

UNIVERSITÀ DEGLI STUDI DI GENOVA
FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
ANNO ACCADEMICO 2009–2010



TESI DI LAUREA

STUDIO E SVILUPPO DI SISTEMI DI ODOMETRIA VISUALE PER
L'ESPLORAZIONE PLANETARIA

07/05/2010

Relatore: *Chiar.^{mo} Prof.* Giuseppe Casalino
Correlatori: *Dott.Ing.* Enrico Simetti
Dott.Ing. Enrica Zereik

Candidati: Fabio Frassinelli
Davide Ducco

*Realizzato presso DIST – Dipartimento di Informatica Sistemistica e Telematica
Laboratorio G.R.A.A.L. – Genoa Robotic And Automation Laboratory*

Indice

1 Introduzione	3
2 L'odometria visuale	13
2.1 L'estrazione di feature	16
2.1.1 I <i>feature detector</i>	17
2.1.2 Criteri per un buon estrattore	19
2.1.3 L'Harris detector	21
2.1.4 Il SURF	27
2.2 Il matching	38
2.2.1 La rettificazione delle immagini	38
2.2.2 Il matching correlation-based	43
2.2.3 Il matching descriptor-based	47
2.3 La triangolazione stereoscopica	48
2.4 Il tracking	55
2.5 La stima	58
3 L'algoritmo di stima	59
3.1 Procedura di Stima	59
3.2 Stima ai Minimi Quadrati	60
3.3 Maximum Likelihood Estimation	62
3.4 Composizione matrici di trasformazione	68

4 Visual Odometry @GRAAL	70
4.1 Mars Exploration Rovers	71
4.1.1 MER visual odometry	72
4.1.2 Le modifiche proposte da MSL	77
4.2 SURF-based visual odometry	86
4.2.1 Overview dell'algoritmo	86
4.2.2 L'estrazione delle feature	88
4.2.3 Il matching e la triangolazione	89
4.2.4 Il tracking fra i keyframe	91
5 Prove Sperimentali	93
5.1 Setup di riferimento	94
5.1.1 Il sistema di telecamere stereo	95
5.1.2 Il rover utilizzato indoor	97
5.1.3 Il rover utilizzato outdoor	99
5.2 I test indoor	103
5.3 I test outdoor	107
6 Conclusioni e sviluppi futuri	113
Bibliografia	116

Elenco delle figure

1.1	Il rover Sojourner	9
1.2	Il rover Spirit	10
1.3	Il percorso del rover Spirit	11
2.1	Le fasi dell'algoritmo di odometria visuale	15
2.2	Il problema dell' <i>apertura</i>	16
2.3	Un pattern univocamente identificabile	17
2.4	Harris corner (a) e SURF detector (b)	19
2.5	Porzioni d'immagine adatte per la collocazione di punti d'interesse	20
2.6	Difficoltà di localizzazione di un corner ripetitivo	21
2.7	(c)-(e) Diagrammi di distribuzione dei gradienti per le finestre indicate in (a). Le posizioni dei corner rilevate sono mostrate in (b).	24
2.8	Classificazione del contenuto immagine secondo l'opeartore di Harris	25
2.9	Rilevazione del corner tramite finestra semplice e finestra pesata	26
2.10	Una funzione di peso gaussiana	27
2.11	Calcolo di una finestra tramite l'utilizzo dell'immagine integrale	29
2.12	Approssimazioni di una Gaussiana (le zone in grigio equivalgono a zero)	31
2.13	Analisi dello spazio dei fattori di scala in differenti versioni	32
2.14	Le dimensioni dei filtri in base alle ottave e ai livelli di scala	33

2.15 Soppressione non-maxima nello spazio e nella scala	34
2.16 Calcolo dell'orientazione in base alle risposte al filtro Haar	36
2.17 Esempio di come sono costruite le finestre per il calcolo dei descrittori secondo l'orientazione assegnata	36
2.18 Componenti del descrittore calcolati su ogni sotto-finestra	37
2.19 Un esempio di matching tra immagini stereo	39
2.20 La geometria epipolare	41
2.21 La geometria epipolare rettificata	41
2.22 Immagini stereo originali e rettificate	42
2.23 Un esempio di occlusione	46
2.24 Il sistema di coordinate stereo	50
2.25 Geometria delle stereocamere	50
2.26 L'errore di quantizzazione	52
2.27 Approssimazione dell'errore di quantizzazione	54
2.28 Un esempio di tracking catturato da un test di prova (notare la presenza di outlier descritti in seguito)	56
3.1 La densità dell'errore rispetto alla distanza del landmark	68
4.1 Costruzione della finestra di ricerca	74
4.2 Il matching attraverso la piramide d'immagini	75
4.3 La diversa interpretazione dell'algoritmo	78
4.4 Costruzione delle finestre di ricerca per il tracking	81
4.5 Differenza tra le finestre di tracking dei due algoritmi	81
5.1 Il sistema di telecamere stereo <i>Bumblebee2</i>	95
5.2 Il rover robuLAB 80	98
5.3 La Bumblebee2 montata sul rover	98
5.4 Vista prospettica del telaio, misure in mm	99

5.5	Il rover utilizzato per le prove outdoor	101
5.6	Il rover con il sistema di aquisizione stereo	102
5.7	Un percorso chiuso stimato dall'algoritmo	103
5.8	Percorso chiuso del rover outdoor	108

Elenco delle tabelle

4.1	Tempi di estrazione del SURF e dell'U-SURF	89
5.1	Le caratteristiche tecniche della Bumblebee2	96
5.2	Gli errori commessi dall'algoritmo con LOG e Hessian threshold 350	105
5.3	Gli errori commessi dall'algoritmo senza LOG e Hessian threshold 350	105
5.4	Gli errori commessi dall'algoritmo con LOG e Hessian threshold 500	106
5.5	Gli errori commessi dall'algoritmo con LOG e Hessian threshold 700	106
5.6	Gli errori commessi dall'algoritmo con LOG e Hessian threshold 350	109
5.7	Gli errori commessi dall'algoritmo senza LOG e Hessian threshold 350	109
5.8	Gli errori commessi dall'algoritmo con LOG e Hessian threshold 400	110
5.9	Gli errori commessi dall'algoritmo senza LOG e Hessian threshold 400	110
5.10	Gli errori commessi dall'algoritmo con LOG e Hessian threshold 500	111
5.11	Gli errori commessi dall'algoritmo con LOG e Hessian threshold 700	111

Prefazione

L'oggetto di questa tesi è stato lo studio e lo sviluppo di tecniche di odometria visuale per la localizzazione di rover nell'ambito dell'esplorazione spaziale.

L'idea del progetto è nata dalla collaborazione stipulata tra *Thales Alenia Space* e il laboratorio *GRAAL* dell'Università di Genova, riguardante lo sviluppo di sistemi robotici e relativi algoritmi di controllo per validare e dimostrare il loro eventuale utilizzo per effettivi sistemi di volo.

Partendo dalle tecniche esistenti allo stato dell'arte, se ne è cercata una che, oltre a rivelarsi piuttosto affidabile, robusta e non eccessivamente pesante dal punto di vista computazionale, potesse essere utilizzata in maniera continuativa, anche durante il movimento del rover, e che fosse totalmente indipendente da una stima iniziale.

Per far ciò, si è provveduto ad un'analisi esaustiva della letteratura in materia, confrontando le diverse tecniche proposte in passato, cercando di evidenziarne limiti e pregi, sempre tenendo ben presente i vincoli operazionali imposti dall'ambito in cui l'applicazione deve operare.

La nostra attenzione si è focalizzata sul progetto MER, tuttora in corso, dell'agenzia spaziale americana NASA, il quale ha dotato i due rover Spirit e Opportunity di un sistema di odometria visuale, da essere tuttavia utilizzato non in maniera continuativa, ma solo in casi particolari, per aiutare il sistema di localizzazione globale.

Si è tentato di replicare questo sistema, introducendo sia i miglioramenti pro-

posti dallo stesso laboratorio americano per la propria missione futura (MSL), sia alcune modifiche da noi proposte per cercare di raggiungere livelli di affidabilità e tempistica sufficientemente adeguati, riscontrando però alcune difficoltà nell'ottenere risultati soddisfacenti sotto i vincoli operazionali inizialmente prefissati.

Di conseguenza, si è sviluppato un ulteriore sistema basato su un approccio radicalmente diverso in quasi tutte le fasi del processo di odometria visuale, conservando solo l'algoritmo di stima finale identico a quello precedente, in modo da cercare di realizzare un sistema più robusto.

Dopo alcuni test svolti sia in ambiente indoor che in ambiente outdoor, i risultati ottenuti da quest'ultima tecnica si sono rivelati abbastanza soddisfacenti, sia dal punto di vista dell'affidabilità della stima, sia dal punto di vista del peso computazionale.

Capitolo 1

Introduzione

La visione è un senso straordinario e potente.

L'abilità di percepire l'ambiente circostante è una capacità fondamentale per ogni essere umano per poter interagire con il mondo esterno. Per l'uomo, così come per gran parte degli esseri viventi, il sistema visivo rappresenta uno dei sensi più sviluppati, soprattutto per quanto riguarda la percezione della propria posizione e la navigazione nello spazio.

Da qualche decennio la visione sta assumendo sempre più importanza anche nel mondo della robotica e negli ultimi anni comincia ad ottenere risultati significativi, anche grazie al progresso della tecnologia riguardante l'acquisizione e l'elaborazione delle immagini.

I campi di applicazione della visione artificiale sono molteplici e tutti di grande interesse per la ricerca odierna: video sorveglianza, modellazione di oggetti e ambienti, interazione tra uomo e calcolatore, etc. Tra tutti questi utilizzi, quello che ultimamente sta riscontrando una grande attenzione è quello orientato al controllo di veicoli autonomi, sia indoor che outdoor, in modo da favorirne la localizzazione e semplificare il difficile task della navigazione.

Il problema della localizzazione di un sistema autonomo è di importanza fondamentale in quanto necessario per la navigazione, tuttavia il problema è che con

le tecniche finora utilizzate (i.e. odometria meccanica) si potrebbero commettere errori rilevanti e le azioni compiute dal robot potrebbero essere portate a termine con esito negativo. Molto spesso, infatti, tali sistemi assolvono a compiti delicati, perciò è importante disporre di dati completi e robusti su cui basare algoritmi di decisione, in modo che siano a loro volta robusti.

L'odometria visuale è proprio il processo atto a determinare, attraverso l'utilizzo di un sensore visivo, l'orientazione e la posizione di un robot mobile nello spazio 3D a partire da una sequenza di immagini. Essa può essere considerata la parte di localizzazione di un sistema globale di SLAM (*Simultaneous Localization And Mapping*). Naturalmente gli input visivi non sono adatti solo per la navigazione, ma permettono anche lo svolgimento di numerosi altri task. Per esempio, oltre a orientarsi nell'ambiente circostante, un robot potrebbe generare una ricostruzione 3D, rilevare oggetti d'interesse, classificare il terreno, etc.

Nella ricerca robotica, l'utilizzo della visione per scopi di navigazione terrestre è cominciato sul finire degli anni settanta. Un primo tentativo si trova negli studi compiuti da Moravec[1], che ha usato immagini dell'ambiente circostante per permettere la navigazione di un robot in una stanza, fornendogli anche la capacità di evitare eventuali ostacoli sul percorso. Tuttavia, l'uso della visione nel campo della robotica mobile è stato ostacolato dal limitato potere computazionale dell'hardware di allora. I tipici task di acquisizione e di processing richiedevano uno sforzo eccessivo per i calcolatori di quei tempi, a causa del grande ammontare di dati nelle immagini, le quali quindi non sono state disponibili per gli scopi robotici fino ai recenti sviluppi della visione artificiale. Inoltre l'esistenza di sensori di posizione molto accurati, anche se più costosi, come i Laser range-finder¹, non

¹Un laser range-finder è un dispositivo che utilizza un raggio laser per determinare la distanza da un oggetto. I tipi più comuni di laser range-finder funzionano in base al principio del tempo di volo, lanciando un impulso laser verso un oggetto e misurando il tempo impiegato dall'impulso stesso per essere riflesso dall'obiettivo e tornare indietro.

favorì il pieno e immediato sviluppo di questa tecnica.

Per mantenere costi ridotti, invece, fino ad oggi il sistema più utilizzato per stimare la posizione di robot mobili è stato l'odometria meccanica, metodo che tuttavia, soprattutto in ambienti outdoor, risulta poco preciso e spesso inaffidabile. L'idea di base della ricostruzione odometrica è quella di calcolare la nuova posizione del robot in base alla strada percorsa rispetto alla posizione precedente, calcolo che viene tramite l'integrazione nel tempo dell'informazione sul movimento ricavata dagli encoder: strumenti calettati agli assi delle ruote che misurano la velocità di rotazione delle stesse.

Sfortunatamente, a causa della sua natura integrativa, questo metodo di odometria meccanica è spesso fonte di errori, soprattutto di orientazione che causano a loro volta errori in posizione, i quali crescono proporzionalmente alla distanza percorsa. Esistono in generale due tipologie di errori: quelli *sistematici*, che sono quelli dovuti alle approssimazioni nei calcoli e possono dipendere da vari fattori (diametro delle ruote diseguale, disallineamento delle ruote, risoluzione e campionamento finito degli encoder), e quelli *non sistematici*, causati invece da fattori accidentali come lo slittamento delle ruote o il movimento su un terreno irregolare (più frequenti in ambienti esterni).

L'odometria visuale è anch'essa una tecnica di localizzazione relativa ma capace di ottenere una migliore accuratezza nella stima della posizione, riducendo significativamente l'errore in confronto all'utilizzo dell'odometria meccanica.

Oggigiorno, questo metodo sta attraendo molta attenzione nella comunità scientifica della robotica. Sono state realizzate diverse implementazioni e i risultati sono molto promettenti, anche se c'è ancora spazio per molto lavoro in questa area. I sistemi attuali possono essere ancora migliorati per essere utilizzati su spazi molto più ampi e in tempi molto più brevi.

L'odometria visuale è stata testata su una grande varietà di terreni e di ambienti, sia outdoor sia indoor. Gli algoritmi sviluppati in letteratura sembrano

dare risultati più soddisfacenti in ambienti esterni piuttosto che su superfici indoor, poiché è più semplice l'estrazione di feature univoche e distinguibili rispetto a quelle estratte all'interno di edifici, le quali si dimostrano essere facilmente confondibili tra loro.

Sulle tracce del lavoro di Moravec, sul finire degli anni novanta, Matthies cominciò a trattare la stima del movimento come un problema statistico e sviluppò alcuni metodi per determinare il movimento di un rover e aggiornare i modelli dei landmark riconosciuti nelle immagini. Questo sistema raggiunse un'accuratezza del 2% su una distanza di circa 5.5 metri, con 55 coppie di immagini stereo acquisite[2, 3].

Lavori simili sono stati sviluppati nel corso degli anni (Zhang[4], Lacroix[5], Hirschmuller[6], Nister[7, 8]). In questi ultimi due sistemi le feature vengono selezionate e poi messe in corrispondenza fra loro sulla base di vettori associati, senza alcuna correlazione spaziale, la quale invece è stata utilizzata negli algoritmi meno recenti. Questi approcci richiedono descrittori robusti e significativi per funzionare correttamente anche con spostamenti ampi tra le immagini ma hanno il vantaggio di non avere la necessità di una stima iniziale del movimento. Hirschmuller sfrutta la mappa di disparità costruita dalle telecamere stereo, ma, finché le risorse di calcolo sono limitate sui rover planetari, questa è una soluzione poco sostenibile. L'approccio di Nister assume invece che gli spostamenti tra le immagini acquisite siano sufficientemente ridotti, ma questo requisito entra in conflitto con il fatto che il task di odometria visuale per un rover planetario sia schedulato il meno frequentemente possibile, in modo da minimizzare il consumo delle risorse computazionali. Sono stati sviluppati anche altri tipi di approcci, come per esempio quello di McCarthy e Barnes, i quali hanno presentato le performance di un sistema di stima del moto basato sul flusso ottico[9], oppure quello di Vassallo, Gluckman e altri, che hanno utilizzato immagini omnidirezionali per ottenere la stima del movimento[10, 11, 12].

L'approccio di Matthies, sfruttando i dati dell'odometria meccanica come stima iniziale, è stato utilizzato dalla NASA per testare la tecnica dell'odometria visuale su Marte, anche se ancora con pesanti limitazioni dal punto di vista computazionale.

Infatti, uno dei campi a cui si sta cercando di applicare questa tecnica, in modo da migliorare significativamente le prestazioni della navigazione autonoma di un rover, è proprio l'esplorazione planetaria, ambito verso il quale l'uomo ha da sempre mostrato grande attenzione e interesse, sia per la mera curiosità di nuove scoperte ma soprattutto nell'ottica di valutare le caratteristiche di pianeti diversi dalla Terra e stabilire se possano eventualmente diventare una fonte di sostegno per il nostro pianeta. Nonostante la grande varietà di sistemi robotici esistenti, le missioni attualmente in corso utilizzano solamente rover (robot su ruote), sia per il fatto che i terreni che si cercano di esplorare sono relativamente percorribili, sia perché la locomozione su ruote risulta tecnicamente la più semplice nella realizzazione². Tuttavia, i robot planetari sono, per caratteristiche tecniche e costruttive, molto differenti da quelli terrestri e il loro design è vincolato da particolari requisiti, dovuti alla diversa tipologia di ambiente in cui devono andare ad operare. Per esempio, oltre alle condizioni ambientali estreme che possono presentarsi su un altro pianeta, si devono tenere anche in considerazione la durata limitata dei materiali e le ridotte performance computazionali ed energetiche, poiché non tutto l'hardware e il software esistenti possono essere utilizzati per le missioni spaziali: solamente le tecnologie che sono *space-qualified* (e che risultano meno avanzate di quelle attualmente usate nelle applicazioni terrestri) possono essere adoperate nei sistemi di volo[13].

Il primo tentativo di navigazione planetaria per mezzo di un robot mobile, senza un vero equipaggio umano *in-situ*, venne fatto nel 1997, quando il rover

²Esistono tuttavia alcuni studi riguardo il possibile utilizzo futuro di *legged-robot*, per scenari estremamente impervi da esplorare, per i quali sarebbe impossibile l'attraversamento su ruote.

americano *Sojourner* (Figura 1.1) fu impiegato per la prima esplorazione di Marte. In questa missione, l'autonomia del rover era molto limitata: *Sojourner* possedeva la sola capacità di riconoscere ed evitare eventuali ostacoli che si presentavano sul suo percorso, mentre tutto il resto delle azioni veniva guidato *step-by-step* attraverso comandi provenienti da operatori terrestri.

Questo fu già un grande passo in avanti dal punto di vista della sicurezza, poiché, per la prima volta nella storia, una missione interplanetaria veniva eseguita senza il ricorso ad un equipaggio vero e proprio. A causa però della scarsa autonomia del rover, l'esplorazione risultò estremamente lenta e difficoltosa, anche perché le comunicazioni ad una distanza così elevata rimangono tutt'oggi molto difficili e costose.

La massima velocità del rover era di 0.01 m/s e *Sojourner* percorse solo alcune centinaia di metri nei suoi 83 sol³ di vita sul pianeta rosso, prima di perdere ogni contatto con il lander⁴ e quindi con la Terra, probabilmente per un guasto alla batteria.

Nella successiva missione Mars Exploration Rovers (MER) partita nel 2004, due nuovi rover, *Spirit* e *Opportunity* sono stati dotati di nuove caratteristiche e abilità che conferiscono loro una grande autonomia durante la navigazione e l'esecuzione dei task.

Ad oggi, i due rover continuano a fornire il loro contributo scientifico, ben al di là delle migliori attese:

- *Spirit*, dopo i 90 giorni di vita che erano stati previsti, ha continuato a marciare sul suolo marziano, festeggiando nel gennaio del 2010 i 6 anni

³Il sol è il nome dato al giorno marziano, il quale dura leggermente di più di un giorno terrestre (circa 24 ore e 37 minuti).

⁴Il lander è un tipo di navicella spaziale che effettua la discesa e la sosta sulla superficie di un corpo celeste. Il compito di un lander è quello di trasportare un rover sul pianeta, facendolo atterrare senza subire danni alla strumentazione.



Figura 1.1: Il rover Sojourner

di lavoro sul pianeta rosso e dimostrando una inaspettata resistenza alle intemperie climatiche che spesso si verificano su Marte.

- *Opportunity*, atterrato tre settimane dopo il suo rover gemello dalla parte opposta del pianeta, è anch'esso tutt'oggi operativo e ha percorso più di 20 Km di superficie marziana.

