Politecnico di Milano
Facoltà di Ingegneria dei Sistemi
Corso di Studi in Ingegneria Matematica

Tesi di Laurea Magistrale

STIMA DEL MOTO DA SEQUENZE DI IMMAGINI OMNIDIREZIONALI

Relatore:
Prof. Marco MARCON

Correlatore:
Dott. Emanuele PLEBANI

Tesi di Laurea Magistrale di:
Francesco GROSSI
Matr. 750680

Anno Accademico 2010-2011

Sommario

La tesi è stata sviluppata nell'ambito del progetto europeo Astute, che mira a portare innovazione tecnologica nel campo del mercato automobilistico. Uno degli ambiti di innovazione riguarda la stima degli spostamenti dell'autovettura mediante tecniche di analisi di immagine applicate ai dati forniti da una videocamera a bordo.

La ricostruzione tridimensionale a partire da sequenze di immagini, nota anche come *Structure from Motion*, è un'area di ricerca di grande interesse, grazie alla grande diffusione e ai miglioramenti delle tecnologie di acquisizione di immagini digitali. Per un'applicazione in contesti urbani risulta interessante utilizzare videocamere omnidirezionali, che possono essere realizzate mediante ottiche particolari o incollando fra loro immagini scattate con videocamere tradizionali. La possibilità di estendere il campo visivo a 360° consente infatti di migliorare la qualità della ricostruzione.

La tesi propone due metodi di stima del moto di una videocamera omnidirezionale. Il primo metodo utilizza coppie di viste per stimare gli spostamenti a partire dal calcolo della matrice essenziale; il secondo metodo utilizza terzetti di viste per calcolare il tensore trifocale, da cui vengono estratti gli spostamenti. I due metodi sono messi a confronto sia su dati sintetici che sperimentali, utilizzando le viste panoramiche di Google Street View.

Abstract

The present thesis has been developed in the context of the Astute European Project, which aims at bringing technological innovation in the automotive market. One of the proposed innovations is the estimation of the car movements using image analysis techniques applied to the data collected by an onboard camera.

Tridimensional reconstruction from sequences of images, a process called *Structure from Motion*, is an area of research which has been highly developed in recent years, following the widespread availability and technological improvements of digital image acquisition systems. Applications of *SfM* in urban contexts could benefit from the use of of omnidirectional images, generated with special optics applied to an image sensor or by stitching together multiple pictures. The large field of view can in fact improve the quality of the reconstruction.

The thesis presents two methods for the estimation of the movements of an omnidirection camera. The first algorithm uses pairs of views to estimate the egomotion by calculating the essential matrix; the second uses triplets of views to compute the trifocal tensor, from which the movements are estimated. The two methods are compared in different test cases, using both synthetic and experimental data obtained from Google Street View.

Indice

Elenco delle figure

Introduzione

L'ambito di azione: la *Structure from Motion*

È in larga parte grazie alla vista che l'uomo vive e conosce il mondo circostante. Con il progresso tecnologico anche i computer hanno accesso alle immagini del mondo visibile: fotografie e video digitali sono dati disponibili in enormi quantità e a bassi costi, pronti per essere analizzati e processati.

Tra le varie informazioni che l'uomo ottiene tramite la vista c'è il senso dello spazio, che permette di stimare dimensioni, distanze, movimenti. Senza farci caso, semplicemente osservando gli oggetti intorno cambiare grandezza e prospettiva, ognuno è in grado di capire con buona approssimazione di quanto si è mosso. Tutto questo avviene grazie alle informazioni recuperate dal sistema visivo, in cui gli occhi forniscono flussi di immagini da due diversi punti di vista, requisito fondamentale per avere il senso della tridimensionalità, e all'elaborazione dell'esperienza, grazie alla quale siamo in grado di conoscere le dimensioni tipiche degli oggetti comuni. Grazie alle possibilità concesse negli ultimi anni dall'onnipresenza delle videocamere digitali e dall'aumento di potenza di calcolo e memoria, anche i computer sono oggi in grado di mimare alcune funzioni del sistema visivo umano, in certi ambiti anche con migliore precisione.

Una delle possibili applicazioni dell'analisi d'immagine è quella della ricostruzione 3D del mondo a partire da fotografie prese da diversi punti di vista, sinteticamente detta *Structure from Motion* (SfM). Le viste possono arrivare da più apparecchi fotografici o da un singolo flusso video, ad esempio una videocamera che si muove e riprende la scena da diversi punti; tramite queste informazioni è possibile ricostruire il mondo tridimensionale e stimare la posizione della videocamera stessa. Esiste ormai una vasta letteratura sull'ambito, anche formalizzata in libri di testo (a titolo di esempio [8]); i vari metodi che vengono proposti per realizzare la *Structure from Motion* sono tipicamente modulari, e consistono in diverse fasi:

1. Identificazione di punti notevoli dell'immagine, che possano essere riconoscibili anche cambiando punto di vista.

2. Riconoscimento delle corrispondenze, per capire quali punti in un'immagine corrispondano a quali altri nelle diverse immagini, in modo da avere informazioni sullo spostamento dell'osservatore.

3. Calcolo della geometria della scena, per capire da quali punti di vista sono state riprese le immagini.

4. Ricostruzione della scena tridimensionale.

La modularità consente un miglioramento focalizzato in uno specifico settore, e la letteratura in ciascun ambito è molto ampia. Per quanto riguarda il punto 1, sono ormai affermati metodi di identificazione di punti notevoli *locali*, a cui vengono assegnati descrittori, ovvero "codici di identificazione", che siano il più possibile invarianti a cambi di luminosità, scala, rotazioni e distorsioni prospettiche. Tra i più utilizzati ricordiamo la Scale Invariant Feature Transform (SIFT, [12]) e la Speeded Up Robust Feature (SURF, [1]).

Il riconoscimento delle corrispondenze avviene tipicamente in due passi: una prima selezione si basa sull'accoppiamento dei punti notevoli in base alla distanza euclidea dei loro descrittori, visti come vettori di uno spazio n-dimensionale (ad esempio SIFT usa descrittori con 128 elementi). Se ad esempio un descrittore è invariante per cambi di posizione e rotazione, lo stesso punto notevole in un'altra immagine avrà un descrittore identico, e quindi distanza euclidea nulla. Naturalmente, essendo il descrittore una semplificazione locale dell'immagine, possono verificarsi casi in cui punti diversi hanno descrittori simili, e che vengono quindi stimati come corrispondenze. Per ovviare a questo problema si ricorre tipicamente a metodi ispirati al RANdom SAmple Consensus (RANSAC, [4]), che cercano di distinguere le corripondenze corrette da quelle non corrette provando molteplici modelli e selezionando quello che meglio descrive l'insieme di punti.

Il calcolo della geometria della scena si basa sull'utilizzo dei vincoli generati dalle corrispondenze di punti, la cosiddetta *geometria epipolare*. A seconda del numero di viste utilizzate esistono diversi metodi, di cui in [8, 21] è possibile trovare una trattazione completa.

Infine, la ricostruzione della scena tridimensionale avviene tramite metodi di triangolazione, che stimano la posizione dei punti dello spazio a partire dalle loro proiezioni sulle immagini e da videocamere i cui parametri siano noti.

I metodi descritti sono ormai affermati per quanto riguarda le videocamere tradizionali, che hanno angoli di visuale minori di 180°. Esistono però lenti con un campo visivo maggiore, le cosiddette *fisheye*, ed esistono algoritmi in grado di incollare diverse immagini per generare una vista omnidirezionale, a 360°. Il tipico esempio è Google Street View ([9]), dove all'utente viene proposta la vista panoramica, omnidirezionale, delle strade di quasi tutto il

mondo. Sebbene i metodi di generazione di queste immagini possano essere molto diversi, entrambe possono essere descritte con il modello di videocamera omnidirezionale.

È possibile usare metodi di Structure from Motion anche con immagini omnidirezionali, che offrono un notevole vantaggio in fatto di quantità di oggetti inquadrati nella scena e aiutare quindi il processo di ricostruzione. Uno dei problemi della ricostruzione da più viste prese lungo un percorso allineato con l'asse ottico è che la prospettiva degli oggetti inquadrati non cambia significativamente in una videocamera tradizionale, cosa che aumenta l'incertezza della ricostruzione; con viste omnidirezionali vengono invece utilizzati anche gli oggetti che compaiono "lateralmente" alla videocamera, quelli per cui la variazione di prospettiva è maggiore.

La ricostruzione con videocamere omnidirezionali è stata portata avanti con diversi metodi (vedi ad esempio [6, 2, 17, 5, 18]); parte della letteratura disponibile per le videocamere tradizionali è applicabile o adattabile al caso omnidirezionale, come proposto in [20], anche se manca una sistematizzazione rigorosa. Tra i risultati più interessanti ricordiamo la ricostruzione metrica ottenuta da immagini omnidirezionali non calibrate di Micusik ([13, 14]) e la Structure from Motion ottenuta da Torii e altri ([19]) a partire da coppie di immagini di Google Street View.

Finalità della tesi

Questa tesi è stata realizzata nell'ambito del progetto europeo ASTUTE [10], *Pro-active decision support for data-intensive environments*, che mira ad applicare tecnologie avanzate di interazione uomo-macchina come supporto alle decisioni in contesti di sovraccarico di informazioni. Il Politecnico di Milano partecipa al progetto, collaborando con altre aziende italiane, nell'offrire un contributo allo sviluppo di tecnologie di analisi d'immagine da impiegare in ambito automobilistico.

Uno degli strumenti più utili per il supporto alle decisioni durante la guida è il navigatore GPS con navigazione *turn-by-turn*; tuttavia in città vi sono situazioni, dette di *canyon urbano* a causa della altezza e vicinanza degli edifici a bordo strada, in cui la cattiva ricezione del segnale rende impreciso il posizionamento GPS. Il navigatore inoltre non ha modo di calcolare l'orientamento spaziale dell'automobile, che viene solitamente estrapolato dai movimenti passati. Un aiuto in entrambi i casi può arrivare dall'analisi di immagine: effettuando Structure from Motion da immagini prese da una videocamera presente sull'abitacolo, si può integrare e raffinare il posizionamento stimato da GPS.

Utilizzare videocamere tradizionali presenta però diversi problemi: anche utilizzando un buon grandangolo, buona parte dei punti notevoli identificati nei fotogrammi del video è relativa ad oggetti disposti lungo l'asse ottico, con i problemi di precisione di stima prima evidenziati; d'altra parte le zone laterali delle immagini cambierebbero molto rapidamente a causa della velocità dell'automobile, cosa che le renderebbe poco affidabili. Una videocamera omnidirezionale riuscirebbe ad ovviare a questi problemi.

Inoltre, una delle aziende coinvolte nel progetto ha sviluppato competenze nella creazione di immagini omnidirezionali a 360° incollando immagini riprese con videocamere tradizionali. Nell'ottica di sfruttare questa tecnologia abbiamo quindi studiato l'uso di queste immagini nell'ambito della Structure from Motion.

La tesi ha quindi come obiettivo lo sviluppo di un metodo di stima del moto proprio della videocamera utilizzando immagini omnidirezionali. Una delle scelte che ci si trova ad affrontare è quella del numero di immagini da utilizzare contemporaneamente: la scena 3D può essere ricostruita utilizzando almeno due viste mediante la matrice fondamentale, ma sono stati proposti metodi per un numero maggiore di viste. Questa tesi si propone in particolare di studiare la ricostruzione a partire da tre viste utilizzando il tensore trifocale, e di confrontarne le prestazioni con il caso a due viste nell'ambito di scenari urbani, in vista di un'implementazione prototipale per il progetto ASTUTE.

Capitolo 1

Geometria proiettiva ed epipolare

In questo capitolo vengono trattate le basi di geometria e di modellizzazione delle videocamere necessarie per la ricostruzione del movimento proprio delle videocamere. La trattazione inizia con una panoramica sulle videocamere tradizionali, sulle quali è presente una ricca bibliografia, per passare poi alle videocamere omnidirezionali.

1.1 Geometria proiettiva per videocamere tradizionali

In questa sezione introduciamo gli elementi di geometria proiettiva che verranno usati nella tesi; iniziamo con una trattazione delle videocamere tradizionali in quanto la maggior parte dei metodi a cui si farà riferimento sono stati sviluppati in questo contesto.

Rappresentazione omogenea di linee e punti Un retta nel piano (x, y) è identificata da un'equazione del tipo $ax + by + c = 0$, $a, b, c \in \mathbb{R}$; il vettore $(a, b, c)^\top$ è un rappresentante della classe di equivalenza delle rette del tipo $kax + kby + kc = 0$, visto che la moltiplicazione per una costante non nulla non cambia l'equazione. La rappresentazione vettoriale delle rette, dotate di tale relazione di equivalenza, viene detta *omogenea*. L'insieme delle classi di equivalenza dei vettori di $\mathbb{R} - (0, 0, 0)^\top$ forma lo spazio proiettivo $\mathbb{P}^2$.

Analogamente anche i punti hanno una rappresentazione omogenea. Un punto $\mathbf{x} = (x, y)^\top$ giace su una retta $\mathbf{l} = (a, b, c)^\top$ se e solo se $ax + by + c = 0$; esprimendo la relazione in termini di prodotto scalare, $(x, y, 1)(a, b, c)^\top = (x, y, 1)\mathbf{l} = 0$. I punti di $\mathbb{R}^2$ possono quindi essere rappresentati in modo

omogeneo aggiungendo una terza coordinata pari a 1. Moltiplicando la rappresentazione omogenea di un punto per una costante non nulla, otteniamo ancora un rappresentante della stessa classe di equivalenza, come nel caso delle rette: un vettore omgeneo generico $(x_1, x_2, x_3)^\top$ rappresenta il punto $(\frac{x_1}{x_3}, \frac{x_2}{x_3})^\top$ di $\mathbb{R}^2$. I punti nella loro rappresentazione omogenea sono quindi elementi di $\mathbb{P}^2$.

Sia per le rette che per i punti la rappresentazione omogenea è in realtà ridondante, in quanto è sufficiente specificare due gradi di libertà per fissare un punto o una retta nel piano, quindi ad esempio i due rapporti $\{a : b : c\}$.

Punti e rette con la terza coordinata nulla possono essere aggiunti a $\mathbb{R}^2$ per completare lo spazio proiettivo $\mathbb{P}^2$. Sono punti ideali, detti *punti all'infinito* della forma $(x_1, x_2, 0)^\top$, ognuno dei quali è definito dal rapporto $x_1 : x_2$, che appartengono a una sola retta, la *retta all'infinito*, definita dal vettore $\mathbf{l}_\infty = (0, 0, 1)^\top$.

Punti e retta all'infinito sono utili per semplificare le proprietà di intersezioni di rette e punti. In notazione omogenea, infatti,

- L'intersezione di due rette $\mathbf{l}$ e $\mathbf{l}'$ è il punto $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$.
- La retta passante per due punti $\mathbf{x}$ e $\mathbf{x}'$ ha l'equazione $\mathbf{l} = \mathbf{x} \times \mathbf{x}'$.

Se abbiamo due rette parallele, della forma $\mathbf{l} = (a, b, c)^\top$ e $\mathbf{l}' = (a, b, c')^\top$, l'intersezione è $\mathbf{x} = \mathbf{l} \times \mathbf{l}' = (c' - c)\,(b, -a, 0)^\top = (b, -a, 0)^\top$ a parte il fattore di scala. Nello spazio proiettivo $\mathbb{P}^2$ due rette parallele si intersecano quindi in un punto all'infinito.

Trasformazioni proiettive Nello spazio proiettivo è possibile definire trasformazioni, esprimibili con l'algebra lineare, che possono essere usate per modellizzare il processo di formazione dell'immagine a partire da una scena 3D.

Definizione 1. Una trasformazione proiettiva o omografia è una mappa invertibile $h : \mathbb{P}^2 \mapsto \mathbb{P}^2$ tale che tre punti $\mathbf{x}_1$, $\mathbf{x}_2$ e $\mathbf{x}_3$ appartengano alla stessa retta se e solo se $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$ e $h(\mathbf{x}_3)$ appartengono alla stessa retta.

Vale il seguente

Teorema 2. *Una mappa $h : \mathbb{P}^2 \mapsto \mathbb{P}^2$ è un'omografia se e solo se esiste una matrice 3×3 non singolare $\mathtt{H}$ tale che ogni punto $\mathbf{x}$ di $\mathbb{P}^2$ viene trasformato nel punto $h(\mathbf{x}) = \mathtt{H}\mathbf{x}$.*

Come risultato di questo teorema è possibile definire un'omografia nel seguente modo:

Definizione 3. Un'omografia nel piano è una trasformazione lineare su vettori omogenei rappresentata da una matrice non singolare di dimensioni 3×3 :

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

o più brevemente $\mathbf{x}' = \mathtt{H}\mathbf{x}$.

È da notare che moltiplicando la matrice H per uno scalare non nullo l'omografia rimane invariata, ovvero il risultato è un punto appartenente alla stessa classe di equivalenza del risultato precedente. Per questo la matrice H è anch'essa detta *omogenea*, dato che come per punti e rette, solo il rapporto tra gli elementi è importante. Dei nove elementi di H solo otto sono indipendenti, per cui H ha otto gradi di libertà.

Lo spazio proiettivo $\mathbb{P}^3$ Quanto visto nel piano può essere facilmente esteso nello spazio tridimensionale. Un punto omogenero sarà del tipo $\mathbf{X} = (X_1, X_2X_3, X_4)^\top$, che rappresenta il punto di $\mathbb{R}^3$ $\mathbf{X} = (\ \frac{X_1}{X_4},\ \ \frac{X_2}{X_4},\ \ \frac{X_3}{X_4}\ )^\top$, e che in caso $X_4 = 0$ rappresenta un punto all'infinito.

Un'omografia in $\mathbb{P}^3$ è rappresentata da una matrice non singolare 4×4: $\mathbf{X}' = \mathtt{H}\mathbf{X}$. La matrice H è omogenea e ha 15 gradi di libertà (16 elementi meno la scala).

Se prima un vettore di $\mathbb{R}^3$ esprimeva una retta, ora un vettore 1×4 rappresenta un piano:

$$\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0,$$

sempre in notazione omogenea, che ha tre gradi di libertà. Come nel caso bidimensionale, un punto giace su un piano se vale $\pi^\top \mathbf{X} = 0$.

Per consistenza di notazione, nel testo i punti di $\mathbb{P}^2$ e $\mathbb{R}^2$ saranno indicati con le lettere minuscole, mentre i punti di $\mathbb{P}^3$ e $\mathbb{R}^3$ saranno in maiuscolo.

La matrice di proiezione della videocamera In una videocamera tradizionale gli oggetti presenti nella scena vengono proiettati, attraverso ottiche più o meno complesse, sul sensore, che tipicamente è planare. In termini geometrici i punti di $\mathbb{R}^3$ vengono proiettati su un piano, che viene chiamato *piano immagine*. Tralasciando per il momento gli effetti non lineari dovuti all'ottica, questa operazione è una semplice proiezione da $\mathbb{R}^3$ a $\mathbb{R}^2$, che può essere comodamente descritta utilizzando gli spazi proiettivi $\mathbb{P}^3$ e $\mathbb{P}^2$.

Se infatti i punti sono scritti in forma omogenea, la proiezione assume la forma di prodotto matrice per vettore, come vediamo ad esempio nel semplice

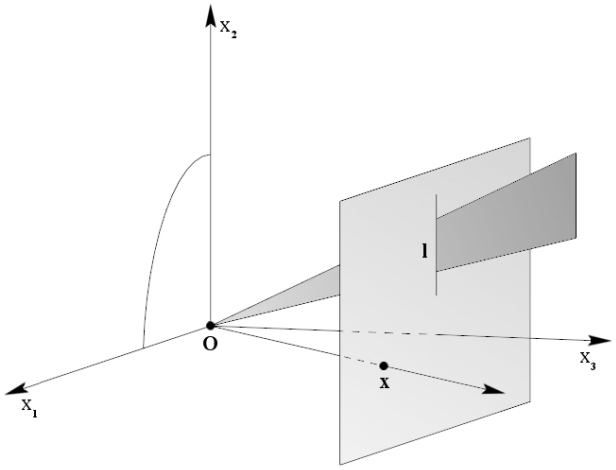

Figura 1.1: Il modello del piano proiettivo: punti e rette di $\mathbb{P}^2$ sono rappresentati da raggi e piani uscenti dall'origine. I punti a coordinata X_3 nulla sono i punti all'infinito.

modello a *pinhole camera*, che sta alla base del funzionamento delle prime macchine fotografiche:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1.1}$$

f, come mostrato in Figura 1.2, è detta lunghezza focale, ed è la coordinata Z del piano immagine. La matrice che effettua la proiezione, che chiameremo $\mathtt{P}$, è detta matrice di proiezione della videocamera (*camera projection matrix*) e ne contiene tutte le informazioni. I punti $\mathbf{x}_i$ sul piano immagine sono generati dai raggi passanti per il centro della videocamera, o centro ottico, che è l'origine, e i punti $\mathbf{X}_i$. Nella realtà il piano immagine è al di là del centro ottico, e l'immagine viene ribaltata.

Generalizzando il semplice modello *pinhole*, una videocamera generica può avere una matrice $\mathtt{P}$ di questo tipo:

$$\mathtt{P} = \mathtt{K}\,[\mathtt{R} \mid \mathbf{t}]$$

dove $\mathtt{R}$ è la matrice di rotazione della videocamera rispetto alle coordinate globali, e $\mathbf{t} = -\mathtt{R}\tilde{\mathbf{C}}$ ne rappresenta la traslazione (con $\tilde{\mathbf{C}}$ centro della videocamera nelle coordinate globali). Queste informazioni sono dette *estrinseci*

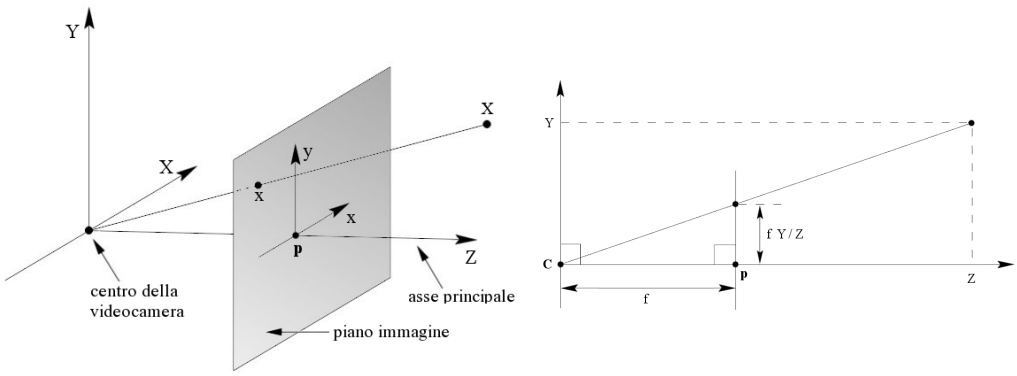

Figura 1.2: Il modello di videocamera di tipo *pinhole*.

della videocamera perché contengono le informazioni di posizionamento della videocamera nello spazio.

`K` è detta matrice di calibrazione, perché contiene i parametri interni, o *intrinseci della videocamera*. Una generica videocamera CCD può essere rappresentata da

$$\mathtt{K} = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{1.2}$$

dove $\alpha_x = fm_x$ e $\alpha_y = fm_y$ rappresentano la lunghezza focale della camera in termini di dimensione dei pixel (che potrebbero non avere uguale fattore di forma in x e y), e m_x e m_y rappresentano la densità lineare dei pixel nelle due direzioni. Se chiamiamo *asse principale* la retta perpendicolare al piano immagine e passante per il centro della camera e *punto principale* l'intersezione di questo asse con il piano, il punto $\widetilde{\mathbf{x}}_0 = (x_0, y_0) = (m_x p_x, m_y p_y)$ rappresenta la posizione del *punto principale* in termini di pixel, visto che non è detto che il piano immagine sia centrato rispetto all'asse principale.

1.2 Geometria epipolare per videocamere tradizionali

Matrice fondamentale `F` e matrice essenziale `E`

Se riprendiamo una scena tridimensionale da diversi punti di vista, le immagini ottenute non saranno completamente indipendenti l'una dall'altra, ma mostreranno gli oggetti presenti secondo ben precise regole prospettiche, a seconda

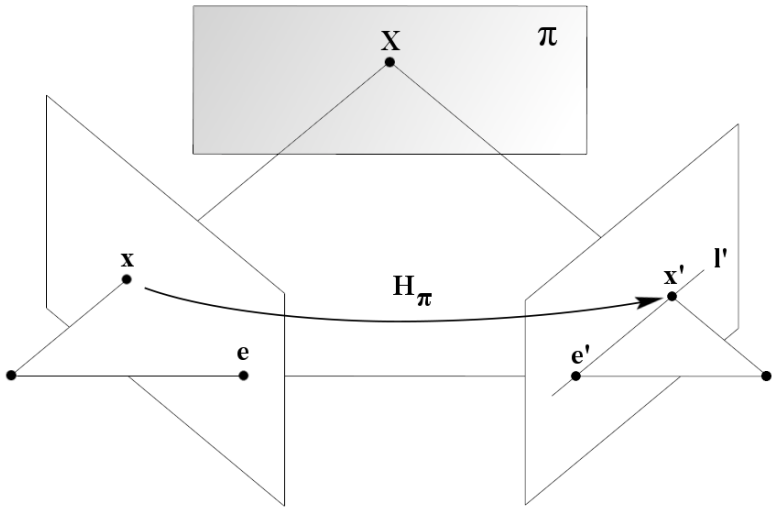

Figura 1.3: Rappresentazione della geometria epipolare tra due viste.

della calibrazione interna della videocamera e dalla sua posizione nello spazio. La geometria epipolare descrive questi vincoli di geometria proiettiva.

