riferimento alla forma canonica di raggiungibilità e relativo polinomio caratteristico della matrice  $\hat{\mathbf{A}}$ . Per concludere, si può notare come (i) la soluzione non preveda la determinazione esplicita della trasformazione  $\mathbf{T}$  necessaria per l'ottenimento della forma canonica di raggiungibilità del sistema e (ii) la matrice  $\mathbf{M}_{\mathbf{r}}$  risulti sempre invertibile poiché condizione necessaria e sufficiente per l'assegnamento degli autovalori è che il sistema da controllare sia completamente raggiungibile, il che garantisce la non singolarità della matrice  $\mathbf{M}_{\mathbf{r}}$ .

# Bibliografia

- [1] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlking, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation. *IEEE Robot. Automat. Mag.*, 19(3):80–91, 2012.
- [2] Daniel Bardsley and Li Bai. 3D Reconstruction Using the Direct Linear Transform with a Gabor Wavelet Based Correspondence Measure. Technical report.
- [3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110:346–359, 2008.
- [4] F Berry, P Martinet, J Gallice, Universite Blaise, and Pascal Clermont Ferrand. Real time visual servoing around a complex object. In *IEICE Transactions on Information and Systems, Special Issue on Machine Vision Applications, E83-D(7):1358–1368*, 2000.
- [5] Paolo Bolzern, Riccardo Scattolini, and Nicola Schiavoni. *Fondamenti di controlli automatici*. McGraw Hill, 2007.
- [6] Basilio Bona. *Metodi di Controllo Per Manipolatori Industriali*. 2009.
- [7] Dr. Gary Rost Bradski and Adrian Kaehler. *Learning OpenCV, 1st Edition*. O’Reilly Media, Inc., first edition, 2008.
- [8] AlistairJ. Bray and Vaclav Hlavac. Properties of Local Geometric Constraints. In Peter Mowforth, editor, *BMVC91*, pages 95–103. Springer London, 1991.
- [9] Herman Bruyninx. *Real-Time and Embedded Guide*. 2000.
- [10] François Chaumette and Seth Hutchinson. Visual servo control. I. Basic approaches. *IEEE Robotics and Automation Magazine*, 13:82–90, 2006.

- [11] François Chaumette and Seth Hutchinson. Visual servo control. II. Advanced approaches [Tutorial]. *IEEE Robotics and Automation Magazine*, 14:109–118, 2007.
- [12] Changhyun Choi and Henrik I Christensen. Real-time 3D model-based tracking using edge and keypoint features for robotic manipulation. In *ICRA*, pages 4048–4055. IEEE, 2010.
- [13] Changhyun Choi and Henrik I Christensen. 3D pose estimation of daily objects using an RGB-D camera. In *IROS*, pages 3342–3349. IEEE, 2012.
- [14] Alvaro Collet, Dmitry Berenson, Siddhartha S Srinivasa, and Dave Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *ICRA*, pages 48–55. IEEE, 2009.
- [15] Andrew I Comport, Éric Marchand, and François Chaumette. Robust model-based tracking for robot vision. In *IROS*, pages 692–697. IEEE, 2004.
- [16] P I Corke. *High-performance Visual Closed-loop Robot Control*. University of Melbourne, 1994.
- [17] Peter I Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.
- [18] Peter I. Corke and Malcolm C. Good. Dynamic effects in visual closed-loop systems. *IEEE Transactions on Robotics and Automation*, 12:671–683, 1996.
- [19] Dmitrij Csetverikov. Basic Algorithms for Digital Image Analysis.
- [20] Alun C Evans, Neil A Thacker, and John E W Mayhew. The use of Geometric Histograms for Model-Based Object Recognition. In *Proc 4th BMVC*, 1993.
- [21] M Bonfè D Fabbri and C Fantuzzi. Non-Linear PID Control for Robotic Manipulator Visual Servoing.
- [22] Hannes Fassold, Jakub Rosner, Peter Schallauer, and Werner Bailer. Realtime KLT feature point tracking for high definition video. *GraVisMa*, 2009.
- [23] Olivier Faugeras and F Lustman. Motion and structure from motion in a piecewise planar environment. Technical Report RR-0856, June 1988.