Spirit e Opportunity, così, sono stati resi capaci di navigare in modo sicuro e più efficiente anche su terreni sabbiosi e/o in forte pendenza, e tutto ciò ha comportato un crescente guadagno scientifico della missione, grazie alla notevole riduzione del numero di giorni impiegati per raggiungere o attraversare delle aree di interesse. Durante i primi anni della missione, il sistema di odometria visuale passò dall'essere una semplice capacità aggiuntiva, integrata sui rover per favorirne dei test direttamente sul campo, a diventare una caratteristica cruciale per



Figura 1.2: Il rover Spirit

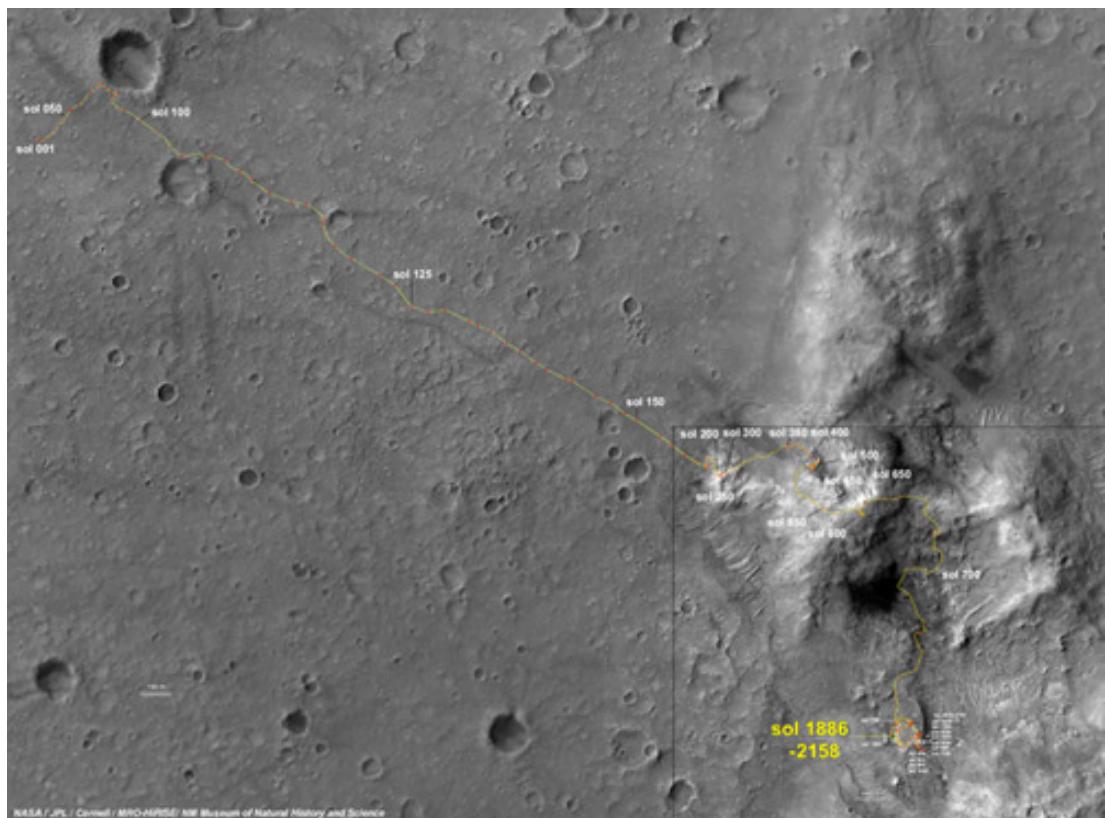


Figura 1.3: Il percorso del rover Spirit

la sicurezza dei veicoli ma solo in determinate occasioni a causa dell'alto costo computazionale.

Inizialmente, infatti, questa tecnica non venne usata in ogni comando di navigazione, poiché si riteneva che il team stesso che curava la missione da Terra non fosse abbastanza familiare con tale sistema. Tuttavia, dopo aver dimostrato una buona efficienza anche sul pianeta marziano, l'odometria visuale cominciò ad essere schedulata con più frequenza, tenendo sempre in considerazione che il tempo di computazione necessario alla CPU da 20 MHz, di cui sono dotati i due rover, per l'acquisizione e l'elaborazione delle immagini riduce estremamente la velocità di navigazione, e quindi si deve bilanciare la necessità di ottenere una buona stima della posizione dei veicoli con il desiderio di coprire lunghe distanze sul pianeta rosso.

Proprio questo aspetto è diventato motivo di ricerca, per cercare di sviluppare algoritmi che, pur con macchine non computazionalmente all'avanguardia, come quelle *space-qualified*, riescano ad ottenere una buona stima della posizione in tempi accettabili, per poter schedulare l'odometria visuale in modo più continuo e fruttuoso per le operazioni di navigazione e localizzazione.

Molte tecniche, che sono state sviluppate per applicazioni terrestri, hanno dimostrato buoni risultati e, sommando a questo il continuo sviluppo della tecnologia, quindi, si può pronosticare che per la prossima generazione di rover planetari sarà possibile percorrere lunghe distanze autonomamente ed esplorare in modo approfondito luoghi e territori spaziali in tempo sempre più ridotto e con minor sforzo da parte degli operatori terrestri.

Insomma, nel futuro, l'odometria visuale diventerà un punto fondamentale per la localizzazione e per la navigazione autonoma dei rover planetari.

Capitolo 2

L’odometria visuale

In robotica e nella visione artificiale, l’odometria visuale è il processo che determina la posizione e l’orientazione di un rover in movimento, esaminando i cambiamenti avvenuti sulle immagini acquisite prima e dopo lo spostamento.

Allo stato dell’arte esistono due principali classi di tecniche con cui, fino ad oggi, si è cercato di affrontare questo problema: i metodi *dense motion based* e quelli *feature based*[14].

Gli algoritmi basati sul *dense motion*, anche conosciuto come *optical flow*, seguono il movimento dei pattern di luminosità sull’intera immagine inquadrata[15]. I campi di flusso calcolati da questi metodi sono tipicamente utili anche per ulteriori scopi, come l’obstacle avoidance o altri task di basso livello, ma è molto complicato correlarli con la geometria globale della scena inquadrata. Tuttavia esiste un’eccezione degna di nota in letteratura, cioè un recente lavoro sul flusso stereoscopico[16].

I metodi *feature based* invece tengono traccia solo di un piccolo numero di punti (feature) da immagine a immagine. L’uso delle feature riduce drasticamente la complessità di calcolo e la quantità di dati prelevati dai frame di cui ci si deve occupare, rendendo così più realistica la possibilità di ottenere performance in real-time.

Tali metodi, poichè si basano principalmente sulla posizione dei punti d'interesse anzichè sul confronto diretto dei dati delle immagini, risultano più robusti alle eventuali variazioni di illuminazione della scena inquadrata e al rumore dovuto all'acquisizione. Inoltre essi consentono uno spostamento più ampio tra un frame e l'altro e necessitano di un minor carico computazionale. D'altra parte, la loro accuratezza nella stima è generalmente inferiore a quella dei metodi basati sul *dense motion*.

Di seguito si illustrerà la tecnica *feature-based* applicata alle immagini stereo acquisite dal sistema e verranno poi analizzati i passi fondamentali nei loro caratteri generali, essendo sempre gli stessi in qualunque algoritmo sviluppato e differenziandosi soltanto nel modo in cui vengono implementati. In ogni sistema di tale tipo infatti si sono evidenziate cinque fasi costanti (Figura 2.1):

1. Estrazione delle feature: procedura in cui vengono selezionati i punti d'interesse dell'immagine acquisita.
2. Matching: ricerca dei punti corrispondenti a quelli estratti nell'altra immagine stereo.
3. Triangolazione: generazione di un insieme di punti tridimensionali a partire dalle corrispondenze individuate nelle due viste.
4. Tracking: procedimento che tiene traccia nelle acquisizioni successive delle feature presenti nell'immagine di partenza.
5. Stima del movimento: algoritmo che, dati due set di coordinate 3D riferite alle posizioni delle feature rispettivamente prima e dopo lo spostamento, stima il movimento avvenuto tra le due immagini.

In questo capitolo, quindi, si tenterà di chiarire il loro significato e fare una panoramica delle tecniche utilizzate in letteratura per sviluppare tali procedure,

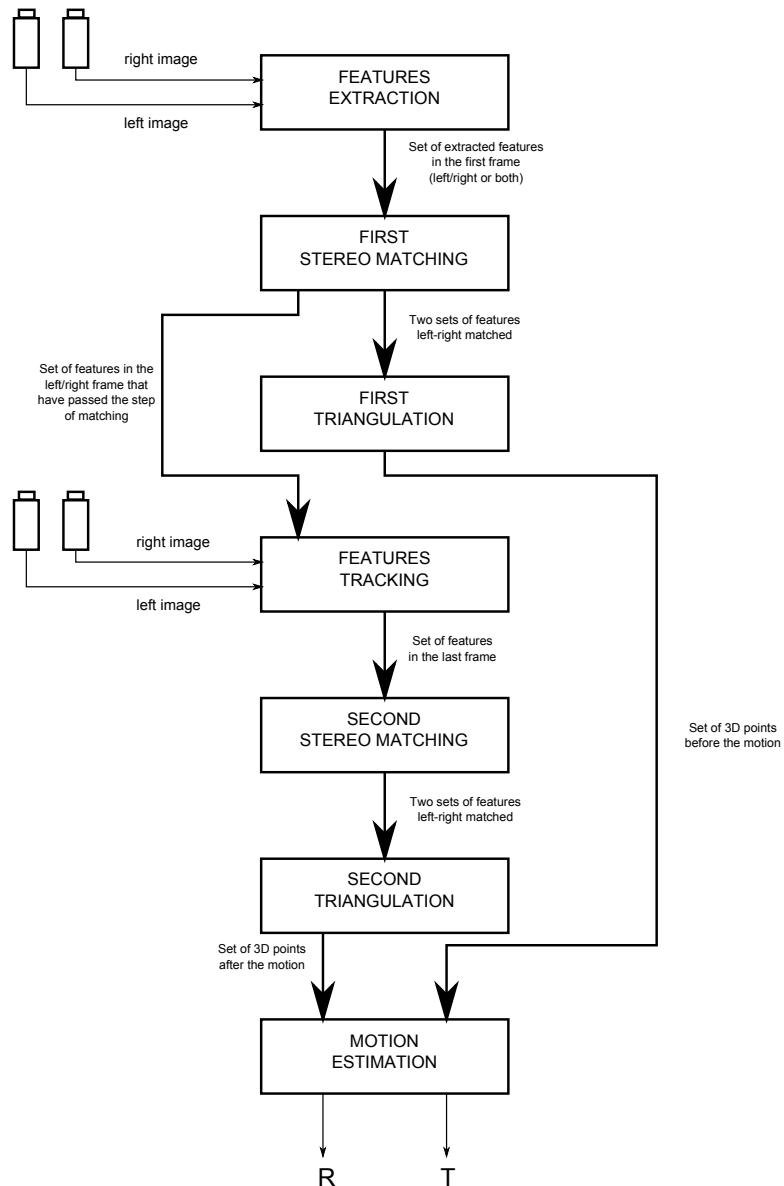


Figura 2.1: Le fasi dell'algoritmo di odometria visuale

mentre i capitoli successivi illustreranno in maniera più approfondita le tecniche utilizzate in questo progetto, con i rispettivi risultati ottenuti.

2.1 L'estrazione di feature

La stima del movimento in una sequenza di immagini in molti casi risulta essere un problema malposto. Se si considerano piccole regioni di un'immagine o anche singoli pixel, solitamente l'informazione disponibile non è sufficiente per determinare lo spostamento in maniera affidabile[17]. Considerando per esempio una piccola parte di un'immagine con una zona di bordo lineare continua (Figura 2.2), è molto difficile trovare la posizione corrispondente in una immagine più grande, poichè il pattern si ripete lungo tutta la linea di bordo.

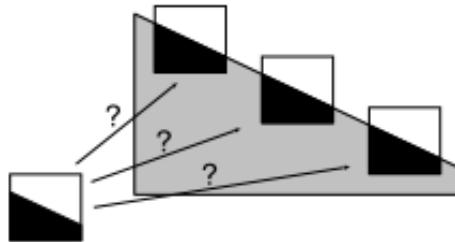


Figura 2.2: Il problema dell'*apertura*

In questo caso, infatti, l'edge è caratterizzato da una sola direzione, perpendicolare alla linea di bordo, lungo la quale l'immagine è uniforme, rendendo così impossibile la ricerca del miglior punto di matching. Questo non determinismo della soluzione è noto in letteratura come “problema dell'apertura” (*aperture problem*).

La possibilità di determinare in modo univoco e affidabile la posizione del pattern aumenta considerevolmente se quest'ultimo mostra variazioni in due direzioni differenti. E' questo il caso, per esempio, dei *corner* (Figura 2.3), nei quali

la posizione della feature può essere determinata inequivocabilmente in entrambe le direzioni.

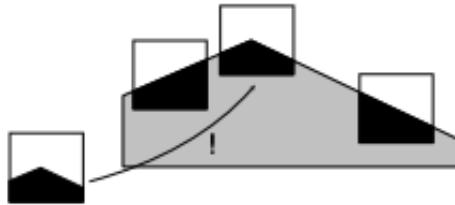


Figura 2.3: Un pattern univocamente identificabile

Quindi, allo scopo di determinare la stima del movimento attraverso le immagini, poichè si ottiene un'informazione scarsa dalle aree uniformi o da quelle caratterizzate da bordi monodimensionali, l'idea che sta alla base dei metodi *feature based* è quella di concentrarsi solo su un piccolo insieme di *interest point*, cioè sui punti che introducono poca incertezza nel calcolo della soluzione.

Fino ad oggi sono stati sviluppati numerosi algoritmi per la rilevazione di feature, che sostanzialmente si differenziano per la replicabilità dei corner estratti dall'immagine e per il carico computazionale utilizzato. Alcuni possiedono anche le caratteristiche di essere invarianti alle rotazioni e/o ai cambiamenti di scala.

2.1.1 I *feature detector*

In letteratura esistono un buon numero di estrattori di feature e l'argomento rimane ancora oggi un'area di ricerca attiva. Uno dei primi è stato quello di Moravec[18], un semplice algoritmo *ad-hoc* per rilevare i punti di un'immagine che mostrano variazioni di luminosità in direzioni differenti. Shi e Tomasi[19] sono partiti dal problema della stima del movimento e hanno definito una tecnica che fosse appropriata per tale scopo.

Alcuni autori invece hanno proposto l’utilizzo di estrattori che di fatto sono rilevatori di *corner*, cioè di punti che rientrano strettamente nella definizione, senza tenere in considerazione che potrebbero esserci altri punti “interessanti” per quanto riguarda la stima del movimento. Infatti, è importante sottolineare che per l’applicazione dell’odometria visuale non importa tanto se le feature siano posizionate esattamente nella specifica locazione in cui il “vero corner” risiede, quanto che l’estrattore individui coerentemente le stesse posizioni dei punti corrispondenti in ciascuna immagine.

Uno fra i più comunemente usati è l’*Harris corner detector*[20], sul quale si è fortemente basata l’implementazione di Shi e Tomasi, ma che si differenzia per il criterio con cui vengono selezionate le feature. Più recentemente sono stati sviluppati nuovi estrattori, sostanzialmente diversi dai precedenti, con migliori capacità di *detection* al prezzo però di un maggiore costo computazionale. Tra questi si possono citare il SUSAN detector [21], che possiede una buona capacità di localizzazione dei corner, il FAST[22], il SIFT [23] e le sue successive evoluzioni, il SURF e l’U-SURF[24], che utilizzano particolari tipi di dati associati ai punti estratti, chiamati descrittori, i quali “descrivono” distintamente i punti rilevati e permettono un eccellente operazione di matching tra le varie feature. Alcuni di questi estrattori, inoltre, godono della notevole caratteristica di essere invarianti ad alcune classi di trasformazioni, come le rotazioni o i cambiamenti di scala. Questi fattori risultano poco rilevanti quando le acquisizioni dei frame sono molto ravvicinate nel tempo, in quanto le trasformazioni tra le immagini sono di piccola entità, ma si dimostrano invece determinanti per il tracking delle feature quando intercorre un lasso di tempo significativo fra due frame successivi. Inoltre, grazie alla continua evoluzione delle capacità computazionali dell’hardware, anche detector come il SIFT e il SURF possono essere utilizzati senza dover rinunciare al calcolo dell’odometria visuale on-line.

Dopo una breve discussione sui criteri di selezione dei punti d’interesse, ver-

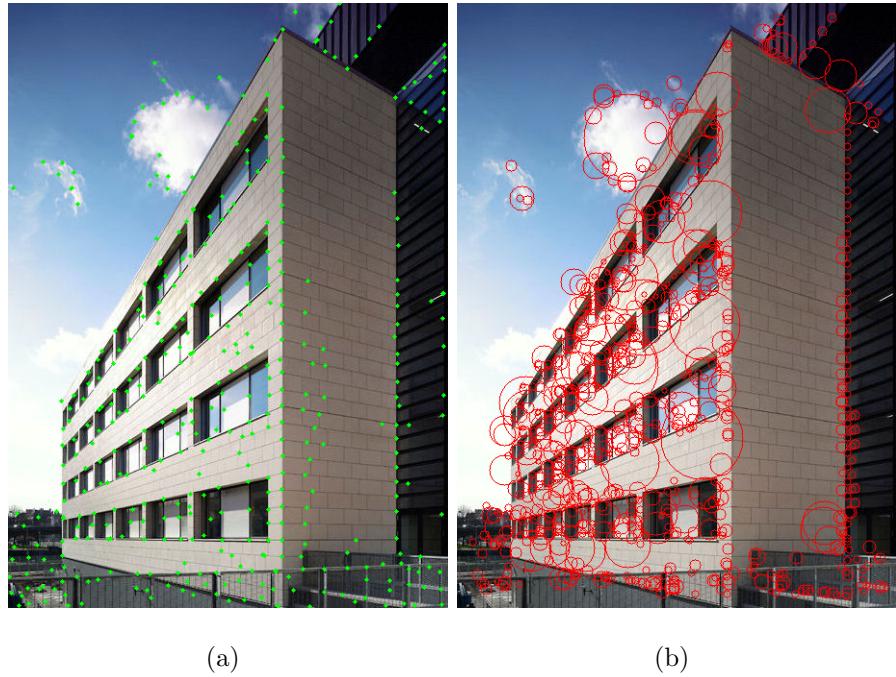


Figura 2.4: Harris corner (a) e SURF detector (b)

ranno presentati i due tipi di estrattori che sono stati utilizzati in questo progetto, l’Harris corner detector e il SURF (Figura 2.4), mostrando le loro principali caratteristiche ed i motivi per cui sono stati scelti per i nostri scopi.

2.1.2 Criteri per un buon estrattore

I punti d’interesse utilizzati per gli algoritmi di odometria visuale, e per la visione in generale, dovrebbero essere collocati in posizioni in cui il contenuto dell’immagine attorno alla feature rilevata permetta di identificare la posizione corrispondente in una seconda immagine con una buona affidabilità. Come già accennato, individuare un punto-feature su una linea non è sufficiente, poichè essa è un pattern monodimensionale e perciò non permette di definire la posizione del punto in maniera univoca.

Di conseguenza, le “buone” feature devono necessariamente mostrare un cam-

biamento di contenuto immagine in almeno due direzioni, in modo che possano essere localizzate in maniera precisa. Nella pratica, quindi, corner, giunzioni a T o texture complesse forniscono una buona localizzazione (Figura 2.5).

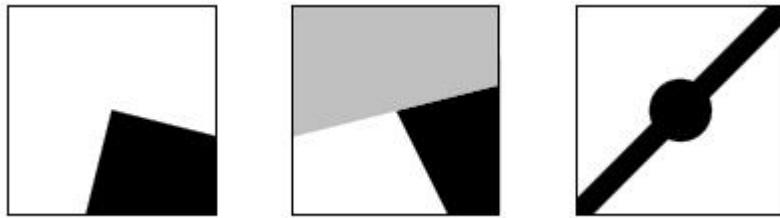


Figura 2.5: Porzioni d'immagine adatte per la collocazione di punti d'interesse

Tuttavia, i punti d'interesse vengono selezionati solo localmente (e non con una visione d'insieme dell'immagine) e un punto che a prima vista sembrerebbe semplice da seguire nella fase di tracking potrebbe rivelarsi una cattiva scelta se fossero presenti delle ambiguità nell'immagine, per esempio, a causa di pattern ripetitivi (Figura 2.6).

Questo problema può essere risolto solo globalmente e, inoltre, non dovrebbe interessare questo progetto, il quale prevede la realizzazione di un algoritmo di odometria visuale per l'esplorazione planetaria, in cui difficilmente si incontreranno terreni con pattern ripetitivi (tipici di ambienti indoor).

Per quanto riguarda i successivi passi dell'algoritmo, nei quali si cerca di stabilire le corrispondenze tra punti appartenenti a diverse coppie di immagini, ci sono due proprietà estremamente importanti per i detector.