Prima di proseguire, introduciamo le seguenti definizioni, iniziando da una geometria epipolare con due viste: la *baseline* è il segmento che congiunge i due centri delle videocamere; l'epipolo è il punto di intersezione tra la *baseline* e il piano immagine. Nella prima vista quindi l'epipolo $\mathbf{e}$ è l'immagine del centro della seconda videocamera, nella seconda vista il punto $\mathbf{e}'$ è l'immagine del centro della prima videocamera. Un piano è detto epipolare se contiene la *baseline*; l'intersezione tra un piano epipolare e il piano immagine è detta retta epipolare.

Nel caso di due viste, la geometria epipolare può essere algebricamente rappresentata con la matrice fondamentale `F`, di cui mostriamo brevemente la derivazione.

Passo 1: trasferimento di un punto attraverso un piano Sia π un piano non passante per i centri $\mathbf{C}$ e $\mathbf{C}'$ delle videocamere come mostrato in Figura 1.3, il punto $\mathbf{X}$ è proiettato in $\mathbf{x}$ sul primo piano immagine, e in $\mathbf{x}'$ sul secondo: tra $\mathbf{x}$ e $\mathbf{x}'$ il passaggio avviene tramite il piano π. Dato che $\mathbf{X}$ giace sulla congiungente $\mathbf{C} - \mathbf{x}$, il punto proiettato $\mathbf{x}'$ appartiene alla retta epipolare $\mathbf{l}'$. I punti $\mathbf{x}$ e $\mathbf{x}'$ sono quindi entrambi immagini del punto 3D $\mathbf{X}$. Questo ragionamento vale per ogni coppia di punti corrispondenti $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$. In conseguenza, l'insieme di tutti i punti $\mathbf{x}_i$ della prima immagine e dei corrispondenti $\mathbf{x}'_i$ della seconda sono proiettivamente equivalenti, dato che sono

proiettivamente equivalenti all'insieme planare dei punti $\mathbf{X}_i$. Esiste quindi un'omografia $\mathtt{H}_\pi$ che mappa ogni $\mathbf{x}_i$ nel rispettivo $\mathbf{x}'_i$.

Passo 2: costruzione della retta epipolare Dato un punto $\mathbf{x}'$, la retta epipolare $\mathbf{l}'$ che passa per $\mathbf{x}'$ e per l'epipolo $\mathbf{e}'$ è $\mathbf{l}' = \mathbf{e}' \times \mathbf{x}' = [\mathbf{e}']_\times \mathbf{x}'$, dove l'operatore $[\mathbf{e}']_\times$ è così definito:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

e rappresenta il prodotto vettoriale in forma matriciale.

Dato che $\mathbf{x}'$ può essere scritto come $\mathbf{x}' = \mathtt{H}_\pi \mathbf{x}$, abbiamo

$$\mathbf{l}' = [\mathbf{e}']_\times \mathtt{H}_\pi \mathbf{x} = \mathtt{F}\mathbf{x} \tag{1.3}$$

avendo definito la matrice fondamentale $\mathtt{F} = [\mathbf{e}']_\times \mathtt{H}_\pi$.

La matrice fondamentale è di rango 2, in quanto $rank([\mathbf{e}']_\times) = 2$ e $rank(\mathtt{H}_\pi) = 3$. Gode inoltre delle seguenti proprietà:

1. Corrispondenze di punti: per qualsiasi coppia di punti corrispondenti $\mathbf{x}$, $\mathbf{x}'$ che siano immagine dello stesso punto $\mathbf{X}$, vale

$$\mathbf{x}'^\top \mathtt{F}\mathbf{x} = 0 \tag{1.4}$$

2. Trasposizione: se $\mathtt{F}$ è la matrice fondamentale della coppia di videocamere $(\mathtt{P}, \mathtt{P}')$, allora $\mathtt{F}^\top$ è la matrice fondamentale della coppia di videocamere $(\mathtt{P}', \mathtt{P})$.

3. Rette epipolari: per ogni punto $\mathbf{x}$ della prima immagine, la corrispondente retta epipolare è $\mathbf{l}' = \mathtt{F}\mathbf{x}$.

4. Gli epipoli: per ogni punto $\mathbf{x}$ diverso da $\mathbf{e}$, la retta epipolare $\mathbf{l}' = \mathtt{F}\mathbf{x}$ contiene l'epipolo $\mathbf{e}'$. Quindi $\mathbf{e}'$ soddisfa $\mathbf{e}'^\top(\mathtt{F}\mathbf{x}) = (\mathbf{e}'^\top \mathtt{F})\mathbf{x} = 0$ per ogni $\mathbf{x}$, ovvero $\mathbf{e}'$ è l'autovettore che genera lo spazio nullo sinistro di $\mathtt{F}$. Analogamente, $\mathtt{F}\mathbf{e} = 0$, quindi l'epipolo $\mathbf{e}$ è il vettore nullo destro di $\mathtt{F}$.

5. Gradi di libertà: $\mathtt{F}$ ha 7 gradi di libertà, dato che essendo una matrice omogenea è definita a meno della scala; inoltre $\det\mathtt{F} = 0$, poiché è di rango 2, cosa che toglie un altro grado di libertà.

Estrazione delle matrici delle videocamere dalla matrice F La matrice fondamentale può essere usata per calcolare le matrici delle videocamere corrispondenti, e quindi ottenere, conoscendo la calibrazione K, i parametri estrinseci delle videcamere, e quindi la rototraslazione relativa tra i due punti di vista. Se non è nota la calibrazione, invece, vale il seguente teorema:

Teorema 4. *Se* H *è una matrice* 4×4 *che rappresenta un'omografia in* $\mathbb{R}^3$*, allora le matrici fondamentali corrispondenti alle coppie di videocamere* $(\mathtt{P}, \mathtt{P}')$ *e* $(\mathtt{PH}, \mathtt{P}'\mathtt{H})$ *sono uguali.*

Questo risultato implica che dalla sola matrice fondamentale è possibile estrarre le matrici delle videocamere *a meno un una trasformazione proiettiva.*

Coordinate normalizzate e matrice essenziale Consideriamo la matrice della videocamera decomposta come $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]$, e sia $\mathbf{x} = \mathtt{P}\mathbf{X}$ un punto dell'immagine. Se conosciamo la matrice di calibrazione K possiamo applicare la sua inversa a $\mathbf{x}$ in modo da ottenere delle coordinate normalizzate, che non dipendono dalla calibrazione ma solo dalla posizione della videocamera: $\hat{\mathbf{x}} = \mathtt{K}^{-1}\mathbf{x}$, che equivale ad avere punti immagine generati da una videocamera con matrice normalizzata $\hat{\mathtt{P}} = [\mathtt{R} \mid \mathbf{t}]$.

Consideriamo la coppia di videocamere con matrici normalizzate $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$ e $\mathtt{P}' = [\mathtt{R} \mid \mathbf{t}]$. La matrice fondamentale corrispodente è chiamata *matrice essenziale*, e ha la forma

$$\mathtt{E} = [\mathbf{t}]_\times \mathtt{R} = \mathtt{R}[\mathtt{R}^\top \mathbf{t}]_\times.$$

Definizione 5. L'equazione che definisce la matrice essenziale è

$$\hat{\mathbf{x}}'^\top \mathtt{E} \hat{\mathbf{x}} = 0 \tag{1.5}$$

che è equivalente a quella della matrice fondamentale, salvo per l'uso delle coordinate normalizzate.

Sostituendo $\hat{\mathbf{x}}$ e $\hat{\mathbf{x}}'$ nella 1.5 abbiamo il legame tra matrice fondamentale ed essenziale:

$$\mathbf{x}'^\top \mathtt{K}^{-\top} \mathtt{E} \mathtt{K}^{-1} \mathbf{x} = 0$$

da cui

$$\mathtt{E} = \mathtt{K}'^\top \mathtt{F} \mathtt{K}.$$

La matrice essenziale ha solo 5 gradi di libertà: rotazione e traslazione hanno ognuna tre gradi di libertà, ma la ambiguità di scala, dato che la matrice è una quantità omogenea, li riduce di uno. Rispetto alla matrice fondamentale quindi E soddisfa ulteriori vincoli: è infatti possibile dimostrare la seguente

Proposizione 6. *Una matrice* 3×3 *è una matrice essenziale se e solo se due dei suoi valori singolari sono uguali, e il terzo è nullo.*

La scala della matrice essenziale dipende dai due valori singolari uguali, che di solito vengono posti pari a 1, in modo da fissare la scala a un valore convenzionale.

Il vantaggio della matrice essenziale è che da essa è possibile estrarre le matrici delle videocamere senza l'ambiguità proiettiva che è presente con la matrice fondamentale. Definite infatti le matrici

$$\mathtt{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{e} \qquad \mathtt{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

vale il seguente risultato:

Proposizione 7. *Data la matrice essenziale con scomposizione ai valori singolari* $\mathtt{E} = \mathtt{U}\mathrm{diag}(1,1,0)\mathtt{V}^\top$*, se la matrice della prima videocamera è* $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$*, ci sono quattro possibili scelte per la matrice* $\mathtt{P}'$ *della seconda videocamera, e sono:*

$$\mathtt{P}' = \left[\mathtt{U}\mathtt{W}\mathtt{V}^\top \mid +\mathbf{u}_3\right] \text{ o } \left[\mathtt{U}\mathtt{W}\mathtt{V}^\top \mid -\mathbf{u}_3\right] \text{ o } \left[\mathtt{U}\mathtt{W}^\top\mathtt{V}^\top \mid +\mathbf{u}_3\right] \text{ o } \left[\mathtt{U}\mathtt{W}^\top\mathtt{V}^\top \mid -\mathbf{u}_3\right] \quad (1.6)$$

essendo $\mathbf{u}_3$ *la terza colonna di* $\mathtt{U}$*.*

Ricordando che la matrice di una videocamera ha la forma $\mathtt{P} = [\mathtt{R} \mid \mathbf{t}]$, vediamo che esistono due possibili risultati sia per $\mathtt{R}$ che per $\mathbf{t}$, cosa che dipende dalle possibili configurazioni spaziali delle videocamere, in termini di rotazione e traslazione, che producono punti immagine consistenti con quelli misurati. Si tratta sostanzialmente di rotazioni o traslazioni delle videocamere che produrrebbero gli stessi punti sull'immagine. Delle quattro configurazioni una sola è quella reale, in cui i punti ricostruiti sono di fronte ad entrambe le videocamere, ed è possibile selezionarla utilizzando un punto dell'immagine come campione, in modo da scegliere la soluzione in cui $\mathbf{C} - \mathbf{x} - \mathbf{X}$, rispettivamente centro della videocamera, punto immagine e punto ricostruito, e $\mathbf{C}' - \mathbf{x}' - \mathbf{X}$ siano allineati in quest'ordine.

Ricordiamo che per costruzione $\|\mathbf{u}_3\| = 1$, ovvero la scala della ricostruzione è fissata una volta imposto come unitario il valore della *baseline*.

Il tensore trifocale

Nel caso di tre viste, il tensore trifocale gioca lo stesso ruolo della matrice fondamentale, ovvero ingloba la geometria epipolare e i vincoli proiettivi che gli oggetti presenti nelle tre viste hanno tra loro.

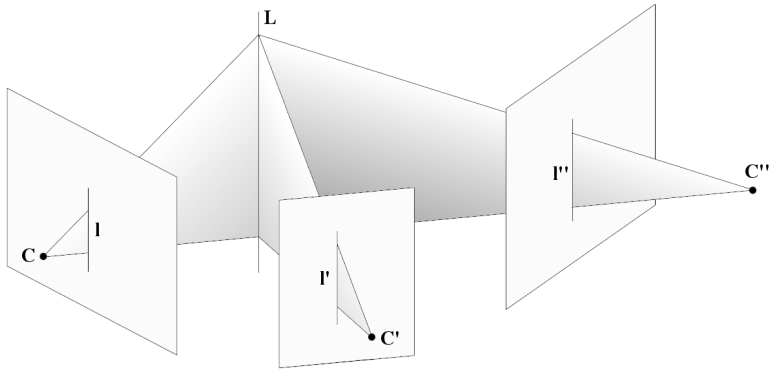

Figura 1.4: Rappresentazione dei vincoli di geometria epipolare per tre viste: i tre piani uscenti dalle videocamere devono incontrarsi in una retta.

Supponiamo che una retta nello spazio sia vista da tre videocamere diverse e proviamo a esprimere algebricamente i vincoli geometrici della sua proiezione nelle tre immagini. Siano $\mathbf{l}_i \longleftrightarrow \mathbf{l}'_i \longleftrightarrow \mathbf{l}''_i$ le tre proiezioni della retta $\mathbf{L}$, e siano $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = [\mathtt{A} \mid \mathbf{a}_4]$ e $\mathtt{P}'' = [\mathtt{B} \mid \mathbf{b}_4]$ le matrici delle videocamere, dove $\mathtt{A}$ e $\mathtt{B}$ sono matrici 3×3 di cui $\mathbf{a}_i$ e $\mathbf{b}_i$ sono le colonne. Il fatto di fissare la prima camera non fa perdere di generalità al procedimento, in quanto, come abbiamo visto nella sezione precedente, è possibile effettuare ricostruzioni 3D solo a meno di una trasformazione proiettiva, in assenza di ulteriori informazioni, esterne alle immagini. Allora:

- $\mathbf{a}_4$ e $\mathbf{b}_4$ sono epipoli, immagini del centro ottico della prima videocamera visualizzati nella seconda e terza immagine rispettivamente, e li indicheremo come $\mathbf{e}' = \mathtt{P}'\mathbf{C}$ ed $\mathbf{e}'' = \mathtt{P}''\mathbf{C}$
- $\mathtt{A}$ e $\mathtt{B}$ sono le omografie tra la prima e la seconda videocamera, e tra la prima e la terza.

Ricordiamo ora un risultato di geometria proiettiva:

Proposizione 8. *Data una videocamera di matrice* $\mathtt{P}$*, e una retta* $\mathbf{l}$ *sul piano immagine, il piano generato nello spazio dai raggi passanti per il centro della videocamera e per la retta* $\mathbf{l}$ *ha equazione* $\mathtt{P}^\top\mathbf{l}$*.*

Le tre rette sui piani immagine proiettano tre piani che hanno equazioni

$$\pi = \mathtt{P}^\top\mathbf{l} = \begin{pmatrix} \mathbf{l} \\ 0 \end{pmatrix} \quad \pi' = \mathtt{P}'^\top\mathbf{l}' = \begin{pmatrix} \mathtt{A}^\top\mathbf{l}' \\ \mathbf{a}_4^\top\mathbf{l}' \end{pmatrix} \quad \pi'' = \mathtt{P}''^\top\mathbf{l}'' = \begin{pmatrix} \mathtt{B}^\top\mathbf{l}'' \\ \mathbf{b}_4^\top\mathbf{l}'' \end{pmatrix}$$

che devono incontrarsi nella retta $\mathbf{L}$, vincolo che si esprime algebricamente richiedendo che la matrice 4×3 $\mathtt{M} = [\pi, \pi', \pi'']$ abbia rango 2. Infatti, i punti sulla

retta $\mathbf{L}$ sono del tipo $\mathbf{X} = \alpha\mathbf{X}_1 + \beta\mathbf{X}_2$, con $\mathbf{X}_1$ e $\mathbf{X}_2$ linearmente indipendenti. I punti $\mathbf{X}$ appartengono inoltre ai tre piani, dunque $\pi^\top\mathbf{X} = \pi'^\top\mathbf{X} = \pi''^\top\mathbf{X} = 0$. Segue che $\mathtt{M}^\top\mathbf{X} = 0$, quindi $\mathtt{M}$ ha uno spazio nullo di dimensione 2, dato che $\mathtt{M}^\top\mathbf{X}_1 = 0$ e $\mathtt{M}^\top\mathbf{X}_2 = 0$.

Dunque le colonne di $\mathtt{M}$ sono linearmente dipendenti, ovvero, data

$$\mathtt{M} = [\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3] = \begin{bmatrix} \mathbf{l} & \mathtt{A}^\top\mathbf{l}' & \mathtt{B}^\top\mathbf{l}'' \\ 0 & \mathbf{a}_4^\top\mathbf{l}' & \mathbf{b}_4^\top\mathbf{l}'' \end{bmatrix}$$

si può scrivere $\mathbf{m}_1 = \alpha\mathbf{m}_2 + \beta\mathbf{m}_3$. Grazie allo zero presente nella matrice $\mathtt{M}$, vale $\alpha = k\left(\mathbf{b}_4^\top\mathbf{l}''\right)$ e $\beta = -k\left(\mathbf{a}_4^\top\mathbf{l}'\right)$ per qualche k. Sostituendo nelle prime tre righe abbiamo, a meno di un fattore di scala

$$\mathbf{l} = \left(\mathbf{b}_4^\top\mathbf{l}''\right)\mathtt{A}^\top\mathbf{l}' - \left(\mathbf{a}_4^\top\mathbf{l}'\right)\mathtt{B}^\top\mathbf{l}'' = \left(\mathbf{l}''^\top\mathbf{b}_4\right)\mathtt{A}^\top\mathbf{l}' - \left(\mathbf{l}'^\top\mathbf{a}_4\right)\mathtt{B}^\top\mathbf{l}''.$$

La i-esima coordinata di $\mathbf{l}$ sarà quindi

$$l_i = \mathbf{l}'^\top\left(\mathbf{a}_i\mathbf{b}_4^\top\right)\mathbf{l}'' - \mathbf{l}'^\top\left(\mathbf{a}_4\mathbf{b}_i^\top\right)\mathbf{l}''$$

e introducendo la notazione

$$\mathtt{T}_i = \mathbf{a}_i\mathbf{b}_4^\top - \mathbf{a}_4\mathbf{b}_i^\top$$

la relazione può essere scritta come

$$l_i = \mathbf{l}'^\top\mathtt{T}_i\mathbf{l}''.$$

Definizione 9. L'insieme delle tre matrici $\{\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3\}$ compone il tensore trifocale in notazione matriciale.

Le matrici $\mathtt{T}_i$ sono quindi "fette" di un tensore $3\times3\times3$, che esprime i vincoli tra i punti immagine in tre viste. Del tensore trifocale è possibile dimostrare le seguenti proprietà:

- Il tensore ha 18 gradi di libertà: ogni videocamera ha 11 gradi di libertà, ma l'ambiguità proiettiva del sistema di riferimento fissato dalla matrice della prima videocamera toglie 15 gradi di libertà, dunque $33 - 15 = 18$.
- Corrispondenze tra rette: per qualsiasi tripletta di rette corrispondenti $\mathbf{l}$, $\mathbf{l}'$ e $\mathbf{l}''$ che siano immagine della stessa retta $\mathbf{L}$, vale

$$\mathbf{l}'^\top\left[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3\right]\mathbf{l}'' = \mathbf{l}^\top \tag{1.7}$$

- Corrispondenze tra punti: per qualsiasi tripletta di punti corrispondenti $\mathbf{x}$, $\mathbf{x}'$ e $\mathbf{x}''$ che siano immagine dello stesso punto $\mathbf{X}$, vale

$$[\mathbf{x}']_\times \left(\sum_i x^i \mathtt{T}_i \right) [\mathbf{x}'']_\times = \mathtt{0}_{3\times 3} \tag{1.8}$$

- Ogni matrice $\mathtt{T}_i$ è di rango 2, e anche la somma $(\sum_i x^i \mathtt{T}_i)$ ha rango 2.

- Epipoli: dati $\mathbf{u}_i$ e $\mathbf{v}_i$, rispettivamente i vettori nulli sinistri e destri delle matrici $\mathtt{T}_i$, gli epipoli sono i vettori nulli delle seguenti matrici:

$$\mathbf{e}'^\top [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] = 0 \qquad \text{e} \qquad \mathbf{e}''^\top [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] = 0$$

- Matrici fondamentali: dato il tensore trifocale $[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]$, le matrici fondamentali tra la prima e seconda, e prima e terza vista sono

$$\mathtt{F}_{21} = [\mathbf{e}']_\times [\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]\, \mathbf{e}'' \qquad \text{e} \qquad \mathtt{F}_{31} = [\mathbf{e}'']_\times \left[\mathtt{T}_1^\top, \mathtt{T}_2^\top, \mathtt{T}_3^\top \right] \mathbf{e}'$$

- Le matrici delle videocamere, posta $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, si trovano, una volta normalizzati gli epipoli, come

$$\mathtt{P}' = [[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]\, \mathbf{e}'' \mid \mathbf{e}'] \qquad \text{e} \qquad \mathtt{P}'' = \left[\left(\mathbf{e}'' \mathbf{e}''^\top - \mathtt{I} \right) \left[\mathtt{T}_1^\top, \mathtt{T}_2^\top, \mathtt{T}_3^\top \right] \mathbf{e}' \mid \mathbf{e}'' \right]$$

  ed è importante notare che le tre matrici sono pari alle effettive matrici di videocamera a meno di una trasformazione proiettiva, come per il caso della matrice fondamentale.

Ricostruzione 3D

Il nostro scopo è quello di riuscire a ricostruire la struttura 3D di una scena e il posizionamento delle videocamere, problemi che sono strettamente legati tra loro. Infatti, una volta note le matrici delle videocamere e una serie di corrispondenze tra i punti immagine delle diverse viste, è possibile ricostruire i punti nello spazio mediante tecniche di triangolazione. La triangolazione è il procedimento attraverso cui, noti due punti e due vettori uscenti da tali punti, viene calcolato il punto che forma il terzo vertice del triangolo. Il livello di precisione con cui riusciamo a calcolare le matrici delle videocamere determina quindi la precisione nella ricostruzione.

Gli elementi che abbiamo a disposizione sono delle corrispondenze $\mathbf{x}_i \longleftrightarrow \mathbf{x}'_i$ di punti o rette nelle immagini, per cui non sono noti i punti nello spazio $\mathbf{X}_i$, le matrici delle videocamere e la geometria epipolare. Con un numero sufficiente di corrispondenze è possibile calcolare la geometria epipolare, e come visto nei paragrafi precedenti, estrarre da essa le matrici delle videocamere.

Il metodo base utilizzato è come segue:

1. Calcolo della geometria epipolare (matrice fondamentale, tensore trifocale) da corrispondenze di punti immagine

2. Calcolo delle matrici delle videocamere dalla geometria epipolare

3. Calcolo dei punti nello spazio $\mathbf{X}_i$ mediante triangolazione, proiettando i raggi dei punti $\mathbf{x}_i$ e $\mathbf{x}'_i$ che partono dai centri delle videocamere.

I metodi per il calcolo della geometria epipolare a partire da corrispondenze saranno trattati nel capitolo successivo, mentre in questo paragrafo ci concentriamo sul livello di precisione che è possibile raggiungere nella ricostruzione.

Ambiguità di ricostruzione Senza qualche informazione del posizionamento della scena rispetto a un sistema di riferimento, è innanzitutto chiaro che la scena può essere ricostruita al più con un'ambiguità di rototraslazione - ad esempio, a partire da due fotografie di un corridoio non è possibile conoscere latitudine e longitudine, né orientamento del corridoio stesso.

Nemmeno la scala può essere ricostruita, in quanto dalle foto non è possibile sapere se il corridoio sia alto 3 metri o 3 centimetri. Nell'esperienza comune, il senso della vista ci restituisce il senso delle dimensioni solo quando nella scena è presente qualche oggetto noto, di cui quindi conosciamo l'altezza approssimativa e con cui possiamo "fissare la scala".

Senza conoscere la matrice di calibrazione `K` abbiamo visto che è possibile ricostruire le matrici delle videocamere a meno di una trasformazione proiettiva, a partire dalla sola matrice fondamentale. Questo implica che le videocamere e quindi anche la scena ricostruita saranno soggette ad un'ambiguità proiettiva. Questo risultato è generalizzabile al caso di più viste.

È possibile migliorare questo risultato aggiungendo ulteriori informazioni, come la matrice `K` o conoscendo dimensioni o posizionamento reciproco di punti nella scena, per arrivare ad una ricostruzione metrica della scena, in cui sia cioè possibile misurare distanze, angoli e aree. L'obiettivo nel nostro caso, in cui vogliamo ricostruire il movimento di una videocamera, è proprio quello della ricostruzione metrica.

Qualora sia nota la matrice essenziale è possibile, come visto nei paragrafi precedenti, calcolare le matrici delle videocamere, e di conseguenza anche la scena 3D, a meno dell'ambiguità di scala: ci troviamo quindi già in ambito metrico.

Rimane naturalmente l'ambiguità di scala, che è necessario fissare utilizzando informazioni esterne che forniscano la distanza reale tra punti nello spazio. Nel caso di coppie di videocamere, la *baseline* fornisce questa scala; nel caso invece di una videocamera che si sposti nello spazio, è possible fare ricorso a dati GPS che misurino la posizione della videocamera.

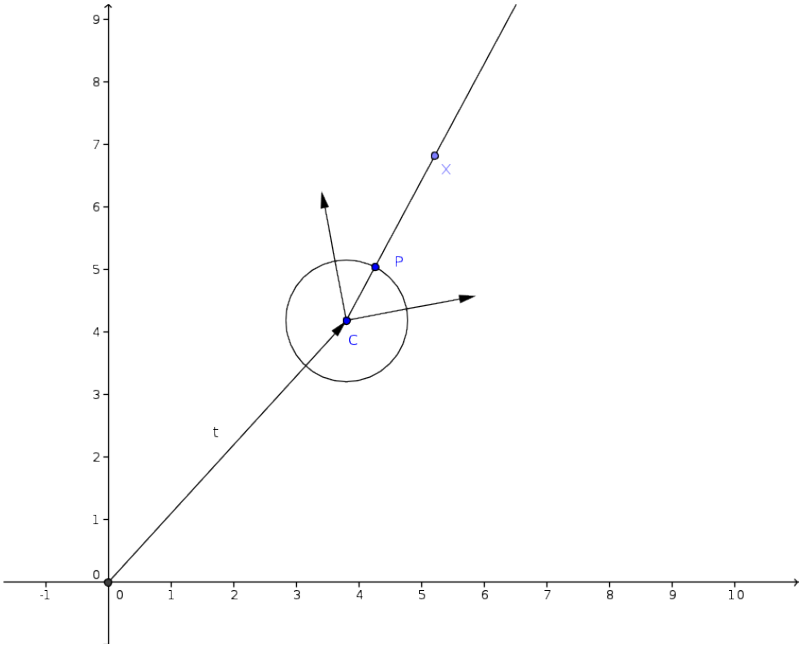

Figura 1.5: Il modello di videocamera omnidirezionale, per semplicità visualizzato sul piano: il sistema di riferimento locale è identificato da $(\mathtt{R}, \mathbf{t})$ rispetto al sistma di riferimento globale. Il punto $\mathbf{X}$ è proiettato sulla sfera mediante normalizzazione.

