UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA DELL'AUTOMAZIONE

A.A. 2008/2009

Tesi di Laurea

VISIONE ARTIFICIALE PER L'AUTOLOCALIZZAZIONE E LA RICOSTRUZIONE DI AMBIENTI

RELATORE

Ing. Francesco Martinelli

CANDIDATO

Federico Rea

Ai miei nonni

Indice

Introduzione

La computer vision è un campo in rapida crescita, sia grazie alla disponibilità di telecamere sempre più precise e meno costose, sia grazie alle elevate capacità di calcolo dei moderni computer. I livelli raggiunti sono anche il frutto dell'intensa attività di ricerca degli ultimi decenni, che dai primi anni Ottanta ad oggi ha portato ad una crescita esponenziale delle pubblicazioni scientifiche di settore. I campi applicativi della computer vision sono innumerevoli. La telecamera è infatti uno strumento molto flessibile, non invasivo, ed in grado di operare ad elevate distanze, se dotata dell'ottica opportuna. Il prezzo di questa flessibilità è la necessità di utilizzare di volta in volta algoritmi ad hoc, che in funzione delle applicazioni possono comportare costi computazionali molto alti. La versatilità è legata alla grande mole di informazioni che le immagini trasportano, ed alla possibilità di estrapolare solo quelle necessarie alla specifica applicazione. A differenza di un comune laser per misurazioni, infatti, le immagini immagazzinano dati provenienti da infinite direzioni, veicolati dai raggi di luce emessi o riflessi dai corpi. D'altro canto, la precisione dei risultati può essere fortemente condizionata da fattori che vanno dalla risoluzione della telecamera alla configurazione degli algoritmi. La computer vision trova sempre più spazio nell'automazione dei processi industriali, nel controllo di qualità, in applicazioni militari ed aerospaziali, nell'ingegneria edile e nell'architettura, nella sorveglianza e nel controllo del traffico. L'autolocalizzazione e la ricostruzione ambientale, oggetto della presente

tesi, sono le tematiche della computer vision che presentano maggiori analogie con le funzioni del nostro apparato visivo. Data una sequenza di fotogrammi di uno stesso ambiente, ricavati con continuità da una macchina fotografica o da una telecamera in movimento, l'algoritmo di autolocalizzazione riproduce la traiettoria del dispositivo nelle coordinate di un riferimento arbitrario. La ricostruzione ambientale prevede la generazione di un modello tridimensionale dell'ambiente, più o meno dettagliato anche in base ad esigenze di *real-time*. Il principio di funzionamento del sistema si basa, come la *stereopsi* nel mondo animale, sulla possibilità di fondere in una *mappa di profondità* le informazioni provenienti da due viste separate di una stessa scena. Le leggi matematiche che legano la posizione spaziale di un punto alla sua posizione in un'immagine, erano già note nella seconda metà del Settecento. Poco dopo la nascita della fotografia, nella prima metà dell'Ottocento, furono impiegate per dare vita al primo esempio di *fotogrammetria*. Tale tecnica, che consentiva di acquisire dati metrici partendo da una coppia di immagini, è stata il precursore degli attuali algoritmi di ricostruzione. Sebbene, in un immaginario collettivo che è il prodotto di decenni di filmografia fantascientifica, la navigazione di robot mobili in autonomia potrebbe ritenersi il più interessante seguito degli algoritmi di autolocalizzazione e ricostruzione, i loro attuali utilizzi riguardano principalmente la riproduzione di edifici, monumenti, e l'analisi della morfologia terrestre basata su riprese aeree. Si noti che, sebbene la ricostruzione sembri prevalere sull'autolocalizzazione in molte delle applicazioni proposte, i corrispondenti algoritmi sono strettamente interdipendenti, ed il funzionamento dell'uno non può prescindere dal funzionamento dell'altro. Le possibili applicazioni crescono nel caso in cui l'autolocalizzazione e la ricostruzione vengano fatte confluire nel campo della *realtà aumentata*. In cinematografia sono utilizzati software in grado di mostrare, durante le riprese, la collocazione di eventuali

oggetti e personaggi fittizi, frutto della *computer grafica*, come se fossero realmente presenti nella scena al fianco degli attori. Simili software sono in via di sviluppo nel campo medico, dove saranno utilizzati in chirurgia e riabilitazione, affiancati da opportuni *visori stereoscopici*.

La presente tesi è suddivisa in sei capitoli. Il Capitolo 1 introduce gli strumenti, hardware e software, utilizzati per la realizzazione di un sistema visivo per l'autolocalizzazione e la ricostruzione ambientale. Ci si sofferma in particolare sulle librerie OpenCV, le cui funzioni si sono rivelate preziose nella realizzazione del sistema, e sono richiamate nell'intera trattazione. Nel Capitolo 2 viene introdotto il concetto di elaborazione delle immagini a basso livello, e ne viene fornito un primo esempio applicativo attraverso il rilevamento dei punti di interesse. Viene poi mostrato come ottenerne un inseguimento preciso attraverso sequenze di immagini acquisite da telecamera, sfruttando l'algoritmo di Lucas-Kanade. La prima parte del Capitolo 3 è dedicata ai concetti fondamentali della geometria proiettiva, che vanno dalle coordinate omogenee, al loro impiego nel passaggio da uno spazio euclideo ad uno proiettivo, fino alle trasformazioni notevoli nello spazio proiettivo. Viene poi introdotto un modello matematico per la telecamera, ed analizzato un algoritmo di calibrazione largamente utilizzato per la stima dei parametri ad essa relativi. Si coglie l'occasione della calibrazione per descrivere due importanti algoritmi frequentemente utilizzati nel resto della trattazione. Nel Capitolo 4 si delinea un procedimento per ottenere, a meno di una deformazione lineare, la traiettoria della telecamera e la ricostruzione dell'ambiente. Il Capitolo 5 introduce gli strumenti necessari alla stima dei parametri della deformazione, consentendo di rettificare i risultati del Capitolo 4 e di raggiungere lo scopo della tesi. Il Capitolo 6 propone una rassegna di algoritmi che, applicati al sistema realizzato, potrebbero consentire di aumentare il dettaglio e la precisione dei

risultati.

Capitolo 1

Descrizione dell'hardware e del software utilizzati

1.1 La telecamera ed il supporto meccanico

Per la realizzazione del sistema di visione si è utilizzata la telecamera portatile Sony DCR-HC51, avente un sensore CCD di risoluzione 720×576 pixel. La telecamera è dotata di zoom ottico, inibito durante la sperimentazione, e di messa a fuoco automatica. È stato realizzato un semplice meccanismo per consentire, qualora necessario, un moto puramente traslatorio del dispositivo (figura 1.1). Si tratta di un pattino a base piatta, sul quale è fissata la telecamera, vincolato a muoversi lungo una guida cilindrica. Quando la struttura è collocata su di un piano, la base piatta del pattino ne inibisce la rotazione, e la telecamera non può che traslare. Nel caso sia necessario un moto generico della telecamera, lo si ottiene movimentando l'intera struttura, operazione abbastanza agevole data la sua leggerezza. La connessione della telecamera al calcolatore, un Pentium Dual-Core 2.50 GHz, è di tipo firewire. Tale connessione, a differenza della USB, non richiede l'installazione di driver specifici, a vantaggio di una maggiore compatibilità con i software e le librerie di sviluppo.

Figura 1.1: Telecamera e guida utilizzate nella realizzazione del sistema di visione.

1.2 Le librerie di sviluppo

Gli algoritmi utilizzati nell'autolocalizzazione e nella ricostruzione ambientale sono particolarmente onerosi dal punto di vista computazionale, aspetto che ha influito in modo sostanziale sulla scelta del linguaggio di programmazione. Sebbene *MATLAB* sia provvisto di numerose librerie, più o meno standard, legate alla computer vision, la sua natura di interprete di comandi lo rende inadatto agli scopi della presente tesi, essendo i programmi interpretati notoriamente lenti. La scelta è allora ricaduta sul compilatore *C* e su *OpenCV*, una libreria open source di funzioni rivolte alla computer vision e particolarmente indicata per applicazioni real-time. La prima release risale al gennaio del 1999, ed è stata seguita da numerosi aggiornamenti fino all'attuale versione *2.0*. Allo sviluppo di OpenCV hanno contribuito università, aziende, ed utenti appartenenti all'ampia community esistente sul web. OpenCV ha fornito

un importante impulso alla diffusione ed allo sviluppo delle tecniche della computer vision, facendo risparmiare agli sviluppatori il tempo necessario all'implementazione di algoritmi già consolidati. Il contratto di licenza fornito con OpenCV è particolarmente interessante. A differenza delle tipiche licenze fornite con le librerie open source, che impongono che il codice che ne fa uso sia sottoposto ad analoghe licenze, o non sia utilizzato per fini commerciali, la libreria OpenCV fornisce piena libertà allo sviluppatore. Una ragione è da ricercarsi nel fatto che le prestazioni di OpenCV, nata come progetto di Intel, possono essere notevolmente incrementate con l'acquisto delle librerie IPP (*Integrated Performance Primitives*), prodotte dalla stessa Intel per ottimizzare il funzionamento del codice sui propri processori. Riguardo il valore tecnico della libreria OpenCV, si rilevi il fatto che è stata un punto chiave nella realizzazione del sistema di visione del robot *Stanley*, prodotto dall'università di Stanford, che nel 2005 vinse la competizione *DARPA Grand Challenge* (figura 1.2), organizzata dal dipartimento della difesa americano per sollecitare lo sviluppo di tecniche innovative nel controllo dei veicoli senza pilota.

Nel presente progetto, la libreria OpenCV è stata inserita nell'ambiente di sviluppo *Microsoft Visual C++ 2008 Express Edition*, utilizzato con il sistema operativo *Windows XP*. A tal fine, è stato sufficiente aggiungere le directory opportune di OpenCV all'insieme delle cartelle di ricerca delle librerie, dei file di inclusione e dei file eseguibili, che si trova in:

```
Strumenti -> Opzioni -> Directory di VC++
```

Le librerie specifiche devono essere richiamate dalle proprietà del progetto, impostando il campo:

```
Proprieta -> Linker -> Input -> Dipendenze aggiuntive
```

Figura 1.2: Foto tratta dal DARPA Grand Challenge del 2006 (*Wikipedia Creative Commons*). I sistemi di visione di alcuni dei veicoli in competizione sono basati sulla libreria OpenCV.

Infine, i file sorgente che utilizzino le funzioni OpenCV, devono richiamare i file di inclusione opportuni.

OpenCV non si limita a fornire specifiche funzioni nell'ambito della computer vision. Lo sviluppatore viene supportato con strutture di dati ad hoc, funzioni per la gestione dei dispositivi di acquisizione video, per la gestione dei file, per la visualizzazione delle immagini, e funzioni matematiche (dal calcolo delle radici di un polinomio alla decomposizione in valori singolari). Una parte di OpenCV è invece dedicata al *Machine Learning*, un campo tecnicamente molto lontano dalla computer vision, ma che nella pratica vi è spesso affiancato. Unica nota dolente è probabilmente l'assenza di una documentazione esaustiva. Molto del materiale reperibile sul web è ripreso più o meno fedelmente dalla scarsa documentazione ufficiale. L'unico libro disponibile [1], tra i principali riferimenti bibliografici della presente tesi, raramente descrive nel dettaglio gli algoritmi utilizzati. Ciò rende a volte difficoltoso impostare gli argomenti delle

funzioni, e difficile valutare la possibilità di implementare algoritmi alternativi nel caso in cui i risultati non siano graditi. Gli identificatori delle funzioni OpenCV sono riconoscibili, nel resto della trattazione, per la presenza del prefisso *cv*. Le funzioni aggiuntive, implementate nell'ambito del lavoro di tesi, hanno identificatori preceduti dal prefisso *cvx* (*Computer Vision eXtended*).

Capitolo 2

Punti di interesse: rilevamento, matching e tracking

2.1 Image processing a basso livello

L'apparato visivo umano svolge un numero elevato di funzioni, come il riconoscimento di forme, la collocazione di corpi nello spazio, la localizzazione di sorgenti luminose. Dispositivi artificiali che, allo stato attuale della tecnica, vogliano imitare tali funzioni ad *alto livello*, devono solitamente seguire uno schema di tipo *bottom-up*. Tale schema prevede una prima estrapolazione di informazioni basata sull'elaborazione di proprietà associate ai *pixel*, gli elementi costitutivi dell'immagine nell'approccio informatico, seguita da ulteriori elaborazioni che gradualmente conducono ad informazioni più facilmente intepretabili dall'utente umano. La fase iniziale viene definita *low-level image processing*, argomento ampiamente trattato in [2]. Proprietà del pixel sono la posizione che occupa nell'immagine ed il colore associato, la cui codifica dipende dallo standard scelto. Lo spettro dei colori visibili può essere quasi interamente riprodotto attraverso la miscelazione additiva dei tre colori base *rosso*, *verde* e *blu*, abbreviati dall'acronimo *RGB*. Le *porpore* sono gli unici colori non riproducibili dalla terna RGB. Sebbene esistano altri standard basati su canali aggiuntivi e differenti colori di base,

la presente trattazione descrive la realizzazione di un sistema di visione basato su standard RGB. Nella sostanza lo standard viene abbandonato nella fase iniziale del programma, quando il flusso video viene convertito in *scala di grigi* per migliorare le prestazioni (lavorare su un singolo canale risulta meno oneroso dal punto di vista computazionale). I tre canali RGB presentano contenuti informativi molto simili in presenza di immagini *reali*[1]. Per tale ragione si è scelto di convertire il flusso video tramite selezione del canale R, ed esclusione dei due rimanenti. La funzione *cvSplit* svolge tale compito.

2.2 Rilevamento dei punti di interesse

Primo passo verso la realizzazione di un sistema visivo di ricostruzione è il rilevamento dei *punti di interesse*. Nella letteratura tecnica si utilizzano spesso in modo intercambiabile le diciture *punto di interesse*, *corner* o *feature*, sebbene abbiano significati sostanzialmente diversi. Un punto di interesse è un punto avente posizione rilevabile in modo robusto, cioè al variare dello stato dell'immagine e dei parametri dell'algoritmo entro un range sufficientemente grande. Il concetto di punto di interesse include non strettamente quello di corner, generato dall'incontro di due bordi dominanti ed aventi differenti direzioni. Il termine feature è il più generico, potendosi riferire ad un'ampia varietà di elementi, tra i quali punti, linee ed ellissi.

L'*algoritmo di Harris* ([3]) è probabilmente il più diffuso algoritmo per il rilevamento dei punti di interesse. Un'immagine in scala di grigi può essere interpretata come una funzione bidimensionale $I(x, y)$, che associa alle coordinate di un pixel l'intensità di grigio corrispondente. Anche le derivate parziali di $I(x, y)$, a prescindere

[1]Sono le immagini solitamente prodotte da dispositivi di acquisizione quali fotocamere o telecamere. Si contrappongono alle immagini *artificiali* o *sintetiche*, quali possono essere le *clipart*.

dall'ordine, sono funzioni bidimensionali delle coordinate x ed y, e possono viceversa essere interpretate come immagini. L'*immagine del gradiente* è l'insieme delle immagini associate alle derivate parziali prime $\partial I(x,y)/\partial x$ e $\partial I(x,y)/\partial y$. Analogamente, l'*immagine hessiana* è l'insieme delle immagini associate alle derivate parziali seconde $\partial I(x,y)/\partial x^2$, $\partial I(x,y)/\partial x \partial y$, $\partial I(x,y)/\partial y \partial x$ e $\partial I(x,y)/\partial y^2$, la cui *autocorrelazione* (dettagliatamente descritta in [2]) conduce alle immagini $R_{xx}(x,y)$, $R_{xy}(x,y)$, $R_{yx}(x,y)$ ed $R_{yy}(x,y)$. Si affianchino le autocorrelazioni in una matrice 2×2:

$$M(x,y) = \begin{bmatrix} R_{xx}(x,y) & R_{xy}(x,y) \\ R_{yx}(x,y) & R_{yy}(x,y) \end{bmatrix} \tag{2.2.1}$$

Nella definizione di Harris, i punti di interesse corrispondono ad autovalori sufficientemente grandi di $M(x,y)$. Nell'algoritmo gli autovalori non vengono realmente calcolati, per alleggerire il carico computazionale, e si utilizza una strategia quasi equivalente basata sul calcolo della traccia e del determinante di $M(x,y)$. Una variante dell'algoritmo è stata ricavata da Shi e Tomasi in [4], e si basa sulla constatazione che risultati validi, ed in alcuni casi migliori, si ottengono imponendo che il più piccolo degli autovalori sia maggiore di un'opportuna soglia. La funzione *cvGoodFeaturesToTrack* implementa entrambi gli algoritmi, ma nel progetto si è utilizzata la variante di Shi e Tomasi. La scelta dell'identificatore *cvGoodFeaturesToTrack* non è casuale. Si dimostra facilmente come ad autovalori non piccoli della matrice $M(x,y)$, corrispondano punti facilmente tracciabili con l'algoritmo di Lucas-Kanade, proposto nella sezione 2.3.

In molte applicazioni si richiede, nella conoscenza della posizione dei punti di interesse, un grado di accuratezza che le coordinate di un pixel non possono garantire. Per tale ragione sono state sviluppate tecniche che consentono di stimare in *sottopixel* la posizione di tali punti. Si può facilmente constatare come in generale sia nullo il

prodotto scalare tra un vettore applicato nel punto di interesse ed il gradiente calcolato nel secondo estremo del vettore. Questa osservazione, alla base di molte tecniche di *subpixel accuracy*, viene sfruttata nel generare un sistema lineare di equazioni, la cui risoluzione fornisce una stima migliore del punto di interesse. Si tratta esattamente del procedimento messo in atto dalla funzione *cvFindCornerSubPix* ([1]). Strategie più intuitive si basano sull'ingrandimento dell'immagine in prossimità dei punti di interesse, attraverso l'applicazione di algoritmi di interpolazione opportuni.

2.3 Matching dei punti di interesse

Gli algoritmi di *matching* elaborano due immagini della stessa scena al fine di rilevare corrispondenze tra punti *omologhi*[2], generando quella che viene definita *mappa di disparità*. Le corrispondenze possono essere utilizzate per scopi diversi, dalla *mosaicizzazione* al *tracking*. L'algoritmo di matching deve assecondare quelli che sono gli obiettivi del sistema visivo, dai quali dipende la tipologia delle immagini da elaborare. Nel caso della mosaicizzazione, ad esempio, ed in tutti i casi in cui le immagini siano ricavate da viste aventi baseline di lunghezze considerevoli, si utilizzano algoritmi specifici di *wide baseline matching*. Nel tracking le immagini sono invece ricavate da viste molto vicine, il che consente di trascurare la funzionalità *scale invariant* spesso inclusa negli algoritmi.

L'algoritmo di Lukas-Kanade, opportunamente implementato, rappresenta attualmente lo stato dell'arte tra gli algoritmi di matching. Date le immagini associate a due diverse viste di una scena, si ipotizzi di aver rilevato un punto di interesse, di coordinate $\mathbf{x_1}$, sulla prima immagine, e di volerne trovare l'omologo sulla seconda.

[2]Saranno definiti omologhi, nella presente trattazione, punti ottenuti proiettando uno stesso punto scena su differenti piani immagine.

L'algoritmo consiste nel centrare in $\mathbf{x_1}$ una finestra w arbitrariamente grande, per poi cercarne una versione deformata nella seconda immagine. Si rappresenti con $W(\mathbf{x};\mathbf{p})$ l'insieme delle possibili deformazioni. La funzione trasforma le coordinate in base al valore assunto dai parametri $\mathbf{p}$. La complessità di $W(\mathbf{x};\mathbf{p})$ e la dimensione di $\mathbf{p}$ sono scelti arbitrariamente, seppure compatibilmente con quelle che potrebbero essere le effettive deformazioni agenti sulle immagini. Il problema di ricerca si formalizza come un problema di minimizzazione:

$$\min_{\mathbf{p}} \sum_{\mathbf{x}\in w} [I(W(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2 \tag{2.3.1}$$

dove le funzioni bidimensionali $T(\mathbf{x})$ e $I(\mathbf{x})$ rappresentano le intensità di grigio della prima e della seconda immagine. Con un cambio di variabile, la (2.3.1) può essere riscritta nella forma:

$$\min_{\mathbf{\Delta p}} \sum_{\mathbf{x}\in w} [I(W(\mathbf{x};\mathbf{p_0}+\mathbf{\Delta p})) - T(\mathbf{x})]^2 \tag{2.3.2}$$

dove $\mathbf{p_0}$ è una stima iniziale di $\mathbf{p}$. Si utilizzi il metodo di Gauss-Newton per la minimizzazione della (2.3.2). Prima fase del metodo consiste nella linearizzazione di $I(W(\mathbf{x};\mathbf{p_0}+\mathbf{\Delta p})) - T(\mathbf{x})$ intorno a $\mathbf{\Delta p} = \mathbf{0}$:

$$I(W(\mathbf{x};\mathbf{p_0})) + \nabla I(\mathbf{x}) \frac{\partial W(\mathbf{x};\mathbf{p_0}+\mathbf{\Delta p})}{\partial \mathbf{\Delta p}}\Big|_{\mathbf{\Delta p}=\mathbf{0}} \mathbf{\Delta p} - T(\mathbf{x}) \tag{2.3.3}$$

La funzione viene inserita nella (2.3.2), ottenendo una quadrica minimizzabile analiticamente rispetto a $\mathbf{\Delta p}$. Ipotizzando che il minimo si trovi in $\mathbf{\Delta p} = \mathbf{\Delta p_0}$, la stima iniziale viene aggiornata additivamente nel modo seguente:

$$\mathbf{p_0} = \mathbf{p_0} + \mathbf{\Delta p_0} \tag{2.3.4}$$

La reiterazione dei passaggi porta l'algoritmo a convergenza. Ad ogni iterazione si rende necessaria la determinazione dell'Hessiana:

$$\sum_{\mathbf{x}\in w} \left[\nabla I(\mathbf{x}) \frac{\partial W(\mathbf{x};\mathbf{p_0}+\mathbf{\Delta p})}{\partial \mathbf{\Delta p}}\Big|_{\mathbf{\Delta p}=\mathbf{0}}\right]^T \left[\nabla I(\mathbf{x}) \frac{\partial W(\mathbf{x};\mathbf{p_0}+\mathbf{\Delta p})}{\partial \mathbf{\Delta p}}\Big|_{\mathbf{\Delta p}=\mathbf{0}}\right] \tag{2.3.5}$$

Tale fatto risulta particolarmente oneroso dal punto di vista computazionale. Il problema può essere risolto considerando forme alternative alla (2.3.2), e varianti dell'algoritmo di Gauss-Newton. Sebbene non intuitiva, si dimostra l'equivalenza tra la minimizzazione (2.3.2) eseguita attraverso l'algoritmo di Gauss-Newton, e la minimizzazione:

$$\min_{\mathbf{\Delta p}} \sum_{\mathbf{x}\in w} [I(W(W(\mathbf{x}; \mathbf{\Delta p}); \mathbf{p_0})) - T(\mathbf{x})]^2 \tag{2.3.6}$$

eseguita attraverso un algoritmo che differisce nella sola legge di aggiornamento:

$$W(\mathbf{x}; \mathbf{p_0}) \circ W(\mathbf{x}; \mathbf{\Delta p_0}) \rightarrow W(\mathbf{x}; \mathbf{p_0}) \tag{2.3.7}$$

dove la composizione $\circ$ si ottiene calcolando $W(W(\mathbf{x}; \mathbf{\Delta p_0}); \mathbf{p_0})$. Per distinguere i due algoritmi si parla rispettivamente di *diretto additivo* e *diretto composizionale*. Si osservi che la (2.3.7) non esprime in modo esplicito l'aggiornamento di $\mathbf{p_0}$, che dipende dalla struttura di $W(\mathbf{x}; \mathbf{p})$. Ove tale esplicitazione non fosse possibile, verrebbe meno l'utilizzabilità dell'algoritmo diretto composizionale. Affinché l'agoritmo possa convergere, $W(\mathbf{x}; \mathbf{p})$ deve inoltre garantire l'esistenza dell'elemento *neutro* rispetto alla composizione. L'insieme delle deformazioni $W(\mathbf{x}; \mathbf{p})$, insieme all'operazione di composizione, deve quindi formare un *semigruppo*. L'algoritmo diretto composizionale è computazionalmente paragonabile al diretto additivo. Il salto di qualità si ottiene grazie all'algoritmo *inverso composizionale*, ricavato in modo concettualmente analogo al precedente, cambiando la funzione da minimizzare e la legge di aggiornamento:

$$\min_{\mathbf{\Delta p}} \sum_{\mathbf{x}\in w} [T(W(\mathbf{x}; \mathbf{\Delta p})) - I(W(\mathbf{x}; \mathbf{p_0}))]^2 \tag{2.3.8}$$

$$W(\mathbf{x}; \mathbf{p_0}) \circ W(\mathbf{x}; \mathbf{\Delta p_0})^{-1} \rightarrow W(\mathbf{x}; \mathbf{p_0}) \tag{2.3.9}$$