La prima è la *ripetibilità*, cioè la capacità dell'estrattore, in immagini che mostrano lo stesso contenuto da un punto di vista poco differente, di rilevare le stesse feature in entrambi i frame. I punti che vengono individuati solo in una delle due immagini non possono necessariamente trovare una corrispondenza nell'altra, diventando quindi inutili per i passi successivi dell'algoritmo. Ancor peggio, queste feature potrebbero a tutti gli effetti rappresentare una sorgente

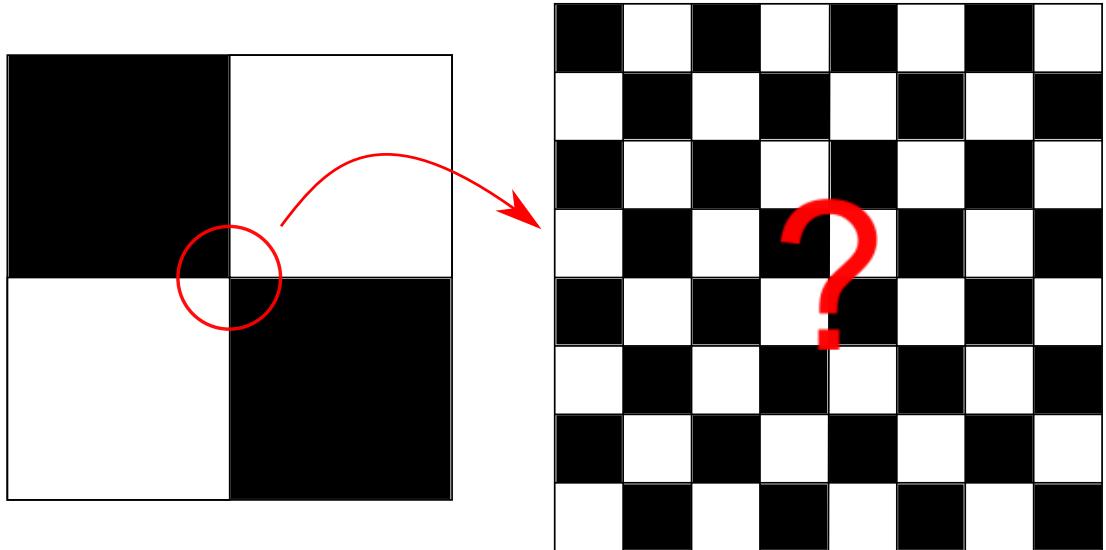


Figura 2.6: Difficoltà di localizzazione di un corner ripetitivo

d'errore, conducendo a corrispondenze errate e rendendo meno robusta la stima del moto.

Un secondo importante criterio per stabilire la bontà di un estrattore è l'*invarianza* dei punti rilevati alle trasformazioni (di scala e rotazionali) presenti tra le due immagini.

2.1.3 L'Harris detector

L'Harris corner detector, come detto in precedenza, è un estrattore molto simile all'operatore di Shi-Tomasi, il quale è un'algoritmo sviluppato proprio a partire dal problema della stima del movimento, cercando di trovare le condizioni migliori in cui quest'ultimo può essere risolto. L'operatore suddetto, infatti, non cerca punti d'interesse che siano definiti espressamente come *corner*, ma sceglie quelli che, per l'algoritmo, sono semplici da seguire nella procedura di tracking. Poiché in questo progetto è stata utilizzata una funzione basata sull'operatore di Shi-

Tomasi, si illustrerà tale estrattore e, di seguito, si mostreranno le differenze dei criteri di scelta delle feature selezionate.

Assumendo che avvenga un movimento traslazionale \mathbf{d} tra due frames I_t e I_{t+1} e considerando una finestra W attorno alla feature, l'errore di matching \mathbf{E} può essere scritto come

$$E = \sum_{\mathbf{p} \in W} (I_t(\mathbf{p} + \mathbf{d}) - I_{t+1}(\mathbf{p}))^2. \quad (2.1)$$

Dopo aver approssimato I_t per mezzo di un'espansione di Taylor lineare, per cui $I_t(\mathbf{p} + \mathbf{d}) \approx I_t(\mathbf{p}) + \nabla I_t^\top \mathbf{d}$, si può riscrivere l'errore:

$$E = \sum_{\mathbf{p} \in W} (I_t(\mathbf{p}) - I_{t+1}(\mathbf{p}) + \nabla I_t^\top \mathbf{d})^2. \quad (2.2)$$

Per determinare il movimento \mathbf{d} , si minimizza l'errore ponendo a zero le derivate:

$$\frac{\partial E}{\partial \mathbf{d}} = 2 \sum_{\mathbf{p} \in W} (I_t(\mathbf{p}) - I_{t+1}(\mathbf{p}) + \nabla I_t^\top \mathbf{d}) \nabla I_t = 0, \quad (2.3)$$

da cui segue che

$$\sum_{\mathbf{p} \in W} (\nabla I_t \nabla I_t^\top) \cdot \mathbf{d} = \sum_{\mathbf{p} \in W} \nabla I_t \cdot (I_t(\mathbf{p}) - I_{t+1}(\mathbf{p})), \quad (2.4)$$

che è un sistema di equazioni lineari $\mathbf{G}\mathbf{d} = \mathbf{e}$, dove

$$\mathbf{G} = \begin{bmatrix} \sum(\partial I_t(\mathbf{p})/\partial x)^2 & \sum(\partial I_t(\mathbf{p})/\partial x)(\partial I_t(\mathbf{p})/\partial y) \\ \sum(\partial I_t(\mathbf{p})/\partial x)(\partial I_t(\mathbf{p})/\partial y) & \sum(\partial I_t(\mathbf{p})/\partial y)^2 \end{bmatrix} \quad (2.5)$$

$$\mathbf{e} = \begin{pmatrix} \sum(\partial I_t(\mathbf{p})/\partial x) \cdot (I_t(\mathbf{p}) - I_{t+1}(\mathbf{p})) \\ \sum(\partial I_t(\mathbf{p})/\partial y) \cdot (I_t(\mathbf{p}) - I_{t+1}(\mathbf{p})) \end{pmatrix} \quad (2.6)$$

in cui tutte le sommatorie vanno fatte sulla finestra W .

Per risolvere il sistema in modo affidabile nell'incognità \mathbf{d} , esso deve essere ben condizionato¹. Shi e Tomasi sostengono che il sistema è tale se entrambi

¹Un problema è ben condizionato quando la soluzione del problema con delle piccole variazioni non differisce molto dalla soluzione del problema originale.

gli autovalori di \mathbf{G} non differiscono molto tra loro e sono superiori ad una soglia minima. Parlando in termini di immagini, due autovalori piccoli corrispondono ad un contenuto quasi uniforme, mentre due autovalori con grandezza molto diversa indicano la possibile presenza di un edge lineare. Di conseguenza gli autori hanno proposto l'uso del più piccolo dei due autovalori λ_1, λ_2 come criterio per rilevare i punti feature.

Anzichè derivare semplicemente l'Harris detector da quello di Shi-Tomasi mostrandone le differenze, si cercherà di dare una spiegazione alternativa, più intuitiva, di come è definito e di come funziona questo estrattore. Si prenda in esame il contenuto di una finestra W attorno alla posizione del pixel considerato e si osservino i vettori gradiente della finestra stessa. L'idea di base è classificare il contenuto immagine della finestra basandosi sulla distribuzione dei vettori gradiente. La Figura 2.7 mostra i diagrammi di distribuzione dei vettori gradiente per tre diverse finestre dell'immagine di esempio.

Se si considera la finestra 1, che contiene un edge lineare, si ottiene una distribuzione del vettore gradiente come quella mostrata in Figura 2.7(c). Da notare che tutti i vettori gradiente hanno approssimativamente la stessa orientazione, in cui l'asse principale sul quale si distribuiscono è perpendicolare all'edge dell'immagine. Per un corner invece (Figura 2.7(e)), si può notare che ci sono differenti orientazioni del gradiente, ciascuna corrispondente a quella dei bordi del corner stesso. Quest'ultima configurazione è confrontabile ad un contenuto immagine più complesso (Figura 2.7(d)), in cui i vettori sono distribuiti in maniera più o meno uniforme. Infine, per porzioni di immagine con solo piccole variazioni di luminanza², tutti i vettori gradiente sono vicini allo zero. Per classificare le diverse situazioni elencate sopra, si può modellare la distribuzione dei vettori con una Gaussiana bidimensionale di cui gli assi principali possono essere ottenuti come le componenti principali della matrice di correlazione dei vettori gradiente.

²Misura dell'intensità di luminosa

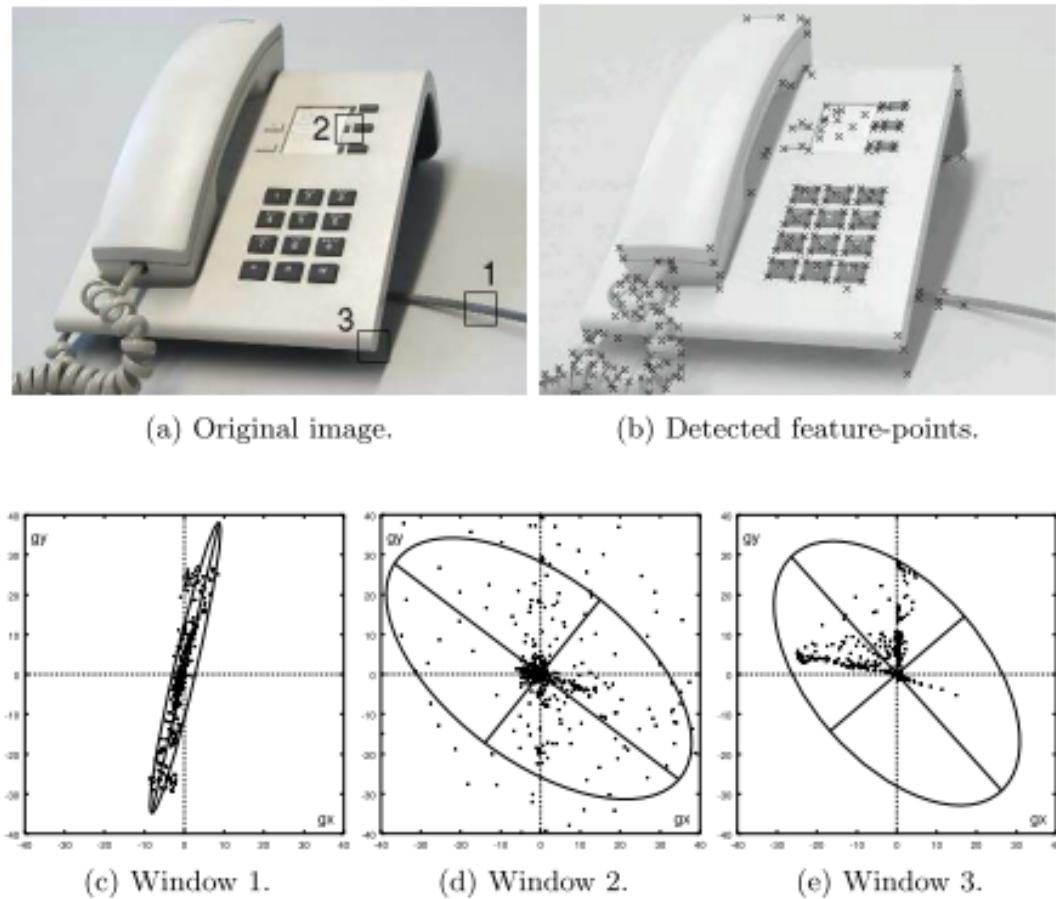


Figura 2.7: (c)-(e) Diagrammi di distribuzione dei gradienti per le finestre indicate in (a). Le posizioni dei corner rilevate sono mostrate in (b).

Si noti che la matrice di correlazione è esattamente la stessa (\mathbf{G}) di quella definita in precedenza per l'operatore Shi-Tomasi. Oltre tutto, le lunghezze degli assi principali corrispondono proprio agli autovalori λ_1, λ_2 della matrice di correlazione. Se si osservano tali lunghezze per ciascun tipo di contenuto immagine, si può concludere che due assi di piccola lunghezza indicano la presenza di una porzione di immagine piatta, uniforme; un asse di lunghezza grande rispetto all'altro invece corrisponde ad una struttura lineare (come un bordo), mentre due assi entrambi di considerevoli dimensioni indicano un corner o comunque un altro contenuto complesso dell'immagine.

Seguendo queste osservazioni, Harris e Stephens hanno proposto lo schema in Figura 2.8 per classificare ogni specifica distribuzione del vettore gradiente in una delle classi *flat*, *edge* e *corner*.

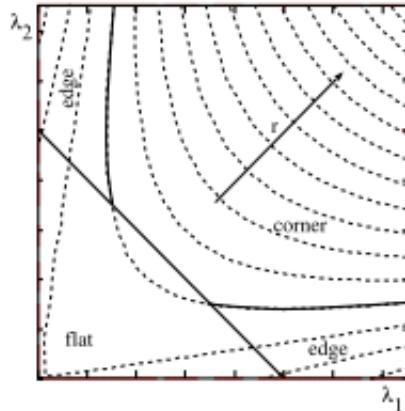


Figura 2.8: Classificazione del contenuto immagine secondo l'opeartore di Harris

Secondo questo schema, una porzione di immagine dal contenuto uniforme è rilevata se si verifica la condizione $\lambda_1 + \lambda_2 < \tau_{th}$, dove τ_{th} è la soglia scelta per distinguere un'immagine piatta da un'altra con un contenuto più significativo.

La funzione per stabilire la presenza o meno di un corner nella porzione

considerata è stata definita come:

$$r = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = \text{Det}(\mathbf{G}) - k \cdot \text{Tr}(\mathbf{G})^2 > 0 \quad (2.7)$$

dove $k \approx 0.06$.

Da notare che la funzione precedente è stata scelta in modo che non sia necessario calcolare effettivamente gli autovalori. In tal maniera, invece, risulta sufficiente trovare il determinante e la traccia della matrice di correlazione, sfruttando la loro equivalenza, rispettivamente, al prodotto e alla somma degli autovalori stessi. Per determinare la posizione dei punti rilevati come corner, e quindi come feature, vengono calcolati i minimi locali di r e la feature viene collocata effettivamente in quella posizione se $r > 0$ e se risulta positivo anche un altro test sulla non uniformità del contenuto ($\text{Tr}(\mathbf{G}) \geq \tau_{th}$, con τ_{th} soglia di uniformità del contenuto immagine).

L'estrattore di Harris, tuttavia, presenta un problema, analogamente a Shi-Tomasi, per cui la posizione del punto corrispondente al corner rilevato potrebbe non trovarsi al centro della finestra considerata (poichè l'algoritmo tende a massimizzare tutte le possibili strutture presenti all'interno della porzione di immagine, senza privilegiare alcun pixel) (Figura 2.9). Questo inconveniente può essere comunque facilmente superato grazie all'utilizzo di una funzione di peso (Figura 2.10), che assegna un'importanza maggiore ai pixel centrali, prevenendo così l'effetto di “decentramento” del corner.

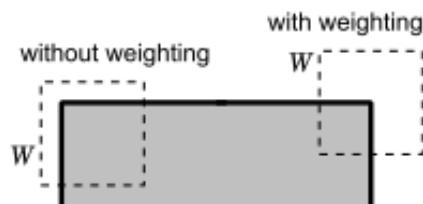


Figura 2.9: Rilevazione del corner tramite finestra semplice e finestra pesata

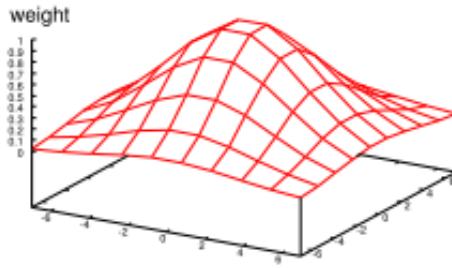


Figura 2.10: Una funzione di peso gaussiana

L’Harris detector appartiene alla prima generazione di estrattori realizzati³ e possiede il pregio di essere computazionalmente molto leggero, impiegando poco tempo ad estrarre molte feature anche su immagini di grandi dimensioni. Gli aspetti negativi sono la sua scarsa ripetibilità su immagini simili o poco differenti le une dalle altre e la sua non invarianza al fattore di scala. Tuttavia, nonostante questi difetti, quello di Harris è stato (ed è tuttora) uno degli estrattori più utilizzati, poichè viene sfruttata la sua velocità di estrazione, caratteristica fondamentale nei casi in cui l’hardware a disposizione per il processing delle immagini non possieda una potenza di calcolo elevata.

2.1.4 Il SURF

L’estrattore SURF (Speeded-Up Robust Feature) è uno dei più recenti algoritmi di estrazione di feature nelle immagini ed è sostanzialmente diverso dall’Harris: più complesso, può essere visto come una evoluzione del SIFT (Scale Invariant Feature Transform). Così come quest’ultimo, infatti, il SURF associa ai punti d’interesse individuati nel frame un descrittore, cioè un vettore caratteristico dell’intorno di tali pixel, il quale deve essere *distintivo*, e quindi permettere di riconoscere una particolare feature dalle altre, e al tempo stesso *robusto* al rumore, agli errori di rilevazione e alle deformazioni geometriche che avvengono tra le immagini.

³La sua pubblicazione risale al 1988

Tali descrittori infatti vengono poi utilizzati per trovare le corrispondenze tra le feature di immagini diverse, confrontando semplicemente la distanza (per esempio quella euclidea) che esiste tra questi vettori.

L'Integral image

Il SURF sfrutta una rappresentazione intermedia dell'immagine da processare, detta “integral image”, la quale viene calcolata molto rapidamente a partire dal frame originale e viene utilizzata per velocizzare in maniera cospicua il calcolo delle finestre necessarie all'estrattore.

Data un'immagine I e un punto di coordinate (x, y) , l'immagine integrale $I_{\Sigma}(x, y)$ è calcolata tramite la somma dei valori dell'intensità dei pixel compresi tra il punto suddetto e l'origine, posta nell'angolo superiore sinistro dell'immagine stessa, ovvero:

$$I_{\Sigma}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y) \quad (2.8)$$

Utilizzando questa tecnica, il compito di calcolare la somma delle intensità di una qualsiasi porzione dell'immagine si riduce alla sola esecuzione di un paio di operazioni e alcuni accessi in memoria. Considerando, infatti, un rettangolo di vertici A, B, C e D (Figura 2.11), la somma dell'intensità dei suoi pixel è uguale a:

$$\Sigma = A + D - (C + B) \quad (2.9)$$

dove con A, B, C e D si intendono i valori integrali corrispondenti alle coordinate del relativo vertice. Il grande vantaggio di tale metodo è che il calcolo è invariante rispetto alla dimensione dell'area in esame: infatti l'estrattore SURF sfrutta appieno tale vantaggio per effettuare in modo efficiente la convoluzione dei filtri, di dimensione variabile, applicati all'immagine.

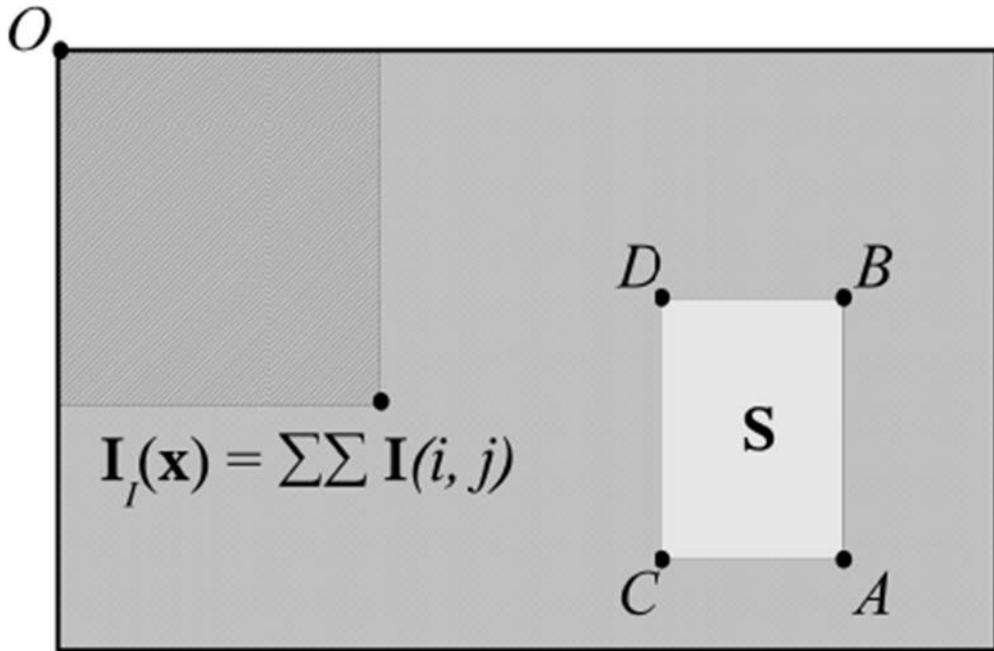


Figura 2.11: Calcolo di una finestra tramite l'utilizzo dell'immagine integrale

Punti d'interesse basati sulla matrice Hessiana

L'estrattore SURF si basa principalmente sul determinante della matrice Hessiana per le sue notevoli performance dal punto di vista dell'accuratezza. Più precisamente, tale tecnica di estrazione rileva le strutture *blob-like*⁴ nelle posizioni in cui il determinante di tale matrice risulta massimo.