1.3 Videocamere omnidirezionali

Il modello di videocamera omnidirezionale Parliamo di camere omnidirezionali quando l'angolo di visuale supera i 180° ([13]). In questo caso il modello a *pinhole camera* non è più adeguato per descrivere la formazione dell'immagine, in quanto al massimo i raggi uscenti dal centro della videocamera e passanti per il piano immagine possono coprire un emispazio. La proiezione nel modello tradizionale è formalizzabile come

$$\exists \alpha : \ \alpha \mathbf{x} = \mathtt{P}\mathbf{X}$$

con $\mathtt{P} \in \mathbb{R}^{3\times 4}$ è una trasformazione proiettiva, $\mathbf{X} \in \mathbb{R}^4 - \mathbf{0}$ è un punto omogeneo dello spazio e $\mathbf{x} \in \mathbb{R}^3 - \mathbf{0}$ è un punto omogeneo sul piano immagine.

Il modello di camera omnidirezionale, con angolo di visuale a 360°, è visualizzato in Figura 1.5: il centro della videocamera è il punto $\mathbf{C}$, centro di una sfera di raggio unitario, e i raggi partono dal centro e incontrano la superficie S^2 della sfera, che è l'equivalente del piano immagine. I punti dello spazio vengono quindi proiettati sulla superficie della sfera, e i raggi sono semirette.

Formalmente, la proiezione avviene come

$$\exists \alpha : \ \alpha \mathbf{x} = \mathtt{P}\mathbf{X}$$

dove $\mathtt{P}$ e $\mathbf{X}$ sono gli stessi di prima e $\mathbf{x}$ è un vettore di $\mathbb{R}^3$, vincolato sulla superficie S^2 che rappresenta un punto sull'immagine. Se $\mathbf{C}$ è l'origine degli assi, l'equazione della proiezione è semplicemente

$$\mathbf{x} = \frac{1}{\|\mathbf{X}\|}\mathbf{X}.$$

È importante notare che $\mathbf{x}$ non è in notazione omogenea, anche se in realtà ha gli stessi gradi di libertà di un punto sul piano. Se passiamo in coordinate sferiche, infatti, un punto $\mathbf{x}$ è identificato da due angoli, ad esempio inclinazione dal piano verticale $0 \leq \varphi \leq \pi$, che corrisponde alla latitudine e azimuth $0 \leq \theta \leq 2\pi$, che corrisponde alla longitudine:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\theta \sin\varphi \\ \sin\theta \sin\varphi \\ \cos\varphi \end{pmatrix}.$$

Il modello sferico centrato nell'origine degli assi può essere generalizzato separando il sistema di riferimento locale, della sfera, da quello globale. Una videocamera omnidirezionale sarà quindi caratterizzata da una terna cartesiana, identificate da una matrice di rotazione $\mathtt{R}$ e da un vettore di traslazione $\mathbf{t}$ che esprimono il cambio di coordinate tra il sistema di riferimento globale e quello locale. Per passare dal sistema globale S a quello locale S' basta effettuare quindi il cambio di coordinate

$$\mathbf{X}' = \mathtt{R}\left(\mathbf{X} - \mathbf{t}\right),$$

con analoga formula inversa

$$\mathbf{X} = \mathtt{R}^{-1}\mathbf{X}' + \mathbf{t}.$$

Essendo le videocamere omnidirezionali prive di uno specifico orientamento, la matrice $\mathtt{R}$ è sostanzialmente legata a come viene salvata l'immagine in formato digitale; il vettore $\mathbf{t}$ esprime la posizione della videocamera nello spazio.

Geometria proiettiva sulla sfera Con il modello di camera omnidirezionale viene definita una geometria proiettiva basata sulla dualità tra cerchi massimi e punti sulla sfera, analogamente a come nella geometria proiettiva sul piano la dualità è tra punti e rette.

Data una retta $\mathbf{L}$ nello spazio, esiste un piano π passante per $\mathbf{L}$ e per il centro $\mathbf{C}$, che definisce un cerchio massimo C sulla sfera, ovvero un cerchio di diametro massimo. Il vettore definisce a sua volta lo stesso cerchio massimo è definibile anche a partire dal vettore $\mathbf{n} \in S^2$ normale a questo piano; anche il vettore $-\mathbf{n}$ definisce lo stesso piano, per cui esiste una corrispondenza biunivoca tra i punti sull'emisfero positivo della sfera S^2_+ e i cerchi massimi sulla sfera. Possiamo quindi dare la seguente definizione:

Definizione 10. Dato un vettore $\mathbf{n} \in S^2_+$ e $\mathbf{x} \in \mathbb{R}^3$, un cerchio massimo C è definito come

$$C = \left\{ \mathbf{x} \mid \mathbf{n}^\top \mathbf{x} = 0,\ \|\mathbf{x}\| = 1 \right\}.$$

Formalmente, le funzioni che mappano punti in cerchi massimi e viceversa sono:

$$f(C) = \lambda \frac{\mathbf{x} \times \mathbf{y}}{\|\mathbf{x} \times \mathbf{y}\|},\ \mathbf{x}, \mathbf{y} \in C,\ \lambda \in \{-1, 1\}$$

$$f^{-1}(\mathbf{n}) = \left\{ \mathbf{x} \mid \mathbf{n}^\top \mathbf{x} = 0,\ \|\mathbf{x}\| = 1,\ \mathbf{n} \in S^2_+ \right\}$$

dove λ è scelto in modo che $f(C) \in S^2_+$.

Analogamente è possibile definire una corrispondenza tra rette su un piano e punti sulla superficie sferica:

$$g(\mathbf{l}) = \lambda \frac{\xi \times \eta}{\|\xi \times \eta\|},\ \xi = \left(\mathbf{x}^\top, 1\right),\ \eta = \left(\mathbf{y}^\top, 1\right),\ \mathbf{x}, \mathbf{y} \in \mathbf{l}$$

$$g^{-1}(\mathbf{n}) = \left\{ \xi \mid \mathbf{n}^\top \xi = 0,\ \xi = \left(\mathbf{x}^\top, 1\right),\ \mathbf{n} \in S^2_+ \right\}$$

Geometria epipolare per due viste Siano $\mathbf{C}$ e $\mathbf{C}'$ i centri di due camere, e siano $\mathtt{R}$ e $\mathbf{t}$ matrice di rotazione e vettore di traslazione che definiscono la trasformazione delle coordinate tra la prima camera e la seconda. Assumiamo per semplicità $\mathbf{C} = \mathbf{0}$. La proiezione dei punti dello spazio sulle sfere è rispettivamente, per la prima e la seconda videocamera:

$$\lambda \mathbf{x} = \mathbf{X},\ \lambda \in \mathbb{R}$$

$$\lambda' \mathbf{x}' = \mathbf{X}' = \mathtt{R}\mathbf{X} + \mathbf{t}\ \lambda' \in \mathbb{R}.$$

Abbiamo quindi l'equazione

$$\lambda' \mathbf{x}' - \mathtt{R}\mathbf{X} - \mathbf{t} = 0.$$

I vettori che compongono la somma dell'equazione formano un piano, per cui come visto nel caso della matrice fondamentale è possibile scrivere il vincolo epipolare per camere omnidirezionali:

$$\mathbf{x}'^\top \mathtt{E} \mathbf{x} = 0,\ \mathtt{E} = [\mathbf{t}]_\times \mathtt{R}.$$

Osserviamo che ci troviamo già nel caso di matrice essenziale, in quanto il modello omnidirezionale di videocamera non ingloba informazioni quali la lunghezza focale o la dimensione dei pixel, ma si limita a tenere conto del solo posizionamento spaziale della videocamera. I punti $\mathbf{x} \in S^2$ sono quindi interpretabili come coordinate immagine normalizzate, e forniscono vincoli sufficienti a calcolare la matrice essenziale. La ricostruzione delle matrici delle videocamere e della scena 3D nel caso di videocamere omnidirezionali è dunque in ambito metrico.

I cerchi epipolari È utile vedere come cambia la retta epipolare nel caso sferico. La dualità tra punti sulla sfera e cerchi massimi implica che due punti $\mathbf{x}$ e $\mathbf{x}'$ individuano due cerchi massimi corrispondenti $\mathbf{l}$ e $\mathbf{l}'$, appartenenti ai piani π e π' di $\mathbb{R}^3$. L'intersezione di questi due piani, che passano per i centri delle videocamere, è la retta $\mathbf{L}$, come visualizzato in Figura 1.6. Chiamiamo $\mathbf{X}_0$ il punto di intersezione di questa retta con il piano epipolare che contiene $\mathbf{x}$, $\mathbf{x}'$ e $\mathbf{X}$, e anche i due centri $\mathbf{C}$, $\mathbf{C}'$, dato che i punti $\mathbf{C}$, $\mathbf{x}$ e $\mathbf{X}$ e gli analoghi $\mathbf{C}'$, $\mathbf{x}'$ e $\mathbf{X}$ sono allineati. Notiamo che $\mathbf{X}_0$ è anche il punto di intersezione tra la retta $\mathbf{s}$, passante per $\mathbf{C}$ e perpendicolare a $\mathbf{x} - \mathbf{C}$, e la retta $\mathbf{r}$, passante per $\mathbf{C}'$ e perpendicolare a $\mathbf{x}' - \mathbf{C}'$. Abbiamo quindi due triangoli rettangoli identificati dai punti $\mathbf{XCX}_0$ e $\mathbf{XC'X}_0$, che condividono la stessa ipotenusa e quindi anche lo stesso cerchio circoscritto, che è chiamato *cerchio epipolare.* Il cerchio epipolare ha centro $e_0 = \frac{1}{2}(\mathbf{X} + \mathbf{X}_0)$ e raggio $e_r = \frac{1}{2}\|\mathbf{X} + \mathbf{X}_0\|$.

Geometria epipolare per tre viste Anche per le videocamere omnidirezionali è possibile derivare il tensore trifocale seguendo la traccia vista nel caso di videocamere tradizionali ([20]).

Siano $\mathbf{C} = \mathbf{0}$, $\mathbf{C}'$ e $\mathbf{C}''$ i tre centri delle videocamere, $\mathtt{R}'$ e $\mathbf{t}'$ e $\mathtt{R}''$ e $\mathbf{t}''$ le trasformazioni di coordinate tra il sistema di riferimento della prima sfera e la seconda e tra la prima e la terza. Sia $\mathbf{L}$ una retta nello spazio, e siano $\mathbf{L}'$ e $\mathbf{L}''$ l'equazione della retta vista nei sistemi della seconda e terza sfera. La retta $\mathbf{L}$ viene proiettata sulle tre sfere sotto forma di cerchi massimi $\mathbf{l}$, $\mathbf{l}'$, e $\mathbf{l}''$, che possono essere espressi dualmente con i punti $\mathbf{a}$, $\mathbf{a}'$, e $\mathbf{a}'' \in S^2$. La relazione tra le rette $\mathbf{L}$, $\mathbf{L}'$ e $\mathbf{L}''$ e i punti $\mathbf{a}$, $\mathbf{a}'$, e $\mathbf{a}''$ può essere espressa come

$$\mathbf{a}^\top \mathbf{L} = 0$$

$$\mathbf{a}'^\top \mathbf{L}' = \mathbf{a}'^\top (\mathtt{R}'\mathbf{L} + \mathbf{t}') = 0$$
$$\mathbf{a}''^\top \mathbf{L}'' = \mathbf{a}''^\top (\mathtt{R}''\mathbf{L} + \mathbf{t}'') = 0$$

Le tre equazioni possono essere riunite nella matrice $\mathtt{M}$

$$\mathtt{M}\tilde{\mathbf{L}} = 0$$

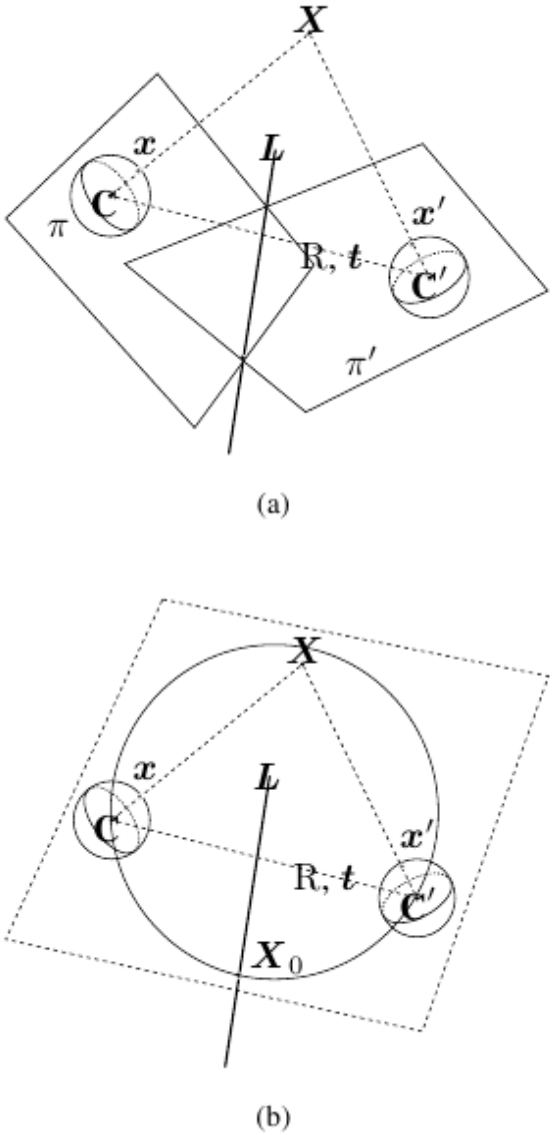

Figura 1.6: I cerchi epipolari nel caso di due viste omnidirezionali.

con

$$\mathtt{M} = \begin{bmatrix} \mathbf{a}^\top & 0 \\ \mathbf{a}'^\top \mathtt{R}' & \mathbf{a}'^\top \mathbf{t}' \\ \mathbf{a}''^\top \mathtt{R}'' & \mathbf{a}''^\top \mathbf{t}'' \end{bmatrix}$$

e $\tilde{\mathbf{L}}^\top = \left(\mathbf{L}^\top, 1\right)^\top$. Come nel caso tradizionale $\mathbf{L}$, $\mathbf{L}'$ e $\mathbf{L}''$ sono la stessa retta se e solo se rank$\mathtt{M} = 2$, ovvero se è possibile esprimere la prima colonna di $\mathtt{M}$ come combinazione lineare delle altre due:

$$\mathbf{m}_1 = \alpha \mathbf{m}_2 + \beta \mathbf{m}_3,$$

da cui si ottiene l'equazione

$$\mathbf{a} = \mathbf{a}''^\top \left(\mathbf{t}'' \mathtt{R}'^\top\right) \mathbf{a}' - \mathbf{a}'^\top \left(\mathbf{t}' \mathtt{R}''^\top\right) \mathbf{a}''$$

che espressa elemento per elemento diventa

$$a_i = \mathbf{a}'^\top \left(\mathtt{R}_i'^\top \mathbf{t}''\right) \mathbf{a}'' - \mathbf{a}'^\top \left(\mathbf{t}' \mathtt{R}_i''^\top\right) \mathbf{a}''$$

dove $\mathtt{R}_i'$ che è la i-esima riga della matrice di rotazione della seconda videocamera.

Il tensore trifocale è quindi la tripletta $\{\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3\}$ definita da $\mathtt{T}_i = \mathtt{R}_i'^\top \mathbf{t}'' - \mathbf{t}' \mathtt{R}_i''^\top$, che soddisfa le equazioni

$$a_i = \mathbf{a}'^\top \mathtt{T}_i \mathbf{a}'' \tag{1.9}$$

$$\mathbf{a}^\top = \mathbf{a}'^\top \left[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3\right] \mathbf{a}''.$$

Come nel caso tradizionale, la corrispondenza tra punti sulle tre sfere soddisfa l'equazione
1.8:

$$\left[\mathbf{x}'\right]_\times \left(\sum_i x^i \mathtt{T}_i\right) \left[\mathbf{x}''\right]_\times = \mathtt{0}_{3\times 3}.$$

È da sottolineare il fatto che i punti di cui stiamo parlando nel caso omnidirezionale non sono in notazione omogenea, ma sono punti nello spazio.

Cerchi epipolari e ricostruzione 3D Una tripletta di punti corrispondenti sulle sfere genera tre cerchi epipolari, come visto prima. Questo permette una ricostruzione 3D, come illustrato in figura 1.7b: i cerchi epipolari si devono incontrare nel punto ricostruito $\mathbf{X}$, se le corrispondenze sono esatte. Abbiamo quindi un modo per misurare l'errore di ricostruzione, nel caso i tre cerchi massimi non si incontrino in un punto.

Inoltre i tre cerchi massimi, nel caso ideale, si incontrano in un punto qualsiasi sia la configurazione spaziale del sistema di videocamere, dato che per tre punti non allineati (due centri di videocamera e il punto $\mathbf{X}$) passa sempre una circonferenza. Questo permette la ricostruzione 3D anche in casi particolari, come tre videocamere allineate, esempio in cui i tre piani epipolari non sarebbero linearmente indipendenti e non potrebbero essere usati per la triangolazione. In un caso di questo tipo, che assomiglia al caso di fotogrammi ripresi lungo il percorso di un'automobile, solo i punti giacenti sulla retta di moto non sono ricostruibili, mentre per il resto dello spazio non ci sono problemi.

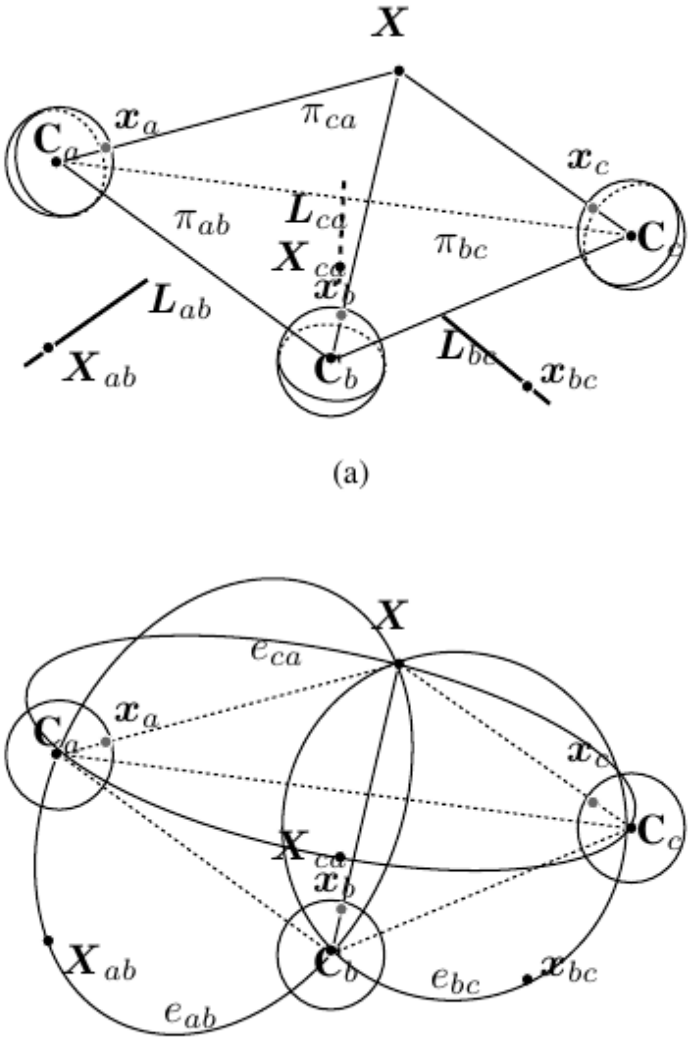

Figura 1.7: Cerchi epipolari nel caso di tre viste omnidirezionali

Capitolo 2

Metodi di stima e algoritmi

In questo capitolo vengono affrontati in ordine logico i vari passi necessari per l'intero processo che, da una serie di immagini omnidirezionali, restituisce una nuvola di punti 3D e le posizioni spaziali della videocamera. I passi nei quali si è fatto ricorso a strumenti di terze parti sono trattati brevemente.

2.1 Immagini omnidirezionali

Dati sperimentali del progetto Astute La tesi è stata svolta nell'ambito del progetto Astute, come visto nell'introduzione. Una delle aziende coinvolte ha messo a disposizione un software per la creazione di immagini panoramiche a 360° mediante *stitching* di diverse immagini prese con un grandangolo tradizionale, come mostrato in Figura 2.1. La raccolta dei dati mediante questa soluzione ha però subito dei ritardi nell'evoluzione del progetto, per cui non è stato possibile avere immagini sperimentali di questo tipo in quantità sufficiente per condurre un test approfondito.

Dati sperimentali da Google Street View Un'altra possibile fonte di dati è il servizio Google Street View, che offre una copertura della rete stradale eccezionale, a fronte di una minore qualità dei dati. Le "bolle" di Street View sono caricate nel browser come "tessere" di un mosaico. Attraverso un script in Python ideato all'interno dell'Image and Sound Processing Group del Politecnico di Milano, è stato possibile scaricare le tessere, di dimensione 512×512 pixel, e ricomporle ottenendo un'immagine panoramica completa, con il livello di zoom desiderato (si va da un minimo di 2×4 tessere a un massimo di 5×10).

Utilizzare le immagini di Google Street View comporta alcune problemi di qualità dei dati: in primo luogo le immagini sono soggette a *watermark*,

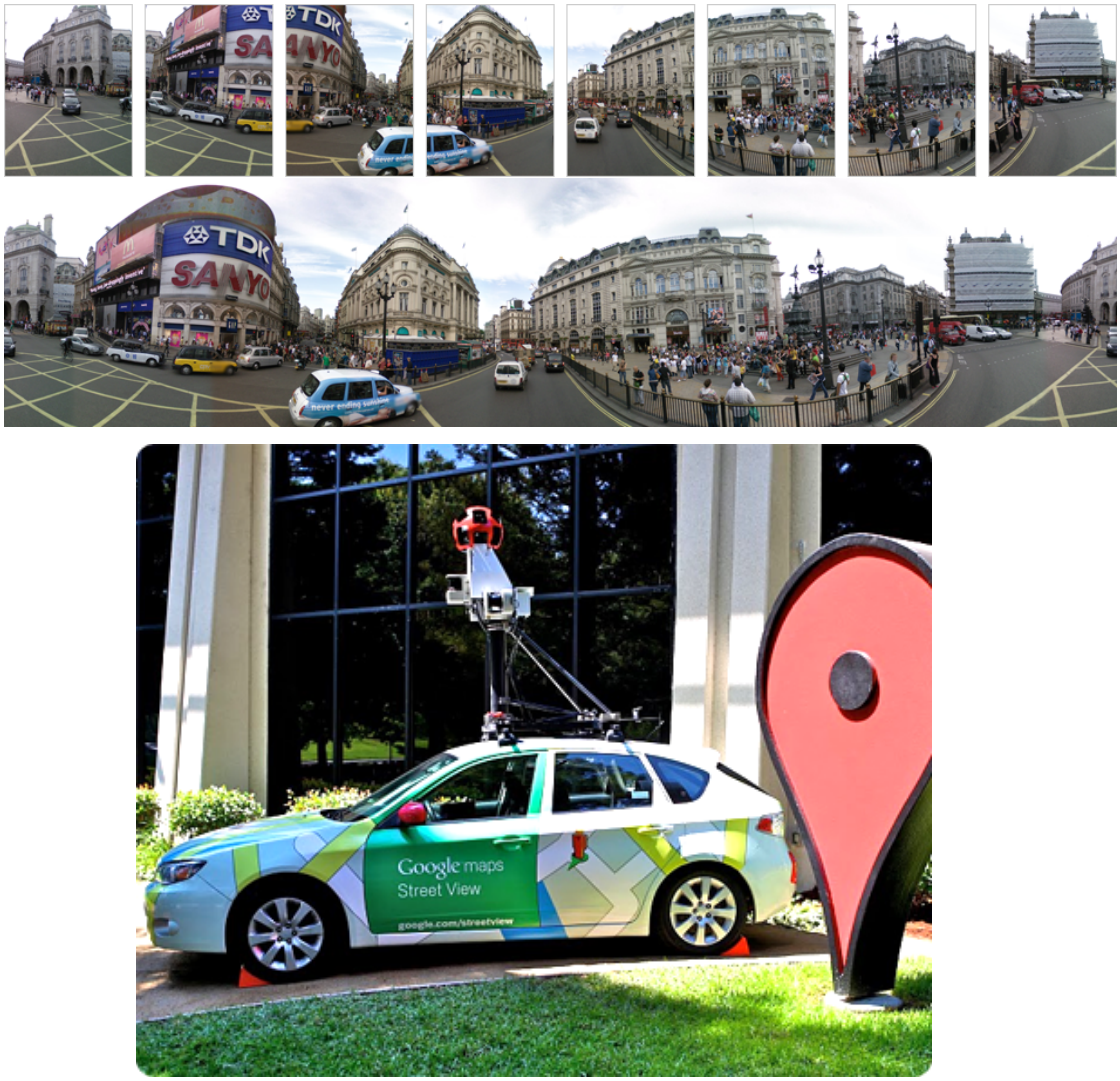

Figura 2.1: In alto, la formazione delle immagini panoramiche di Google Street View: a partire da diverse foto grandangolari viene creata l'immagine finale in formato equirettangolare. In basso, una delle automobili con cui Google raccoglie le immagini.