Per ricavare $W(\mathbf{x}; \mathbf{\Delta p_0})^{-1}$ è sufficiente imporre:

$$W(\mathbf{x}; \mathbf{\Delta p_0}) \circ W(\mathbf{x}; \mathbf{\Delta p_0})^{-1} \rightarrow W(\mathbf{x}; \mathbf{0}) \tag{2.3.10}$$

ed applicare le regole della composizione. Le deformazioni $W(\mathbf{x};\mathbf{p})$ devono garantire l'esistenza dell'elemento *inverso* rispetto alla composizione, e quindi formare un *gruppo*. La maggior parte delle deformazioni di utilità pratica soddisfa questa ipotesi. Il fatto che l'Hessiana non dipenda dalla stima dei parametri, e possa essere calcolata una sola volta in fase di precomputazione, è il principale vantaggio introdotto dall'algoritmo inverso composizionale, di seguito esposto:

Precomputazioni ($\mathbf{p}$ è una stima nota dei parametri):

1) Calcolo del gradiente $\nabla T(\mathbf{x})$ su w

2) Valutazione della Jacobiana $\partial W(\mathbf{x};\mathbf{p})/\partial \mathbf{p}$ in $(\mathbf{x};\mathbf{0})$

3) Calcolo di $\nabla T(\mathbf{x}) \cdot \partial W(\mathbf{x};\mathbf{p})/\partial \mathbf{p}$ su w

4) Calcolo dell'Hessiana $H = \sum_x [\nabla T(\mathbf{x}) \cdot \partial W(\mathbf{x};\mathbf{p})/\partial \mathbf{p}]^T \cdot [\nabla T(\mathbf{x}) \cdot \partial W(\mathbf{x};\mathbf{p})/\partial \mathbf{p}]$

Iterazioni (fino a che $\|\Delta \mathbf{p}\| > \epsilon$) :

5) Calcolo di $I(W(\mathbf{x};\mathbf{p}))$ su w

6) Calcolo dell'errore $[I(W(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$ su w

7) Calcolo di $\sum_x [\nabla T(\mathbf{x}) \cdot \partial W(\mathbf{x};\mathbf{p})/\partial \mathbf{p}]^T \cdot [I(W(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$

8) Calcolo di $\Delta p = H^{-1} \sum_x [\nabla T(\mathbf{x}) \cdot \partial W(\mathbf{x};\mathbf{p})/\partial \mathbf{p}]^T \cdot [I(W(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]$

9) Aggiornamento della deformazione $W(\mathbf{x};\mathbf{p}) \circ W(\mathbf{x};\Delta \mathbf{p})^{-1} \rightarrow W(\mathbf{x};\mathbf{p})$

Nell'algoritmo riportato e negli algoritmi precedenti, la funzione $T(\mathbf{x})$, a differenza di $I(\mathbf{x})$, non è mai valutata fuori dalla finestra w. Questo aspetto ha rilevanza tecnica, in quanto consente di tralasciare i pixel che nella prima immagine non appartengono a w. Nel sistema visivo realizzato, l'algoritmo inverso composizionale lavora sull'insieme

delle deformazioni affini:

$$W(\mathbf{x};\mathbf{p}) = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \tag{2.3.11}$$

Questa scelta conduce a buoni risultati qualora sia costante l'intensità di colore dei punti nella scena, ed i punti di interesse siano collocati su superfici abbastanza regolari. In [5] sono descritte dettagliatamente molte varianti dell'algoritmo di Lucas-Kanade. Alcune si basano sull'utilizzo di metodi di minimizzazione diversi da quello di Gauss-Newton, altre sull'introduzione di accorgimenti per ottimizzarne le prestazioni in presenza di fattori quali variazioni lineari di luminosità e rumore gaussiano.
Per quanto concerne l'implementazione dell'algoritmo di Lucas-Kanade, si consideri che le prestazioni dell'intero sistema visivo ne sono fortemente condizionate. Ad ogni iterazione è eseguita una scansione lineare della finestra w, che generalmente contiene molte centinaia di pixel. L'ottimizzazione del codice è un passaggio dovuto, se si desidera che il sistema funzioni in tempi ragionevoli. Le modifiche apportate al codice, dopo una prima implementazione poco prestante, hanno riguardato la sostituzione di chiamate a funzione con il corrispondente codice *inline*, l'aggiunta di variabili per la memorizzazione di dati intermedi, l'utilizzo intensivo di operatori facilmente ottimizzabili dal compilatore.

2.4 Tracking dei punti di interesse

Gli algoritmi di matching rilevano corrispondenze su coppie di immagini, e si basano spesso su complesse considerazioni teoriche. Gli algoritmi di *tracking*, o *inseguimento*, forniscono strutture dati ed operazioni elementari che permettono di estendere il matching a lunghe sequenze video. Dietro il *tracking* si nascondono quindi difficoltà prettamente tecniche e realizzative. Il sistema implementato fonde due distinte tipolo-

gie di matching, entrambe basate sul metodo di Lucas-Kanade. La prima tipologia è inclusa nella funzione *cvCalcOpticalFlowPyrLK* (dettagliatamente descritta in [6]), in grado di rilevare traslazioni di punti anche grazie all'ausilio di tecniche di *sottocampionamento piramidale*. Quest'ultime consentono di allargare il range di traslazioni rilevate, a scapito della velocità dell'algoritmo. Nel sistema in esame tale opzione non è stata attivata non tanto per il deterioramento delle prestazioni, quanto per il fatto che le immagini acquisite da telecamera sono associate a viste molto vicine, e l'algoritmo piramidale introdurrebbe rischi (e pochi vantaggi) legati a possibili falsi positivi nel caso di pattern ripetitivi. La seconda tipologia di matching utilizzata è il metodo di Lucas-Kanade inverso composizionale agente su deformazioni affini, implementato nella funzione *cvxLKInvComp*. Un semplice *tracker* è ottenibile con l'ausilio della sola *cvCalcOpticalFlowPyrLK*, di volta in volta applicata ad immagini adiacenti. Potendo infatti rilevare, in assenza di sottocampionamento piramidale, *quasi-traslazioni* di poche decine di pixel, come constatato sperimentalmente, la vicinanza delle viste risulta necessaria al suo corretto funzionamento. Un tracker basato sulla sola funzione *cvCalcOpticalFlowPyrLK* sarebbe soggetto al cosiddetto fenomeno di *deriva* (o *drift*) dei punti, che consiste nel loro progressivo allontanamento dalla posizione attesa. Il sistema realizzato sfrutta la *cvxLKInvComp* per "agganciare" punti corrispondenti in immagini non adiacenti, risolvendo il problema della deriva. Sebbene la *cvxLKInvComp* possa rilevare un ampio insieme di deformazioni, il suo utilizzo in autonomia all'interno di un tracker non produrrebbe i risultati aspettati, in quanto per una corretta convergenza sono solitamente richieste distanze di pochi pixel tra i punti omologhi, non garantite dalle frequenze di acquisizione delle telecamere di largo consumo. Il sistema di tracking implementato si basa sulle strutture *CvxView* e *CvxTrack*:

```
typedef struct CvxView{
     IplImage* image;
     IplImage* image_x;
     IplImage* image_y;
}CvxView;

typedef struct CvxTrack{
     CvxView* head_view;
     CvPoint2D32f head;
}CvxTrack;
```

i cui campi non inerenti all'inseguimento sono stati omessi per chiarezza. I campi di *CvxView* sono tre puntatori ad immagini, la vera e propria immagine associata alla vista e le due derivate parziali nelle direzioni principali. Il primo campo di *CvxTrack* è un puntatore alla vista di origine della traccia, il secondo memorizza le coordinate del primo punto della traccia. Di seguito è riportato lo pseudocodice dell'algoritmo di tracking:

```
1. QueryFrame() -> views(1).image
2. InterestingPoints(views(1).image, MAXPOINTS, MINDIST, NULL) ->
2. -> tails, tailsCounter
3. for i = 1 : tails_counter
4.     views(1) -> tracks(i).headView
5.     tails(i) -> tracks(i).head
6.     i -> tracksIndexes(i)
7.     [0, 0, 0, 0] -> tailsWrtHeads(i, 1:4)
8. tailsCounter -> tracksCounter
```

```
9. for h = 1 : infinity
10.    QueryFrame() -> views(h).image
11.    LucasKanade2DOF(views(h-1).image, vews(h).image, tails) ->
11.    -> newTails, newTailsFound
12.    for i = 1 : tailsCounter
13.         if(!newTailsFound(i))
14.              NULL -> tracksIndexes(i)
15.    for i = 1 : tailsCounter
16.         if(newTailsFound(i))
17.              newTails(i) -> tails(i)
18.              tails(i)-tracks(tracksIndexes(i)).head ->
18.              -> tailsWrtHeads(i, 5:6)
19.              LucasKanade6DOF(tracks(tracksIndexes(i)).head,
19.              , tracks(tracksIndexes(i)).headView.image,
19.              , tailsWrtHead(i, 1:6)) -> tails(i), tailsWrtHead(i, 1:4)
20.              if(!tails(i))
21.                   NULL -> tracksIndexes(i)
22.    0 -> j
23.    for i = 1 : tailsCounter
24.         if(tracksIndexes(i))
25.              tailsWrtHeads(i, 1:4) -> tailsWrtHeads(j, 1:4)
26.              tracksIndexes(i) -> tracksIndexes(j)
27.              tails(i) -> tails(j)
28.              j++
29.    j -> tailsCounter
```

```
30.    empty -> mask
31.    for i = 1 : tailsCounter
32.         tails(i) -> mask
33.    InterestingPoints(MAXPOINTS-tailsCounter, MINDIST, mask) ->
33.    -> tails2, tailsCounter2
34.    for i = tailsCounter : tailsCounter+tailsCounter2
35.         tails2(i-tailsCounter) ->
35.         -> tracks(tracksCounter+i-tailsCounter).head
36.         views(h) -> tracks(tracksCounter+i-tailsCounter).headView
37.         [0, 0, 0, 0] -> tailsWrtHeads(i, 1:4)
38.         tracksCounter+i-tailsCounter -> tracksIndexes(i)
39.         tails2(i-tailsCounter) -> tails(i)
40.    tailsCounter+tailsCounter2 -> tailsCounter
41.    tracksCounter+tracksCounter2 -> tracksCounter
```

Nella riga 1, la prima immagine prelevata dal dispositivo di acquisizione è collocata nella prima struttura dell'array *views* di *CvxView*. I quattro argomenti di *InterestingPoints*, equivalente a *cvGoodFeaturesToTrack*, sono l'immagine, il massimo numero di punti di interesse da rilevare, la minima distanza tra i punti, ed una maschera che ne inibisce la ricerca in particolari regioni dell'immagine (impostato a *NULL* se non utilizzata). La funzione memorizza i punti ed il loro effettivo numero rispettivamente in *tails* e *tailsCounter*. L'array *tails* contiene in generale le coordinate di tutti i punti "attivi" nell'immagine della vista corrente, cioè di tutti quei punti in corso di inseguimento o appena rilevati come punti di interesse. I punti memorizzati in *tails* in questa prima fase rientrano esclusivamente nella seconda categoria. Le righe 3 - 7 inizializzano gli array *tracks*, *tracksIndexes* e *tailsWrtHeads*. Il primo ha

lo scopo di memorizzare tutte le tracce rilevate dal programma, intese come singoli inseguimenti di punti sulla scena, a prescindere dal fatto che l'inseguimento sia o meno ancora in corso. In *tracksIndexes* sono memorizzati gli indici delle tracce attive, ed in *tailsWrtHeads*, che è un array bidimensionale, le stime delle deformazioni che nelle tracce attive conducono dalla *testa* (*head*), nell'immagine di partenza, alla *coda* (*tail*), nell'immagine corrente. Questi valori sono le condizioni iniziali fornite all'algoritmo di Lucas-Kanade affine. In *tracksCounter*, inizializzato nella riga 8, è registrato il numero totale di tracce rilevate. Si tratta chiaramente di un valore non decrescente. Dalla riga 9 ha inizio il corpo del programma, con un ciclo iterativo ripetuto fino alla richiesta di interruzione dell'utente. Nella riga 10 una nuova immagine viene prelevata dal dispositivo di acquisizione, e poi confrontata con la precedente alla ricerca di corrispondenze per mezzo della funzione *LucasKanade2DOF*, l'algoritmo di Lucas-Kanade per deformazioni traslatorie, equivalente a *cvCalcOpticalFlowPyrLK*. Ovviamente la funzione riceve anche i punti *tails*, ai quali saranno fatti corrispondere in uscita i relativi *newTails*. Scelte tecniche impongono che gli array in ingresso ed in uscita abbiano uguali lunghezze, nonostante le corrispondenze possano non essere rilevate. Per tale ragione l'array di stato *newTailFound* specifica gli elementi validi di *newTails*. Nel caso in cui le code manchino le corrispondenze, le relative tracce vengono "annotate" (righe 12 - 14), per poi essere disattivate in un secondo momento. L'annotazione consiste nell'inserimento di un opportuno simbolo nell'array *tracksIndexes*, in sostituzione dell'indice di traccia. I meccanismi di attivazione e disattivazione delle tracce si basano su operazioni di *push* ed *eliminazione* di elementi dalle strutture *tracksIndexes*, *tails* e *tailsWrtHeads*. L'annotazione di un elemento di *tracksIndexes* impone, al momento opportuno (righe 22 - 29), la sua eliminazione e quella dei corrispondenti elementi (aventi stessa posizione) in *tails* e *tailsWrtHeads*.

L'aggiunta di tracce avviene gestendo gli array come dei comuni *stack* (righe 3 - 7 e 34 - 39). L'esecuzione di *lucasKanade2DOF* è seguita, per le corrispondenze rilevate, da quella di *LucasKanade6DOF*, equivalente a *cvxLKInvComp*. Delle sei stime di partenza dei parametri affini, da fornire alla funzione per ogni coppia di punti, quattro sono state generate o in fase di inizializzazione (riga 7), o nell'iterazione precedente. La funzione sovrascrive infatti tali valori, per riutilizzarli come stime iniziali nell'iterazione successiva. Le due stime mancanti provengono direttamente da *LucasKanade2DOF* (righe 17 - 18). Le righe 30 - 41 sono rivolte alla creazione di nuove tracce, seguendo i procedimenti già descritti ed aggiornando opportunamente i contatori.
Le figure 2.1 e 2.2 sono immagini prelevate da una sequenza di tracking. Le finestre utilizzate per l'inseguimento sono rappresentate in rosso.

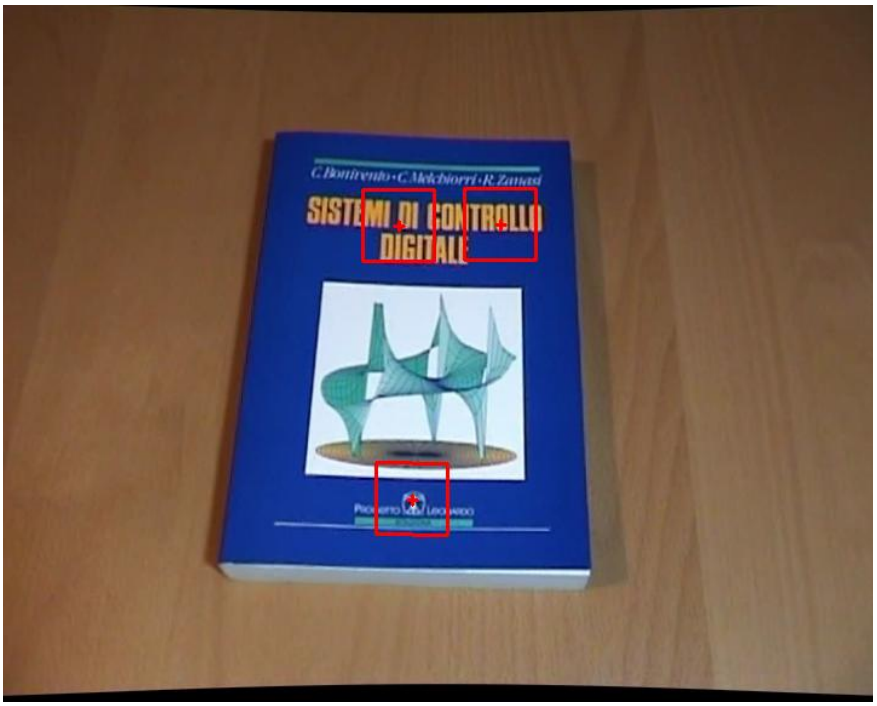

Figura 2.1: Prima immagine nella sequenza di tracking.

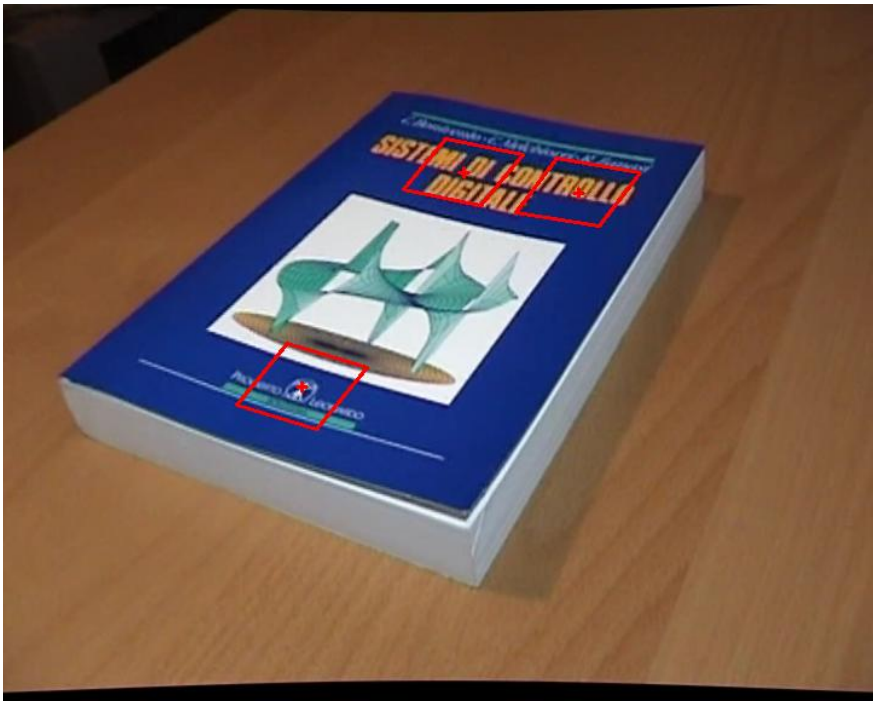

Figura 2.2: Ultima immagine nella sequenza di tracking.

Capitolo 3

Modello e calibrazione della telecamera

3.1 Geometria proiettiva

3.1.1 Spazio proiettivo e coordinate omogenee

Il calcolo dei limiti è un noto strumento per l'analisi asintotica di modelli matematici. La disparità di trattamento riservata ai *punti all'infinito* può in talune circostanze, e nella presente, essere messa da parte in favore di una trattazione più omogenea. Un primo passo in questa direzione è la definizione di uno *spazio proiettivo* $\mathbf{P}^n$ (si vedano [7], [8] e [9]), ottenuto aggiungendo i punti all'infinito allo *spazio euclideo* $\mathbf{R}^n$. Tale definizione avrebbe poca rilevanza pratica se non fosse sostenuta da opportuni strumenti matematici. L'introduzione di *coordinate omogenee* rappresenta la vera innovazione in tal senso. Si consideri il piano euclideo $\mathbf{R}^2$, e si utilizzino le coordinate $\tilde{\mathbf{x}} = (x_1, x_2)^T$, definite disomogenee, per rappresentarne un generico punto. Le corrispondenti coordinate omogenee sono $\mathbf{x} = (kx_1, kx_2, k)^T$ con $k \in R \setminus 0$. Il vincolo $k \neq 0$, garantisce che a coordinate disomogenee diverse corrispondano coordinate omogenee diverse, ma non solo. Per tornare alle coordinate di partenza, le prime due coordinate omogenee devono essere divise per la terza, in particolare

$\tilde{\mathbf{x}} = (kx_1/k, kx_2/k)^T = (x_1, x_2)^T$. Se k potesse valere 0, si avrebbe il rapporto indeterminato $0/0$. Si è visto come trasformare le coordinate di un punto al finito da disomogenee ad omogenee, e viceversa. I punti all'infinito, invece, nascono in coordinate omogenee, non esistendo modo di rappresentarli in coordinate disomogenee. In particolare sono caratterizzati dalla forma $\mathbf{x} = (kx_1, kx_2, 0)^T$ con $k \in R \setminus 0$. Anche in questo caso esistono infinite rappresentazioni omogenee associate allo stesso punto, che questa volta è all'infinito. Si noti come il tentativo di trasformare le coordinate in disomogenee conduca alla divisione per 0, ad ulteriore conferma del fatto che non esiste rappresentazione disomogenea di punti all'infinito. Si consideri ora una retta nel piano euclideo, avente equazione $ax + by + c = 0$. In essa compaiono esclusivamente coordinate disomogenee, che non consentono di "testare" l'appartenenza alla retta di un punto espresso in coordinate omogenee. Nel caso si tratti di un punto al finito, è sufficiente convertirne le coordinate e sostituirle nell'equazione. I punti all'infinito, per quanto visto, non godono dello stesso privilegio, e non possono appartenere alla retta. Sebbene sia controintuitivo immaginare che la retta non contenga punti all'infinito, si ricordi che lo stesso vale per l'intero piano euclideo. L'estensione dell'equazione allo spazio proiettivo avviene nel modo seguente:

$$ax + by + c = 0 \rightarrow a\frac{x_1}{x_3} + b\frac{x_2}{x_3} + c = 0 \rightarrow ax_1 + bx_2 + cx_3 = 0 \tag{3.1.1}$$

in cui $(x_1, x_2, x_3)^T$ sono le coordinate omogenee del punto, e le frecce rappresentano una sequenza costruttiva, e non implicazione matematica. In un primo passaggio l'equazione viene allargata ai punti al finito (la divisione per x_3 impone che questa sia diversa da 0) in coordinate omogenee. Nel secondo passaggio vengono inclusi anche i punti all'infinito, moltiplicando la seconda equazione per x_3. Si parte quindi dall'idea di delineare una curva nello spazio proiettivo che contenga gli stessi punti al finito

della curva corrispondente nello spazio euclideo, per poi constatare come in modo del tutto gratuito si siano aggiunti alla curva nuovi punti, tutti all'infinito. È importante osservare come i punti all'infinito siano scritti "nel DNA" dell'equazione di partenza, e non siano frutto di arbitrarietà. Il procedimento descritto può essere allargato a tutte le curve che nel piano siano espresse in forma implicita. Più in generale, tutti i concetti finora esposti possono essere estesi a spazi euclidei e proiettivi di dimensioni maggiori. In $\mathbf{P}^2$ i punti all'infinito formano una retta (la *retta all'infinito*), che malgrado la contraddizione in termini può essere immaginata come una circonferenza che avvolge il piano euclideo. Analogamente i punti all'infinito di $\mathbf{P}^3$ formano un piano (il *piano all'infinito*), immaginabile come una sfera che avvolge $\mathbf{R}^3$. Le coordinate omogenee rappresentano l'operando ideale per molte operazioni elementari nella computer vision e nella computer graphics. Nel piano proiettivo, l'appartenenza di un punto ad una retta può essere riscritta nella forma $(x_1, x_2, x_3) \cdot (a, b, c)^T$, cioè come prodotto scalare tra le coordinate omogenee del punto ed il vettore dei coefficienti della retta. Anche il vettore dei coefficienti risulta definito a meno di un fattore di scala diverso da 0, grazie all'*omogeneità* dell'equazione, termine utilizzato in questo caso per comunicare *l'assenza del termine noto* (anche nello spazio euclideo l'equazione della retta è omogenea in tal senso). Definiti due vettori dei coefficienti per due distinte rette, il loro prodotto vettoriale fornisce il punto di intersezione delle rette espresso in coordinate omogenee. Nel caso le rette siano parallele, l'ultima coordinata del punto risulta nulla, poiché il punto è all'infinito. Per il principio di *dualità* punto-retta, la retta passante per due punti si ottiene calcolando il prodotto vettoriale delle loro coordinate. Nello spazio proiettivo, analoga dualità ed operazioni concettualmente identiche si ripetono su punti e piani. Le relazioni di appartenenza e di intersezione sono gli elementi costitutivi di operazioni più complesse utilizzate in modo intensivo

nel seguito della trattazione. Ciò giustifica l'importanza attribuita alle coordinate omogenee.