Considerando il caso generale di una funzione continua di due variabili $f(x, y)$, l'Hessiana è la matrice delle derivate parziali della funzione stessa:

$$H(f(x, y)) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial^2 x} & \frac{\partial f(x, y)}{\partial x \partial y} \\ \frac{\partial f(x, y)}{\partial x \partial y} & \frac{\partial f(x, y)}{\partial^2 y} \end{bmatrix} \quad (2.10)$$

e il suo determinante si ottiene

$$|H(f(x, y))| = \frac{\partial f(x, y)}{\partial^2 x} \frac{\partial f(x, y)}{\partial^2 y} - \left(\frac{\partial f(x, y)}{\partial x \partial y} \right)^2. \quad (2.11)$$

⁴Le blob feature sono definite come punti e/o regioni dell'immagine di contenuto più chiaro o più scuro rispetto ai pixel che li circondano.

Poichè il determinante coincide con il prodotto degli autovalori della matrice, è possibile stabilire se il punto considerato è un estremo della funzione, basandosi sul segno del determinante stesso: se è negativo, gli autovalori hanno segno differente e quindi il punto non è un estremo locale, in caso contrario si tratta di un punto di massimo o di minimo. Spostando il problema nel campo delle immagini, si sostituisce la funzione $f(x, y)$ con l'intensità dei pixel $I(x, y)$ e per calcolare le derivate parziali seconde dell'immagine si utilizza la convoluzione con un filtro appropriato.

Il SURF si serve di un filtro Gaussiano del secondo ordine normalizzato, che permette un'analisi spaziale e su diversi fattori di scala. Infatti, dato un punto $\mathbf{x} = (x, y)$ di un'immagine I , la matrice Hessiana $\mathcal{H}(x, \sigma)$ in \mathbf{x} alla scala fissata σ è definita come segue:

$$\mathcal{H}(x, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \quad (2.12)$$

dove $L_{xx}(\mathbf{x}, \sigma)$ è la convoluzione della derivata di secondo ordine della Gaussiana $\frac{\partial^2}{\partial x^2}g(\sigma)$ con l'immagine I nel punto \mathbf{x} , e analogamente $L_{xy}(\mathbf{x}, \sigma)$ e $L_{yy}(\mathbf{x}, \sigma)$.

Le Gaussiane si prestano in modo ottimale all'analisi su diversi fattori di scala [25], ma nella pratica devono essere necessariamente discretizzate e approssimate. Tutto ciò conduce ad una diminuzione della ripetibilità, la quale, tuttavia, risulta accettabile per il grande vantaggio che si ottiene dalla rapidità con cui vengono svolte le convoluzioni.

Spingendo l'approssimazione della matrice Hessiana all'estremo, si può utilizzare un *box filter* il quale può essere calcolato ad un costo computazionale estremamente basso, utilizzando le immagini integrali. Inoltre, le sue performance sono paragonabili o, in alcuni casi, addirittura migliori di quelle di una Gaussiana discretizzata e approssimata.

Per esempio, i box filter 9×9 in Figura 2.12 sono le approssimazioni di una Gaussiana con $\sigma = 1.2$ e rappresentano la scala più bassa per calcolare le risposte

di convoluzione.



Figura 2.12: Approssimazioni di una Gaussiana (le zone in grigio equivalgono a zero)

Il determinante approssimato dell'Hessiana rappresenta insomma la risposta del *blob* nella posizione \mathbf{x} dell'immagine. Queste risposte sono salvate in una mappa su differenti scale, in cui, poi, vengono determinati i massimi locali, come spiegato di seguito.

Rappresentazione *Scale-space*

E' necessario che i punti d'interesse siano rilevati con fattori di scala differenti, se non altro perchè la ricerca delle corrispondenze spesso richiede il confronto di immagini in cui le feature sono viste a distanze diverse. Un altro vantaggio, dovuto all'utilizzo dei box filter e delle immagini integrali, è che non è necessario applicare iterativamente lo stesso filtro all'output di diversi livelli d'immagine precedentemente filtrati, ma si possono applicare box filter di dimensioni diverse, alla stessa velocità di computazione, direttamente sull'immagine originale e, eventualmente, anche in parallelo.

Di conseguenza, lo *scale-space* viene analizzato aumentando le dimensioni del filtro di base, anzichè riducendo iterativamente l'immagine di partenza, come mostrato in Figura 2.13.

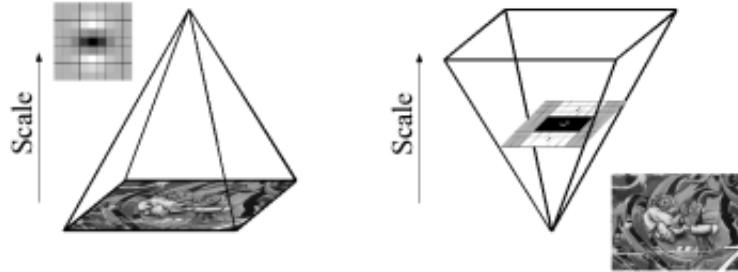


Figura 2.13: Analisi dello spazio dei fattori di scala in differenti versioni

Il risultato del filtro 9×9 , introdotto nel precedente paragrafo, è considerato come il livello di scala iniziale (approssimando le derivate di una Gaussiana con $\sigma = 1.2$), e i livelli successivi si ottengono filtrando la stessa immagine con maschere gradualmente più grandi, mantenendo le dovute proporzioni, sempre con l'enorme vantaggio di un costo computazionale fisso. Occorre ricordare, inoltre, che, non sottocampionando l'immagine, si evita il fastidioso problema dell'*aliasing*⁵.

Lo *scale-space* si divide in ottave, e ogni ottava rappresenta una serie di risposte ai filtri citati in precedenza, ottenute convoluendo la stessa immagine in ingresso con filtri di dimensioni crescenti. In totale, un'ottava comprende un fattore di scala uguale a 2, e ciascuna è suddivisa in un numero costante di livelli. A causa della natura discreta delle immagini integrali, la minima differenza di scala dipende dalla lunghezza dei lobi delle derivate parziali seconde nella direzione di derivazione.

La costruzione dello *scale-space* comincia dal filtro base 9×9 , che calcola la risposta del frame per la scala più piccola, e poi vengono applicati in sequenza

⁵L'aliasing è l'effetto indesiderato che si verifica nella creazione di immagini e suoni, quando avviene la creazione di una falsa (alias) frequenza che si sovrappone a quella desiderata, provocando del rumore di sottofondo (per i suoni) e contorni imprecisi o l'effetto "scalettato" nel tracciare le rette o le curve.

filtri 15×15 , 21×21 e 27×27 , grazie ai quali si guadagna più di un fattore 2 di scala. Ciò è necessario, poichè viene fatta una soppressione *non-maxima* sia spazialmente sia sulle scale vicine, indi per cui la prima e l'ultima mappa di risposte Hessiane nello stack non possono contenere dei punti di massimo, essendo usate solo per motivi di comparazione. Di conseguenza, per la prima ottava, dopo l'interpolazione, la scala più piccola possibile diventa $\sigma = 1.6 = 1.2\frac{12}{9}$, equivalente ad un filtro 12×12 , mentre quella più grande è $\sigma = 3.2 = 1.2\frac{24}{9}$ [26].

Si possono fare considerazioni analoghe per le ottave successive. Per ogni altra ottava, l'incremento delle dimensioni del filtro raddoppia, andando da 6, 12, 24 fino a 48 (Figura 2.14). Allo stesso tempo, anche gli intervalli di campionamento per l'estrazione dei punti d'interesse possono essere raddoppiati così come per ogni nuova ottava. Ciò riduce il tempo computazionale e la perdita in accuratezza è paragonabile a quella che si avrebbe sottocampionando l'immagine con l'approccio tradizionale.

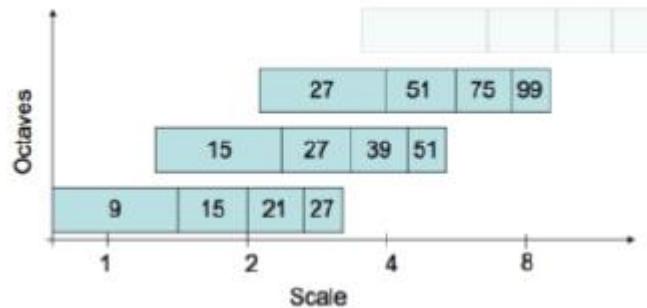


Figura 2.14: Le dimensioni dei filtri in base alle ottave e ai livelli di scala

Localizzazione dei punti d'interesse

Dopo aver costruito le mappe contenenti le risposte dell'immagine ai vari livelli di scala delle diverse ottave, occorre individuare la posizione dei punti d'interesse e il loro fattore di scala corrispondente.

Come già anticipato, una soppressione *non-maxima* consente di avere già un set di punti candidati: ciascun pixel nella mappa delle risposte viene confrontato con i suoi vicini, sia quelli della risposta stessa su cui risiede tale pixel, sia quelli corrispondenti nelle risposte di scala superiore e inferiore(Figura 2.15).

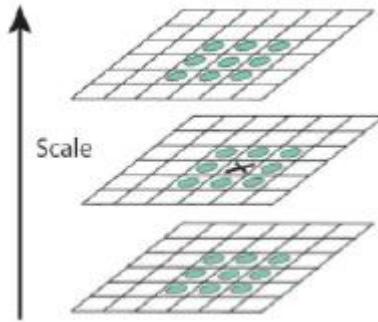


Figura 2.15: Soppressione non-maxima nello spazio e nella scala

Si ottengono di conseguenza un insieme di punti d'interesse filtrati, che sono minimi o massimi nello *scale-space*. Infine, occorre interpolare i dati vicini con l'obiettivo di localizzare il più accuratamente possibile i punti, sia nello spazio che nella scala[27].

Descrizione dei punti d'interesse

I descrittori contengono l'informazione sulla distribuzione dell'intensità dei pixel vicini al punto d'interesse, simile a quella sul gradiente estratto dal SIFT[23].

I vettori del SURF si basano sulle risposte ai filtri di Haar del primo ordine, piuttosto che sul gradiente, sfruttando la velocità delle immagini integrali e usando solo 64 dimensioni. Questi accorgimenti, assieme ad un'indicizzazione in base al segno del Laplaciano, riducono notevolmente sia il tempo per il calcolo che quello per il matching delle feature, e, inoltre, è stato dimostrato che simultaneamente aumentano la robustezza.

La costruzione dei descrittori avviene in due fasi: per ogni punto d'interesse viene individuata l'orientazione principale e, in seguito, si costruisce una finestra centrata sul punto stesso, dalla quale, tramite l'uso congiunto dei filtri Haar e le immagini integrali, si estrae un vettore di 64 componenti⁶. La versione velocizzata di questo estrattore, chiamata U-SURF (Upright SURF), notevolmente più rapida dal punto di vista computazionale, non calcola l'orientazione per costruire il descrittore associato al punto estratto, risultando più sensibile ai cambi di orientazione. Tuttavia, per molte applicazioni in cui l'invarianza alla rotazione non è un fattore estremamente critico, l'U-SURF può essere utilizzato con l'ottenimento di robuste performance fino a rotazioni di 15° circa tra le immagini.

Per individuare l'orientazione principale del punto estratto, in modo da renderlo invariante alla rotazione, si calcolano le risposte al filtro Haar nelle direzioni x e y , entro una finestra circolare attorno al punto stesso, le quali poi vengono filtrate con una Gaussiana. In seguito, le risposte ottenute si rappresentano come punti nello spazio (vedi Figura 2.16), da cui si determina l'orientazione dominante che viene assegnata alla feature.

A questo punto si costruisce una finestra quadrata, centrata attorno al punto d'interesse e orientata lungo la direzione calcolata in precedenza, la cui dimensione dipende dalla scala alla quale la feature è stata rilevata (Figura 2.17).

Tale finestra viene divisa in 16 regioni quadrate e, per ciascuna di queste, si calcolano le risposte al filtro Haar su 25 punti campionati regolarmente, pesandole con una Gaussiana centrata, in modo da aumentare la robustezza rispetto alle deformazioni geometriche e agli errori di localizzazione. Tali risposte, denotate con d_x e d_y , vengono sommate su ciascuna porzione della finestra e i risultati formano un primo insieme di dati del descrittore. Per conservare l'informazione

⁶Il descrittore può anche essere *extended* a 128 componenti, ma poichè il guadagno in accuratezza è relativamente basso, in questo progetto si utilizza la versione standard, in modo da privilegiare la velocità di computazione.

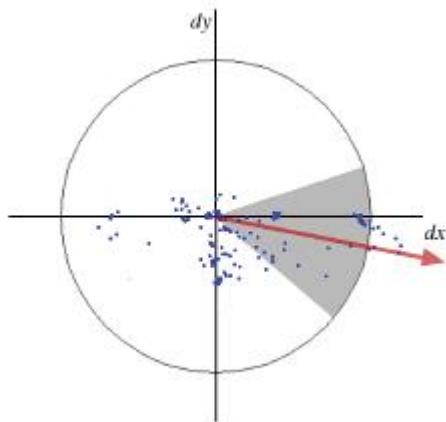


Figura 2.16: Calcolo dell'orientazione in base alle risposte al filtro Haar



Figura 2.17: Esempio di come sono costruite le finestre per il calcolo dei descrittori secondo l'orientazione assegnata

anche sulla polarità dei cambiamenti di intensità, si calcolano anche le somme dei valori assoluti delle risposte (Figura 2.18). Quindi, da ogni regione si ottiene un vettore \mathbf{v} , che rappresenta come è strutturata l'intensità di quest'ultima, costituito da quattro componenti: $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Concatenando tutti i vettori delle 16 diverse regioni in cui è stata divisa la finestra, si ottiene un descrittore di 64 elementi.

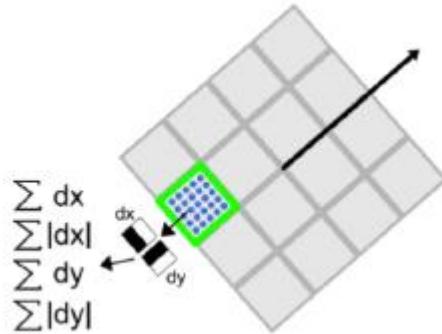


Figura 2.18: Componenti del descrittore calcolati su ogni sotto-finestra

Valutazioni sulle performance

Rispetto all'Harris detector descritto in precedenza, il SURF è sicuramente più pesante dal punto di vista computazionale, dovendo calcolare, oltre alla posizione delle feature, anche i descrittori associati a queste ultime.

Tuttavia, il SURF, proprio per la caratteristica di associare al punto un vettore descrittivo, ha una ripetibilità notevolmente maggiore, vantando quindi una superiore robustezza, nonché una buona invarianza alle rotazioni, ai cambiamenti di scala e ai cambi d'illuminazione nell'immagine.

Pur basando la sua informazione sulla distribuzione spaziale del gradiente, analogamente al SIFT, questo estrattore risulta migliore nella maggior parte dei casi. La sua superiorità è dovuta al fatto che il SURF integra l'informa-

zione sul gradiente calcolato su sotto-finestre, mentre il SIFT dipende solo dalle orientazioni dei gradienti individuali, risultando più sensibile al rumore.

2.2 Il matching

Il problema del *matching*, data una coppia di immagini stereo, consiste nell'associare le feature estratte in una vista con i punti corrispondenti nella seconda, in modo da poter ricavare mediante procedura di triangolazione la posizione del landmark nello spazio tridimensionale (Figura 2.19).

L'associazione dei punti è una parte molto delicata e difficile, poichè le fonti d'errore sono molteplici e anche piccole inesattezze spesso conducono a risultati molto differenti da quelli attesi.

L'assunzione generale è che le due immagini acquisite non sono molto diverse fra di loro e, quindi, l'intorno di una feature estratta non dovrebbe variare molto tra un frame e l'altro. Tuttavia, è molto probabile che un punto d'interesse di un'immagine abbia più di una possibile corrispondenza e diventa necessario scegliere l'accoppiamento migliore sulla base di qualche criterio.

Nell'affrontare tale problema, esistono particolari vincoli che limitano e quindi aiutano la ricerca delle corrispondenze, come per esempio il vincolo epipolare, approfondito in seguito, e il vincolo di ordinamento⁷.

2.2.1 La rettificazione delle immagini

La calibrazione di una telecamera stereo è un punto fondamentale nel processo dell'odometria visuale e, più in generale, nella visione artificiale. Esistono vari metodi, più o meno complicati, che permettono di trovare i parametri intrinseci

⁷Tale vincolo assume che, dati due punti P_1 e P_2 nella prima immagine, se P_1 è a sinistra rispetto a P_2 , la stessa relazione deve valere per i rispettivi corrispondenti nella seconda immagine.

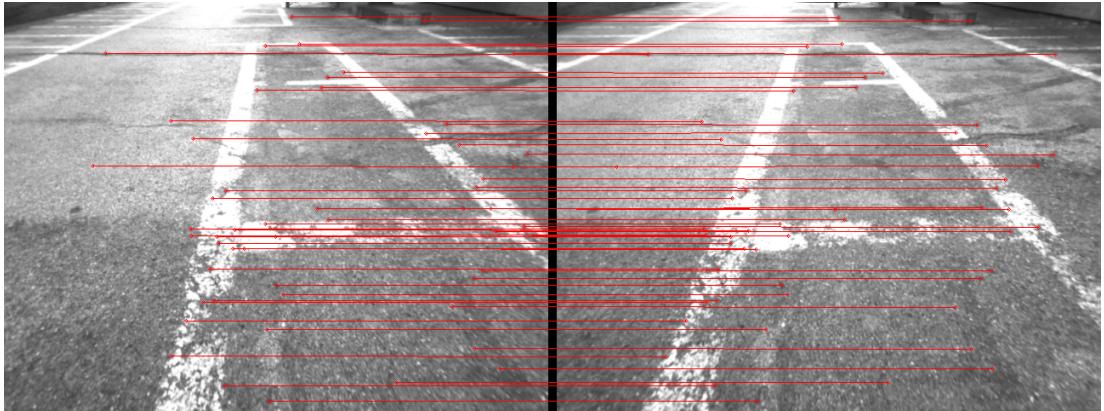


Figura 2.19: Un esempio di matching tra immagini stereo

di calibrazione, ma non saranno illustrati in questa tesi, poichè la tecnologia utilizzata in questo progetto permette di ottenere direttamente dal software della telecamera i dati necessari per una perfetta calibrazione.

In questo lavoro è stata utilizzata la telecamera stereo *Bumblebee2* (Capitolo 5.1.1), la quale fornisce una serie di caratteristiche di acquisizione, controllo e calibrazione delle immagini qualitativamente all'avanguardia.

Grazie alle librerie disponibili, quindi, le immagini in ingresso all'algoritmo di odometria visuale sviluppato risultano già pre-processate e rettificate attraverso alcune semplici chiamate a funzione standard all'interno del codice.

Con *pre-processing* si intende preparare le immagini “grezze” che vengono acquisite dai sensori della telecamera per poter essere utilizzate nei passi successivi del sistema. In particolare sono necessarie due elaborazioni delle immagini, prima che possano diventare disponibili per l'odometria visuale: il filtraggio e la loro rettificazione.

Il primo è uno step funzionale al secondo, poichè, oltre ad eliminare parte del rumore presente al momento dell'acquisizione delle immagini, il filtraggio permette di evitare lo sgradevole effetto dell'*aliasing*, che potrebbe essere intro-

dotto dalla rettificazione. Quest'ultima è il processo di correzione delle immagini in ingresso contro le possibili distorsioni delle lenti, che spesso possono capitare per disallineamenti fra le lenti e/o fra i sensori. La correzione determina una trasformazione su ciascun piano immagine in modo che coppie di linee epipolari coniugate diventino collineari e parallele a uno degli assi immagine (solitamente quello orizzontale). Le immagini rettificate possono quindi essere pensate come se fossero state acquisite da una camera ideale, con i sensori e le lenti perfettamente calibrate.

Un modello di telecamera *pinhole*⁸ implica che il corrispondente di un dato punto giaccia sulla sua linea epipolare nell'altra immagine. Facendo riferimento alla Figura 2.20, siano $C1$ e $C2$ i centri ottici delle due telecamere ed M il generico punto proiettato. Le due proiezioni $m1$ e $m2$ di M devono giacere sul piano P individuato dai punti $C1$, $C2$ e M . Quindi, supponendo di conoscere $m1$ (in questo caso la feature estratta), si sa che la proiezione di M sulla seconda vista, $m2$, deve giacere lungo la retta, detta epipolare, intersezione fra il piano immagine I_2 e il piano P .

Le corrispondenze fra i punti, quindi, possono essere trovate semplicemente controllando ogni punto sulla suddetta linea. Un grande vantaggio si ha quando le linee epipolari delle due immagini sono parallele e orizzontali, poichè i punti corrispondenti si trovano necessariamente sulla stessa riga nelle due viste: ciò si verifica solo nel caso in cui la terna di riferimento solidale con la seconda telecamera ha la stessa orientazione di quella solidale alla prima e si discosta da questa per una semplice traslazione ortogonale all'asse ottico. In un sistema di acquisizione generico, per ottenere tale vantaggio occorre rettificare le immagini, allineando le linee epipolari con le righe del piano immagine (Figura 2.21).