Figura 2.2: Particolare di una immagine di Google Street View di bassa qualità.

ovvero una scritta in sovraimpressione che le identifica come appartenenti a Google; in secondo luogo le zone di giunzione delle varie fotografie sono a volte visibili, come in Figura 2.2. Queste fonti di disturbo possono generare errori sistematici nella generazione dei punti notevoli dell'immagine.

Una seconda criticità viene dalla bassa qualità delle zone ad alta latitudine, ovvero vicino a *zenith* e *nadir* della sfera. In queste zone Google fornisce un'immagine interpolata, non nativa, per non mostrare il tetto della Google Car. È stato quindi deciso di rimuovere queste zone dell'immagine prima del passo di feature detection, conservando la parte centrale di immagine, con ampiezza un'angolare di 60°. In questo modo il carico computazionale è ridotto, anche se in situazioni di canyon urbano, quando gli edifici a lato della strada sono vicini tra loro, sarebbe utile avere anche informazioni su latitudini più elevate.

Inoltre, le fotografie presenti su Street View non sono la sequenza ordinata di viste prese da una sola automobile, che è l'ambito di azione per cui è pensato il metodo presentato in questa tesi. Le sequenze sono infatti frutto dei dati raccolti da diverse automobili in momenti diversi, per cui è possibile che da una bolla all'altra cambi parecchio lo scenario, sia in termini di luci e colori, che di elementi strutturali (es. lavori in corso) ma, cosa più importante, l'orientamento non è garantito. Le strade a due sensi di marcia ad esempio possono essere state raccolte in parte in un senso e in parte nell'altro senso, cosa che pregiudica la continuità dei dati.

Un ulteriore limite arriva dalla risoluzione massima utilizzabile dalle immagini: Google Street View utilizza infatti il formato equirettangolare, spiegato nel prossimo paragrafo, fino alla risoluzione di 833×1666 pixel; per risoluzioni maggiori l'angolo di visuale non è più 360°, ma è minore, di una quantità ignota. Questo effetto è particolarmente evidente visualizzando l'immagine omnidirezionale in proiezione stereografica, come mostrato in Figura 2.3. Per questo nell'algoritmo abbiamo potuto utilizzare fotografie a risoluzione non eccezionale, cosa che influisce sul numero di corrispondenze che si riescono a

Figura 2.3: Immagini panoramiche di Google Street View in proiezione stereografica (effetto “small worlds”). L’immagine in basso è la proiezione di un’immagine ad alta risoluzione, e risulta evidente come non sia panoramica a 360°.

ottenere tra una vista e l'altra.

Infine, la criticità maggiore delle immagini di Google è la sparsità: tipicamente tra una bolla e l'altra la *baseline* è di circa 10-15 metri, per cui il punto di vista di edifici e oggetti vicini alla strada può cambiare radicalmente tra una bolla e l'altra; ancora più importante è la criticità nel caso dell'utilizzo di tre viste, visto che la *baseline* tra la prima e la terza camera arriva a 30 metri. Il risultato è che le corrispondenze trovate possono essere troppo poche per la stima accurata del tensore trifocale. Uno degli articoli di riferimento ([19]) ha usato bolle di Street View ottenute chiedendole a Google per scopi di ricerca, dato che l'azienda in fase di raccolta dati utilizza una campionatura più fitta.

Immagini equirettangolari Sia le immagini di Google Street View che quelle generate per il progetto ASTUTE sono immagini panoramiche in formato equirettangolare. La proiezione equirettangolare, che è uno dei possibili modi per rappresentare una superficie sferica su un piano, viene effettuata campionando la sfera a passo angolare costante, sia in latitudine che in longitudine, con raggi che partono dal centro della sfera stessa. Un esempio è mostrato in Figura 2.4. Avendo N punti, avremo che il passo di capionamento sarà:

$$d\theta = \frac{2\pi}{N}.$$

Ad ogni pixel dell'immagine equirettangolare viene quindi assegnato il valore dell'immagine sulla sfera nel punto in cui il raggio corrispondente al pixel la interseca, dopo opportuna interpolazione. Il risultato è un'immagine fortemente deformata vicino ai poli, dove la campionatura sulla sfera è molto fitta. Una proprietà utile ai fini della feature detection è che la deformazione è costante ad una determinata latitudine, e che la zona vicino all'equatore, che nelle immagini realistiche è la più ricca di informazioni, è poco deformata. È inoltre molto semplice passare dalle coordinate immagine a quelle sulla sfera: essendo $0 \leq \varphi \leq \pi$ l'inclinazione rispetto al piano verticale e $0 \leq \theta \leq 2\pi$ l'azimuth, e dette (x, y) le coordinate del pixel sull'immagine, abbiamo:

$$\varphi = \pi \frac{y}{H}$$

$$\theta = 2\pi \frac{L - x}{L}$$

essendo (L, H) le dimensioni dell'immagine in pixel. Si noti che l'azimuth decresce con x, cosa che deriva dal fatto che se immaginiamo l'immagine incollata su una sfera, la coordinata orizzontale in pixel è crescente in senso orario, mentre l'azimuth è crescente in senso antiorario.

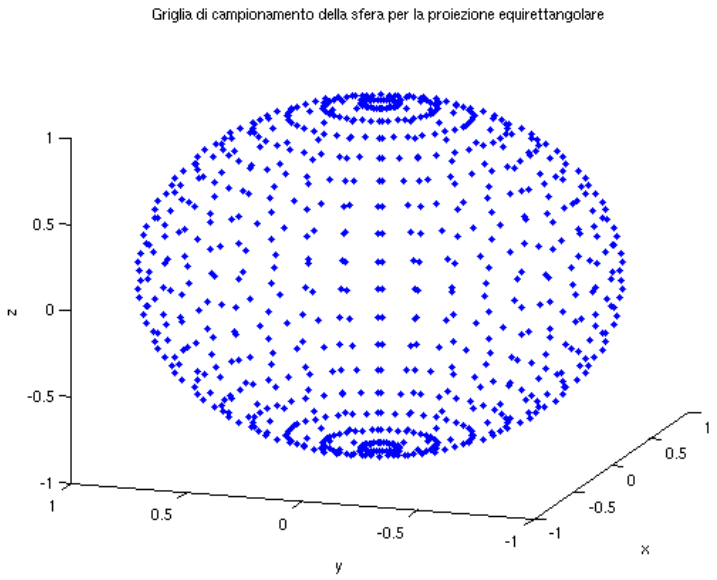

Figura 2.4: Griglia di campionamento della sfera per la proiezione equirettangolare: ad ogni punto corrisponderà un pixel dell'immagine. Dato che ogni cerchio orizzontale è formato dallo stesso numero di punti, la campionatura è più fitta vicino ai poli.

Dati sintetici Oltre alle immagini realistiche sono stati condotti anche test su dati sintetici, ottenuti generando una serie di punti 3D e proiettandoli sulle sfere. In questo modo si sono potute separare i problemi di *feature detection* da quelli della stima della geometria della scena.

2.2 Identificazione dei punti notevoli dell'immagine tramite SIFT

Il passo di identificazione dei punti notevoli dell'immagine è stato effettuato appoggiandosi su uno dei più noti metodi per la *feature detection*, la Scale-Invariant Feature Transform (SIFT). L'algoritmo produce, a partire da un'immagine, un insieme di punti notevoli. Ogni punto è corredato di descrittore, un vettore a 128 elementi che "riassume" le caratteristiche del punto stesso, ed è inviariante per traslazioni, riscalamenti e rotazioni, parzialmente invariante per cambi di illuminazione e robusto per distorsioni geometriche locali. I punti sono identificati come i punti estremanti del risultato della differenza di due gaussiane (DoG) applicate all'immagine a vari livelli di riscalamento, in modo da selezionare i punti che rimangono consistenti anche per diversi livelli

di ingrandimento. Il descrittore di un punto è poi generato utilizzando i pixel adiacenti, in modo da renderlo una caratteristica locale, indipendente dalla traslazione.

Per un'immagine di dimensioni 750×1500 presa da Google Street View e adeguatamente ritagliata escludendo le zone oltre una certa latitudine, SIFT genera circa 1500 punti notevoli; le corrispondenze tra due immagini consecutive sono tipicamente nell'ordine del centinaio; su immagini 1500×3000, a fronte di 5000 punti notevoli, abbiamo 500 corrispondenze tra immagini consecutive. È da notare che non tutte le corrispondenze sono esatte, dato che il matching tra punti notevoli può portare ad accoppiamenti errati. Nel caso delle immagini 1500×3000, ad esempio, gli accoppiamenti esatti sono nell'ordine dei 150.

Essendo SIFT un metodo ormai affermato, abbiamo usato VLFeat, una libreria opensource in C che contiene un'implementazione efficiente dell'algoritmo ed offre un'interfaccia per le funzioni Matlab. In particolare la funzione `[f,d] = vl_sift(Image)` calcola i punti caratteristici e i corrispondenti descrittori. VLFeat offre anche la funzione di matching `[matches,scores] = vl_ubcmatch(d1,d2)`, per poter trovare le corrispondenze tra punti notevoli di due immagini.

Nella tesi, l'algoritmo SIFT viene applicato all'immagine omnidirezionale in proiezione equirettangolare. Questa procedura, scelta per la sua semplicità, può presentare alcune criticità: il descrittore SIFT è infatti robusto, non invariante, per deformazioni prospettiche. In una proiezione equirettangolare la deformazione prospettica è costante lungo le righe dell'immagine, ma varia in base alla "latitudine" del punto. Questo fatto può pregiudicare il riconoscimento e l'accoppiamento di punti notevoli tra immagini consecutive. Dato che non è possibile proiettare una superficie sferica su un piano senza deformazioni, una possibile soluzione potrebbe essere di calcolare i punti SIFT su sezioni rettificate dell'immagine equirettangolare, come mostrato in Figura 2.5. È possibile infatti ottenere sezioni localmente non deformate di un'immagine equirettangolare, allo stesso modo in cui in Google Street View vengono fatte visualizzare all'utente. L'idea è di ricampionare l'immagine equirettangolare in modo da proiettare la superficie sferica sul piano tangente alla sfera e perpendicolare all'asse ottico.

2.3 Stima robusta con RANSAC

Errori di misurazione È inevitabile che i dati utilizzati per stimare la geometria epipolare contengano degli errori. Un primo tipo di errori è causato da imprecisioni nella localizzazione dei punti notevoli delle immagini: una diffe-

Figura 2.5: In alto, dettaglio di una proiezione equirettangolare della panoramica di Piazza del Popolo; in basso, la sua rettificazione. Si può notare come le linee rette tornino ad essere tali.

renza anche di un pixel può causare notevoli errori di ricostruzione, se il punto 3D $\mathbf{X}$ corrispondente è molto lontano, come tipicamente avviene. L'errore di posizionamento dei punti notevoli sul piano immagine viene amplificato dalla distanza, così che il raggio uscente dal centro della sfera e passante per il punto notevole $\mathbf{x}$ non passerà esattamente per il punto $\mathbf{X}$. Questo tipo di errori è inevitabile, e per quanto minimizzato dalla bontà dell'algoritmo di localizzazione dei punti notevoli, che arriva a precisioni di parecchio inferiori al pixel, è sempre presente. Sì può minimizzarne l'effetto utilizzando più punti possibile, risolvendo ai minimi quadrati il problema, in modo che gli errori tendano a compensarsi.

Errori da false corrispondenze Un secondo tipo di errori sono le corrispondenze errate (*false matches*), punti notevoli che l'algoritmo di matching riconosce come simili ma che in realtà appartengono a parti completamente diverse dell'immagine. Questo tipo di problematica è molto frequente quando nella scena sono presenti oggetti con texture ripetute, situazione che nell'ambito della fotografia urbana è inevitabile, ad esempio nel caso delle facciate delle case. Se le corrispondenze errate vengono utilizzate nel calcolo della geometria epipolare i risultati diventano inaffidabili, dato che

- le stime ai minimi quadrati vengono influenzate notevolmente dagli *outlier* e
- la percentuale di corrispondenze errate può essere molto alta, anche maggiore delle corrispondenze corrette.

Occorre quindi rimuovere le corrispondenze errate (*outliers*) in modo da effettuare il calcolo della geometria epipolare con le sole corrispondenze corrette (*inliers*). Il metodo ormai affermato è quello del *RANdom SAmple Consensus*, un metodo iterativo non deterministico per trovare il modello che descriva correttamente più dati possibile.

Algoritmo RANSAC Nel nostro caso il modello che stiamo cercando è la geometria epipolare tra le videocamere che prendiamo in considerazione, rappresentata dalla matrice essenziale o dal tensore trifocale, rispettivamente per due e tre viste. È necessario specificare una soglia t, errore oltre il quale un punto è considerato *outlier*. I passi sono:

1. Viene selezionato in modo casuale un sottoinsieme dei dati originali; tipicamente vengono scelti insiemi minimi, ovvero il numero minimo di dati che è necessario per stimare la geometria epipolare (con gli algoritmi usati, sono 8 punti per la matrice essenziale e 7 per il tensore trifocale),

in modo da aumentare la probabilità che il sottoinsieme scelto sia privo di *outlier*;

2. Viene stimata la geometria epipolare $\mathtt{M}$ utilizzando i punti del sottoinsieme scelto

3. Viene calcolato l'errore per ogni punto: nel nostro caso, dal modello vengono estratti $\mathtt{R}$ e $\mathbf{t}$ delle videocamere coinvolte, calcolati i punti $\mathbf{X}_i$, e riproiettati tali punti sulle sfere, ottenendo i punti $\hat{\mathbf{x}}_i$. L'errore di ogni punto è semplicemente l'errore di riproiezione (linearizzato, prendendo come misura la corda sulla sfera), ovvero $d_i = \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2$. Per essere considerato *inlier* un punto deve avere $d_i < t$, ovvero, dato il modello $\mathtt{M}$, il punto riproiettato $\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_i(\mathtt{M})$ deve essere sufficientemente vicino al punto misurato $\mathbf{x}_i$. Vengono contati gli *inlier*, ottenendo così un voto di consenso dei punti rispetto al modello $\mathtt{M}$. Se il modello corrente ha un punteggio più alto di quello del miglior modello finora stimato, viene salvato, insieme al suo punteggio e agli indici dei punti considerati *inlier*.

4. I passi 1, 2, 3 sono ripetuti finché non si raggiunge la desiderata probabilità di aver analizzato almeno un sottoinsieme di dati privo di *outlier*. È possibile stimare tale probabilità, che di solito viene posta al 99%, in base al numero di *inlier* dei modelli analizzati via via. Naturalmente il numero di iterazioni necessarie cresce rapidamente con l'aumentare della percentuale di *outlier* e con la dimensione del sottoinsieme.

5. Al termine delle iterazioni dell'algoritmo vengono considerati come dati affidabili gli *inlier* del miglior modello trovato, e viene eseguita una nuova stima della geometria epipolare utilizzando non più un sottoinsieme minimo ma tutti gli *inlier*.

Funzioni di misura dell'errore Il passo del punto 3 necessita di una funzione di calcolo di un errore. Oltre all'errore di riproiezione sopra proposto, sono stati usati altri due metodi, più semplici.

Nel caso a due viste è possibile usare la formula dell'errore di Sampson, che effettua una stima di primo ordine sull'errore. Per ogni coppia $\mathbf{x}$, $\mathbf{x}'$ la distanza è così calcolata:

$$d = \frac{(\mathbf{x}'\mathtt{E}\mathbf{x})^2}{\|\mathtt{E}\mathbf{x}\|^2 + \|\mathtt{E}^\top\mathbf{x}'\|^2}.$$

Nel caso a tre viste un metodo rapido è quello di provare la bontà delle corrispondenze tra i punti: dato che il residuo dall'equazione 1.8 dovrebbe

essere una matrice $\mathtt{0}_{3\times 3}$, poniamo

$$d = \left\| [\mathbf{x}']_\times \left(\sum_i x^i \mathtt{T}_i \right) [\mathbf{x}'']_\times \right\| \tag{2.1}$$

dove la norma $\|\cdot\|$ può essere una norma matriciale, ad esempio la norma 2:

$$\|\mathtt{A}\|_2 = \left(\sum_{i,j} |a_{ij}|^2 \right)^{\frac{1}{2}}.$$

Un metodo analogo di test sulla bontà delle corrispondenze può essere usato anche per il caso a due viste. In questo caso è sufficiente porre:

$$d = \left| \mathbf{x}'^\top \mathtt{E} \mathbf{x} \right|. \tag{2.2}$$

Dato che RANSAC effettua molti tentativi la scelta di un metodo rapido per calcolare la bontà di un modello è essenziale per avere buone prestazioni. La scelta della soglia di tolleranza di RANSAC va calibrata in base al tipo di problema e al tipo di funzione distanza scelta; nel caso dell'errore di riproiezione la scelta è stata di $t = 0.001$, dato che prende in considerazione due punti appartenenti a una sfera unitaria. Nei casi in cui è stato usato come errore il residuo delle equazioni 2.1 e 2.2 sono stati scelti valori di $t \in (0.001, 0.005)$.

Configurazioni degeneri Infine, è utile notare che nel caso di stima tradizionale di matrice essenziale o tensore trifocale con RANSAC viene introdotto anche un test, tra il passo 1 e il passo 2, per determinare se un insieme di punti è degenere. Se i sottoinsiemi minimi contengono punti disposti in particolari configurazioni, come punti allineati o complanari, il contenuto informativo è minore: le equazioni che vengono generate per calcolare la geometria epipolare non sono linearmente indipendenti, e portano alla stima di modelli errati ma che soddisfano un grande numero di *inlier*. Questo problema può essere evitato in due modi:

- Verificare che il sottoinsieme di punti scelto non sia degenere. La letteratura nel caso di videocamere omnidirezionali è assente, ma nelle prove effettuate insiemi di punti complanari o collineari causano stime errate. Il caso di punti complanari in particolare è critico in quanto è facile che avvenga in contesti urbani, dato che le facciate degli edifici hanno forme molto regolari.

- Utilizzare una funzione distanza che evidenzi errori di stima dovuti agli insiemi degeneri. La semplice norma delle corrispondenze tra punti non

è adeguata, in quanto viene calcolata usando le stesse equazioni che sono linearmente dipendenti. La misura dell'errore di riproiezione, invece, effettuando la triangolazione dei punti e verificando la consistenza dei risultati ottenuti, è un test adeguato. Un sottoinsieme minimo degenere infatti produce risultati completamente sbagliati, e anche le riproiezioni vengono influenzate: il modello viene quindi scartato perché è supportato da pochi *inlier*. La controindicazione di questo metodo è che i tempi di calcolo si allungano notevolmente, dato che questo tipo di funzione distanza è computazionalmente molto più oneroso che non la 2.1.

2.4 Stima della matrice essenziale

Nel caso di due viste la geometria epipolare per il caso omnidirezionale è descritta dalla matrice essenziale, come visto nel capitolo precedente, che è definita dall'equazione

$$\mathbf{x}'^{\top}\mathbf{E}\mathbf{x} = 0. \tag{2.3}$$

Avendo sufficenti coppie di punti corrispondenti nelle due viste è possibile calcolare la matrice essenziale. Esistono diversi metodi, elaborati per le videocamere tradizionali, i cui obiettivi sono principalmente due:

- ottenere una matrice fondamentale che soddisfi in modo ottimale le corrispondenze dell'equazione (2.3)
- ottenere una matrice fondamentale geometricamente consistente, ovvero che goda delle proprietà che teoricamente dovrebbe avere e che non sono automaticamente garantite dalla minimizzazione degli errori delle corrispondenze.

Questi due obiettivi possono essere raggiunti in passi separati: prima viene calcolata una matrice che minimizzi un errore geometrico, e poi vengono imposti i vincoli che rendono consistente la geometria epipolare - rango due nel caso della matrice fondamentale, due valori singolari uguali e il terzo nullo nel caso della matrice essenziale.

Mentre i metodi per videocamere tradizionali devono tenere in considerazione varie criticità, come il cattivo condizionamento dovuto ai grossi rapporti di scala che possono esserci tra i punti sul piano immagine e all'ambiguità proiettiva, nel caso omnidirezionale le cose sono più semplici. I punti sono disposti sulla sfera unitaria, e quindi già "normalizzati"; siamo già in ambito di ricostruzione metrica, un contesto in cui l'unica ambiguità è quella della scala; tutto il mondo circostante è visibile, e questo comporta che possono esserci molte corrispondenze "buone", ovvero che comportino un basso errore di

ricostruzione. Le migliori corrispondenze sono quelle in cui l'angolo $\widehat{\mathbf{CXC}'}$ è il più vicino possibile ai 90°, dato che in questo modo la triangolazione risente in modo minore degli errori di misurazione dei punti immagine.

Il metodo adottato in questa tesi è in due passi:

1. Calcolo di una stima della matrice essenziale con l'algoritmo lineare a 8 punti
2. Imposizione algebrica dei vincoli di consistenza geometrica

Passo 1: algoritmo a 8 punti Otto corrispondenze sono sufficienti per calcolare in modo lineare i 9 elementi della matrice essenziale: partendo dall'equazione (2.3) e scrivendo i punti $\mathbf{x} = (x_1, x_2, x_3)$ e $\mathbf{x}' = (x'_1, x'_2, x'_3)$, è possibile sviluppare algebricamente il prodotto vettore-matrice-vettore di una corrispondenza:

$$\begin{bmatrix} x'_1 & x'_2 & x'_3 \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$$e_{11}x'_1x_1+e_{12}x'_1x_2+e_{13}x'_1x_3+e_{21}x'_2x_1+e_{22}x'_2x_2+e_{23}x'_2x_3+e_{31}x'_3x_1+e_{32}x'_3x_2+e_{33}x'_3x_3 = 0$$

che può essere scritta sotto forma di prodotto scalare

$$\mathbf{y} \cdot \mathbf{e} = 0 \tag{2.4}$$

con

$$\mathbf{y} = \begin{bmatrix} x'_1x_1 & x'_1x_2 & x'_1x_3 & x'_2x_1 & x'_2x_2 & x'_2x_3 & x'_3x_1 & x'_3x_2 & x'_3x_3 \end{bmatrix} \tag{2.5}$$

$$\mathbf{e} = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{21} & e_{22} & e_{23} & e_{31} & e_{32} & e_{33} \end{bmatrix}$$

La (2.4) rappresenta un'equazione che lega le coordinate immagine della corrispondenza $\mathbf{x} \longleftrightarrow \mathbf{x}'$ al valore degli elementi della matrice essenziale. Avendo n coppie di punti corrispondenti, è possibile costruire il sistema lineare

$$\mathtt{A}\mathbf{e} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} \mathbf{e} = 0, \tag{2.6}$$

dove $\mathtt{A}$ è costruita accostando i vettori $\mathbf{y}_i$, $i = 1 \div n$ calcolati come la (2.5). Il problema (2.6) è una sistema lineare omogeneo, ed $\mathbf{e}$ può essere determinata a meno di un fattore di scala, motivo per cui bastano 8 equazioni. Perché esista

una soluzione è necessario che rank$\mathtt{A} \leq 8$, e se rank$\mathtt{A} = 8$ la soluzione è unica e può essere trovata linearmente, essendo il vettore che genera lo spazio nullo destro di $\mathtt{A}$.

Nel caso reale di corrispondenze soggette ad errore, avendo più di 8 equazioni si avrà che rank$\mathtt{A} = 9$, dato che la matrice ha 9 colonne. È possibile quindi trovare una soluzione ai minimi quadrati ricorrendo alla scomposizione ai valori singolari (SVD). In questo caso la soluzione sarà il vettore singolare corrispondente al valore singolare minimo della matrice $\mathtt{A}$, ovvero, detta

$$\mathtt{A} = \mathtt{U}\mathtt{S}\mathtt{V}^\top$$

la scomposizione SVD, la soluzione sarà l'ultima colonna di $\mathtt{V}$: $\mathbf{e} = \mathbf{v}_9$. La soluzione così trovata minimizza la quantità $\|\mathtt{A}\mathbf{e}\|$ sotto la condizione $\|\mathbf{e}\| = 1$.

Passo 2: imposizione dei vincoli di consistenza geometrica La matrice $\mathtt{E}$ ottenuta riorganizzando gli elementi di $\mathbf{e}$ non è geometricamente consistente, dato che in generale non avrà due valori singolari uguali e il terzo nullo richiesti dalla Proposizione 6. Imporre questi vincoli significa trovare la matrice $\mathtt{E}'$ che minimizzi la norma

$$\|\mathtt{E} - \mathtt{E}'\|_\infty = \max_i \sum_j \left| e_{ij} - e'_{ij} \right|$$

con il vincolo detto sui valori singolari.

È facile imporre tale vincolo ricorrendo sempre alla scomposizione ai valori singolari: si scompone

$$\mathtt{E} = \mathtt{U}\mathtt{D}\mathtt{V}^\top$$

dove $\mathtt{D}$ è la matrice diagonale dei valori singolari.

Si pone poi

$$\mathtt{E}' = \mathtt{U}\widetilde{\mathtt{D}}\mathtt{V}^\top$$

dove

$$\widetilde{\mathtt{D}} = \begin{bmatrix} a & & \\ & a & \\ & & 0 \end{bmatrix}.$$

Il valore di a è arbitrario, e fissa la scala della ricostruzione. Per semplicità abbiamo posto $a = 1$: in questo modo la *baseline* $\mathbf{t}$ avrà norma unitaria.

2.5 Stima del tensore trifocale

Nel caso di tre viste, utilizzando corrispondenze di punti o linee è possibile calcolare il tensore trifocale $\mathcal{T}$, che riassume la geometria epipolare. Avremo

ancora due passaggi, il primo di stima lineare e il secondo di imposizione algebrica dei vincoli.