- [24] D Forsyth, J L Mundy, A Zisserman, C Coelho, A Heller, and C Rothwell. Invariant descriptors for 3D object recognition and pose. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(10):971–991, October 1991.
- [25] H Fujimoto. Visual servoing of 6 DOF manipulator by multirate control with depth identification. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 5, 2003.
- [26] H Fujimoto and Y Hori. Visual servoing based on multirate sampling control-application of perfect disturbance rejection control. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 711–716 vol.1, 2001.
- [27] Andrea Fusiello. *Visione Computazionale*. 2010.
- [28] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, February 1962.
- [29] Seth Hutchinson, Gregory D. Hager, and Peter I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12:651–670, 1996.
- [30] Johannes Kilian. Simple Image Analysis By Moments.
- [31] D Kragic. *Visual Servoing for Manipulation: Robustness and Integration Issues*. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden, 2001.
- [32] D Kragic and H I Christensen. Survey on visual servoing for manipulation. Technical report, Kungl Tekniska Hogskolan & Universitet Stockholms.
- [33] Danica Kragic and Henrik I Christensen. Robust Visual Servoing. *I. J. Robotic Res.*, 22(10-11):923–940, 2003.
- [34] Danica Kragic, Lars Petersson, and Henrik I Christensen. Visually guided manipulation tasks. *Robotics and Autonomous Systems*, 40(2-3):193–203, 2002.
- [35] M Pawan Kumar, Saurabh Goyal, Sujit Kuthirummal, C V Jawahar, and P J Narayanan. Discrete contours in multiple views: approximation and recognition. *IMAGE AND VISION COMPUTING*, 22, 2004.

- [36] Y Lamdan and H J Wolfson. Geometric Hashing: A General And Efficient Model-based Recognition Scheme. In *Computer Vision., Second International Conference on*, pages 238–249, December 1988.
- [37] Vincenzo Lippiello. *Multi-Object and Multi-Camera Robotic Visual Servoing*. PhD thesis, Scuola di Dottorato di ricerca in Ingegneria dell’Informazione, Università degli studi di Napoli Federico II, 2003.
- [38] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.
- [39] Università degli studi di Pavia Luca Lombardi. *Visione Artificiale*.
- [40] Andrea Luzzana, Mattia Rossetti, Paolo Righettini, and Patrizia Scandurra. Modeling synchronization / communication patterns in Vision-Based Robot Control applications using ASMs. In Elvinia Derrick, John and Fitzgerald, John and Gnesi, Stefania and Khurshid, Sarfraz and Leuschel, Michael and Reeves, Steve and Riccobene, editor, *Abstract State Machines, Alloy, B, VDM, and Z*, volume 7316, pages 331–335. Springer Berlin Heidelberg, 2012.
- [41] Andrea Luzzana, Mattia Rossetti, and Patrizia Scandurra. A Formal High-level Modeling Approach to Develop Reliable Components in Vision-based Robotics. In *ICSEA 2012, The Seventh International Conference on Software Engineering Advances*, pages 78–83, 2012.
- [42] Marcello Bonfè Elena Mainardi and Cesare Fantuzzi. Variable Structure PID Based Visual Servoing for Robotic Tracking and Manipulation. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robot and Systems EPFL, Lausanne, Switzerland*.
- [43] Ezio Malis, Manuel Vargas, and Thème Num. Deeper understanding of the homography decomposition for vision-based control. INRIA Research Report #6303, 2007.
- [44] E Marchand, F Spindler, and F Chaumette. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4):40–52, December 2005.



- [45] Éric Marchand and François Chaumette. Feature tracking for visual servoing purposes. *Robotics and Autonomous Systems*, 52(1):53–70, 2005.
- [46] Youcef Mezouar and Peter K Allen. Visual servoed micropositioning for protein manipulation tasks. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1766–1771. IEEE, 2002.
- [47] Mahadevamurty Nemani, Tsu-Chin Tsao, and Seth Hutchinson. Multi-Rate Analysis and Design of Visual Feedback Digital Servo-Control System. *Journal of Dynamic Systems, Measurement, and Control*, 116(1):45–55, March 1994.
- [48] Giorgio Panin. *Model-based visual tracking : the OpenTL framework*. Hoboken, N.J. Wiley, 2011.
- [49] Dipartimento di Ingegneria dell’Informazione dell’università di Parma Paolo Medici. Elementi di analisi per Visione Artificiale.
- [50] Nadia Payet and Sinisa Todorovic. From contours to 3D object detection and pose estimation. In Dimitris N Metaxas, Long Quan, Alberto Sanfeliu, and Luc J Van Gool, editors, *ICCV*, pages 983–990. IEEE, 2011.
- [51] J A Piepmeier, G V McMurray, and H Lipkin. A dynamic Jacobian estimation method for uncalibrated visual servoing. In *Advanced Intelligent Mechatronics, 1999. Proceedings. 1999 IEEE/ASME International Conference on*, pages 944–949, 1999.
- [52] M Pressigout and E Marchand. A model free hybrid algorithm for real time tracking. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–97–100, September 2005.
- [53] S Ralis, Barmeshwar Vikramaditya, and Bradley J Nelson. Visual Servoing Frameworks for Microassembly of Hybrid MEMS. In A Sulzmann and Bradley J Nelson, editors, *Microrobotics and Micromanipulation, Proceedings of SPIE Vol. 3519*, pages pp. 70 – 79, November 1998.
- [54] Aibing Rao, Rohini K Srihari, and Zhongfei Zhang. Geometric Histogram: A Distribution of Geometric Configurations of Color Subsets, 2000.
- [55] A. Sanderson and L. Weiss. Image-based visual servo control using relational graph error signals. In *Proc. IEEE*, pages 1074–1077, 1980.