3.1.2 Trasformazioni lineari nello spazio proiettivo

Siano $\mathbf{x}$ le coordinate di un punto in $\mathbf{P}^n$, e sia H una matrice invertibile in $\mathbf{R}^{(n+1)\times(n+1)}$. La trasformazione $H\mathbf{x} \rightarrow \mathbf{x}'$ rappresenta la più generica trasformazione lineare ed invertibile del punto nello spazio proiettivo, definita *trasformazione proiettiva*, *proiettività* od *omografia*. Nel caso di $\mathbf{P}^2$ si parla anche di *collinearità*, trattandosi di una trasformazione invertibile tale da preservare l'allineamento tra i punti. Nello spazio euclideo la relazione $=$ di *uguaglianza tra coordinate* esprime anche l'*uguaglianza tra punti*. Nello spazio proiettivo, visto l'impiego di coordinate omogenee, questa equivalenza non sussiste, e si rende necessario introdurre una relazione $\simeq$ che esprima l'uguaglianza tra punti a prescindere dall'uguaglianza tra coordinate. Questo non significa che non possa essere applicata alle coordinate stesse. A tal fine si stabilisce che due vettori in relazione $\simeq$ sono uguali a meno di un fattore di scala reale e non nullo. Lo stesso discorso si ripete, con la stessa simbologia, per le trasformazioni lineari e le matrici associate. L'omogeneità delle matrici è una conseguenza diretta dell'omogeneità delle coordinate. Nello spazio $\mathbf{P}^2$ la matrice associata ad una generica omografia vanta nove gradi di libertà, tanti quanti sono i suoi elementi, ma l'omografia in sé ha soli otto gradi di libertà, poiché vi sono infinite matrici ad essa associate[1]. Imponendo dei vincoli sulla matrice, si ottengono corrispondenti traformazioni via via più specifiche: *affini*, *simili* ed *euclidee*. La diminuzione dei gradi di libertà induce un aumento delle proprietà *invarianti*, come la misura degli angoli tra rette nel caso delle *similarità*. In aggiunta alle trasformazioni elencate, proiettività di particolare

[1]Nel seguito della trattazione, in favore di una chiarezza non sempre presente in letteratura, si distinguerà tra gradi di libertà *effettivi* e gradi di libertà della *rappresentazione*.

interesse sono le *prospettività*, ottenute attraverso la *proiezione centrale* (introdotta nella sezione 3.2.2) di uno spazio proiettivo su un altro.

3.1.3 Trasformazione proiettiva di curve notevoli

L'equazione della conica, come l'equazione della retta, può essere generalizzata al piano proiettivo:

$$ax^2+bxy+cy^2+dx+ey+f=0 \rightarrow ax_1^2+bx_1x_2+cx_2^2+dx_1x_3+ex_2x_3+fx_3^2=0 \quad (3.1.2)$$

I coefficienti dell'equazione possono essere raggruppati nella matrice C, dando luogo alla più sintetica notazione $\mathbf{x}^T C\mathbf{x}=0$. Analogo procedimento si ripete per la quadrica, che in P^3 assume la forma $\mathbf{X}^T Q\mathbf{X}=0$. Si ipotizzi di applicare la trasformazione omografica H_2 al piano proiettivo contenente la retta $\mathbf{l}$ e la conica C. Le due curve si trasformano rispettivamente in $\mathbf{l}'=H_2^{-T}\mathbf{l}$ e $C'=H_2^{-T}CH_2^{-1}$. Applicando invece la trasformazione omografica H_3 allo spazio proiettivo contenente Q, si ottiene $Q'=H_3^{-T}QH_3^{-1}$.

3.2 Un modello per la telecamera

Le lenti impiegate nella costruzione delle telecamere moderne possono essere estremamente complesse. Si tratta solitamente di elementi ottici in vetro o in plastica trasparente, in grado di concentrare o divergere i raggi luminosi. Tale proprietà dipende dalla forma, dal materiale di costruzione e dal fluido in cui sono immerse. Sebbene nel prosieguo della trattazione si faccia esclusivo riferimento al modello *pinhole*[2], e tale modello non preveda l'impiego di lenti, la conoscenza qualitativa del meccanismo di formazione delle immagini attraverso lenti può essere di aiuto nel caso insorgano fenomeni visivi estranei al modello stesso (distorsioni, *blur*, etc.). Nella sezione 3.2.1

[2]Detto anche *stenopeico*, dal greco *stenos opaios*, che significa piccolo foro.

si analizzerà un semplice modello di telecamera basato su *lente sottile*, mentre nella sezione successiva (3.2.2) verrà introdotto il modello pinhole.

3.2.1 Telecamera a lente sottile

Le superfici opposte di una lente sono spesso porzioni di sfera (*lenti sferiche*), le cui curvature determinano la tipologia della lente stessa (lente *biconvessa*, *piano-convessa*, *biconcava*, etc.). Si parla di lente sottile qualora il suo spessore risulti sufficientemente piccolo rispetto ai raggi di curvatura. Il comportamento della lente sottile è completamente caratterizzato dalle due seguenti asserzioni ([10]):

1) Tutti i raggi paralleli all'asse ottico convergono alla distanza focale f.

2) I raggi che passano attraverso il centro della lente non vengono deflessi.

Dalle asserzioni si ricava l'*equazione della lente sottile*, valida per tutti i raggi:

$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f} \tag{3.2.1}$$

che si basa sulla notazione della figura 3.1. Una lettura informale della (3.2.1) è la seguente. Si consideri una sorgente di luce puntiforme collocata a distanza S_1 dal piano della lente. La sorgente produce raggi che si propagano, in assenza di *occlusioni*, in tutte le direzioni. Tra i raggi provenienti dalla sorgente, quelli che attraversano la lente vengono deflessi in modo da essere convogliati in un unico punto, la cui distanza S_2 dal piano della lente è funzione di f ed S_1. La mancata *messa a fuoco*, o *spherical blur*, è una diretta conseguenza della (3.2.1). La corretta messa a fuoco, infatti, si ottiene nel caso in cui la lente convogli i raggi in un punto collocato sul piano immagine, che si ricorda essere parallelo al piano della lente. Per poter mettere a fuoco sorgenti poste a distanze diverse, sarebbe inverosimilmente necessario collocare piani immagine a distanze diverse dalla lente. Per la (3.2.1), esiste un solo piano di sorgenti, parallelo al

piano della lente, che il sistema può mettere a fuoco contemporaneamente. Le sorgenti poste a profondità diverse vengono percepite *sfocate*, tanto più quanto maggiore è la distanza dal piano messo a fuoco.

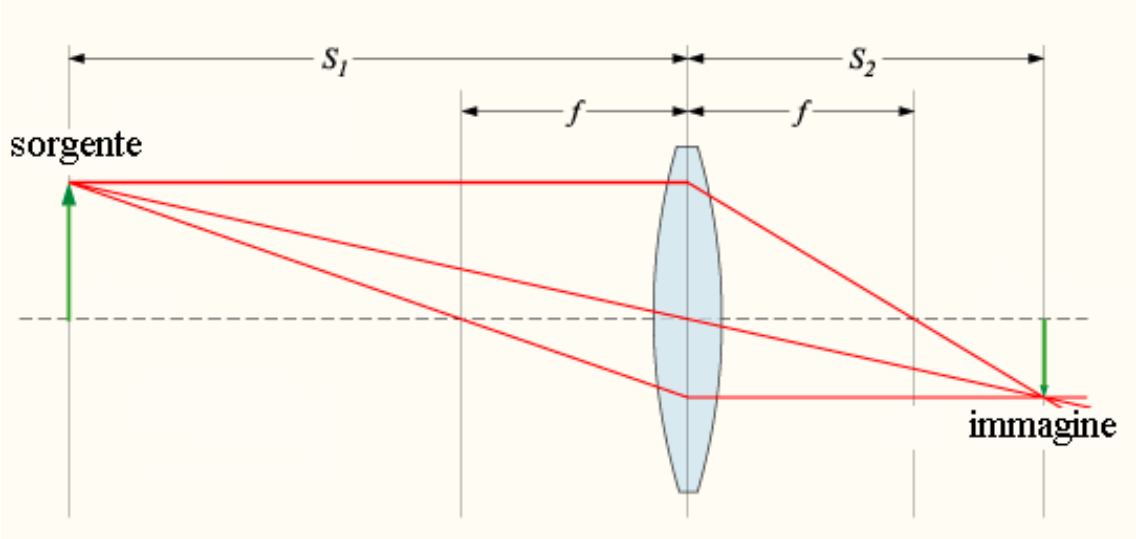

Figura 3.1: Funzionamento della lente sottile (*Wikipedia Creative Commons*).

3.2.2 Telecamera pinhole

Il modello pinhole si ottiene idealizzando ulteriormente il modello di telecamera a lente sottile. I raggi che attraversano il centro della lente sottile non vengono deflessi. Cosa accadrebbe se l'intera lente, ad eccezione del suo centro, fosse opacizzata rispetto ai raggi di luce? Gli unici raggi ad attraversarla sarebbero quelli passanti per il centro della lente, con deflessione nulla. Si osservi che un siffatto modello sarebbe identico, dal punto di vista funzionale, ad uno ottenuto forando una superficie opaca (verrebbe infatti meno il ruolo della lente). Quest'ultimo è definito modello pinhole (figura 3.2). Il solo fatto di avere eliminato la deflessione dei raggi apporta notevoli semplificazioni nella descrizione matematica del modello. Si consideri inoltre che esistendo una biunivocità tra i punti della scena ed i raggi passanti per il foro, non

risulta necessario introdurre criteri di selezione o mediazione tra raggi in competizione, per stabilirne gli effetti sul piano immagine. D'altro canto ciò impedisce di includere nel modello fenomeni come lo spherical blur, che saranno etichettati come *rumore* nel seguito della trattazione. Si noti che, sebbene la telecamera pinhole sia utilizzata in questo contesto come mero modello semplificativo di una telecamera moderna ben più complessa, i primi apparecchi fotografici si basavano esattamente sullo stesso principio. Tali apparecchi soffrivano lunghi tempi di esposizione (un solo raggio trasmette meno energia di un fascio di raggi), mentre traevano vantaggio dall'assenza di spherical blur. Un aspetto che nel passaggio dalla lente sottile al modello pinhole può creare non poca confusione, è la definizione che nei due casi di dà di *fuoco* e *distanza focale*. Nel primo caso si definisce fuoco l'intersezione tra l'asse ottico e la deflessione di un raggio ad esso parallelo. La distanza focale è la distanza tra il fuoco ed il piano della lente. Nel modello pinhole il fuoco coincide con il foro che consente il passaggio dei raggi. La distanza focale è la distanza del fuoco dal piano immagine.

Per una rappresentazione matematica del modello, si introducano i tre riferimenti di

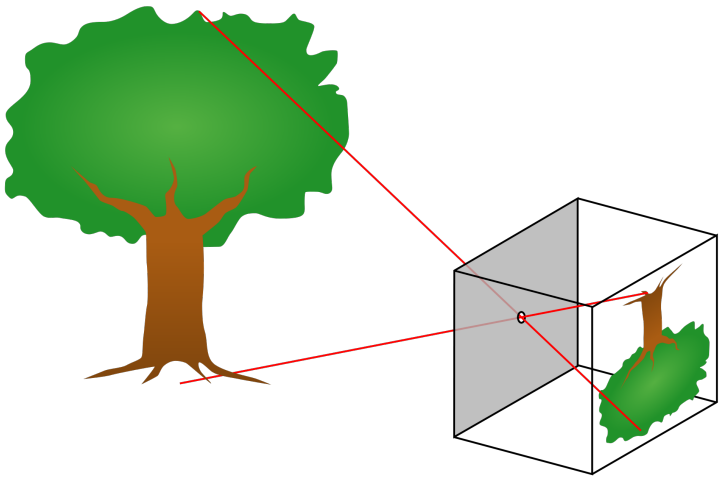

Figura 3.2: Funzionamento del modello pinhole (*Wikipedia Creative Commons*).

figura 3.3. Il riferimento $(\mathbf{X_{cam}}, \mathbf{Y_{cam}}, \mathbf{Z_{cam}})$ è definito *locale*, ha origine nel fuoco della *vista*, ed è disposto in modo tale che il sottoriferimento $(\mathbf{X_{cam}}, \mathbf{Y_{cam}})$ sia equiorientato rispetto al *riferimento del piano immagine* $(\mathbf{X_{im}}, \mathbf{Y_{im}})$ (la cui collocazione è nota in partenza e dipende dalla collocazione del dispositivo). L'allineamento delle origini dipende dall'allineamento tra il foro e l'elemento fotosensibile. Il *riferimento globale* $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ è scelto arbitrariamente nella scena, in base agli scopi della particolare applicazione.

Alla vista è associata una matrice $P \in \mathbf{R}^{3\times4}$, definita *matrice di proiezione*, il cui

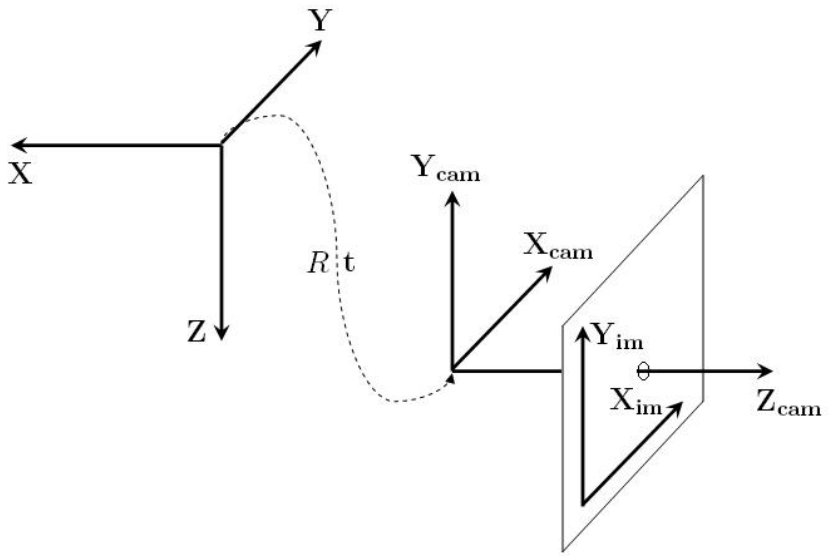

Figura 3.3: Rappresentazione dei riferimenti utilizzati.

scopo è quello di proiettare sul piano immagine, secondo il modello pinhole, punti appartenenti alla scena. Tutti i punti sono espressi in coordinate omogenee. La matrice può essere decomposta nel prodotto $K \left[\; R \,\middle|\, \mathbf{t} \; \right]$, in cui R e $\mathbf{t}$ rappresentano rispettivamente l'orientamento e la posizione del riferimento locale rispetto a quello

globale, e K la *matrice di calibrazione* (definita positiva):

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2.2}$$

I cinque parametri che compaiono nella (3.2.2) sono definiti *parametri intrinseci* della vista. In particolare α_x ed α_y sono fattori di scala agenti sulle direzioni principali, x_0 ed y_0 (coordinate del *punto principale*) dipendono dall'allineamento tra il riferimento locale ed il riferimento dell'immagine, ed s quantifica l'eventuale inclinazione delle coordinate (solitamente prossima a 0). Quali informazioni pratiche porta con sé la conoscenza di K? Siano $\tilde{\mathbf{X}}$ le coordinate disomogenee di un generico punto nella scena, espresse nel riferimento globale. Si ipotizzi che il riferimento globale coincida con il riferimento locale di una vista avente K come matrice di calibrazione. Le coordinate $\tilde{\mathbf{X}}$ possono essere riscritte nella forma $\lambda\mathbf{d}$, in cui $\lambda \in \mathbf{R}$, e $\mathbf{d}$ è un versore che punta ad $\tilde{\mathbf{X}}$. Essendo sovrapposti i riferimenti, $P = K \left[\begin{array}{c|c} I & \mathbf{0} \end{array} \right]$ è la matrice di proiezione associata alla vista. Si proietti sull'immagine il punto $\lambda\mathbf{d}$, trasformato in coordinate omogenee:

$$\mathbf{x} \simeq K \left[\begin{array}{c|c} I & \mathbf{0} \end{array} \right] \begin{bmatrix} \lambda\mathbf{d} \\ 1 \end{bmatrix} \tag{3.2.3}$$

La relazione è equivalente a $\mathbf{d} \simeq K^{-1}\mathbf{x}$. Quindi, nel caso siano note K ed $\mathbf{x}$, si può stabilire a quale retta, nella scena, appartenga il punto $\tilde{\mathbf{X}}$. La conoscenza di K consente cioè di utilizzare l'immagine come *rilevatore di direzioni*. Altro utilizzo notevole di K, è nella stima dell'omografia H *indotta* dalla proiezione P tra il piano all'infinito ed il piano immagine. Data una generica vista, l'omografia si ottiene dalla relazione $H \simeq KR$, che implica $HH^T \simeq KK^T$. Nel caso in cui sia nota H e non K, si può calcolare K applicando la *decomposizione di Cholesky*[3] all'ultima relazione. Al

[3]Decomposizione di una matrice hermitiana e definita positiva in una matrice triangolare inferiore e nella sua trasposta coniugata.

contrario H non può essere determinata dalla conoscenza della sola K, perchè ha una struttura più generica.

Per concludere, un'analisi interessante riguarda i gradi di libertà associati alla matrice P. La matrice K fornisce cinque gradi di libertà effettivi (uno per ogni parametro), mentre l'orientamento R e la traslazione $\mathbf{t}$ ne forniscono tre ciascuno. La matrice P ha undici gradi di libertà effettivi, essendo una generica matrice di rango pieno[4] appartenente ad $\mathbf{R}^{3\times4}$. Quindi P assorbe tutti i gradi di libertà delle sue componenti K, R e $\mathbf{t}$. Non sempre i gradi di libertà associati a vettori disomogenei, come $\mathbf{t}$, "sopravvivono" all'applicazione della relazione $\simeq$. Per comprendere questo aspetto, si confrontino le due seguenti relazioni, la seconda delle quali anticipa gli argomenti della sezione 5.1:

$$P \simeq K \left[\; R \,\middle|\, \mathbf{t} \; \right] \tag{3.2.4}$$

$$E \simeq [\mathbf{t}]_\times R \tag{3.2.5}$$

Come il vettore $\mathbf{t}$ è un vettore disomogeneo, la matrice $[\mathbf{t}]_\times$ è una matrice disomogenea, poiché è la matrice antisimmetrica associata a $\mathbf{t}$. Si ipotizzi di conoscere P nella 3.2.4, e di voler calcolare i parametri intrinseci di K, la matrice di rotazione R e la traslazione $\mathbf{t}$. Per prima cosa si può applicare la *decomposizione* RQ[5] alle matrice costituita dalle prime tre colonne di P. Normalizzando la componente triangolare rispetto al suo elemento di posizione $(3,3)$ si ricavano i parametri intrinseci, uguali ai restanti elementi non nulli della matrice. La componente ortogonale non ha bisogno di normalizzazioni, in quanto già coincide con la matrice di rotazione, ortogonale per definizione. Essendo la matrice R ed il vettore $\mathbf{t}$ affiancati, anche quest'ultimo non sarà soggetto a normalizzazioni. Il vettore assume quindi dei valori "prioritari" *indotti*

[4] Solo il rango ridotto impone vincoli che cancellano gradi di libertà.

[5] Fattorizzazione di una matrice *quadrata* in una matrice *triangolare superiore* ed una *ortogonale*.

dalla normalizzazione della matrice di rotazione. Questi valori forniscono le coordinate disomogenee, che possono quindi essere interamente recuperate, mantenendo i tre gradi di libertà di partenza. Il discorso è diverso per la (3.2.5). Ipotizzando di conoscere E, la si può fattorizzare in una matrice antisimmetrica ed una di rotazione applicando gli algoritmi della sezione 5.1. In questo caso, però, l'ambiguità sul fattore di scala di $[\mathbf{t}]_\times$ non può essere risolta, motivo per cui si perde uno dei gradi di libertà ad essa associati.

3.2.3 Distorsione

Il processo di visualizzazione di un dispositivo di acquisizione video non è mai perfettamente coerente con il modello pinhole. La distanza da tale modello è solitamente tanto maggiore quanto minore è la qualità del dispositivo, ed il fenomeno prende il nome di distorsione. La distorsione ha natura sistematica, e tale fatto consente di isolarla all'interno di un opportuno modello matematico. Il modello di Brown ([11]) è uno dei più diffusi modelli di distorsione, ed include sia una componente *radiale* che una *tangenziale*. La distorsione radiale è prodotta dalla curvatura della lente, mentre la distorsione tangenziale dipende dal non perfetto allineamento tra l'asse della lente e quello dell'elemento fotosensibile. La distorsione tangenziale è di norma irrilevante e non verrà presa in considerazione (ulteriori giustificazioni al riguardo saranno fornite nel seguito della trattazione). Nel modello di Brown, la distorsione radiale è funzione della distanza da un *centro di distorsione*, solitamente considerato coincidente con il punto principale:

$$\begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} = \begin{pmatrix} x_d - x_0 \\ y_d - y_0 \end{pmatrix} + \begin{pmatrix} x_d - x_0 \\ y_d - y_0 \end{pmatrix} \cdot L((x_d - x_0)^2 + (y_d - y_0)^2) \qquad (3.2.6)$$

Le coordinate x_d ed y_d rappresentano il punto distorto sull'immagine, x_0 ed y_0 il punto principale, x ed y il punto corretto, ed L la funzione di distorsione. Per rendere più

maneggevole il modello, si sostituisce L con un'opportuna espansione di Taylor in 0. Spesso è sufficiente interrompere l'espansione al secondo ordine. Introducendo la variabile $r = \sqrt{(x_d - x_0)^2 + (y_d - y_0)^2}$ ed i coefficienti k_1 e k_2, si ottiene in definitiva:

$$\begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} = \begin{pmatrix} x_d - x_0 \\ y_d - y_0 \end{pmatrix} + \begin{pmatrix} x_d - x_0 \\ y_d - y_0 \end{pmatrix} \cdot (k_1 r^2 + k_2 r^4) \tag{3.2.7}$$

3.3 Calibrazione

3.3.1 Equazioni per la stima di omografie

Si ipotizzi di conoscere le coordinate di un numero arbitrariamente grande di punti su un piano proiettivo, e di conoscere le coordinate dei *corrispondenti* punti su un secondo piano proiettivo. Si ipotizzi inoltre che le corrispondenze siano veicolate da una matrice omografica H, introdotta nella sezione 3.1.2. Per ogni corrispondenza vale la relazione $\mathbf{x}' \simeq H\mathbf{x}$, che con l'opportuno fattore di scala è un sistema lineare e disomogeneo negli elementi di H. Per evitare inutili complicazioni dovute alla presenza del fattore di scala, si esegue il prodotto vettoriale per $\mathbf{x}'$ di ambo i membri della relazione, ottenendo $\mathbf{x}' \times H\mathbf{x} \simeq \mathbf{0}$. Si tratta di un'operazione lecita, che non interferisce con la presenza del simbolo $\simeq$ (a differenza di quanto accadrebbe, ad esempio, per la somma). La relazione $\mathbf{x}' \times H\mathbf{x} \simeq \mathbf{0}$ è assolutamente equivalente al sistema lineare $\mathbf{x}' \times H\mathbf{x} = \mathbf{0}$, vista la presenza di un membro nullo che rende superfluo il fattore di scala. Avendo a disposizione un numero sufficiente di corrispondenze, e quindi di equazioni omogenee, H si può ricavare applicando gli algoritmi descritti nelle sezioni 3.3.3 e 3.3.4, ed eventualmente applicando una successiva minimizzazione non lineare.