Stima lineare di $\mathcal{T}$ Nel caso della corrispondenza tra punti, l'equazione è la (1.8):

$$[\mathbf{x}']_\times \left(\sum_i x^i \mathtt{T}_i \right) [\mathbf{x}'']_\times = \mathtt{0}_{3\times 3}. \tag{2.7}$$

L'equazione può essere scritta anche in notazione tensoriale come

$$x^i x'^j x''^k \epsilon_{jqs} \epsilon_{krt} \mathcal{T}_i^{qr} = 0_{st} \tag{2.8}$$

dove ϵ_{ijk} è il tensore di Ricci e 0_{st} è un tensore bidimensionale con elementi nulli. Una corrispondenza tra tre punti identifica quindi 9 equazioni, lineari negli elementi $\mathcal{T}_i^{jk}$, di cui è possibile mostrare che solo 4 sono linearmente indipendenti. Accostando un numero sufficiente di equazioni è possibile quindi calcolare linearmente il tensore trifocale, in modo analogo all'algoritmo a 8 punti per la matrice essenziale, scegliendo opportunamente le equazioni. La (2.8) può essere espansa come

$$x^k \left(x'^i x''^m T_k^{jl} - x'^j x''^m T_k^{il} - x'^i x''^l T_k^{jm} + x'^j x''^l T_k^{im} \right) = 0^{ijlm},$$

utilizzando la notazione tensoriale degli indici ripetuti, che riassume una sommatoria su quegli indici (in questo caso la somma è sui k). Una scelta delle 4 equazioni indipendenti può essere fatta fissando $j = m = 3$ e lasciando $i = 1, 2$, $l = 1,\ 2$:

$$\begin{array}{ll}
x^k \left(x'^1 x''^3 T_k^{31} - x'^3 x''^3 T_k^{11} - x'^1 x''^1 T_k^{33} + x'^3 x''^1 T_k^{13} \right) = 0 & i = l = 1 \\
x^k \left(x'^1 x''^3 T_k^{32} - x'^3 x''^3 T_k^{12} - x'^1 x''^2 T_k^{33} + x'^3 x''^2 T_k^{13} \right) = 0 & i = 1,\ l = 2 \\
x^k \left(x'^2 x''^3 T_k^{31} - x'^3 x''^3 T_k^{21} - x'^2 x''^1 T_k^{33} + x'^3 x''^1 T_k^{23} \right) = 0 & i = 2,\ l = 1 \\
x^k \left(x'^2 x''^3 T_k^{32} - x'^3 x''^3 T_k^{22} - x'^2 x''^2 T_k^{33} + x'^3 x''^2 T_k^{23} \right) = 0 & i = l = 2
\end{array}$$

Ogni equazione coinvolge 12 elementi del tensore trifocale; riorganizzando i 27 elementi in un vettore di incognite, è possibile scrivere il problema lineare come

$$\mathtt{A}\mathbf{t} = \mathbf{0}$$

con

$$\mathbf{t}^\top = \left[T_1^{11}, T_1^{12}, T_1^{13}, T_1^{21}, \ldots, T_3^{32}, T_3^{33} \right].$$

La matrice $\mathtt{A}$ è di dimensioni $4n \times 27$, dove n è il numero di triplette di punti corrispondenti. Per avere una soluzione sono sufficienti 7 triplette; tale

soluzione del sistema omogeneo si può trovare ancora ricorrendo alla SVD, come nel caso dell'algoritmo a 8 punti: scomposta

$$\mathtt{A} = \mathtt{U}\mathtt{S}\mathtt{V}^\top$$

la soluzione è

$$\mathbf{t} = \mathbf{v}_{27},$$

l'ultima colonna della matrice $\mathtt{V}$, i cui elementi riorganizzati compongono la stima lineare $\mathcal{T}_0$ del tensore trifocale.

Rispetto al caso tradizionale è stato omesso il passaggio di normalizzazione dei punti immagine, sempre grazie alla geometria sferica, in cui i punti sono già ben distribuiti, e non esistono problemi di scale diverse tra i punti $\mathbf{x}_i$.

Imposizione dei vincoli di consistenza geometrica La soluzione ottenuta linearmente non ha garanzie di consistenza geometrica, come nel caso della matrice essenziale. Occorre dunque imporre algebricamente la consistenza, in questo caso ricorrendo all'equazione usata per definire teoricamente il tensore trifocale:

$$\mathtt{T}_i = \mathbf{a}_i \mathbf{b}_4^\top - \mathbf{a}_4 \mathbf{b}_i^\top,$$

in notazione tensoriale

$$\mathcal{T}_i^{jk} = a_i^j e''^k - e'^j b_i^k \tag{2.9}$$

dove a_j^i è l'elemento della riga i (indice controvariante) colonna j (indice covariante) di $\mathtt{A}$.

Ricordando che $\mathbf{a}_4 = \mathbf{e}'$ e $\mathbf{b}_4 = \mathbf{e}''$ sono le ultime colonne delle matrici $\mathtt{A}$ e $\mathtt{B}$, abbiamo un legame tra gli epipoli e il tensore trifocale. Questo legame consta di 27 equazioni, una per ogni elemento del tensore; riorganizzando gli elementi in vettori si può scrivere

$$\mathtt{E}\mathbf{a} = \mathbf{t} \tag{2.10}$$

dove

$$\mathbf{a}^\top = \left[a_1^1, a_2^1, a_3^1, a_1^2, \cdots, a_3^3, b_1^1, \cdots, b_3^3\right],$$

$$\mathbf{t}^\top = \left[T_1^{11}, T_1^{12}, T_1^{13}, T_1^{21}, \ldots, T_3^{32}, T_3^{33}\right]$$

ed $\mathtt{E}$ è la matrice 27×18 che esprime i legami dell'equazione (2.9), ed è costruita a partire dagli elementi di $\mathbf{e}'$ ed $\mathbf{e}''$. Conoscendo gli epipoli è possibile quindi costruire un tensore trifocale che sia geometricamente consistente e minimizzi l'errore algebrico $\|\mathtt{A}\mathbf{t}\|$. È infatti sufficiente cercare la soluzione che

minimizza $\|\mathtt{A}\mathbf{t}\|$ soggetta al vincolo $\|\mathbf{t}\| = 1$. Utilizzando la (2.10), il problema di minimizzazione vincolata diventa

$$\begin{array}{c} \min_{\mathbf{a}} \|\mathtt{AE}\mathbf{a}\| \\ \text{s.t. } \|\mathtt{E}\mathbf{a}\| = 1 \end{array} \tag{2.11}$$

Una volta ottenuta $\mathbf{a}$, si calcola il tensore trifocale sempre con la (2.10).

Calcolo degli epipoli Per effettuare la minimizzazione vincolata (2.11) è necessario dunque riuscire ad estrarre gli epipoli a partire dalla stima $\mathcal{T}_0$. Dalle proprietà del tensore, viste a pagina a pagina 18, sappiamo che gli epipoli $\mathbf{e}'$ ed $\mathbf{e}''$ sono le perpendicolari comuni ai vettori nulli sinistri (rispettivamente destri) delle tre matrici $\mathtt{T}_i$. In presenza di rumore è necessario risolvere il sistema nel senso dei minimi quadrati, dato che non esiste una soluzione esatta. Si applica la decomposizione ai valori singolari:

1. Per $i = 1, 2, 3$ si calcola mediante SVD il vettore unitario $\mathbf{v}_i$ che minimizza $\|\mathtt{T}_i\mathbf{v}_i\|$, dove $\mathtt{T}_i = \mathcal{T}_i^{\cdot\cdot}$. Si forma poi la matrice

$$\mathtt{V} = \begin{bmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \mathbf{v}_3^\top \end{bmatrix}$$

2. Si calcola l'epipolo $\mathbf{e}''$ come il vettore unitario che minimizza $\|\mathtt{V}\mathbf{e}''\|$, ancora mediante SVD.

L'epipolo $\mathbf{e}'$ viene calcolato in modo analogo, usando $\mathtt{T}_i^\top$ invece delle $\mathtt{T}_i$.

Ottimizzazione con Levenberg-Marquardt Il metodo di minimizzazione che calcola il tensore trifocale a partire dagli epipoli, $(\mathbf{e}', \mathbf{e}'') \mapsto \mathtt{AE}\mathbf{t}$ può essere visto come una mappa $\mathbb{R}^6 \to \mathbb{R}^{27}$. È possibile applicare un metodo iterativo, come Levenberg-Marquardt, per ottenere un risultato ottimale; il passo di imposizione dei vincoli appena visto equivale alla prima iterazione del metodo (per dettagli vedi [16, 8]), ovvero a partire da una stima iniziale degli epipoli viene fatto un passo di ottimizzazione. Applicando l'algoritmo LM fino a convergenza è possibile migliorare la stima lineare. Gli ingredienti dell'algoritmo sono

- La variabile da ottimizzare, che nel nostro caso è il vettore di $\mathbb{R}^6$ composto dal concatenamento degli epipoli $\mathbf{e} = \begin{bmatrix} \mathbf{e}' \\ \mathbf{e}'' \end{bmatrix}$, con il valore iniziale $\mathbf{e}_0$ ottenuto dal tensore trifocale stimato linearmente $\mathcal{T}_0$.

- La funzione obiettivo $\mathbf{g}(\mathbf{e}) \in \mathbb{R}^{27}$, che è composta dai seguenti passaggi:
  - calcolo della matrice $\mathtt{E}$ utilizzando gli elementi di $\mathbf{e}$
  - minimizzazione vincolata (2.11) che restituisce la soluzione $\mathbf{t}$
  - calcolo del residuo $\mathbf{g} = \mathtt{A}\mathbf{t}$
- Un numero massimo di iterazioni e una tolleranza sull'errore relativo sotto la quale l'algoritmo viene fermato.

2.6 Calcolo di R, t e il problema della scala

Dato che il nostro scopo è di stimare il moto della videocamera, è necessario estrarre le informazioni di rototraslazione tra le varie viste.

A livello di notazione, indicheremo con $\mathbf{t}$ e $\mathbf{t}_{21}$, $\mathbf{t}_{31}$ i vettori di traslazione che vengono calcolati rispettivamente dalla matrice essenziale e dal tensore trifocale, e che sono espressi nel sistema di riferimento della sfera corrente, la k-esima. Nel caso delle tre viste indicheremo con $\mathbf{t}_{21}^k$ il vettore di traslazione finale, espresso nel sistema di riferimento globale. Dato che (rumore a parte) $\mathbf{t}_{21}^k = \mathbf{t}_{31}^{k-1}$ essendo gli stessi punti, esprimeremo infine la posizione della sfera k-esima con $\mathbf{t}^k$, sia nel caso a due che per tre viste.

Ricostruzione con due viste

Estrazione di R, t Tutte le informazioni di geometria epipolare sono contenute nella matrice essenziale $\mathtt{E}$. Dalla teoria sappiamo che è possibile ricostruire la posizione delle videocamere a meno di una rototraslazione (da un ipotetico sistema di riferimento globale) e di un fattore di scala. Possiamo quindi senza perdita di generalità far coincidere la prima videocamera con il sistema di riferimento globale e calcolare $\mathtt{R}$ e $\mathbf{t}$ fra la prima e la seconda videocamera.

Dalla teoria sappiamo che per le videocamere tradizionali è possibile calcolare la matrice della videocamera $\mathtt{P}' = [\mathtt{R} \mid \mathbf{t}]$ sfruttando il risultato (1.6):

$$\mathtt{P}' = \left[\mathtt{U}\mathtt{W}\mathtt{V}^\top \mid +\mathbf{u}_3\right] \text{ o } \left[\mathtt{U}\mathtt{W}\mathtt{V}^\top \mid -\mathbf{u}_3\right] \text{ o } \left[\mathtt{U}\mathtt{W}^\top\mathtt{V}^\top \mid +\mathbf{u}_3\right] \text{ o } \left[\mathtt{U}\mathtt{W}^\top\mathtt{V}^\top \mid -\mathbf{u}_3\right].$$

Nel caso delle videocamere omnidirezionali non ha senso parlare di matrice di proiezione della videocamera, ma si dimostra che i parametri estrinseci sono ancora determinabili usando l'espressione (1.6). La differenza è che la proiezione sul piano immagine non avviene attraverso la moltiplicazione $\mathbf{x} = \mathtt{P}'\mathbf{X}$ ma attraverso la normalizzazione delle coordinate dei punti nello spazio espressi nel sistema di riferimento locale, identificato da $\mathtt{R}$ e $\mathbf{t}$.

Avendo quattro possibili soluzioni è necessario selezionare quella corretta. Per far questo, detti $\mathbf{C}$ e $\mathbf{C}'$ i centri delle videocamere, e $\mathbf{x}$ e $\mathbf{x}'$ i punti immagine, è sufficente ricostruire il punto 3D $\mathbf{X}$ ($\mathbf{X}$' se espresso nel sistema di riferimento della seconda sfera) mediante triangolazione, e controllare che le triplette $\mathbf{C}, \mathbf{x}, \mathbf{X}$ e $\mathbf{C}', \mathbf{x}', \mathbf{X}'$ siano allineate in quest'ordine. Come vedremo dalla triangolazione, i punti delle triplette non saranno perfettamente allineati, ma è sufficiente controllare che il prodotto scalare tra i vettori $\mathbf{x} - \mathbf{C}$ e $\mathbf{X} - \mathbf{x}$ sia positivo. Nella seconda vista si applica lo stesso procedimento, avendo cura prima di effettuare il cambio di coordinate $\mathbf{X}' = \mathtt{R}(\mathbf{X} - \mathbf{t})$ e di porre $\mathbf{C}' = \mathbf{0}$.

Sistema di riferimento globale Avendo una serie di fotografie omnidirezionali I_k prese dalla stessa videocamera ed analizzate a coppie, diventa necessario risolvere due problematiche. In primo luogo, il calcolo del sistema di riferimento: visto che ogni coppia di viste è considerata separatamente nei calcoli, il metodo proposto riesce a calcolare solo rotazione e traslazione relative tra le due viste, dimenticandosi delle informazioni precedenti. È comunque sufficiente utilizzare $\mathtt{R}$ e $\mathbf{t}$ relative di ogni camera per calcolare via via gli estrinseci cumulativi per la videocamera $k+1$-esima:

$$\mathtt{R}^{k+1} = \mathtt{R}\mathtt{R}^{k}$$

$$\mathbf{t}^{k+1} = \mathbf{t}^{k} + \left(\mathtt{R}^{k}\right)^{-1}\mathbf{t}$$

Propagazione della scala Un secondo problema è quello della scala: ogni volta che vengono calcolati $\mathtt{R}$ e $\mathbf{t}$, l'algoritmo calcola una *baseline* di norma unitaria. È necessario quindi propagare l'informazione di scala scelta per la prima coppia di videocamere (che può essere fissata arbitrariamente oppure mediante qualche tipo di dato esterno) anche alle altre viste: serve un elemento che faccia da ponte tra una coppia di viste e quella successiva. L'unico tipo di informazione di questo tipo sono i punti 3D ricostruiti visibili in almeno tre immagini consecutive.

Dette $\mathtt{P}$, $\mathtt{P}'$ e $\mathtt{P}''$ le tre videocamere, chiamiamo $\mathbf{X}_i$ i punti che sono visibili dalle prime due viste, $\mathbf{Y}_i$ i punti visibili nelle seconde due. Intersecando i due insiemi di punti possiamo trovare i punti $\mathbf{Z}_i$ comuni alle tre viste. Nel caso siano stati utilizzati descrittori SIFT è sufficiente cercare le corrispondenze tra la prima e la seconda, e tra la seconda e la terza vista, ed effettuare un *join* dei due insiemi usando come chiave i descrittori della seconda vista. L'operazione di *join* consiste nel selezionare tutti i punti che compaiono in entrambi gli insiemi e hanno lo stesso descrittore nella seconda vista. In Matlab può essere usato il comando `intersect`. I punti $\mathbf{Z}_i$ devono quindi essere ricostruiti in modo identico a partire dalle due coppie di videocamere, assumendo di aver

tenuto traccia degli estrinseci cumulativi, a parte il fattore di scala mancante. Se posizioniamo il nostro sistema di riferimento sulla seconda sfera, questo significa che i punti ricostruiti $\hat{\mathbf{Z}}_i$ saranno semplicemente riscalati lineramente rispetto agli $\mathbf{Z}_i$. Recuperare il fattore di scala a questo punto è semplice: calcolando i baricentri

$$\overline{\mathbf{Z}}_{old} = \frac{1}{N} \sum_i \mathbf{Z}_i$$

e

$$\overline{\mathbf{Z}}_{new} = \frac{1}{N} \sum_i \hat{\mathbf{Z}}_i,$$

dobbiamo ottenere due punti allineati con il centro della seconda videocamera $\mathbf{C}'$, di cui si può calcolare il rapporto delle distanze rispetto al centro della videocamera, e quindi la scala

$$k = \frac{\left\| \overline{\mathbf{Z}}_{old} - \mathbf{C}' \right\|}{\left\| \overline{\mathbf{Z}}_{new} - \mathbf{C}' \right\|}.$$

La vera *baseline* stimata sarà dunque $\tilde{\mathbf{t}} = k\mathbf{t}$, essendo $\mathbf{t}$ la *baseline* stimata dall'algoritmo a 8 punti.

Il metodo proposto per il calcolo della scala presenta diverse criticità, essendo sensibile alla presenza di *outlier*, che supponiamo essere stati rimossi mediante il RANSAC utilizzato per la stima della matrice essenziale. Potrebbero verificarsi inoltre casi di baricentri coincidenti con il centro della videocamera. Nell'articolo [19], gli autori hanno proposto un metodo più robusto basato sulla scelta della scala che minimizza la distanza tra i punti ricostruiti nella prima e nella seconda coppia di viste, e seleziona con RANSAC gli *inlier* (il metodo è descritto in [11]). Avremmo quindi un doppio RANSAC annidato: all'esterno un ciclo per trovare gli *inlier* della geometria epipolare tra due viste, e all'interno un ciclo per trovare gli *inlier* comuni alle tre viste.

Ricostruzione con tre viste

Estrazione delle matrici essenziali Nel caso di tre viste abbiamo calcolato il tensore trifocale; da questo è possibile estrarre la geometria epipolare fra coppie di videocamere e quindi le rototraslazioni relative tra la prima e la seconda vista, e tra la prima e la terza.

Dalla teoria sappiamo che le matrici fondamentali vengono calcolate a partire dal tensore trifocale utilizzando gli epipoli:

$$\mathtt{F}_{21} = [\mathbf{e}']_\times \left[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3\right] \mathbf{e}'' \qquad \text{e} \qquad \mathtt{F}_{31} = [\mathbf{e}'']_\times \left[\mathtt{T}_1^\top, \mathtt{T}_2^\top, \mathtt{T}_3^\top\right] \mathbf{e}'.$$

Essendo i nostri punti normalizzati ci troviamo in realtà già nel caso metrico, per cui le matrici risultanti saranno essenziali, e le chiameremo $\mathtt{E}_{21}$ ed $\mathtt{E}_{31}$.

Calcolo degli estrinseci Da queste matrici essenziali possiamo ancora estrarre le quantità $\mathtt{R}_{21}$, $\mathbf{t}_{21}$ e $\mathtt{R}_{31}$, $\mathbf{t}_{31}$ ricorrendo al metodo proposto per le due viste. Rimane però un problema: con il metodo standard, avremmo $\|\mathbf{t}_{31}\| = \|\mathbf{t}_{21}\| = 1$, cosa che in generale è falsa. Occorre quindi riuscire ad assegnare alle due *baseline* le corrette proporzioni. In realtà quest'informazione è già contenuta nelle matrici essenziali, dato che sono calcolate dallo stesso tensore trifocale e quindi hanno lo stesso fattore di scala. I valori delle *baseline* non normalizzate che potremmo ottenere sono quindi arbitrari, ma utili in quanto rispettano le corrette proporzioni tra loro. Per estrarre quest'informazione è sufficiente salvare il fattore di scala di ognuna delle due matrici essenziali, che coincide con la coppia di valori singolari non nulli ottenuti dalla decomposizione SVD:

$$\mathtt{E} = \mathtt{U} \begin{bmatrix} s_1 & & \\ & s_2 & \\ & & 0 \end{bmatrix} \mathtt{V}^\top$$

I due valori singolari s_1 ed s_2 dovrebbero essere identici, ma a causa del rumore potrebbero essere diversi: per maggior robustezza prendiamo il fattore di scala della prima coppia di videocamere come

$$s' = \frac{s_1 + s_2}{2}.$$

Una volta calcolato questo valore possiamo procedere come prima, fissando quindi la scala della prima coppia a 1 mediante l'imposizione di valori singolari unitari:

$$\mathtt{S}' = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix}.$$

Analogamente possiamo calcolare il fattore di scala della seconda coppia s''. Il fattore di scala tra le due coppie sarà allora

$$s = \frac{s'}{s''}$$

e imporremo la scala della seconda coppia come

$$\mathtt{S}'' = \begin{bmatrix} s & & \\ & s & \\ & & 0 \end{bmatrix}.$$

In questo modo date tre viste e le relative corrispondenze, l'algoritmo calcola le matrici $\mathtt{R}_{21}$ e $\mathtt{R}_{31}$ che esprimono le rotazioni relative tra la prima e la seconda e la prima e la terza vista, e i vettori di traslazione $\mathbf{t}_{21}$ e $\mathbf{t}_{31}$ che puntano in direzione delle rispettive sfere; $\mathbf{t}_{21}$ avrà norma unitaria mentre $\mathbf{t}_{31}$ sarà scalato in proporzione.

Propagazione della scala Propagare le informazioni della scala da un terzetto di viste all'altro è poi banale: al passo k-esimo l'algoritmo restituisce una $\mathbf{t}_{21}$ di norma unitaria, ma sappiamo che $\mathbf{t}_{21}^{k} \equiv \mathbf{t}_{31}^{k-1}$ di cui conosciamo la misura calcolata al passo precedente. È sufficente quindi riscalare le *baseline* trovate dall'algoritmo usando quest'informazione:

$$\tilde{\mathbf{t}}_{21} = \left\| \mathbf{t}_{31}^{k-1} - \mathbf{t}_{21}^{k-1} \right\| \mathbf{t}_{21}$$

$$\tilde{\mathbf{t}}_{31} = \left\| \mathbf{t}_{31}^{k-1} - \mathbf{t}_{21}^{k-1} \right\| \mathbf{t}_{31}$$

Sovrapposizione tra i terzetti di sfere È utile fare un'osservazione sulla sovrapposizione dei terzetti di sfere usate per i calcoli, visto che è possibile scegliere tra due opzioni: una sovrapposizione minima, e una abbondante. Nel primo caso analizzeremo le sfere 1, 2 e 3, poi le sfere 3, 4 e 5, e via così. Un procedimento di questo tipo è orientato a rendere più rapidi i calcoli, dato che si avanza di due sfere ogni volta; rimane altresì il problema della propagazione dell'informazione della scala, che può essere risolto in modo analogo a quanto descritto nel caso delle due viste: calcolo della scala a partire dai punti in comune a due terzetti consecutivi di sfere. Tale metodo è però difficilmente praticabile, almeno per fotografie sparse, in quanto non è semplice trovare punti che siano visibili da tutte e 5 le sfere.

Nel secondo caso, che è quello portato avanti nella tesi, vengono analizzate le sfere 1, 2 e 3, poi le 2, 3 e 4, e così via. Stiamo in pratica sovrapponendo ampiamente i terzetti di sfere in modo da poter guadagnare in robustezza della stima, e ridurre così i problemi di deriva.

2.7 Triangolazione dei punti

La ricostruzione dei punti 3D viene usata all'interno degli algoritmi proposti in diversi casi, ad esempio per scegliere tra le 4 possibili coppie di $\mathtt{R}$ e $\mathbf{t}$ quella corretta, o per propagare l'informazione di scala nel caso a due viste. L'idea di base della triangolazione è di propagare i raggi passanti per i centri di videocamera e i punti sul piano immagine: avendo almeno due viste tali raggi dovrebbero incontrarsi nel punto dello spazio che ha generato i punti immagine.

Il condizionale è d'obbligo, in quanto nel mondo reale è inevitabile avere errori che pregiudicano questa proprietà.

Sono stati sviluppati diversi metodi di triangolazione, che mirano a minimizzare quantità come l'errore di riproiezione, o l'errore algebrico (per maggiori dettagli vedi [8]). Una delle proprietà richieste a questi metodi è di essere invarianti per trasformazioni proiettive, dato che l'ambito comune in cui vengono usati è quello di ricostruzione proiettiva.

Nel nostro caso siamo invece già in ambito metrico, motivo per cui è possibile permettersi una semplificazione di questo passaggio. È stato scelto un semplice metodo geometrico basato sulla proiezione dei raggi generati dai punti immagine. Conoscendo gli estrinseci delle due sfere e i punti immagine sulla superficie delle sfere stesse, è possibile calcolare le equazioni delle rette dei raggi, $\mathbf{r}$ e $\mathbf{r}'$. Utilizzando formule di geometria cartesiana è inoltre possibile calcolare il punto $\mathbf{A}_0$ della retta $\mathbf{r}$ in cui la distanza dalla retta $\mathbf{r}'$ è minimo, e l'analogo punto $\mathbf{B}_0$ sulla retta $\mathbf{r}'$. Una volta calcolati i punti $\mathbf{A}_0$ e $\mathbf{B}_0$, una buona approssimazione del punto da ricostruire è il punto medio $\hat{\mathbf{X}} = \frac{1}{2}(\mathbf{A}_0 + \mathbf{B}_0)$. Il punto $\hat{\mathbf{X}}$ è anche quello che distribuisce l'errore di riproiezione equamente su entrambe le sfere coinvolte, in quanto i punti riproiettati $\hat{\mathbf{x}}$ e $\hat{\mathbf{x}}'$ sono i punti medi degli archi aventi come estremi le proiezioni dei punti $\mathbf{A}_0$ e $\mathbf{B}_0$.

