UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Ricostruzione 3D di una frana e monitoraggio dello smottamento tramite una coppia di fotocamere

Relatore: Emanuele Menegatti

Correlatore: Mauro Antonello

Laureando: *Lorenzo Visentin*

Data di laurea: 9 dicembre 2014

ANNO ACCADEMICO 2014/2015

Alla mia famiglia, che mi ha sostenuto durante gli anni di studio.
Agli amici con cui ho condiviso questa esperienza universitaria.
Al dott. Mauro Antonello, per avermi assistito nello svolgimento della tesi.

The LaTeX template for this thesis is available at
`www.dei.unipd.it/~avanzini`

Indice

Sommario

I temi della prevenzione e del controllo del territorio hanno cominciato a riscuotere un interesse sempre maggiore negli ultimi decenni. L'obiettivo di questa tesi è quello di fornire un supporto alle attività di monitoraggio, al fine di limitare le conseguenze degli eventi naturali di una certa entità. In particolare, un sistema composto da due fotocamere ha permesso di analizzare l'evoluzione della frana del Tessina, localizzata nella provincia di Belluno. Mediante l'uso di tecniche come il template matching è stato possibile individuare i vettori spostamento associati al terreno; tali vettori sono poi stati ottimizzati ricorrendo ad un algoritmo innovativo. Pur non possedendo dati di calibrazione, il modello 3D della frana è stato ricostruito ricorrendo alla Structure From Motion; infine, i vettori ricavati nella fase precedente sono stati proiettati sulla scena tridimensionale.

Prefazione

Con *visione artificiale* si intende tutto ciò che riguarda l'elaborazione digitale di immagini da parte di una macchina. Solitamente questa locuzione identifica l'insieme dei processi coinvolti nella costruzione di un modello 3D della realtà; tuttavia, è importante anche saper interpretare l'informazione contenuta in questo modello. I benefici in termini economici e di qualità del prodotto che si possono ottenere, hanno fatto sì che negli ultimi anni questa disciplina si sia diffusa con successo in numerosi settori. La visione artificiale si ritrova in ambito industriale, per esempio per il riconoscimento dei difetti [15], ma anche in campo medico [5] e nel controllo di veicoli autonomi [12].

L'obiettivo di questa tesi è quello di sfruttare gli algoritmi di visione per dare un supporto alle attività di monitoraggio ambientale a fini di prevenzione. La storia ha infatti avuto modo di insegnarci che una manutenzione inadeguata e il disinteresse nei confronti dei temi ambientali spesso aumentano il rischio che eventi naturali di una certa entità portino con sé danni e distruzione. La realizzazione di sistemi che siano in grado di monitorare costantemente e in modo autonomo l'evoluzione del territorio si rivela quindi di vitale importanza nell'ottica di voler limitare conseguenze calamitose.

Sebbene i risultati conseguiti con questo lavoro si possano applicare anche ad altri territori, l'ambiente naturale oggetto dell'analisi è la frana del Tessina, localizzata in provincia di Belluno. La sua superficie è costituita da materiale incoerente che, soggetto all'azione della forza di gravità e a causa delle piogge, tende progressivamente a scivolare verso valle. In passato sono stati impiegati sistemi laser con lo scopo di rilevare come la conformazione del paesaggio evolve nel tempo. Tuttavia l'elevato costo e la scarsa flessibilità di questa tipologia di strumentazione hanno favorito recentemente l'impiego di sistemi alternativi. Nel caso in esame si è fatto uso di una coppia di fotocamere disposte sulla sommità della scena, che hanno rilevato giorno per giorno le trasformazioni che avvenivano nella frana. Il contributo dato con questa tesi è stato la creazione di un software in grado di captare gli spostamenti di materiale e di visualizzarli sotto forma di vettori in sovrimpressione alle immagini. A tale scopo si è fatto uso di una tecnica nota come *template matching*; l'apporto originale che è stato dato consiste nell'ottimizzazione dei vettori individuati grazie al confronto con le informazioni ricavabili da una sequenza di scatti consecutivi.

Nella seconda parte della tesi è stato costruito un modello tridimensionale della scena catturata dalle due fotocamere. Dalla teoria è infatti risaputo che a partire da una coppia di fotogrammi che riprendono uno stesso oggetto da due punti di vista distinti, è possibile risalire alla conformazione dell'oggetto nello spazio 3D. Solitamente questo risultato si può

ottenere grazie a processi di *calibrazione*; le grandi dimensioni della zona ripresa ed una serie di inconvenienti relativi alla posizione reciproca delle fotocamere hanno tuttavia impedito di applicare questo criterio.

A questo problema si è posto rimedio facendo uso della *Structure From Motion*; questa tecnica è molto diffusa nell'ambito della robotica in quanto permette ad un robot di ricostruire in tre dimensioni gli oggetti che lo circondano. Solitamente la Structure From Motion viene utilizzata quando si dispone di una sola telecamera che si muove nel tempo all'interno di un ambiente; anche quando le telecamere sono due si può ricorrere a questo metodo, dato che questa situazione si può ricondurre al caso precedente. In particolare, non disponendo di dati di calibrazione, sono stati impiegati due approcci distinti: il primo ignora la conoscenza dei parametri intrinseci delle fotocamere, mentre il secondo ne effettua una stima e per mezzo di essa ricostruisce il modello 3D in modo più affidabile. Una volta ricostruito l'ambiente tridimensionale è possibile proiettare i vettori spostamento individuati in precedenza su questo modello.

Capitolo 1

Introduzione

1.1 Obiettivi e soggetti coinvolti

I temi della prevenzione e del controllo del territorio hanno cominciato a riscuotere un interesse sempre maggiore negli ultimi decenni; nel corso della storia, infatti, si ha avuto modo di constatare come una manutenzione inadeguata e un monitoraggio poco accurato dell'ambiente abbiano spesso accentuato i danni provocati dagli eventi naturali catastrofici, con conseguenti perdite umane ed ingenti danni economici. Al fine di scongiurare il manifestarsi di eventi simili, il Dipartimento di Ingegneria Civile, Edile e Ambientale dell'università di Padova ha preso in considerazione la tematica del monitoraggio di ambienti naturali di grandi proporzioni.

Si tratta di un progetto piuttosto ampio che coinvolge il Dipartimento di Ingegneria Civile, Edile e Ambientale, il Dipartimento di Ingegneria dell'Informazione e l'azienda IT+Robotics, spin-off dell'università di Padova. Quest'ultima in particolare, che fin dalla sua nascita collabora attivamente con il laboratorio di sistemi intelligenti autonomi (IAS-Lab) del Dipartimento di Ingegneria dell'Informazione, ha avuto il compito di fornire la strumentazione necessaria a catturare le informazioni rilevanti e ha messo a disposizione le proprie competenze; il lavoro vero e proprio di analisi è stato invece commissionato allo IAS-Lab. Questa tesi, realizzata proprio in questo laboratorio, ha l'obiettivo di contribuire alle attività di monitoraggio realizzando e perfezionando algoritmi in linguaggio C++ che possano essere applicati all'osservazione dell'evoluzione del territorio. L'interpretazione dei dati rilevati ed i conseguenti provvedimenti da intraprendere sono invece a discrezione del Dipartimento di Ingegneria Civile, Edile e Ambientale.

1.2 Analisi della frana del Tessina

Il paesaggio preso in esame è la frana del Tessina, nel comune di Chies d'Alpago in provincia di Belluno. Questa frana è costituita da un declivio molto esteso caratterizzato da una superficie di materiale incoerente che, soggetto all'azione della forza di gravità e a causa delle piogge, è indotto ad uno scivolamento verso valle. Il movimento franoso ha

avuto origine cinquant'anni fa, quando una serie di successivi distacchi aveva interessato le pendici del Monte Teverone, segnando profondamente la morfologia dell'area.

La rilevazione dei movimenti del suolo avviene per mezzo di una coppia di fotocamere piazzate in prossimità della scena. In passato questa operazione è stata realizzata sfruttando sistemi laser con un buon livello di precisione. L'impiego di telecamere al posto dei laser presenta lo svantaggio di richiedere un'elaborazione molto più difficoltosa; tuttavia, a questo punto sfavorevole si contrappongono due vantaggi fondamentali: un prezzo molto più basso ed una flessibilità d'uso maggiore.

L'analisi di un terreno soggetto a colamenti può essere effettuata sfruttando tecniche note in letteratura che consentono di determinare come avviene lo spostamento di un flusso di materiale viscoso. In particolare, l'intento è stato quello di ottenere i vettori di spostamento associati ai punti che costituiscono la frana. La tecnica sulla quale mi sono basato si chiama PIV (Particle Image Velocimetry) e consiste nel determinare il moto delle particelle che compongono un flusso a partire da immagini scattate in sequenza; combinando poi le informazioni deducibili dalla coppia di fotocamere si possono ricavare gli spostamenti in tre dimensioni. L'attività di individuare i vettori che descrivono il moto del terreno non è esente da problematiche. Questo perché il terreno su cui poggiano le fotocamere, così come quello oggetto dell'analisi, è esso stesso sottoposto a fenomeni di cedimento; inoltre nel corso dell'anno le condizioni di luminosità e la vegetazione presente cambiano notevolmente.

La seconda questione affrontata è stata la creazione di un modello tridimensionale della frana, sul quale andare a proiettare i vettori che rappresentano gli spostamenti individuati nella fase precedente. Date le distanze e le dimensioni considerevoli della zona coinvolta nello scivolamento, risulta impossibile applicare i metodi standard che fanno uso della calibrazione per ottenere una scena 3D; tale inconveniente è stato risolto avvalendosi di procedimenti alternativi.

Una quantità di tempo considerevole durante la realizzazione del lavoro è stata spesa al fine di ridurre l'onere computazionale degli algoritmi creati; sebbene non si tratti di un requisito essenziale per quest'applicazione, è opportuno ridurre al minimo il tempo di calcolo per garantire che il software possa essere eseguito almeno una volta al giorno e per assicurare maggiore tempestività qualora vengano rilevati fenomeni sospetti.

Nonostante il lavoro sia stato realizzato su una frana in particolare, è applicabile anche ad altri ambienti naturali: è sufficiente intervenire sui parametri in ingresso ed eventualmente riadattare qualche dettaglio implementativo.

1.3 Articoli consultati prima della realizzazione

La realizzazione del lavoro di tesi è stata preceduta da una laboriosa ricerca e consultazione di articoli attinenti la materia. Durante questa fase mi sono concentrato prevalentemente sull'approfondimento dei vari aspetti della PIV, sui campi di applicazione e sulle possibili implementazioni; questo perché in un primo momento la modalità con cui procedere ed il prodotto finale da ottenere non erano ancora ben chiari. Una volta

cominciata la stesura del codice di librerie e programmi eseguibili, ho dovuto affrontare altre questioni; l'attività di ricerca in letteratura si è quindi protratta per tutta la durata della tesi e col tempo si è orientata verso letture più significative.

Nel seguito presento una breve panoramica in cui riporto il contenuto dei più rilevanti tra i primi articoli letti; gli articoli consultati in un momento successivo, durante la scrittura del codice, sono citati nei capitoli successivi. In generale si può notare che gli ambiti a cui vengono applicati i concetti di mio interesse sono molto diversificati tra loro.

L'articolo *"Stereo particle image velocimetry measurement of 3D soil deformation around laterally loaded pile in sand"* [30] riporta un esempio di utilizzo della Stereo PIV (tecnica simile alla PIV che sfrutta una coppia di telecamere) per la misurazione della deformazione del suolo attorno ad un palo piantato nella sabbia e sottoposto ad uno sforzo. Il metodo esposto dagli autori per trovare i vettori spostamento contiene spunti innovativi: a differenza di quanto fatto nei lavori precedenti, piuttosto di considerare la funzione che mappa i punti nel piano dell'immagine in quelli del piano dell'oggetto, le coordinate di quest'ultimo sono determinate direttamente nel sistema di riferimento del mondo.

In *"Adaptive Cross Correlation for Imaging Displacements in Soils"* [18] viene illustrato un algoritmo avanzato detto Adaptive Cross Correlation (ACC), impiegato in sostituzione della classica Digital Image Correlation (DIC). Questo algoritmo utilizza finestre di dimensione variabile e metodi di spostamento di queste finestre; il risultato ottenuto è una riduzione dell'errore associato alla DIC convenzionale. Il punto cruciale che sta alla base dell'algoritmo è la constatazione che la dimensione del picco della cross-correlazione è proporzionale all'area che le due finestre di interrogazione hanno in comune. Di conseguenza la ACC sfrutta lo spostamento ed il ridimensionamento delle finestre al fine di massimizzare la dimensione dell'area correlata. L'articolo spiega anche come calcolare la cross-correlazione (convenzionale) nel caso in cui le due immagini di una coppia abbiano luminosità diverse. Spiega inoltre come velocizzare il calcolo sfruttando la FFT.

Nell'articolo *"Measuring soil deformation in geotechnical models using digital images and PIV analysis"* [27] è presente un'applicazione della PIV ad immagini di terreni al fine di ottenerne una misura accurata della deformazione. Nel caso di sabbia o di argilla colorata con un'opportuna texture è possibile ottenere una precisione di 1/15 pixel. Anche in questo caso si possono ricavare degli spunti interessanti: per esempio, il fatto di eseguire un'interpolazione spline per la determinazione del massimo della cross-correlazione. Oppure le considerazioni su quali aree di ricerca valutare quando si fa il confronto con la finestra di interrogazione, e sulle dimensioni della finestra di interrogazione. Infine, il fatto di calcolare i vettori spostamento solo per le finestre di interrogazione in cui è presente sufficiente texture.

Ricordo infine per completezza altre letture che ho esaminato (sempre durante la fase iniziale), anche se si sono dimostrate meno utili. L'articolo [16] indaga sulla risoluzione spaziale che si può ottenere quando si usa la PIV, in funzione delle particelle traccianti

e del sistema di ripresa. [23] spiega come ottenere il vettore di spostamento tridimensionale; parla delle possibili configurazioni, dell'errore della componente che esce dal piano e della ricostruzione del vettore 3D tramite metodo geometrico o tramite calibrazione. [24] tratta la PIV applicata ai fluidi allo stato liquido, e sfrutta l'effetto Moiré per la registrazione delle coppie di immagini. In [28] le informazioni sullo spostamento ricavate dai passi precedenti vengono impiegate per l'interrogazione successiva, migliorando il rilevamento (grazie al miglior particle matching) e permettendo di ridurre la dimensione della finestra aumentando così la risoluzione spaziale. Infine in [13] è presentato un algoritmo di registrazione con finestre gaussiane per il calcolo dello spostamento. L'algoritmo riduce gli effetti negativi dovuti alla scelta della dimensione della finestra, adattando da sé la forma della finestra gaussiana sfruttando un metodo ai minimi quadrati pesato.

1.4 Struttura del documento

I prossimi due capitoli contengono richiami teorici necessari per comprendere la realizzazione successiva. In particolare, il Capitolo 2 descrive due tipologie di template matching, feature-based template matching e template-based matching; la prima parte si sofferma sull'algoritmo SIFT per il rilevamento delle corrispondenze, la seconda illustra varie tecniche che valutano la somiglianza di due immagini. Il secondo capitolo di teoria, il Capitolo 3, dapprima propone un modello per rappresentare una fotocamera, poi si serve di questo per analizzare sistemi composti da due o più telecamere. Quando non si dispone dei dati di calibrazione di queste telecamere, o quando se ne possiede solo una parte, è ancora possibile ricostruire una scena in tre dimensioni sfruttando algoritmi di Structure From Motion, molto utilizzati nel campo della robotica. Il Capitolo 4 si occupa dell'implementazione del lavoro, contenente il contributo originale, ed è suddiviso nelle attività di astrazione del dataset, preprocessing, processing e postprocessing. Considerazioni sul lavoro svolto e possibili miglioramenti sono riportati nel Capitolo 5. Per finire nell'Appendice A è presente l'elenco completo di librerie e programmi eseguibili, con i relativi dati di ingresso ed uscita e la descrizione delle funzionalità da essi svolte.

Capitolo 2

Richiami sul template matching

Nell'ambito della visione artificiale spesso si manifesta la necessità di identificare un oggetto contenuto in un'immagine. Questo problema si riscontra ad esempio quando si effettua un controllo della qualità, oppure quando bisogna fornire ad un robot la percezione dell'ambiente che lo circonda. Il *template matching* è una tecnica impiegata a tale scopo; per realizzare questa operazione si possono adottare due diversi approcci: il feature-based template matching (paragrafo 2.1) ed il template-based matching (paragrafo 2.2).

2.1 Feature-based template matching

Il *feature-based template matching* è una tecnica impiegata nel campo dell'elaborazione digitale delle immagini con lo scopo di individuare una porzione di foto (*template*) all'interno di un'altra immagine. Il suo nome è dovuto all'utilizzo delle *feature* (caratteristiche), che rappresentano le entità rilevanti dell'immagine. Il feature-based matching è impiegato nell'algoritmo *SIFT*, che permette di rilevare i punti salienti delle immagini (*keypoint*) e di descrivere tali feature; queste due operazioni sono seguite da una fase di keypoint matching, che individua corrispondenze di feature in una coppia di immagini. La spiegazione dell'algoritmo viene affrontata nel paragrafo 2.1.2.

Talvolta la qualità delle immagini sulle quali viene realizzato il matching non è adeguata; per questo motivo può essere utile un miglioramento preliminare delle immagini sfruttando l'*equalizzazione* o qualche metodo simile (paragrafo 2.1.1).

L'algoritmo SIFT, che rileva le corrispondenze tra i punti delle immagini, può essere sfruttato per distorcere opportunamente una foto in modo da simulare un cambio di punto di vista della fotocamera che l'ha scattata. La ricerca della relazione tra i punti dell'immagine originale e di quella distorta, detta *omografia*, può essere resa più efficace servendosi di un metodo robusto come *RANSAC* (paragrafo 2.1.3). Una volta antidistorte due immagini in modo da renderle sovrapponibili, è possibile formulare un criterio per valutare la somiglianza tra di esse.

2.1.1 Equalizzazione

L'equalizzazione è una tecnica che consente di estendere artificialmente il *range dinamico* di un'immagine, dove con range dinamico si intende il rapporto tra il più grande ed il più piccolo valore di luminosità misurabile. Si hanno in questo modo come conseguenze l'aumento del contrasto complessivo e la maggior visibilità dei dettagli di foto sovra o sottoesposte. Da un punto di vista computazionale quest'operazione non è dispendiosa; inoltre è invertibile, ovvero è possibile riottenere l'istogramma originale. Purtroppo l'uso dell'equalizzazione non sempre porta ai risultati desiderati, causando talvolta un aumento del rumore o l'alterazione dei colori.

Un esempio di equalizzazione di un'immagine fortemente sottoesposta è visibile in Figura 2.1.

Figura 2.1: *Immagine sottoesposta prima e dopo l'equalizzazione. La qualità dell'immagine equalizzata è decisamente superiore rispetto a quella dell'originale.*

Il processo di equalizzazione di un'immagine consiste nella trasformazione del suo *istogramma* delle luminosità al fine di renderlo il più possibile uniforme, ovvero simile ad un segmento orizzontale [4]. L'istogramma di un'immagine in scala di grigi riporta sull'asse x i valori di intensità (da 0 a 255 nel caso di codifica a 8 bit) e sull'asse y la frequenza con cui questi valori compaiono nei pixel. Per fare questo si può ricorrere alla *funzione di ripartizione*, nota anche come funzione di distribuzione cumulativa, definita nel modo seguente [1]:

$$H(i) = \sum_{0 \leq j \leq i} h(j) \qquad 0 \leq i < 255,$$

dove $h(j)$ è la funzione che descrive l'istogramma originale. I valori assunti da $H(i)$ devono essere normalizzati tra 0 e 255 prima di rimappare i valori assunti dai pixel dell'immagine in questo modo:

$$\mathrm{I}_{\mathrm{eq}}(x, y) = H(\mathrm{I}_{\mathrm{orig}}(x, y)),$$

dove I_{eq} e $\mathrm{I}_{\mathrm{orig}}$ sono rispettivamente l'immagine equalizzata e l'originale. Purtroppo l'immagine ottenuta non sempre presenta un istogramma perfettamente piatto; questo si potrebbe ottenere solo se la funzione di densità di probabilità di partenza fosse continua.

Gli istogrammi relativi alle immagini in Figura 2.1 sono riportati in Figura 2.2.

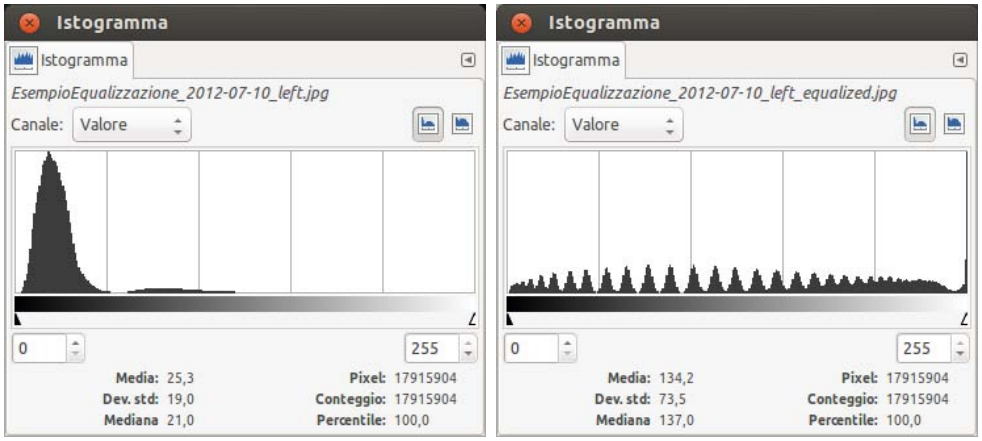

Figura 2.2: *Istogrammi delle immagini in Figura 2.1. Il secondo istogramma, relativo all'immagine equalizzata, ha un andamento quasi piatto.*

La procedura descritta è valida per immagini in scala di grigi ad un solo canale. Se l'immagine è a colori si potrebbe pensare di equalizzare ogni singolo canale R, G, B; in questo modo però si otterrebbe una notevole variazione del bilanciamento del colore. Se invece si trasforma l'immagine nello spazio di colore YCbCr, dove Y rappresenta la luminanza mentre Cb e Cr la crominanza, si può equalizzare il solo canale Y e poi ritornare allo spazio RGB. Così facendo il bilanciamento del colore resta invariato [21].

Alternative all'equalizzazione

La miglior tecnica capace di aumentare in modo efficace il contrasto è nota in letteratura con il termine *Retinex*, contrazione dei termini "retina" e "cortex", che sfrutta la costanza del colore percepita dall'occhio umano al variare della luminosità. Il filtro Retinex permette in una fase successiva di individuare molti più match tra una coppia di immagini rispetto all'equalizzazione. Tuttavia l'operazione richiede molto tempo per essere eseguita e non fornisce benefici rilevanti rispetto all'equalizzazione. Inoltre, quando si confrontano due immagini che riprendono la stessa scena scattate a qualche mese di distanza l'una dall'altra, il cambiamento tra di esse può essere molto marcato, perciò anche una tecnica valida come il Retinex potrebbe fallire e non individuare alcun match soddisfacente.

La Figura 2.3 mostra sulla sinistra tre immagini; quella più in alto rappresenta l'originale, quella in mezzo la stessa immagine a cui è stata applicata l'equalizzazione e l'ultima è l'immagine con filtro Retinex. Sulla destra è raffigurata una foto scattata il giorno successivo con le analoghe elaborazioni. In Figura 2.4 si può notare la diversa quantità di match individuati nei tre casi per la coppia di fotografie precedenti. Infine

la Tabella 2.1 contiene il numero di match, la distanza (norma 2) minima e media fra i relativi descrittori, e i tempi approssimativi necessari per elaborare un'immagine[1].

Figura 2.3: *Sulla colonna sinistra: confronto tra l'immagine originale, equalizzata e con filtro Retinex. Sulla colonna destra: analoghe elaborazioni per la foto scattata il giorno successivo.*

Negli ultimi anni sono stati proposti altri metodi raffinati con lo scopo di aumentare il contrasto dell'immagine e preservare la naturalità dei colori pur mantenendo una bassa complessità computazionale; ne sono un esempio le tecniche *DRSHE (Dynamic Range Separate Histogram Equalization)* [22], *Virtual Histogram Approach* [29] e *Regional Dynamic Histogram Equalization* [14].

[1]Non ho implementato il filtro Retinex in C++; il tempo indicato è quello impiegato dal programma di fotoritocco Gimp per applicare il filtro. I tempi sono stati rilevati su un computer dotato di processore Intel Core i5-3230M e sistema operativo Ubuntu Linux 13.04 64 bit.