3.3.2 Equazioni per la calibrazione

La stima dei parametri intrinseci della telecamera, è stata eseguita sulla base dei risultati conseguiti da Zhang e descritti in [12], [1] e [13]. Il metodo si basa sull'ipotesi che nella scena sia individuabile un piano, sul quale siano collocati punti di coordinate $\mathbf{X}$, note nel riferimento principale. Misurando le corrispondenti coordinate $\mathbf{x}'$ sull'immagine, si può calcolare l'omografia prospettica H che conduce da un piano all'altro. Il passaggio dall'omografia alla matrice di proiezione P è immediato, specie se il riferimento principale è opportunamento posizionato ed orientato. Nella seguente analisi si ipotizzerà che il riferimento abbia origine nel piano, ed asse z ad esso ortogonale. Ciò rende nulla la terza coordinata di $\mathbf{X}$, portando al seguente risultato:

$$\mathbf{x}' \simeq P\mathbf{X} \simeq K \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_4 \end{pmatrix} \simeq \begin{bmatrix} \mathbf{h_1} & \mathbf{h_2} & \mathbf{h_3} \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_4 \end{pmatrix} \tag{3.3.1}$$

in cui $\mathbf{r}_i$ ed $\mathbf{h}_i$ sono le colonne i-esime rispettivamente della matrice di rotazione R e dell'omografia H. In presenza di un numero sufficiente di corrispondenze, la (3.3.1) implica la seguente:

$$K \begin{bmatrix} \mathbf{r_1} & \mathbf{r_2} & \mathbf{t} \end{bmatrix} \simeq \begin{bmatrix} \mathbf{h_1} & \mathbf{h_2} & \mathbf{h_3} \end{bmatrix} \tag{3.3.2}$$

Uguagliando le singole colonne, e considerandone prodotti scalari incrociati, si ottengono le equazioni:

$$\mathbf{h_1}^T K^{-T} K^{-1} \mathbf{h_2} = 0 \tag{3.3.3}$$

$$\mathbf{h_1}^T K^{-T} K^{-1} \mathbf{h_1} = \mathbf{h_2}^T K^{-T} K^{-1} \mathbf{h_2} \tag{3.3.4}$$

La scomparsa delle $\mathbf{r}_i$ è dovuta alla loro ortonormalità. Con la sostituzione $B = K^{-T}K^{-1}$, le equazioni sono lineari ed omogenee nelle incognite costituite dagli elementi di B. Avendo a disposizione un sufficiente numero di omografie prospettiche in configurazione generale, B può essere calcolata a meno di un fattore di scala, vista

l'omogeneità delle equazioni. La matrice di calibrazione si ricava infine applicando a B la *decomposizione di Cholesky*, che consente di fattorizzare una matrice definita positiva in una triangolare inferiore e nella sua trasposta.

3.3.3 Direct Linear Transform

Le equazioni (3.3.3) ed (3.3.4) sono lineari ed omogenee negli elementi di B, come le equazioni ricavate nella sezione 3.3.1 sono lineari ed omogenee negli elementi di H. La risoluzione di sistemi lineari ed omogenei richiede un breve approfondimento, affrontato nella presente sezione. Il sistema $A\mathbf{x} = \mathbf{0}$, con $A \in \mathbf{R}^{m \times n}$, ammette $\mathbf{0}$ come unica soluzione nel caso in cui $rank(A) = n$, ed infinite soluzioni altrimenti. In generale le soluzioni appartengono ad uno spazio di dimensione $n - rank(A)$. Escludendo il caso in cui la soluzione sia unica e nulla, poichè nelle applicazioni reali si ipotizza sempre l'esistenza di una soluzione non banale, la questione si sposta sulla scelta della soluzione. In presenza di un sufficiente numero di equazioni indipendenti, lo spazio delle soluzioni ha dimensione unitaria ed il problema si semplifica. Una prima possibilità consiste nell'assegnare un valore non nullo ad una delle incognite, rendendo in tal modo disomogeneo il sistema, e risolvibile senza ambiguità. Il metodo fallisce qualora, nella soluzione reale, l'incognita assuma un valore nullo o prossimo allo 0. Come alternativa, la soluzione può essere scelta uguale al "più piccolo" autovettore di $A^T A$ (o meglio l'autovettore corrispondente al suo più piccolo autovalore), ed avente norma unitaria. Tale scelta garantisce il soddisfacimento del sistema $A\mathbf{x} = \mathbf{0}$, ma si dimostra anche essere ottima nella minimizzazione di un eventuale *errore* in presenza di equazioni non esatte. Nella pratica le equazioni sono affette da rumore, che rende pieno il rango della matrice di sistema. L'unica soluzione teorica in tali circostanze, è rappresentata dal vettore nullo, privo di significato pratico nella maggior parte dei

casi. Il problema viene aggirato imponendo che la soluzione abbia norma non nulla, ad esempio unitaria, e che venga minimizzata la norma del *residuo* $A\mathbf{x} = \epsilon$, definito anche *errore algebrico*:

$$\min_{\mathbf{x}} \|A\mathbf{x}\|^2 \qquad \text{s.t. } \|\mathbf{x}\|^2 = 1$$

$$F(\mathbf{x}, \lambda) = \mathbf{x}^T A^T A\mathbf{x} - \lambda(\mathbf{x}^T\mathbf{x} - 1)$$

$$\frac{\partial F(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = 2\mathbf{x}^T A^T A - 2\lambda\mathbf{x}^T = \mathbf{0} \Leftrightarrow (A^T A - \lambda I)\mathbf{x} = \mathbf{0}$$

$$\frac{\partial F(\mathbf{x}, \lambda)}{\partial \lambda} = 0 \Leftrightarrow \|\mathbf{x}\|^2 = 1$$

L'applicazione del metodo dei moltiplicatori di Lagrange, mostra chiaramente come la soluzione al problema sia costituita da un autovettore normalizzato di $A^T A$. Su quale degli n autovettori ricade la scelta? Il fatto che $\mathbf{x}$ abbia norma unitaria, conduce all'implicazione $(A^T A - \lambda I)\mathbf{x} = \mathbf{0} \Rightarrow \mathbf{x}^T A^T A\mathbf{x} = \|A\mathbf{x}\|^2 = \lambda$, che evidenzia come l'autovalore λ più piccolo, ed il relativo autovettore, costituiscano la soluzione del problema. Si noti come tale autovettore possa essere ottenuto equivalentemente a partire dalla *decomposizione in valori singolari* (SVD) della matrice di sistema. L'autovettore coincide infatti con l'ultima colonna della matrice V nella fattorizzazione $A = UWV^T$. Le fasi di costruzione della matrice di sistema, e di risoluzione attraverso decomposizione in valori singolari, si riassumono nel *direct linear transform* (DLT), algoritmo basilare ai fini della trattazione. La minimizzazione dell'errore algebrico può condurre a soluzioni diverse da quelle che intuitivamente si attenderebbero ([14]). Per tale ragione il DLT è spesso utilizzato per generare delle stime iniziali, fornite in un secondo momento ad algoritmi iterativi che provvedono alla minimizzazione di grandezze più significative, come l'*errore geometrico*. I metodi impiegati nella risoluzione dei sistemi lineari disomogenei, soffrono notoriamente il *mal condiziona-*

mento[6] della matrice di sistema, che può misurarsi come rapporto tra il più grande ed il più piccolo degli autovalori o dei valori singolari, rispettivamente nel caso di sistemi quadrati o rettangolari. Anche il DLT è soggetto a fenomeni di mal condizionamento, dipendenti dal rapporto tra il più grande ed il secondo più piccolo dei valori singolari (si osservi che nel DLT una matrice di sistema *ideale* ha uno ed un solo valore singolare nullo). L'applicazione di un'opportuna strategia di normalizzazione, esaustivamente descritta in [18] e [19] ed analizzata in seguito nelle specifiche applicazioni, è essenziale per l'ottenimento di risultati significativi.

3.3.4 RANSAC

L'algoritmo DLT viene spesso inserito all'interno di un algoritmo più vasto, il *RANSAC*[7], che ha lo scopo di escludere dal processo di stima tutti quei dati che, sulla base di un criterio opportuno, si ritengano troppo lontani da una stima attendibile. Tali dati vengono definiti *outlier*, mentre si definiscono *inlier* i dati ritenuti validi. Il RANSAC appartiene alla categoria degli algoritmi di stima *robusti*, cioè insensibili alla presenza di dati errati o particolarmente rumorosi. La genericità dell'algoritmo impone l'introduzione di una notazione generica, riportata nello schema seguente insieme al suo funzionamento:

[6] La soluzione di un sistema mal condizionato, è molto sensibile a variazioni che agiscano sugli elementi della matrice di sistema, dovute ad esempio a misurazioni rumorose.

[7] Abbreviazione di *RANdom SAmple Consensus*.

Notazione

S: Insieme di tutti i dati (inlier ed outlier).

s: Sottoinsieme di S.

S_i: Insieme degli inlier, impostato dall'algoritmo ad ogni iterazione.

t: Massima distanza dal modello.

T: Minimo numero di inlier necessario all'accettazione del modello.

N: Numero massimo di iterazioni.

Algoritmo

1) Seleziona casualmente da S un sottoinsieme s di dati, e genera il corrispondente modello.

2) Determina l'insieme di dati S_i che distano meno di t dal modello. S_i rappresenta il *consenso* associato alla selezione s, e contiene gli inlier di S rispetto ad s.

3) Se la dimensione di S_i è maggiore della soglia T, genera un nuovo modello basato su tutti i dati presenti in S_i ed esce dall'algoritmo.

4) Se la dimensione di S_i è minore della soglia T, ripete i punti precedenti con una nuova selezione s.

5) Dopo N iterazioni, se possibile calcola il modello usando il più grande insieme S_i ottenuto.

Si consideri come esempio la stima della matrice fondamentale, che sarà introdotta nel capitolo 4. Il DLT ha bisogno di almeno otto corrispondenze per calcolare F, quindi s deve avere cardinalità non più piccola di 8. Il generico dato di cui si parla nello schema si traduce nella corrispondenza tra due punti. Come primo passaggio si selezionano casualmente un sufficiente numero di corrispondenze, e le si usa per una stima di F

basata sul DLT. Ipotizzando di rappresentare con $\mathbf{x}$ ed $\mathbf{x}'$ la generica corrispondenza, si valuta la distanza d dal modello di tutte le corrispondenze utilizzando il seguente algoritmo:

- $\mathbf{x}'/x'_3 \rightarrow \mathbf{x}'$
- $F\mathbf{x} \rightarrow \mathbf{l}$
- $\mathbf{l}/\sqrt{l_1^2 + l_2^2} \rightarrow \mathbf{l}$
- $|\mathbf{x}'^T\mathbf{l}| \rightarrow d_1$
- $\mathbf{x}/x_3 \rightarrow \mathbf{x}$
- $F^T\mathbf{x}' \rightarrow \mathbf{l}'$
- $\mathbf{l}'/\sqrt{l'^2_1 + l'^2_2} \rightarrow \mathbf{l}'$
- $|\mathbf{x}^T\mathbf{l}'| \rightarrow d_2$
- $\sqrt{d_1^2 + d_2^2} \rightarrow d$

L'algoritmo per la valutazione delle distanze varia chiaramente di caso in caso. Le corrispondenze sufficientemente vicine formano l'insieme S_i degli inlier. Se S_i è sufficientemente grande, viene utilizzato nella sua interezza per una stima raffinata di F, e l'algoritmo termina con questo risultato. Altrimenti la procedura si ripete su un nuovo insieme s di punti. Dal punto di vista realizzativo, la verifica sul già avvenuto utilizzo di un insieme si è ottenuta con l'ausilio di una matrice sparsa[8]. Solitamente il numero delle possibili combinazioni di elementi da inserire in s è molto grande, motivo per cui il RANSAC deve essere interrotto dopo un prefissato numero di iterazioni. In

[8]Le matrici sparse sono allocate dalla funzione *cvCreateSparseMat*.

tal caso la stima si effettua sul più grande insieme di inlier ricavato durante l'algoritmo, del quale quindi si deve tenere traccia. Molte delle funzioni incluse nella libreria OpenCV prevedono l'applicazione opzionale del RANSAC, e sono state implementate sulla base dello schema RANSAC anche alcune delle funzioni ad hoc necessarie al sistema di visione.

3.3.5 Minimizzazione iterativa

I risultati ottenuti minimizzando l'errore algebrico non sempre sono soddisfacenti. Nel caso della calibrazione, ad esempio, il DLT conduce generalmente a stime piuttosto lontane dagli effettivi valori dei parametri intrinseci. I dati sperimentali presentati in [12] evidenziano chiaramente questo aspetto. Si tratta di scostamenti che in alcuni casi raggiungono il 10% del valore reale del parametro. I dati evidenziano anche come l'applicazione del DLT ad insiemi diversi di equazioni (chiaramente ricavati dalla stessa telecamera), conduca a risultati piuttosto variabili. Lo scoglio dell'instabilità viene superato minimizzando iterativamente ([15]), a valle del DLT, un'opportuna funzione di costo, utilizzando gli stessi risultati del DLT come condizioni di partenza. Seguendo tale iter si ottengono di volta in volta stime pressoché identiche dei parametri. Non tutti i problemi di stima, per la risoluzione dei quali si utilizzi il DLT, richiedono che venga applicata una minimizzazione iterativa finale. Nel calcolo della matrice fondamentale, ad esempio, in presenza di dati opportunamente normalizzati il DLT fornisce ottimi risultati.

Le funzioni di costo minimizzate con tecniche iterative sono tipicamente rappresentabili come somme di quadrati. Gli algoritmi generalmente applicati a funzioni di questo tipo sono l'algoritmo di *Gauss-Newton* e quello di *Levenberg-Marquardt*. Si definiscano *parametri* le variabili che compaiono nella funzione di costo e rispetto alle

quali si voglia eseguire la minimizzazione. L'algoritmo di Gauss-Newton approssima la funzione di costo con una funzione *quadrica*, intorno ad una stima iniziale dei parametri. Calcola poi algebricamente (in forma chiusa) il minimo della quadrica, ed utilizza i corrispondenti parametri minimizzanti come stima iniziale per una successiva iterazione. Si dimostra come l'algoritmo vada a convergenza se sono verificate semplici ipotesi. Per una descrizione più formale, si introduca la funzione di costo scalare $F(\mathbf{x}) = \mathbf{f}(\mathbf{x})^T\mathbf{f}(\mathbf{x})/2$, nella quale $\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & .. & f_m(\mathbf{x}) \end{pmatrix}^T$. L'approssimazione quadrica di $F(\mathbf{x})$ la si può ottenere a partire da un'approssimazione lineare di $\mathbf{f}(\mathbf{x})$, passaggio preso in prestito dall'algoritmo di *Newton*:

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^2) \Rightarrow \mathbf{f}(\mathbf{h}+\mathbf{x}) \simeq \mathbf{l}(\mathbf{h}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h} \Rightarrow$$

$$\Rightarrow F(\mathbf{x}+\mathbf{h}) \simeq L(\mathbf{h}) = F(\mathbf{x}) + \mathbf{h}^T\mathbf{J}^T\mathbf{f}(\mathbf{x}) + \frac{1}{2}\mathbf{h}^T\mathbf{J}^T\mathbf{J}\mathbf{h} \tag{3.3.5}$$

Nelle relazioni (3.3.5), la matrice $\mathbf{J}(\mathbf{x})$ rappresenta la *Jacobiana* di $\mathbf{f}(\mathbf{x})$ ed $\mathbf{h}$ una *variazione* dei parametri. Per quanto anticipato, l'obiettivo è ora la minimizzazione algebrica della funzione scalare $L(\mathbf{h})$, chiaramente non rispetto ad $\mathbf{x}$ (che è fissato e rappresenta il centro dell'approssimazione quadrica) ma rispetto alla variazione $\mathbf{h}$. Il passaggio è semplice, poiché basta calcolarne il gradiente ed uguagliarlo a $\mathbf{0}$. Il gradiente di $L(\mathbf{h})$ è $\mathbf{J}^T\mathbf{f}(\mathbf{x}) + \mathbf{J}^T\mathbf{J}\mathbf{h}$, quindi la variazione minimizzante è *una* soluzione del sistema lineare $\mathbf{J}^T\mathbf{J}\mathbf{h} = -\mathbf{J}^T\mathbf{f}(\mathbf{x})$, ma quale? Poiché $L(\mathbf{h})$ è una funzione quadrica, è necessario e sufficiente che la sua matrice *Hessiana* (costante) sia definita positiva affinché il gradiente si annulli in un solo punto e quel punto sia un minimo. L'Hessiana di $L(\mathbf{h})$ è $\mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x})$, sicuramente definita positiva nell'ipotesi *conservativa* che $\mathbf{J}(\mathbf{x})$ abbia rango pieno. Quindi, sotto tale ipotesi (che deve essere garantita per ogni stima intermedia dei parametri), l'algoritmo di Gauss-Newton funziona correttamente. L'algoritmo di Levenberg-Marquardt generalizza l'algoritmo di Gauss-Newton con

l'aggiunta di una componente *smorzante* alla matrice di sistema $\mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x})$. Il nuovo sistema per il calcolo di $\mathbf{h}$ è il seguente:

$$(\mathbf{J}^T\mathbf{J} + \mu I)\mathbf{h} = -\mathbf{J}^T\mathbf{f}(\mathbf{x}) \tag{3.3.6}$$

Imponendo $\mu \geq 0$, la matrice di sistema è invertibile ed il sistema ammette una ed una sola soluzione. Quando $\mu = 0$ l'algoritmo si comporta coma l'algoritmo di Gauss-Newton. Quando μ è sufficientemente grande da rendere trascurabile $\mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x})$, allora $\mathbf{h} = -\mathbf{J}^T\mathbf{f}(\mathbf{x})/\mu$. Essendo $\mathbf{J}^T\mathbf{f}(\mathbf{x})$ il gradiente di $F(\mathbf{x})$, allora la variazione $\mathbf{h}$ rappresenta un *piccolo* passo nel verso opposto al gradiente, esattamente come nel metodo *steepest descent*. Il valore di μ evolve durante l'esecuzione dell'algoritmo, adattandosi alle caratteristiche di $F(\mathbf{x})$. In particolare μ tende a diminuire nel caso in cui $L(\mathbf{h})$ fornisca una buona approssimazione di $F(\mathbf{x}+\mathbf{h})$, mentre tende ad aumentare in caso contrario. In altri termini, l'algoritmo raggiunge le prestazioni dell'algoritmo di Gauss-Newton nel caso in cui la funzione sia abbastanza regolare, mentre procede con più cautela in caso contrario, scegliendo di muoversi a piccoli passi nel verso opposto al gradiente (scelta sempre vincente, qualora non si sappia come procedere). L'indice che fornisce una misura della regolarità di $F(\mathbf{x})$ viene definito *rapporto di guadagno*:

$$\rho = \frac{F(\mathbf{x}) - F(\mathbf{x}+\mathbf{h})}{L(\mathbf{0}) - L(\mathbf{h})} \tag{3.3.7}$$

Come μ debba evolvere in funzione di ρ dipende da scelte specifiche, generalmente corrispondenti a leggi matematiche affatto intuitive.

Si ipotizzi che il vettore $\mathbf{x}$ dei parametri appartenga ad $\mathbf{R}^n$, e quindi che la matrice di sistema nella (3.3.6) abbia dimensione $n \times n$. Il sistema (3.3.6) può essere risolto invertendo $(\mathbf{J}^T\mathbf{J} + \mu I)$. L'operazione di inversione ha complessità computazionale $O(n^3)$, quindi il numero di elaborazioni eseguite cresce molto rapidamente al crecere

di n. Informazioni a priori sulla struttura della Jacobiana possono essere utilizzate per velocizzare enormemente la risoluzione della (3.3.6). Si ipotizzi che la Jacobiana abbia la seguente struttura a blocchi:

$$J = \left[\begin{array}{c|ccc} A_1 & B_1 & & \\ A_2 & & B_2 & \\ \vdots & & & \ddots \\ A_N & & & B_N \end{array}\right] \tag{3.3.8}$$

Sfruttando la particolare struttura, si determini la matrice di sistema della (3.3.6):

$$\left[\begin{array}{cccc} A_1^T & A_2^T & \dots & A_N^T \\ B_1^T & & & \\ & B_2^T & & \\ & & \ddots & \\ & & & B_N^T \end{array}\right] \left[\begin{array}{cccc} A_1 & B_1 & & \\ A_2 & & B_2 & \\ \vdots & & & \ddots \\ A_N & & & B_N \end{array}\right] + \mu I =$$

$$= \left[\begin{array}{ccccc} \sum_{i=1}^N A_i^T A_i & A_1^T B_1 & A_2^T B_2 & \dots & A_N^T B_N \\ (A_1^T B_1)^T & B_1^T B_1 & & & \\ (A_2^T B_2)^T & & B_2^T B_2 & & \\ \vdots & & & \ddots & \\ (A_N^T B_N)^T & & & & B_N^T B_N \end{array}\right] + \mu I =$$

$$= \left[\begin{array}{ccccc} U & W_1 & W_2 & \dots & W_N \\ W_1^T & V_1 & & & \\ W_2^T & & V_2 & & \\ \vdots & & & \ddots & \\ W_N^T & & & & V_N \end{array}\right] + \mu I = \left[\begin{array}{ccccc} U^* & W_1 & W_2 & \dots & W_N \\ W_1^T & V_1^* & & & \\ W_2^T & & V_2^* & & \\ \vdots & & & \ddots & \\ W_N^T & & & & V_N^* \end{array}\right] \tag{3.3.9}$$

Nella (3.3.9) si sono utilizzate le relazioni $U = \sum_{i=1}^N A_i^T A_i$, $V_i = B_i^T B_i$, $W_i = A_i^T B_i$, $U^* = U + \mu I$ e $V_i^* = V_i + \mu I$. Sostituendo $\mathbf{f}(\mathbf{x})$ con $\begin{pmatrix} \epsilon_1 & \epsilon_2 & \dots & \epsilon_N \end{pmatrix}^T$, si determini il vettore dei termini noti della 3.3.6:

$$-\left[\begin{array}{cccc} A_1^T & A_2^T & \dots & A_N^T \\ B_1^T & & & \\ & B_2^T & & \\ & & \ddots & \\ & & & B_N^T \end{array}\right] \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^N A_i^T \epsilon_i \\ B_1^T \epsilon_1 \\ \vdots \\ B_N^T \epsilon_N \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \epsilon_A \\ \epsilon_{B_1} \\ \vdots \\ \epsilon_{B_N} \end{pmatrix} \stackrel{def}{=} \begin{pmatrix} \epsilon_A \\ \epsilon_B \end{pmatrix}$$

Il partizionamento della matrice di sistema induce il partizionamento $\begin{pmatrix} \delta_A & \delta_B \end{pmatrix}^T$ del vettore delle incognite $\mathbf{h}$. Definite $V^* = diag(V_1^*, V_2^*, ..., V_N^*)$ e $W = \begin{bmatrix} W_1 & W_2 & \dots & W_N \end{bmatrix}$, il sistema 3.3.6 si può riscrivere nella forma:

$$\begin{bmatrix} U^* & W \\ W^T & V^* \end{bmatrix} \begin{pmatrix} \delta_A \\ \delta_B \end{pmatrix} = \begin{pmatrix} \epsilon_A \\ \epsilon_B \end{pmatrix} \tag{3.3.10}$$

Si moltiplichino i membri della 3.3.10 per un'opportuna matrice:

$$\begin{bmatrix} I & -WV^{*-1} \\ \mathbf{0} & I \end{bmatrix} \begin{bmatrix} U^* & W \\ W^T & V^* \end{bmatrix} \begin{pmatrix} \delta_A \\ \delta_B \end{pmatrix} = \begin{bmatrix} I & -WV^{*-1} \\ \mathbf{0} & I \end{bmatrix} \begin{pmatrix} \epsilon_A \\ \epsilon_B \end{pmatrix} \Leftrightarrow$$

$$\begin{bmatrix} U^* - WV^{*-1}W^T & \mathbf{0} \\ W^T & V^* \end{bmatrix} \begin{pmatrix} \delta_A \\ \delta_B \end{pmatrix} = \begin{pmatrix} \epsilon_A - WV^{*-1}\epsilon_B \\ \epsilon_B \end{pmatrix} \tag{3.3.11}$$

La presenza di un blocco nullo nella matrice a primo membro permette di calcolare δ_A separatamente, risolvendo il sistema $(U^* - WV^{*-1}W^T)\delta_A = \epsilon_A - WV^{*-1}\epsilon_B$. La matrice utilizzata per trasformare il sistema non è il frutto di complesse considerazione algebriche. Si è semplicemente cercata una trasformazione che rendesse nullo il blocco $(1, 2)$ della matrice sistema, e contemporaneamente non modificasse il blocco $(2, 2)$. È infatti la struttura di quest'ultimo che apporta il contributo maggiore alla risoluzione del sistema. Una volta calcolata δ_A, si dovrebbe procedere al calcolo di δ_B risolvendo il sistema $V^*\delta_B = \epsilon_B - W^T\delta_A$. Ma la matrice V^* è diagonale a blocchi, quindi il sistema si può suddividere in N sottosistemi singolarmente risolvibili! In particolare si tratta dei sottosistemi $\delta_{B_i} = V_i^{*-1}(\epsilon_{B_i} - W_i^T\delta_A)$. Si noti che in letteratura compare frequentemente anche la matrice $Y_i = W_iV_i^{*-1}$, non utilizzata nella presente esposizione. Sebbene non utile alla comprensione dei passaggi algebrici, ed anzi controproducente in quanto appesantisce la notazione, la matrice racchiude un prodotto frequentemente eseguito dall'algoritmo, aspetto da non sottovalutare nel caso se ne esegua un'implementazione. Una volta risolto il sistema sfruttando la struttura a blocchi di $\mathbf{J}(\mathbf{x})$, il metodo di Levenberg-Marquardt procede normalmente. Questa

variante dell'algoritmo viene definita *sparsa*, ed è molto utilizzata in computer vision negli algoritmi fuori linea per il raffinamento dei risultati (*bundle adjustment*).