Dati i punti $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{B}_1$ e $\mathbf{B}_2$, rispettivamente centro e punto immagine della prima videocamera, centro e punto immagine della seconda videocamera, espressi nello stesso sistema di riferimento globale, avremo dunque

$$n_a = [(\mathbf{B}_2 - \mathbf{B}_1) \times (\mathbf{A}_1 - \mathbf{B}_1)] \cdot [(\mathbf{A}_2 - \mathbf{A}_1) \times (\mathbf{B}_2 - \mathbf{B}_1)]$$

$$n_b = [(\mathbf{A}_2 - \mathbf{A}_1) \times (\mathbf{A}_1 - \mathbf{B}_1)] \cdot [(\mathbf{A}_2 - \mathbf{A}_1) \times (\mathbf{B}_2 - \mathbf{B}_1)]$$

$$d = [(\mathbf{A}_2 - \mathbf{A}_1) \times (\mathbf{B}_2 - \mathbf{B}_1)] \cdot [(\mathbf{A}_2 - \mathbf{A}_1) \times (\mathbf{B}_2 - \mathbf{B}_1)]$$

$$\mathbf{A}_0 = \mathbf{A}_1 + \frac{n_a}{d}(\mathbf{A}_2 - \mathbf{A}_1)$$

$$\mathbf{B}_0 = \mathbf{B}_1 + \frac{n_b}{d}(\mathbf{B}_2 - \mathbf{B}_1)$$

$$\hat{\mathbf{X}} = \frac{\mathbf{A}_0 + \mathbf{B}_0}{2}$$

Nel caso delle tre viste effettuiamo la triangolazione utilizzando coppie di videocamere, ad esempio la prima con la seconda e la prima con la terza, e calcoliamo $\hat{\mathbf{X}}$ come la media tra i due punti $\hat{\mathbf{X}}_{12}$ e $\hat{\mathbf{X}}_{13}$ ottenuti.

Algoritmo 2.1 Stima del moto della videocamera utilizzando due viste.

Obiettivi

Date N immagini omnidirezionali in formato equirettangolare, calcolare posizione e orientamento della videocamera nel momento dello scatto.

Algoritmo

Per $k = 2 \div N$

1. Calcolo dei punti notevoli $\mathbf{x}_i$ con SIFT nelle immagini $k-1$ e k.
2. Matching tra i punti notevoli trovati per trovare le corrispondenze $\mathbf{x}_i^{k-1} \leftrightarrow \mathbf{x}_i^k$, espresse in coordinate sferiche.
3. Stima con RANSAC della matrice essenziale $\mathtt{E}$ e degli *inlier* $\mathbf{x}_j^{k-1}$ e $\mathbf{x}_j^k$, utilizzando l'algoritmo a 8 punti per calcolare la matrice a partire dalle corrispondenze.
4. Calcolo di $\mathtt{R}$, $\mathbf{t}$ a partire dalla matrice essenziale mediante scomposizione SVD. La *baseline* ha norma unitaria.
5. Se $k = 2$
   (a) $\mathtt{R}^k = \mathtt{R}$, $\mathbf{t}^k = \mathbf{t}$
   (b) Triangolazione dei punti immagine per ottenere i punti 3D $\mathbf{X}^k$
6. Se $k > 2$
   (a) Triangolazione dei punti immagine per ottenere i punti 3D $\hat{\mathbf{X}}^k$
   (b) Selezione mediante matching dei punti immagine che sono tra loro corrispondenti nelle immagini $k-2$, $k-1$, k.
   (c) Calcolo dei baricentri $\bar{\mathbf{X}}^k$ e $\bar{\mathbf{X}}^{k-1}$ dei punti selezionati e stima della scala $k = \frac{\left\|\bar{\mathbf{X}}^k - \mathbf{t}^{k-1}\right\|}{\left\|\bar{\mathbf{X}}^{k-1} - \mathbf{t}^{k-1}\right\|}$.
   (d) Riscalamento di $\mathbf{t}$ e dei punti 3D per ottenere gli $\mathbf{X}^k$
   (e) $\mathtt{R}^k = \mathtt{R}\mathtt{R}^k$, $\mathbf{t}^k = \mathbf{t}^k + \left(\mathtt{R}^k\right)^{-1}\mathbf{t}$

Algoritmo 2.2 Stima del moto della videocamera utilizzando tre viste.

Obiettivi

Date N immagini omnidirezionali in formato equirettangolare, calcolare posizione e orientamento della videocamera nel momento dello scatto.

Algoritmo

Per $k = 1 \div N - 2$

1. Calcolo dei punti notevoli $\mathbf{x}_i$ con SIFT delle immagini k, $k+1$ e $k+2$.
2. Matching tra i punti notevoli trovati per trovare le corrispondenze $\mathbf{x}_i^k \leftrightarrow \mathbf{x}_i^{k+1}$ e $\mathbf{x}_i^{k+1} \leftrightarrow \mathbf{x}_i^{k+2}$, espresse in coordinate in coordinate sferiche. Join dei due insiemi per trovare le corrispondenze tra le tre immagini.
3. Stima con RANSAC del tensore trifocale $\mathcal{T}$ e degli *inlier* $\mathbf{x}_j^k$, $\mathbf{x}_j^{k+1}$ e $\mathbf{x}_j^{k+2}$:
   (a) Calcolo della stima $\mathcal{T}_0$ mediante stima ai minimi quadrati a partire dalle corrispodenze tra punti
   (b) Iterazioni di Levenberg-Marquardt sul metodo di ottimizzazione vincolata, inizializzato con $\mathcal{T}_0$
4. Calcolo di $\mathtt{E}_{21}$ ed $\mathtt{E}_{31}$ a partire dal tensore trifocale e calcolo del fattore di scala s tra la seconda e la terza videocamera.
5. Calcolo di $\mathtt{R}_{21}$, $\mathbf{t}_{21}$ $\mathtt{R}_{31}$, $\mathbf{t}_{31}$ a partire dalle matrici essenziali. La *baseline* $\mathbf{t}_{21}$ ha norma unitaria, mentre $\mathbf{t}_{31}$ è scalata del fattore s.
6. Se $k = 1$
   (a) $\mathtt{R}^2 = \mathtt{R}_{21}$, $\mathbf{t}^2 = \mathbf{t}_{21}$, $\mathtt{R}^3 = \mathtt{R}_{31}$, $\mathbf{t}^3 = \mathbf{t}_{31}$.
   (b) Triangolazione dei punti immagine per ottenere i punti 3D $\mathbf{X}^k$
7. Se $k > 1$
   (a) Riscalamento di $\mathbf{t}_{21}$e $\mathbf{t}_{31}$ moltiplicandoli per il fattore $k = \left\|\mathbf{t}^{k+1} - \mathbf{t}^k\right\|$.
   (b) Triangolazione dei punti immagine per ottenere i punti 3D $\mathbf{X}^k$
   (c) $\mathtt{R}^{k+2} = \mathtt{R}_{31}\mathtt{R}^{k+1}$, $\mathbf{t}^{k+2} = \mathbf{t}^{k+1} + \left(\mathtt{R}^{k+1}\right)^{-1}\mathbf{t}_{31}$

Capitolo 3

Risultati sperimentali

3.1 Risultati con dati sintetici

Per mettere alla prova i metodi sviluppati sono stati generati dati sintetici in Matlab, in modo che fosse possibile misurare con precisione gli errori nella localizzazione delle sfere. È importante misurare qual è l'errore di stima per ogni coppia o terzetto di sfere considerata, ma anche quantificare la deriva totale, ovvero come gli errori si accumulano quando vengono analizzate molte sfere, dato che l'obiettivo del metodo è di stimare il moto a partire da una serie di immagini.

Nello spazio 3D sono stati generati N punti casuali $\mathbf{X}_i$, con $N = 50$ o $N = 100$. Con uno script vengono poi generate le sfere, con distanze e angoli di rotazione casuali. Ogni sfera è identificata dalla coppia $(\mathtt{R}_{real}, \mathbf{t}_{real})$, che esprime rotazione e traslazione rispetto al sistema di riferimento globale. Su ogni sfera vengono poi generati i punti immagine $\mathbf{x}_i$ proiettando i punti $\mathbf{X}_i$. La proiezione avviene in due passi: prima i punti vengono convertiti nel sistema di riferimento della sfera stessa, in seguito vengono normalizzati in modo che abbiano norma unitaria, e giacciano quindi sulla sfera di raggio unitario che identifica la videocamera omnidirezionale. I punti immagine sono salvati in vettori di dimensioni $3 \times N$, per cui la colonna i-esima dei vettori di ogni sfera è la proiezione dell'i-esiamo punto 3D.

Per simulare il rumore, vengono introdotti due tipi di errori dopo la proiezione:

- Rumore gaussiano nella proiezione dei punti sulla sfera, che simula un errore con deviazione standard di σ pixel nella localizzazione del punto notevole. I test sono stati effettuati con $\sigma = 0.3$ e $\sigma = 3$ pixel. Utilizzando metodi come SIFT si può ottenere una precisione inferiore al

pixel, necessaria per una buona ricostruzione, ma nel caso più realistico l'errore più superare il pixel di ampiezza.

- False corrispondenze, con una percentuale intorno al 30% rispetto al totale.

Caso ideale

Il caso senza rumore è un modo semplice per testare il funzionamento dell'algoritmo in condizioni ottimali, che in realtà non si verificano mai. Ci aspettiamo una stima perfetta del moto della videocamera, e gli errori ottenuti sono infatti vicini a quelli numerici, sia nel caso a due viste che in quello a tre.

Per le due viste i risultati sono mostrati in Figura 3.1: con un percorso lungo 10 sfere l'errore totale di localizzazione, preso come la somma degli errori delle *baseline:*

$$e_{tot} = \sum_{k=2}^{N} \left\| \mathbf{t}_{est}^{k} - \mathbf{t}_{real}^{k} \right\|$$

risulta $e_{tot} = 7.810^{-13}$, mentre per tre viste $e_{tot} = 1.110^{-12}$. Stiamo parlando in entrambi i casi di errori trascurabili, imputabili sostanzialmente alle operazioni effettuate con l'aritmetica imperfetta del calcolatore, per cui è normale che, essendo il tensore trifocale calcolato con un procedimento più complesso, risulti lievemente più alto l'errore. La potenza del tensore trifocale si mostra nel caso con rumore, dove le stime sono migliori.

Caso con rumore

Per affrontare casi più realistici, i dati sintetici vengono contaminati con gli errori prima presentati, in modo via via più pesante. Le figure mostrano i risultati sempre nel caso di un percorso di 10 sfere generate in modo casuale.

Solo errori di misurazione Iniziamo con l'introdurre errori nella localizzazione dei punti notevoli: alle coordinate di tutti i punti immagine viene sommato un errore gaussiano con deviazione standard di 0.3 pixel. In Figura 3.2 sono visibili i risultati per $N = 50$ corrispondenze: i casi a due e tre viste sono difficilmente distinguibili, e infatti presentano rispettivamente $e_{tot} = 0.07$ ed $e_{tot} = 0.06$. Gli errori introdotti vengono ben contrastati dalla stima ai minimi quadrati, che si rivela efficace anche con un numero di punti non molto alto. Aumentando il numero di corrispondenze, migliora automaticamente la stima, anche quando si aumenta l'intensità degli errori.

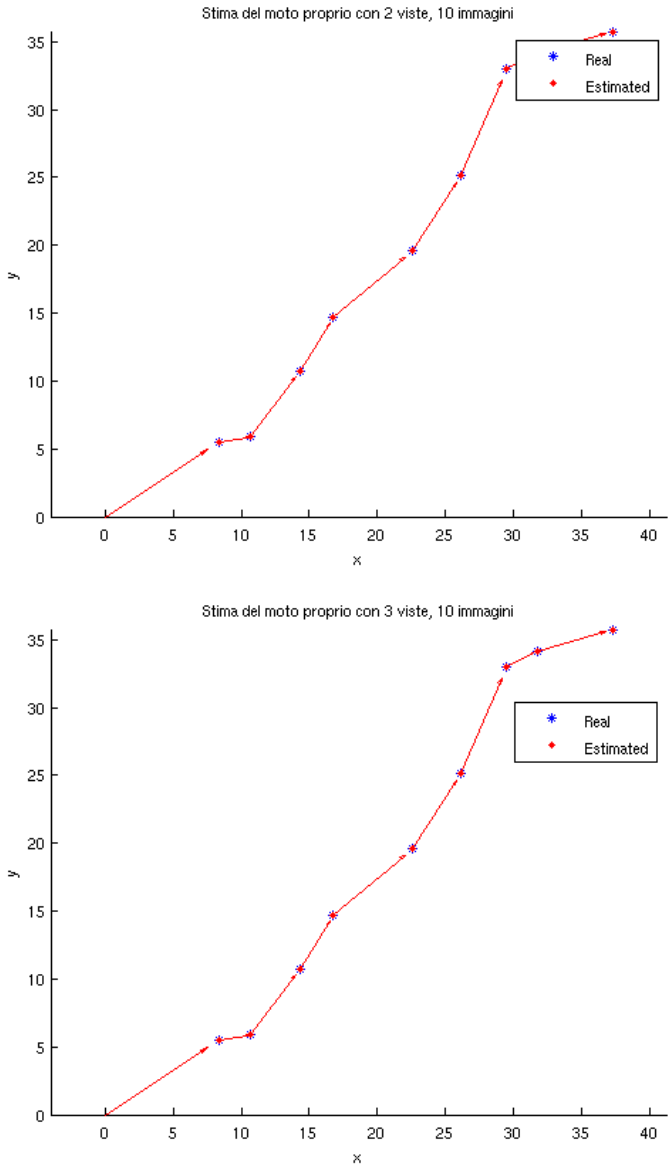

Figura 3.1: Stima del moto proprio con due e tre viste, nel caso ideale.

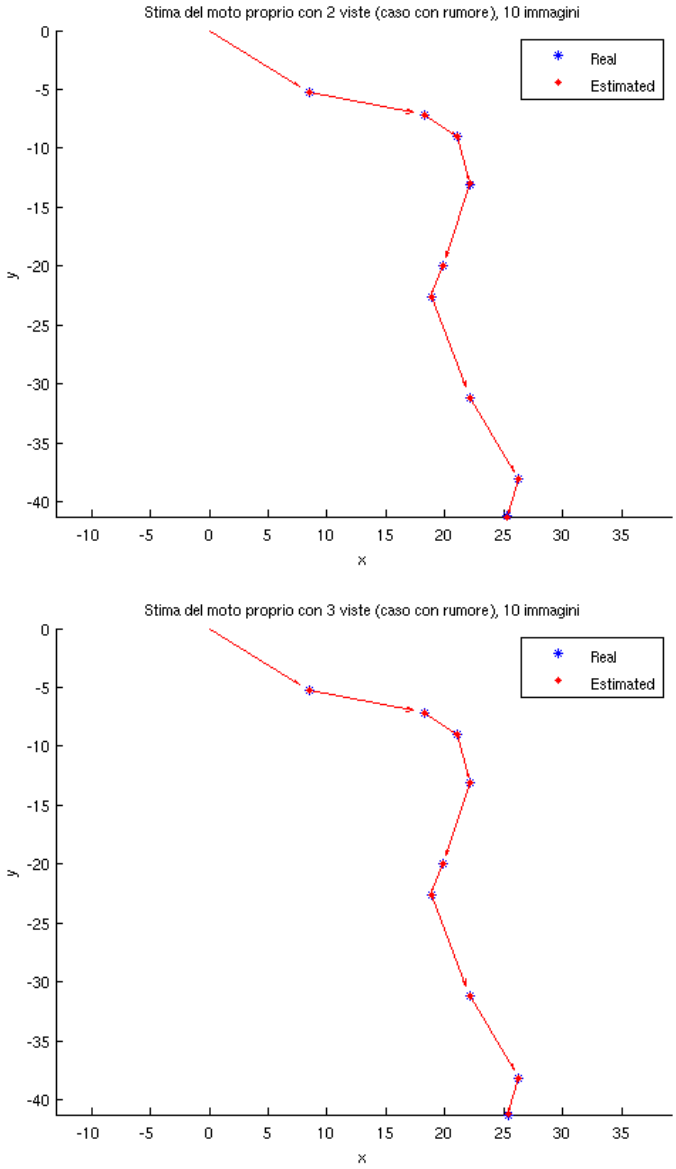

Figura 3.2: Stima del moto proprio con due e tre viste, nel caso in cui sia presente del rumore di localizzazione dei punti notevoli, ma nessuna falsa corrispondenza.

Con bassi errori di misurazione e false corrispondenze Introducendo le false corrispondenze il problema diventa più difficile. L'algoritmo RANSAC è infatti in grado di rimuovere tutte le false corrispondenze quando i dati che sono davvero *inlier* sono perfettamente accurati. Se invece anche gli *inlier* presentano errori, si genera un effetto a catena: i modelli calcolati da RANSAC usando insiemi minimi sono significativamente influenzati dagli errori, per cui risulta difficile separare nettamente *inlier* ed *outlier*. Se una falsa corrispondenza viene inserita tra quelle buone, il risultato finale viene notevolmente danneggiato.

Abbiamo iniziato per questo inserendo, su $N = 50$, un 30% di false corrispondenze e un errore di misurazione molto basso, nell'ordine di 0.03 pixel. In questo caso i risultati, mostrati in Figura 3.4, evidenziano come entrambi i metodi, a due e tre viste, si comportino bene. In particolare per le due viste $e_{tot} = 0.3$ ed $e_{tot} = 0.009$: il secondo metodo inizia ad essere decisamente più affidabile del primo, anche se entrambi i risultati si possono considerare accettabili, essendo l'errore minore dello 0.5% sul percorso totale.

Con errori di misurazione e false corrispondenze Aumentando l'intensità dell'errore di misurazione a 0.3 pixel appaiono evidenti i limiti del metodo con due viste, come mostrato in Figura 3.5 con $N = 50$. In questo caso $e_{tot} = 5.7$, ovvero il 9% del percorso totale compiuto dalla videocamera. Con tre viste invece abbiamo $e_{tot} = 0.56$, cioè lo 0.9% del percorso totale, un ordine di grandezza in meno. La differenza è dovuta alla robustezza del RANSAC: nel caso delle due viste può succedere che alcuni dati *outlier* non vengano scartati, e compromettano la stima. Essere *inlier* nel caso delle tre viste è infatti un requisito più stringente (tre vincoli invece di due) e i dati vengono filtrati meglio.

Aumentando il numero di punti totali, con $N = 200$, otteniamo un miglioramento dei risultati in entrambi i casi, come mostrato in Figura 3.6: per le due viste abbiamo $e_{tot} = 1.9$, il 3% in termini relativi, mentre per tre viste abbiamo $e_{tot} = 0.1$, lo 0.2% relativo. La stima ai minimi quadrati riesce a bilanciare gli errori migliorando il risultato.

Aumentando ancora l'intensità dell'errore e il numero di punti utilizzati, con $N = 400$ e $\sigma = 3$ pixel di errore, notiamo che sia il metodo a due che tre sfere si comportano bene, come mostrato in Figura 3.3: il grande numero di punti utilizzabili per i calcoli permette di scartare accuratamente gli *outlier* e di compensare il forte rumore con l'uso dei minimi quadrati. Il prezzo che si paga è un aumento notevole del numero di iterazioni RANSAC necessarie, che crescono sia con il numero di *outlier* che con il numero totale di punti. Nel caso della stima a due viste il metodo è comunque abbastanza rapido, mentre nel

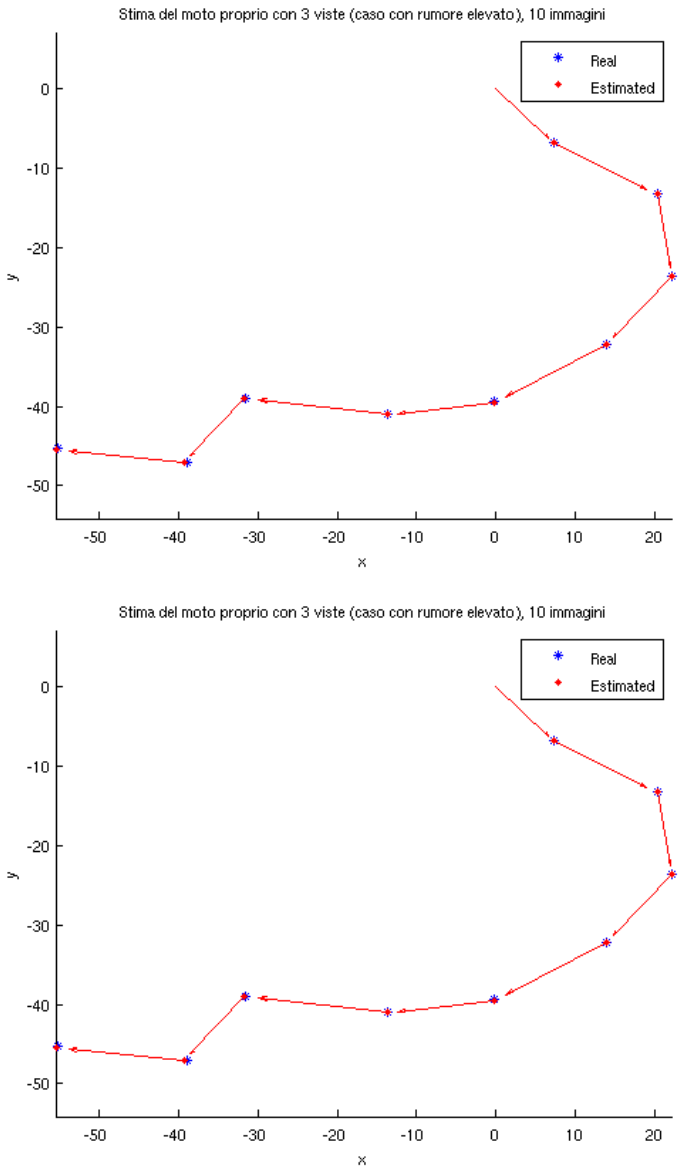

Figura 3.3: Stima del moto proprio con due e tre viste, nel caso in cui sia presente un forte rumore di localizzazione dei punti notevoli e false corrispondenze, ma il numero di punti sia elevato.

caso a tre viste i tempi di calcolo si allungano. A livello di accuratezza della ricostruzione, con due viste abbiamo $e_{tot} = 5.9$ equivalente al 4.8% sul percorso, mentre con tre viste abbiamo $e_{tot} = 1.6$ equivalente al 1.3% sul percorso.

La criticità maggiore risiede comunque nel trovare un metodo in grado di filtrare tutti gli *outlier*, e nel caso delle tre viste, che impongono un triplo vincolo, questo si traduce in una maggiore robustezza e precisione. Per capire l'effetto che può avere l'inclusione di un *outlier* nella stima del modello basta vedere la Figura 3.7: per una coppia di sfere la stima è stata falsata, e il risultato è un errore di deriva che viene propagato anche alle sfere successive, che vengono stimate correttamente, come si vede dal fatto che il percorso stimato e quello reale sono paralleli.

In sintesi, se è noto un numero sufficientemente grande di corrispondenze buone ($N > 100$) entrambi i metodi funzionano bene, riuscendo a rimuovere gli *outlier* e a compensare gli errori di misurazione. Se i punti utilizzati sono invece pochi il più robusto è il metodo a tre viste, mentre quello a due viste, pur più veloce, risente degli errori e presenta una deriva maggiore.

Prestazioni e velocità Per quanto riguarda la stima dei modelli con i dati sintetici, la parte più onerosa dal punto di vista computazionale è l'applicazione di RANSAC per trovare gli *inlier*, che è fortemente influenzata dalla percentuale di dati corretti sul totale. Nei casi visti l'algoritmo ha dovuto provare anche diverse centinaia di modelli, sopratutto nel caso a due viste, dove l'insieme minimo è di 8 elementi, per cui la probabilità di ottenere un insieme privo di *outlier* è bassa, nel nostro caso

$$p = (0.7)^8 = 0.0576,$$

senza contare gli elementi che potrebbero essere *inlier* ma con errori di misura così grandi da essere assimilabili ad *outlier*. Con tre viste l'insieme minimo utilizzato è di 7 elementi, che porta a $p = 0.0824$, un leggero miglioramento, ma esistono anche metodi che utilizzano solo 6 elementi, rinunciando ad una conoscenza completa di alcuni elementi del tensore ([8] pag. 511).

3.2 Risultati con dati sperimentali

I dati sperimentali sono sequenze di fotografie di Google Street View, si cui abbiamo la localizzazione con una certa approssimazione, dato che Google non fornisce le coordinate GPS di ogni bolla. Di fronte alla richiesta di posizionarsi nella coordinata (x, y), Google Street View fornisce l'immagine omnidirezionale più vicina alla coordinata. L'approssimazione può quindi essere di una decina

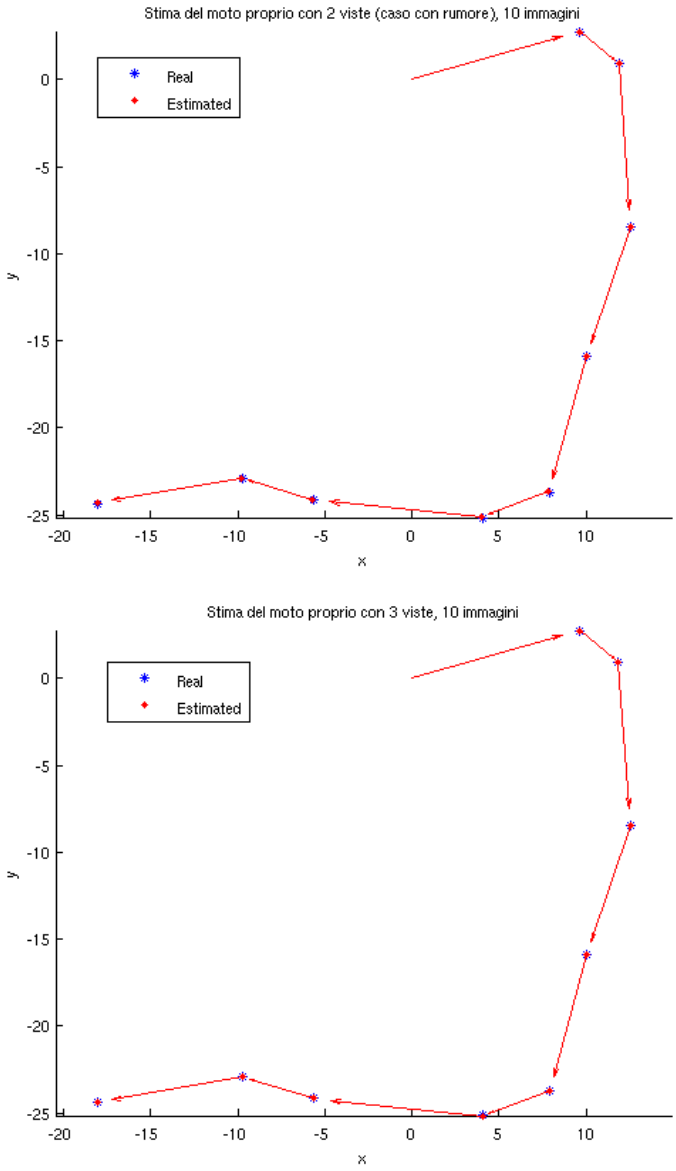

Figura 3.4: Stima del moto proprio con due e tre viste, nel caso in cui sia presente del rumore a bassa intensità e false corrispondenze.