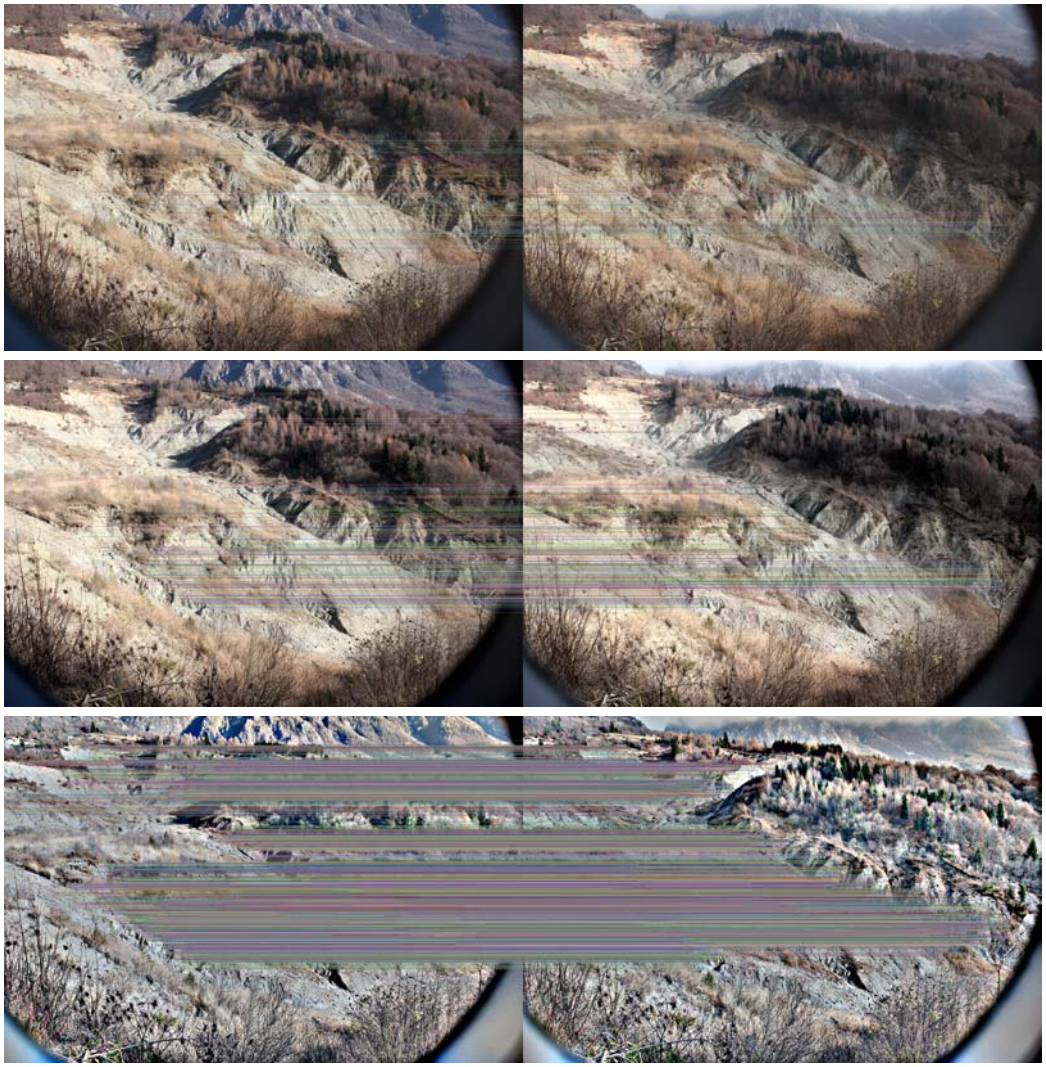

Figura 2.4: *Confronto del numero di match trovati per la coppia originale, equalizzata e con filtro Retinex; utilizzando queste ultime due tecniche la quantità di corrispondenze individuate è molto maggiore.*

Elaborazione	N. di match	Distanza minima	Distanza media	Tempo [ms]
Nessuna	111	47.4	140.3	0
Equalizzazione	593	35.1	126.4	243
Retinex	9720	26.9	134.1	$\simeq 27000$

Tabella 2.1: *Confronto del numero di match nel caso di nessuna elaborazione, equalizzazione e Retinex, distanza minima e media fra descrittori, e tempo approssimativo di elaborazione di una singola immagine.*

2.1.2 SIFT

Come si evince dall'articolo in cui è stato presentato [19], l'algoritmo *SIFT (Scale-invariant feature transform)* consente di estrarre dalle immagini delle caratteristiche (*feature*) distintive ed invarianti, in modo da individuare le corrispondenze (*match*) di oggetti o scene. Queste feature sono invarianti alla scala e alla rotazione, e permettono di effettuare il matching anche nel caso di distorsioni, cambio del punto di vista, rumore e variazioni di luminosità [26]. L'algoritmo consta di 4 passi fondamentali, che saranno ora approfonditi.

Rilevamento degli estremi

È evidente che per rilevare uno stesso oggetto in due immagini in cui è raffigurato con due dimensioni diverse non si può usare la stessa finestra: infatti, intuitivamente per individuare l'oggetto più grande bisognerebbe usare una finestra altrettanto grande [1].

Per determinare i contorni all'interno di un'immagine, ossia i punti di rapida variazione nell'intensità di colore, si può ricorrere al *Laplaciano* (divergenza del gradiente) *della funzione Gaussiana (LoG)*. Variare il parametro σ della gaussiana è equivalente a cambiare la scala dell'immagine. Per esempio, la convoluzione della foto con un kernel gaussiano avente parametro σ piccolo produce dei picchi per spigoli di dimensioni ridotte, mentre con un σ elevato si individuano spigoli grandi.

Operando in questo modo si possono trovare i massimi locali, ovvero i punti di variazione molto rapida, al variare della scala. Da un punto di vista pratico, l'elenco di punti chiave (*keypoint*) sarà contenuto in una lista di terne (x, y, σ), dove x e y costituiscono le coordinate del punto individuato mentre σ rappresenta la scala.

Purtroppo il calcolo del LoG è piuttosto costoso; per porre rimedio all'inconveniente l'algoritmo SIFT usa al suo posto le *Differenze di Gaussiane (DoG)*, che sono un'approssimazione dei LoG. Nella Figura 2.5 è visibile il modo in cui SIFT produce le Differenze di Gaussiane. A partire dalla foto originale si applica più volte la convoluzione con una gaussiana al fine di ottenere immagini distanziate di un fattore k nella scala dello spazio; queste immagini sono riportate sulla sinistra. Le immagini adiacenti vengono sottratte per ottenere le Differenze di Gaussiane, visibili sulla destra. Ogni volta che σ raddoppia si prende in considerazione una nuova *ottava* e l'immagine gaussiana viene sotto-campionata di un fattore 2; dopodiché il procedimento si ripete.

Una volta individuate le DoG, avviene la ricerca degli estremi locali al variare della scala e dello spazio. Un pixel di un'immagine viene confrontato con i suoi 8 punti adiacenti e con i 9 punti corrispondenti alle scale superiore ed inferiore (Figura 2.6). Se il pixel considerato è un estremo locale (minimo o massimo) allora rappresenta un potenziale keypoint in quella scala.

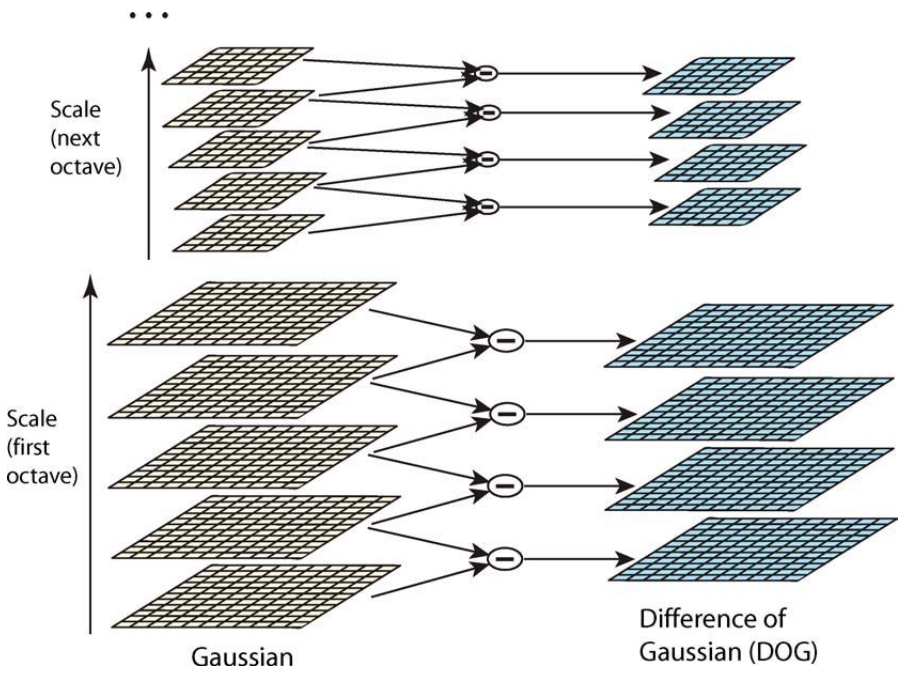

Figura 2.5: *Immagini gaussiane (a sinistra) e DoG ottenute a partire da esse (a destra). In alto è visibile l'ottava successiva.*

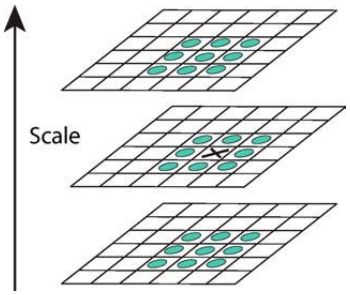

Figura 2.6: *Un pixel viene confrontato con i suoi 8 adiacenti e con i 9 delle scale superiore ed inferiore.*

Localizzazione dei keypoint

Una volta ottenuti i potenziali keypoint, questi devono essere ulteriormente raffinati. Usando un'espansione in serie di Taylor si individua in modo più preciso la posizione dell'estremo locale.

Affinché il punto sia considerato keypoint deve possedere un valore d'intensità superiore ad una soglia di contrasto minima. Inoltre, siccome le Differenze di Gaussiane presentano dei picchi in corrispondenza dei bordi degli oggetti anche quando questi sono irrilevanti perché sensibili al rumore, è necessario scartare questi punti di estremo. A tale scopo viene usata una tecnica simile a quella applicata da Harris nel suo algoritmo di ricerca degli angoli [8]. Usando una matrice hessiana si calcolano le curvature principali (gli autovalori dell'hessiano sono proporzionali alle curvature principali delle DoG). In corrispondenza dei bordi un autovalore è più grande dell'altro; imponendo che il loro rapporto sia inferiore ad una soglia prestabilita si possono eliminare questi punti indesiderati.

Riassumendo, SIFT rimuove i punti a basso contrasto e mantiene solo quelli di interesse scartando i bordi.

Assegnazione di un'orientazione

Per ottenere l'invarianza alla rotazione degli oggetti viene assegnata un'orientazione a ciascun keypoint. A seconda della scala si considera un opportuno intorno del keypoint e si calcolano intensità e direzione del gradiente in quella regione. Da queste quantità si ottiene un istogramma composto da 36 intervalli che coprono complessivamente 360 gradi. A questo punto si considerano il picco dell'istogramma e tutti i picchi superiori all'80% del picco massimo; così facendo si creano più keypoint dotati di stessa posizione e scala, ma di direzioni differenti. L'operazione ha lo scopo di migliorare la stabilità nella successiva fase di matching.

Creare i descrittori dei punti chiave

Per creare un *descrittore* si considera un intorno di 16×16 pixel attorno al keypoint scelto. L'intorno è suddiviso in 16 sotto-blocchi di 4×4 pixel, per ciascuno dei quali è creato un istogramma con 8 intervalli, per un totale di 128 intervalli. Inoltre sono state prese ulteriori misure per ottenere robustezza alle variazioni di illuminazione, rotazione, ecc.

Matching dei keypoint

L'operazione di *matching* consiste nel trovare coppie di keypoint corrispondenti all'interno di due immagini contenenti lo stesso oggetto. Per determinare i match candidati, per ciascun keypoint della seconda immagine si cerca il keypoint nella prima immagine il cui descrittore ha distanza euclidea minima dal descrittore del keypoint nella seconda immagine.

Tuttavia può succedere che a causa di occlusioni, rumore o altri motivi non sia presente un keypoint nella prima immagine corrispondente ad un keypoint nella seconda. Per questa ragione, oltre a considerare la distanza del match migliore, si osserva il secondo match più vicino. Si è verificato infatti che solitamente quando il match non è di buona qualità i due match hanno distanze molto simili tra loro. Se invece il rapporto tra le distanze dei match è inferiore ad una certa soglia (per esempio 0.8) il match è corretto. Questa constatazione si evince dal grafico in Figura 2.7, che riporta sulle ascisse il rapporto tra le distanze e sulle ordinate le densità di probabilità dei match corretti e dei match sbagliati. Si vede che oltre il valore di ascissa 0.8 la probabilità che un match sia buono è bassa e la probabilità che sia di bassa qualità è elevata. A questo metodo è stato dato il nome *ratio test*.

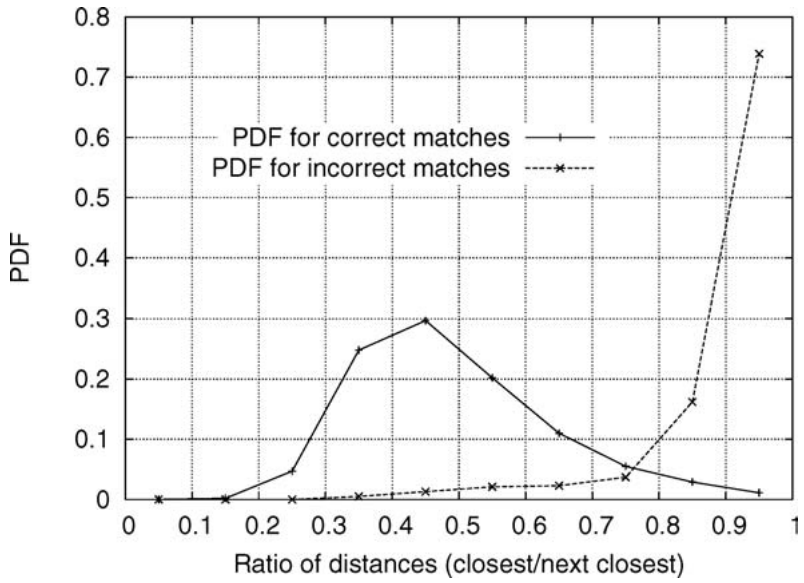

Figura 2.7: *Densità di probabilità nel caso di match corretti e sbagliati. Si vede che oltre il valore di ascissa 0.8 la probabilità che un match sia buono è bassa e la probabilità che sia di bassa qualità è elevata.*

2.1.3 Antidistorsione delle immagini

Omografia e correzione del punto di vista

In alcune situazioni si ha la necessità di antidistorcere, ovvero deformare, un'immagine in modo tale da simulare un cambiamento del punto di vista della fotocamera che ha scattato la scena. In generale questo non è possibile, se non nel caso in cui si vogliano rimappare solo i punti di un determinato piano o quando le due ipotetiche fotocamere condividono lo stesso pinhole[2].

[2] `http://www.ce.unipr.it/people/medici/geometry/node110.html`

Nonostante questo vincolo, è possibile effettuare un'antidistorsione accettabile servendosi di un'*omografia*. In matematica l'omografia è una relazione tra punti di due spazi tali per cui ogni punto di uno spazio corrisponde ad uno ed un solo punto del secondo spazio. Nell'ambito della visione artificiale i due spazi dai quali vengono prelevati i punti rappresentano l'immagine originale e quella antidistorta.

La trasformazione tra i punti dell'immagine originale e quelli dell'immagine antidistorta è espressa per mezzo di una matrice H di dimensioni 3×3 detta *matrice omografica*:

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \simeq H \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix},$$

dove $\begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T$ rappresenta un pixel dell'immagine da antidistorcere espresso in coordinate omogenee e $\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix}^T$ il relativo pixel nell'immagine antidistorta. La matrice si può ottenere qualora ci siano corrispondenze tra un totale di quattro punti a tre a tre non allineati [11].

Quando le corrispondenze di cui si dispone non sono tutte corrette, la matrice omografica che descrive la trasformazione tra le due foto potrebbe essere imprecisa. Ciò nonostante, il problema è eliminabile facendo ricorso ad opportuni algoritmi, come ad esempio RANSAC.

RANSAC

RANSAC (RANdom SAmple Consensus) è un metodo iterativo per la stima di parametri di un modello matematico a partire da un insieme di dati contenente outlier. È un algoritmo non deterministico, pertanto produce un risultato corretto solo con una certa probabilità, che aumenta al crescere delle iterazioni consentite [6]. Il set completo di dati da cui l'algoritmo costruisce il modello è composto da inlier, ossia dati la cui distribuzione caratterizza in modo appropriato il modello, e outlier, ovvero dati che non rappresentano il modello. Gli outlier possono essere dovuti a rumore, misure scorrette, ecc.

A differenza del metodo dei minimi quadrati, RANSAC crea il modello a partire da un sottoinsieme casuale dei dati, che considera inlier; tutti gli altri dati vengono testati sul modello stimato e, se un punto rientra in tale modello, viene considerato anch'esso un inlier ipotetico. Il modello stimato è ritenuto buono se consente di produrre una quantità sufficiente di inlier ipotetici. In tal caso il modello viene nuovamente stimato su tutti gli inlier ipotetici, non solo su quelli iniziali. Infine la bontà del modello viene valutata attraverso una stima dell'errore che si commette associando agli inlier ipotetici il modello. Queste operazioni vengono ripetute ad ogni iterazione e producono di volta in volta un modello che può essere rifiutato a causa del basso numero di punti classificati come inlier, o un modello corretto assieme ad una corrispondente misura dell'errore.

Un esempio di retta (il modello matematico) stimata con quest'algoritmo, in cui sono visibili inlier ed outlier, è riportato in Figura 2.8[3].

[3] `http://commons.wikimedia.org/wiki/File:Fitted_line.svg`

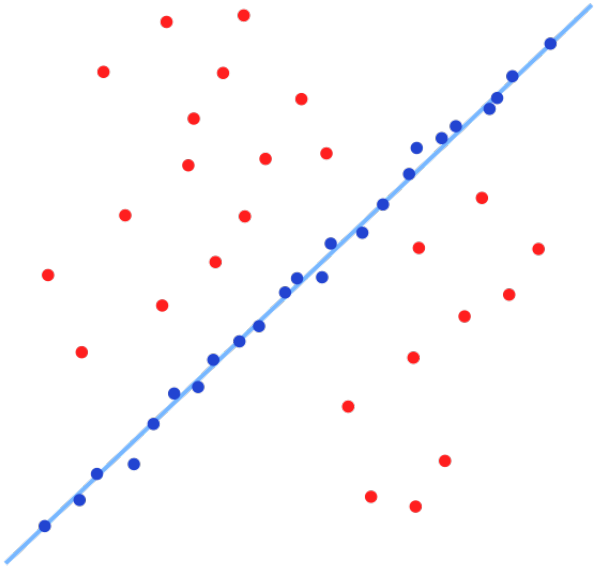

Figura 2.8: *Esempio di applicazione dell'algoritmo RANSAC. I punti blu sono gli inlier, mentre quelli rossi sono gli outlier.*

2.2 Template-based matching

Si è già detto che il template matching può essere realizzato osservando due filosofie distinte: feature-based template matching, di cui si è parlato nel paragrafo 2.1, e template-based matching. Come nell'approccio feature-based, anche nel *template-based matching* lo scopo è identificare una porzione di immagine, detta *template*, all'interno di un'altra immagine contenente un oggetto simile.

Per svolgere questa attività, il template, avente di solito forma quadrata o rettangolare, viene fatto scorrere sopra l'immagine in cui bisogna cercare l'oggetto rappresentato dal template. Il template, che d'ora in poi identifico con $T(x', y')$, viene spostato pixel per pixel finché tutta l'immagine $I(x + x', y + y')$ viene coperta; dopodiché si associa ad ogni pixel il valore di un'opportuna funzione che valuta la bontà del match. Se il template T ha dimensioni $n' \times m'$ e l'immagine I ha dimensioni $n \times m$, allora l'immagine R risultante dallo scorrimento di T sopra I ha dimensioni $(n - n' + 1) \times (m - m' + 1)$. Il miglior match si trova in corrispondenza del punto (x_i, y_i) avente valore massimo o minimo, a seconda della funzione impiegata.

2.2.1 Valutare la bontà di un match

Il valore associato ad ogni pixel della suddetta matrice risultante R è calcolato a partire da un'opportuna funzione. La funzione più intuitiva è nota con l'acronimo *SAD*

(Sum of Absolute Differences):

$$R(x,y) = \sum_{x',y'} |T(x',y') - I(x+x',y+y')|,$$

dove, per precisione, $0 \leq x < n,\ 0 \leq y < m,\ 0 \leq x < n' - n,\ 0 \leq y < m' - m$. Come dice il nome, tale funzione calcola le differenze in valore assoluto per ciascuna coppia di pixel (presi da T e I) e ne effettua la somma; chiaramente il miglior match si ha in corrispondenza del valore minimo assunto dalla funzione. Uno degli inconvenienti di questa funzione è quello di essere sensibile ad un'eventuale variazione di luminosità tra l'oggetto contenuto nel template e l'oggetto contenuto nell'immagine in cui deve essere cercato.

Un altro metodo simile a SAD e che presenta lo stesso inconveniente è *SSD (Sum of Squared Differences)*, in cui al posto di calcolare il valore assoluto si eleva al quadrato [1]:

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2. \tag{2.1}$$

Una terza tecnica, spesso utilizzata, è la *cross-correlazione*, detta anche correlazione incrociata o correlazione mutua, operazione simile alla convoluzione che misura la somiglianza tra due forme d'onda. Nel campo discreto e in una dimensione, la cross-correlazione si definisce in questo modo [18]:

$$(f \star g)[n] = \sum_{m=-\infty}^{+\infty} f^*[m]g[m+n],$$

dove f^* è il complesso coniugato di f. In due dimensioni, come nel caso delle immagini, e impiegando la stessa simbologia di prima, si può esprimere così:

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y')). \tag{2.2}$$

Tuttavia nemmeno l'uso di questa formula garantisce risultati soddisfacenti.

Per ottenere un riconoscimento ottimale del miglior match bisogna ricorrere alla versione normalizzata della cross-correlazione. Ad ogni passo viene sottratto il valore di luminosità media e si divide per la deviazione standard dell'immagine; così facendo il matching diventa invariante alla luminosità. La cross-correlazione normalizzata è equivalente ad un prodotto scalare tra vettori normalizzati:

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}, \tag{2.3}$$

dove

$$T'(x', y') = T(x', y') - \frac{1}{n' \cdot m'} \sum_{x'', y''} T(x'', y''),$$

$$I'(x + x', y + y') = I(x + x', y + y') - \frac{1}{n \cdot m} \sum_{x'', y''} I(x + x'', y + y'').$$

Poiché il template-based matching è un'operazione alquanto onerosa, nel tempo sono stati introdotti alcuni accorgimenti che permettono una riduzione del costo computazionale. Un esempio sono le immagini piramidali, copie dell'immagine originale filtrate in frequenza e sottocampionate nello spazio: le immagini a bassa risoluzione posso essere usate per una ricerca grossolana del template nell'immagine, mentre quelle ad alta risoluzione si adoperano per una ricerca locale più fine della posizione esatta del match. Se la zona in cui cercare il miglior match è già nota si può anche circoscrivere il template matching solo a quell'area.

Capitolo 3

Richiami sulla ricostruzione 3D

La ricostruzione in tre dimensioni di una scena può essere ottenuta seguendo svariati procedimenti, la cui scelta dipende dalla strumentazione di cui si dispone, dalla modalità con cui l'oggetto viene ripreso e dalla natura dell'oggetto stesso. I sistemi di acquisizione delle immagini possono assumere varie configurazioni: esistono sistemi stereoscopici formati da una coppia di fotocamere, sistemi dotati di un numero considerevole di macchine fotografiche disposte attorno alla scena e sistemi che fanno uso di una sola telecamera.

A prescindere dal sistema utilizzato, l'attività di ricostruzione deve essere preceduta da un'analisi di tipo geometrico e algebrico. In particolare è opportuno formulare un modello per la fotocamera al fine di ottenerne la calibrazione (paragrafo 3.1); tale modello si può estendere al caso in venga impiegata una coppia di fotocamere (paragrafo 3.2). Sistemi stereoscopici e sistemi che si possono ricondurre ad essi trovano notevoli applicazioni nel campo della robotica (paragrafo 3.3).

3.1 Modello di una fotocamera

La prima operazione da svolgere quando si vogliono ricavare informazioni di tipo geometrico dalle immagini è la creazione di un modello che descriva in modo semplice e completo il funzionamento di una fotocamera. La rappresentazione più comune di una macchina fotografica è nota come *modello pinhole*.