⁸Una telecamera pinhole consiste in una telecamera semplice, senza lenti e con una apertura singola molto piccola, in cui la luce proveniente dalla scena inquadrata passa attraverso un singolo punto e proietta un'immagine invertita sul piano immagine.

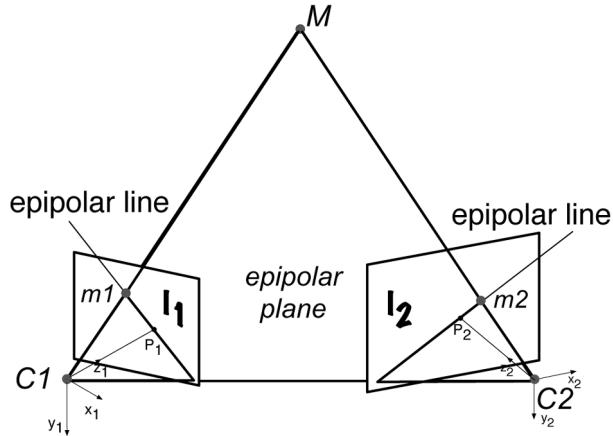


Figura 2.20: La geometria epipolare

Grazie alla telecamera stereo utilizzata in questo progetto, la *Bumblebee2*, la procedura di rettificazione è eseguita tramite una semplice chiamata a funzione disponibile con le librerie software della telecamera stessa.

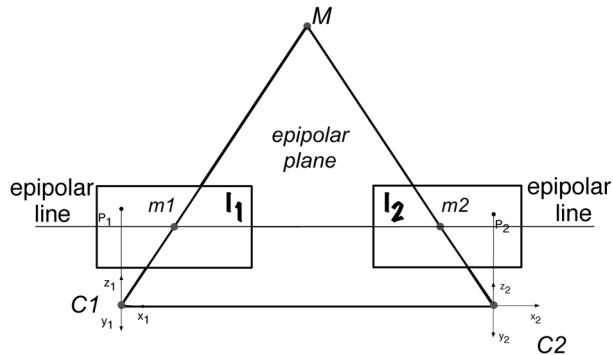


Figura 2.21: La geometria epipolare rettificata

A partire da questa semplificazione, esistono vari metodi con cui cercare le corrispondenze e dipendono fortemente dalla tipologia di feature estratte. Di seguito si descriveranno i due principali filoni sui quali sono basati i sistemi realizzati in questo progetto.

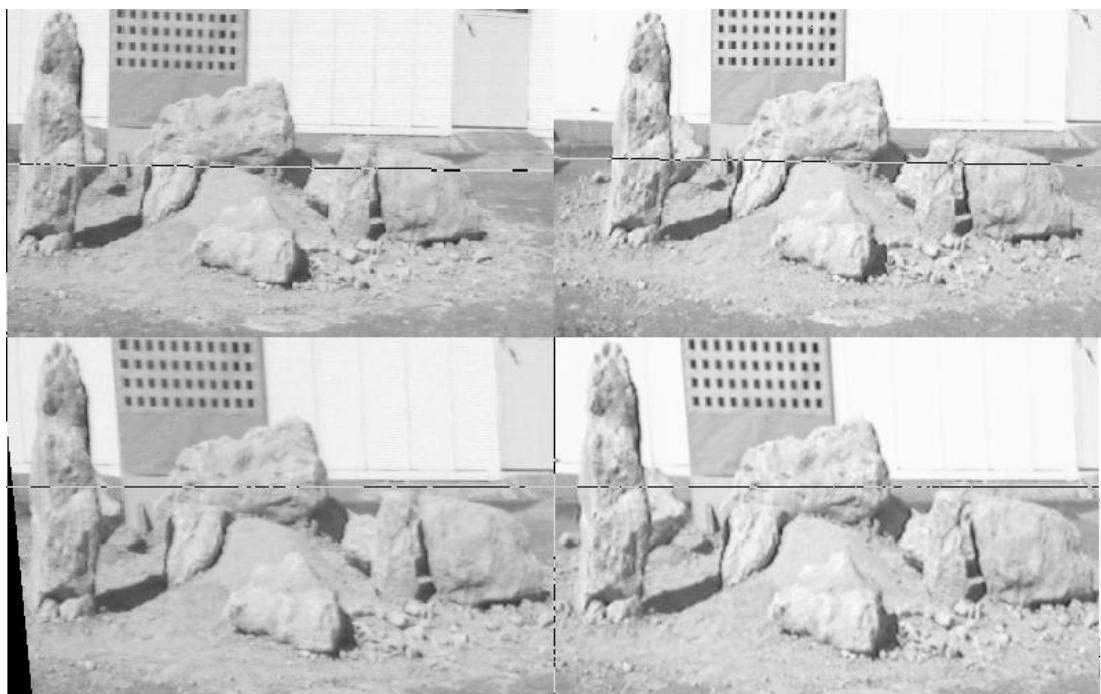


Figura 2.22: Immagini stereo originali e rettificate

2.2.2 Il matching correlation-based

Questo metodo, come dice il nome stesso, è fondato su una ricerca basata sulla misura di correlazione, la quale stabilisce se l'intorno di un punto nell'altra immagine è somigliante o meno a quello considerato. Tale tecnica, inoltre, è spesso utilizzata in combinazione con un estrattore di feature “semplice”, come per esempio l'Harris detector, che non associa al punto d'interesse alcun tipo di informazione aggiuntiva, come invece accade per altri estrattori come il SIFT o il SURF.

Dato un punto, si definisce una finestra $(N+1) \times (N+1)$ attorno ad esso come *template* per eseguire un confronto con i punti nell'altra immagine della coppia acquisita. A questo punto, occorre cercare un'altra finestra, di uguali dimensioni, che sia il più possibile somigliante al template definito e di conseguenza si troverà il punto corrispondente, che sarà il centro della finestra in questione.

Ovviamente, considerando immagini di dimensioni rilevanti (in questo progetto si sono utilizzate principalmente due risoluzioni: 1024×768 e 640×480), è temporalmente impossibile confrontare un punto di un'immagine con tutti quelli dell'altra. Inoltre, con un numero così elevato di candidati fra cui scegliere il match corretto, aumenta considerevolmente la possibilità di sbagliare e associare due feature simili ma diverse. Bisogna quindi necessariamente ridurre il set di punti a cui applicare la misura di correlazione, sia per motivi computazionali che per motivi di accuratezza. In questo senso, un aiuto fondamentale arriva proprio dal vincolo epipolare introdotto in precedenza, grazie al quale, uno stesso punto della scena inquadrata si trova in generale ad un'ascissa diversa nelle due immagini, ma sicuramente sulla stessa ordinata, riducendo in maniera notevole il numero di confronti da calcolare⁹.

⁹Si ricorda che la notazione delle coordinate immagine è quella standard che pone l'origine nell'angolo in alto a sinistra e identifica l'ascissa x con le colonne immagine e l'ordinata y con le righe.

Inoltre, se si tiene conto che la proiezione di un punto 3D nella vista di sinistra non potrà avere un'ascissa minore rispetto alla proiezione dello stesso nella vista di destra, allora si riducono anche i pixel candidati al matching sulla linea epipolare.

Stabilito dunque quali punti bisogna confrontare per poter individuare la giusta corrispondenza, bisogna scegliere quale tipo di misura di correlazione utilizzare, la quale rifletterà la somiglianza fra i punti delle due immagini.

Sum of Absolute Difference

Il metodo della “somma delle differenze assolute” è estremamente semplice e largamente usato negli algoritmi di stima del moto. Il suo funzionamento si basa sul valore assoluto della differenza tra ciascun pixel del template di base e il pixel corrispondente nella finestra usata per il confronto. Tali differenze sono sommate, creando una misura di similarità fra gli intorni dei due punti a confronto.

$$SAD(x_l, y_l, x_r, y_r) = \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} \sum_{j=-\frac{N}{2}}^{\frac{N}{2}} |I_{left}(x_l + i, y_l + j) - I_{right}(x_r + i, y_r + j)| \quad (2.13)$$

dove $P_l(x_l, y_l)$ e $P_r(x_r, y_r)$ sono i due punti, rispettivamente sulle immagini di sinistra e destra, dei quali si vuole trovare la misura di correlazione.

Punti molto simili avranno intorni altrettanto somiglianti e, di conseguenza, le differenze fra i vari pixel risulteranno limitate, se non nulle (caso limite in cui le due finestre coincidono perfettamente), per cui come somma si otterrà un valore basso. Si sceglierà quindi come corrispondenza il punto centrale appartenente alla finestra che ha dato come risultato il valore minore fra tutte le ricerche eseguite.

Questa tecnica è estremamente veloce per merito della sua semplicità ed inoltre possiede la caratteristica di essere anche parallelizzabile, poiché analizza ciascun pixel separatamente, rendendo così possibile l'implementazione di un codice estremamente veloce nel calcolo.

Cross-correlation

Per quantificare la somiglianza tra due finestre di correlazione, si possono scegliere criteri differenti dalla SAD, i quali producono risultati affidabili in un tempo di calcolo piuttosto accettabile.

Denotiamo con $I_l(x_l, y_l)$ e $I_r(x_r, y_r)$ i valori dell'intensità dei pixel rispettivamente del template base e della finestra da confrontare. Le finestre hanno in generale dimensioni $(N + 1) \times (N + 1)$, di conseguenza gli indici che appaiono nelle formule di seguito variano tra $-\frac{N}{2}$ e $+\frac{N}{2}$ sia per l'indice i che per l'indice j . Sono stati testati tutti i criteri sotto riportati e hanno mostrato risultati piuttosto analoghi.

$$C_1(\cdot) = \frac{\sum_{i,j} [I_l(x_l + i, y_l + j) - I_r(x_r + i, y_r + j)]^2}{\sqrt{\sum_{i,j} I_l(x_l + i, y_l + j)^2} \sqrt{\sum_{i,j} I_r(x_r + i, y_r + j)^2}}$$

$$C_2(\cdot) = \frac{\sum_{i,j} I_l(x_l + i, y_l + j) I_r(x_r + i, y_r + j)}{\sqrt{\sum_{i,j} I_l(x_l + i, y_l + j)^2} \sqrt{\sum_{i,j} I_r(x_r + i, y_r + j)^2}}$$

$$C_3(\cdot) = \frac{\sum_{i,j} [(I_l(x_l + i, y_l + j) - \overline{I_l(x_l, y_l)}) - (I_r(x_r + i, y_r + j) - \overline{I_r(x_r, y_r)})]^2}{\sqrt{\sum_{i,j} [I_l(x_l + i, y_l + j) - \overline{I_l(x_l, y_l)}]^2} \sqrt{\sum_{i,j} [I_r(x_r + i, y_r + j) - \overline{I_r(x_r, y_r)}]^2}}$$

$$C_4(\cdot) = \frac{\sum_{i,j} [I_l(x_l + i, y_l + j) - \overline{I_l(x_l, y_l)}] [I_r(x_r + i, y_r + j) - \overline{I_r(x_r, y_r)}]}{\sqrt{\sum_{i,j} [I_l(x_l + i, y_l + j) - \overline{I_l(x_l, y_l)}]^2} \sqrt{\sum_{i,j} [I_r(x_r + i, y_r + j) - \overline{I_r(x_r, y_r)}]^2}}$$

Il primo e il terzo criterio¹⁰ usano la differenza tra i livelli di grigio delle immagini, per cui il miglior match sarà quello che otterrà la misura di correlazione minore, mentre gli altri due moltiplicano fra loro i valori dei pixel e quindi, in questi casi, la scelta dovrà ricadere sul punto appartenente alla finestra che massimizza il risultato ottenuto. C_3 e C_4 sono simili a C_1 e C_2 rispettivamente, eccetto

¹⁰(\cdot) indica le variabili x, y

per il fatto che ad ogni elemento viene sottratto il valore medio delle intensità dei pixel nella finestra.

Dai test, si verifica che C_3 e C_4 mostrano performance leggermente migliori, perchè risultano essere più invarianti alle trasformazioni delle immagini. C_2 ha prestazioni simili a quest'ultime, tranne quando la differenza nella distribuzione dei livelli di grigio tra i frame diventa importante.

Come mostrato da Nishihara [28], la probabilità che un *mismatch* si verifichi diminuisce all'aumentare della dimensione della finestra di correlazione e della presenza di *texture*. Tuttavia, l'utilizzo di finestre eccessivamente grandi rende il calcolo della misura di correlazione molto pesante, rallentando notevolmente il processo di matching.

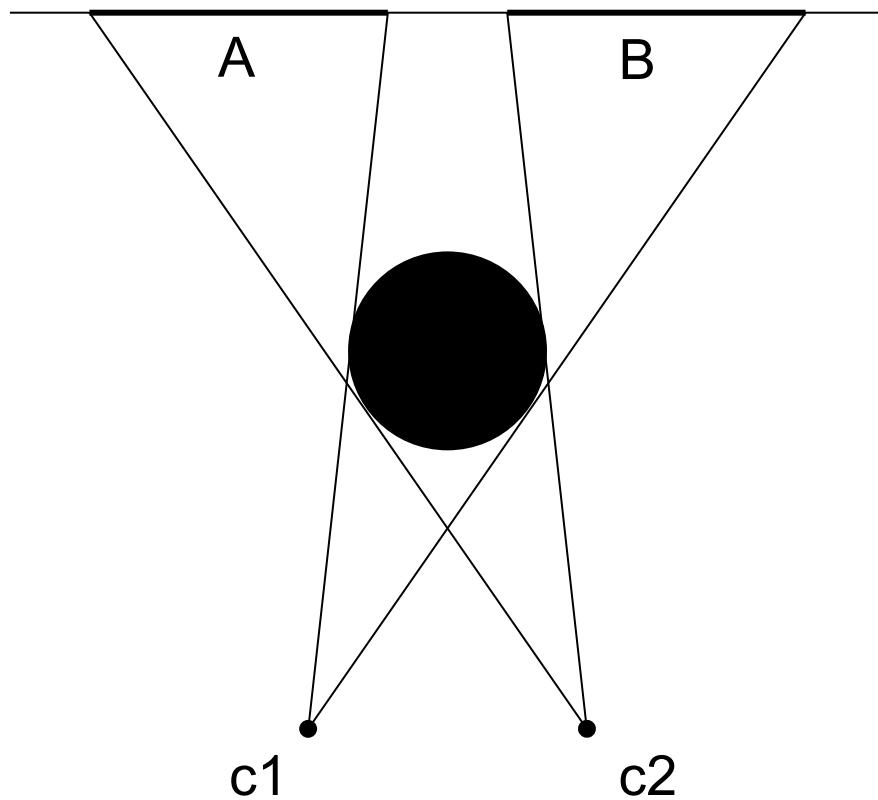


Figura 2.23: Un esempio di occlusione

Qualsiasi criterio di correlazione venga utilizzato, occorre prevedere la scelta

di una soglia sui valori di correlazione calcolati: tale vincolo dovrà prevenire il fatto che in alcuni casi il matching di un punto tra due viste differenti risulti impossibile da trovare, a causa ad esempio di occlusioni. Parti della scena, infatti, potrebbero apparire solamente in una delle due immagini stereo, come è il caso della Figura 2.23, in cui la parte di scena *A* è visibile solo dalla telecamera *c1*, mentre la parte di scena *B* viene inquadrata soltanto dalla telecamera *c2*.

2.2.3 Il matching descriptor-based

Quando l'estrattore di feature aggiunge all'informazione sulle coordinate del punto d'interesse anche un vettore contenente una descrizione del punto estratto, delle sue caratteristiche e del suo intorno, allora la ricerca delle corrispondenze può sfruttare tutta questa conoscenza sulle feature per velocizzare notevolmente l'operazione di matching.

In questo caso è fondamentale avere a disposizione le feature estratte sia a sinistra sia a destra, in modo che i confronti possano essere fatti direttamente sui punti rilevati e solo su quelli, senza tenere conto degli altri: data una feature nella vista di sinistra, la ricerca per la sua corrispondenza dovrà essere svolta solamente sulle feature estratte nel frame di destra, a differenza del matching basato su correlazione che deve controllare quasi tutti i punti che giacciono sulla stessa riga.

Inoltre, un altro aspetto decisivo per ottenere un buon livello di accuratezza con questo tipo di matching è l'utilizzo di un estrattore di feature che presenti un alto grado di ripetibilità, ossia, su immagini che differiscono di poco tra loro, come una coppia di immagini stereo, i punti d'interesse rilevati dovranno essere all'incirca gli stessi.

Sebbene questo sistema di matching sia molto vantaggioso dal punto di vista computazionale, occorre ricordare che necessita il calcolo di ben due estrazioni

su entrambe le viste e che tali operazioni, basandosi su detector che costruiscono anche i descrittori per ogni feature, richiedono un tempo di estrazione più elevato rispetto al caso degli estrattori più semplici.

Questa tecnica si basa essenzialmente sulla misura di similarità definita in termini di distanza euclidea tra i descrittori delle feature. In questo tipo di matching, viene utilizzato il noto metodo *Nearest-Neighbour-Ratio*[23]: dato un punto-feature F_1 nella vista di sinistra, si trova la corrispondenza con un punto-feature F_2 nella vista di destra se F_2 è il “vicino più vicino” (nearest neighbour) di F_1 (nello spazio delle feature), e il rapporto tra la distanza di F_1 da F_2 e quella di F_1 dal suo secondo vicino non eccede una soglia τ (nel riferimento citato in precedenza, l’autore suggerisce $\tau = 0.8$).

L’affidabilità di questo tipo di matching migliora notevolmente attraverso la bidirezionalità dell’operazione stessa: un match è valido solo se la feature F_1 , estratta sul frame di sinistra, seleziona come corrispondenza la feature F_2 sul frame di destra e, viceversa, quest’ultima sceglie come sua controparte la feature F_1 stessa (*mutual consistency check*). L’algoritmo di matching può essere ulteriormente migliorato sfruttando particolari vincoli, come quello epipolare e la disparità positiva, in modo da velocizzare ancor di più il tempo di calcolo ma soprattutto per controllare un minor numero di feature, diminuendo la possibilità di trovare dei *falsi positivi*¹¹.

2.3 La triangolazione stereoscopica

Una parte essenziale di tutto il sistema dell’odometria visuale è lo step di triangolazione, in cui, a partire dalle corrispondenze trovate nella fase di matching, si ottengono le coordinate 3D delle feature presenti nella scena (landmark), poichè

¹¹Un falso positivo è un punto che, pur non avendo una corrispondenza, trova un’associazione valida secondo il criterio di matching, conducendo così ad errori nella stima del movimento.

le proiezioni di uno stesso punto nelle due viste forniscono un'indicazione della posizione del punto stesso nello spazio.

Nel setup utilizzato in questo progetto, con due sensori di acquisizione posizionati ad una *baseline*¹² fissa, il processo di triangolazione si basa sulle proporzioni esistenti tra i triangoli simili che si formano in base alla geometria del sistema.

Per ottenere le coordinate tridimensionali, si considera la semplice struttura che rappresenta il sistema di acquisizione utilizzato, la quale prevede due telecamere identiche, poste ad una distanza fissata B tra loro (la baseline), con distanza focale uguale, pari a f , orientate in modo che gli assi ottici siano paralleli fra loro, con i piani focali coincidenti (Figura 2.25).

Il sistema di coordinate mondo C_W è posizionato al centro fra i due sistemi di acquisizione (Figura 2.24). Per i sistemi di riferimento di ciascuna delle telecamere, l'asse X si estende parallelo al piano immagine da sinistra a destra, l'asse Y si estende verso il basso parallelo al piano immagine, mentre l'asse Z coincide con l'asse ottico delle telecamere e l'origine coincide con il centro di proiezione delle lenti. Le coordinate immagine sono denotate con x e y , con gli assi paralleli al sistema di coordinate del sensore di acquisizione.

Definendo d_1 e d_2 come le distanze delle proiezioni del punto P dal centro immagine (c_x, c_y) lungo l'asse X , attraverso le relazioni fra i triangoli simili, si ottengono le seguenti relazioni

$$\begin{cases} \frac{Z}{\frac{B}{2}+X} = \frac{f}{d_1} \\ \frac{Z}{\frac{B}{2}-X} = \frac{f}{-d_2} \end{cases} \quad (2.14)$$

dalle quali si può ricavare il valore della profondità del punto P :

$$Z = \frac{B f}{d_1 - d_2} \quad (2.15)$$

Si noti che sostituendo i valori d_1 e d_2 con i loro equivalenti $x_l - c_x$ e $x_r - c_x$, la loro differenza è scrivibile come $x_l - x_r$, la quale corrisponde alla disparità d ,

¹²La baseline è la distanza fra i centri ottici dei due sensori.