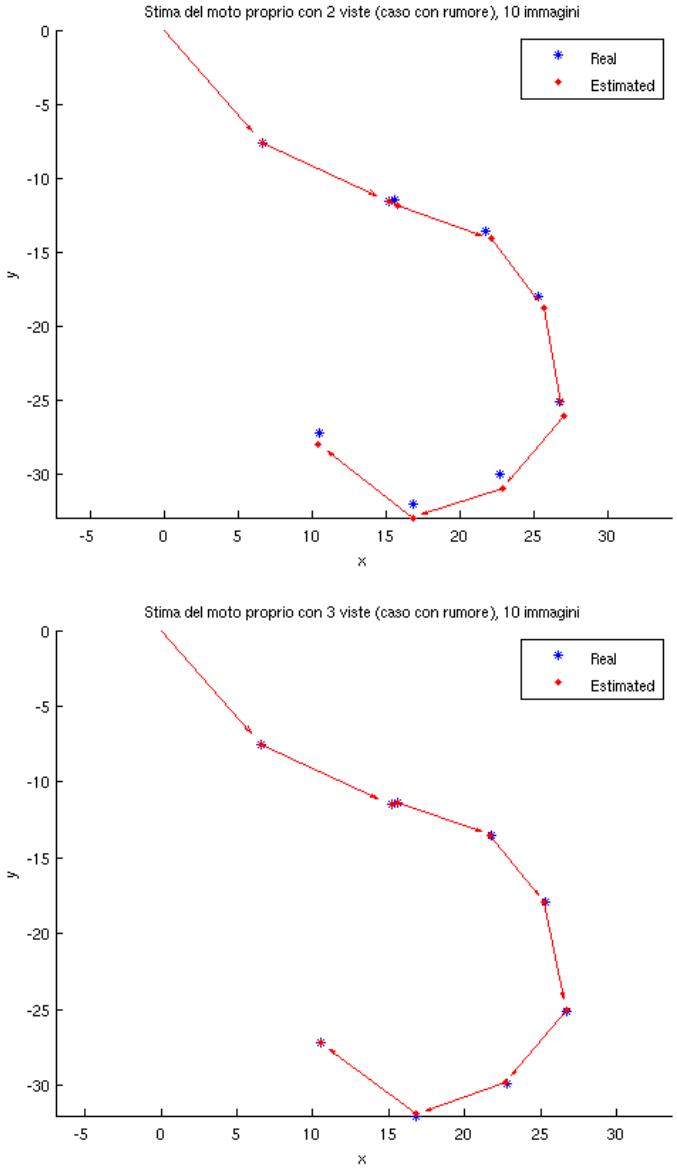

Figura 3.5: Stima del moto proprio con due e tre viste, nel caso in cui sia presente del rumore a bassa intensità.

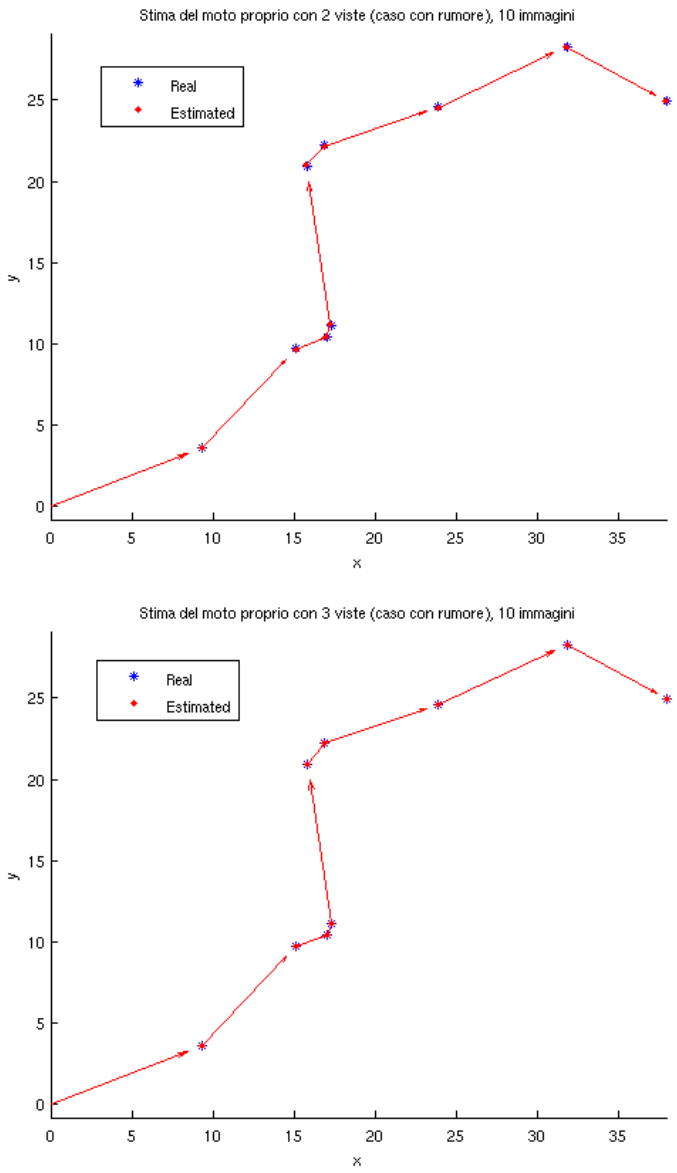

Figura 3.6: Stima del moto proprio con due e tre viste, nel caso in cui sia presente del rumore e false corrispondenze, ma il numero di corrispondenze sia elevato.

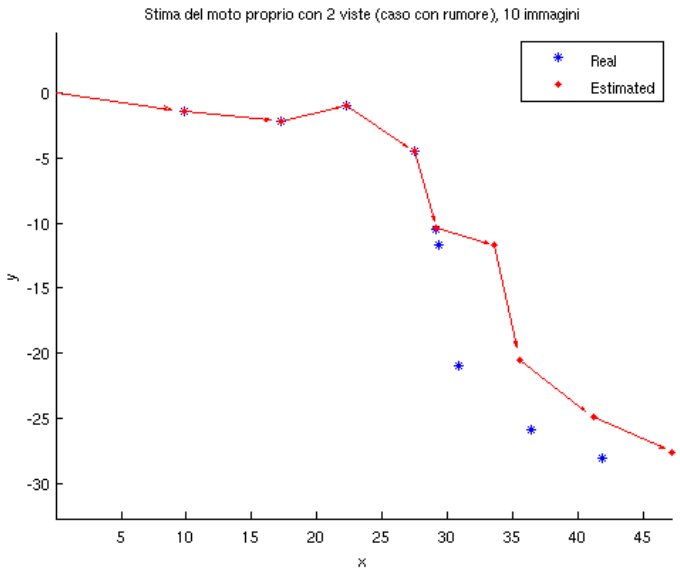

Figura 3.7: Un caso in cui RANSAC non è riuscito a rimuovere un *outlier* consistente, che ha quindi provocato un errore molto grande nella ricostruzione di una sfera.

di metri, su un percorso di un centinaio di metri. Questa incertezza viene poi amplificata su tutto il percorso ricostruito, dato che utilizziamo le coordinate GPS per fissare la scala iniziale.

Inoltre, come evidenziato nel capitolo precedente, non c'è garanzia che le immagini di Street View siano prese dalla stessa videocamera, dato che nel sito possono essere mischiate immagini prese in giorni diversi, e anche in sensi di marcia diversi. Queste ipotesi sono essenziali per il corretto funzionamento del metodo, per cui è stato necessario cercare zone in cui fossimo ragionevolmente sicuri che fossero soddisfatte.

I test sulle immagini sperimentali vanno quindi considerati come prove qualitative, di cui non possiamo misurare rigorosamente l'errore. Verificheremo quindi che la forma del percorso, e grosso modo la scala, rispecchino la realtà.

Test su rettilineo in Via Sforza a Lodi Un primo esempio riguarda il caso di un rettilineo, nella fattispecie Via Sforza a Lodi. Sono state utilizzate 6 viste omnidirezionali, di cui tre sono mostrate in Figura 3.8. La risoluzione delle fotografie è di 833×1666 pixel, generate a partire dalla vista di Google Street View nelle coordinate GPS mostrate in tabella 3.1. Come si può vedere dalla tabella, immagini con questa risoluzione generano circa 1500 punti notevoli

#	Latitudine	Longitudine	# punti SIFT	# match 3 viste	# *inlier* 3 viste
1	45.311486	9.490736	1510	71	32
2	45.311426	9.490739	1480	102	32
3	45.311356	9.490752	1612	55	32
4	45.311177	9.490768	1691	63	35
5	45.311233	9.490765	1683	-	-
6	45.311096	9.490779	1560	-	-

Tabella 3.1: Dati relativi alle sei viste utilizzate per stimare il moto della videocamera nel test Via Sforza.

nella sola zona centrale dell'immagine (ricordiamo che stiamo scartando il 70% dell'immagine, e conservando solo il 30% vicino all'equatore).

Di questi punti, solo 200 circa vengono stimati come corrispondenti tra un'immagine e quella consecutiva. Quando poi cerchiamo le corrispondenze tra tre immagini consecutive, i match si riducono a circa 70, come mostrato in Figura 3.9. A questi va sommato l'effetto di RANSAC, che cerca di eliminare gli *outlier*, ovvero corrispondenze false: tipicamente sono considerati *inlier* circa il 50% dei match fin qui trovati, cosa che riduce parecchio il numero di punti utilizzabili per stimare la geometria epipolare, come visibile in Figura 3.10, e aumenta di molto il tempo che RANSAC usa per avere la confidenza richesta di aver analizzato almeno un sottoinsieme minimo privo di *outlier*. Il fatto che RANSAC non sia deterministico è evidente dal fatto che ripetendo più volte la stima con gli stessi dati otteniamo un numero ogni volta diverso di *inlier*, anche se la variazione è di poche unità.

Il numero di iterazioni RANSAC necessarie dipende fortemente dalla soglia di errore impostata per discriminare *inlier* e *outlier*: nel nostro caso, con $e = 0.001$ sono necessarie circa 2000 iterazioni per ogni terzetto di sfere; con $e = 0.01$ sono circa 200. Con un $e = 0.002$ i risultati sono accettabili.

Una considerazione da fare sui dati è che il numero di corrispondenze *inlier* è piuttosto basso e non garantisce una stima affidabile della geometria epipolare: come visto con i dati sintetici, l'ideale è avere diverse centinaia di corrispondenze, in modo da bilanciare gli errori di misurazione. Un modo semplice per ottenere un miglioramento è di usare immagini a risoluzione maggiore, ma nel caso di Google Street View non è possibile, come evidenziato nella Sezione 2.1 a pagina 31. Ad esempio, con un'immagine 1500×3000 la situazione cambia: avremo più di 5000 punti notevoli dopo il SIFT, circa

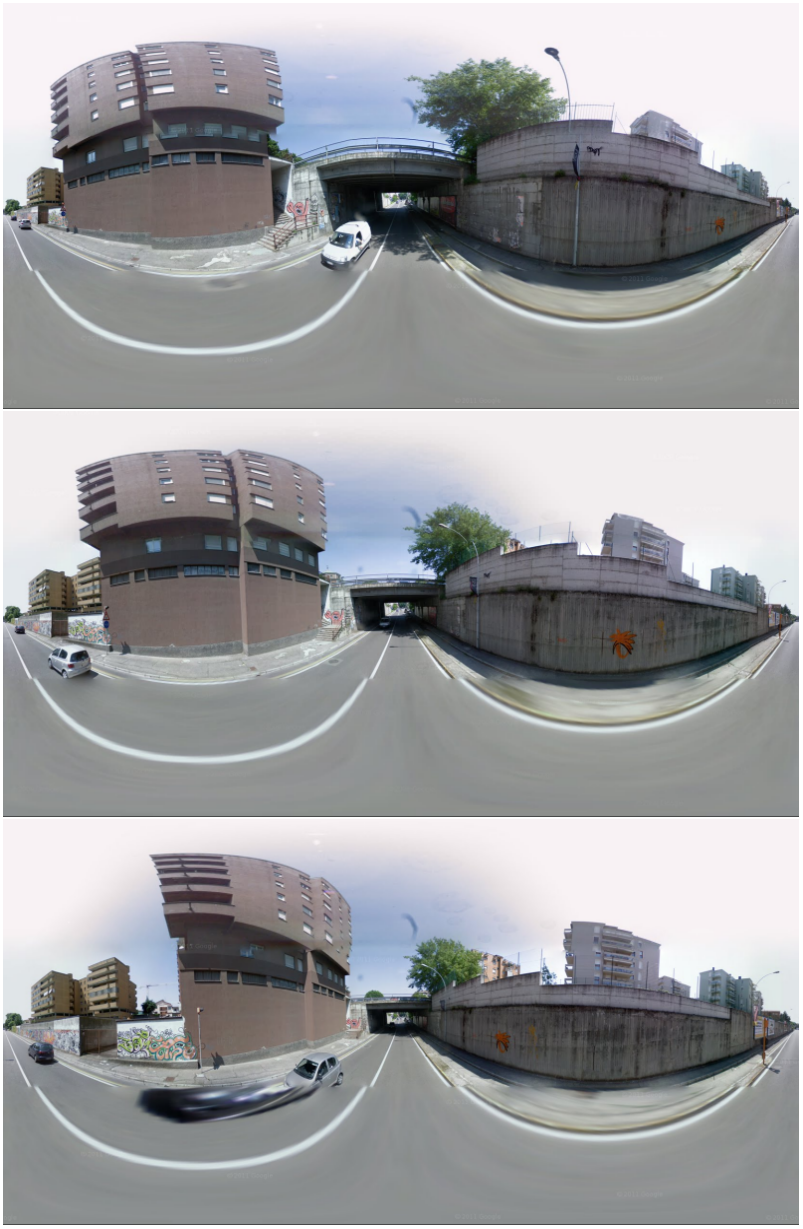

Figura 3.8: Esempio di tre viste omnidirezionali consecutive del test di stima Via Sforza.

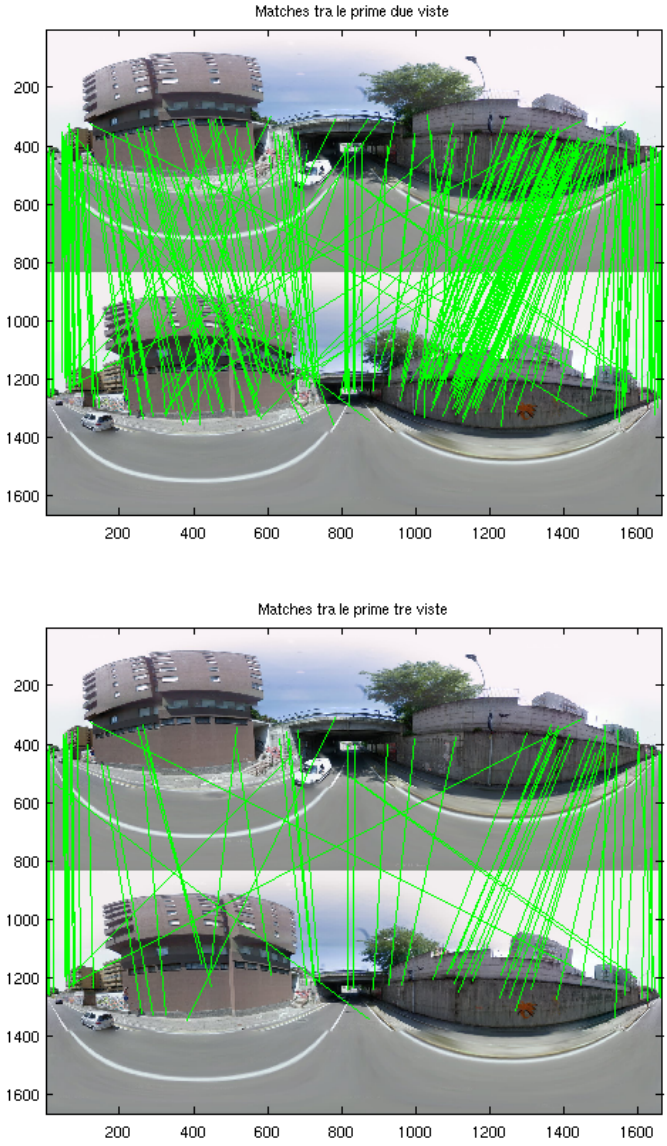

Figura 3.9: In figura sono mostrate la corrispondenze tra punti SIFT delle prime due immagini. In alto abbiamo le corrispondenze basate sulla distanza tra descrittori; in basso, le corrispondenze rimaste dopo l'intersezione con quelle tra la seconda e terza immagine.

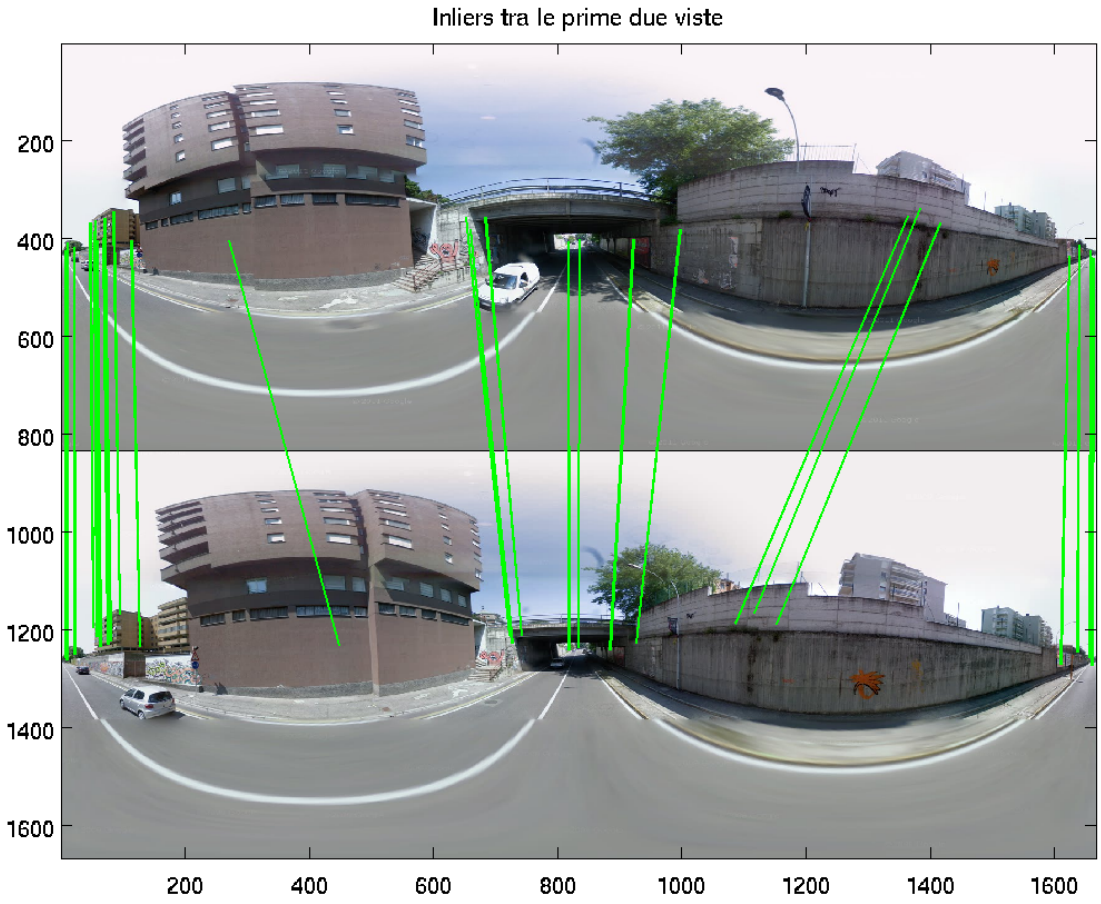

Figura 3.10: In figura sono mostrate la corrispondenze tra punti SIFT delle prime due immagini, dopo l'applicazione di RANSAC per il calcolo del tensore trifocale.

#	# punti SIFT	# match 3 viste	# *inlier* 3 viste
1	1800	134	67
2	1721	140	60
3	1707	141	60
4	1681	110	48
5	1740	-	-
6	1597	-	-

Tabella 3.2: Dati relativi alle sei viste utilizzate per stimare il moto della videocamera nel test Via Sforza.

700 corrispondenze tra due immagini consecutive, 300 corrispondenze in tre immagini, di cui un centinaio di *inlier*.

Con i dati utilizzati, la stima del moto che otteniamo dal metodo è buona, come si può vedere in Figura 3.11, sia come direzione che come scala, presentando un errore del 5% circa sulla strada percorsa in 6 viste. Per fissare la scala della ricostruzione è stata utilizzata la distanza tra le coordinate GPS dei punti utilizzati in Google Maps, per cui il risultato è poco affidabile, dato che non sono note con precisione le coordinate delle viste.

Test su curva in Piazza Castello a Torino Un secondo test è stato pensato per mettere alla prova il metodo in situazioni di curve. Il problema è più difficile che nel caso dei rettilinei, perché la scena può cambiare notevolmente tra un fotogramma e l'altro, cosa che riduce molto il numero di corrispondenze su cui si può far conto. Nel caso delle immagini di Google Street View, la sparsità delle fotografie è un fattore fortemente limitante: dato che la distanza tipica tra due bolle è nell'ordine dei 14 metri, è facile che da una vista all'altra l'auto abbia già girato l'angolo. Con le immagini della risoluzione disponibile, le corrispondenze tra due immagini consecutive sono nell'ordine del centinaio; dopo il RANSAC è facile che rimangano meno di 20 *inlier*, cosa che pregiudica una stima affidabile.

Per superare questi problemi è stata scelta una zona urbana con ampi spazi, in modo da bilanciare la sparsità delle immagini. Effettuare la stima del moto in una zona con ampi spazi e fotogrammi sparsi è simile ad effettuarla in spazi ristretti con molti fotogrammi. In Piazza Castello a Torino è presente una sequenza di fotogrammi che effettua una curva, come mostrato in Figura 3.12.

Utilizzando sei fotogrammi disposti lungo la curva è stata effettuata la stima del moto di Figura 3.12. In questo caso le corrispondenze tra i terzetti di viste, indicate per completezza in Tabella 3.2, sono circa 140, di cui 50-60

Figura 3.11: In alto: foto da satellite di Via Sforza, dove la freccia verde indica il punto di inizio della misurazione; in basso il percorso ricostruito, a meno di una rototraslazione.

inlier, cosa che permette una ricostruzione adeguata. È stata scelta una soglia di errore per RANSAC di $e = 0.002$, che ha significato circa 2000 iterazioni per la stima robusta del tensore trifocale; come si nota dalla Figura 3.13, gli *outlier* vengono filtrati efficacemente. I risultati sono soddisfacenti: la forma della curva è ben rispettata, e la ricostruzione della scala è pienamente compatibile con quella della mappa, con un errore cumulativo del 5% sul percorso compiuto.

Figura 3.12: In alto: foto da satellite di Piazza Castello a Torino, dove la freccia verde indica il punto di inizio della misurazione, che procede verso il basso seguendo la curva; in basso il percorso ricostruito, a meno di una rototraslazione.

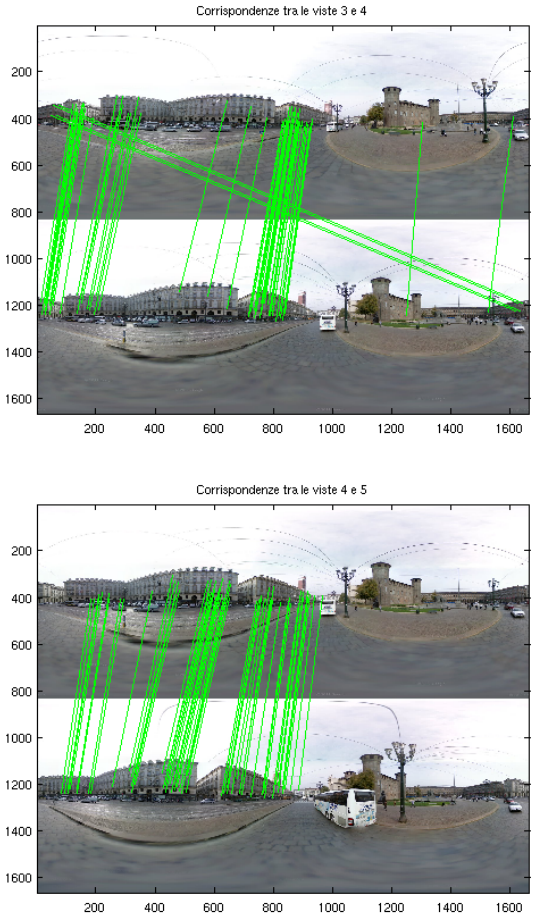

Figura 3.13: In figura sono mostrate la corrispondenze che sono considerate *inlier* tra le immagini 3 e 4, e tra la 4 e la 5. Nell'immagine in alto viene evidenziato il fatto che le immagini sono omnidirezionali: alcune corrispondenze sono a cavallo del bordo dell'immagine, per cui sono lontane in termini di pixel immagine, ma vicine in coordinate sferiche.