3.1.1 Modello geometrico

Questo modello approssimato descrive la relazione tra un punto del mondo reale e la sua proiezione sul piano immagine; la fotocamera prevista dal modello non è dotata di obiettivo, bensì di un foro attraverso il quale passano i raggi di luce. Da un punto di vista geometrico, questa fotocamera semplificata è composta da un *asse ottico*, che interseca il *piano immagine* nel *punto principale*; il punto in cui è posto il pinhole è detto *centro ottico*. Per evitare che l'oggetto proiettato sul piano immagine sia rovesciato, spesso si pone quest'ultimo davanti al pinhole, alla stessa distanza f (*lunghezza focale*) [7]. Un

punto Q nello spazio viene proiettato nel punto q del piano immagine. Quanto esposto è illustrato in Figura 3.1.

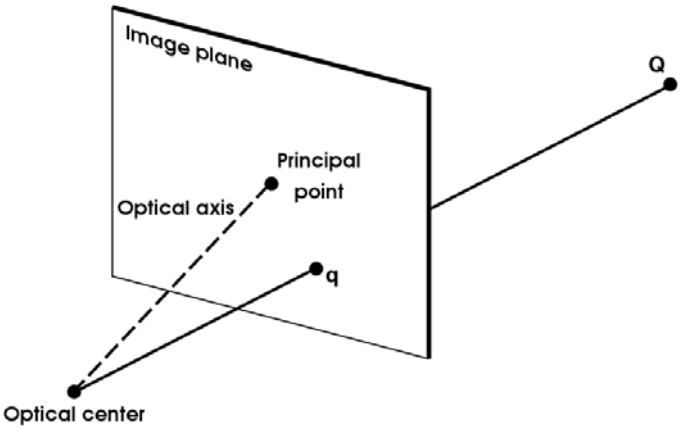

Figura 3.1: *Modello pinhole.*

3.1.2 Modello algebrico

Per poter fare considerazioni matematiche sul modello appena descritto, è necessario rappresentarlo in modo algebrico; a tale scopo si fa uso di quattro sistemi di riferimento.

Innanzitutto definisco il sistema di riferimento 3D solidale alla fotocamera, avente origine in corrispondenza del centro ottico, asse Z coincidente con l'asse ottico e assi X e Y paralleli al piano immagine, perpendicolari tra loro ed allineati alla griglia del sensore della fotocamera [25]. Il secondo sistema di riferimento è quello dell'immagine, avente origine nel punto principale e assi x e y paralleli agli assi X e Y. Si veda la Figura 3.2. Dato un punto $Q^C = \begin{bmatrix} X^C & Y^C & Z^C & 1 \end{bmatrix}^T$ espresso in coordinate omogenee nel sistema di riferimento della fotocamera, sfruttando relazioni di similitudine fra triangoli si ottiene il punto q corrispondente, che nel sistema di riferimento del piano immagine, ha coordinate:

$$x = f\frac{X^C}{Z^C},$$
$$y = f\frac{Y^C}{Z^C}.$$

In coordinate omogenee:

$$q = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X^C \\ Y^C \\ Z^C \\ 1 \end{bmatrix}. \tag{3.1}$$

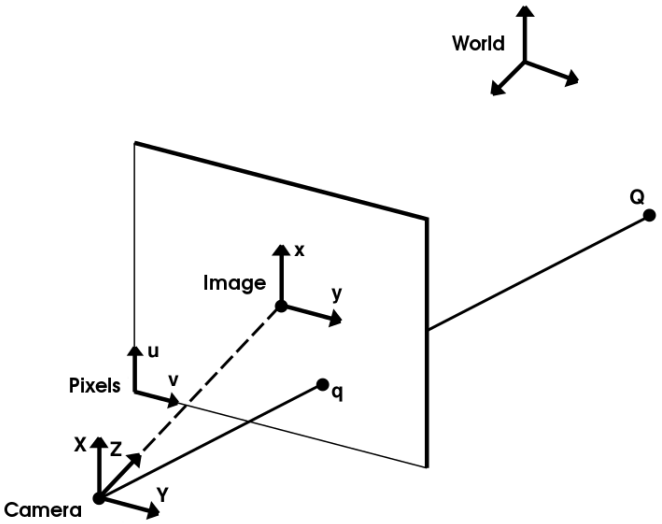

Figura 3.2: *Modello pinhole e sistemi di riferimento.*

Nelle equazioni precedenti ho usato un sistema metrico per esprimere le coordinate di un punto nel piano immagine. Tuttavia il sensore di una fotocamera digitale è formato da una griglia di pixel; pertanto è opportuno effettuare un altro cambiamento di sistema di riferimento (definisco u e v gli assi del nuovo sistema in pixel, come in Figura 3.2). Siano $-x_0$ e $-y_0$ le coordinate dell'angolo in basso a sinistra del sensore nel sistema di riferimento del piano immagine, e siano k_u e k_v le densità dei pixel nelle due direzioni. Le nuove coordinate di q sono ottenute mediante una traslazione ed un cambio di unità:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3.2}$$

Combinando le equazioni 3.1 e 3.2 si ottiene la relazione tra coordinate 3D rispetto alla fotocamera e sistema 2D espresso in pixel:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X^C \\ Y^C \\ Z^C \\ 1 \end{bmatrix}. \tag{3.3}$$

Siccome la fotocamera è libera di muoversi è necessario ricorrere ad un ulteriore sistema di riferimento esterno fisso, chiamato sistema di riferimento del mondo. La posizione e l'orientazione della macchina fotografica rispetto al mondo soddisfano la seguente

relazione (l'apice ^{m} denota una coordinata nel mondo):

$$\begin{bmatrix} X^C \\ Y^C \\ Z^C \end{bmatrix} = \begin{bmatrix} R & -Rt \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \begin{bmatrix} X^m \\ Y^m \\ Z^m \\ 1 \end{bmatrix} . \tag{3.4}$$

Per concludere riporto il modello algebrico completo, ottenuto combinando le equazioni 3.3 e 3.4:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R & -Rt \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \begin{bmatrix} X^m \\ Y^m \\ Z^m \\ 1 \end{bmatrix} . \tag{3.5}$$

Infine si definiscono:

$$K = \begin{bmatrix} k_u f & 0 & k_u x_0 \\ 0 & k_v f & k_v y_0 \\ 0 & 0 & 1 \end{bmatrix} , \tag{3.6}$$

detta *calibration matrix* e

$$P \sim \begin{bmatrix} K & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} R & -Rt \\ \mathbf{0}^T & 1 \end{bmatrix} ,$$

matrice 3×4 chiamata *matrice di proiezione*, che pone in relazione un punto 3D nel mondo con un pixel del sensore della fotocamera.

3.1.3 Parametri e calibrazione di una fotocamera

I parametri contenuti nell'equazione 3.5 si possono distinguere in due tipologie: *parametri intrinseci* e *parametri estrinseci*. I primi esprimono le proprietà della macchina fotografica e sono contenuti nella calibration matrix K (f, k_u, k_v, x_0 e y_0). Quelli estrinseci sono la matrice di rotazione R ed il vettore traslazione t che rappresentano orientazione e posizione della fotocamera rispetto al mondo.

L'obiettivo della *calibrazione* di una fotocamera è ricavare i parametri intrinseci, ovvero quelli propri della macchina fotografica impiegata. Tuttavia, siccome non è facile disaccoppiare parametri intrinseci ed estrinseci, solitamente vengono calcolati entrambi. Queste informazioni possono essere ottenute servendosi di motivi, o pattern, la cui configurazione geometrica è nota. Sfruttando le dimensioni conosciute di tale modello e le coordinate dei pixel in cui vengono proiettati gli elementi del pattern, facendo un po' di considerazioni matematiche è possibile ricavare i parametri cercati. Per approfondire la trattazione si faccia riferimento a [25].

3.2 Immagini stereoscopiche

Quando si dispone di almeno due foto che raffigurano lo stesso oggetto, è possibile comporre una scena in tre dimensioni. Solitamente le foto sono ottenute per mezzo di una coppia di fotocamere opportunamente collocate.

3.2.1 Cenni di geometria epipolare

Come già fatto con il modello pinhole per estrarre informazioni geometriche da un'immagine, anche quando si ha a che fare con due fotocamere è indispensabile introdurre un modello [4].

Suppongo allora che le macchine fotografiche siano poste una di fianco all'altra, con i relativi piani immagine perfettamente complanari, assi ottici paralleli e lunghezze focali $f_l = f_r$. Siano inoltre le immagini provenienti dalle macchine allineate riga per riga. Dato un punto P nello spazio, questo viene proiettato nei punti p_l e p_r sui piani immagine, con ascisse x^l e x^r. La configurazione descritta è riportata in Figura 3.3.

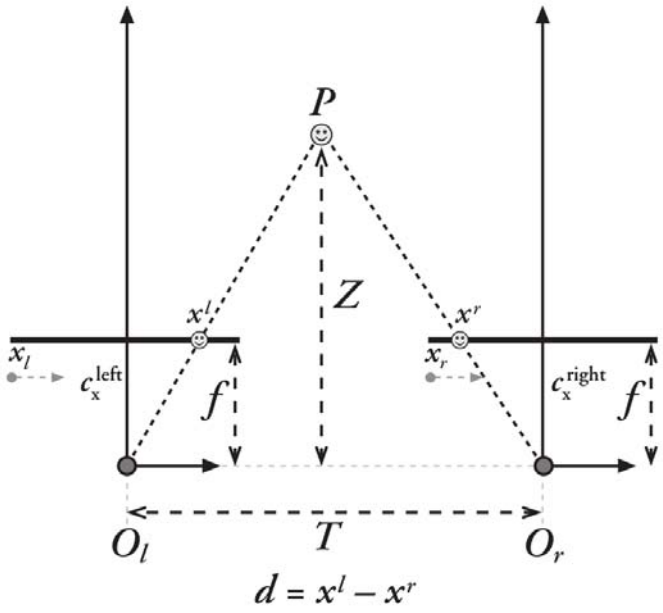

Figura 3.3: *Configurazione semplificata di una stereocamera.*

Definendo la quantità $d = x^l - x^r$, detta *disparità*, si nota che d è inversamente proporzionale alla profondità Z di P. Più precisamente, la seguente relazione di similitudine esprime il rapporto tra disparità e profondità:

$$\frac{T - (x^l - x^r)}{Z - f} = \frac{T}{Z} \Rightarrow Z = \frac{fT}{x^l - x^r}.$$

Come si vede, questa particolare disposizione delle fotocamere semplifica notevolmente il calcolo delle profondità a partire dai valori di disparità. Purtroppo tale configurazione non si verifica mai nella pratica; è quindi utile ricondursi al caso facile appena descritto allineando artificialmente le due macchine sfruttando la matematica.

La geometria di una coppia di fotocamere è detta *geometria epipolare*; essa è formata da un modello pinhole per ciascuna macchina e introduce la nozione di *epipolo*.

Dati una configurazione qualsiasi delle due fotocamere e un punto P nello spazio, si ha che questo viene proiettato nei punti p_l e p_r dei piani immagine. Un epipolo e_l sul piano immagine sinistro è l'immagine del centro di proiezione O_r dell'altra fotocamera (lo stesso dicasi per l'epipolo destro e_r). Si veda la Figura 3.4.

Il piano in cui giacciono P, e_l ed e_r è chiamato *piano epipolare*, e le linee che congiungono p_l con e_l e p_r con e_r sono le *linee epipolari*.

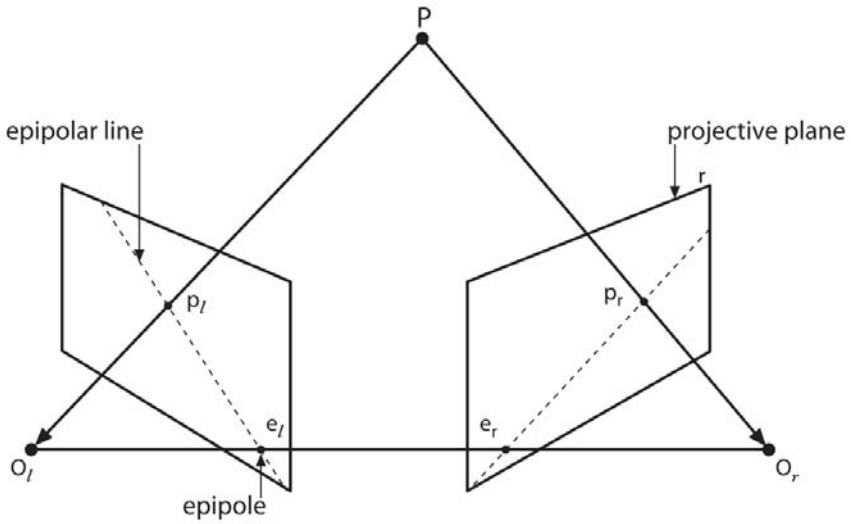

Figura 3.4: *Geometria epipolare di una configurazione qualsiasi.*

Per capire l'importanza degli epipoli, ricordo che un punto p_r sul piano immagine destro è la proiezione di infiniti punti giacenti sulla retta definita da p_r ed O_r. La proiezione di questi punti sul piano immagine sinistro è contenuta proprio nella linea epipolare sinistra (cioè la retta che congiunge p_l ed e_l). Pertanto, se si vuole cercare il punto P nel piano immagine sinistro, basta limitare la ricerca ai punti che compongono la linea epipolare. Si noti come questo procedimento sia riconducibile intuitivamente al caso in cui i due piani immagine erano allineati riga per riga nella configurazione semplificata.

3.2.2 Calibrazione stereo e ricostruzione 3D

La *calibrazione stereo* consente di ricavare le matrici di rotazione e traslazione tra una coppia di fotocamere disposte in modo qualsiasi. In altre parole, la calibrazione

rende i piani immagine delle due macchine complanari. La *stereo rettificazione* è invece il processo che allinea le immagini prodotte riga per riga.

La calibrazione stereo si può compiere ancora una volta ricorrendo ad un pattern prestabilito (per esempio una scacchiera) fotografato da vari punti di vista. Quando sono note la rotazione e la traslazione tra le fotocamere si può usare l'algoritmo di *Bouguet* per la rettificazione [3]; questo minimizza l'errore di riproiezione della coppia di immagini e ne massimizza l'area in comune.

Una volta completate queste due operazioni bisogna individuare le corrispondenze sulla stessa riga nella coppia di immagini; in pratica, dato un pixel dell'immagine sinistra collocato ad una certa riga, si cerca il pixel che rappresenta il medesimo elemento nell'immagine destra, sempre sulla stessa riga. Questo si può realizzare tramite un algoritmo appropriato, come ad esempio *SGBM (Semi-global block matching)* [1]. L'output di SGBM è una *mappa di disparità*, che associa ad ogni pixel dell'immagine originale la sua disparità, e quindi anche la profondità, data la relazione di proporzionalità inversa tra queste due quantità. Quando si possiede la mappa di disparità si può modellare la scena tridimensionale mediante un procedimento di *riproiezione*.

3.3 Structure From Motion

Nell'ambito della robotica, i sistemi di visione sono di fondamentale importanza al fine di ricostruire l'ambiente all'interno del quale un robot si muove. Questo processo è noto in letteratura come *SLAM (Simultaneous Localization And Mapping)* e consiste nella costruzione di una mappa di un ambiente sconosciuto e nel tener traccia della posizione rispetto ad essa. SLAM può essere realizzato servendosi di molte tipologie di sensori: laser rangefinders, lidar 3D e sonar; tuttavia la diffusione di sensori digitali negli ultimi anni ha incentivato l'uso delle fotocamere a tale scopo [20]. In questo contesto si è sviluppata la tecnica chiamata *Structure From Motion*, il cui intento è ricostruire la scena tridimensionale a partire da una sequenza di immagini scattate durante il movimento del robot.

3.3.1 Ricostruzione 3D senza parametri intrinseci

La quantità di sensori di cui è dotato un robot varia considerevolmente a seconda delle sue dimensioni e delle finalità per cui è stato progettato. Da un punto di vista teorico, affinché il robot possa "interpretare" lo spazio che lo circonda e muoversi al suo interno evitando gli ostacoli, è sufficiente l'ausilio di una sola telecamera. In questo caso, si può ricostruire la scena tridimensionale simulando un sistema a due o più fotocamere spostando la telecamera nel tempo; infatti, l'azione di inquadrare un oggetto fermo con una sola macchina fotografica scattando due foto da punti di vista diversi si riconduce al caso di inquadrare lo stesso oggetto con due macchine diverse, questione già affrontata nel paragrafo 3.2. Pertanto le analisi già svolte si possono applicare anche alla situazione in cui si utilizza una sola telecamera. Tuttavia, poiché non si dispone di una coppia di

telecamere aventi posizione reciproca fissata, non è possibile eseguire una calibrazione stereo standard; bisogna quindi ricorrere ad altri stratagemmi.

Matrice fondamentale

La *matrice fondamentale* F è una matrice 3×3 utilizzata nell'ambito della visione stereoscopica. In geometria epipolare, data una coppia di punti corrispondenti x e x' presi da una coppia di immagini, il prodotto Fx rappresenta una linea epipolare; vale quindi la seguente relazione [11]:

$$x'^T F x = 0. \tag{3.7}$$

In altre parole, questa matrice impone un vincolo, detto vincolo epipolare, sulla proiezione dei punti della scena sui piani immagine. La matrice fondamentale si può ricavare da almeno 7 corrispondenze; tuttavia, quando le posizioni dei punti sono affette da rumore, è preferibile ricorrere ad un algoritmo robusto come RANSAC (paragrafo 2.1.3).

Algoritmo di Hartley

Nel paragrafo 3.2.2 si è detto che la rettificazione di una coppia di immagini può essere ottenuta eseguendo l'algoritmo di Bouguet quando si è a conoscenza della rototraslazione tra le due telecamere. Quando questo dato, ricavabile con la calibrazione stereo, non è noto, si può ricorrere all'algoritmo di *Hartley* [9].

Questo algoritmo permette di ricostruire una scena tridimensionale semplicemente a partire dalla matrice fondamentale F e da una serie di corrispondenze tra i punti della coppia di immagini. L'algoritmo di Hartley presenta tuttavia due svantaggi [4]:

- Non si conosce la scala dell'immagine: non si può cioè capire se l'oggetto rappresentato è di grandi dimensioni e si trova in posizione ravvicinata, o se è piccolo e lontano dalla telecamera.
- Non conoscendo i parametri intrinseci, l'oggetto può essere ricostruito a meno di una trasformazione prospettica; infatti, a oggetti di forme e dimensioni diverse possono corrispondere punti sulle immagini aventi stesse coordinate 2D.

Le situazioni descritte sono visibili in Figura 3.5.

Dopo aver eseguito la rettificazione stereo con l'algoritmo di Hartley si può procedere come di consueto con la procedura standard: identificazione delle corrispondenze sulle stesse righe, calcolo della matrice di disparità e riproiezione.

3.3.2 Ricostruzione 3D con parametri intrinseci

Per sopperire agli inconvenienti della ricostruzione usando l'algoritmo di Hartley, si possono fare ipotesi sui parametri intrinseci della fotocamera e sfruttare altre tecniche che si servono di queste informazioni al fine di ottenere una scena 3D più accurata.

Nel paragrafo 3.1.3 si è detto che i parametri intrinseci di una fotocamera sono quelli contenuti nella matrice di calibrazione K (equazione 3.6). Quando non si dispone di

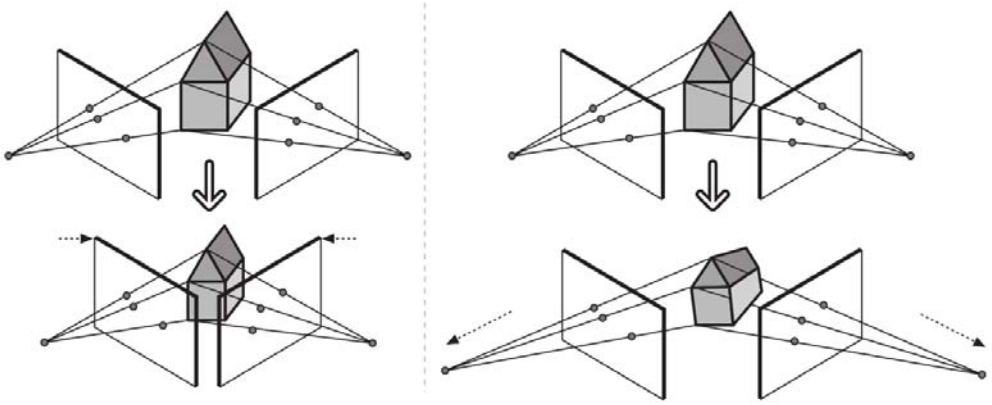

Figura 3.5: *Errori di ricostruzione che può commettere l'algoritmo di Hartley: la scala non è nota e l'oggetto è ricostruito a meno di una trasformazione prospettica.*

questa matrice, è comunque possibile ipotizzarne il contenuto qualora si conoscano le specifiche della fotocamera e dell'obiettivo impiegati: è sufficiente sapere le dimensioni metriche del sensore, la sua risoluzione e la lunghezza focale dell'obiettivo. La matrice K così ricavata non è esente da imprecisioni (nelle fotocamere reali l'asse ottico dell'obiettivo non passa mai esattamente per il punto centrale del sensore), ma è abbastanza accurata da permettere un'affidabile ricostruzione tridimensionale a partire dai valori in essa contenuti.

Matrice di proiezione

La *matrice di proiezione*, o *camera matrix*, P è la matrice 3×4 che lega le coordinate 3D nel sistema di riferimento del mondo alle coordinate 2D sul piano immagine. Dato un punto tridimensionale X espresso in coordinate omogenee e un punto bidimensionale x in coordinate omogenee che rappresenta la proiezione di X sul piano immagine, vale la relazione:

$$x \sim PX.$$

Il simbolo $\sim$ significa che l'uguaglianza è vera a meno di una costante.

Si può dimostrare che una coppia di matrici di proiezione (P, P') (una per ogni fotocamera) determina univocamente una matrice fondamentale F; viceversa, una matrice fondamentale determina una coppia di matrici di proiezione a meno di una moltiplicazione a destra per una matrice di trasformazione H. Detto in altre parole, alle coppie di matrici di proiezione (P, P') e $(PH, P'H)$ corrisponde la stessa matrice fondamentale F [11]. Sfruttando quest'ambiguità, si può far ricorso alla forma canonica della coppia di matrici di proiezione, in cui la prima matrice è $P = [I \mid \mathbf{0}]$.

Siano $P = K[R \mid t]$ una matrice di proiezione, con K matrice di calibrazione, e $x = PX$ un punto dell'immagine. Se K è nota, allora si può moltiplicare x per la sua

matrice inversa K^{-1} ottenendo $\hat{x} = K^{-1}x = [R \mid t]\ X$, dove $\hat{x}$ è il punto x espresso in *coordinate normalizzate*. Questa operazione è equivalente a prendere l'immagine di X rispetto alla matrice di proiezione $[R \mid t]$, avente la matrice identità come matrice di calibrazione. $K^{-1}P = [R \mid t]$ è chiamata *matrice di proiezione normalizzata*, e non contiene informazioni sulla matrice di calibrazione.

Ora, data la coppia di matrici normalizzate $P = [I \mid \mathbf{0}]$ e $P' = [R \mid t]$, la matrice fondamentale ad esse associata è chiamata *matrice essenziale*.

Matrice essenziale

Come la matrice fondamentale (paragrafo 3.3.1), anche la *matrice essenziale* E è una matrice 3×3 che mette in relazione le corrispondenze in una coppia di immagini stereoscopiche. Dalle argomentazioni del paragrafo precedente si desume che, data una coppia di punti corrispondenti con coordinate normalizzate $\hat{x}$ e $\hat{x}'$, vale la relazione:

$$\hat{x}'^T E \hat{x} = 0. \tag{3.8}$$

Sostituendo ad $\hat{x}$ e $\hat{x}'$ le relative definizioni e ricordando l'equazione 3.7, la formula 3.8 diventa $x'^T K'^{-T} E K^{-1} x = 0$, da cui la relazione:

$$E = K'^T F K. \tag{3.9}$$

Ricavare la matrice di proiezione dalla matrice essenziale

La *decomposizione ai valori singolari (SVD)* è una fattorizzazione di una matrice M di dimensioni $m \times n$ della forma $M = U\Sigma V^*$, dove U è una matrice unitaria di dimensioni $m \times m$, Σ è una matrice diagonale di dimensioni $m \times n$ i cui valori sulla diagonale sono chiamati *valori singolari* e V^* è la trasposta coniugata di una matrice unitaria V di dimensioni $n \times n$.