3.3.6 Matlab Calibration Toolbox

Sebbene MATLAB sia sconsigliabile in applicazioni di tipo real-time, si tratta di un ambiente di programmazione particolarmente funzionale nello svolgimento di compiti *fuori linea*, tra cui la calibrazione. Sviluppato dall'università finlandese di Oulu, il *Matlab Calibration Toolbox* ([20]) fornisce un sistema altamente interattivo per la calibrazione delle telecamere. Nella sezione 3.3.2 si è esposto un metodo grazie al quale, nota la collocazione nella scena di punti coplanari, è possibile ottenere una stima dei parametri intrinseci. La comune scacchiera da gioco è sicuramente l'*oggetto di calibrazione* che più va incontro alle esigenze dell'algoritmo. Nel caso della scacchiera, i punti di coordinate note sono i vertici delle case. Scegliendo un riferimento globale avente origine coincidente con uno dei vertici, ed asse z ortogonale alla scacchiera, le coordinate assolute di tutti i vertici sono facilmente calcolabili, considerato che le case sono quadrati di uguale lato. Il toolbox di calibrazione riceve in ingresso una serie di fotogrammi aventi una scacchiera come soggetto, di volta in volta ripresa sotto differenti angolazioni. Rilevate in ciascuna immagine le posizioni dei vertici, viene eseguito l'algoritmo della sezione 3.3.2. Le distorsioni, volutamente tralasciate fino a questo punto, devono ora essere tenute in considerazione. Essendo ricavato a partire dal modello pinhole, l'algoritmo non può fornire risultati esatti in presenza di distorsioni. L'entità di quest'ultime, d'altro canto, non può superare determinati limiti, imposti dall'utilizzabilità della telecamera (a meno che non si parli di telecamere ad uso speciale). L'errore legato alle distorsioni si può allora ritenere sufficientemente piccolo, motivo per cui la stima ottenuta può ancora essere considerata come un buon

punto di partenza in una successiva stima iterativa, che in questo caso tenga anche conto dei parametri delle distorsioni includendo, ad esempio, la 3.2.7. Si noti, infine, che quanto riferito nella sezione 3.3.3 riguardo ai differenti risultati forniti dalla minimizzazione algebrica e dalla minimizzazione geometrica, risulta particolarmente vero nel caso della calibrazione (lo si è potuto constatare sperimentalmente su immagini *rettificate*, cioè prive di distorsioni). Oltre ai parametri intrinseci ed ai parametri delle distorsioni, il toolbox calcola anche le relative incertezze ed i *parametri estrinseci*. In base al valore delle incertezze l'utente può interagire con il programma, aggiungendo ad esempio nuove immagini o imponendo un rilevamento più preciso dei vertici. Le incertezze sui parametri delle distorsioni sono un valido strumento per stabilire quali distorsioni agiscano effettivamente sul dispositivo. È infatti consigliabile escludere dal modello di distorsioni quei parametri i cui valori siano paragonabili a quelli delle relative incertezze. Nel caso specifico, ciò ha portato alla scelta del modello semplificato 3.2.7. I parametri estrinseci, di scarsa utilità nella calibrazione di una sola telecamera, forniscono le posizioni e gli orientamenti relativi tra la scacchiera e la telecamera (figura 3.5). Nella figura 3.4 è mostrata la scacchiera in una delle angolazioni riprese, insieme ai vertici rilevati dal programma. Gli effetti delle distorsioni sull'immagine sono diagrammati nella figura 3.6, e sono stati eliminati dal sistema di visione combinando le funzioni *cvInitUndistortMap* e *cvRemap*. Nella tabella seguente sono riportati i risultati della calibrazione:

Modello pinhole

$\alpha_x = 786.33413 \pm 0.95328$ $\alpha_y = 862.96504 \pm 1.05943$

$x_0 = 362.27912 \pm 1.20780$ $y_0 = 282.87035 \pm 1.49081$

$s = 0.00103 \pm 0.00024$

Distorsioni

$k_1 = -0.26725 \pm 0.00343$ $k_2 = 0.27560 \pm 0.01662$

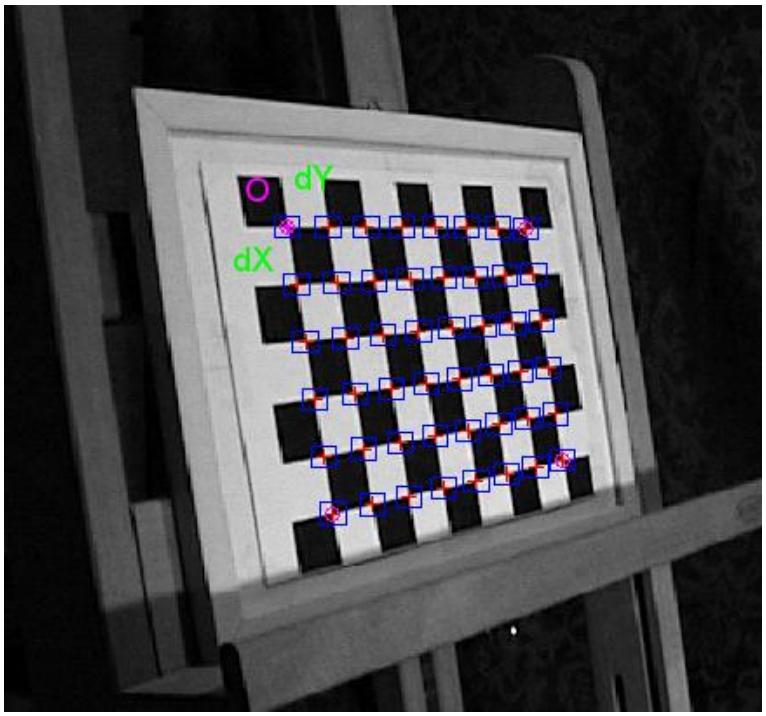

Figura 3.4: Una delle quaranta immagini utilizzate per la calibrazione della telecamera. Si noti la scelta del riferimento globale da parte del toolbox.

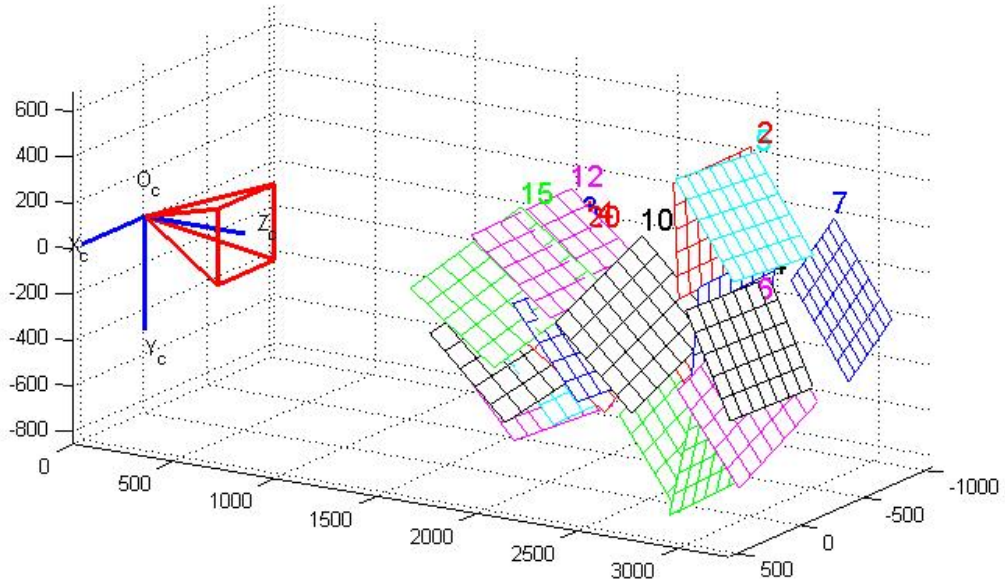

Figura 3.5: Posizioni ed orientamenti della scacchiera rispetto alla telecamera, ritenuta fissa.

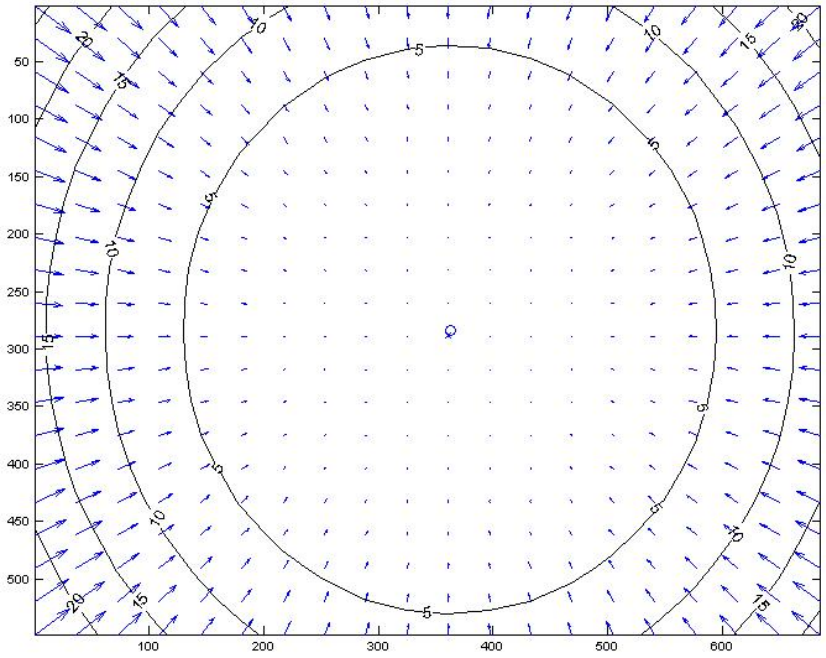

Figura 3.6: Il campo vettoriale (blu) mostra gli effetti delle distorsioni sull'immagine. La forma delle isocurve è indicativa della presenza della sola componente radiale.

Capitolo 4

Ricostruzione proiettiva del moto e dell'ambiente

4.1 Geometria epipolare

La geometria epipolare studia le intersezioni dei piani immagine con il *pencil*[1] di piani avente la baseline come asse. Avendo a disposizione due immagini di una stessa scena, e su ciascuna di esse le proiezioni $\mathbf{x}$ ed $\mathbf{x}'$ di uno stesso punto della scena $\mathbf{X}$, ci si chiede se esista una relazione matematica tra le coordinate dei punti $\mathbf{x}$ ed $\mathbf{x}'$. La risposta affermativa al quesito è il frutto di poche semplici considerazioni. I raggi retroproiettati da $\mathbf{x}$ ed $\mathbf{x}'$ si intersecano in $\mathbf{X}$, ed appartengono al piano $\mathbf{\Pi}$. Che la baseline appartenga a $\mathbf{\Pi}$ si evince immediatamente dalla figura 4.1, quindi il piano risulterebbe sovradeterminato qualora fossero contemporaneamente noti i due raggi e la baseline (sono sufficienti due rette a definire un piano). Si ipotizzi allora che $\mathbf{x}$ e la baseline siano le uniche entità note. Il raggio retroproiettato da $\mathbf{x}'$ deve appartenere al piano generato dalla baseline e dal raggio retroproiettato da $\mathbf{x}$. Il vincolo sul raggio si traduce in un vincolo sulla posizione di $\mathbf{x}'$ che, appartenente al proprio piano immagine, è anche vincolato a giacere su $\mathbf{\Pi}$, e quindi sulla loro intersezione, definita *retta epipolare*. Il piano $\mathbf{\Pi}$ è anche definito *piano epipolare*, ed

[1] Insieme costituito da ∞^1 elementi.

epipoli sono definite le intersezioni della baseline con i piani immagine. Gli epipoli sono sufficienti a caratterizzare la geometria epipolare, essendo i centri dei due pencil formati, su ciascuna immagine, da tutte le possibili rette epipolari.

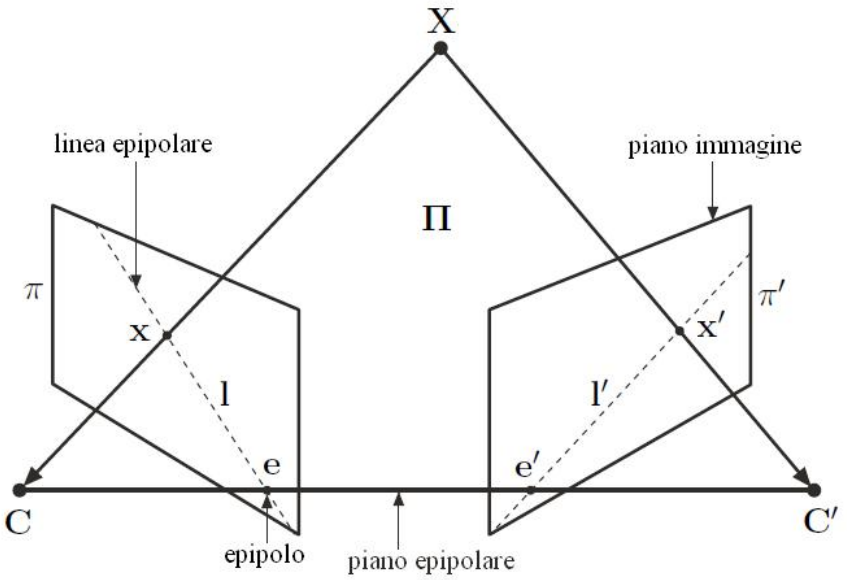

Figura 4.1: Descrizione della figura.

4.2 Calcolo della matrice fondamentale

I vettori $\mathbf{x}$ ed $\mathbf{x}'$, introdotti nella sezione 4.1, sono vincolati algebricamente dalla *matrice fondamentale*, che codifica la geometria epipolare:

$$\mathbf{x}'^T F \mathbf{x} = 0 \tag{4.2.1}$$

Coerentemente con quanto formulato nella sezione 4.1, la relazione (4.2.1) può essere interpretata come l'appartenenza di un punto immagine ad un'opportuna retta epipolare, espressa dal prodotto scalare $\mathbf{x}'^T \mathbf{l}' = 0$, in cui $\mathbf{l}' = F\mathbf{x}$. Gli epipoli $\mathbf{e}$ ed $\mathbf{e}'$ si ottengono rispettivamente come *kernel* destro e sinistro di F. Gli epipoli non possono coincidere con il vettore nullo poiché, per quanto esposto nella sezione 3.1.1, non es-

prime coordinate omogenee. Di conseguenza la matrice F non può essere invertibile, affinché gli epipoli possano esistere, e deve avere rango 2, per garantire la loro unicità. La matrice fondamentale appartiene ad $\mathbf{R}^{3\times3}$, quindi vanta in partenza nove gradi di libertà, portati ad otto dal vincolo $det(F) = 0$. La geometria epipolare associata è soggetta all'ulteriore vincolo del fattore di scala, dovuto all'omogeneità della (4.2.1), che porta a sette i gradi di libertà effettivi (contro gli otto della rappresentazione matriciale). La geometria epipolare è univocamente ricavabile dalle matrici di proiezione associate alle viste. Alternativamente, note le due matrici di calibrazione K e K', la traslazione $\mathbf{t}$ e l'orientamento R relativi, può essere ricavata dalla seguente relazione:

$$F \simeq K'^{-T}[\mathbf{t}]_{\times}RK^{-1} \tag{4.2.2}$$

Si noti come i sedici gradi di libertà effettivi che compaiono a membro destro, rispettivamente cinque per ogni matrice di calibrazione, tre per la matrice di rotazione e tre per la traslazione, si riducano ai sette gradi di libertà effettivi del membro sinistro. Dei gradi di libertà persi, otto sono dovuti ai vincoli imposti dai prodotti matriciali, il restante alla presenza della matrice *disomogenea* $[\mathbf{t}]_{\times}$, che rappresenta la traslazione (si veda al riguardo la sezione 3.2.2). L'equazione (4.2.1) è immediatamente riconducibile ad un'equazione lineare ed omogenea negli elementi di F. Accumulando un sufficiente numero di equazioni, una per ogni corrispondenza tra $\mathbf{x}$ ed $\mathbf{x}'$, ed applicando il DLT, si ricava una stima della matrice F. Affinché l'algoritmo fornisca risultati attendibili, è indispensabile che i dati in ingresso, nella fattispecie le coordinate dei punti, siano opportunamente normalizzati (come accennato nella sezione 3.3.3). In [18] si introduce informalmente ed intuitivamente una possibile strategia di normalizzazione, la cui ottimalità è dimostrata in [19]. Si consideri la prima delle due immagini. Come primo passo, le sue coordinate vanno traslate in modo da spostare nell'origine

il centroide dei punti. Secondo passo è lo *scaling isotropo*, che ha lo scopo di fissare a $\sqrt{2}$ la media delle distanze dei punti dall'origine. Stesso procedimento va eseguito sulla seconda immagine. Nella sostanza, lo scopo della normalizzazione è quello di rendere *abbastanza omogenei* gli elementi della matrice di sistema, al fine di ridurre il mal condizionamento. Si osservi che, rappresentando i punti in coordinate omogenee, la trasformazione *affine* imposta dall'algoritmo di normalizzazione si traduce in una semplice trasformazione lineare (questo "retroscena" rende le coordinate omogenee appetibili anche in altri contesti), avente matrice:

$$T = \begin{bmatrix} \sqrt{2}/\sigma & 0 & -\sqrt{2}x_0/\sigma \\ 0 & \sqrt{2}/\sigma & -\sqrt{2}y_0/\sigma \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2.3}$$

dove x_0 ed y_0 sono le coordinate del centroide, e σ il fattore di scala. Per l'applicazione del DLT in presenza di coordinate normalizzate, basti considerare che:

$$\mathbf{x'}^T F \mathbf{x} = 0 \Leftrightarrow (T'\mathbf{x'})^T (T'^{-T} F T^{-1})(T\mathbf{x}) = 0 \Leftrightarrow \overline{\mathbf{x}}'^T \overline{F} \overline{\mathbf{x}} = 0 \tag{4.2.4}$$

dove T e T' sono le matrici di trasformazione nelle due immagini, $\overline{\mathbf{x}}$ ed $\overline{\mathbf{x}}'$ le nuove coordinate, ed $\overline{F}$ la nuova matrice fondamentale da stimare, attraverso un procedimento questa volta privo di mal condizionamento. Ricavata $\overline{F}$, si calcola $F = T'^T \overline{F} T$. In presenza di dati rumorosi, la matrice ha generalmente rango pieno, a dispetto del rango 2 idealmente atteso. La tecnica utilizzata per ridurre il rango di F consiste nel calcolarne la fattorizzazione SVD, nel sostituire 0 al più piccolo valore singolare, e nel ricomporre la fattorizzazione nella nuova F. La stima ottenuta può a questo punto essere utilizzata come stima di partenza nell'ambito di una minimizzazione iterativa di un opportuno errore geometrico. Nel caso della matrice fondamentale, il DLT produce risultati particolarmente soddisfacenti (come evidenziato in [18]), direttamente utilizzabili in applicazioni non particolarmente esigenti. La funzione

cvFindFundamental svolge in autonomia i passaggi analizzati nella presente sezione. Sebbene i risultati forniti siano particolarmente precisi, fatto facilmente verificabile con l'ausilio dell'algoritmo per la valutazione delle distanze introdotto nella sezione 3.3.4, la *cvFindFundamental* ha rappresentato un serio ostacolo al corretto funzionamento del sistema visivo. Si è potuto constatare come la motivazione risiedesse nella non univocità dei risultati prodotti dalla funzione, dovuta a quella che tecnicamente viene definita *configurazione degenere* dei dati in ingresso.

4.2.1 Configurazioni degeneri

Come ampiamente descritto nella sezione 3.3.3, il DLT si basa sulla risoluzione di un sistema lineare ed omogeneo, che nel caso ideale possiede ∞^1 soluzioni. In presenza di dati rumorosi, si è già constatato come la matrice di sistema assuma rango pieno, seppur molto debolmente, e ciò ha condotto a formulare il DLT come un problema di minimizzazione. La *degenerazione* ha origini diametralmente opposte. Nel caso in cui i dati siano insufficienti, o mostrino un'eccessiva interdipendenza, la matrice di sistema sarà caratterizzata da un rango inferiore al rango atteso, corrispondente ad uno spazio delle soluzioni di dimensione più che unitaria. La non univocità della soluzione, riportata in coordinate disomogenee, determina la degenerazione, dettagliatamente descritta in [7] e [21]. Si può constatare come per sua natura la degenerazione sia un problema non risolvibile, se non modificando i dati in ingresso o il modello stesso. Per tale ragione in letteratura si affrontano principalmente problemi di *rilevamento* di configurazioni degeneri e di *modifica* del modello. Sebbene si potrebbe ritenere facile cosa rilevare la degenerazione semplicemente analizzando il rango della matrice di sistema, nella realtà la presenza di rumore e di outlier rende le cose molto più complicate. Gli outlier, che in quanto dati erronei sono tipicamente

in contrasto con il modello reale, potrebbero "arricchire" le equazioni del sistema nascondendo la degenerazione ad un eventuale rilevatore basato su rango. Le tipiche configurazioni degeneri, relativamente alla stima della matrice fondamentale, si hanno in presenza di sovrapposizione dei centri associati alle due viste (moto degenere della telecamera), o nel caso in cui i punti proiettati sulle due immagini siano collocati, nella scena, su uno stesso piano (ambiente degenere). Ovviamente, anche configurazioni prossime alle configurazioni degeneri portano a stime di F non attendibili. In fase simulativa, la degenerazione si è rivelata un ostacolo difficilmente superabile. Si ritiene che le cause siano da cercarsi nel limitato spazio di lavoro della telecamera, e nella relativa vicinanza degli oggetti ripresi. Il problema è stato superato vincolando meccanicamente il moto della telecamera. Nelle figure 4.2-4.5 sono mostrate quattro immagini identiche a coppie, ed alcune delle rette epipolari associate alle due matrici fondamentali ricavate. Dalle differenze tra le rette epipolari emerge come le matrici fondamentali siano differenti, sebbene ricavate da coppie identiche di immagini.

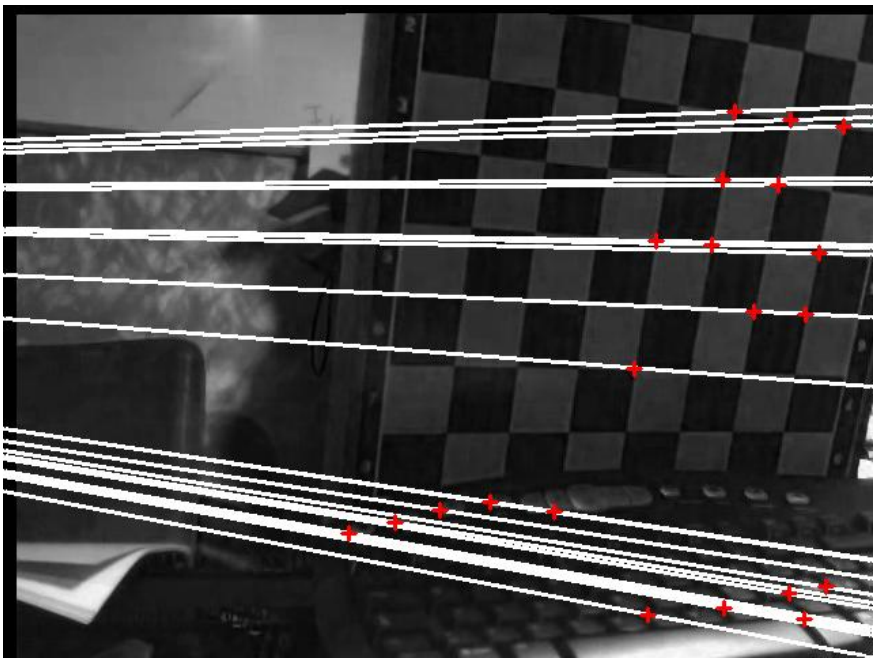

Figura 4.2: Prima immagine associata alla prima stima.