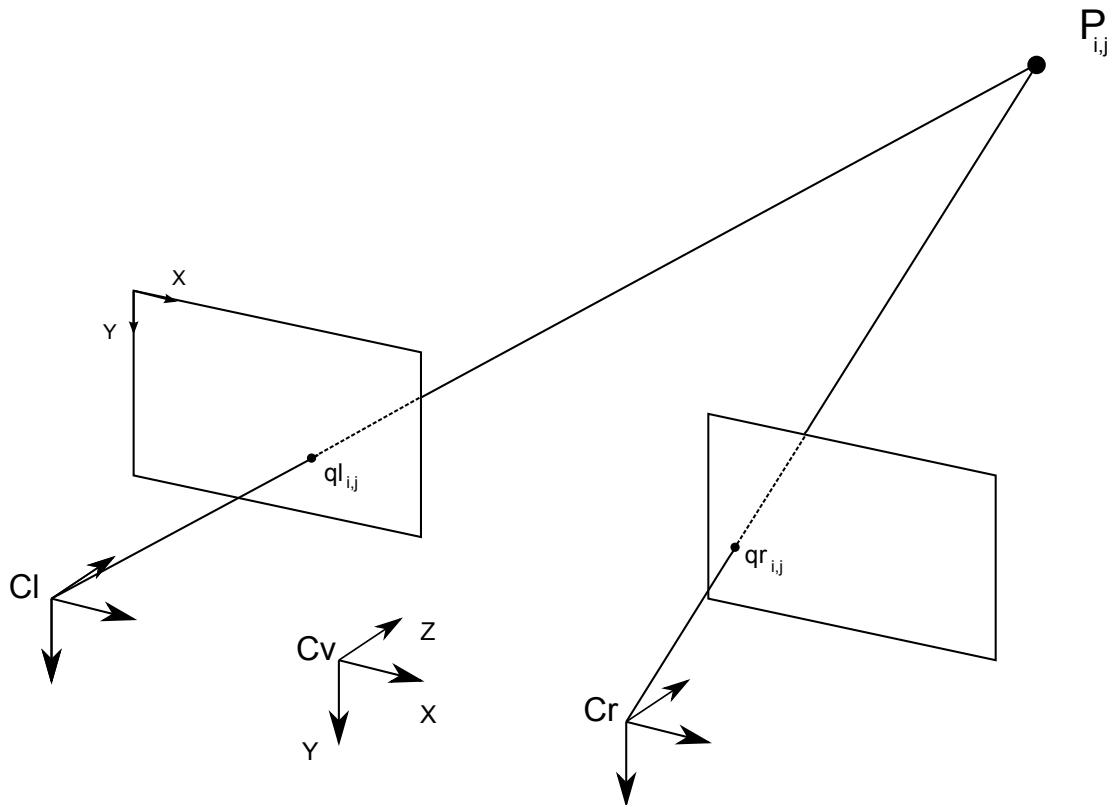


Figura 2.24: Il sistema di coordinate stereo

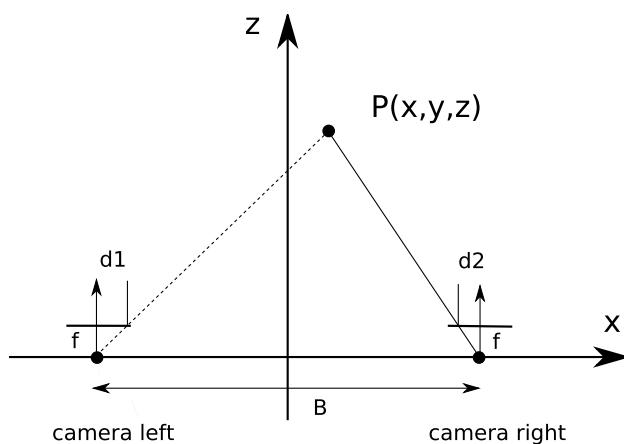


Figura 2.25: Geometria delle stereocamere

definita proprio come la differenza tra le ascisse delle due proiezioni di uno stesso punto su una coppia di immagini stereo.

Sostituendo il risultato trovato in una delle equazioni precedenti si ottiene:

$$X = \frac{B d_1}{d} - \frac{B}{2}. \quad (2.16)$$

Per ottenere il valore di Y , è necessario riscrivere le equazioni della prospettiva considerando l'asse Y, ricavando:

$$Y = \frac{B(y_l - c_y)}{d}. \quad (2.17)$$

Tuttavia, tale step introduce un'incertezza dovuta ad errori nella misurazione delle coordinate immagine. In questo progetto, come suggerito da Matthies e ripreso nell'algoritmo della missione MER, si associa una covarianza ai punti ottenuti tramite le formule di triangolazione. Infatti, come mostrato in [2], se si suppone di proiettare un punto reale P sul frame di sinistra alle coordinate (x_l, y_l) e analogamente su quello di destra in (x_r, y_r) , a causa di errori di misura, il sistema determinerà x_l e x_r con un certo errore, che a sua volta causerà una stima non corretta della posizione stessa del punto P . La Figura 2.26 illustra questo problema per errori causati dalla quantizzazione dell'immagine: a causa della risoluzione finita del sistema di acquisizione, la locazione stimata del punto P può giacere in qualsiasi punto nella regione che circonda la posizione reale.

Per tenere conto di tale incertezza, Matthies suggerisce di assumere un errore gaussiano bidimensionale nelle misurazioni delle coordinate immagine e di derivare le distribuzioni gaussiane 3D che descrivono l'errore nelle coordinate dei punti tridimensionali calcolati. Per tali punti, la vera distribuzione sarebbe non gaussiana, poichè la triangolazione è un'operazione non lineare. Nonostante ciò, si può approssimare come una gaussiana per semplicità e perchè fornisce un modello d'errore sufficientemente adeguato quando i punti non si trovano a distanze eccessive dal sistema di acquisizione.

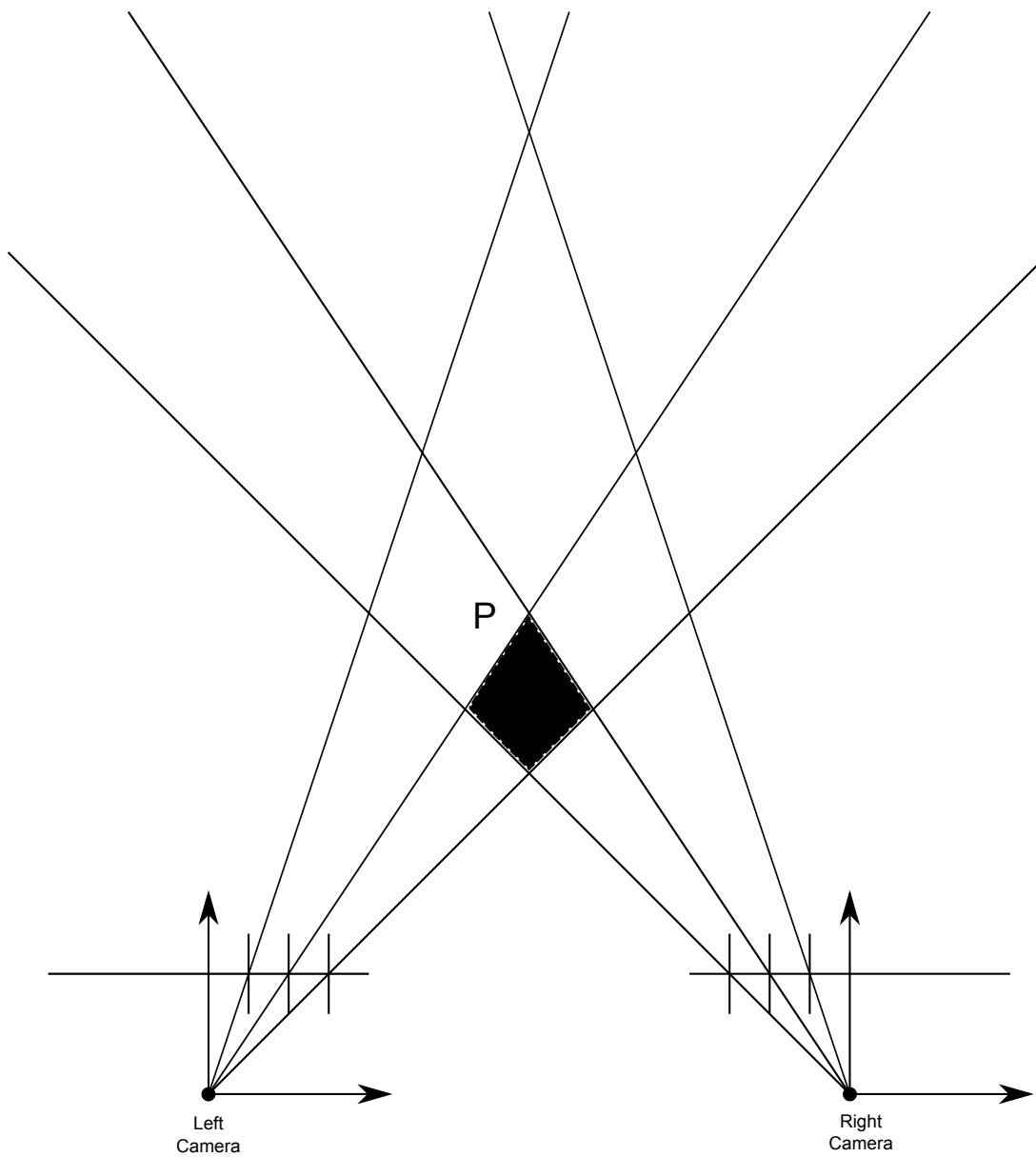


Figura 2.26: L'errore di quantizzazione

Siano date le coordinate immagine $l = [x_l, y_l]$ e $r = [x_r, y_r]$ sull'immagine di sinistra e di destra rispettivamente e si considerino come vettori aleatori a distribuzione normale con media μ_l e μ_r e matrici di covarianza Σ_l e Σ_r . Utilizzando le equazioni di triangolazione (2.3), si ottiene la stima delle coordinate del punto tridimensionale $[X, Y, Z]^T = f(l, r)$ e, in seguito, si ricavano le distribuzioni di X , Y e Z come funzioni dei vettori aleatori l ed r . Se la triangolazione fosse un'operazione lineare, P sarebbe normale con media $\mu_P = f(\mu_l, \mu_r)$ e covarianza

$$\Sigma_P = J \begin{bmatrix} \Sigma_l & 0 \\ 0 & \Sigma_r \end{bmatrix} J^T, \quad (2.18)$$

dove J è la matrice delle derivate prime parziali del punto rispetto alle coordinate immagine x_l, y_l, x_r, y_r .

Poichè f è non lineare, queste espressioni non sono soddisfatte esattamente ma si possono come approssimazione adeguata. I valori esatti di media e covarianza delle coordinate immagine da utilizzare nelle equazioni precedenti non sono noti, ma si può approssimare la media con le coordinate ottenute dallo step di matching e le covarianze con matrici identità. Ciò è equivalente a trattare le coordinate immagine come scorrelate fra loro con la varianza di un pixel. Tuttavia si potrebbe ottenere una migliore approssimazione delle matrici di covarianza per mezzo dei metodi proposti in [29, 30].

Geometricamente, questa scelta rappresenta la distribuzione di P come un ellissoide che approssima la densità dell'errore reale. Questo è illustrato in Figura 2.27, dove l'ellisse descrive il contorno del modello di errore e il diamante invece rappresenta l'errore di quantizzazione di Figura 2.26. Occorre notare che per punti vicini i contorni tenderanno ad essere più sferici, mentre più lontana sarà la stima dei punti, maggiore diventerà l'eccentricità¹³ dei contorni: questo perchè più un punto si trova distante dal sistema di acquisizione, maggiore è la

¹³L'eccentricità di un ellisse si può definire, in maniera intuitiva, come una misura di quanto il suo aspetto si discosti da quello di una circonferenza.

difficoltà e l'incertezza nello stabilire in maniera precisa la sua coordinata lungo l'asse ottico, motivo per cui la densità di probabilità dell'errore di misura cresce in quella direzione all'aumentare della distanza.

Le precedenti considerazioni illustrano l'importanza di modellare l'incertezza delle misure con una densità Gaussiana tridimensionale, piuttosto che solo con un fattore d'incertezza scalare, come è avvenuto in lavori antecedenti [1].

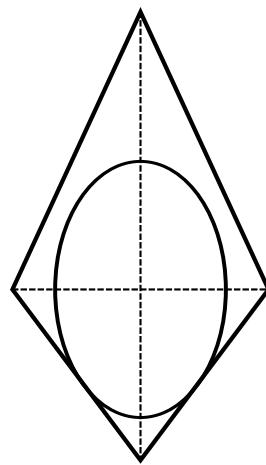


Figura 2.27: Approssimazione dell'errore di quantizzazione

I modelli d'errore scalari infatti sono equivalenti a matrici di covarianza diagonali $\Sigma = sI$, in cui I è una matrice identica 3×3 . Questo modello è appropriato quando i landmark sono molto vicini alla telecamera, ma fallisce in modo critico con l'aumentare della distanza. Infatti il limite di questa approssimazione è la sua incapacità nel rappresentare l'asimmetria della densità dell'errore reale causata della non linearità dell'operazione di triangolazione: questo effetto si manifesta in maniera maggiore quando i punti sono più distanti dalla telecamera.

2.4 Il tracking

La fase di tracking delle feature è simile a quella di matching e utilizza le stesse misure per stabilire la somiglianza tra i punti, ma anzichè applicarle sui punti dell’altra immagine stereo acquisita nel medesimo istante, cerca di trovare la corrispondenza in un frame temporalmente diverso.

Anche questa è una fra le operazioni più importanti all’interno dell’algoritmo di odometria visuale, in quanto dovrà “seguire” nei frame successivi le feature estratte nella coppia di immagini di partenza (Figura 2.28). Al termine di questa fase, un ulteriore step di matching e di triangolazione provvederà a generare un secondo set di punti 3D in modo da avere a disposizione le posizioni degli stessi landmark prima e dopo il movimento.

In letteratura esistono diversi algoritmi di tracking, tra i quali si può citare il KLT (*Kanade-Lucas-Tomasi*) feature tracker, che si basa sul precedente lavoro di Lucas e Kanade [31]. L’implementazione completa è stata sviluppata da Tomasi e Kanade [32]: le feature estratte dal Shi-Tomasi detector vengono inseguite per mezzo di un metodo che minimizza la differenza tra due finestre costruite attorno a ciascun punto. Tale tipo di algoritmo tuttavia richiede che l’acquisizione dei frame venga fatta in maniera ravvicinata nel tempo, in modo che immagini diverse si discostino minimamente; tutto ciò richiederebbe uno sforzo computazionale troppo elevato rispetto ai requisiti di un sistema adibito alla navigazione planetaria. Infatti, un rover spaziale non deve occuparsi esclusivamente del task di odometria visuale ma deve svolgere numerosi altri compiti, ragion per cui la CPU deve essere occupata il meno possibile.

Per l’obiettivo di questo progetto, occorre quindi un sistema di tracking con il compito di trovare la corrispondenza di una feature in un frame relativamente distante nel tempo da quello in cui tale punto è stato estratto, risultando più simile ad un matching, in cui però non si terrà conto del vincolo epipolare.



Figura 2.28: Un esempio di tracking catturato da un test di prova (notare la presenza di outlier descritti in seguito)

Le tecniche esistenti di feature tracking, esattamente come il matching, ricadono in due filoni principali: i metodi *descriptor-based* e quelli *correlation-based*.

Le tecniche basate sui descrittori estraggono un set di feature su ogni frame acquisito, e, successivamente, cercano di stabilire le corrispondenze tra i due insiemi di punti. Tali tecniche richiedono che vengano estratte pressapoco le stesse feature, in modo affidabile e consistente, in entrambi i frame: ovviamente ci sono problemi intrinseci dovuti al movimento del rover, che farà uscire dalla scena inquadrata feature estratte nel frame di partenza e, allo stesso tempo, favorirà l'ingresso di nuovi punti d'interesse, a causa dell'acquisizione di nuove porzioni di sfondo. Il grande svantaggio di tali tecniche è che gli errori nelle corrispondenze tendono ad essere molto grandi, se l'algoritmo non garantisce una buona affidabilità.

I metodi basati invece sulla correlazione necessitano solamente l'estrazione di un unico set di feature sul frame di partenza. La posizione dei punti estratti nei frame seguenti viene trovata compiendo una ricerca su una finestra di dimensioni opportune, cercando il template che meglio si correla con quello attorno al punto nel primo frame. Uno svantaggio di questa tecnica è che il tempo di calcolo necessario dipende fortemente dalle dimensioni della finestra sulla quale si esegue la ricerca. L'utilizzo di finestre di dimensioni ridotte richiede un tempo di calcolo minore, tuttavia è necessario disporre di una stima abbastanza accurata del punto in cui la feature si è spostata, proveniente da un sistema di odometria alternativo, tenendo conto del fatto che, inoltre, un errore proveniente da tale stima condurrebbe al fallimento della procedura di tracking. In assenza di questa informazione, la dimensione della finestra di ricerca sarà notevolmente maggiore, con un conseguente aumento dello sforzo computazionale, a meno di non mettere pesanti limiti sulla velocità del rover, sia lineare che angolare, facendo in modo che in frame successivi i punti da trackare non si spostino di troppi pixel. Nel caso in cui il rover superasse la massima velocità consentita, la fase di tracking

fallirebbe totalmente, poichè, soprattutto per i punti più vicini alla telecamera, l'algoritmo dovrebbe esplorare una finestra in cui la feature cercata potrebbe non esistere, conducendo quindi alla perdita della stessa, o, ancor peggio, ad una associazione errata.

Un ulteriore svantaggio sorge in presenza di rotazioni tra le immagini, le quali conducono a problemi nell'associazione dei vari punti.

2.5 La stima

Come step finale viene utilizzato un algoritmo che, a partire dagli insiemi di punti 3D ottenuti attraverso le fasi precedenti, genera una stima del movimento stesso.

In letteratura sono state adoperate varie tipologie di algoritmi, assieme ai quali spesso sono state integrate tecniche di individuazione e rimozione degli outlier¹⁴, come per esempio il RANSAC, in modo da rendere la stima finale più robusta ed accurata.

Si dedicherà il capitolo 3.1 all'illustrazione dell'algoritmo di stima utilizzato in questo progetto, il quale verrà descritto in modo dettagliato, essendo un passo fondamentale per ottenere un buon sistema di odometria visuale.

¹⁴Gli outlier sono, per definizione, in un insieme di misure, punti che rappresentano valori anomali, che sono quindi chiaramente distanti dalle altre osservazioni disponibili.

Capitolo 3

L'algoritmo di stima

3.1 Procedura di Stima

Dopo ogni procedura di matching e la rispettiva triangolazione, si dispone di un set di osservazioni

$$Q_{i,j} = P_{i,j} + v_{i,j} \quad (3.1)$$

dove $P_{i,j}$ è la posizione 3D della j^{th} feature nel frame corrente e $v_{i,j}$ è un vettore aleatorio Gaussiano a media nulla e covarianza Σ_v . In questa sezione, si illustrerà un metodo che utilizza le sequenze di osservazioni precedenti insieme alla equazione del movimento

$$P_{i,j} = R_i P_{i-1,j} + T_i \quad (3.2)$$

per calcolare le stime $\hat{\Theta}_i$, \hat{T}_i dei parametri del movimento, dove $\hat{\Theta}_i$ è il vettore costituito dai tre angoli estratti dalla matrice di rotazione R_i . Un aspetto che complica la stima è la non linearità dell'equazione del moto (3.2) siccome anche $P_{i-1,j}$ è incognita. Si supererà il problema della non linearità usando una formulazione ai minimi quadrati semplificata per calcolare una stima iniziale dei parametri del movimento, successivamente si linearizzerà l'equazione del moto in un intorno della stima iniziale e si userà un metodo di risoluzione iterativo.

Per chiarezza, si presenterà il problema in due step. Nel primo si discuterà la stima ai minimi quadrati iniziale dei parametri del movimento. Successivamente, verranno descritti la linearizzazione e il metodo iterativo derivando una stima maximum-likelihood. Siccome i passaggi seguenti si riferiranno sempre a coordinate frame “corrente” e “precedente”, la notazione verrà semplificata eliminando il pedice i dai parametri del moto. Inoltre si condenseranno gli indici nelle osservazioni e nelle feature scrivendo Q_{Pj} , Q_{Cj} , P_{Pj} , e P_{Cj} anzichè $Q_{i-1,j}$, $Q_{i,j}$, $P_{i-1,j}$, e $P_{i,j}$.