Si dimostra che, date la scomposizione SVD (a meno della scala) della matrice essenziale $E = U\ diag(1, 1, 0)\ V^T$ e la matrice di proiezione $P = [I \mid \mathbf{0}]$, la matrice di proiezione P' relativa all'altra fotocamera può assumere quattro configurazioni distinte:

$$\begin{aligned} P' &= [UWV^T \mid +u_3], \\ P' &= [UWV^T \mid -u_3], \\ P' &= [UW^TV^T \mid +u_3], \\ P' &= [UW^TV^T \mid -u_3], \end{aligned}$$

con

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

Nella pratica solo una di queste quattro soluzioni si può riscontrare, perché le altre tre producono punti 3D che si trovano dietro, non davanti, almeno una delle due fotocamere,

come si vede in Figura 3.6. Questa condizione si può verificare imponendo che la coordinata x_3 di un punto (x_1, x_2, x_3) relativo al sistema di riferimento della telecamera sia positiva.

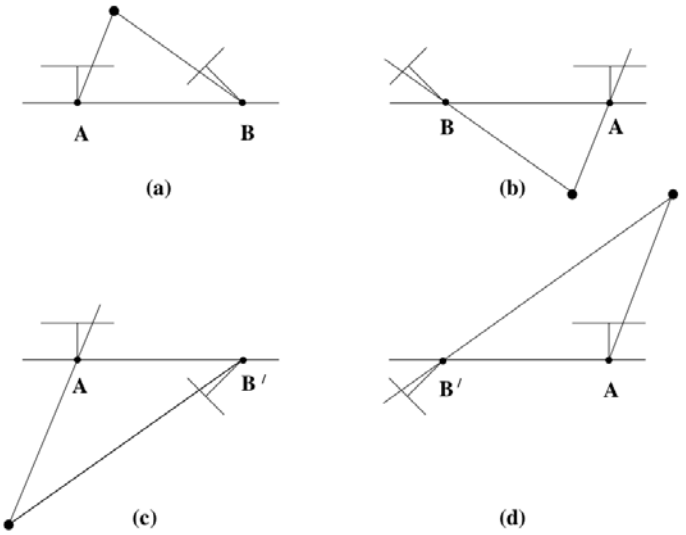

Figura 3.6: *Configurazioni possibili delle due fotocamere; solo la situazione (a) può verificarsi nella realtà.*

Infine bisogna anche controllare se $R = UWV^T$ (o UW^TV^T) è effettivamente una matrice di rotazione, ovvero se $\det(R) = +1$; nel caso in cui $\det(R) = -1$ basta cambiare il segno di E.

Triangolazione

Il problema della *triangolazione* consiste nel determinare le coordinate di un punto nello spazio 3D date le sue proiezioni sui piani immagine di due o più telecamere. Quando si conoscono le matrici di proiezione, la triangolazione è una questione apparentemente banale, riconducendosi al problema di trovare l'intersezione tra due rette. Tuttavia, quando è presente del rumore, le coordinate delle proiezioni sui piani immagine non sono precise; di conseguenza le rette che partono dai punti sulle immagini non si intersecano tra loro. In questo caso è necessario riformulare il problema in un altro problema di minimizzazione ai minimi quadrati. Esistono vari metodi che consentono di minimizzare l'errore commesso; uno dei più comuni, anche se non molto efficace, è la *triangolazione lineare ai minimi quadrati* [10].

Dati i punti corrispondenti sui piani immagine x e x' e le relative matrici di proiezione

P e P', bisogna calcolare le coordinate del punto X nello spazio tali che:

$$\begin{aligned} x &= PX, \\ x' &= P'X, \end{aligned} \tag{3.10}$$

dove $x = w(u, v, 1)^T$ e $x' = w'(u', v', 1)^T$, con w e w' fattori di scala non noti, e (u, v) e (u', v') coordinate osservate sulle immagini. Si indichino con ${p_i}^T$ e ${p_i}'^T$ le righe delle due matrici di proiezione. Allora l'equazione 3.10 si può riscrivere:

$$\begin{cases} wu = p_1^T X, \\ wv = p_2^T X, \\ w = p_3^T X, \end{cases}$$

e sostituendo la terza equazione nelle altre due:

$$\begin{cases} up_3^T X = p_1^T X, \\ vp_3^T X = p_2^T X. \end{cases}$$

Considerando entrambe le fotocamere si ottiene un sistema di quattro equazioni:

$$\begin{cases} (up_3^T - p_1^T)X = 0, \\ (vp_3^T - p_2^T)X = 0, \\ (u'p'^T_3 - p'^T_1)X = 0, \\ (v'p'^T_3 - p'^T_2)X = 0. \end{cases}$$

Scomponendo P e P' nelle loro componenti e riscrivendo in coordinate omogenee $X = \begin{bmatrix} U & V & W & 1 \end{bmatrix}^T$, si ricava il sistema non omogeneo finale della forma $AX = B$:

$$\begin{bmatrix} up_{31} - p_{11} & up_{32} - p_{12} & up_{33} - p_{13} \\ vp_{31} - p_{21} & vp_{32} - p_{22} & vp_{33} - p_{23} \\ u'p'_{31} - p'_{11} & u'p'_{32} - p'_{12} & u'p'_{33} - p'_{13} \\ v'p'_{31} - p'_{21} & v'p'_{32} - p'_{22} & v'p'_{33} - p'_{23} \end{bmatrix} \cdot X = \begin{bmatrix} -up_{34} + p_{14} \\ -vp_{34} + p_{24} \\ -u'p'_{34} + p'_{14} \\ -v'p'_{34} + p'_{24} \end{bmatrix}.$$

Una soluzione al problema dei minimi quadrati può essere trovata sfruttando la matrice pseudo-inversa o una decomposizione ai valori singolari.

Sovracampionamento con il metodo moving least squares

Il metodo esposto nel paragrafo 3.3.2 si basa sulla stima dei parametri intrinseci in modo da rendere più affidabile la ricostruzione 3D. Tuttavia, questo modo di procedere presenta uno svantaggio: la triangolazione impiegata per determinare le coordinate 3D di un punto si può applicare solo a punti corrispondenti nelle due immagini. Se, dato un punto bidimensionale x nell'immagine sinistra, non è noto il suo corrispondente x'

nell'immagine destra, non si può applicare la triangolazione e quindi non si possono ricavare le coordinate del punto nella scena reale. Pertanto la scena ricostruita può essere incompleta se non si possiede una quantità sufficiente di corrispondenze tra le immagini.

In questa situazione può essere utile aumentare la densità della nuvola di punti 3D ricostruita effettuando un sovracampionamento. Questo può essere realizzato, per esempio, sfruttando il metodo *moving least squares*. Questo metodo permette di trovare funzioni continue a partire da un insieme di punti sparsi sfruttando tecniche ai minimi quadrati pesati [17].

Dati una funzione $f : \mathbb{R}^n \to \mathbb{R}$ ed un insieme di punti $S = \{(x_i, f_i) | f(x_i) = f_i\}$, dove $x_i \in \mathbb{R}^n$ e $f_i \in \mathbb{R}$, l'approssimazione moving least squares di grado m attorno al punto x è $\tilde{p}(x)$, dove $\tilde{p}$ minimizza l'errore ai minimi quadrati pesato:

$$\sum_{i \in I} (p(x) - f_i)^2 \theta(||x - x_i||)$$

su tutti i polinomi p di grado m in $\mathbb{R}^n$; $\theta(s)$ è il peso e tende a 0 per $s \to \infty$. In altre parole, questa tecnica consente di stimare una superficie (funzione in due variabili nel caso di oggetti 3D) nell'intorno locale di ciascun punto.

Capitolo 4

Realizzazione

Come già preannunciato nel capitolo introduttivo, l'obiettivo di questa tesi è quello di rilevare i movimenti di un terreno soggetto a frane a partire da una serie di fotogrammi scattati da una coppia di fotocamere poste sulla sommità della scena. L'intera attività può essere suddivisa in quattro macro blocchi, ognuno dei quali è descritto in dettaglio in un paragrafo dedicato: astrazione del dataset di immagini (paragrafo 4.1), preprocessing delle immagini in esso contenute (paragrafo 4.2), processing, che comprende identificazione degli spostamenti e ricostruzione 3D (paragrafo 4.3), e postprocessing (paragrafo 4.4). Una visione d'insieme delle quattro sezioni individuate è riportata in Figura 4.1.

Per quanto riguarda la stesura del codice, il linguaggio utilizzato è il C++; inoltre mi sono servito dei seguenti strumenti esterni:

- *OpenCV*: è una libreria open source scritta in C/C++ utilizzata nell'ambito della visione artificiale, sviluppata da Intel e ora supportata da Willow Garage e Itseez. Le applicazioni spaziano notevolmente e si possono ritrovare in ambito artistico, nello stitching di mappe cartografiche, nella robotica, ecc.
- *PCL (Point Cloud Library)*: altra libreria open source impiegata per l'elaborazione di immagini 2D/3D e di nuvole di punti; PCL contiene algoritmi per la stima delle feature, la ricostruzione di superfici, la registrazione di oggetti tridimensionali, ecc.
- *OpenMP (Open Multi-Processing)*: è un API multipiattaforma per la creazione di applicazioni parallele su sistemi a memoria condivisa.
- *Qt*: è una libreria multipiattaforma per lo sviluppo di programmi con interfaccia grafica in linguaggio C++.

4.1 Dataset di immagini

4.1.1 Descrizione dei dataset

Le immagini scattate dalle fotocamere sono organizzate all'interno di cartelle nel file system del sistema operativo. Il primo insieme di immagini contiene foto riprese da no-

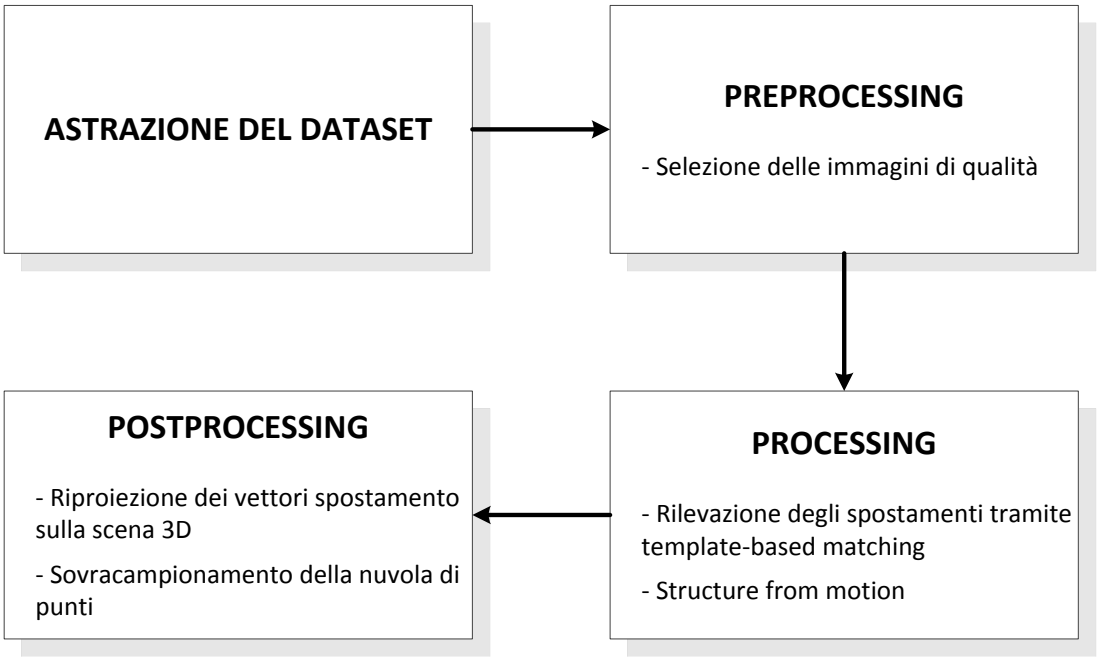

Figura 4.1: *Blocchi principali in cui il lavoro è stato suddiviso. Con l'astrazione del dataset il codice è stato reso indipendente dalla struttura dei dataset forniti. Durante il preprocessing vengono mantenute solo le immagini che soddisfano un determinato requisito di qualità. Il processing consiste nell'individuazione dei vettori spostamento tramite template matching e nella ricostruzione del modello 3D della frana. Nel postprocessing i vettori individuati sono proiettati in 3D sul modello.*

vembre 2011 a marzo 2014 ed è costituito da due cartelle: quella contenente i fotogrammi ottenuti dalla fotocamera sinistra e quella con i fotogrammi della destra. Le foto sono state scattate ogni giorno sempre alla stessa ora e dalla stessa posizione (le fotocamere sono fissate al terreno e protette da un involucro esterno), con parametri di scatto preimpostati; tuttavia nel dataset non sono presenti le immagini di tutti i giorni.

In un momento successivo mi è stato fornito un nuovo dataset di immagini riprese tra luglio e settembre 2014; a differenza del vecchio dataset, questo contiene più immagini scattate nello stesso giorno. Più precisamente ad ogni orario di ripresa sono state catturate tre foto: una con esposizione corretta, una sottoesposta ed una sovraesposta; inoltre, all'interno di una stessa giornata sono state scattate più triple di foto. In questo modo è possibile ricavare immagini qualitativamente superiori combinando le informazioni provenienti da più fotogrammi, per esempio con tecniche HDR (High dynamic range, si veda anche il paragrafo 2.1.1).

Nonostante il vantaggio appena menzionato dell'ultimo dataset, entrambe le collezioni di immagini presentano inconvenienti significativi:

- Le fotocamere non sono ben fissate e quindi si spostano da un giorno all'altro, rendendo necessario un allineamento software delle foto scattate.
- La vegetazione, variabile nel corso dell'anno e piuttosto fitta, impedisce di analizzare gran parte del fotogramma.
- La disposizione delle fotocamere è tale da impedire una ricostruzione completa dell'ambiente in tre dimensioni, in quanto alcune zone della frana sono visibili solo da una delle due macchine fotografiche.
- Nel nuovo dataset ci sono distanze temporali troppo lunghe (fino a due settimane) tra una sequenza di scatti (ripresi nello stesso giorno) e la successiva, rendendo difficile il confronto della morfologia del territorio.

4.1.2 Astrazione dei dataset

Il software che effettua l'analisi dello scivolamento del suolo a partire dalle immagini deve essere indipendente dal dataset utilizzato. In altre parole, a prescindere dalla collezione di fotogrammi utilizzata e dalla struttura in cui è organizzata, i moduli impiegati nella fase successiva devono poter essere eseguiti senza apportare modifiche al codice.

A tal proposito ho creato la classe astratta `Dataset`; sfruttando le sue classi derivate (rispettivamente `Dataset_1` e `Dataset_2` per il vecchio ed il nuovo dataset) è possibile accedere alle immagini in essi contenute impiegando sempre la stessa interfaccia. Un'immagine viene rappresentata per mezzo di una struttura in C++ chiamata `ImageInfo`, mentre la struttura `StereoPairInfo` rappresenta una coppia di immagini, scattate con le fotocamere sinistra e destra. Ciascuna classe derivata da `Dataset` prevede l'inizializzazione di un dizionario (`map` nella standard library di C++) contenente coppie `<Date, StereoPairInfo>` (con `Date` indico una data rappresentata per mezzo delle librerie boost di C++). Una volta terminata l'inizializzazione si può accedere alle foto e ad altre

informazioni relative a dataset diversi invocando metodi con interfacce uguali e fornendo semplicemente le date di inizio e fine.

4.2 Preprocessing

Con *preprocessing* si intendono tutti i provvedimenti intrapresi prima della vera e propria fase di analisi con lo scopo di ottenere un insieme di immagini di qualità soddisfacente su cui poter ricavare delle conclusioni. Infatti, a causa degli eventi atmosferici sfavorevoli (pioggia, neve e nebbia), molti fotogrammi devono essere scartati perché di scarsa qualità; pertanto si è rivelato indispensabile un programma eseguibile che fosse in grado di selezionare solo le foto di qualità e di non considerare le rimanenti.

Il codice che ho scritto in C++ relativo al preprocessing è costituito da una libreria contenente tre classi, che svolgono rispettivamente l'equalizzazione (paragrafo 4.2.1), il feature-based template matching (paragrafo 4.2.2) e l'antidistorsione di un fotogramma (paragrafo 4.2.3), e da un'applicazione eseguibile, che esegue la selezione (paragrafo 4.2.4).

4.2.1 Equalizzazione

L'*equalizzazione*, di cui si è già parlato nel paragrafo 2.1.1 da un punto di vista teorico, è una tecnica che consente di migliorare la distribuzione della luminosità e di aumentare il contrasto di un'immagine. Prima dell'attività di estrazione dei keypoint e matching, può essere utile applicare l'equalizzazione per individuare una quantità di corrispondenze più elevata.

Equalizzazione in OpenCV

Realizzare l'equalizzazione in OpenCV è molto semplice grazie alle funzioni per l'elaborazione degli istogrammi che la libreria mette a disposizione. In particolare la funzione `equalizeHist` equalizza l'istogramma di un'immagine in scala di grigi sfruttando il seguente algoritmo [1]:

- calcola l'istogramma H dell'immagine;
- normalizza l'istogramma in modo tale che la somma delle altezze dei rettangoli sia 255;
- calcola l'integrale dell'istogramma $H'(i) = \sum_{0 \le j < i} H(j)$;
- trasforma l'immagine usando H' come tabella di look-up: $\mathrm{dst}(x, y) = H'(\mathrm{src}(x, y))$.

Siccome le immagini di partenza dei dataset sono a colori e non in scala di grigi, ho dapprima realizzato una conversione nello spazio di colore YCbCr, poi ho equalizzato solo il canale Y (luminanza) e infine ho ricomposto i canali e convertito nello spazio RGB. Per completezza riporto il codice qui di seguito.

Codice 4.1: Codice che effettua l'equalizzazione

```
void equalizeLuminance (Mat img) {

    vector<Mat> channels;
    cvtColor(img, img, CV_BGR2YCrCb);       // Converto da spazio RGB a YCbCr
    split(img, channels);
    equalizeHist(channels[0], channels[0]); // Equalizzo solo il canale Y (luminanza)
    merge(channels, img);
    cvtColor(img, img, CV_YCrCb2BGR);       // Riconverto da spazio YCbCr a RGB
}
```

Valutazione delle alternative

Nel paragrafo 2.1.1 sono stati menzionati metodi alternativi all'equalizzazione, che rispetto a questa presentano punti di forza e svantaggi. La conclusione alla quale sono arrivato dopo numerose prove e valutazioni è che i benefici in termini qualitativi ottenibili con le tecniche più sofisticate non sono così rilevanti da giustificarne l'uso. Infatti, il filtro Retinex consente di ottenere un numero superiore di match, però è computazionalmente dispendioso; inoltre, sia il filtro Retinex sia quelli individuati più recentemente richiederebbero di essere implementati manualmente non essendo disponibili nella libreria OpenCV. Pertanto ritengo che l'equalizzazione rappresenti il miglior compromesso fra tempo d'esecuzione, facilità d'implementazione (si trova anche in OpenCV) e qualità dell'immagine ottenuta. Ho osservato infine che per molte delle immagini analizzate è preferibile non applicare alcun miglioramento preliminare, in quanto una tecnica come l'equalizzazione potrebbe causare una sovraesposizione delle aree di interesse del fotogramma.

4.2.2 Feature-based template matching

La seconda classe di cui è composta la libreria che effettua il preprocessing dei dati si chiama `SiftExtraction`; al suo interno si trovano metodi che permettono di rilevare keypoint, calcolare descrittori, realizzare il feature-based template matching e altre funzionalità.

Estrazione di keypoint in OpenCV

Il rilevamento di keypoint in OpenCV è possibile usando la classe `SiftFeatureDetector`, che accetta in input i seguenti parametri [1].

- `nfeatures`: l'algoritmo conserva gli `nfeatures` keypoint migliori (in base ad una metrica sul contrasto locale). Nel codice ho impostato questa variabile a 1000000: un numero molto elevato rispetto ai keypoint rimanenti dopo i successivi test; tuttavia se avessi scelto un numero inferiore avrei ottenuto una quantità di keypoint buoni troppo bassa.

- `nOctaveLayers`: il numero di layer per ciascuna ottava; come consigliato dall'articolo che presenta SIFT, ho lasciato il valore di default 3; usare un numero più elevato dovrebbe fornire un numero superiore di match a scapito di un costo computazionale maggiore (tuttavia non ho notato differenze rilevanti).

- `contrastThreshold`: soglia imposta sul contrasto minimo. Il valore di default è 0.04; tuttavia, siccome la texture del terreno è piuttosto omogenea, devo essere sicuro di determinare punti di estremo che siano abbastanza significativi, quindi nelle mie applicazioni ho utilizzato un valore più elevato. Non bisogna però alzarlo troppo, altrimenti la quantità di keypoint rilevati è troppo bassa.

- `edgeThreshold`: soglia per rimuovere i keypoint in corrispondenza dei bordi; ho lasciato il valore di default 10.

- `sigma`: la deviazione standard σ della gaussiana applicata all'immagine alla prima ottava; ho lasciato il valore di default 1.6.

Applicando il metodo `detect` ad un oggetto di classe `SiftFeatureDetector` si avvia l'algoritmo che individua i keypoint (si veda il codice seguente).

Codice 4.2: Rilevamento dei keypoint

```
1 SiftFeatureDetector detector(1000000, 3, 0.1);
2 vector<KeyPoint> keypoints;
3 detector.detect(img, keypoints, maskChar);
```

Come si vede, i parametri del metodo `detect` sono l'immagine, il vettore in cui memorizzare i keypoint ed una maschera. Tale maschera specifica in quale porzione dell'immagine ricercare i keypoint; nel mio caso ho impiegato una maschera che nasconde le parti dell'immagine in cui compare la vegetazione e le aree di occlusione dovute alla scatola contenente la fotocamera. Un esempio di maschera è riportato in Figura 4.2.

Figura 4.2: *Immagine originale e immagine con la maschera applicata.*

Calcolo dei descrittori in OpenCV

Dopo aver trovato i keypoint bisogna associare loro dei descrittori. A tale scopo si fa uso della classe `SiftDescriptorExtractor` e del metodo `compute`, che riempie un vettore di descrittori data un'immagine ed il relativo vettore di keypoint.

Codice 4.3: Calcolo dei descrittori

```
1 SiftDescriptorExtractor extractor;
2 Mat descriptors;
3 extractor.compute(img, keypoints, descriptors);
```

Feature-based matching in OpenCV

Per realizzare il matching tra descrittori dei keypoint di una coppia di immagini si può usare la classe `FlannBasedMatcher`. La classe permette di svolgere questa operazione in modo veloce ed efficace sfruttando la libreria FLANN (Fast Approximate Nearest Neighbor Search Library); questa libreria contiene algoritmi ottimizzati per la ricerca dei vicini in grandi dataset di dati. L'onere computazionale di questa classe è decisamente inferiore rispetto a quello della classe `BFMatcher`, dove BF sta per Brute Force; come suggerisce il nome, applicando un approccio a forza bruta, viene confrontata la distanza (euclidea) tra un descrittore nella prima immagine e tutti i descrittori nella seconda immagine.

Al `FlannBasedMatcher` ho applicato il metodo `knnMatch`, che restituisce i k migliori match, dove k è definito dall'utente. Nel mio caso ho specificato $k = 2$, in modo tale da poter applicare il ratio test (paragrafo 2.1.2). Qualora il rapporto tra la distanza del match migliore e quella del secondo match sia inferiore ad una certa soglia, ad esempio 0.6, il match migliore viene conservato, altrimenti vengono scartati entrambi.

Codice 4.4: Matching tra i descrittori

```
1 FlannBasedMatcher matcher;
2 vector<vector<DMatch> > matches;        // Vettore di coppie di match
3 matcher.knnMatch(descriptors1, descriptors2, matches, 2);
4
5 // Ratio test
6 for (int i = 0; i < matches.size(); i++) {
7     if (matches[i][0].distance < 0.6 * matches[i][1].distance) {
8         good_matches.push_back(matches[i][0]);
9     }
10 }
```

Nella Tabella 4.1 sono indicati i tempi approssimativi necessari per il rilevamento dei keypoint in due immagini, il calcolo dei descrittori in due immagini e la rilevazione delle corrispondenze adottando l'approccio basato sulla libreria FLANN.

Operazione	Tempo [s]
Rilevamento keypoint	6.74
Calcolo descrittori	4.83
Matching	0.15

Tabella 4.1: *Tempo necessario per rilevare i keypoint di due immagini, calcolarne i descrittori e determinare il matching tra di essi.*

4.2.3 Antidistorsione delle immagini

Omografia in OpenCV

Una volta individuati i match tra una coppia di immagini scattate con la stessa fotocamera è possibile antidistorcere la seconda immagine in modo che si possa sovrapporre perfettamente alla prima; questa operazione è utile quando, tra uno scatto ed il successivo, si è verificato uno spostamento della macchina fotografica.

L'antidistorsione di una foto in OpenCV a partire da una serie di match noti con un'altra immagine è realizzabile grazie alla funzione `findHomography`. La funzione restituisce una matrice omografica H (paragrafo 2.1.3) che mappa i pixel dell'immagine originale in quelli dell'immagine antidistorta, in modo tale che quest'ultima si sovrapponga all'immagine di riferimento. Siano $\begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T$ un pixel dell'immagine da antidistorcere espresso in coordinate omogenee, $\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix}^T$ il relativo pixel nell'immagine antidistorta e h_{ii} gli elementi di H; la matrice viene trovata in modo che la quantità

$$\sum_i \left(\left(x'_i - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y'_i - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \right)$$

sia minimizzata.