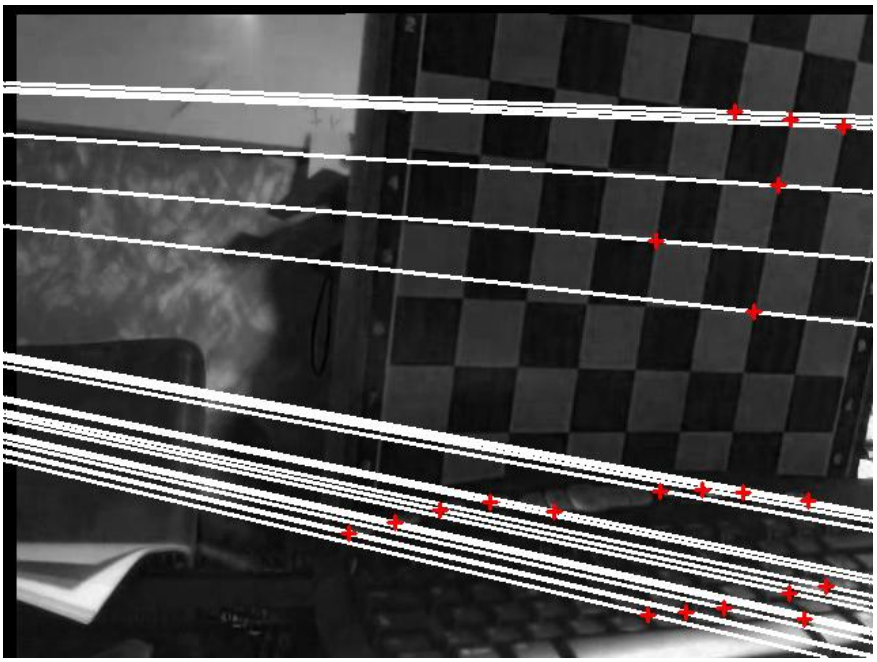

Figura 4.3: Prima immagine associata alla seconda stima.

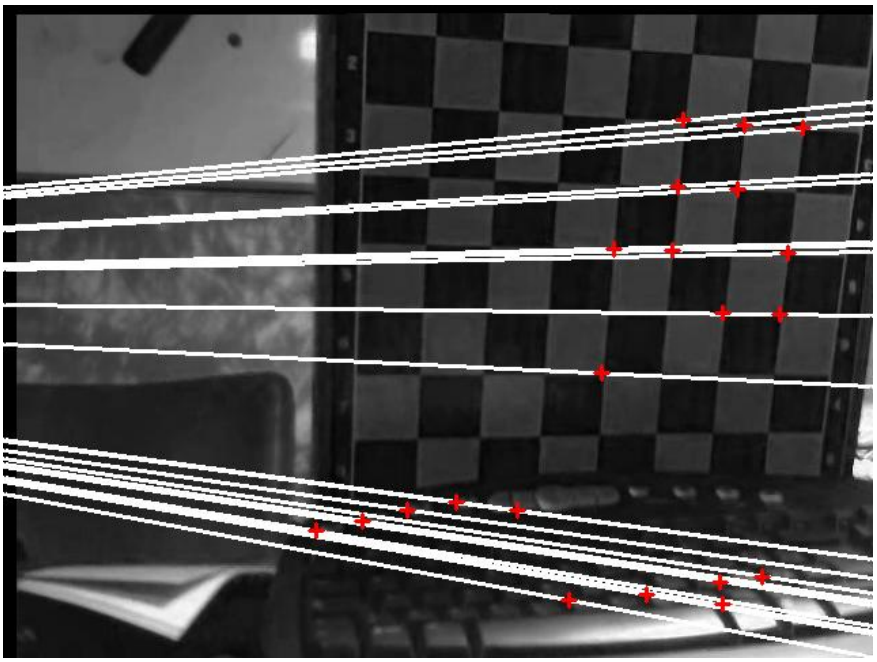

Figura 4.4: Seconda immagine associata alla prima stima.

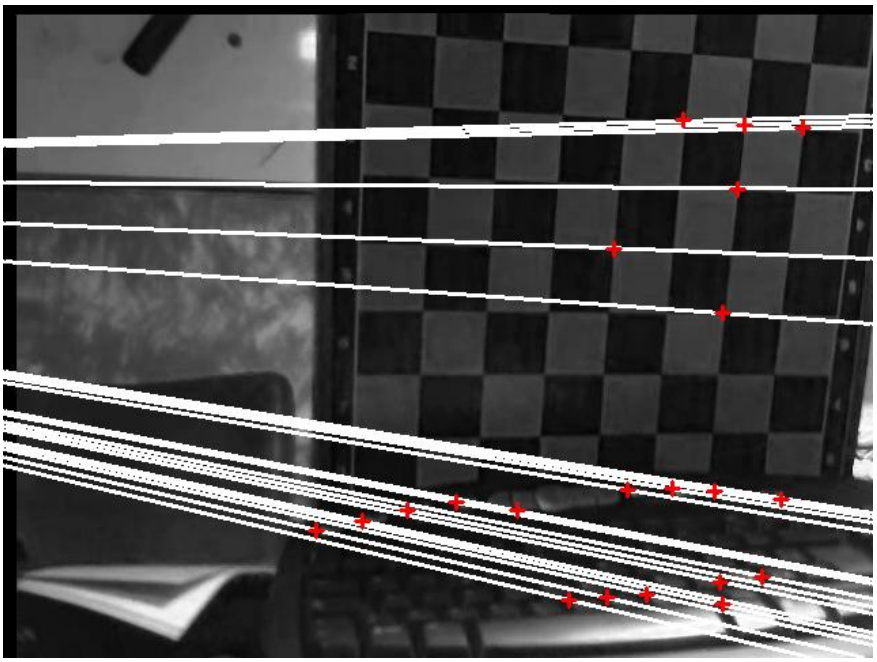

Figura 4.5: Seconda immagine associata alla seconda stima.

4.2.2 Moto vincolato

Alcuni movimenti *speciali* della telecamera, o la conoscenza parziale della calibrazione, consentono si semplificare il calcolo della matrice fondamentale riducendo il numero di gradi di libertà ad essa associati. In particolare, nel moto puramente traslatorio, la matrice fondamentale assume la forma semplificata $F \simeq [\mathbf{e}']_\times$, che richiede il calcolo delle coordinate di $\mathbf{e}'$. In questo caso sono sufficienti, per la stima di F, due sole corrispondenze. L'aspetto più interessante, alla luce dei problemi di degenerazione riscontrati nel funzionamento del sistema di visione, è rappresentato dalla riduzione delle configurazioni degeneri al solo caso in cui i due punti associati alle corrispondenze siano coplanari con i due centri delle viste. Nel moto traslatorio, ogni corrispondenza produce un'equazione del tipo:

$$\mathbf{x}'^T[\mathbf{e}']_\times\mathbf{x} = 0 \Leftrightarrow \begin{bmatrix} x_2x'_3 - x'_2x_3 & x'_1x_3 - x_1x'_3 & x_1x'_2 - x'_1x_2 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = 0 \qquad (4.2.5)$$

La funzione implementata *cvxFindFundamantalTranslation* accumula le equazioni (4.2.5) e le risolve attraverso un algoritmo RANSAC-DLT. Il vincolo che F sia una matrice antisimmetrica rende impraticabile la normalizzazione dei dati. Facendo riferimento alla (4.2.4), si osservi che la trasformazione *implicita* (poiché eseguita a monte dell'effettivo calcolo) della matrice fondamentale $T'^{-T}[\mathbf{e}']_\times T^{-1} \rightarrow \overline{F}$, ne modifica la struttura antisimmetrica, rendendo impossibile la stima di $\overline{F}$ sulla base dell'equazione (4.2.5). Nonostante l'assenza di normalizzazione, sperimentalmente si è potuta constatare la bontà dei risultati della *cvxFindFundamantalTranslation* (probabilmente la riduzione dello spazio dei parametri porta ad un'attenuazione dei fenomeni di mal condizionamento).

4.3 Ambiguità nella ricostruzione

Il sistema di visione in esame è caratterizzato da una telecamera i cui parametri intrinseci non variano nel corso della simulazione. Nel caso in cui tali parametri non siano noti, o non siano noti la posizione o l'orientamento relativi di almeno due viste distinte, il problema della ricostruzione può essere risolto solo *a meno di una trasformazione proiettiva*. Ad esempio, un quadrato nella scena potrebbe essere ricostruito come un generico parallelogramma, ed un cerchio come un'ellisse. In questo caso specifico, l'ambiguità nella ricostruzione sarebbe di tipo *affine*, un particolare tipo di *ambiguità proiettiva*. Una più generica trasformazione proiettiva modificherebbe la posizione del piano all'infinito, rendendo molto più difficile capire come una semplice figura geometrica venga trasformata. È importante porre l'attenzione sul fatto che i parametri intrinseci ed estrinseci utilizzati per risolvere l'ambiguità, non necessariamente devono essere noti all'inizio della simulazione, magari ottenuti applicando un particolare algoritmo fuori linea o misurando fisicamente delle grandezze. Esistono numerosi metodi per ottenere una stima *in linea* di tali parametri, che rientrano nell'insieme dei metodi di *autocalibrazione*. Nel presente progetto si è scelto di stimare i parametri intrinseci fuori linea, come riportato nella sezione 3.3, ed i parametri estrinseci in linea (l'unica cosa ragionevole in questa particolare applicazione), come riportato nella sezione 5.2. Una ricostruzione caratterizzata da ambiguità proiettiva viene anche definita *ricostruzione proiettiva*. Con modalità analoga si definiscono le ricostruzioni associate agli altri tipi di ambiguità. Il fine ultimo è il raggiungimento di una *ricostruzione metrica*, priva di ambiguità a meno di un inevitabile fattore di scala (per imporre il quale è sufficiente definire la lunghezza della baseline tra due generiche viste). Il sistema di visione realizzato si basa su una ricostruzione *stratificata*. In una

prima fase viene generata una ricostruzione proiettiva, e contemporaneamente vengono stimati i parametri estrinseci associati alle prime due viste. In una seconda fase, prima che i risultati vengano forniti all'utente, la ricostruzione proiettiva viene convertita in una ricostruzione metrica. Considerato che i sistemi di ricostruzione sono inevitabilmente soggetti alla propagazione di errori di varia natura, il fatto di non includere tra questi anche gli errori sulla stima dei parametri intrinseci ed estrinseci migliora notevolmente i risultati.

4.4 Ricostruzione dell'ambiente

La *triangolazione*, algoritmo alla base della ricostruzione ambientale, consente di ricavare le coordinate di un punto nella scena a partire dalle sue proiezioni $\mathbf{x_1}$ e $\mathbf{x_2}$ sui piani immagine di due viste, e dalle corrispondenti matrici di proiezione P_1 e P_2. Si ipotizzi che tali matrici siano riferite ad una ricostruzione proiettiva del moto. Ciò significa che esiste un'omografia H tale che $\overline{P}_1 \simeq P_1 H$ e $\overline{P}_2 \simeq P_2 H$, con $\overline{P}_1$ e $\overline{P}_2$ matrici di proiezione *reali* (riferite ad una ricostruzione metrica del moto). Se la triangolazione si basasse sui dati $\mathbf{x_1}$, $\mathbf{x_2}$, P_1 e P_2, allora anche le coordinate $\mathbf{X}$ del punto nella scena si otterrebbero riferite ad una ricostruzione proiettiva dell'ambiente. Il legame di $\mathbf{X}$ con le coordinate $\hat{\mathbf{X}}$ *reali* (riferite ad una ricostruzione metrica dell'ambiente) del punto sarebbe esprimibile dalla relazione $\hat{\mathbf{X}} \simeq H^{-1}\mathbf{X}$. Nella sezione 4.4.1 si affronta il problema della triangolazione nel caso siano note le matrici di proiezione reali, nella sezione 4.4.2 nel caso siano note in una ricostruzione proiettiva.

4.4.1 Triangolazione nella ricostruzione metrica

Il metodo della triangolazione si basa sulle proprietà trigonometriche del triangolo, e consente di determinarne la posizione di un vertice una volta note le posizioni dei

vertici rimanenti e dei corrispondenti angoli. Nella letteratura matematica, l'approccio geometrico è quello più comunemente utilizzato nella formalizzazione del metodo. In tale ottica, la triangolazione esprime il legame tra l'altezza di un triangolo, la sua base e gli angoli ad essa adiacenti:

$$RC = \frac{AB \cdot \sin(\alpha) \cdot \sin(\beta)}{\sin(\alpha + \beta)} \tag{4.4.1}$$

Al fine di garantire una continuità con i formalismi ed i concetti finora introdotti, si

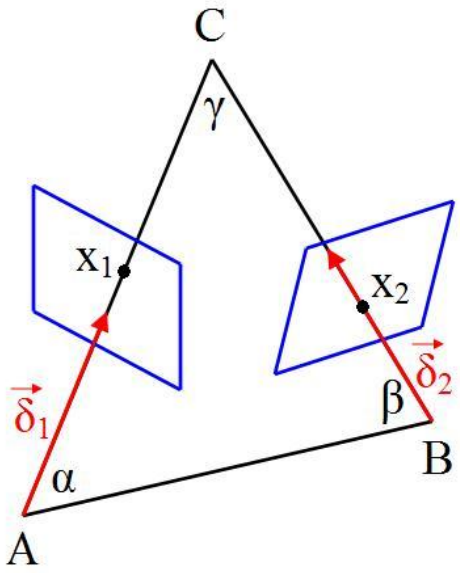

Figura 4.6: Seconda immagine associata alla seconda stima.

utilizzi la notazione di figura 4.6. Si ipotizzi l'esistenza di due viste centrate in A e B. Alle due viste sono associate rispettivamente le matrici di calibrazione K_1 e K_2, ipotizzate note. Siano anche noti l'orientamento R e la posizione $\mathbf{t}$ relativi dei sistemi di riferimento locali. Il punto C si proietta sui due piani immagine rispettivamente nelle coordinate $\mathbf{x_1}$ ed $\mathbf{x_2}$, anch'esse note. Le coordinate $\mathbf{d_1}$ del vettore normalizzato $\overrightarrow{\delta_1}$ nel riferimento locale della prima vista sono uguali, a meno di un fattore di scala, a $K_1^{-1}\mathbf{x_1}$ (sezione 3.2.2). Analogamente, le coordinate $\mathbf{d_2}$ del vettore normalizzato $\overrightarrow{\delta_2}$

nel riferimento locale della seconda vista sono uguali, a meno di un fattore di scala, a $K_2^{-1}\mathbf{x_2}$. Se R esprime la rotazione che dal riferimento locale della prima vista porta al riferimento locale della seconda, e $\mathbf{t}$ le coordinate del vettore $\overrightarrow{AB}$ nel riferimento locale della prima vista, allora il problema della triangolazione si riduce alla risoluzione del seguente sistema lineare nelle incognite $\|\overrightarrow{BC}\|$ e $\|\overrightarrow{AC}\|$:

$$\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC} \Rightarrow \mathbf{t} + \|\overrightarrow{BC}\|R^T\mathbf{d_2} = \|\overrightarrow{AC}\|\mathbf{d_1} \tag{4.4.2}$$

Il sistema risulta mal condizionato nel caso in cui $\|\mathbf{t}\|$ sia piccola rispetto a $\|\overrightarrow{BC}\|$ e $\|\overrightarrow{AC}\|$. In presenza di mal condizionamento, il minimo rumore nella misura delle coordinate $\mathbf{x_1}$ o $\mathbf{x_2}$, e quindi il minimo rumore nelle coordinate $\mathbf{d_1}$ o $\mathbf{d_2}$ da esse ricavate, conduce ad un grande errore nel calcolo di $\|\overrightarrow{BC}\|$ e $\|\overrightarrow{AC}\|$. Si tratta di un concetto che non dovrebbe sorprendere, poiché anche nell'uomo l'apparato visivo è soggetto a questa forma di mal condizionamento. La baseline di circa 6 centimetri tra occhio destro e sinistro, infatti, consente di stimare con buona precisione solo la distanza di oggetti sufficientemente vicini.

Il rumore nella misura di $\mathbf{x_1}$ ed $\mathbf{x_2}$ produce errori rilevanti in presenza di mal condizionamento. Sebbene gli errori prodotti in assenza di mal condizionamento siano di minor importanza, è bene implementare un algoritmo di triangolazione che tenda a minimizzarli. Nel caso di dati perfetti, la triangolazione calcola l'intersezione tra due rette, per ognuna delle quali sono note la posizione di un punto e la direzione. In presenza di rumore, le rette sono in generale *sghembe*, e non ha quindi senso cercarne l'intersezione. Il modo migliore di agire consiste nel cercare il punto intermedio sul segmento di minima distanza che congiunge le rette. Quest'ultimo procedimento non è attuabile nel caso si utilizzi una ricostruzione proiettiva del moto, cioè le matrici di proiezione siano note a meno di una trasformazione proiettiva, poiché il concetto

di distanza perderebbe di significato. L'algoritmo effettivamente implementato nel sistema di visione è descritto nella sezione 4.4.2.

4.4.2 Triangolazione nella ricostruzione proiettiva

A prescindere dal tipo di ricostruzione delle matrici di proiezione, il vincolo epipolare $\mathbf{x_2}^T F \mathbf{x_1} = 0$ garantisce che le rette utilizzate nell'ambito della triangolazione siano incidenti. Dati rumorosi determinano in generale $\mathbf{x_2}^T F \mathbf{x_1} \neq 0$. L'algoritmo *ottimo* di triangolazione valuta i punti $\hat{\mathbf{x}}_\mathbf{1}$ ed $\hat{\mathbf{x}}_\mathbf{2}$ più vicini ad $\mathbf{x_1}$ ed $\mathbf{x_2}$, e tali da soddisfare il vincolo epipolare $\hat{\mathbf{x}}_\mathbf{2}^T F \hat{\mathbf{x}}_\mathbf{1} = 0$. Utilizza poi $\hat{\mathbf{x}}_\mathbf{1}$ ed $\hat{\mathbf{x}}_\mathbf{2}$ per triangolare in assenza di rumore. Solitamente la minimizzazione di distanze geometriche richiede l'impiego di tecniche risolutive iterative. In questo particolare caso, il problema di minimizzazione si traduce in un problema di ricerca delle radici di un polinomio di sesto grado. Come primo passo devono essere trasformate le coordinate sui due piani immagine, in modo tale che i punti $\mathbf{x_1} = (x_1, y_1)$ ed $\mathbf{x_2} = (x_2, y_2)$ ne diventino le rispettive origini. Si possono utilizzare a tal fine le matrici:

$$T_1 = \begin{bmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{bmatrix} \qquad T_2 = \begin{bmatrix} 1 & 0 & -x_2 \\ 0 & 1 & -y_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4.3}$$

che agiscono su coordinate omogenee. La matrice fondamentale deve essere trasformata tenendo conto delle nuove coordinate, in particolare $T_2^{-T} F T_1^{-1} \rightarrow F$. Si ruotino a questo punto le coordinate, in modo tale che gli epipoli vadano a sovrapporsi alle ascisse delle rispettive immagini. In [8] è spiegato nel dettaglio come calcolare le necessarie matrici di rotazione R_1 ed R_2. La matrice fondamentale deve essere nuovamente aggiornata per tenere conto delle trasformazioni, quindi $R_2 F R_1^T \rightarrow F$. A questo punto F dovrebbe poter essere scritta nel modo seguente:

$$F = \begin{bmatrix} f_1 f_2 d & -f_2 c & -f_2 d \\ -f_1 b & a & b \\ -f_1 d & c & d \end{bmatrix} \tag{4.4.4}$$

Si ponga in f_1 l'inverso della norma del primo epipolo ed in f_2 l'inverso della norma del secondo epipolo. Le norme devono rappresentare lunghezze "vere", quindi devono essere riferite a coordinate disomogenee. Si fissino inoltre $a = F_{22}$, $b = F_{23}$, $c = F_{32}$ e $d = F_{33}$, coerentemente con la 4.4.4, e si risolva il polinomio:

$$g(t) = t((at+b)^2 + f_2^2(ct+d)^2)^2 - (ad-bc)(1+f_1^2t^2)^2(at+b)(ct+d) = 0 \quad (4.4.5)$$

OpenCV mette a disposizione *cvSolvePoly* per il calcolo delle radici, funzione che si basa sulla stima degli autovalori di una matrice in forma *compagna* costruita utilizzando i coefficienti del polinomio. Le sei radici devono essere valutate al fine di stabilire quale di esse minimizzi la funzione:

$$s(t) = \frac{t^2}{1+f_1^2t^2} + \frac{(ct+d)^2}{(at+b)^2 + f_2^2(ct+d)^2} \quad (4.4.6)$$

È anche possibile che $s(t)$ raggiunga il minimo assoluto asintoticamente. In tal caso si sceglie $t_{min} = \infty$. A questo punto dell'algoritmo sono note le rette epipolari più vicine ai punti di partenza, rappresentate dai vettori $l_1 = \begin{pmatrix} t_{min}f_1 & 1 & -t_{min} \end{pmatrix}^T$ ed $l_2 = F \begin{pmatrix} 0 & t_{min} & 1 \end{pmatrix}^T$. Poiché le trasformazioni T_i hanno spostato i punti $\mathbf{x_1}$ ed $\mathbf{x_2}$ nelle rispettive origini, i punti cercati $\hat{\mathbf{x}}_\mathbf{1}$ ed $\hat{\mathbf{x}}_\mathbf{2}$ possono essere facilmente ricavati considerando che il punto più vicino all'origine e collocato sulla generica retta $\begin{pmatrix} \lambda & \mu & \nu \end{pmatrix}^T$ ha coordinate omogenee $\begin{pmatrix} -\lambda\nu & -\mu\nu & \lambda^2+\mu^2 \end{pmatrix}^T$. Inoltre il calcolo di $\hat{\mathbf{x}}_\mathbf{1}$ ed $\hat{\mathbf{x}}_\mathbf{2}$ impone il ritorno alle coordinate di partenza attraverso $T_1^{-1}R_1^T$ e $T_2^{-1}R_2^T$. Si prosegue a questo punto con la triangolazione vera a propria. A partire dalle relazioni $\hat{\mathbf{x}}_\mathbf{1} \simeq P_1\hat{\mathbf{X}}$ e $\hat{\mathbf{x}}_\mathbf{2} \simeq P_2\hat{\mathbf{X}}$, si può ricavare un sistema lineare ed omogeneo negli elementi di $\hat{\mathbf{X}}$ (stesso metodo utilizzato per la stima delle omografie), risolvibile tramite DLT. Si noti che non è necessaria alcuna minimizzazione iterativa successiva, in quanto i dati di partenza sono esatti ed errore algebrico e geometrico sono entrambi nulli. L'algorimo di triangolazione è stato implementato nella funzione *cvxOptimal2DTo3D*.

4.5 Calcolo della matrice di proiezione

4.5.1 Dalla matrice fondamentale alla matrice di proiezione

La relazione (4.2.2) mostra come ricavare la matrice fondamentale partendo da K, R e $\mathbf{t}$. Alternativamente, F può essere calcolata sfruttando la conoscenza delle matrici di proiezione:

$$F \simeq [\mathbf{e}']_{\times} P' P^{+} \tag{4.5.1}$$

dove P^{+} è la pseudoinversa di P, ed $\mathbf{e}'$ è uno dei due epipoli. Sebbene nella (4.5.1) non compaiano le sole matrici di proiezione, l'epipolo può essere facilmente calcolato come funzione di P e P', essendo la proiezione del centro della prima vista sul piano immagine della seconda:

$$P\mathbf{C} = \mathbf{0} \xrightarrow{\mathrm{C}} \mathbf{e}' \simeq P'\mathbf{C} \tag{4.5.2}$$

in cui $\mathbf{C}$ è il centro della prima vista espresso nelle coordinate omogenee del riferimento globale. Si noti che il risultante vettore $\mathbf{0}$ non ha senso in termini di coordinate omogenee, infatti il centro vista è l'unico punto dello spazio proiettivo a non poter essere proiettato sul piano immagine della corrispondente vista. Il suo significato è quindi puramente algebrico. Le relazioni (4.5.1) e (4.5.2) mostrano come ricavare F *univocamente* a partire dalle matrici di proiezione. L'operazione non è però biiettiva, poiché ad una stessa F corrispondono infinite coppie $\{P, P'\}$, uguali (in termini omogenei) a meno di una omografia H dello spazio proiettivo. Dato che $\mathbf{x} \simeq P\mathbf{X} \simeq (PH)(H^{-1}\mathbf{X})$, ed analogamente $\mathbf{x}' \simeq P'\mathbf{X} \simeq (P'H)(H^{-1}\mathbf{X})$, le coordinate $\mathbf{x}$ ed $\mathbf{x}'$ sono le stesse sia che venga utilizzata la coppia $\{P, P'\}$, sia che venga utilizzata la coppia $\{PH, P'H\}$. Quindi nei due casi, utilizzando un sufficiente numero di corrispondenze $\{\mathbf{x}, \mathbf{x}'\}$, si ricava la stessa matrice F. Ma tale matrice la si può calcolare equivalentemente usando o la coppia $\{P, P'\}$ o la coppia $\{PH, P'H\}$, che forniscono quindi la stessa matrice

fondamentale, come volevasi dimostrare. Scegliendo opportunamente H, è possibile trasformare lo spazio proiettivo in modo che le matrici di proiezione assumano una *forma canonica*:

$$\{P, P'\} \rightarrow \{\left[\ I \mid \mathbf{0}\ \right], \left[\ M \mid \mathbf{m}\ \right]\} \tag{4.5.3}$$

Esistono infinite omografie che trasformano la coppia $\{P, P'\}$ in forma canonica, ed esistono infinite forme canoniche associate ad una stessa matrice F, genericamente rappresentabili con la coppia:

$$\{\left[\ I \mid \mathbf{0}\ \right], \left[\ [\mathbf{e}']_\times F + \mathbf{e}'\mathbf{v}^T \mid \lambda \mathbf{e}'\ \right]\} \tag{4.5.4}$$

I vettori $\mathbf{e}'$ e $\mathbf{v}$, e lo scalare $\lambda \neq 0$, possono essere scelti arbitrariamente. La simbologia non è casuale, ad esempio si può verificare come $\mathbf{e}'$ sia effettivamente l'epipolo sul piano immagine della seconda vista. Concludendo, date due matrici di proiezione nella generica forma (4.5.3), il calcolo della corrispondente matrice fondamentale risulta particolarmente semplice, poiché $F \simeq [\mathbf{m}]_\times M$.