3.2 Stima ai Minimi Quadrati

Dalle equazioni delle osservazioni (3.1) e del moto (3.2), le osservazioni Q_{Pj} , Q_{Cj} eseguite da due successive coppie di immagini stereo sono messe in relazione ai parametri sconosciuti del moto e dei landmark da

$$Q_{Pj} = P_{Pj} + v_{Pj} \quad (3.3)$$

$$Q_{Cj} = RP_{Pj} + T + v_{Cj} \quad (3.4)$$

Ci saranno un paio di equazioni per ogni feature. Per ottenere una stima iniziale dei parametri del movimento, si ridurranno queste equazioni ad un problema standard ai minimi quadrati per il quale la soluzione esiste ed è nota. Questo è ottenuto eliminando P_{Pj} dalle equazioni (3.3), (3.4) e riscrivendo l'unica equazione rimanente in termini del vettore di errore residuo e_j :

$$e_j = Q_{Cj} - RQ_{Pj} - T \quad (3.5)$$

Prendendo la lunghezza al quadrato di ogni vettore di errore, applicando fattori di peso scalari w_j , e sommando su tutte le feature si ottiene la funzione di costo

$$M(R, T) = \sum_j w_j e_j^T e_j \quad (3.6)$$

La stima ai minimi quadrati è ottenuta minimizzando questa espressione rispetto a Θ e T . La procedura di risoluzione standard di differenziare rispetto a Θ e T non conduce ad un problema di ottimizzazione lineare. Tuttavia, una soluzione diretta è stata ottenuta per un problema analogo nell'analisi di dati psicometrici ([33, 34]). Questa soluzione aumenta l'equazione (3.6) con i moltiplicatori di Lagrange che vincolano R ad essere ortogonale. L'equazione risultante può essere risolta attraverso una decomposizione ai valori singolari per la matrice unica, ortogonale R e il vettore T che minimizza la (3.6).

I passaggi per trovare la soluzione sono i seguenti (ulteriori dettagli in [3]).

Siano

$$\begin{aligned} w &= \sum_j w_j \\ Q_c &= \sum_j w_j Q_{Cj} \\ Q_p &= \sum_j w_j Q_{Pj} \\ A &= \sum_j w_j Q_{Cj} Q_{Pj}^T \\ E &= A - \frac{1}{w} Q_c Q_p^T \end{aligned}$$

Sia $E = USV^T$ la singular value decomposition E . Allora

$$\hat{R} = UV^T \quad (3.7)$$

$$\hat{T} = \frac{1}{w} [Q_c - \hat{R} Q_p] \quad (3.8)$$

Siccome nel set di punti 3D utilizzato per la stima del movimento potrebbero essere presenti degli outlier, si è integrato il tutto con un processo **RANSAC** (Random Sample Consensus) in modo da rendere la stima più robusta ed eliminare i punti che rappresentano fonte di errore:

1. Un piccolo insieme di features (e.g., 3 o 4) è selezionato in modo casuale e la stima ai minimi quadrati viene applicata a questo set. Se i punti scelti sono vicini ad essere collineari, si scartano e si seleziona un altro insieme.
2. Per ogni feature viene calcolato $\hat{Q}_{Cj} = \hat{R}Q_{Pj} + \hat{T}$, ossia la stima del punto 3D dopo il movimento utilizzando la \hat{R} e la \hat{T} calcolate al punto precedente. A questo punto si confronta \hat{Q}_{Cj} con Q_{Cj} : se la differenza tra ogni elemento dei due vettori è minore di una certa soglia, il punto Q_{Cj} sarà dichiarato inlier.
3. I passi 1 e 2 sono ripetuti un numero fissato di volte e la stima del movimento con il più alto numero di inlier viene selezionata. A parità di numero di inlier viene scelta la trasformazione che minimizza la distanza tra i punti \hat{Q}_{Cj} e Q_{Cj} .
4. Una volta terminate le iterazioni si disporrà delle stime iniziali Θ_0 e T_0 ed un set di punti 3D inlier prima e dopo il movimento. Tutti questi dati verranno utilizzati per la successiva stima più accurata, la *maximum likelihood motion estimation*.

3.3 Maximum Likelihood Estimation

La precedente stima ai minimi quadrati di Θ e T è equivalente ad una stima maximum likelihood derivata da un modello di osservazione nel quale le matrici di covarianza dell'errore hanno la forma $\Sigma_v = sI$. La stima del moto risultante può essere sostanzialmente di accuratezza inferiore a quella derivata con un modello dell'errore completo. Sfortunatamente, l'uso di un modello dell'errore completo porta ad un problema di ottimizzazione non lineare che non ha una soluzione diretta. Per illustrare questo problema e mostrare come è stato risolto attraverso

linearizzazione, si userà un modello dell'errore completo per derivare una stima maximum likelihood di Θ e T.

Usando l'equazioni (3.3) e (3.4) per eliminare P_{Pj} come in precedenza, si ottiene

$$Q_{Cj} = RQ_{Pj} + T + v_j \quad (3.9)$$

dove v_j rappresenta l'incertezza sia in Q_{Cj} che nel prodotto RQ_{Pj} . Per semplicità, supponiamo che Q_{Pj} sia senza rumore, in modo che $v_j = v_{Cj}$. Allora la densità di probabilità congiunta condizionata delle osservazioni Q_{Cj} , dati Θ e T, è Gaussiana,

$$f(Q_{C1}, \dots, Q_{Cn} | \Theta, T) \propto \exp \left\{ -\frac{1}{2} \sum_j e_j^T W_j e_j \right\} \quad (3.10)$$

dove $e_j = Q_{Cj} - RQ_{Pj} - T$ e W_j è l'inversa della matrice di covarianza di v_j . Le stime maximum likelihood sono quelle che massimizzano questa densità. Questo è equivalente a trovare Θ e T che minimizzano la sommatoria nell'esponente:

$$\sum_j e_j^T W_j e_j \quad (3.11)$$

Sfortunatamente, il problema di minimizzazione è non lineare e la tecnica che risolve il problema (3.6) non fa lo stesso con il (3.11). Perciò si ricorrerà alla linearizzazione del problema e al calcolo della stima iterativamente.

La linearizzazione è ottenuta prendendo l'espansione del primo ordine dell'equazione (3.4) rispetto agli angoli di rotazione $\Theta_0(\theta_x, \theta_y, \theta_z)$:

$$Q_{Cj} = R P_{Pj} + T + v_{Cj} \quad (3.12)$$

$$\approx R_0 P_{Pj} + \left[\frac{d(R P_{Pj})}{d\Theta} \right]_0 (\Theta - \Theta_0) + T + v_{Cj} \quad (3.13)$$

$$= R_0 P_{Pj} + J_j (\Theta - \Theta_0) + T + v_{Cj} \quad (3.14)$$

R_0 denota la matrice di rotazione per Θ_0 (angolo di partenza) mentre la matrice Jacobiana 3×3 J_j (valutata per $\Theta = \Theta_0$) è data da:

$$J_j = \begin{bmatrix} R_x P_{Pj} & R_y P_{Pj} & R_z P_{Pj} \end{bmatrix} \quad (3.15)$$

R_x, R_y, R_z sono le derivate parziali della matrice di rotazione rispetto agli angoli di rotazione $\theta_x, \theta_y, \theta_z$, rispettivamente. Eliminando ancora una volta P_{Pj} si ottiene

$$Q_{Cj} = R_0 Q_{Pj} + J_j(\Theta - \Theta_0) + T + v_j \quad (3.16)$$

dove v_j è ancora un vettore di rumore Gaussiano, approssimativamente a media nulla e con covarianza $\Sigma_j = \Sigma_{Cj} + R_0 \Sigma_{Pj} R_0^T$. Σ_{Cj} è la covarianza dell'errore relativo all'osservazione Q_{Cj} e Σ_{Pj} è la covarianza dell'errore relativo all'osservazione Q_{Pj} . Entrambe sono calcolate nel seguente modo:

$$\Sigma_P = P' \begin{bmatrix} \Sigma_l & 0 \\ 0 & \Sigma_r \end{bmatrix} P'^T \quad (3.17)$$

dove P' è la matrice Jacobiana delle derivate prime parziali di P rispetto alle locazioni 2D delle feature nell'immagine sinistra e destra, mentre Σ_l e Σ_r sono le covarianze delle feature nell'immagine sinistra e destra rispettivamente. Nella nostra implementazione quest'ultime sono state considerate matrici identiche, il che equivale a supporre le coordinate immagine non correlate con varianza di un pixel.

Si riscriva l'equazione (3.16) come

$$Q_{Cj} - R_0 Q_{Pj} + J_j \Theta_0 = \begin{bmatrix} J_j & I \end{bmatrix} \begin{bmatrix} \Theta \\ T \end{bmatrix} + v_j \quad (3.18)$$

abbreviadola nel modo seguente:

$$Q_j = H_j \begin{bmatrix} \Theta \\ T \end{bmatrix} + v_j \quad (3.19)$$

dove $Q_j = Q_{Cj} - R_0 Q_{Pj} + J_j \Theta_0$ e $H_j = \begin{bmatrix} J_j & I \end{bmatrix}$. Questa equazione modella Q_j come una misura lineare dei parametri sconosciuti Θ e T , con un rumore Gaussiano additivo v_j . La stima maximum likelihood di Θ e T è ottenuta minimizzando la

funzione obiettivo (3.11), con $W_j = (\Sigma_{Cj} + R_0 \Sigma_{Pj} R_0^T)^{-1}$ e con il vettore di errore ridefinito come

$$e_j = Q_j - H_j \begin{bmatrix} \Theta \\ T \end{bmatrix} \quad (3.20)$$

Differenziando la funzione obiettivo rispetto a Θ e T e ponendo le derivate a zero, si ottiene il sistema lineare

$$\left[\sum_j H_j^T W_j H_j \right] \begin{bmatrix} \Theta \\ T \end{bmatrix} = \left[\sum_j H_j^T W_j Q_j \right] \quad (3.21)$$

Invertendo queste equazioni, la stima dei parametri del moto è data da

$$\hat{M} = \begin{bmatrix} \hat{\Theta} \\ \hat{T} \end{bmatrix} = \left[\sum_j H_j^T W_j H_j \right]^{-1} \left[\sum_j H_j^T W_j Q_j \right] \quad (3.22)$$

con una covarianza dell'errore pari a

$$\Sigma_M = \left[\sum_j H_j^T W_j H_j \right]^{-1}. \quad (3.23)$$

L'equazione del moto è poi ri-linearizzata intorno alla nuova stima (il nuovo Θ_0) e la soluzione è ricalcolata. L'intera procedura di linearizzazione e di calcolo è iterata finché $\hat{\Theta} \approx \Theta_0$.

Una soluzione più efficiente può essere ottenuta espandendo il sistema lineare (3.21)

$$\begin{bmatrix} \sum_j J_j^T W_j J_j & \sum_j J_j^T W_j \\ \sum_j W_j J_j & \sum_j W_j \end{bmatrix} \begin{bmatrix} \Theta \\ T \end{bmatrix} = \begin{bmatrix} \sum_j J_j^T W_j Q_j \\ \sum_j W_j Q_j \end{bmatrix} \quad (3.24)$$

Questo sistema partizionato 2×2 può essere risolto per T rispetto a Θ

$$\hat{T} = (\sum_j W_j)^{-1} \left[\sum_j W_j Q_j - (\sum_j W_j J_j) \Theta \right] \quad (3.25)$$

e poi solamente per Θ

$$\begin{aligned} \hat{\Theta} = & \left[\sum_j J_j^T W_j J_j - \sum_j J_j^T W_j (\sum_j W_j)^{-1} \sum_j W_j J_j \right]^{-1} \\ & \left[\sum_j J_j^T W_j Q_j - \sum_j J_j^T W_j (\sum_j W_j)^{-1} \sum_j W_j Q_j \right] \end{aligned} \quad (3.26)$$

Come nel caso dei minimi quadrati, si ottiene una soluzione per la traslazione ottima rispetto alla rotazione ottima. Siccome questa soluzione è basata sulla linearizzazione intorno ad un set di angoli Θ_0 , si itera la soluzione per Θ finché non converge, successivamente si calcola T come passo finale della soluzione.

Espandendo Q_j nella (3.25), T può essere espressa come

$$\begin{aligned}\hat{T} &= (\sum_j W_j)^{-1} \left[\sum_j W_j(Q_{Cj} - R_0 Q_{Pj} + J_j \Theta_0) - (\sum_j W_j J_j) \Theta \right] \\ &= (\sum_j W_j)^{-1} \left[\sum_j W_j(Q_{Cj} - R_0 Q_{Pj}) + \sum_j W_j J_j (\hat{\Theta} - \Theta_0) \right]\end{aligned}\quad (3.27)$$

Se $\hat{\Theta}$ ha raggiunto la convergenza, allora $\hat{\Theta} - \Theta_0 \approx 0$ e \hat{T} è ridotto a

$$\hat{T} = (\sum_j W_j)^{-1} \sum_j W_j(Q_{Cj} - \hat{R} Q_{Pj})\quad (3.28)$$

Riassumendo, dall'equazione (3.18)

$$Q_j = Q_{Cj} - R_0 Q_{Pj} + J_j \Theta_0 = \begin{bmatrix} J_j & I \end{bmatrix} \begin{bmatrix} \Theta \\ T \end{bmatrix} + v_j\quad (3.29)$$

si ottiene la rotazione attraverso

$$\begin{aligned}\hat{\Theta} &= \left[\sum_j J_j^T W_j J_j - \sum_j J_j^T W_j (\sum_j W_j)^{-1} \sum_j W_j J_j \right]^{-1} \\ &\quad \left[\sum_j J_j^T W_j Q_j - \sum_j J_j^T W_j (\sum_j W_j)^{-1} \sum_j W_j Q_j \right]\end{aligned}$$

Si itera il procedimento finché $\hat{\Theta} \approx \Theta_0$. Successivamente la traslazione ottima è ottenuta come

$$\hat{T} = (\sum_j W_j)^{-1} \sum_j W_j(Q_{Cj} - \hat{R} Q_{Pj})$$

Tale modello di stima associa una covarianza dell'errore, che modella il fattore d'incertezza associato alla stima fornita dall'algoritmo. Questa matrice di covarianza può essere calcolata dai termini che si formano nelle soluzioni di Θ e

T . L'espressione per la matrice di covarianza espansa diventa quindi

$$\Sigma_M = \begin{bmatrix} \sum_j J_j^T W_j J_j & \sum_j J_j^T W_j \\ \sum_j W_j J_j & \sum_j W_j \end{bmatrix}^{-1} \quad (3.30)$$

I componenti di questa matrice compaiono tutti come termini in (3.26) e in (3.28). Se, diversamente, si volessero utilizzare solo alcuni elementi della covarianza, l'inversa può essere ottenuta simbolicamente usando le matrici identità per invertire tale matrice a blocchi[35].

Il risultato è

$$\Sigma_M = \begin{bmatrix} N^{-1} & -N^{-1} \overline{J_j^T W_j W_j}^{-1} \\ -\overline{W_j}^{-1} \overline{W_j J_j} N^{-1} & \overline{W_j}^{-1} + \overline{W_j}^{-1} \overline{W_j J_j} N^{-1} \overline{J_j^T W_j W_j}^{-1} \end{bmatrix} \quad (3.31)$$

dove, per comodità di notazione si è sostituito la *overline* alla sommatoria sull'indice j e il termine

$$N = \left[\overline{J_j^T W_j J_j} - \overline{J_j^T W_j W_j} \overline{W_j W_j Q_j} \right]$$

compare nella soluzione di Θ . Di conseguenza, se sono necessarie solo alcune componenti della matrice di covarianza, questo è un modo più semplice per poterle calcolare.

Ricapitolando, si è utilizzato un modello completo di errore Gaussiano per derivare una stima maximum likelihood di Θ e T . Siccome il problema di ottimizzazione risultante è non lineare, si è linearizzato quest'ultimo e si è iterata la soluzione partendo da una stima iniziale ottenuta con il metodo ai minimi quadrati descritto in precedenza. Questa procedura iterativa è considerevolmente più onerosa rispetto alla stima ai minimi quadrati, tuttavia, produce una stima del moto molto più accurata. Una spiegazione intuitiva del perchè e di quando questo accade può essere ottenuta ricordando che la densità dell'errore di osservazione è relativamente compatta quando i landmark sono abbastanza vicini, ma si incrementa quando i landmark sono più distanti (Figura 3.1).

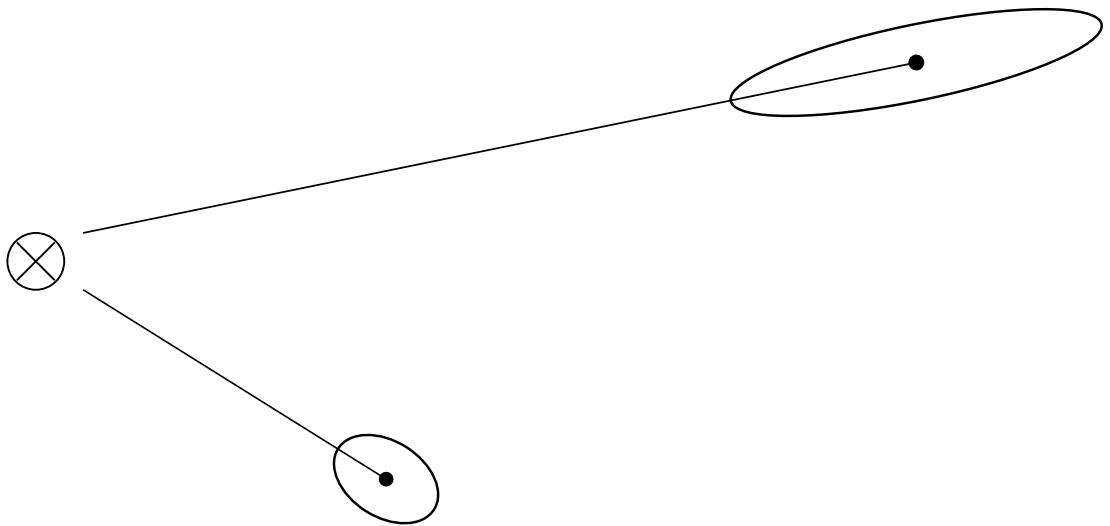


Figura 3.1: La densità dell'errore rispetto alla distanza del landmark

La funzione obiettivo usata nei minimi quadrati (3.6) fornisce un adeguato modello per la densità di errore per i casi in cui i landmark sono vicini ma non altrettanto nel caso di feature lontane, siccome non riflette l'eccentricità della densità. La funzione obiettivo utilizzata dall'approccio maximum likelihood (3.11) modella entrambi i casi attraverso la matrice inversa di covarianza W_j . Quando un landmark è vicino, W_j è quasi equivalente a $w_j I$, mentre quando è distante W_j effettivamente dà un peso minore agli errori di triangolazione lungo la direzione parallela all'asse ottico rispetto alla direzione perpendicolare. Data la natura della triangolazione, questa scelta risulta appropriata.

3.4 Composizione matrici di trasformazione

L'obiettivo della localizzazione è trovare la matrice di trasformazione $\langle_{rover}^{world} M$ ossia dove è posizionata la terna associata al rover rispetto alla terna mondo. Seguendo le note regole di composizione delle matrici di trasformazione in alberi

di posizionamento seriali, la matrice $\langle_{rover}^{world} M$ è scrivibile come:

$$\begin{aligned}\langle_{rover}^{world} M &= \langle_{camera}^{world} M \langle_{rover}^{camera} M \\ &= \langle_{camera(0)}^{world} M \langle_{camera(n)}^{camera(0)} M \langle_{rover}^{camera(n)} M,\end{aligned}$$

dove $\langle_{camera(0)}^{world} M$ indica com'è posizionata la terna associata alla camera all'istante 0¹ rispetto alla terna mondo, mentre $\langle_{rover}^{camera(n)} M$ indica la posizione della terna associata alla camera all'istante n rispetto alla terna rappresentante la posizione del rover. Entrambe le matrici sono costanti e vengono calcolate solamente all'inizializzazione del sistema tramite un'opportuna tecnica di calibrazione.

La matrice $\langle_{camera(n)}^{camera(0)} M$, invece, è la composizione delle matrici di trasformazione che rappresentano il cambio di posizione dalla terna associata alla camera tra l'istante 0 e l'istante n :

$$\langle_{camera(n)}^{camera(0)} M = \langle_{camera(1)}^{camera(0)} M \langle_{camera(2)}^{camera(1)} M \cdots \langle_{camera(n-1)}^{camera(n-2)} M \langle_{camera(n)}^{camera(n-1)} M$$

Occorre notare, tuttavia, che la procedura di stima utilizzata in questo progetto non fornisce $\langle_{i+1}^i M$, bensì $\langle_i^{i+1} M$. Sfruttando le proprietà delle matrici di trasformazione, occorre semplicemente invertire la matrice $\langle_i^{i+1} M$ per ottenere la matrice desiderata.

Tale inversione risulta poco onerosa dal punto di vista computazionale poiché ogni matrice di trasformazione è scrivibile come

$$\langle_a^b M = \begin{bmatrix} \langle_a^b R & \langle_a^b T \\ 0 \cdot 0 \cdot 0 & 1 \end{bmatrix} \quad (3.32)$$

e la rispettiva inversa è semplicemente:

$$\langle_a^b M^{-1} = \langle_b^a M = \begin{bmatrix} \langle_a^b R^T & -\langle_a^b R^T \langle_a^b T \\ 0 \cdot 0 \cdot 0 & 1 \end{bmatrix} \quad (3.33)$$

¹Istante in cui comincia la stima del movimento.