A questo punto si potrebbe obiettare quanto segue: se tra una foto e la successiva si verificasse un'evoluzione della frana, la funzione che calcola l'antidistorsione riterrebbe comunque che i match riscontrati in corrispondenza delle zone in cui è avvenuta la frana non si siano spostati tra un fotogramma e l'altro, restituendo così una matrice errata. Tuttavia bisogna considerare che la matrice omografica è calcolata a partire da una quantità considerevole di keypoint, pertanto, anche se è presente un numero esiguo di match errati per il motivo suddetto, questi risultano poco rilevanti ai fini della determinazione della matrice. Inoltre la matrice è di dimensioni ridotte, quindi non sarebbe in grado di rappresentare con precisione le piccole variazioni locali dovute a spostamenti di porzioni limitate del terreno (una omografia si applica a spazi proiettivi e preserva la collinearità).

Se però fosse l'intera area analizzata ad essere interessata da uno spostamento di materiale, allora probabilmente la matrice calcolata non sarebbe corretta. Tuttavia questo fenomeno non si verifica nel mio caso per due ragioni:

- tra un giorno ed il successivo l'area soggetta a frane è sempre una porzione limitata dell'intera regione considerata;

- l'algoritmo SIFT considera come poco affidabili i match tra porzioni di terreno che sono mutate tra una foto e la successiva e quindi li scarta a priori.

La funzione `findHomography` prevede di default l'uso di tutti i match per calcolare la matrice omografica, sfruttando il metodo dei minimi quadrati. Nel mio caso però non tutti i match forniti alla funzione sono corretti (nonostante il ratio test), e quindi non sarebbe possibile individuare una trasformazione prospettica rigida valida se si impiegasse questo semplice approccio.

Per questo motivo `findHomography` dà la possibilità di usare i metodi RANSAC e LMeDs, i quali scelgono sottoinsiemi casuali di 4 match, stimano la matrice omografica a partire da essi e valutano una metrica di qualità dell'omografia individuata. Il sottoinsieme migliore viene usato per ottenere una stima iniziale della matrice. In ogni caso la matrice viene successivamente raffinata con il metodo di Levenberg-Marquardt (risoluzione di problemi non lineari ai minimi quadrati) per ridurre ulteriormente l'errore di riproiezione.

Antidistorsione della foto

Per concludere l'approfondimento della funzione `findHomography`, resta da spiegare il significato del parametro `ransacReprojThreshold`. Alla luce dei chiarimenti sull'algoritmo RANSAC appena esposti, risulta intuitivo che questo valore rappresenta l'errore di riproiezione massimo consentito espresso in pixel per poter considerare un punto come inlier. Formalmente, se:

$$||\text{dst}_i - H \cdot \text{src}_i|| > \text{ransacReprojThreshold} \tag{4.1}$$

il punto i è considerato un outlier. Nel mio codice ho lasciato questo parametro impostato a 3, il valore di default.

Dopo aver trovato la matrice H con `findHomography`, si può applicare la trasformazione prospettica utilizzando la funzione `warpPerspective`. Questa trasforma un'immagine sfruttando la matrice H data in input in base alla seguente formula:

$$\text{dst}(x, y) = \text{src}\Big(\frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}\Big). \tag{4.2}$$

Per completezza riporto il codice che esegue l'antidistorsione.

Codice 4.5: Antidistorsione dell'immagine

```
1 Mat undistorted_img;
2 // Passo a findHomography le coordinate dei match
3 Mat H = findHomography(best_keypoints2_pt, best_keypoints1_pt, CV_RANSAC);
4 warpPerspective(img, undistorted_img, H, Size(img.cols, img.rows));
```

Un esempio di antidistorsione è visibile in Figura 4.3. Per rendere più chiaro lo scostamento tra fotogramma sorgente e destinazione e la sovrapponibilità tra il fotogramma di destinazione antidistorto e quello sorgente, ho posto quest'ultimo al centro.

Figura 4.3: *Esempio di antidistorsione. Dall'alto: foto scattata il 28-12-2011, foto scattata il 19-11-2011 e foto scattata il 28-12-2011 antidistorta (si noti come quest'ultima sia perfettamente sovrapponibile alla foto centrale).*

Utilità dell'antidistorsione

Si è visto come adottando opportuni accorgimenti sia possibile deformare un'immagine in modo che si possa sovrapporre perfettamente ad un'altra immagine ripresa da un punto di vista diverso.

Questa attività può essere considerata una premessa alla successiva operazione che ho svolto, ovvero il *tamplate-based matching*. Questa tecnica consiste nell'individuare una porzione di un'immagine (template) all'interno di un'altra immagine (si veda il paragrafo 2.2); nel mio caso specifico, in questo modo posso determinare i vettori spostamento che descrivono il moto del materiale soggetto a frane.

Risulta intuitivo perciò che, affinché i vettori siano significativi, le immagini devono essere allineate tra loro. In caso contrario il template matching troverebbe spostamenti nell'intero fotogramma, che però non corrispondono ad un'evoluzione della frana bensì ad un movimento della fotocamera. I provvedimenti che ho preso impediscono il verificarsi di questo fenomeno indesiderato.

4.2.4 Selezione delle immagini corrette

Il motivo per cui ho individuato i match tra due immagini è stabilire se la seconda immagine è di buona qualità o meno. Infatti, assumendo che la prima foto sia valida, si presume che lo sia anche la seconda se tra di esse è presente un numero di corrispondenze superiore ad una certa soglia. La classe `AutomaticImagesSelection` si occupa di svolgere questa mansione: dati in ingresso i giorni di inizio e fine, inserisce in un file con estensione `.yaml` l'elenco di tutte le immagini di qualità soddisfacente scattate durante quell'intervallo di tempo. Quest'applicazione eseguibile è provvista di un'interfaccia grafica creata con la libreria Qt; il modo in cui si presenta è visibile in Figura 4.4.

La selezione delle immagini può essere effettuata sia sul vecchio che sul nuovo dataset, dato che le interfacce dei metodi della classe base `Dataset` sono valide per tutti i tipi di dataset da essa derivati. Il software ha dimostrato in generale un buon comportamento, scartando con buon successo le foto di qualità scadente. Solo in certe situazioni i risultati ottenuti non sono ottimali: per esempio, nel vecchio dataset, le foto scattate tra il 7 gennaio 2012 ed il 10 aprile 2012 presentano una messa a fuoco scorretta, dovuta probabilmente a qualche spostamento accidentale della fotocamera. L'algoritmo di scelta delle foto, com'è lecito aspettarsi, scarta tutti questi fotogrammi; il cambiamento della conformazione della frana avvenuto durante questo periodo di tempo ha però compromesso la possibilità di realizzare un matching tra l'ultima foto accettabile scattata prima del periodo suddetto e la prima foto accettabile ripresa dopo di esso. Si tratta comunque di un caso eccezionale dovuto alla cattiva qualità delle collezioni di immagini fornite.

4.2.5 Riepilogo

Il diagramma di flusso che sintetizza la fase di preprocessing è riportato in Figura 4.5. Si noti in particolare come i match individuati possano essere sfruttati in due modi:

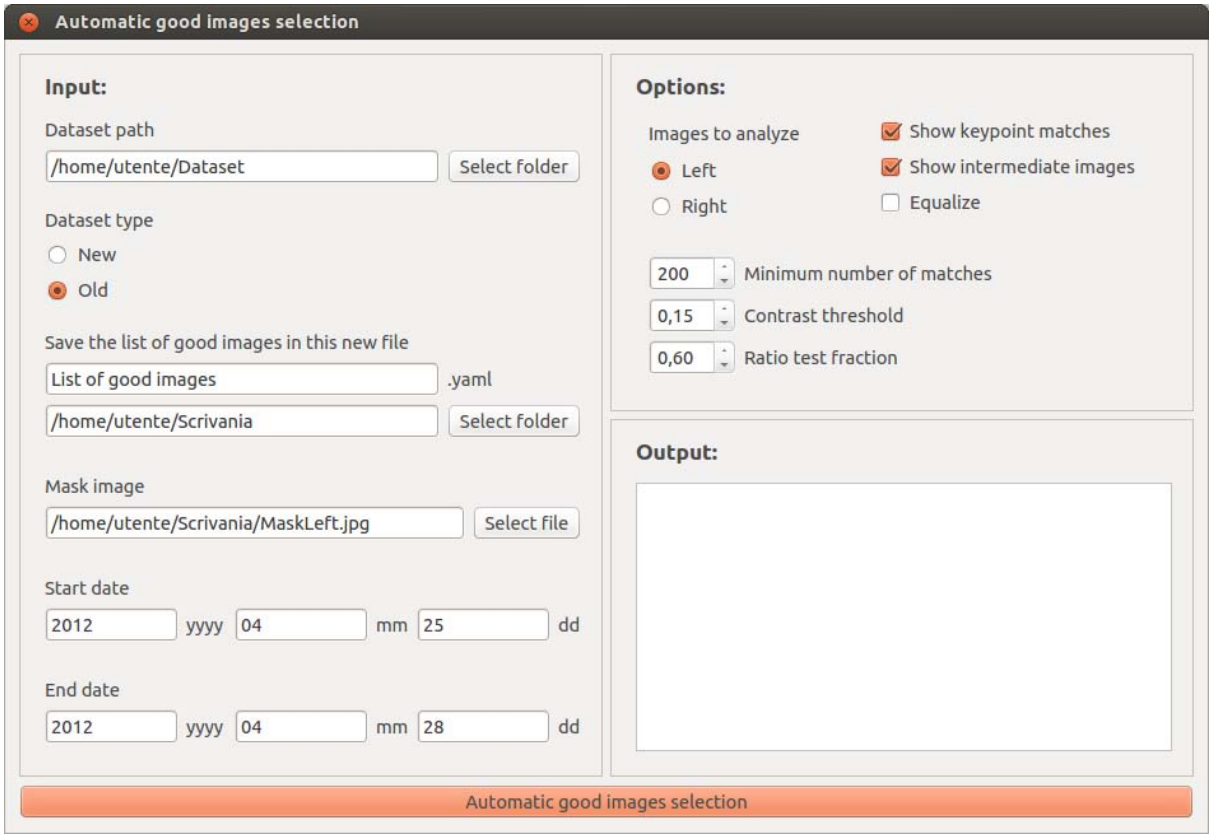

Figura 4.4: *Interfaccia grafica del programma di selezione delle immagini corrette.*

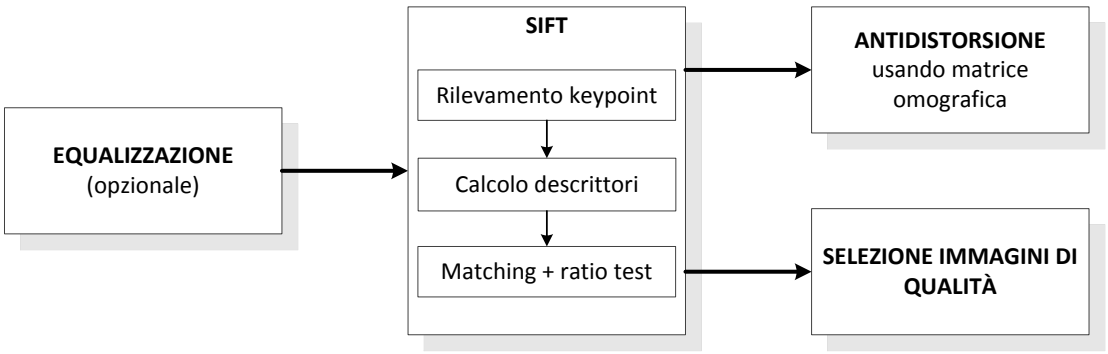

Figura 4.5: *Diagramma di flusso del preprocessing.*

- per realizzare l'antidistorsione, ovvero per rendere la coppia di foto sovrapponibile;
- per valutare se la seconda immagine è simile alla prima, contando il numero di match trovati.

4.3 Processing

La fase di *processing* rappresenta il punto cardine dell'intero lavoro: consiste nella realizzazione del template-based matching, utilizzato per rilevare i vettori spostamento, e nell'attività di ricostruzione 3D. Quest'ultima non può essere effettuata per mezzo di una calibrazione stereo di tipo standard per due ragioni:

- le fotocamere si muovono l'una rispetto all'altra nel tempo perché non sono ben fissate al terreno, quindi non si possono sfruttare le matrici di rotazione e traslazione ottenute con la calibrazione stereo;
- la calibrazione prevede una fase iniziale in cui viene acquisito da varie posizioni uno stesso pattern (ad esempio una scacchiera); date le dimensioni notevoli dell'area ripresa, la scacchiera dovrebbe avere un lato con lunghezza dell'ordine dei cento metri, presupposto non applicabile.

Per questi motivi bisogna fare ricorso a tecniche di Structure From Motion.

Il codice scritto in C++ è composto da due librerie, che effettuano il template-based matching (paragrafo 4.3.1) e l'implementazione della Structure From Motion (paragrafo 4.3.2). Inoltre sono presenti tre programmi eseguibili, che si occupano rispettivamente di visualizzare i vettori spostamento in due dimensioni e di ricostruire la scena secondo due approcci distinti.

4.3.1 Template-based matching

Il template-based matching consente di individuare una porzione d'immagine, chiamata template, all'interno di un'altra foto contenente un oggetto simile. In particolare, nel caso del monitoraggio di una frana, questa tecnica può essere utilizzata per determinare i vettori spostamento associati al materiale sottoposto ai cedimenti (a tal proposito si veda il paragrafo 1.2, che introduce il concetto di Particle image velocimetry). Esistono più funzioni che valutano la "similitudine" di due immagini; per esempio la formula SSD (Sum of Squared Differences), la cross-correlazione e la cross-correlazione normalizzata (paragrafo 2.2.1).

Per rilevare uno spostamento del terreno sono necessarie due immagini: la prima viene suddivisa in celle, ciascuna delle quali rappresenta un template; per ogni template si va a cercare all'interno della seconda immagine la posizione in corrispondenza della quale la funzione che calcola la somiglianza è massimizzata (o minimizzata). A tale posizione viene associato il miglior match; se la posizione è la stessa del template nell'immagine di

partenza allora non c'è stato alcuno spostamento, altrimenti viene tracciato un vettore tra il punto centrale del template ed il punto centrale della migliore porzione individuata.

Idealmente, si potrebbe associare un vettore spostamento ad ogni punto dell'immagine originale (il punto al centro del template); siccome però si tratta di un'operazione molto costosa da un punto di vista computazionale, può essere preferibile cercare i vettore a distanze maggiori rispetto al singolo pixel. In ogni caso, supponendo che l'evoluzione della frana tra un giorno ed il successivo non sia troppo rapida, la ricerca del miglior match si può limitare ad un'area posta attorno alla posizione del template originario. Più precisamente, l'area al di sopra del template non viene considerata (dato che la frana non può procedere verso l'alto), mentre sotto e ai lati è stata esaminata una superficie di ampiezza pari a quella del template; per maggior chiarezza si veda la Figura 4.6.

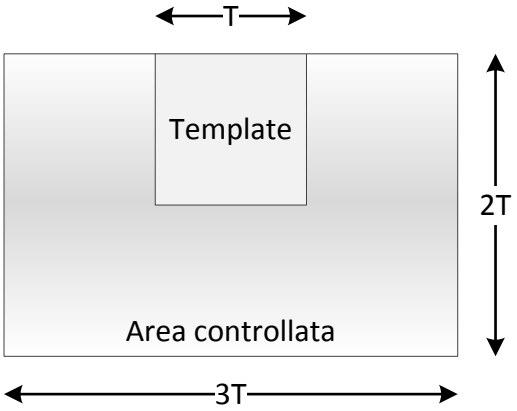

Figura 4.6: *Area all'interno della quale viene realizzato il template matching. L'area al di sopra del template non viene considerata (dato che la frana non può procedere verso l'alto), mentre sotto e ai lati è stata esaminata una superficie di ampiezza pari a quella del template.*

Template-based matching in OpenCV

In OpenCV il template matching si può realizzare grazie alla funzione `matchTemplate` [1]; questa richiede di fornire in ingresso l'immagine che rappresenta il template, l'immagine in cui ricercare il template ed il metodo da impiegare per valutare la somiglianza (ovvero la funzione di valutazione). In uscita si ottiene un'immagine in scala di grigi che codifica la bontà del match per ogni spostamento analizzato (il template viene fatto scorrere di pixel in pixel all'interno dell'immagine). Per determinare la posizione del mi-

glior match si usa la funzione `minMaxLoc`, che individua il minimo ed il massimo valore contenuto in una matrice e le relative posizioni.

Per quanto riguarda le funzioni valutate per determinare la somiglianza, OpenCV mette a disposizione sei metodi: `SQDIFF` (Sum of Squared Differences, si veda la formula 2.1), `SQDIFF_NORMED` (versione normalizzata di `SQDIFF`), `CCORR` (cross-correlazione, formula 2.2), `CCORR_NORMED`, `CCOEFF` e `CCOEFF_NORMED` (cross-correlazione normalizzata, formula 2.3). Come prevedibile, la funzione che fa uso della cross-correlazione normalizzata è di gran lunga quella che fornisce i migliori risultati (per curiosità, il nome `CCOEFF_NORMED` usato da OpenCV è dovuto al fatto che la cross-correlazione normalizzata è la versione bidimensionale del Pearson product-moment correlation coefficient). Il confronto tra i vari metodi, applicati ad una coppia di immagini, è illustrato in Figura 4.7; i vettori spostamento sono calcolati ogni 50 pixel, sia lungo l'asse x che lungo l'asse y, e vengono determinati solo in corrispondenza delle aree bianche di un'opportuna maschera che nasconde le zone che non sono di interesse.

Parallelizzazione del template-based matching

Il template-based matching è un'operazione molto onerosa: per ogni pixel per il quale si vuole determinare un vettore spostamento bisogna infatti stimare una matrice risultato; ogni pixel di questa matrice è stato calcolato a sua volta applicando una delle note formule, che considerano tutti i pixel del template e dell'immagine a cui questo si sovrappone. Per limitare almeno in parte questo inconveniente, può essere utile ricorrere alla libreria OpenMP che, con semplici istruzioni e le dovute accortezze, consente di sfruttare appieno la potenza di calcolo del processore distribuendo l'elaborazione su tutti i core. Dalla Tabella 4.2 si evince che quando si analizza una sequenza di quattro immagini il risparmio di tempo ammonta all'incirca ad un fattore due in un processore dotato di quattro core logici; il template matching è realizzato su una coppia di immagini di dimensioni 5184 per 3456 pixel (la maschera si può intuire dalle immagini in Figura 4.7) e i vettori spostamento sono distanziati tra loro di 50 pixel.

Operazione	Tempo [s]
Template matching senza OpenMP	38.35
Template matching con OpenMP	20.89

Tabella 4.2: *Tempo necessario per effettuare il template-based matching senza e con parallelizzazione, con ottimizzazione dei vettori attivata.*

Ottimizzazione dei vettori spostamento

Sebbene la cross-correlazione normalizzata fornisca in generale risultati accettabili, capita di frequente che alcuni dei vettori spostamento individuati siano scorretti. Per ridurre la quantità di match errati si può applicare un punteggio a ciascun vettore, che

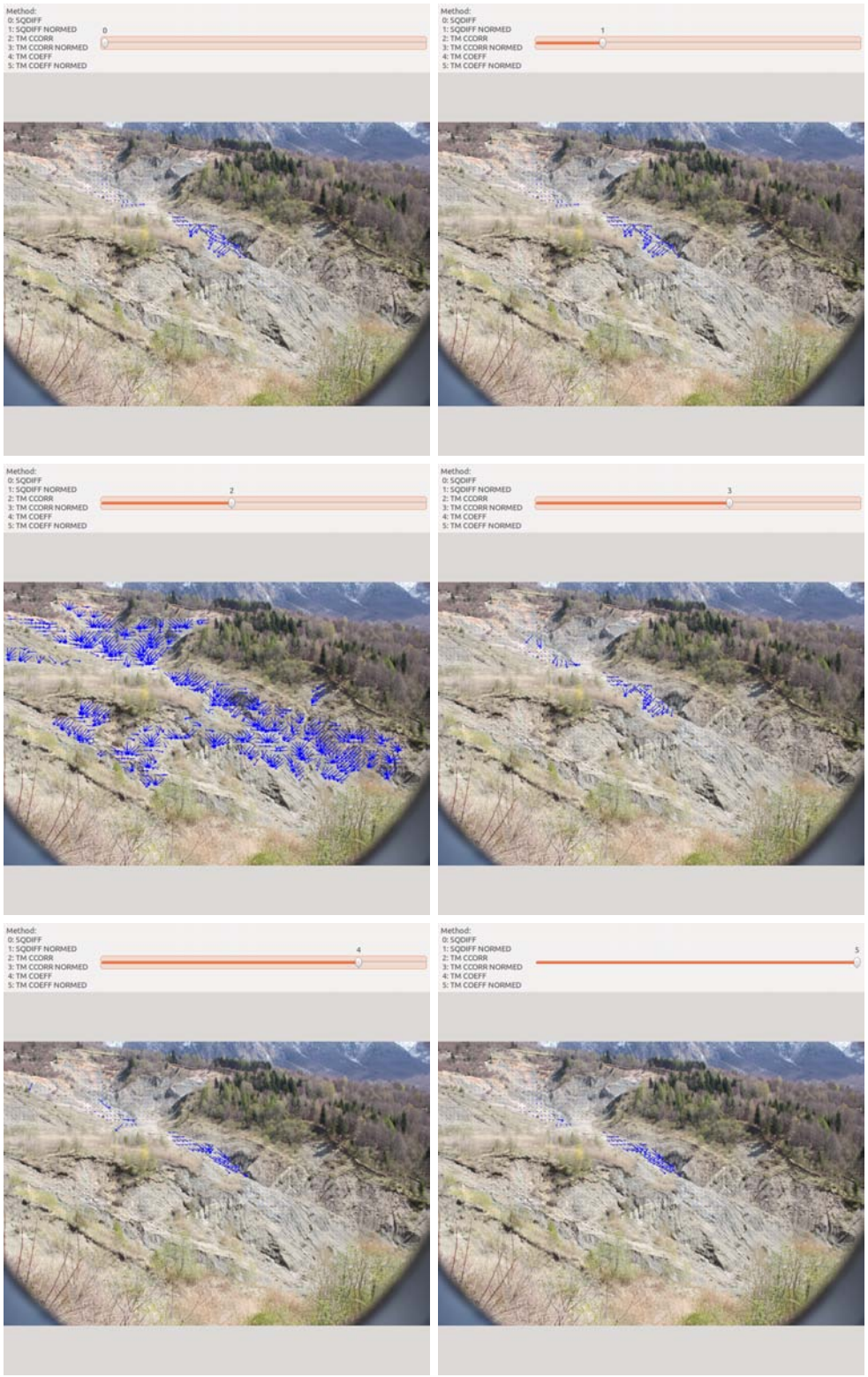

Figura 4.7: *Confronto tra i metodi per il template matching di OpenCV; nell'ordine, da sinistra verso destra e dall'alto verso il basso,* `SQDIFF`*,* `SQDIFF_NORMED`*,* `CCORR`*,* `CCORR_NORMED`*,* `CCOEFF` *e* `CCOEFF_NORMED`*. Si noti come l'ultimo metodo fornisca risultati decisamente migliori.*

ne rappresenta la bontà: il vettore è considerato corretto solo se questo parametro è superiore ad una certa soglia.

Per valutare la qualità di un vettore è necessario effettuare confronti con i vettori rilevati in altre immagini: identificando una data con una variabile, se lo scopo è quello di trovare i vettori che rappresentano lo spostamento tra la data x e la data y, con x e y non adiacenti, si dovrà confrontare anche x con le foto scattate prima di y (tra $x+1$ e $y-1$) e con quelle successive a y. Il punteggio assegnato è una sintesi di più valutazioni: in particolare, un vettore spostamento trovato fra le date x e y deve soddisfare i seguenti requisiti:

- deve avere modulo maggiore o uguale a quello del vettore trovato fra x e z, con z compreso tra $x+1$ e $y-1$;
- deve avere modulo minore o uguale a quello del vettore trovato fra x e z, con z maggiore di y;
- deve avere direzione simile a quella del vettore trovato fra x e z, con z maggiore di x;
- deve avere direzione simile a quella dei suoi vettori adiacenti (nella stessa immagine).

Il punteggio finale è una media pesata del punteggio assegnato a ciascuna di queste affermazioni; questa tecnica è nota col nome di *comitato*.