4.5.2 Dalla ricostruzione dell'ambiente alla matrice di proiezione

Nella sezione 4.4 si è descritto come ottenere la ricostruzione dell'ambiente a partire dalla ricostruzione del moto attraverso la triangolazione. La presente sezione introduce il problema diametralmente opposto. Note le coordinate di un sufficiente numero di punti nella scena, e le corrispondenti proiezioni sul piano immagine di una vista, si vuole determinare la matrice di proiezione ad essa associata. Anche in questo caso, a coordinate dei punti riferite ad una ricostruzione proiettiva corrisponderà una matrice di proiezione riferita ad una ricostruzione proiettiva, secondo le formule già presentate nella sezione 4.4. L'algoritmo non introduce alcuna novità rispetto a quanto descritto

nelle precedenti sezioni. Ad ogni corrispondenza $\{\mathbf{x}, \mathbf{X}\}$ è associata una relazione:

$$\mathbf{x} \simeq P\mathbf{X} \tag{4.5.5}$$

con P matrice incognita. Come già visto nella stima delle omografie (sezione 3.3.1), la relazione (4.5.5) può essere trasformata attraverso il prodotto vettoriale di ambo i membri per $\mathbf{x}$. In tal modo si ottengono tre equazioni omogenee e lineari nelle componenti di P, di cui in generale due indipendenti. Accumulando sufficienti equazioni, la matrice di proiezione può essere stimata applicando il DLT, magari associato al RANSAC. Come ampiamente descritto nella sezione 4.2, anche in questo caso converrebbe normalizzare le coordinate prima dell'esecuzione dell'algoritmo, per poi *denormalizzare* il risultato come ultimo passaggio, al fine di ridurre eventuali fenomeni di mal condizionamento. La normalizzazione non sarebbe praticabile nel caso in cui nella scena fossero contemporaneamente presenti punti vicini al centro della vista e punti da esso lontani. Ciò comporterebbe infatti l'addensarsi nel centroide di tutti i punti vicini, e probabilmente la perdita delle informazioni ad essi associate a causa delle approssimazioni numeriche. Si tratta di un problema che riguarda esclusivamente i punti nella scena, non essendo vincolati a giacere all'interno di un'immagine.
OpenCV non include una funzione per la stima della matrice di proiezione, motivo per cui è stata implementata la funzione *cvxFindProjectionMatrix*, basata proprio sull'algoritmo DLT-RANSAC normalizzato. Vista la bontà dei risultati, non si è ritenuto necessario introdurre un'ulteriore fase di minimizzazione iterativa di un errore geometrico.

4.6 Integrazione degli algoritmi

Nella sezione 4.3 si è fatto cenno alla struttura *stratificata* del sistema implementato. La ricostruzione proiettiva e gli algoritmi che ad essa conducono ([23]) sono trattati nella presente sezione. Il passaggio dalla ricostruzione proiettiva alla ricostruzione metrica è argomento del capitolo 5. La prima fase della ricostruzione proiettiva consiste nella ricerca di una matrice fondamentale che associ la prima vista ad una vista *successiva*. Perché non scegliere come vista successiva direttamente la seconda? Trattandosi di una sequenza di immagini acquisite da telecamera, la seconda vista potrebbe essere molto vicina alla prima, e questo produrrebbe difficoltà nella stima di F dovute a *degenerazione* dei dati. Stabilire qualora una particolare configurazione delle viste e dei punti nella scena rappresenti una configurazione degenere non è cosa semplice. Nel sistema realizzato, la selezione della vista spetta ad un opportuno segnale esterno. Tale segnale potrebbe provenire da un utente umano, oppure potrebbe essere generato automaticamente sulla base di informazioni *odometriche*. Ricapitolando, l'algoritmo ha inizio con la selezione di punti di interesse nella prima immagine, inseguiti nella sequenza video fino all'arrivo di un segnale esterno che avvia il calcolo della matrice fondamentale. Tale calcolo può essere eseguito dalla funzione *cvFindFundamental*, oppure dalla funzione *cvxFindFundamentalTranslation*, nel solo caso in cui le due viste siano caratterizzate dallo stesso orientamento. Sfruttando le nozioni introdotte nella sezione 4.5.1, ed in particolare imponendo $\mathbf{v} = \mathbf{0}$ e $\lambda = 1$ nella forma canonica (4.5.4), dalla matrice fondamentale si ricavano le seguenti matrici di proiezione:

$$\{ \left[\begin{array}{c|c} I & \mathbf{0} \end{array} \right] , \left[\begin{array}{c|c} [\mathbf{e}']_{\times} F & \mathbf{e}' \end{array} \right] \} \tag{4.6.1}$$

Le coordinate $\mathbf{e}'$ dell'epipolo si ottengono calcolando il kernel sinistro di F, quindi la (4.6.1) è completamente determinata dalla matrice fondamentale. Le matrici di proiezione vengono utilizzate per generare una ricostruzione proiettiva dell'ambiente, sfruttando le corrispondenze nelle immagini delle rispettive viste. L'algoritmo impiegato è quello descritto nella sezione 4.4.2, ed implementato nella funzione *cvxOptimal2Dto3D*. In occasione della triangolazione, per fare in modo che la baseline sia sufficientemente lunga, si è deciso di utilizzare dati provenienti dalla ricostruzione metrica. I punti le cui coordinate nella scena sono state stimate, continuano ad essere inseguiti nelle immagini successive, consentendo di calcolare nuove matrici di proiezione grazie alla funzione *cvxFindProjectionMatrix*. Contemporaneamente, per ogni nuova immagine, vengono rilevati nuovi punti di interesse, in modo che possano essere inseguiti ed utilizzati, al momento opportuno, per ampliare la ricostruzione dell'ambiente. Il programma si basa quindi su un'alternanza ciclica di ricostruzione dell'ambiente e ricostruzione del moto.

Capitolo 5

Dalla ricostruzione proiettiva alla ricostruzione metrica

5.1 Calcolo della matrice essenziale

La *matrice essenziale* è una specializzazione della matrice fondamentale, ottenuta *normalizzando* le coordinate dei punti sui piani immagine. A differenza della normalizzazione introdotta nelle sezioni precedenti per risolvere problemi di mal condizionamento, questa si basa sulla trasformazione $\hat{\mathbf{x}} \simeq K^{-1}\mathbf{x}$. Si applichi tale trasformazione all'equazione del vincolo epipolare:

$$\mathbf{x_2}^T F \mathbf{x_1} = 0 \Leftrightarrow \hat{\mathbf{x}}_\mathbf{2}^T K_2^T F K_1 \hat{\mathbf{x}}_\mathbf{1} = 0 \Leftrightarrow \hat{\mathbf{x}}_\mathbf{2}^T E \hat{\mathbf{x}}_\mathbf{1} = 0 \tag{5.1.1}$$

Dalla (5.1.1) emerge una prima definizione $E \simeq K_2^T F K_1$. La matrice essenziale può essere espressa alternativamente dalla relazione $E \simeq [\mathbf{t}]_\times R$, già introdotta nella sezione 3.2.2 nell'ambito dell'analisi sui gradi di libertà effettivi. Tale definizione mostra come la matrice essenziale dipenda esclusivamente dalla posizione e dall'orientamento relativi tra le viste ad essa associate. Nota la matrice essenziale, R può essere ricavata univocamente, mentre $\mathbf{t}$ a meno di un fattore di scala (in realtà esiste sia per R che per $\mathbf{t}$ un'ulteriore ambiguità, facilmente eliminabile in entrambi i casi come verrà mostrato nella sezione 5.2). Esprimendo $\mathbf{t}$ delle coordinate disomogenee, l'arbitrarietà nel

fattore di scala si traduce nell'arbitrarietà nella scelta della lunghezza della baseline. Si ha quindi una perdita di informazione nel passaggio $\{R, \mathbf{t}\} \rightarrow E$, giustificabile con la perdita di un grado di libertà discussa nella sezione 3.2.2. Si può dimostrare che una matrice 3×3 è una matrice essenziale se e solo se ha due valori singolari identici ed il terzo è nullo.

5.2 Calcolo della rotazione e della traslazione

Due metodi alternativi per il calcolo di R e $\mathbf{t}$ sono descritti dettagliatamente in [22] e [7]. L'approccio algebrico utilizzato in [7], in particolare, risulta molto utile ai fini di un'applicazione immediata. Si definisca la matrice W nel modo seguente:

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2.1}$$

e si decomponga in valori singolari la matrice essenziale:

$$E \simeq U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T \tag{5.2.2}$$

dove U e V sono matrici ortogonali. Si imponga, arbitrariamente, che il vettore $\mathbf{t}$ abbia norma unitaria. Esistono allora quattro coppie $\{R, \mathbf{t}\}$ associate alla stessa matrice essenziale, ed in particolare:

$$\{UWV^T, \mathbf{u_3}\} \qquad \{UWV^T, -\mathbf{u_3}\} \qquad \{UW^TV^T, \mathbf{u_3}\} \qquad \{UW^TV^T, -\mathbf{u_3}\}$$

dove $\mathbf{u_3} = U \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$. Sebbene, dal punto di vista matematico, tutte e quattro le coppie siano associate alla stessa matrice essenziale E, e quindi siano da ritenersi *corrette* in tal senso, solo una delle alternative vanta un significato *pratico*. Si ipotizzi che $\mathbf{x_1}$ ed $\mathbf{x_2}$ siano punti corrispondenti sui piani immagine di due viste, e che sia nota la matrice essenziale E ad esse associata. Per ogni coppia $\{R, \mathbf{t}\}$ possibile, si

determinino le matrici di proiezione $P_1 = \left[\ I \mid \mathbf{0}\ \right]$ e $P_2 = \left[\ R \mid \mathbf{t}\ \right]$, e si calcolino le coordinate $\mathbf{X}$ del punto di proiezioni $\mathbf{x_1}$ ed $\mathbf{x_2}$ (si veda la sezione 4.4.2). Il fatto di non includere i parametri intrinseci in P_1 e P_2 non è rilevante ai fini della presente analisi. In generale, alle quattro alternative corrispondono differenti coordinate $\mathbf{X}$, cioè punti diversamente collocati nella scena. Solo in un caso, però, il punto ha profondità positiva rispetto ad entrambe le viste. La corrispondente coppia $\{R, \mathbf{t}\}$ fornisce l'orientamento e la posizione cercati.

La funzione *cvxEssentialToRt* riceve in ingresso la matrice fondamentale, la matrice di calibrazione ed una lista di coppie $\{\mathbf{x_1}, \mathbf{x_2}\}$. Come primo passo calcola la matrice essenziale utilizzando la (5.1.1). Dalla matrice essenziale vengono ricavate le quattro possibili coppie $\{R, \mathbf{t}\}$. Si ipotizzi di scegliere $\{UWV^T, \mathbf{u_3}\}$, corrispondente alle matrici di proiezione $P_1 = \left[\ I \mid \mathbf{0}\ \right]$ e $P_2 = \left[\ UWV^T \mid \mathbf{u_3}\ \right]$. Per ogni coppia $\{\mathbf{x_1}, \mathbf{x_2}\}$ nella lista di corrispondenze tra punti immagine, viene calcolata la posizione del relativo punto nella scena utilizzando la funzione *cvxOptimal2DTo3D*. Per ogni punto nella scena, vengono poi calcolate le profondità rispetto alle due viste, utilizzando la funzione *cvxDepth*. Un contatore memorizza in quanti casi siano positive entrambe le profondità. Tale valore, confrontato con i corrispondenti valori derivanti dall'applicazione dello stesso procedimento alle tre rimanenti coppie $\{R, \mathbf{t}\}$, consente di stabilire in modo *robusto* quali siano la posizione e l'orientamento cercati.

Per quanto riguarda il calcolo della profondità di un punto rispetto ad una vista, si ipotizzi che valgano le relazioni di *uguaglianza* $\mathbf{x} = \omega \begin{pmatrix} x & y & 1 \end{pmatrix}^T = P \begin{pmatrix} X & Y & Z & T \end{pmatrix}^T = P\mathbf{X}$. La matrice di proiezione P può essere riscritta come $\left[\ M \mid \mathbf{p_4}\ \right]$. La profondità di $\mathbf{X}$ rispetto alla vista di matrice P, può essere calcolata attraverso la formula seguente:

$$depth(\mathbf{X}; P) = \frac{sign(det(M))\omega}{T\|\mathbf{m_3}\|} \tag{5.2.3}$$

dove $\mathbf{m_3}$ è la terza riga di M.

5.3 Calcolo del fattore di scala e della proiezione del piano all'infinito

Le matrici di proiezione (4.6.1), ricavate dalla matrice fondamentale associata alle prime due viste *sufficientemente lontane*, sono state utilizzate nella sezione 4.6 come punto di partenza per gli algoritmi di ricostruzione. Se al loro posto si fossero utilizzate le matrici:

$$\{K \left[\begin{array}{c|c} I & \mathbf{0} \end{array} \right], K \left[\begin{array}{c|c} R & \mathbf{t} \end{array} \right]\} \tag{5.3.1}$$

nelle quali K è la matrice di calibrazione ottenuta con il procedimento fuori linea descritto nella sezione 3.3, ed R e $\mathbf{t}$ sono la posizione e l'orientamento relativi ottenuti a partire dalla stessa F con l'ausilio degli algoritmi del presente capitolo, si sarebbe ottenuta fin dalla sezione 4.6 una ricostruzione *metrica* dell'ambiente e del moto. Gli errori nella stima di K, R e $\mathbf{t}$, sono il motivo per il quale si è preferita attuare una ricostruzione *stratificata*. Nella ricostruzione stratificata, infatti, tali errori non sono soggetti a propagazione, in quanto applicati ai risultati in uscita. Sebbene R e $\mathbf{t}$ siano il risultato di un calcolo puramente algebrico, sono soggetti ad errori dovuti alla non corretta stima di F e K, che conduce alla matrice essenziale *inesatta* $\tilde{E} = Udiag\{\alpha, \beta, \gamma\}V^T$, con α e β non nulli e *quasi* uguali, e γ *quasi* non nullo. A titolo informativo, si noti che la *matrice essenziale* più vicina ad $\tilde{E}$, in termini di norma di Frobenius, si dimostra essere $E = Udiag\{\frac{\alpha+\beta}{2}, \frac{\alpha+\beta}{2}, 0\}V^T$. Questo passaggio migliorativo non è richiesto nell'algoritmo implementato per il caloclo di R e $\mathbf{t}$, in quanto sono utilizzate a tal fine le sole matrici U e V.

A differenza di una ricostruzione puramente metrica, che si baserebbe sulle matrici di proiezione (5.3.1), la ricostruzione stratificata richiede il calcolo di un'omografia H

che consenta il passaggio dalla ricostruzione proiettiva alla ricostruzione metrica, utilizzando le formule introdotte nella sezione 4.4. Tale omografia non dipende in modo *esplicito* da R e $\mathbf{t}$, ma dipende dalla *proiezione del piano all'infinito*, $\mathbf{p}$, e dal *fattore di scala* k, comunque calcolabili a partire da K, R e $\mathbf{t}$. La struttura della matrice omografica si può ricavare sulla base di semplici considerazioni. Le prime due matrici di proiezione, nella ricostruzione proiettiva, hanno forma canonica, aspetto finora evidenziato in diverse occasioni. Si ipotizzi che, nella ricostruzione metrica cercata, il riferimento globale coincida con il riferimento locale della prima vista, e che quindi la matrice di proiezione ad essa associata sia $K \left[\begin{array}{c|c} I & \mathbf{0} \end{array} \right]$. La matrice H dovrebbe essere tale da soddisfare la seguente relazione:

$$\left[\begin{array}{c|c} I & \mathbf{0} \end{array} \right] H = K \left[\begin{array}{c|c} I & \mathbf{0} \end{array} \right] \tag{5.3.2}$$

che implica:

$$H = \begin{bmatrix} K & \mathbf{0} \\ \mathbf{v^T} & k \end{bmatrix} \tag{5.3.3}$$

nella quale $\mathbf{v}$ e $k \neq 0$ devono essere determinati. L'omografia trasforma il piano all'infinito, rappresentato dalle coordinate $\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T$, nel modo seguente:

$$\begin{bmatrix} K & \mathbf{0} \\ \mathbf{v^T} & k \end{bmatrix}^{-T} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} K^{-T} & -K^{-T}\mathbf{v}/k \\ \mathbf{0} & 1/k \end{bmatrix} = \frac{1}{k} \begin{pmatrix} -K^{-T}\mathbf{v} \\ 1 \end{pmatrix} = \frac{1}{k} \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix} \tag{5.3.4}$$

Nelle equazioni si è imposta l'uguaglianza $-K^{-T}\mathbf{v} = \mathbf{p}$, che utilizzata nella (5.3.3) fornisce:

$$H = \begin{bmatrix} K & \mathbf{0} \\ -\mathbf{p^T}K & k \end{bmatrix} \tag{5.3.5}$$

Sebbene il vettore $\mathbf{p}$ venga identificato, per semplicità notazionale, con la proiezione del piano all'infinito, si può constatare come in realtà non rappresenti le coordinate di un piano. A questo punto si può imporre che l'omografia trasformi opportunamente

anche la seconda matrice di proiezione, data nella forma generica $\left[\ A \mid \mathbf{a}\ \right]$:

$$\left[\ A \mid \mathbf{a}\ \right] \begin{bmatrix} K & \mathbf{0} \\ -\mathbf{p^T}K & k \end{bmatrix} \simeq K \left[\ R \mid \mathbf{t}\ \right] \Leftrightarrow$$

$$\Leftrightarrow \left[\ AK - \mathbf{a}\mathbf{p^T}K \mid k\mathbf{a}\ \right] \simeq \left[\ KR \mid K\mathbf{t}\ \right] \tag{5.3.6}$$

Si noti che, nelle relazioni (5.3.2-5.3.6), i simboli $=$ e $\simeq$ sono stati introdotti in base ad un preciso criterio. L'uguaglianza è stata utilizzata nel calcolo *esatto* di H, mentre non la si sarebbe potuta utilizzare nella (5.3.6), non essendo nota la scala di $\left[\ A \mid \mathbf{a}\ \right]$. Rappresentando con α il fattore di scala, la (5.3.6) conduce alle equazioni:

$$\alpha KRK^{-1} = (A - \mathbf{a}\mathbf{p^T}) \tag{5.3.7}$$

$$\alpha K\mathbf{t} = \mathbf{a}k \tag{5.3.8}$$

Senza entrare nel merito, si consideri che l'equazione (5.3.7) può essere interpretata come l'equivalenza tra *omografie indotte dal piano all'infinito*, nella sua posizione originaria e dopo essere stato proiettato attraverso H. Nel sistema implementato, la (5.3.7) è stata risolta con l'ausilio della matrice *pseudoinversa*, dopo aver attuato la sostituzione $\left[\ [\mathbf{e}']_\times F \mid \mathbf{e}'\ \right] \rightarrow \left[\ A \mid \mathbf{a}\ \right]$. Si tratta infatti di un sistema lineare e non omogeneo nelle incognite α e $\mathbf{p}$. La pseudoinversa garantisce la minimizzazione dell'errore prodotto dalla non esatta conoscenza di K, R, $\mathbf{t}$, A ed $\mathbf{a}$. Al fine di evitare eventuali mal condizionamenti, si è scelto di imporre che KRK^{-1}, F ed $\mathbf{e}'$ avessero norma unitaria. Tale risultato, nel caso di F ed $\mathbf{e}'$, è stato garantito scegliendo opportunamente la seconda matrice di proiezione nella ricostruzione proiettiva. Per quanto riguarda KRK^{-1}, è stato invece introdotto il cambio di variabile $\beta = \alpha\|KRK^{-1}\|$ nella (5.3.7), ottenendo:

$$\beta \frac{KRK^{-1}}{\|KRK^{-1}\|} = (A - \mathbf{a}\mathbf{p^T}) \tag{5.3.9}$$

Risolto il sistema rispetto a β e $\mathbf{p}$, si è calcolata $\alpha = \beta / \|KRK^{-1}\|$. Sebbene α non compaia nell'omografia cercata, il suo valore è necessario alla determinazione di k, ottenuta a partire dall'equazione 5.3.8. In particolare:

$$\alpha K\mathbf{t} = \mathbf{a}k \Rightarrow |k| = |\alpha| \frac{\|K\mathbf{t}\|}{\|\mathbf{a}\|} \tag{5.3.10}$$

Chiaramente l'equazione non consente di conoscere il segno di k. Lo si scelga, in un primo momento, arbitrariamente. Si valuti poi la disuguaglianza $\|k\mathbf{a} - \alpha K\mathbf{t}\| < \|(-k)\mathbf{a} - \alpha K\mathbf{t}\|$. Il segno scelto deve essere mantenuto nel caso in cui la disuguaglianza sia verificata, ed invertito altrimenti. Calcolati $\mathbf{p}$ e k, l'omografia (5.3.5) può essere utilizzata per rettificare la ricostruzione proiettiva discussa nel capitolo 4.

5.4 Simulazioni

Sebbene il software sia stato implementato per lavorare in modalità *real-time*, la necessità di ottenere risultati abbastanza precisi ha condotto alla scelta di fornire in ingresso al sistema un filmato memorizzato, piuttosto che l'uscita video della telecamera. In tal modo, nella scelta del compromesso tra velocità di elaborazione e precisione dei dati, si è potuto fornire maggior peso a questi ultimi, aumentando ad esempio la dimensione delle finestre utilizzate nella fase di tracking. Il sistema di visione è stato testato sulla scena rappresentata nella figura 5.1. In essa sono presenti un libro e due piani approssimativamente ortogonali. Su ogni piano è raffigurata una scacchiera 7×9, le cui case hanno lati di 3 centimetri. Le due scacchiere sono disposte in modo tale che i loro lati più lunghi siano approssimativamente paralleli. L'utilizzo di *pattern* così facilmente riconoscibili, come quelli forniti dalle scacchiere, non è un requisito imposto dal sistema. Si tratta invece di una scelta nata dall'esigenza di valutare in modo immediato la correttezza dei risultati. Durante la prima fase della

simulazione, la telecamera è stata fatta traslare manualmente di 30 centimetri, lungo la baseline delle prime due viste, utilizzando la guida mostrata nella figura 1.1 (lo stesso movimento sarebbe facilmente riproducibile da un robot mobile). Il completamento della traslazione è stato comunicato al programma attraverso la pressione di un opportuno carattere sulla tastiera, ricevuto il quale sono state calcolate le posizioni della telecamera agli estremi della baseline ed una prima ricostruzione dell'ambiente, mostrata nella figura 5.2. L'intera guida è stata poi spostata sul piano in modo tale che la telecamera generasse una traiettoria chiusa. Si noti che il sistema è stato progettato per lavorare su traiettorie generiche, e non necessariamente planari. Durante lo spostamento, il programma ha elaborato per ogni fotogramma le corrispondenti posizioni della telecamera, ed ha contemporaneamente ampliato la ricostruzione dell'ambiente, mostrata nella sue interezza nella figura 5.3. Nella figura 5.4 sono stati aggiunti manualmente degli elementi grafici, che rendono più facilmente interpretabili le nuvole di punti prodotte dal software. Nelle figure 5.5-5.7 sono mostrate sia la ricostruzione dell'ambiente che la traiettoria della telecamera, viste da differenti angolazioni. I risultati grafici sono stati integrati con risultati numerici, ottenuti misurando sul modello le distanze tra i vertici delle case, e rilevando come queste siano effettivamente prossime ai 3 centimetri misurabili sulla scacchiera reale.

Figura 5.1: Fotogramma prelevato dal filmato utilizzato nella simulazione.