- [56] K Sasajima and H Fujimoto. 6 DOF Multirate Visual Servoing for Quick Moving Objects. In *American Control Conference, 2007. ACC '07*, pages 1538–1543, 2007.
- [57] Robert D Schiftenbauer. A Survey of Aspect Graphs, 2001.
- [58] John Schulman, Alex Lee, Jonathan Ho, and Pieter Abbeel. Tracking deformable objects with point clouds. In *ICRA*, pages 1130–1137. IEEE, 2013.
- [59] J Shi and C Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, June 1994.
- [60] M Sitti, I Bozma, and A Denker. Visual tracking for moving multiple objects: an integration of vision and control. In *Industrial Electronics, 1995. ISIE '95., Proceedings of the IEEE International Symposium on*, volume 2, pages 535–540 vol.2, 1995.
- [61] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007.
- [62] Stefan May Kai Pervoelez, Hartmut Surmann Fraunhofer Institute for Intelligent Analysis, and Information Systems. 3D Cameras: 3D Computer Vision of wide Scope.
- [63] Geoffrey Taylor and Lindsay Kleeman. Fusion of Multimodal Visual Cues for Model-Based Object Tracking. In *In Australasian Conference on Robotics and Automation (ACRA2003), Brisbane, Australia*, 2003.
- [64] F Tombari and Università degli studi di Bologna S. Salti. 3D Computer Vision Course.
- [65] Tinne Tuytelaars and Krystian Mikolajczyk. *Local Invariant Feature Detectors: A Survey*. Now Publishers Inc., Hanover, MA, USA, 2008.
- [66] M Vargas, A R Malpesa, and F R Rubio. Modelling and control of a visual servoing system. *Syst. Anal. Model. Simul.*, 38(2):401–417, August 2000.
- [67] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object Tracking: A Survey. *ACM Comput. Surv.*, 38(4), December 2006.
- [68] Youngrock Yoon, Guilherme N DeSouza, and Avinash C Kak. Real-time tracking and pose estimation for industrial objects using geometric features. In *ICRA*, pages 3473–3478. IEEE, 2003.

# Sitografia

- [site1] Comedi, Linux Control and Measurement Device Interface – <http://www.comedi.org/>.
- [site2] IDS Ensenso – <http://en.ids-imaging.com/store/produkte/kameras/ensenso-n10-3d-usb-2-0.html>.
- [site3] Infranor – <http://www.infranor.com/>.
- [site4] Libdc1394 – <http://damien.douxchamps.net/ieee1394/libdc1394/>.
- [site5] Linux Realtime Suite – <http://www.rtaixml.net/realtime-suite>.
- [site6] Linux RTAI – <https://www.rtai.org/>.
- [site7] Linux RTAI-LXRT – [https://www.rtai.org/userfiles/documentation/magma/html/api/lxrt\\_faq.html](https://www.rtai.org/userfiles/documentation/magma/html/api/lxrt_faq.html).
- [site8] Mavilor Motors – <http://www.mavilor.es/>.
- [site9] National Instruments – <http://italy.ni.com/>.
- [site10] OpenCv – <http://opencv.org/>.
- [site11] Parker Motors – <http://www.parker.com>.
- [site12] Point Cloud Library – <http://pointclouds.org/>.
- [site13] Sony XCDV60cr – <http://www.sony.it/pro/product/industrial-cameras-ieee1394/xcd-v60cr/overview/>.
- [site14] The Imaging Source – [http://www.theimagingsource.com/it\\_IT/](http://www.theimagingsource.com/it_IT/).
- [site15] Visionlink – <http://visionlink.it/>.
- [site16] VISP, Visual Servoing Platform – <http://www.irisa.fr/lagadic/visp/visp.html>.