Se il punteggio complessivo non supera la soglia minima, bisogna cercare un altro vettore; la strategia che ho adottato per individuare altri vettori è quella di esaminare i massimi locali dell'immagine in scala di grigi restituita dal template matching. Si è già visto infatti che il vettore spostamento viene tracciato dalla posizione originale alla posizione in cui è localizzato il massimo globale; se questo massimo non fornisce un vettore di qualità soddisfacente, si individua quello associato al secondo massimo, e così via finché un massimo che porta ad un vettore adeguato è stato trovato o sono stati esauriti tutti i massimi locali dell'immagine.

Durante la scrittura del codice che realizza questa funzionalità mi sono scontrato con il problema di dare la definizione di massimo locale per una funzione in due variabili: infatti l'immagine in scala di grigi contiene del rumore, perciò se non si definisce accuratamente la nozione di massimo si rischia di prendere dei punti che corrispondono alle oscillazioni del rumore. Per risolvere la questione ho deciso di imporre una soglia al di sotto della quale le variazioni di intensità non vengono considerate.

In Figura 4.8 si può apprezzare la differenza tra il confronto di una coppia di immagini senza (in alto) e con (in basso) correzione dei vettori. In entrambi i casi le due immagini appartengono al vecchio dataset e sono state scattate il 25 aprile 2012 e il 2 maggio 2012; per la correzione dei vettori ho analizzato le immagini comprese tra il 25 aprile ed il 4 maggio (ho considerato solo le foto che non sono state scartate dall'algoritmo di selezione delle immagini di qualità).

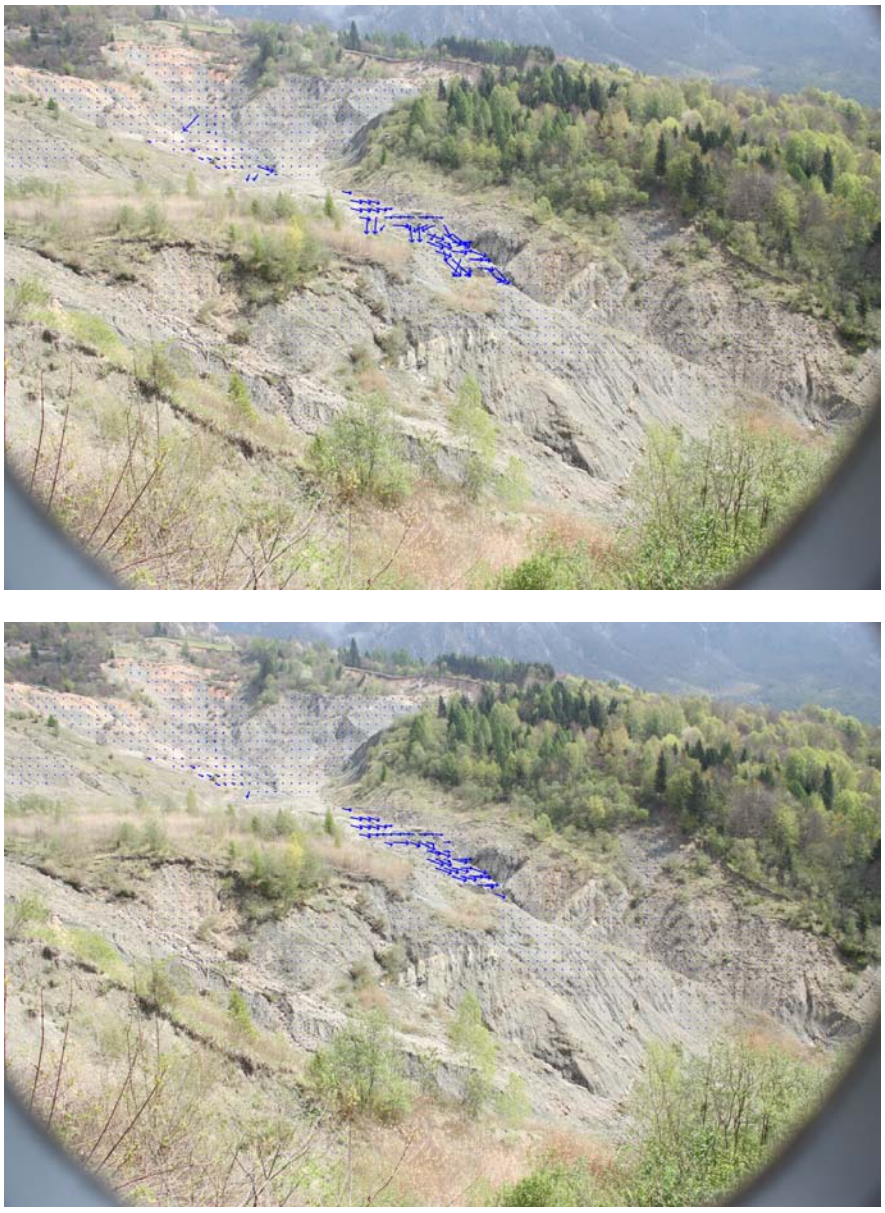

Figura 4.8: *Vettori spostamento che rappresentano l'evoluzione tra il 25 aprile 2012 e il 2 maggio 2012. In alto i vettori senza correzioni e in basso con correzioni.*

Visualizzazione dei vettori spostamento

Le immagini di questo paragrafo sono il risultato dell'applicazione che effettua il template-based matching. Anche in questo caso ho realizzato un'interfaccia grafica che rende più intuitivo l'uso del programma (si veda la Figura 4.9).

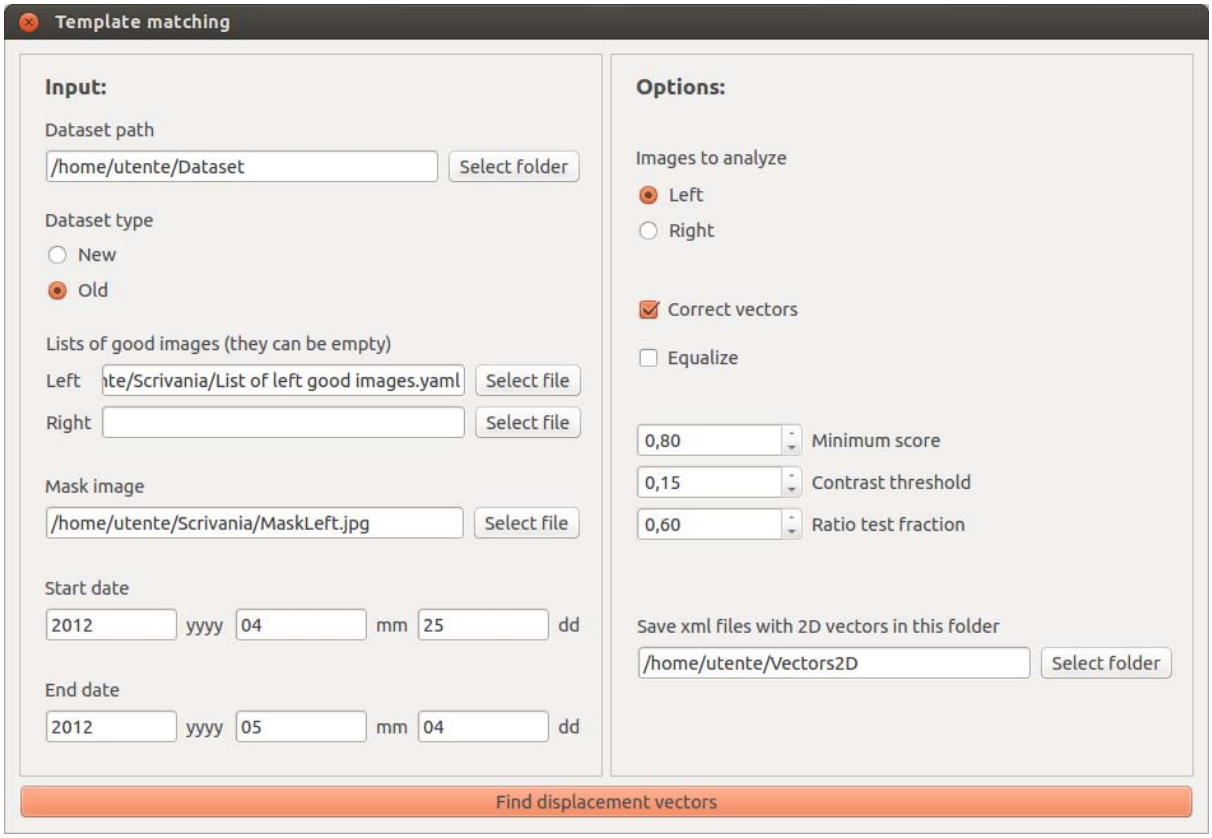

Figura 4.9: *Interfaccia grafica del programma che realizza il template matching.*

L'applicazione richiede che siano specificate le date di inizio e fine e qualche altro parametro; inoltre è necessario fornire gli elenchi delle immagini di qualità ottenute in precedenza alla fine del preprocessing. Dopo aver completato il matching e l'eventuale correzione dei vettori spostamento, vengono mostrate in sequenza le immagini analizzate, con i vettori sovrimpressi. L'evoluzione giorno per giorno della frana è deducibile dai vettori spostamento corretti della Figura 4.10 (l'algoritmo ha analizzato le foto di buona qualità comprese tra il 25 aprile 2012 e il 4 maggio 2012).

4.3.2 Structure From Motion

La ricostruzione della frana in tre dimensioni si può ricavare seguendo due approcci distinti: il primo non presuppone la conoscenza di alcun parametro delle fotocamere (metodo senza calibrazione), il secondo prevede la conoscenza dei parametri intrinseci delle

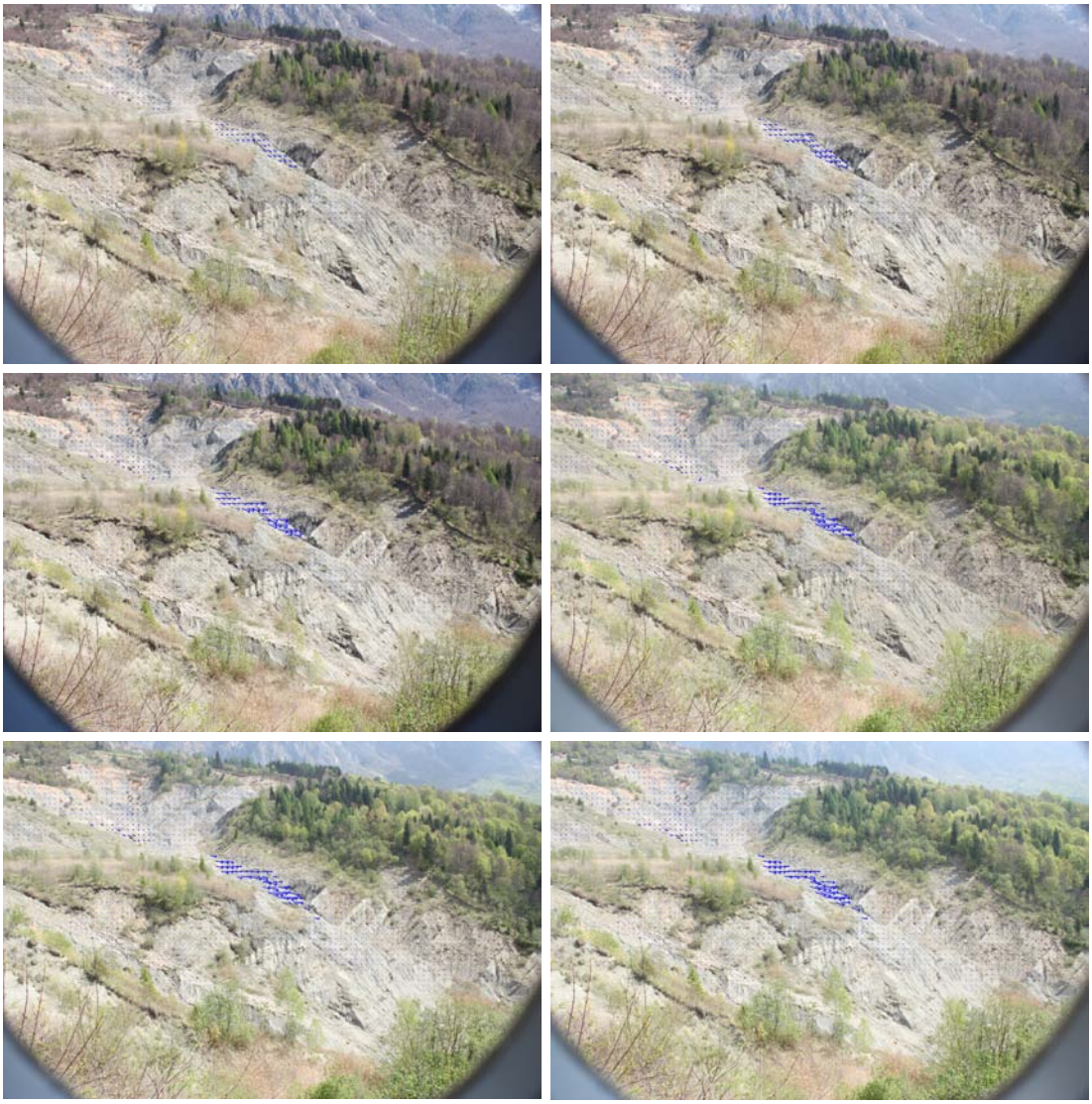

Figura 4.10: *Evoluzione della frana dal 25 aprile 2012 al 4 maggio 2012, con correzioni applicate ai vettori.*

due macchine fotografiche. Per realizzare la Structure From Motion ho creato una libreria contenente la classe base `StructureFromMotion` e le due classi derivate `StructureFromMotionUncalibrated` e `StructureFromMotionSemicalibrated`, i cui nomi ricordano che la prima non fa uso di dati di calibrazione, mentre la seconda utilizza solo i parametri intrinseci. La classe `StructureFromMotion` contiene la definizione dei metodi comuni alle due classi derivate; in particolare specifica il metodo `findAllMatches` che, note le date di inizio e fine, svolge la sequenza di operazioni seguente.

- Esegue il template-based matching tra la foto scattata alla data d'inizio e tutte le altre comprese nell'intervallo, con correzione dei vettori spostamento. Questo passo ha lo scopo di individuare le aree in cui c'è stata un'evoluzione della frana e quelle in cui non si sono verificati movimenti; l'elaborazione viene svolta sia per le immagini sinistre sia per quelle destre.

- Individua keypoint e descrittori di tutte le immagini nell'intervallo; dalla seconda immagine in poi, nelle zone in cui al passo precedente si sono riscontrati spostamenti di materiale, non rileva i keypoint. In generale ho cercato keypoint con una densità molto elevata, in modo da trovarne il più possibile; questo è stato possibile utilizzando un valore di `contrastThreshold` inferiore (si veda il paragrafo 4.2.2).

- Per ciascuna coppia di immagini comprese nell'intervallo (immagine sinistra e rispettiva destra) effettua il feature-based template matching per ricavare le corrispondenze tra keypoint. Tutti i match trovati vengono inseriti in un unico pool.

- Eventualmente rimuove i duplicati dal pool di match.

Queste attività sono in comune ai due approcci; dopodiché essi procedono indipendentemente. Una sintesi di questo elenco è riportato in Figura 4.11.

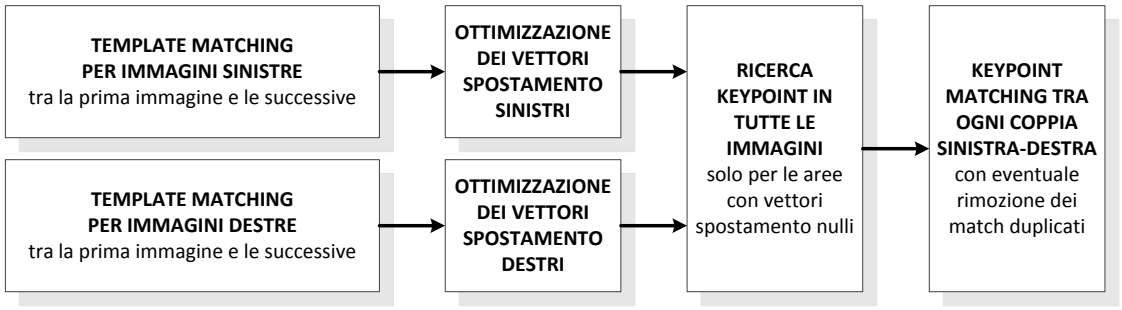

Figura 4.11: *Diagramma di flusso della Structure From Motion comune ai due approcci.*

Ricostruzione 3D senza parametri intrinseci in OpenCV

Il processo di ricostruzione 3D secondo l'approccio che non fa uso di dati di calibrazione si articola nei seguenti punti (per la teoria si veda il paragrafo 3.3.1).

- Calcola la matrice fondamentale F a partire dai match ricavati in precedenza; per fare questo chiama la funzione `findFundamentalMat` di OpenCV, che trova F tale che $\begin{bmatrix} p_2 & 1 \end{bmatrix}^T F \begin{bmatrix} p_1 & 1 \end{bmatrix} = 0$, dove p_1 e p_2 sono una coppia di punti corrispondenti. Per determinare una matrice più affidabile usa l'algoritmo RANSAC (se ci sono almeno otto corrispondenze).

- Compie la stereo rettificazione sfruttando l'algoritmo di Hartley grazie alla funzione `stereoRectifyUncalibrated` di OpenCV che, date le corrispondenze e la matrice F appena calcolata, restituisce due matrici omografiche $H1$ e $H2$ da applicare alla coppia di immagini originali.

- Applica l'algoritmo SGBM per determinare le corrispondenze sulla stessa riga e ricava la matrice di disparità (fa uso della classe `StereoSGBM`).

- Normalizza i valori della mappa di disparità e riproietta quest'ultima nello spazio tridimensionale (funzione `reprojectImageTo3D`).

La Figura 4.12 mostra la scena tridimensionale della frana vista da due punti di vista diversi; questa è stata ottenuta da sei coppie di immagini del nuovo dataset, con date comprese tra 1 agosto 2014 e 17 settembre 2014. Si può notare che con questo metodo la ricostruzione non è propriamente ottimale; inoltre sono assenti le aree della frana di maggior interesse.

La Figura 4.13 invece riassume la procedura di ricostruzione senza dati di calibrazione (il diagramma di flusso è la prosecuzione di quello riportato in Figura 4.11).

Ricostruzione 3D con parametri intrinseci in OpenCV

Pur non disponendo di informazioni sulla calibrazione, è stato possibile stimare i parametri intrinseci delle fotocamere a partire dai dati Exif associati ai file immagine. I passi da eseguire per ricostruire la frana in tre dimensioni a partire dalle informazioni stimate sono già stati illustrati da un punto di vista teorico nel paragrafo 3.3.2.

- Calcola la matrice fondamentale F usando la funzione `findFundamentalMat`. La qualità finale dipende fortemente dal valore del parametro d'ingresso `param1` che rappresenta la massima distanza ammessa da un punto ad una linea epipolare in pixel, oltre la quale il punto è considerato un outlier. Purtroppo non esiste una regola inequivocabile per attribuire un valore a questo parametro; in generale, più è basso (dell'ordine di 0.1) maggiore è la probabilità che la matrice fondamentale calcolata sia migliore.

- Ottiene la matrice essenziale E a partire dalla matrice fondamentale F per mezzo della formula 3.9 e usando i parametri intrinseci stimati.

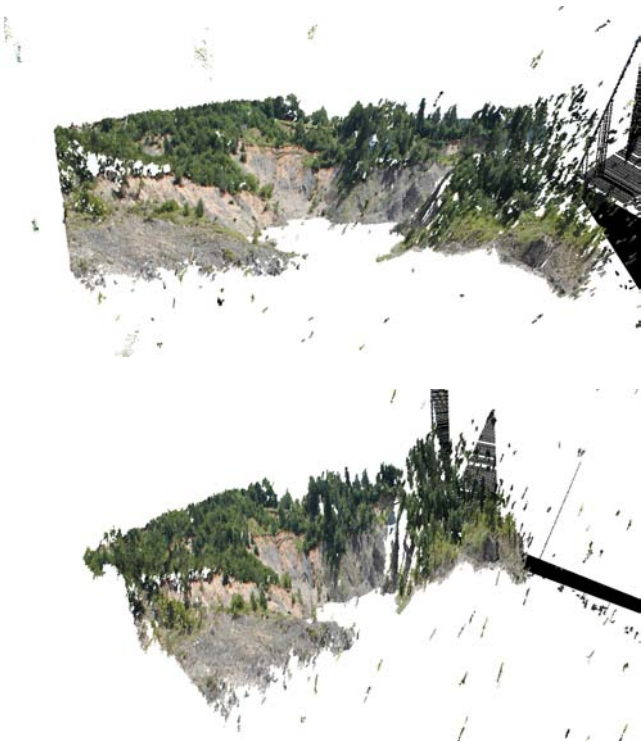

Figura 4.12: *Scena ricostruita senza conoscere i parametri intrinseci.*

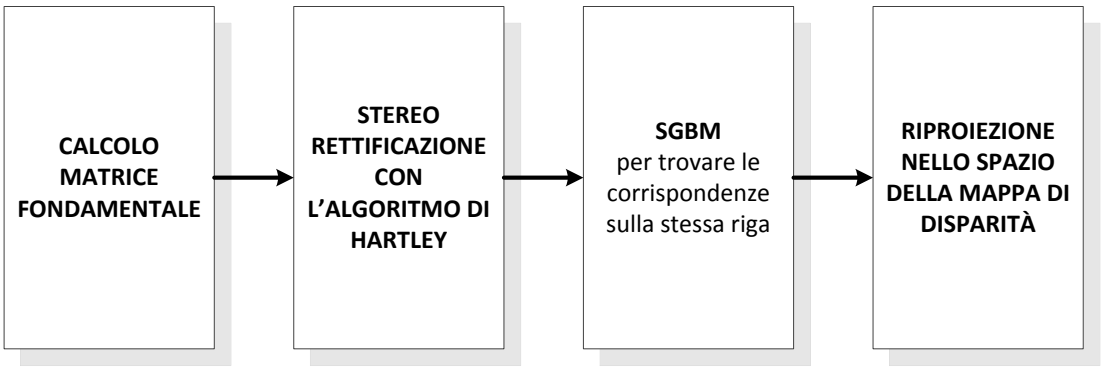

Figura 4.13: *Continuazione del diagramma di flusso della Structure From Motion nel caso in cui non si utilizzano i parametri intrinseci.*

- Fissata la matrice di proiezione $P = [I \mid \mathbf{0}]$ relativa alla fotocamera sinistra, sceglie la matrice di proiezione P' per la fotocamera destra tra le quattro possibili configurazioni che può assumere.

- Effettua la triangolazione dei keypoint nello spazio; se i punti 3D trovati non stanno di fronte ad entrambe le telecamere, sceglie un'altra configurazione per la matrice P'.

Il risultato finale, ottenuto dalle stesse immagini del caso precedente, è visibile in Figura 4.14; sebbene la nuvola di punti ricostruita sia poco densa e non sia molto comprensibile a prima vista, la ricostruzione è molto più affidabile rispetto a quanto avveniva con l'altro approccio. Ciò non toglie che quanto ottenuto non sia del tutto soddisfacente, ma questo è dovuto principalmente alla pessima qualità dei dataset forniti (per le ragioni spiegate nel paragrafo 4.1.1). Anche la maschera utilizzata e i parametri impostati sono decisivi per la bontà della ricostruzione; esistono inoltre fotogrammi che, nonostante abbiano superato il test di selezione delle immagini, non consentono di produrre una buona nuvola di punti. Per quantificare l'accuratezza del modello 3D generato ho calcolato l'errore di riproiezione medio per entrambe le fotocamere.

Ricordo inoltre che alla nuvola di punti visualizzata è stato applicato un filtraggio statistico (*SOR, Statistical Outlier Removal*), che consente di rimuovere i punti spuri facendo considerazioni sulla loro distribuzione nello spazio. Infine, per quanto riguarda la complessità computazionale della ricostruzione, non dovendo eseguire l'algoritmo SGBM, che richiede un certo tempo, il paesaggio 3D viene generato quasi istantaneamente.

Gran parte dei dettagli implementativi della Structure From Motion a partire dai parametri intrinseci è stata tratta dal libro [2] e dal relativo codice esemplificativo[1].

Per finire, lo schema in Figura 4.15 riassume la procedura di ricostruzione che fa uso dei parametri intrinseci (anche in questo caso il diagramma di flusso è la prosecuzione di quello della Figura 4.11); la Figura 4.16 riporta invece l'interfaccia grafica del programma realizzato.

4.4 Postprocessing

Durante la fase di processing sono stati individuati e corretti i vettori spostamento ed è stato costruito un modello tridimensionale della frana. A questo punto può essere utile aumentare la densità della nuvola di punti ricavata con l'approccio che utilizza i parametri intrinseci, dato che attualmente contiene solo i keypoint (paragrafo 4.4.1). Infine i vettori spostamento possono essere proiettati in tre dimensioni e sovrimpressi alla nuvola di punti (paragrafo 4.4.2).