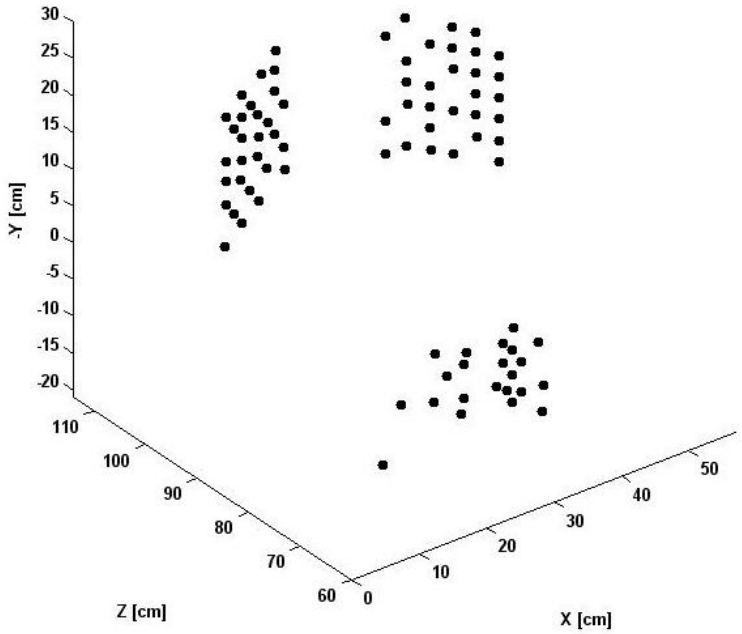

Figura 5.2: Ricostruzione parziale dell'ambiente.

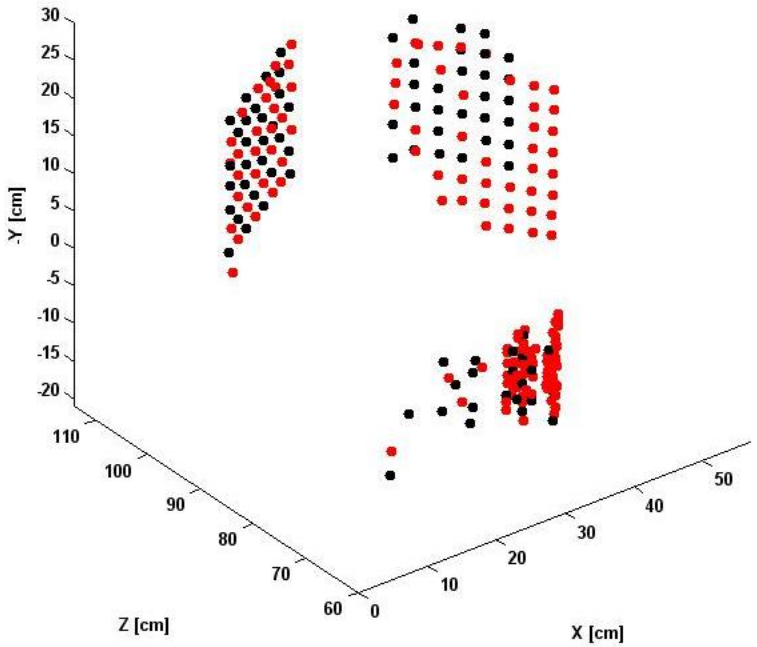

Figura 5.3: Ricostruzione completa dell'ambiente.

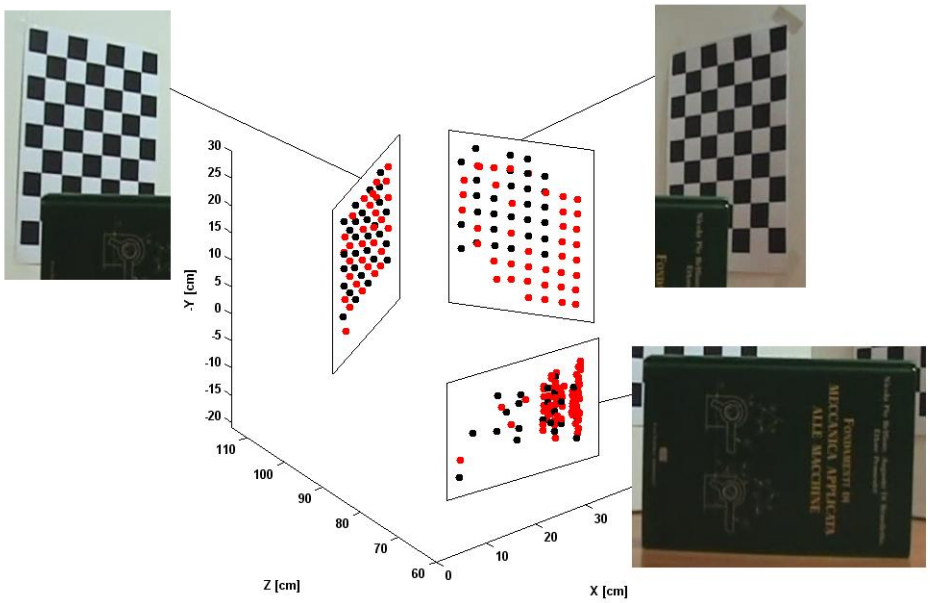

Figura 5.4: Corrispondenze tra nuvole di punti ed oggetti nel fotogramma.

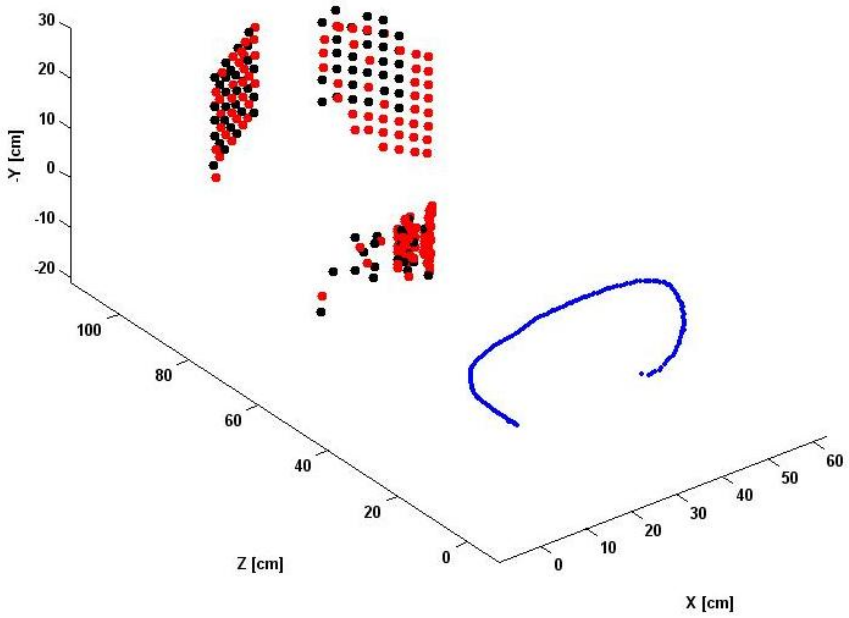

Figura 5.5: Traiettoria della telecamera (blu) e ricostruzione dell'ambiente.

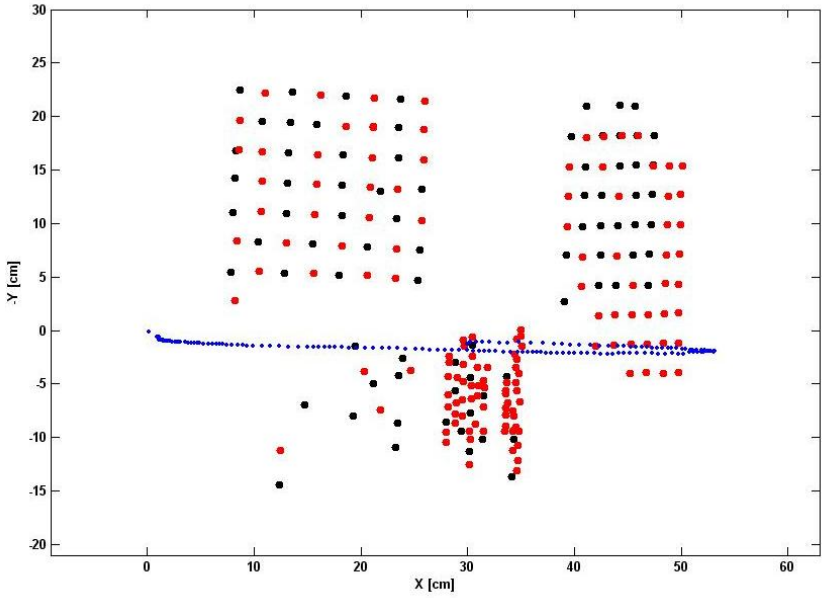

Figura 5.6: Traiettoria della telecamera (blu) e ricostruzione dell'ambiente viste frontalmente.

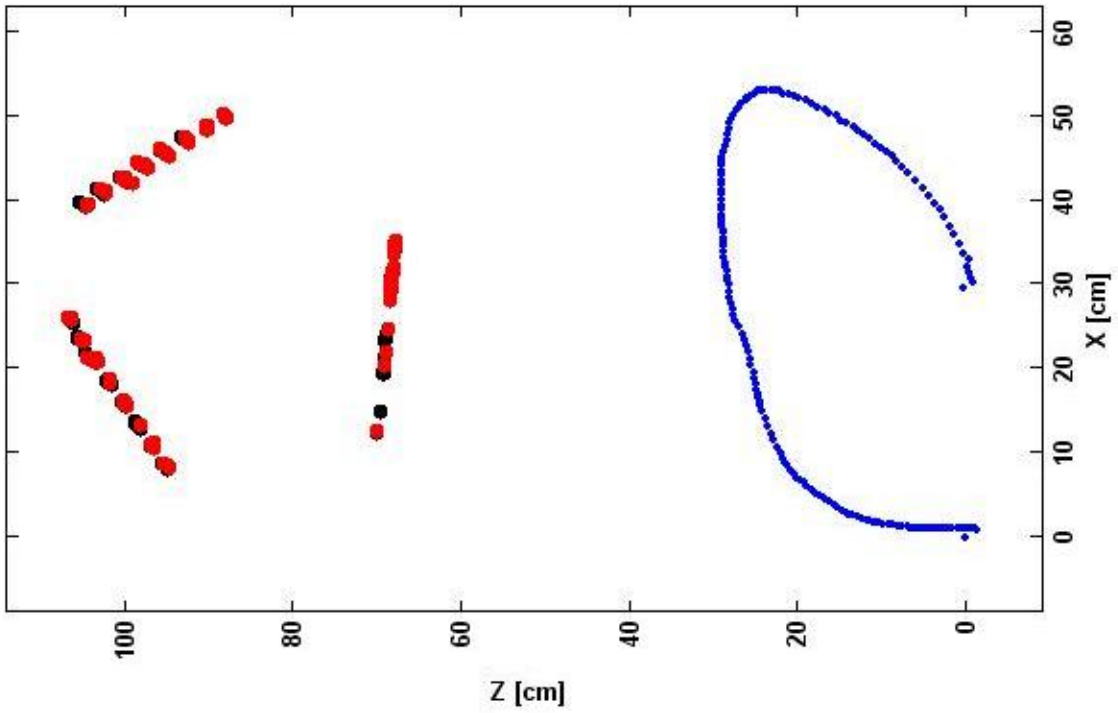

Figura 5.7: Traiettoria della telecamera (blu) e ricostruzione dell'ambiente viste dall'alto.

5.5 Autocalibrazione

La relazione $(A - \mathbf{ap^T})K \simeq KR$, utilizzata per calcolare la proiezione del piano all'infinito, fornisce anche un punto di partenza per l'introduzione delle tecniche di *autocalibrazione* ([24]). Si modifichi leggermente la notazione della relazione, in modo tale che essa esprima un legame tra la prima vista ed una generica vista i-esima (non solo la seconda, come accadeva nel capitolo 4), e che le due viste abbiano parametri intrinseci in generale differenti (per la prima volta nella presente trattazione):

$$(A_i - \mathbf{a_i p^T})K_1 \simeq K_i R_i \tag{5.5.1}$$

Moltiplicando entrambi i membri per le rispettive trasposte, e sfruttando le proprietà delle matrici di rotazione, si ottiene la relazione:

$$(A_i - \mathbf{a_i}\mathbf{p}^T)K_1K_1^T(A_i - \mathbf{a_i}\mathbf{p}^T)^T \simeq K_iK_i^T \tag{5.5.2}$$

La (5.5.2) può essere ricondotta al più a 5 equazioni indipendenti, in quanto omogenea e dotata di simmetria. Si ipotizzi che le incognite siano 13, il massimo valore possibile, rappresentate dai parametri intrinseci di K_1 e K_i, e dal vettore $\mathbf{p}$. Ad ogni vista diversa dalla prima è associata una relazione del tipo (5.5.2). Si potrebbe pensare di poter accumulare abbastanza viste, e quindi equazioni, da poter calcolare tutte le incognite. Purtroppo ogni vista introduce, insieme alle 5 equazioni indipendenti nella migliore delle ipotesi, anche le 5 incognite relative ai parametri intrinseci della vista medesima. Se non fosse per questo "perfetto" bilanciamento, si avrebbe a disposizione un sistema per la stima *in linea* di tutti i parametri necessari alla ricostruzione metrica, sfruttando le sole informazioni provenienti dalla ricostruzione proiettiva. Questo bilanciamento verrebbe meno nel caso si imponessero dei vincoli sulle matrici di calibrazione. Prima di procedere con un esempio pratico, si noti che le equazioni fornite dalla (5.5.2) non sono necessariamente lineari, a causa della sua struttura. Un metodo per diminuirne le non linearità consiste nell'effettuare le sostituzioni $K_1K_1^T \rightarrow \omega^{*\mathbf{1}}$ e $K_iK_i^T \rightarrow \omega^{*\mathbf{i}}$:

$$(A_i - \mathbf{a_i}\mathbf{p}^T)\omega^{*\mathbf{1}}(A_i - \mathbf{a_i}\mathbf{p}^T)^T \simeq \omega^{*\mathbf{i}} \tag{5.5.3}$$

dove $\omega^{*\mathbf{j}}$, con $j = \{1, i\}$, ha un suo significato geometrico ed è definita *immagine duale della conica assoluta*. Allo stesso scopo, la (5.5.3) può essere trasformata nel modo seguente:

$$(A_i - \mathbf{a_i}\mathbf{p}^T)\omega^{*\mathbf{1}}(A_i - \mathbf{a_i}\mathbf{p}^T)^T \simeq \left[\begin{array}{c|c} A_i & \mathbf{a_i} \end{array}\right] \begin{bmatrix} \omega^{*\mathbf{1}} & -\omega^{*\mathbf{1}}\mathbf{p} \\ -\mathbf{p}^T\omega^{*\mathbf{1}} & \mathbf{p}^T\omega^{*\mathbf{1}}\mathbf{p} \end{bmatrix} \left[\begin{array}{c|c} A_i & \mathbf{a_i} \end{array}\right]^T \overset{def}{\simeq}$$

$$\stackrel{def}{\simeq} P_i Q^* P_i^T \simeq \omega^{*\mathbf{i}} \tag{5.5.4}$$

dove Q^* ha un suo significato geometrico ed è definita *duale della quadrica assoluta.* È evidente come la (5.5.4) fornisca una relazione molto semplice tra le nuove incognite, costituite dagli elementi di Q^* e $\omega^{*\mathbf{i}}$. In generale non si tratta ancora di equazioni lineari, per la presenza del fattore di scala. Si noti che la trasformazione di incognite $\{K_1, K_i, \mathbf{p}\} \rightarrow \{Q^*, \omega^{*\mathbf{i}}\}$ non avrebbe avuto senso se non fosse stata invertibile. Per ottenere K_i da $\omega^{*\mathbf{i}}$ è sufficiente applicare la decomposizione di Cholesky, e con un semplice procedimento sono calcolabili anche K_1 e $\mathbf{p}$ a partire da Q^*. Chiaramente la convenienza nell'attuazione di tali trasformazioni dipende anche dal tipo di vincoli imposti sulle incognite di partenza, e da come tali vincoli si traducano sulle nuove incognite. Un caso comune è quello in cui sia nota la collocazione del punto principale. Traslando le coordinate dell'immagine si può ottenere una nuova vista il cui punto principale abbia coordinate nulle. Ciò porta all'annullamento di ${\omega^{*\mathbf{i}}}_{13}$ e ${\omega^{*\mathbf{i}}}_{23}$, ed alle due equazioni lineari $(P_i Q^* P_i^T)_{13} = 0$ e $(P_i Q^* P_i^T)_{23} = 0$. Con un numero sufficiente di viste si può stimare Q^*, e tutte le altre incognite a seguire. Il metodo è stato sperimentato sul sistema di visione, in un primo momento utilizzando dati sintetici, e poi dati reali. Dal punto di vista tecnico la sua implementazione è risultata particolarmente semplice, mentre numerose difficoltà si sono riscontrate nel funzionamento dell'algoritmo. In particolare sono stati ricavati risultati validi in presenza di dati sintetici, a conferma della sua corretta implementazione, e risultati evidentemente errati in presenza di dati reali. La spiegazione, che trova qualche conferma in [7], è probabilmente da cercarsi nel limitato spazio di lavoro della telecamera, e nella conseguente assenza di equazioni sufficientemente caratterizzanti.

Capitolo 6

Conclusioni e sviluppi futuri

La tesi fornisce una descrizione dettagliata degli algoritmi necessari al funzionamento di un sistema di autolocalizzazione e ricostruzione dell'ambiente, dal rilevamento dei punti di interesse al calcolo dell'omografia per la rettificazione della ricostruzione proiettiva. Si è rivolto maggiore interesse agli aspetti meno discussi in letteratura, ma comunque essenziali per il funzionamento del sistema. Particolarmente interessante è stato constatare come scene povere di particolari e movimenti ridotti della telecamera siano tra le principali cause di mal funzionamento degli algoritmi, dalla triangolazione all'autocalibrazione. L'idea che l'insorgenza di configurazioni degeneri potesse ritenersi un evento molto raro, è stata frequentemente contraddetta in fase simulativa. Il problema delle configurazioni degeneri nella stima della matrice fondamentale, è stato risolto molto efficacemente imponendo alla telecamera un moto inizialmente vincolato, ed utilizzando i relativi algoritmi. I risultati ottenuti sono particolarmente soddisfacenti, a conferma delle grandi potenzialità espresse in questo campo dalla geometria proiettiva.

Un problema endemico dei sistemi di autolocalizzazione e ricostruzione ambientale è rappresentato dalla propagazione degli errori, dovuta all'esecuzione in cascata degli algoritmi. Seppur in forma ridotta, il problema permane nonostante si utilizzino

telecamere più precise o algoritmi più performanti. La collocazione di landmark o l'impiego di sequenze video non troppo lunghe rappresentano due possibili strategie risolutive. Un'ulteriore strategia si basa sulla ricerca di corrispondenze tra immagini distanti all'interno della sequenza, in modo da utilizzare dati più affidabili per la ricostruzione presente. Nei sistemi real-time potrebbe trattarsi di una tecnica computazionalmente troppo onerosa.

Nel caso in cui si ritenessero superflue la conoscenza della traiettoria e di un modello ambientale d'insieme, si potrebbe utilizzare un sistema stereoscopico per ottenere di volta in volta una ricostruzione istantanea dell'ambiente, meglio nota come mappa di profondità. L'impiego di due telecamere consentirebbe anche di rilevare correttamente la profondità di corpi in movimento. Questa tipologia di sistemi è frequentemente utilizzata nella risoluzione di problemi di obstacle avoidance. Il sistema di visione realizzato è stato progettato per lavorare su scene statiche, sebbene la presenza di piccoli corpi in movimento possa essere tollerata ed i relativi punti interpretati come outlier. Con l'aggiunta di pochi algoritmi e di una seconda telecamera, il suo funzionamento potrebbe essere esteso alle scene dinamiche. Il problema principale da affrontare in questo caso consisterebbe nell'individuazione dei corpi fissi rispetto ai quali effettuare la ricostruzione della traiettoria. Si noti inoltre che l'utilizzo di due telecamere per rilevare corpi in movimento ha senso nel solo caso in cui le telecamere siano sincronizzate, cioè catturino differenti viste della scena in istanti identici. Esistono differenti strategie per la sincronizzazione delle telecamere, dall'invio di un segnale di clock esterno all'utilizzo della frequenza di funzionamento della rete di alimentazione.

Il sistema implementato produce come risultato un nuvola di punti corrispondenti ai punti di interesse rilevati nelle immagini. Trattandosi di punti non collocati con continuità nello spazio, si parla di ricostruzione sparsa. A partire dalla ricostruzione sparsa,

esistono algoritmi in grado di condurre ad una ricostruzione densa dell'ambiente. Si tratta innanzitutto di attuare un procedimento di rettificazione epipolare delle immagini, grazie al quale linee epipolari corrispondenti siano orizzontali ed ugualmente collocate sulle rispettive immagini, seguito da una fase di matching denso dei punti, solitamente basato su algoritmi derivanti dal calcolo delle variazioni. La ricostruzione densa rappresenta un punto di contatto tra gli algoritmi della computer vision e quelli della computer graphics.

Gli algoritmi basati sulla geometria proiettiva sono molto promettenti, e probabilmente renderanno obsolete molte tecniche attualmente basate sull'elaborazione a basso livello delle immagini. L'impiego della geometria proiettiva rende necessario un bagaglio teorico approfondito, giustificato dall'elevato valore aggiunto fornito ai sistemi di visione. Sebbene OpenCV permetta di ridurre notevolmente i tempi di sviluppo mettendo a disposizione un'ampia libreria di funzioni, la loro integrazione non può prescindere da certe competenze.

Bibliografia

[1] G. Bradski - A. Kaehler, *"Learning OpenCV: Computer Vision with the OpenCV Library"*, O'Reilly Media, 2008.

[2] R. C. Gonzalez - R. E. Woods, *"Digital Image Processing"*, Prentice Hall, 2002.

[3] C. Harris - M. Stephens, *"A combined corner and edge detector"*, Proceedings of the 4th Alvey Vision Conference, 1988.

[4] J. Shi. - C. Tomasi, *"Good features to track"*, 9th IEEE Conference on Computer Vision and Pattern Recognition, 1994.

[5] S. Baker - I. Matthews, *"Lucas-Kanade 20 Years On: A Unifying Framework"*, International Journal of Computer Vision, 2004.

[6] J. Y. Bouguet, *"Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm"*, `http://robots.stanford.edu/cs223b04/algo_tracking.pdf`, 2004.

[7] R. Hartley - A. Zisserman, *"Multiple View Geometry in Computer Vision"*, Cambridge University Press, 2003.

[8] O. Faugeras - Q. T. Luong, *"The Geometry of Multiple Images"*, MIT Press, 2001.

[9] D. A. Forsyth - J. Ponce, *"Computer Vision: A Modern Approach "*, Prentice Hall, 2003.

[10] L. Sigal, *"Computer Graphics, Lecture Notes"*, `www.cs.brown.edu/~ls/teaching08/LS07_Cameras_Part1.pdf`, 2008.

[11] D. C. Brown, *"Lens Distortion for Close-Range Photogrammetry"*, Photometric Engineering, 1971.

[12] Z. Zhang, *"A Flexible New Technique for Camera Calibration"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

[13] G. Medioni - S. B. Kang, *"Emerging Topics in Computer Vision"*, Prentice Hall, 2005.

[14] F. Bookstein, *"Fitting conic sections to scattered data"*, Computer Graphics and Image Processing, 1979.

[15] P. Mittrapiyanuruk, *"A Memo on How to Use the Levenberg-Marquardt Algorithm for Refining Camera Calibration Parameters"*, `http://cobweb.ecn.purdue.edu/~kak/courses-i-teach/ECE661/HW5_LM_handout.pdf`, 2006.

[16] K. Madsen - H. B. Nielsen - O. Tingleff, *"Methods for non-linear least squares problems"*, `http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3215/pdf/imm3215.pdf`, 2004.

[17] M. I. A. Lourakis - A. A. Argyros, *"The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm"*, `http://www.ics.forth.gr/~lourakis/sba/tr340.ps`, 2004.

[18] R. I. Hartley, *"In Defence of the 8-point Algorithm"*, IEEE Transaction on Pattern Recognition and Machine Intelligence, 1997.

[19] M. Mühlich - R. Mester, *"The Role of Total Least Squares in Motion Analysis"*, Lecture Notes in Computer Science, 1998.

[20] J. Y. Bouguet, *"Camera Calibration Toolbox for Matlab"*, `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`.

[21] P. H. S. Torr - A. Zisserman - S. J. Maybank, *"Robust detection of degenerate configurations for the fundamental matrix"*, Fifth International Conference on Computer Vision (ICCV'95), 1995.

[22] E. Trucco - A. Verri, *"Introductory Techniques for 3-D Computer Vision"*, Prentice Hall, 1998.

[23] M. Pollefeys - L. Van Gool - M. Vergauwen - F. Verbiest - K. Cornelis - J. Tops - R. Koch, *"Visual modeling with a hand-held camera"*, International Journal of Computer Vision, 2004.

[24] O. D. Faugeras - Q. T. Luong - S. J. Maybank, *"Camera self-calibration: Theory and experiments"*, Second European Conference on Computer Vision Santa Margherita Ligure, 1992.