Conclusioni

Risultati

La tesi, inserita nell'ambito del progetto europeo Astute, ha avuto come scopo lo studio e la realizzazione di metodi per la stima del moto proprio a partire da sequenze di videocamere omnidirezionali.

Il lavoro di ricerca teorica ha evidenziato come nel campo delle videocamere omnidirezionali vi sia ancora spazio per una rigorosa sistematizzazione dei metodi, e che le potenzialità di questi strumenti di acquisizione digitale nel campo della Structure from Motion siano ancora da sfruttare a fondo, in particolare per quanto riguarda l'utilizzo di tre viste. L'applicazione del tensore trifocale alla ricostruzione 3D omnidirezionale è ancora agli inizi, e il metodo che è stato proposto in questa tesi va principalmente a confrontarsi con i metodi per due viste che sono stati finora utilizzati nelle pubblicazioni.

Nella tesi sono stati quindi implementati due metodi, rispettivamente utilizzando coppie e terzetti di viste.

È stato proposto un primo metodo che utilizza coppie di viste. Dalle immagini omnidirezionali estrae i punti notevoli mediante SIFT, calcola le corrispondenze tra i punti notevoli, ed effettua una stima robusta mediante RANSAC della matrice essenziale mediante una stima lineare con successiva imposizione dei vincoli algebrici. Da questa calcola la rototraslazione esistente tra le due videocamere, e stima il fattore di scala utilizzando i punti ricostruiti precedentemente come riferimento. I risultati ottenuti con i dati sintetici mostrano una buona robustezza generale, anche se l'errore di deriva aumenta sensibilmente nel caso in cui siano presenti errori di misurazione nell'ordine di qualche pixel. Il limite più grande del metodo risiede tuttavia nella propagazione della scala, problema che necessita l'implementazione di un ulteriore metodo di selezione robusta dei punti comuni a tre viste consecutive. Avendo dati sperimentali molto disturbati non è stato quindi possibile applicarvi questo metodo con profitto.

Un secondo metodo, con tre viste, estrae i punti notevoli con SIFT e calcola le corrispondenze nelle tre immagini. Effettua poi una stima robusta con RAN-

SAC del tensore trifocale, effettuando iterazioni di tipo Levenberg-Marquard sul metodo compsto da una stima lineare a cui si impone il rispetto dei vincoli algebrici. Dal tensore trifocale vengono estratte le matrici essenziali delle coppie di videocamere, e da queste vengono calcolate le rototraslazioni relative. La propagazione della scala al passo corrente viene effettuata sfruttando il fatto che la *baseline* tra la prima e la seconda videocamera del terzetto è nota, in quanto è stata calcolata al passo precedente come *baseline* tra la seconda e la terza. I risultati ottenuti con i dati sintetici mostrano una buona robustezza del metodo anche con dati di bassa qualità, sia per errori di misurazione che per false corrispondenze. L'applicazione ai dati sperimentali ha incontrato grossi limiti nella sparsità e nella bassa risoluzione delle immagini disponibili, così che le corrispondenze da utilizzare per i calcoli sono risultate numericamente esigue; inoltre non erano note le posizioni reali delle immagini a disposizione, cosa che preclude la possibilità di una stima quantitativa dell'errore. È stato però possibile applicare ad almeno due casi (un rettilineo e una curva) il metodo, con risultati qualitativamente soddisfacenti.

Uno degli scopi che ci si proponeva in questo lavoro era quello di confrontare i metodi con due e tre viste. Gli esperimenti fatti a questo proposito, sia in termini di dati sintetici che sperimentali, evidenziano i vantaggi dell'uso del tensore trifocale. In particolare:

1. La robustezza della stima della geometria epipolare rispetto ad errori di misurazione e alle false corrispondenze è notevolmente superiore rispetto al caso delle due viste. Considerato che gli errori di stima del moto si accumulano lungo il percorso della videocamera, è fondamentale che siano minimizzati per contenere l'errore di deriva.

2. La propagazione dell'informazione di scala è semplice ed affidabile, a differenza del caso a due viste. In quest'ultimo caso, inoltre, è necessario comunque stimare la scala utilizzando punti visibili in tre viste consecutive, cosa che rende il metodo un ibrido tra i metodi a due e tre viste. Utilizzare direttamente tre viste sia per la stima della geometria epipolare che per la propagazione della scala è più completo.

Per contro, le criticità maggiori sono

1. Il tempo di calcolo aumenta rispetto al caso con due viste, data la maggiore complessità dei calcoli necessari per ottenere il tensore trifocale. Questo può rendere più difficile un'implementazione *real-time* del metodo, anche se gli sviluppi tecnologici possono ovviare al problema.

2. Per l'uso su sequenze di sfere è necessario che le immagini siano prese a breve distanza l'una dall'altra. La sparsità delle bolle presenti su Google

Street View è la principale responsabile della difficoltà ad applicare il metodo in questo ambito. Lo stesso discorso vale anche per il metodo con due viste.

Ulteriori Sviluppi

L'obiettivo della tesi di preparare una *pipeline* completa di stima del moto proprio di una videocamera omnidirezionale è stato affrontato utilizzando un approccio modulare. In questo modo è stato possibile combinare vari tasselli, lasciando spazio a miglioramenti specifici.

Il calcolo dei punti notevoli delle immagini, il recupero delle corrispondenze, la trasformazione delle coordinate da equirettangolari a sferiche, i metodi di stima della geometria epipolare sono operazioni indipendenti, che possono essere sostituite con metodi alternativi. Indichiamo alcune direzioni che potrebbero essere interessanti per un miglioramento della *pipeline*.

La fase di estrazione dei punti notevoli può essere oggetto di miglioramento sia scegliendo diversi descrittori che agendo su sezioni rettificate dell'immagine, come prospettato nella sezione 2.2 a pagina 36. Per quanto riguarda il calcolo dei punti notevoli, potrebbero essere utilizzate differenti metodi per l'estrazione dei *corner* in modo da favorire l'emergere di punti ad angolo retto, molto comuni in ambito urbano: l'algoritmo proposto utilizza differenze di gaussiane (DoG), ma potrebbe essere interessante sperimentare un metodo di tipo Harris Corner Detector ([7]). Una seconda possibilità è di cambiare descrittore, ad esempio puntando su descrittori invarianti per affinità o distorsioni prospettiche (si veda ad esempio [3]).

Per migliorare le prestazioni in termini di velocità, oltre a un'implementazione dell'algoritmo in codice compilabile, si può agire sul collo di bottiglia, che per l'algoritmo è la fase di eliminazione degli *outlier* tramite RANSAC. Esistono evoluzioni di questo algoritmo che permettono di rendere più rapido il processo di eliminazione degli *outlier*, come il *pre-emptive RANSAC* proposto in [15].

Il metodo proposto è stato pensato per essere corredato di informazioni GPS o di odometria meccanica, ovvero le informazioni di spostamento dell'automobile. La scala della ricostruzione viene infatti fissata dalla *baseline* imposta tra le prime due sfere. Una stima iniziale errata della scala avrebbe quindi ripercussioni su tutta la catena di sfere ricostruita. Per evitare questo problema si può applicare un meccanismo di feedback costante tra le informazioni GPS e la *pipeline* di ricostruzione, in modo che la precisione della scala migliori con l'aumentare del numero di immagini utilizzate.

Appendice A

Codice Matlab

A.1 Stima del moto con tre viste su immagini reali

```
%Uses google street view images to compute SfM
%(using trifocal tensor)

%import image data
clear all;
run('../vlfeat-0.9.13/toolbox/vl_setup.m');
%precomputed SIFT image points and matches
load('gsv/imagedata.mat');


%config variables
baseline = 10; %user assigned scale factor
numSph = 9;
err = 0.004; %for RANSAC error

%cumulative values
cumR = eye(3);
cumt = zeros(3,1);


for k = 1:numSph-2

    fprintf('\n Sphere %d',k)

    matches12 = s{k}.matches_with_next;
    matches23 = s{k+1}.matches_with_next;

    %join of the matches sets
    %1) find the points of interest of
```

```
    %the im2 which appears in both match sets
    [matches,ind12,ind23] = intersect(matches12(2,:),matches23(1,:));
    %2) creates new triple match set
    matches=[matches12(1,ind12);matches12(2,ind12);matches23(2,ind23)];
    fprintf(', %d matches.\n',length(matches))

    %copy matching points for ease of use
        x1 = s{k}.x(:,matches(1,:));
        x2 = s{k+1}.x(:,matches(2,:));
        x3 = s{k+2}.x(:,matches(3,:));

        %calls RANSAC routine on the
        %trifocal tensor estimation function
[T,R21,t21,R31,t31,inliers] = fg_ransac_triftensor(x1,x2,x3,err);

%first 3 spheres
if (k==1)

    t21 = t21*baseline; %fixing scale
    t31 = t31*baseline;

    %save computed values
    s{1}.R=eye(3);
    s{1}.t=zeros(3,1);
    s{2}.R = R21;
    s{2}.t = t21;
    s{2}.dt = t21;
    s{1}.T = T;
    s{3}.R = R31;
    s{3}.t = t31;


    %triangulate points:  mean of the 2 triangulations
    C1 = [0;0;0];
    C2 = s{2}.t;
    C3 = s{3}.t;
    Xest = zeros(size(s{1}.x));
    for i=inliers
        X1 = s{1}.x(:,i);
        X2 = s{2}.t + s{2}.R \ s{2}.x(:,i);
        X3 = s{3}.t + s{3}.R \ s{3}.x(:,i);
        %triangulate with spheres s1-s2, s1-s3 (widest baseline)
        Xest(:,i) = 0.5*(fg_midpoint_triangulation(C1,X1,C2,X2)...
        + fg_midpoint_triangulation(C1,X1,C3,X3));
    end



    s{1}.X = Xest;
```

```
    s{1}.inliers = inliers;

    %update cum values
    cumR = s{2}.R;
    cumt = s{2}.t;

else %general case:  k=2,3,4,5..

    %since ||t21||=1 this is the scale
    scale = norm(s{k+1}.t - s{k}.t);
    t21 = t21*scale;
    t31 = t31*scale;

    %triangulate points in the (k-1)-th coord system (which is
    %the one which has cumR, cumt from the global coord system)
    %using mean of the 2 triangulations
    C1 = zeros(3,1); %s{k-1} sphere
    C2 = t21;
    C3 = t31;
    Xest = zeros(size(s{k}.x));
    for i=inliers
        X1 = s{k}.x(:,i);
        X2 = t21 + R21 \ s{k+1}.x(:,i);
        X3 = t31 + R31 \ s{k+2}.x(:,i);
        %triangulate with spheres s1-s2, s1-s3 (widest baseline)
        %note:  Xest are in the k-th coord system
        Xest(:,i) = 0.5*(fg_midpoint_triangulation(C1,X1,C2,X2)...
        + fg_midpoint_triangulation(C1,X1,C3,X3));
    end

    %convert 3d points to global coord system
    for i = inliers
        Xest(:,i) = cumt + cumR \ Xest(:,i); %transform to global
    end


    %aggiorno valori k+1
    s{k+2}.t = cumt + cumR \ t31;
    s{k+2}.R = R31 * cumR;

    %update cum values
    cumt = s{k+1}.t;
    cumR = s{k+1}.R;
    s{k}.X = Xest;
    s{k}.T = T;
    s{k}.inliers = inliers;

end
end
```

```
%print results
figure;
hold on;
err = 0;
estold=zeros(3,1);
for i=2:numSph
est = s{i}.t;
quiver(estold(1),estold(2),est(1)-estold(1),est(2)-estold(2),'r');
plot(est(1),est(2),'*r');
legend('Estimated');
axis('equal');
estold=est;
end
plot(0,0,'*b');
```

A.2 Stima del tensore trifocale e di R, t

```
function [Tfin,R21,t21,R31,t31] = fg_linear_trifocal_tensor(x1,x2,x3)
%FG_LINEAR_TRIFOCAL_TENSOR Computes trifocal tensor using linear algorithm
% x1:  3xn vector (x,y,z) with image coordinates from first camera
% x2:  3xn vector with image coordinates from second camera
% x3:  3xn vector with image coordinates from third camera
%
% Note:  the output has norm(t21)=1, and norm(t31) consistent with that.

A = buildAmatrix(x1,x2,x3);
[ , ,V] = svd(A);
t0 = V(:,27); %solution is vector corresp.  to smallest singular value
T0 = permute( reshape(t0,3,3,3) , [2 1 3] ); %reorder

% Ensure that that trifocal tensor is geometrically valid by retrieving
% its epipoles and performing algebraic minimization
[e2,e3] = e_from_T(T0);

%levenberg marquardt optimization
e0 = [e2;e3];
[e,Ssq,CNT] = LMFsolve(@(e) objective_function(e,A),e0,'XTol',1e-10);
Ssq;
CNT
change_from_x0 = norm(e-e0);
```

```
e2 = e(1:3);
e3 = e(4:6);
E = E_from_ee(e2,e3);
t = minAlg_5p6(A,E);

Tfin = permute( reshape(t,3,3,3) , [2 1 3] );

%extract essential matrices
E21 = fg_skew(e2)*[Tfin(:,:,1)*e3,Tfin(:,:,2)*e3,Tfin(:,:,3)*e3];
E31 = fg_skew(e3)*[Tfin(:,:,1)'*e2,Tfin(:,:,2)'*e2,Tfin(:,:,3)'*e2];


%R,t retrieval

%second camera (E21):
[U,S,V] = svd(E21);
W = [0 -1 0; 1 0 0; 0 0 1];

s2 = (S(1,1)+S(2,2))*0.5; %scale information of sphere 2

%rotation matrices
R1 = U*W*V';
R1 = R1/det(R1); %normalize to have det(R) = 1

R2 = U*W'*V';
R2 = R2/det(R2);



%extraction of baseline t
t1 = V*W*diag([1 1 0])*V';
t1 = [t1(3,2) t1(1,3) t1(2,1)];

t_est = t1'*10;

%initialize
R=eye(3);
t = zeros(3,1);


%reconstruct a 3d point to choose the right R,t:  the
%Xest point should be in front of both cameras
a1 = [0;0;0]; %1st camera centre
a2 = x1(:,1); %1st camera point (already global system)

%with R1,t
b1 = t_est; %2nd camera centre
b2 = t_est + R1\x2(:,1);%2nd camera image point, in global coord
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
```

```
%test:  Xest is in front of both cameras
'R1,t';
angle1 = dot(a2-a1,Xest-a2);
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R1;
t = t_est;
end

%with R1,-t
b1 = -t_est;
b2 = -t_est + R1\x2(:,1);
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
%test:  Xest is in front of both cameras
'R1,-t';
angle1 = dot(a2-a1,Xest-a2);
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R1;
t = -t_est;
end


%R2,t
b1 = t_est;
b2 = t_est + R2\x2(:,1);
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
%test:  Xest is in front of both cameras
'R2,t';
angle1 = dot(a2-a1,Xest-a2);
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R2;
t = t_est;
end

%R2,-t
b1 = -t_est;
b2 = -t_est + R2\x2(:,1);
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
%test:  Xest is in front of both cameras
'R2,-t';
angle1 = dot(a2-a1,Xest-a2);
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R2;
t = -t_est;
end
```

```
R21 = R;
t21 = t;

X21 = zeros(3,1);
a1=zeros(3,1);
b1 = t21;
for i=1:length(x1)
a2 = x1(:,i);
b2 = t21 + R21\x2(:,i);
X21 = X21 + fg_midpoint_triangulation(a1,a2,b1,b2);
end
X21 = X21 / length(x1);


%third camera:
[U,S,V] = svd(E31);

s3 = (S(1,1)+S(2,2))*0.5; %scale information of sphere 3
W = [0 -1 0; 1 0 0; 0 0 1];

%rotation matrices
R1 = U*W*V';
R1 = R1/det(R1); %normalize to have det(R) = 1

R2 = U*W'*V';
R2 = R2/det(R2);

%extraction of baseline t
t1 = V*W*diag([s2/s3 s2/s3 0])*V';
t1 = [t1(3,2) t1(1,3) t1(2,1)];
t_est = t1'*10;

%initialize
R=eye(3);
t = zeros(3,1);


%reconstruct a 3d point to choose the right R,t:  the Xest
%point should be in front of both cameras
a1 = [0;0;0]; %1st camera centre
a2 = x1(:,1); %1st camera point (already global system)

%with R1,t
b1 = t_est; %2nd camera centre
b2 = t_est + R1\x3(:,1);%2nd camera image point in global coord
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
%test:  Xest is in front of both cameras
'R1,t';
angle1 = dot(a2-a1,Xest-a2);
```

```
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R1;
t = t_est;
end

%with R1,-t
b1 = -t_est;
b2 = -t_est + R1\x3(:,1);
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
%test:  Xest is in front of both cameras
'R1,-t';
angle1 = dot(a2-a1,Xest-a2);
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R1;
t = -t_est;
end


%R2,t
b1 = t_est;
b2 = t_est + R2\x3(:,1);
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
%test:  Xest is in front of both cameras
'R2,t';
angle1 = dot(a2-a1,Xest-a2);
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R2;
t = t_est;
end

%R2,-t
b1 = -t_est;
b2 = -t_est + R2\x3(:,1);
Xest = fg_midpoint_triangulation(a1,a2,b1,b2);
%test:  Xest is in front of both cameras
'R2,-t';
angle1 = dot(a2-a1,Xest-a2);
angle2 = dot(b2-b1,Xest-b2);
if (angle1>0 && angle2>0)
R = R2;
t = -t_est;
end

R31 = R;
t31 = t;
```

```
t21 = t21 * 0.1;
t31 = t31 * 0.1;

end


function A = buildAmatrix (x1, x2, x3)
%build the A matrix for linear estimation of the trifocal tensor
    zero = zeros(length(x1),1);
    for k=1:3 %da ridurre a funzione
        % Build the k-th A matrix to get T_k
        A{k} = [
        %1st equation
        (-x1(k,:).*x2(3,:).*x3(3,:))', zero, ...
        (x1(k,:).*x2(3,:).*x3(1,:))', zero, zero, zero,...
        (x1(k,:).*x2(1,:).*x3(3,:))', zero,...
        (-x1(k,:).*x2(1,:).*x3(1,:))'  ; ...
        %2nd equation
        zero, (-x1(k,:).*x2(3,:).*x3(3,:))',...
        (x1(k,:).*x2(3,:).*x3(2,:))', zero, zero, zero, zero,...
        (x1(k,:).*x2(1,:).*x3(3,:))', ...
        (-x1(k,:).*x2(1,:).*x3(2,:))'; ...
        %3rd equation
        zero, zero, zero, (-x1(k,:).*x2(3,:).*x3(3,:))',...
        zero, (x1(k,:).*x2(3,:).*x3(1,:))', ...
        (x1(k,:).*x2(2,:).*x3(3,:))', zero, ...
        (-x1(k,:).*x2(2,:).*x3(1,:))'; ...
        %4th equation
        zero, zero, zero, zero, (-x1(k,:).*x2(3,:).*x3(3,:))',...
        (x1(k,:).*x2(3,:).*x3(2,:))', zero, ...
        (x1(k,:).*x2(2,:).*x3(3,:))', (-x1(k,:).*x2(2,:).*x3(2,:))';
        ];
    end
    A = [A{1}, A{2}, A{3}];


end

%-------------------------------------
% Function to build the relationship matrix which represent
% T_i^jk = a_i^j * b_4^k - a_4^j * b_i^k
% as t = E * aa, where aa = [a'(:)  ; b'(:)], (note:  for representation only)


    function E = E_from_ee(e2,e3)


e2Block = [ diag([-e2(1)*ones(1,3)])
diag([-e2(2)*ones(1,3)])
```

```
diag([-e2(3)*ones(1,3)])];

e3Block = zeros(9,3);
e3Block(1:3,1) = e3;
e3Block(4:6,2) = e3;
e3Block(7:9,3) = e3;

E = zeros(27,18);
E( 1:  9,[1:3,10:12]) = [e3Block e2Block];
E(10:18,[4:6,13:15]) = [e3Block e2Block];
E(19:27,[7:9,16:18]) = [e3Block e2Block];

end


function F = objective_function(e,A)
%extract e2, e3
e2 = e(1:3);
e3 = e(4:6);

%compute E matrix
E = E_from_ee(e2,e3);

%solve minimization problem (t=Ea)
t = minAlg_5p6(A,E);

%compute error vector err=AEa=At
F = A*t;
end
```

Ringraziamenti

Grazie è una parola che andrebbe detta molto, molto più spesso. Purtroppo sono caratterialmente un po' asciutto, per cui approfitto di queste righe per riconoscere i miei debiti francamente. Ciò che sono e ho realizzato non è infatti che un debito, cosa di cui sono perfettamente consapevole: niente di quello che ho e sono è meritato. Tutto è grazia.

Ringrazio innanzitutto Marco ed Emanuele, che mi hanno seguito in questa avventura con tanta pazienza per le mie sparizioni e solidarietà per il mio percorso di vita, così ricco in questi ultimi mesi. Non solo docenti ma compagni di strada.

Ringrazio il *noster Politeknik,* da cui sembra che stia uscendo, per quanto bene mi ha formato. Se Dio vuole, in questi cinque anni son diventato ingegnere senza perdere del tutto l'amore per ciò che è umano.

Ringrazio i miei compagni di corso più vicini, i *survivors* delle matricole classe 2006: Ale, Davide, Francesco, Federico, Filippo. Compagni di cammino in molte occasioni: di questi anni ricorderò soprattutto i progetti fatti insieme per gli esami, fatti di ore di lavoro collaborativo, di idee cestinate e caffé disgustosi, di corse contro il tempo e di belle soddisfazioni.

Ringrazio l'AC e la FUCI, intensi ambiti di impegno e crescita (e di utilizzo del tempo libero!), che così tanto hanno contribuito a formare quello che sono. Dato che sono nettamente di parte, e che ormai son vecio, posso aggiungere che della FUCI, al di là degli incarichi o delle riunioni, ricordo tutti i volti fucini (perché coi nomi si sa che incespico), volti gioiosi e limpidi e appassionati, capaci di speranza. I nomi sarebbero troppi da ricordare, vi ringrazio tutti, da quelli che mi hanno visto entrare al primo anno quella sera a Lodi in cui si parlava di felicità, a quelli che ho abbracciato al Congresso a Reggio Calabria col magone che mi bloccava le parole.

Ringrazio la mitica compagnia d'amici che siamo: colorita, franca, allegra, generosa. Cinque anni e mezzo fa non pensavo che potesse esistere una comunità simile, e invece ho avuto la grazia di trovarmici dentro. Grazie per tutto. Grazie anche ai "vecchi amici": che bello che il nostro legame abbia saputo superare il trauma da fine liceo, e che stia diventando più saldo col tempo.

Ringrazio la famiglia allargata, nonni zii zie e cugini vari, per la bella esperienza di comunità che anche qui trovo. Siamo in tanti e pure riusciamo a dire la parola "noi" in modo vero. Oggi metto un piede nel mondo con una sicurezza che viene anche dalla solidità della nostra bella esperienza di famiglia. Un grazie particolare per quanto mi avete dato va a Luca e a Gianfra, e un pensiero speciale, quotidiano, a Stefano e a Matteo. Grazie anche a Moni, sorella di vita più che cugina.

Non finirò mai di ringraziare la famiglia ristretta, vero crogiuolo: è qui che si sta gomito a gomito e si imparano e vivono in profondità le relazioni. Mamma, Papà, Ale, Simo: so che non potrò mai restituire quanto ricevuto; proverò a trasmetterlo al mondo.

Rimangono i due grazie più grandi, come sempre alla fine.

A Dany, tu che sei la donna della mia vita, grazie per tutto quanto è stato e per quanto sarà (e so che sarà bello).

A Dio, a cui devo tutto, grazie di cuore: non meritavo niente e ho avuto tutto. Ricordamelo ogni giorno e aiutami a spandere amore in questo bel mondo.

Bibliografia

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. 2006.

[2] Peng Chang and Martial Hebert. Omni-directional structure from motion. *Proceedings of the IEEE Workshop on Omnidirectional Vision*, pages 127–133, 2000.

[3] Liang Cheng. Robust affine invariant feature extraction for image matching. *Geoscience and Remote Sensing Letters, IEEE*, 5:246 – 250, 2008.

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM 24*, 6:381–395, 1981.

[5] Jun Fujiki, Akihiko Torii, and Shotaro Akaho. Epipolar geometry via rectification of spherical images. *MIRAGE 2007*, pages 461–471, 2007.

[6] Joshua Gluckman and Shree Nayar. Ego-motion and omnidirectional cameras. *Sixth International Conference on Computer Vision*, pages 999–1005, 1998.

[7] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[8] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2000.

[9] http://maps.google.it/intl/it/help/maps/streetview/.

[10] http://www.astute project.eu/.

[11] Fredrik Kahl. Multiple view geometry and the l_∞ norm. *Proceedings of the tenth IEEE international Conference on Computer Vision*, 2005.

[12] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[13] Branislav Micusik. *Two-View Geometry of Omnidirectional Cameras*. PhD thesis, Faculty of the Electrical Engineering, Czech Technical University in Prague, 2004.

[14] Branislav Micusik, Daniel Martinec, and Tomas Pajdla. 3d metric reconstruction from uncalibrated omnidirectional images. 2004.

[15] David Nister. Preemptive ransac for live structure and motion estimation. *Ninth IEEE International Conference on Computer Vision*, 6:199 – 206, 2003.

[16] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Matematica Numerica*. Springer, 2008.

[17] Tomas Svoboda and Tomas Pajdla. Epipolar geometry for central catadioptric cameras. *International Journal of Computer Vision*, 1:23–27, 2002.

[18] Tomas Svoboda, Tomas Pajdla, and Vaclav Hlavac. Epipolar geometry for panoramic cameras. 1998.

[19] Akihiko Torii, Michal Havlena, and Tomas Pajdla. From google street view to 3d city models. *IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 2009.

[20] Akihiko Torii, Atsushi Imiya, and Naoya Ohnishi. Two- and three- view geometry for spherical cameras.

[21] Zhengoyu Zhang. Determining the epipolar geometry and its uncertainty: A review. 1996.