[1] `https://github.com/MasteringOpenCV/code/tree/master/Chapter4_StructureFromMotion`

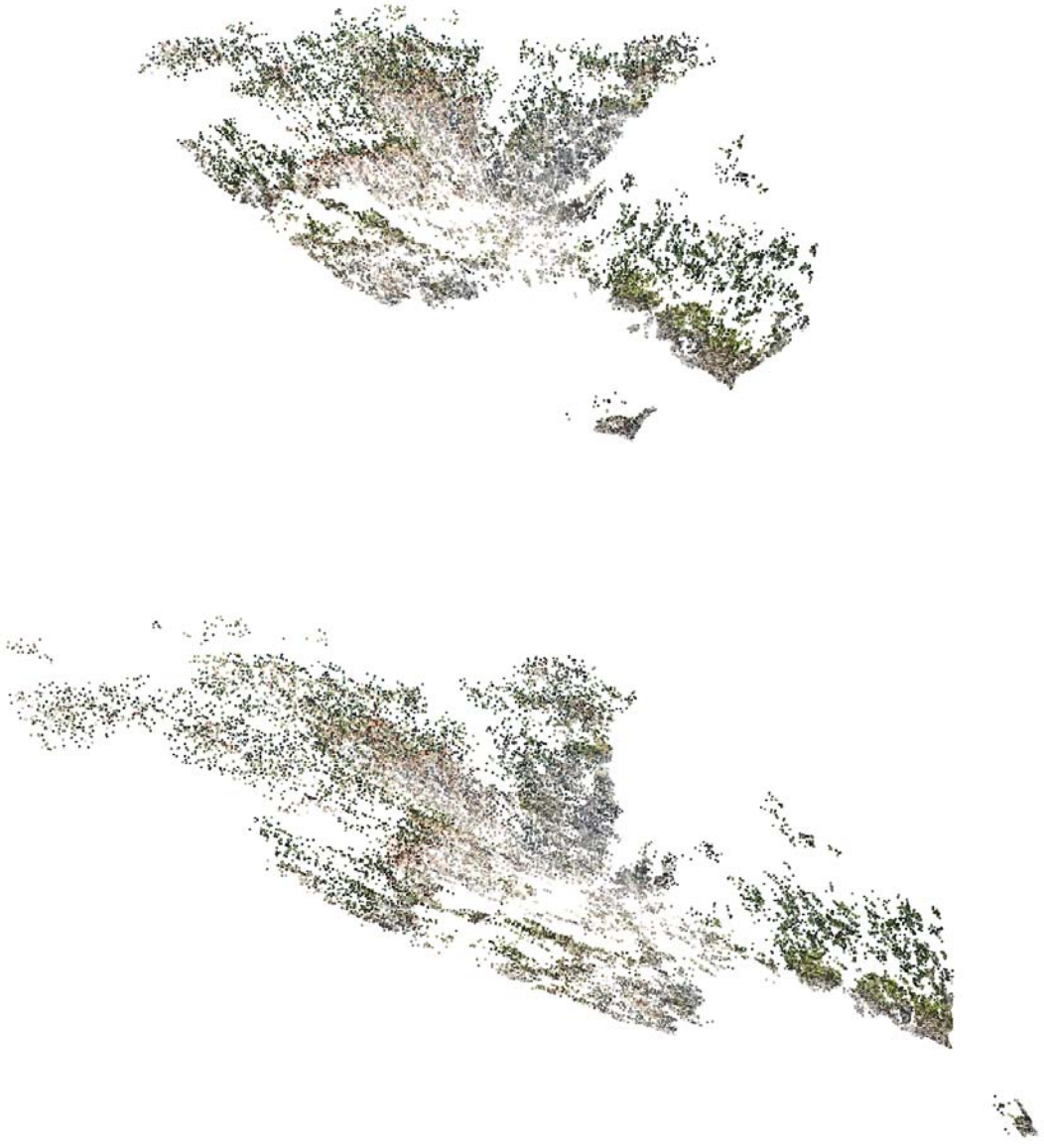

Figura 4.14: *Scena ricostruita a partire dalla stima dei parametri intrinseci.*

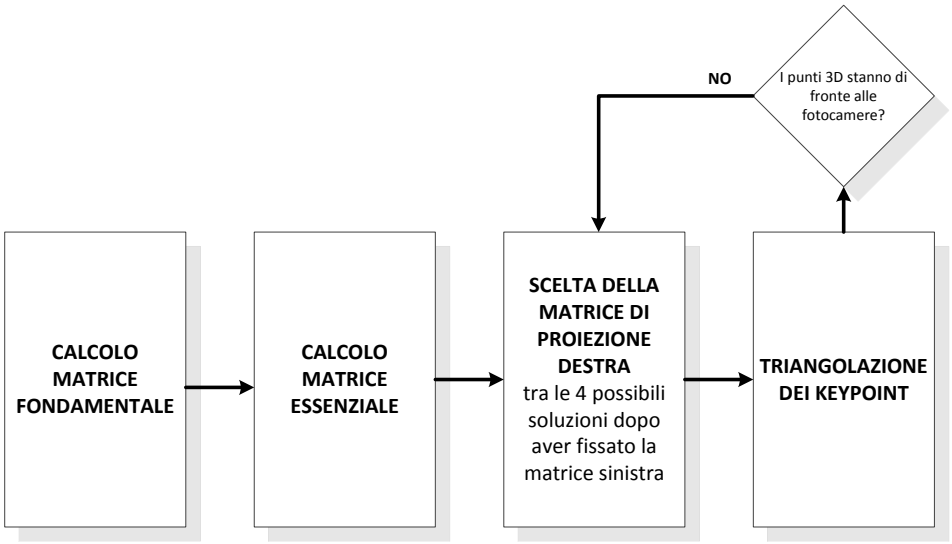

Figura 4.15: *Continuazione del diagramma di flusso della Structure From Motion nel caso in cui si utilizzano i parametri intrinseci.*

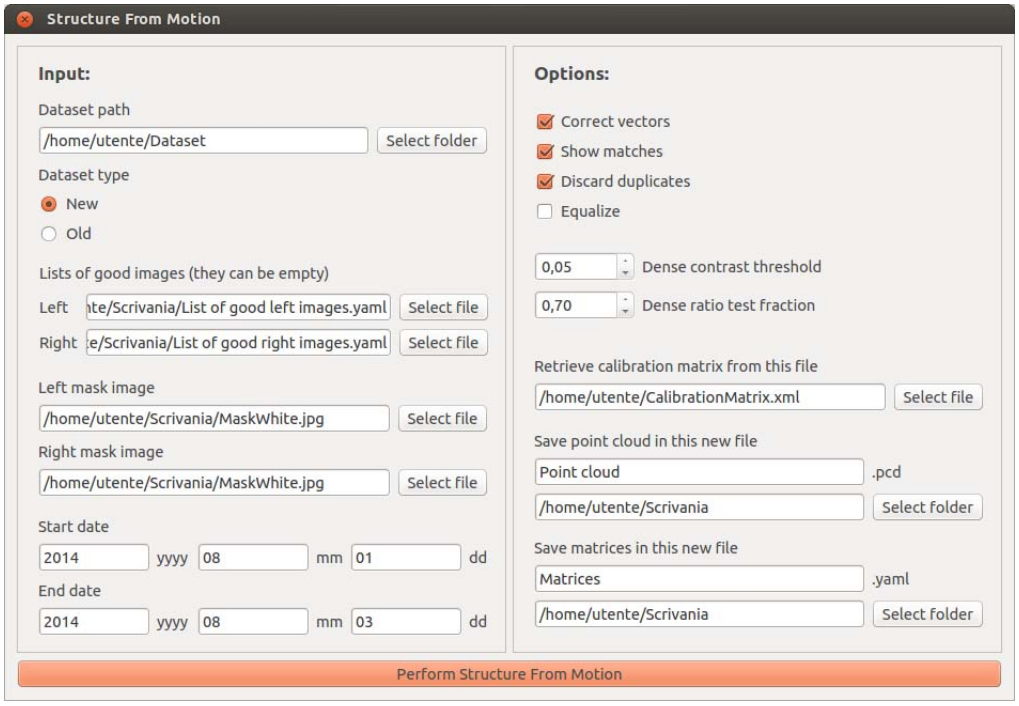

Figura 4.16: *Interfaccia grafica del programma che realizza la Structure From Motion.*

4.4.1 Sovracampionamento della nuvola di punti con PCL

Il metodo moving least squares (paragrafo 3.3.2), che stima una superficie nell'intorno di un punto, può essere impiegato per aumentare la densità della nuvola di punti che è stata ottenuta mediante il secondo approccio descritto. La libreria PCL, oltre ad implementare il filtraggio statistico (SOR), può infatti applicare il moving least squares per aumentare il numero di punti di cui è composta la nuvola. In particolare è stato scelto di approssimare la superficie attorno ad ogni punto con un polinomio di secondo grado; inoltre è stata impostata una densità di 3000 punti distribuiti casualmente ed in modo uniforme all'interno di un opportuno raggio. Il codice che realizza il sovracampionamento è riportato qui di seguito:

Codice 4.6: Codice che effettua il sovracampionamento

```
1 pcl::search::KdTree<pcl::PointXYZRGB>::Ptr tree (new pcl::search::KdTree<pcl::
      PointXYZRGB>);
2 pcl::MovingLeastSquares<pcl::PointXYZRGB, pcl::PointXYZRGB> mls;
3 // Set parameters
4 mls.setComputeNormals(false);
5 mls.setInputCloud(point_cloud_sor);
6 mls.setPolynomialFit(true);
7 mls.setSearchMethod(tree);
8 mls.setSearchRadius(1.2);
9 mls.setUpsamplingMethod(pcl::MovingLeastSquares<pcl::PointXYZRGB, pcl::PointXYZRGB>::
      RANDOM_UNIFORM_DENSITY);
10 mls.setPointDensity(3000);
11 mls.setPolynomialOrder(2);
12 // Reconstruct
13 mls.process(*point_cloud_mls);
```

Il sovracampionamento applicato alla nuvola riportata in Figura 4.14 produce il risultato di Figura 4.17.

Per intuire meglio le corrispondenze tra la scena fotografata e la scena ricostruita, si faccia riferimento alla Figura 4.18, in cui il paesaggio in tre dimensioni è visto dall'alto.

4.4.2 Riproiezione dei vettori spostamento in 3D

Per proiettare i vettori spostamento ricavati con il template matching sulla ricostruzione in tre dimensioni della frana, devono essere note le coordinate 3D nel sistema di riferimento del mondo dei punti d'inizio e fine di ogni vettore. Siccome gli spostamenti per le immagini sinistre e destre sono ricavati indipendentemente, queste informazioni non sono conosciute. La strategia che ho adottato per risolvere il problema è quella di cercare i due keypoint più vicini agli estremi di un vettore. Infatti le coordinate dei keypoint sono note per entrambe le immagini e quindi se ne può calcolare la posizione nello spazio.

Approssimando gli estremi dei vettori spostamento con i keypoint a loro più vicini, si commette un errore che è tanto maggiore quanto meno densa è la nuvola di punti. Per ottenere una buona rappresentazione dei vettori nello spazio è quindi preferibile partire da una nuvola di punti costruita a partire da un numero consistente di immagini.

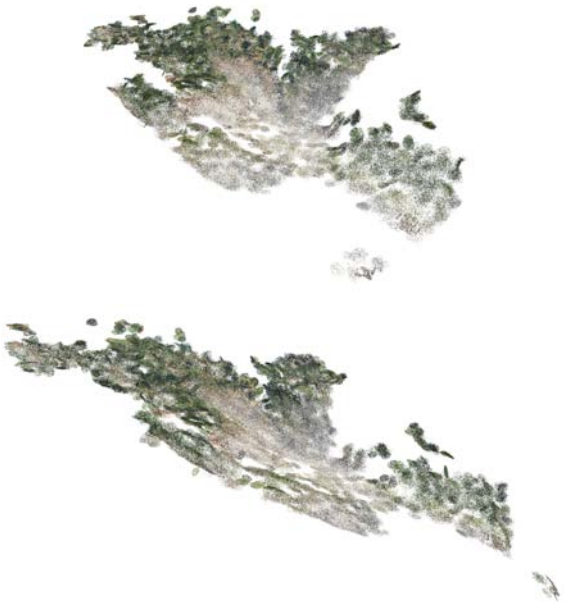

Figura 4.17: *Nuvola di punti della Figura 4.14 sovracampionata, vista approssimativamente dagli stessi due punti di vista.*

Figura 4.18: *Corrispondenze tra i punti dell'immagine scattata dalla fotocamera e la scena tridimensionale sovracampionata.*

L'interfaccia grafica del programma eseguibile che effettua il sovracampionamento e traccia i vettori spostamento nello spazio è visibile in Figura 4.19 e produce il risultato in Figura 4.20.

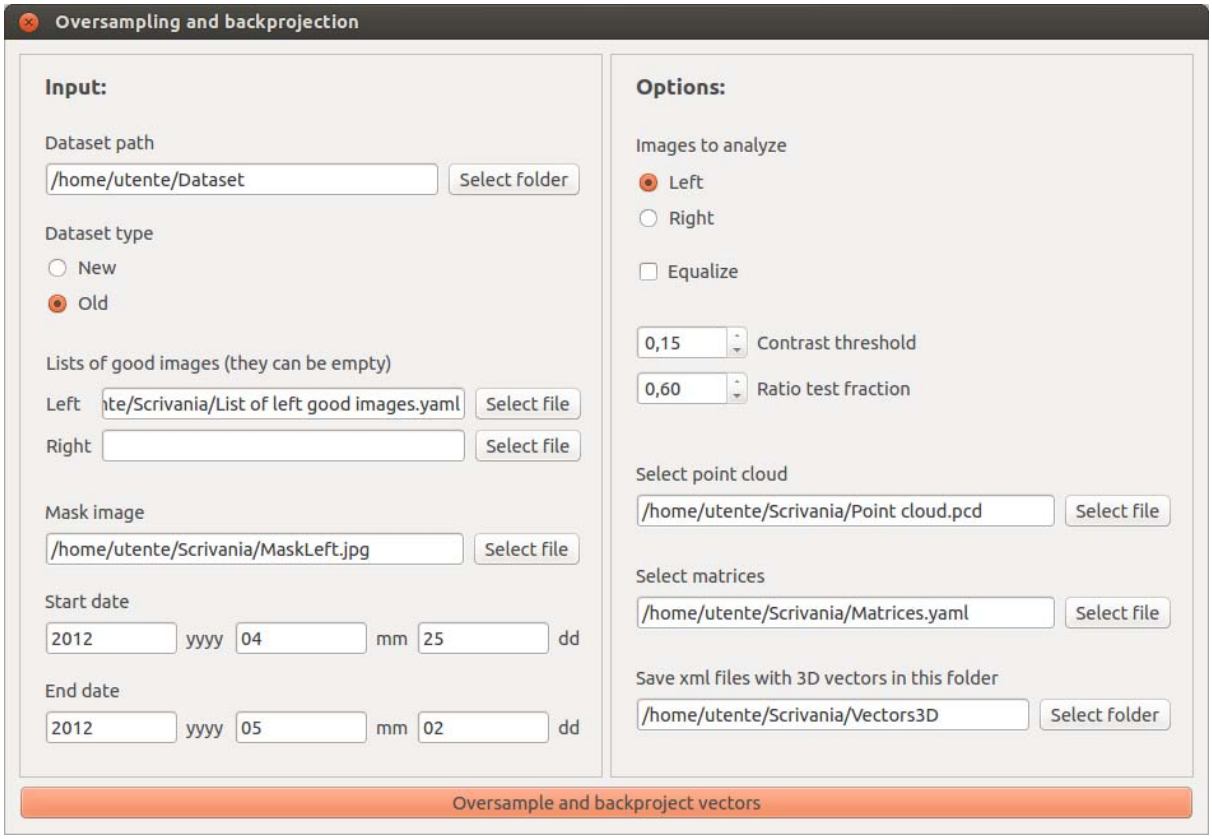

Figura 4.19: *Interfaccia grafica del programma che effettua il sovracampionamento e proietta i vettori in 3D.*

La ricostruzione in figura è stata ottenuta dal nuovo dataset; in particolare la nuvola di punti è stata ricavata dalle foto scattate tra l'1 agosto 2014 e il 3 agosto 2014 (con `dense_contrast_threshold` $= 0.05$, `dense_ratio_fraction` $= 0.7$ e rimozione dei punti duplicati); le linee blu rappresentano gli spostamenti di materiale avvenuti tra l'1 agosto 2014 e l'1 settembre 2014.

Per completare il paragrafo, nella Figura 4.21 raccolgo assieme tutti i diagrammi di flusso dettagliati esposti finora.

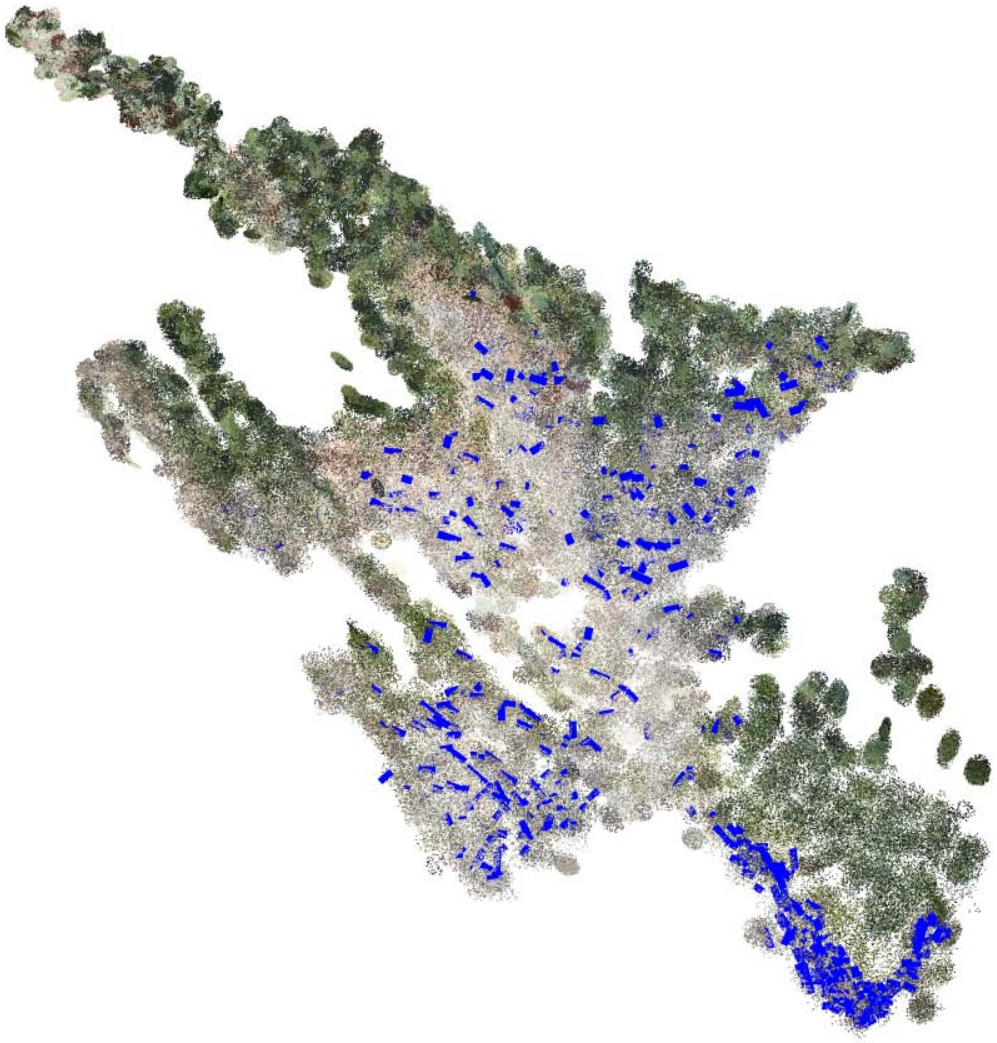

Figura 4.20: *Nuvola di punti sovracampionata con vettori spostamento sovrimpressi.*

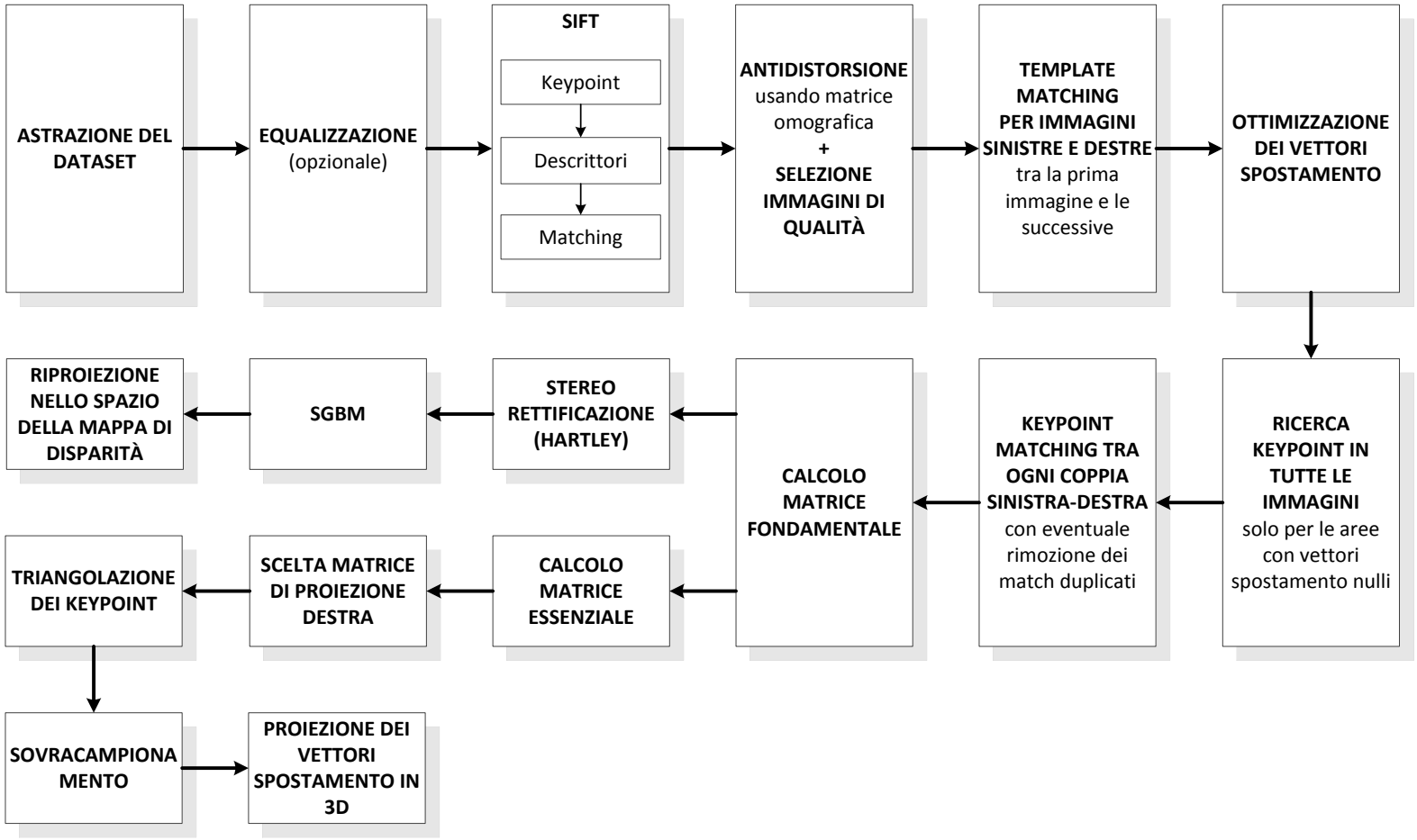

Figura 4.21: *Diagramma di flusso complessivo dettagliato.*

Capitolo 5

Considerazioni e sviluppi futuri

5.1 Considerazioni

In questo lavoro sono state presentate applicazioni software che possono rappresentare un valido contributo nelle attività di monitoraggio di ambienti naturali per scopi di prevenzione. In particolare, l'ambiente oggetto dell'analisi è la frana del Tessina, localizzata in provincia di Belluno. Per osservare il territorio si è fatto uso di una coppia di fotocamere; questa soluzione presenta il vantaggio di essere molto più economica e flessibile rispetto ad altre tecnologie, però richiede un attento studio del problema e una serie di considerazioni geometriche ed algebriche non banali.

Per tenere traccia dell'evoluzione della superficie terrestre è stato realizzato un programma che calcola i vettori spostamento relativi alla scena osservata. Questa operazione è stata realizzata sfruttando tecniche note in letteratura e ormai consolidate, come il template matching. Inoltre tali vettori sono stati ottimizzati grazie ad un algoritmo innovativo che tiene conto degli spostamenti ricavati da una sequenza di scatti consecutivi.

Nella seconda parte della tesi è stata ricostruita in tre dimensioni la scena che rappresenta la frana sfruttando due approcci distinti; il primo non fa uso di alcuna informazione sulle fotocamere utilizzate, il secondo costruisce il modello a partire da una stima dei parametri intrinseci. Si è constatato che quest'ultimo metodo fornisce risultati più affidabili, sebbene la nuvola di punti ottenuta non sia molto densa. Anche in questo caso però la ricostruzione non sempre è corretta. Infatti, la qualità della nuvola di punti dipende da svariati fattori, tra cui la bontà dei match individuati e la maschera impiegata per rimuovere gli elementi di disturbo dai fotogrammi. In generale, la qualità finale dipende fortemente dalla matrice fondamentale calcolata a partire dai match. OpenCV mette a disposizione l'algoritmo RANSAC al fine di aumentare la robustezza e l'affidabilità del calcolo di questa matrice; nonostante ciò, in alcune situazioni l'esito della ricostruzione non è del tutto ottimale. Alcuni suggerimenti per migliorare la creazione del modello tridimensionale del paesaggio sono riportati nel paragrafo 5.2.

La qualità della ricostruzione 3D e dei vettori spostamento calcolati dipende per gran parte dai dataset di immagini forniti. Purtroppo ho avuto modo di constatare che la bontà dei fotogrammi lascia spesso a desiderare; pertanto riporto nel seguito una serie

di accorgimenti che possono essere adottati al fine di ottenere un dataset maggiormente utilizzabile:

- inserire nel dataset le foto scattate ogni giorno da entrambe le fotocamere, in modo da evitare intervalli di tempo troppo lunghi non coperti da scatti;
- rimuovere la vegetazione presente nella zona antistante alle due fotocamere;
- posizionare le fotocamere in modo tale che l'intera scena sia visibile da entrambe;
- fissare meglio le fotocamere al suolo;
- sfruttare tecniche note nell'ambito della visione artificiale per ottenere dati di calibrazione e informazioni sulla distorsione degli obiettivi delle macchine fotografiche;
- applicare la tecnica HDR alle foto scattate con tempi d'esposizione diversi per ricavarne immagini qualitativamente superiori.

5.2 Sviluppi futuri

Il template matching applicato ripetutamente a varie porzioni dell'immagine ha consentito di rilevare i vettori spostamento associati al materiale di cui è composta la frana. Questa operazione è stata eseguita richiamando per ogni blocchetto di immagine la funzione `matchTemplate` di OpenCV; sebbene questa sia stata realizzata in modo efficiente dagli sviluppatori della libreria, può ancora essere risparmiata una notevole quantità di tempo di calcolo. Infatti, ogni volta che viene chiamata, la funzione esegue operazioni algebriche su pixel che erano già stati analizzati durante le iterazioni precedenti. Pertanto la complessità computazionale può essere ridotta in modo considerevole sfruttando strategie come le *integral image*; durante lo svolgimento della tesi ho preso in considerazione anche queste tecniche, ma le difficoltà riscontrate nel calcolo delle formule e nell'implementazione mi hanno portato ad orientare l'attenzione verso altri aspetti più rilevanti. L'uso delle integral image permetterebbe di determinare i vettori spostamento associati a tutti i pixel del fotogramma, e non solo ad un loro sottoinsieme, in tempi ragionevoli.

Un altro aspetto che può essere migliorato è il rilevamento delle feature da parte dell'algoritmo SIFT; durante questa attività, infatti, si è riscontrato l'uso di un'eccessiva quantità di memoria del computer sul quale viene eseguita (è necessario disporre di un computer con almeno 8 GB di memoria RAM). L'uso di altri valori per i parametri (per esempio cercare un numero inferiore di keypoint) o altri accorgimenti potrebbero ridurre lo spreco di memoria, consentendo di eseguire l'applicazione su computer dotati di risorse computazionali inferiori.

Per quanto riguarda la ricostruzione della scena in tre dimensioni, si è visto che l'approccio che fa uso dei parametri intrinseci fornisce buoni risultati. Tuttavia può capitare che il processo fallisca; inoltre sarebbe preferibile produrre una nuvola di punti più densa. Per rimediare a tali inconvenienti, potrebbe rivelarsi utile impiegare metodi di triangolazione più sofisticati, nonché disporre di dati di calibrazione e dei coefficienti

che descrivono la distorsione introdotta dagli obiettivi. Senza ricorrere a queste ulteriori informazioni, la ricostruzione può essere comunque migliorata ricorrendo a tecniche come il *bundle adjustment*. Questa consente infatti di risolvere problemi di ottimizzazione, con lo scopo di raffinare le coordinate dei punti del modello 3D, i parametri intrinseci delle fotocamere, la loro posizione reciproca e la distorsione. Tenendo conto dell'errore di riproiezione si possono inoltre mettere in pratica alcune accortezze, come ad esempio scartare i punti che presentano un errore superiore alla media.

Per concludere la trattazione, si può affermare che il progetto realizzato ha permesso di semplificare e automatizzare le attività di supporto alla prevenzione di catastrofi naturali. Sebbene il lavoro necessiti di qualche perfezionamento, l'applicazione della visione artificiale a questo tipo di attività rappresenta di per sé un passo avanti rispetto agli approcci adottati finora, a dimostrazione dell'importanza che questa disciplina riscuote sempre più in molti campi della ricerca scientifica.

Appendice A

Documentazione

Questa appendice spiega come fare buon uso delle librerie e dei programmi eseguibili scritti. Dapprima si riporta un elenco completo di file, suddivisi per moduli, poi ciascuno di essi viene chiarito nei paragrafi successivi. Per una descrizione dettagliata di ogni singolo metodo si faccia riferimento alla documentazione generata con il tool Doxygen allegata al codice sorgente.

- *dataset*
  - `dataset`: contiene la classe base astratta che definisce le interfacce che un dataset deve esporre.
  - `dataset_1`: contiene la classe derivata che rappresenta il vecchio dataset.
  - `dataset_2`: contiene la classe derivata che rappresenta il nuovo dataset.
- *preprocessing*
  - `equalization`: equalizzazione di un'immagine.
  - `sift_extraction`: rilevazione dei keypoint, calcolo dei descrittori e feature-based template matching.
  - `undistortion`: gestione delle matrici omografiche per antidistorcere un'immagine.
  - `automatic_images_seletion`: (eseguibile con interfaccia grafica) selezione delle immagini di qualità in un dataset.
- *processing*
  - `template_matching`: template-based matching per ricavare i vettori spostamento e loro ottimizzazione.
  - `images_template_matching`: (eseguibile con interfaccia grafica) template matching con correzione dei vettori spostamento e loro visualizzazione in due dimensioni.

  - `structure_from_motion`: contiene la classe base astratta contenente i metodi in comune ai due approcci di ricostruzione adottati.
  - `structure_from_motion_uncalibrated`: contiene la classe derivata che crea la scena in tre dimensioni senza impiegare parametri intrinseci.
  - `structure_from_motion_semicalibrated`: contiene la classe derivata che crea la scena in tre dimensioni a partire da una stima dei parametri intrinseci.
  - `main_uncalibrated`: (eseguibile) avvia il processo di ricostruzione in tre dimensioni della frana senza impiegare parametri intrinseci.
  - `exe_sfm_semicalibrated`: (eseguibile con interfaccia grafica) avvia il processo di ricostruzione in tre dimensioni della frana facendo uso dei parametri intrinseci.

- *postprocessing*
  - `backprojection_and_mls`: (eseguibile con interfaccia grafica) carica da file una nuvola di punti che rappresenta la frana e altri dati, fa un sovracampionamento e traccia i vettori spostamento 3D in sovrimpressione.

A.1 Il modulo *dataset*

Il modulo *dataset* è costituito da una sola libreria, contenente i file `dataset.h`, `dataset.cpp`, `dataset_1.h`, `dataset_1.cpp`, `dataset_2.h` e `dataset_2.cpp`. Lo scopo delle classi contenute in questo modulo è quello di fornire un'astrazione dei dataset memorizzati su file system contenenti immagini ed altre informazioni. Il file `dataset.cpp` contiene la dichiarazione della classe `Dataset`; tra le sue funzionalità ricordo la possibilità di restituire una sequenza di immagini o di coppie presenti nel dataset, e di cercare i file contenenti le matrici omografiche.

Un'immagine è codificata attraverso la struttura `ImageInfo`, contenente il percorso (`path`, gestito con le librerie boost di C++) della foto, il percorso del file contenente keypoint e descrittori salvati, e la data di scatto (`date`, gestita sempre con le librerie boost). Una coppia di immagini (sinistra e destra) è invece rappresentata per mezzo della struttura `StereoPairInfo`, contenente un campo `ImageInfo` per ciascuna delle due immagini e la relativa data di scatto.

Il compito più importante delle classi derivate `Dataset_1` e `Dataset_2` è quello di realizzare i metodi `initialize`, che analizzano il dataset memorizzato nel file system e ne codificano il contenuto all'interno di un dizionario (sono presenti due versioni, una senza parametri e una con due parametri d'ingresso). Questo è il primo metodo che deve essere invocato dopo aver creato un esemplare di `Dataset`; tutti gli altri metodi fanno riferimento al dizionario creato dai metodi `initialize`. I costruttori delle classi derivate da `Dataset` dovrebbero ricevere un solo parametro, il path del dataset; si assume che tutti i file di cui il dataset è composto siano contenuti al suo interno e che abbiano una struttura predefinita.

A.2 Il modulo *preprocessing*

Il modulo *preprocessing* è costituito da un'omonima libreria, contenente i file `equalization.h`, `equalization.cpp`, `sift_extraction.h`, `sift_extraction.cpp`, `undistortion.h` e `undistortion.cpp`; sono inoltre presenti i file `automatic_images_selection.h` e `automatic_images_selection.cpp`, ed il relativo `main.cpp`.

A.2.1 La libreria `preprocessing`

`equalization.cpp`

Contiene la definizione della classe `Equalization`, che mette a disposizione due metodi per l'equalizzazione delle immagini: uno effettua l'equalizzazione canale per canale, l'altro solo della luminanza, così viene preservato il colore.

`sift_extraction.cpp`

Contiene la classe `SiftExtraction` che, facendo uso dei metodi di OpenCV, mette a disposizione le seguenti funzionalità:

- rilevamento dei keypoint;
- calcolo dei relativi descrittori;
- feature-based template matching con ratio test;
- salvataggio o caricamento di keypoint e descrittori da file.

`undistortion.cpp`

Contiene la classe `Undistortion`, che consente di:

- antidistorcere un'immagine data una matrice omografica;
- trovare la matrice omografica che permetta di sovrapporre due foto scattate da punti di vista leggermente diversi;
- salvare una matrice omografica su file.

A.2.2 L'eseguibile `AutomaticImagesSelection`

Il modulo ***preprocessing*** include il file `main.cpp`, contenente il metodo `main` che crea un esemplare di `AutomaticImagesSelection`. Questo procedimento un po' macchinoso è previsto dalla libreria Qt, che permette di creare l'interfaccia grafica del software. L'applicazione salva in un file con estensione `.yaml` il nome delle immagini che superano il test di qualità. Una volta avviata, compare la finestra già riportata in Figura 4.4.

Input

- Il path del dataset.
- La tipologia del dataset; l'implementazione attuale prevede la gestione di due tipi di dataset: il vecchio e il nuovo.
- Il nome del file `.yaml` in cui inserire l'elenco delle foto di qualità e il percorso della cartella in cui dev'essere contenuto.
- La maschera da utilizzare per nascondere le parti che possono portare a dei match errati (come la vegetazione).
- La data di inizio dell'analisi.
- La data di fine dell'analisi.
- La fotocamera di cui si vogliono analizzare gli scatti (sinistra o destra).
- Alcune opzioni come la visualizzazione dei match o delle altre immagini intermedie e l'eventuale equalizzazione (sconsigliata se non necessaria).
- Alcuni parametri specifici come il numero minimo di match per considerare la seconda immagine di buona qualità, il valore di contrasto per la ricerca dei keypoint e la frazione per il ratio test. Quando si seleziona la fotocamera (sinistra o destra) vengono reimpostati i valori di default per questi parametri.

Ricordo che, nel caso in cui il programma venga eseguito più volte, i keypoint ed altre informazioni vengono lette dai file salvati nelle esecuzioni precedenti. Pertanto se si modificano alcune opzioni, come l'equalizzazione, il contrasto o altro, bisogna prima cancellare questi file da disco, altrimenti vengono prelevate le informazioni ottenute utilizzando i vecchi parametri. Questa osservazione vale anche per i programmi eseguibili descritti nel seguito.

Output

Il programma analizza una per una le immagini comprese nell'intervallo temporale indicato. In uscita produce un file contenente l'elenco delle immagini considerate di qualità. Questo file viene poi utilizzato durante la fase di inizializzazione del dataset nelle fasi successive (processing e postprocessing). Per questa ragione, il programma deve essere eseguito prima di procedere con la ricerca dei vettori spostamento e la ricostruzione della scena 3D; se si saltasse questo passaggio, nelle fasi successive si potrebbero riscontrare problemi, dato che le elaborazioni sarebbero condotte anche sulle immagini di scarto.

A.3 Il modulo *processing*

Il modulo *processing* comprende le librerie `template_matching` e `structure_from_motion`; la prima è composta dai file `template_matching.h` e `template_matching.cpp`, la seconda dai file `structure_from_motion.h`, `structure_from_motion.cpp`, `structure_from_motion_uncalibrated.h`, `structure_from_motion_uncalibrated.cpp`, `structure_from_motion_semicalibrated.h` e `structure_from_motion_semicalibrated.cpp`.

Inoltre sono presenti i file `images_template_matching.cpp`, `main_uncalibrated.cpp`, `exe_sfm_semicalibrated.cpp` e altri di minor importanza.

A.3.1 La libreria `template_matching`

Contiene la definizione dei metodi della classe `TemplateMatching`; questi consentono di realizzare il template-based matching per individuare l'evoluzione del terreno, effettuare l'ottimizzazione dei vettori spostamento e disegnarli in due dimensioni sovrapposti a un'immagine.

A.3.2 La libreria `structure_from_motion`

`structure_from_motion.cpp`

Contiene la classe base `StructureFromMotion`, che definisce le funzionalità in comune alle le classi `StructureFromMotionUncalibrated` e `StructureFromMotionSemicalibrated`. Il suo compito è quello di determinare tutti i match tra feature delle coppie di immagini e di inserirli in un unico pool; per identificare le zone in cui si sono verificati movimenti di materiale, effettua preventivamente il template matching.

`structure_from_motion_uncalibrated.cpp`

La classe derivata `StructureFromMotionUncalibrated` genera la nuvola di punti della frana a partire dai match generati dalla classe base. Per ottenere questo risultato si basa sull'approccio che non fa uso dei parametri intrinseci delle fotocamere. La nuvola può essere ispezionata con il visualizzatore di PCL.

`structure_from_motion_semicalibrated.cpp`

La classe derivata `StructureFromMotionSemicalibrated` genera la nuvola di punti della frana a partire dai match generati dalla classe base. Per ottenere questo risultato si basa sull'approccio che fa uso dei parametri intrinseci stimati delle fotocamere. La nuvola può essere ispezionata con il visualizzatore di PCL; inoltre viene salvata su file assieme ad altre informazioni per consentire successivamente al modulo *postprocessing* di stamparci in sovrimpressione i vettori spostamento 3D.

A.3.3 L'eseguibile `ImagesTemplateMatching`

Richiama i metodi della classe `TemplateMatching` con lo scopo di individuare i vettori spostamento a partire da più immagini scattate con la stessa fotocamera; tali vettori vengono corretti e tracciati in 2D sopra le immagini. I parametri forniti all'applicazione hanno significati simili a quelli dell'eseguibile del modulo preprocessing (si veda la Figura 4.9).

Input

- Il path del dataset.
- La tipologia del dataset; l'implementazione attuale prevede la gestione di due tipi di dataset: il vecchio e il nuovo.
- I file `.yaml` prodotti durante il preprocessing, che contengono l'elenco di immagini sinistre e destre di qualità soddisfacente (opzionali).
- La maschera da utilizzare per nascondere le parti che possono portare a dei match errati (come la vegetazione).
- La data di inizio dell'analisi.
- La data di fine dell'analisi.
- La fotocamera di cui si vogliono analizzare gli scatti (sinistra o destra).
- Altre opzioni.
- Il punteggio minimo che devono ottenere i vettori per poter essere considerati accettabili; il valore del contrasto e della frazione per il ratio test utilizzati per l'allineamento preliminare delle immagini.
- Il path della cartella in cui salvare i vettori 2D individuati (viene creato un file con estensione `.xml` per ogni immagine).

Output

Dopo aver trovato i vettori spostamento e averli corretti, le immagini esaminate vengono mostrate in sequenza con tali vettori sovrimpressi in 2D; i vettori sono anche salvati su disco nel path indicato.

A.3.4 Gli eseguibili `SFMUncalibrated` e `SFMSemicalibrated`

Poiché l'applicazione `SFMUncalibrated` si è dimostrata di minor utilità rispetto a `SFMSemicalibrated`, solo quest'ultima è stata corredata di un'interfaccia grafica; segue ora una sua descrizione (si veda la Figura 4.16).

Input

- Il path del dataset.
- La tipologia del dataset; l'implementazione attuale prevede la gestione di due tipi di dataset: il vecchio e il nuovo.
- I file `.yaml` prodotti durante il preprocessing, che contengono l'elenco di immagini sinistre e destre di qualità soddisfacente (opzionali).
- Le maschere da utilizzare per nascondere le parti che possono portare a dei match errati (come la vegetazione in movimento); si consiglia di impiegare una maschera che non nasconda una porzione troppo estesa dell'immagine, altrimenti aumenta la probabilità che la ricostruzione ottenuta sia errata. In un'implementazione futura è auspicabile usare maschere diverse per la fase preliminare di allineamento delle immagini e per quelle successive di template matching e ricostruzione 3D.
- La data di inizio dell'analisi.
- La data di fine dell'analisi.
- Altre opzioni (si consiglia di lasciare i segni di spunta di default).
- I valori di contrasto e ratio test applicati ai keypoint che costituiscono il modello della frana. In un'implementazione futura è auspicabile introdurre la scelta di questi parametri anche per la fase preliminare di allineamento.
- Il file contenente la matrice di calibrazione (attualmente viene usata la stessa per le due fotocamere).
- Il nome e la cartella in cui salvare la nuvola di punti ottenuta.
- Il nome e la cartella in cui salvare le matrici necessarie in seguito.

Output

Entrambe le applicazioni generano e visualizzano la nuvola di punti tridimensionale che rappresenta la frana. `SFMSemicalibrated` salva la nuvola e altre informazioni rispettivamente in un file `pcd` e un file `yaml`; questi dati vengono sfruttati in seguito dal modulo *postprocessing*, che traccia i vettori spostamento 3D sopra la nuvola salvata.

A.4 Il modulo *postprocessing*

Questo modulo comprende i file `backprojection_and_mls.h`, `backprojection_and_mls.cpp` e `main_backprojection.cpp`. Essi costituiscono il programma eseguibile `Backprojection` (si veda la Figura 4.19). Il suo compito è quello di effettuare il sovracampionamento della nuvola di punti usando il metodo moving least squares e di tracciare i vettori spostamento 3D in sovrimpressione alla nuvola di punti caricata da file.

Input

- Il path del dataset.
- La tipologia del dataset; l'implementazione attuale prevede la gestione di due tipi di dataset: il vecchio e il nuovo.
- I file `.yaml` prodotti durante il preprocessing, che contengono l'elenco di immagini sinistre e destre di qualità soddisfacente (opzionali).
- La maschera da utilizzare per nascondere le parti che possono portare a dei match errati (come la vegetazione in movimento); la maschera è usata solo per realizzare il template matching, non per la ricostruzione 3D, pertanto non deve essere necessariamente uguale a quella usata in `SFMUncalibrated` e `SFMSemicalibrated`.
- La data di inizio dell'analisi.
- La data di fine dell'analisi.
- La fotocamera di cui si vogliono analizzare gli scatti (sinistra o destra).
- I valori di contrasto e ratio test per l'allineamento delle immagini, e altre opzioni.
- La nuvola di punti con estensione `.pcd` da sovracampionare e sopra la quale disegnare i vettori spostamento.
- Il file `.yaml` prodotto durante il *processing* e relativo alla nuvola di punti caricata.
- Il path della cartella in cui salvare i vettori 3D individuati (viene creato un file con estensione `.xml` per ogni immagine).

Output

La nuvola di punti caricata da file, sovracampionata e con i vettori spostamento ricavati col template matching sovrimpressi; i vettori 3D sono anche salvati su disco nel path indicato. I vettori mostrati sono quelli che esprimono il movimento di materiale avvenuto tra il primo e l'ultimo giorno dell'intervallo esaminato ottenuti con la fotocamera selezionata. La nuvola può essere ruotata ed ispezionata a piacere tramite il visualizzatore di PCL.

Elenco delle figure

Elenco delle tabelle

Bibliografia

[1] Documentazione di OpenCV. `http://docs.opencv.org/`.

[2] D.L. Baggio, D.M. Escrivá, N. Mahmood, R. Shilkrot, S. Emami, K. Ievgen, and J. Saragih. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing Ltd, 2012.

[3] J.-Y. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`.

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

[5] T.F. Cootes and C.J. Taylor. Statistical models of appearance for medical image analysis and computer vision. In *Medical Imaging 2001*, pages 236–248. International Society for Optics and Photonics, 2001.

[6] M.A. Fischler and R.C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[7] D.A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[8] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[9] R.I. Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, 1999.

[10] R.I. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[11] R.I. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] M. Hebert, T. Kanade, and I. Kweon. *3-D Vision Techniques for Autonomous Vehicles*. Springer, 1990.

[13] J. Huang, X. Pan, X. Peng, Y. Yuan, C. Xiong, J. Fang, and F. Yuan. Digital image correlation with self-adaptive gaussian windows. *Experimental Mechanics*, 53(3):505–512, 2013.

[14] T. Iwanami and T. Goto. An adaptive contrast enhancement using regional dynamic histogram equalization. *2012 IEEE International Conference on Consumer Electronics (ICCE)*, pages 719–722, 2012.

[15] H. Jia, Y.L. Murphey, J. Shi, and T.-S. Chang. An intelligent real-time vision system for surface defect detection. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 239–242. IEEE, 2004.

[16] C.J. Kähler, S. Scharnowski, and C. Cierpka. On the resolution limit of digital particle image velocimetry. *Experiments in fluids*, 52(6):1629–1639, 2012.

[17] David Levin. The approximation power of moving least-squares. *Mathematics of Computation of the American Mathematical Society*, 67(224):1517–1531, 1998.

[18] J. Liu and M. Iskander. Adaptive cross correlation for imaging displacements in soils. *Journal of computing in civil engineering*, 18(1):46–57, 2004.

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[20] M. Magnabosco and T.P. Breckon. Cross-spectral visual simultaneous localization and mapping (slam) with sensor handover. *Robotics and Autonomous Systems*, 61(2):195–208, 2013.

[21] Sarif Kumar Naik and C.A. Murthy. Hue-preserving color image enhancement without gamut problem. *Image Processing, IEEE Transactions on*, 12(12):1591–1598, 2003.

[22] G.-H. Park, H.-H. Cho, and M.-R. Choi. A contrast enhancement method using dynamic range separate histogram equalization. *IEEE Transactions on Consumer Electronics*, 54(4):1981–1987, 2008.

[23] A.K. Prasad. Stereoscopic particle image velocimetry. *Experiments in fluids*, 29(2):103–116, 2000.

[24] A.K. Prasad and R.J. Adrian. Stereoscopic particle image velocimetry applied to liquid flows. *Experiments in Fluids*, 15(1):49–60, 1993.

[25] P. Sturm. Some lecture notes on geometric computer vision. INRIA Grenoble - Rhône-Alpes.

[26] W. Wei, H. Jun, and T. Yiping. Image matching for geomorphic measurement based on sift and ransac methods. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 2, pages 317–320. IEEE, 2008.

[27] D.J. White, W.A. Take, and M.D. Bolton. Measuring soil deformation in geotechnical models using digital images and piv analysis. In *10th International Conference on Computer Methods and Advances in Geomechanics, Tucson, Arizona*, pages 997–1002, 2001.

[28] C.E. Willert. Adaptive piv processing based on ensemble correlation. In *14th international symposium on applications of laser techniques to fluid mechanics, July*, pages 07–10, 2008.

[29] Z. Xu, H. R. Wu, X. Yu, and B. Qiu. Colour image enhancement by virtual histogram approach. *IEEE Transactions on Consumer Electronics*, 56(2):704–712, 2010.

[30] B.-x. Yuan, W.-w. Chen, T. Jiang, Y.-x. Wang, and K.-p. Chen. Stereo particle image velocimetry measurement of 3d soil deformation around laterally loaded pile in sand. *Journal of Central South University*, 20:791–798, 2013.