Capitolo 4

Visual Odometry @GRAAL

L'obiettivo di questa tesi è stato quello di realizzare un sistema di odometria visuale, che operi in tempo reale - ossia che renda la stima disponibile continuativamente nel tempo e non funzioni solo in modalità *stop-and-go* - e che risulti adatto ad un terreno simile a quello di un altro pianeta e quindi valido per ambienti outdoor percorribili da un rover.

Per far ciò, dopo un attento esame dei progetti esistenti in letteratura, si è tentato di replicare il sistema utilizzato dall'agenzia spaziale americana NASA nella missione MER, e, sapendo che tale setup è stato ed è tuttora utilizzato nella modalità stop-and-go, si è provveduto ad apportargli alcune modifiche in modo che potesse operare anche nella maniera stabilita per questo progetto.

Avendo in seguito riscontrato diverse problematiche nel suo funzionamento in modo continuo a causa della natura intrinseca dell'algoritmo, si è sostanzialmente modificato il sistema, tentando di seguire un approccio radicalmente diverso, basato su caratteristiche differenti da quello precedente e più adatte a fornire la stima del movimento continuativamente.

Di seguito verranno presentati i due approcci nel dettaglio, sottolineando le differenze esistenti nelle varie fasi standard dell'algoritmo illustrate nel capitolo 2, in modo da rendere chiare le varie tecniche ed anche i motivi per cui sono stati

ottenuti risultati differenti con i diversi sistemi.

4.1 Mars Exploration Rovers

Il software dei *Mars Exploration Rovers* [36] ha assicurato fino ad oggi che la navigazione sul pianeta rosso di *Spirit* e *Opportunity* avvenisse in sicurezza, interrompendo comandi di guida verso tipi di terreno potenzialmente pericolosi e fornendo check preventivi nelle zone molto sdruciolevoli, permettendo così il percorrimento di tragitti di media distanza in completa autonomia.

Tutto ciò è stato garantito dall'algoritmo di odometria visuale presente a bordo dei due rover, utilizzato in modo sempre più frequente ed efficace su entrambi, al prezzo dello sconveniente costo computazionale che necessita la schedulazione di tale task. Anche se tale ostacolo è in parte dovuto alla limitata potenza di calcolo delle risorse hardware a bordo¹, l'algoritmo è migliorabile in alcuni punti. Le ricerche in corso del *Mars Science Laboratory* si stanno occupando anche di questo, allo scopo di elaborare il sistema di odometria visuale per il rover *Curiosity*², il cui lancio è previsto per l'autunno del 2011.

La tecnica è stata sviluppata per un funzionamento in modalità *stop-and-go*, in cui tutti gli step dell'algoritmo - dall'acquisizione delle immagini al calcolo della stima - vengono eseguiti a motori fermi e quindi non c'è la improrogabile necessità che il risultato giunga in tempi brevi e non troppo variabili, in quanto il rover, prima di schedulare il successivo comando di guida, può aspettare la terminazione del calcolo precedente. In questo modo, i tempi del processo di odometria visuale possono dilatarsi, con una conseguente - del resto accettabile per i requisiti della missione - limitazione sulla possibilità di compiere lunghi tragitti giornalieri.

¹La frequenza del processore è pari a 20 MHz su un sistema operativo VxWorks.

²E' il nome del rover che verrà utilizzato nella missione in questione.

Poichè l'obiettivo di questo progetto è sviluppare un sistema che possa operare in tempo reale, in un primo momento si è cercato di replicare in modo completo l'algoritmo integrato su Spirit e Opportunity, poi sono stati implementati alcuni possibili miglioramenti proposti per la missione MSL, e, infine, si è tentato di soddisfare il requisito di elaborare i risultati con il veicolo in movimento, apportando alcune modifiche.

4.1.1 MER visual odometry

Il principio di funzionamento di questo sistema si basa principalmente sul lavoro di Matthies [2, 3]. L'algoritmo prende in ingresso due coppie di immagini stereo, acquisite rispettivamente prima e dopo il movimento del rover e una stima dello spostamento avvenuto tra i frame, proveniente da un'integrazione delle informazioni fornite dall'odometria meccanica e dal sistema di misura inerziale (accelerometri e giroscopi). Il risultato in uscita è composto una stima del cambiamento di posizione (X,Y,Z) e di orientazione (roll, pitch e yaw) del rover tra le due acquisizioni e una covarianza associata, che fornisce un'indicazione sulla bontà di tale risultato.

Occorre inoltre introdurre il fatto che il seguente algoritmo utilizza le immagini acquisite a diversi livelli di risoluzione, costruiti attraverso un sottocampionamento del frame di partenza. Tale tecnica, detta *a piramide*, permette di migliorare la velocità e l'accuratezza delle fasi di matching e di tracking, poichè i calcoli vengono distribuiti sui vari livelli, in modo che quando viene processata l'immagine a risoluzione maggiore le finestre di ricerca possono essere sufficientemente ridotte.

La selezione delle feature

I punti d'interesse vengono estratti nella vista sinistra della prima coppia d'immagini³ per mezzo di un estrattore basato su Harris corner. Nell'implementazione di questo sistema, il frame è stato diviso in diversi blocchi in modo che la selezione dei punti fosse effettuata su ciascuno di essi, ottenendo quindi un insieme di feature distribuite uniformemente sull'intera immagine, aumentando l'accuratezza della stima del moto.

Matching sulla prima coppia stereo

La fase di matching cerca di trovare per tutte le feature estratte nella prima immagine di sinistra il punto corrispondente nella vista destra utilizzando la *cross correlation pseudo-normalizzata*[1] per valutare quali punti associare e le piramidi d'immagini per velocizzare i tempi di calcolo della correlazione.

La locazione di ciascuna feature selezionata viene sottocampionata fino all'immagine a risoluzione minore, determinando la posizione del template di correlazione. Tale template sarà confrontato con porzioni d'immagine di dimensioni analoghe costruite attorno a ciascun punto della finestra di ricerca nel frame di destra. Grazie alla rettificazione, quest'ultima sarà un box contenente i pixel della stessa riga della feature, compresi fra i valori di disparità corrispondenti ai punti di minima e di massima profondità a cui la feature stessa potrebbe trovarsi (Figura 4.1).

La correlazione spaziale tra il template e le varie finestre indica quale sia il punto corrispondente nell'immagine destra a risoluzione minore. In seguito, la feature e il suo punto match vengono portati al livello di risoluzione successivo della piramide: così si stabilisce un template nell'immagine di sinistra, il qua-

³Con *prima* ci si riferirà ai frame acquisiti prima del movimento, mentre *seconda* indicherà la coppia d'immagini successiva.

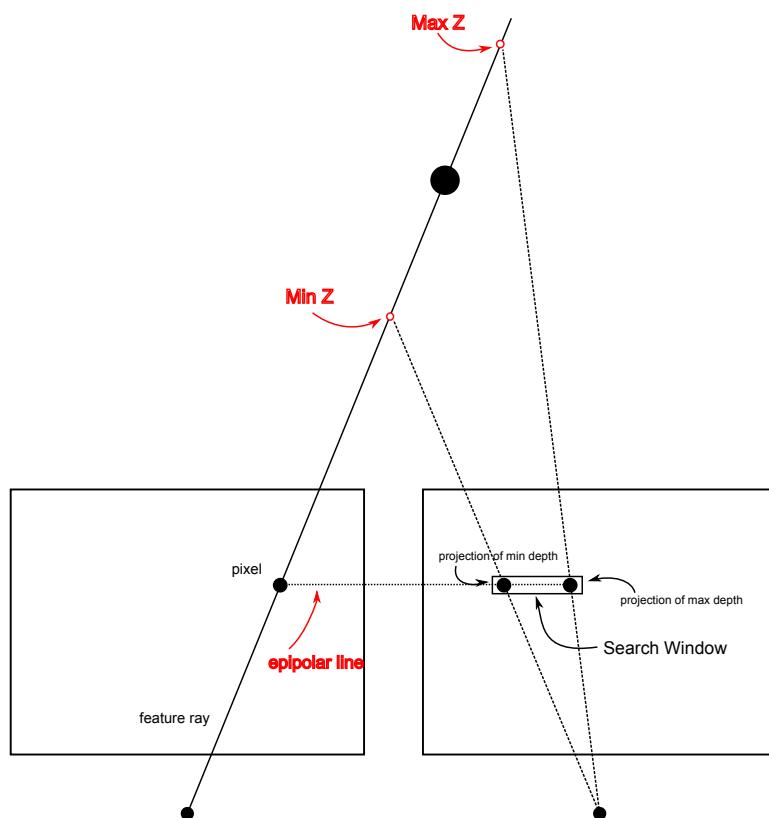


Figura 4.1: Costruzione della finestra di ricerca

le viene confrontato solo con i punti contenuti in una piccola finestra centrata attorno alla posizione sovraccampionata del match. Tale processo si ripete fino alla base della piramide, corrispondente con l'immagine inizialmente acquisita (Figura 4.2).

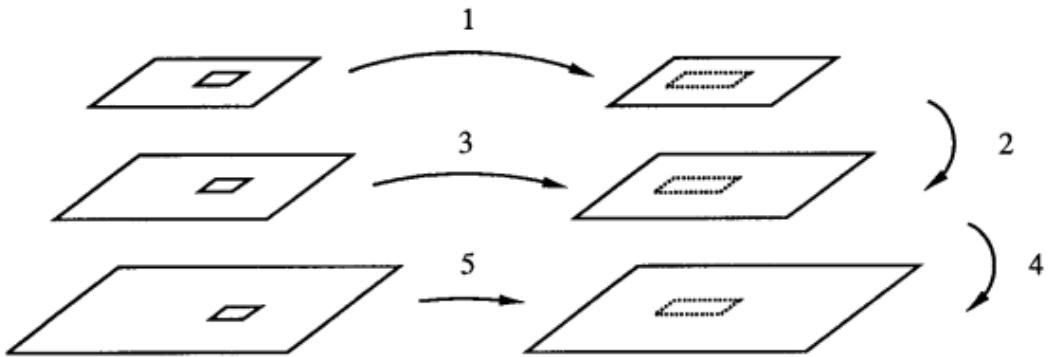


Figura 4.2: Il matching attraverso la piramide d'immagini

La soluzione adottata dalla missione MER, quindi, si basa sull'assunzione che le feature estratte sull'immagine in ingresso siano "buone" anche nei livelli successivi della piramide d'immagini costruita. In realtà, si verifica che tale ipotesi non è sempre vera e, in generale, quindi, non è garantito che un punto che è interessante ad una certa risoluzione, lo sia anche per quella inferiore.

Dopo aver trovato le associazioni, conservando solo quelle che avevano un valore di correlazione maggiore di una certa soglia, viene eseguita la triangolazione tra le coppie di punti ottenute per generare il primo insieme di coordinate tridimensionali e le loro rispettive covarianze.

Tracking delle feature

Le feature sono inseguite dall'immagine di sinistra della prima coppia stereo alla sua corrispondente della seconda acquisizione, utilizzando la stessa identica tecnica del matching, ossia la cross correlation attraverso la piramide d'immagini.

Tuttavia, i bound entro i quali verrà effettuata la ricerca del template devono essere impostati in maniera differente, poichè non sussiste più il vincolo geometrico epipolare.

Anzitutto si proietta ciascun punto 3D definito dalla precedente procedura di matching sulla seconda immagine di sinistra, utilizzando la stima del movimento fornita dalle informazioni provenienti da odometria meccanica e IMU. In seguito, attorno al punto proiettato, viene costruita un'area quadrata definita da un parametro fissato a priori entro la quale verrà effettuata la ricerca di correlazione. Si può notare che tale approccio conduce all'utilizzo di finestre di ricerca di dimensioni maggiori del necessario e quindi ad un costo computazionale inutilmente più elevato, poichè, per esempio, feature che sono più lontane dovrebbero spostarsi meno rispetto a quelle che si trovano più vicino al punto di vista.

Secondo stereo matching

Questo step è molto simile al matching fra la prima coppia di immagini, con alcune eccezioni: la posizione del template viene stabilita per mezzo delle coordinate del punto track⁴ nella seconda immagine di sinistra e la proiezione di questo pixel nella corrispondente immagine di destra, utilizzando la profondità del landmark calcolata attraverso il primo step di matching per impostare il centro della finestra di ricerca. I limiti di tale finestra si fissano sui parametri di minima e massima profondità prestabiliti e sono più limitati rispetto a quelli globali utilizzati nella procedura di tracking, in cui non si ha nessuna informazione sulla possibile profondità della feature.

Come nel tracking però, questo metodo non fa uso di alcuna stima del movimento per costruire una predizione sulla differenza di profondità del punto tra

⁴Un punto track è un punto appartenente alla seconda immagine trovato attraverso la procedura di tracking.

le due acquisizioni, con la conseguenza di rendere le finestre di ricerca più larghe del necessario.

Una seguente triangolazione tra le corrispondenze trovate tra le due viste genera un secondo insieme di punti tridimensionali con le rispettive covarianze.

La stima del movimento

I punti tridimensionali ricavati prima e dopo il movimento e le loro covarianze si utilizzano per determinare la misura dello spostamento avvenuto. Innanzitutto viene eseguito un preliminare test di consistenza geometrica dei landmark (*rigidity test*), basato sulla differenza delle distanze dei punti in ciascuna coppia di immagini, che trova eventuali outlier negli insiemi. In seguito si applica la tecnica dei Minimi Quadrati ai punti rimanenti, la quale genera una stima del moto preliminare, che inizializza il successivo algoritmo di Maximum Likelihood, trovando la soluzione ottima. I dettagli di questo processo sono stati esaminati nel capitolo 3.

4.1.2 Le modifiche proposte da MSL

Come evidenziato in precedenza, l'algoritmo di odometria visuale utilizzato dalla missione MER presenta alcuni difetti che MSL ha cercato di correggere[37]: ci sono varie parti del sistema che, se migliorate, lo renderebbero più efficiente, con la possibilità di aumentare il numero di feature da utilizzare nel calcolo ed eliminare persino la dipendenza da una stima iniziale del movimento avvenuto.

Dopo averne analizzato la tempistica, si è dedotto che il punto di criticità nel peso computazionale globale dell'algoritmo è rappresentato dalla ricerca basata sulla correlazione e proprio in questo senso si sono apportati vari correttivi per cercare di ridurre il più possibile il tempo impiegato nell'esecuzione del task.

Come mostrato in Figura 4.3, l’idea principale è quella di mantenere tutti gli step del sistema di base ma percorrerli in maniera differente: il sistema eseguirà dapprima tutto l’algoritmo solo sull’immagine a risoluzione più bassa - dall’estrazione delle feature alla stima Maximum Likelihood - e il risultato ottenuto, ossia una stima del movimento e la sua relativa covarianza, sarà utilizzato per ridurre le finestre di ricerca nelle successive iterazioni, sulle immagini a risoluzione maggiore.

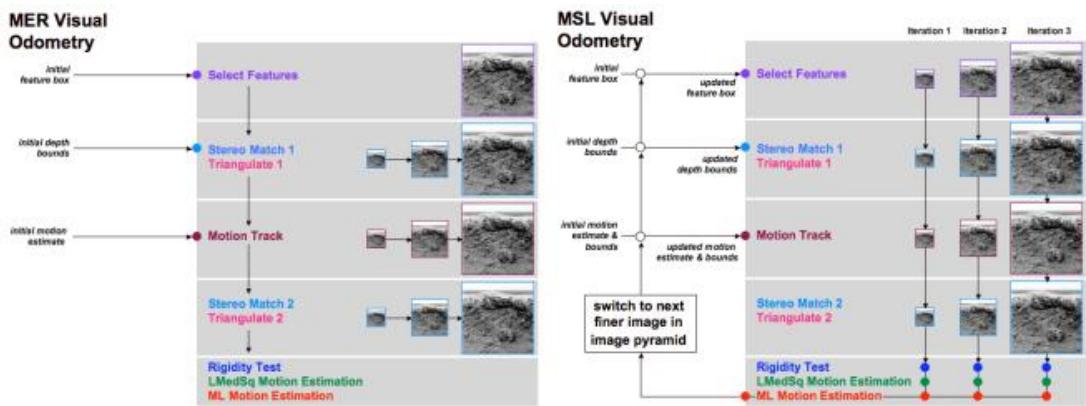


Figura 4.3: La diversa interpretazione dell’algoritmo

L’applicazione iterativa dei passi dell’algoritmo originale incrementa notevolmente il numero di feature inseguite, poichè, durante l’attraversamento della piramide, le finestre di ricerca si restringono basandosi sul fatto che la covarianza della stima del movimento diminuisce di livello in livello. Una volta raggiunta la base della piramide, le finestre saranno talmente piccole che la possibilità di effettuare un tracking errato a causa del rumore e/o di eventuali ambiguità dovrebbe essere notevolmente ridotta. In questo modo, quindi, anche il carico computazionale viene diminuito dal fatto che le porzioni d’immagine in cui viene effettuata la correlazione fra i punti sono larghe quanto basta per contenere la stima del movimento corrente e la sua incertezza. Inoltre, l’estrazione delle fea-

ture viene eseguita ad ogni livello della piramide, ma solamente all'interno della regione definita dalle feature valide nel precedente livello, evitando così l'inutile selezione di punti d'interesse che sarebbero poi nuovamente scartati a causa, per esempio, della loro uscita dall'inquadratura. L'idea che sta alla base di questa modifica è che l'algoritmo, durante le varie iterazioni, riesca ad "imparare" dove sono collocate le feature "buone" in modo da non dover perdere tempo di calcolo prezioso per l'inutile ricerca di punti che poi non serviranno per la stima finale.

Adattamento delle finestre di ricerca

Come detto, la sostanziale differenza proposta da MSL risiede nel fatto che le finestre di ricerca nella fase di tracking si restringono in base alla covarianza associata alla stima del movimento.

Nella prima iterazione dell'algoritmo, occorre stabilire in che modo impostare le finestre di ricerca, garantendo che contengano il punto cercato. Ci possono essere tre possibili approcci: il primo richiede la disponibilità di una stima del movimento proveniente da una sorgente esterna, in modo da poter individuare il punto attorno al quale costruire una regione in cui cercare la corrispondenza. Alternativamente, non avendo a disposizione alcuna stima del movimento, si può stabilire a priori uno spostamento massimo che può avvenire fra un frame e l'altro e trasformare tale informazione in una finestra attorno al punto di partenza nella seconda immagine. Infine, in mancanza di vincoli sulla velocità del rover, l'unica soluzione è quella di effettuare la ricerca sull'intera immagine, che, essendo di dimensioni ridotte grazie al sottocampionamento, non dovrebbe comportare un eccessivo carico computazionale.

Nei successivi livelli si utilizzano la stima del movimento e le incertezze associate, ottenute dall'iterazione precedente, per impostare più correttamente ed efficientemente le finestre in cui cercare il track delle feature. Tali finestre dovranno essere dimensionate in modo che qualsiasi tipo di trasformazione entro i

limiti d’incertezza del movimento proietti ciascuna feature all’interno della porzione d’immagine considerata. L’obiettivo è tradurre la stima e la sua incertezza nei limiti per la finestra di ricerca. Un modo intuitivo per specificare tale incertezza del movimento è di applicare dei bound ai valori d’errore massimo e minimo in tutti i sei gradi di libertà. La loro impostazione viene fatta nel modo seguente (Figura 4.4): in primo luogo si calcolano i limiti dovuti all’incertezza di tipo traslazionale, dopodichè si aggiungono ai primi quelli dovuti all’ambiguità rotazionale.

Per ciascuna feature appartenente alla prima coppia stereo, il punto tridimensionale associato viene trasformato sulla base della stima ottenuta all’iterazione precedente. In seguito si prendono, per ogni grado di libertà, due punti 3D la cui distanza dalla posizione del landmark corrisponde all’errore massimo nelle due direzioni e si proiettano sull’immagine, ottenendo così sei punti per l’incertezza traslazionale e altrettanti per quella rotazionale. Come si può vedere in Figura 4.4, per mezzo di questi punti si ottiene la finestra di ricerca globale, che racchiude l’incertezza su tutti i gradi di libertà.

La Figura 4.5 mette a confronto le finestre di ricerca utilizzate dall’algoritmo di base e da quello proposto da MSL, per un test eseguito su immagini acquisite dai rover su Marte durante un comando di guida in avanti.

Come si può notare, le finestre del primo caso sono tutte delle stesse dimensioni, mentre nel secondo caso cambiano dimensioni in base alla distanza delle feature dal sistema di acquisizione e al movimento previsto del rover. Con gli accorgimenti apportati, le finestre risultano più piccole per la ricerca di punti più distanti, poichè solo l’incertezza rotazionale ha effetti evidenti su tali finestre.

Le differenze fra i due algoritmi mostrano sostanzialmente due effetti principali: il primo che i tempi di calcolo sono differenti poichè l’algoritmo base utilizza finestre di dimensioni notevoli anche per punti molto distanti nell’immagine, impiegando molto più tempo per effettuare la correlazione; inoltre, con le

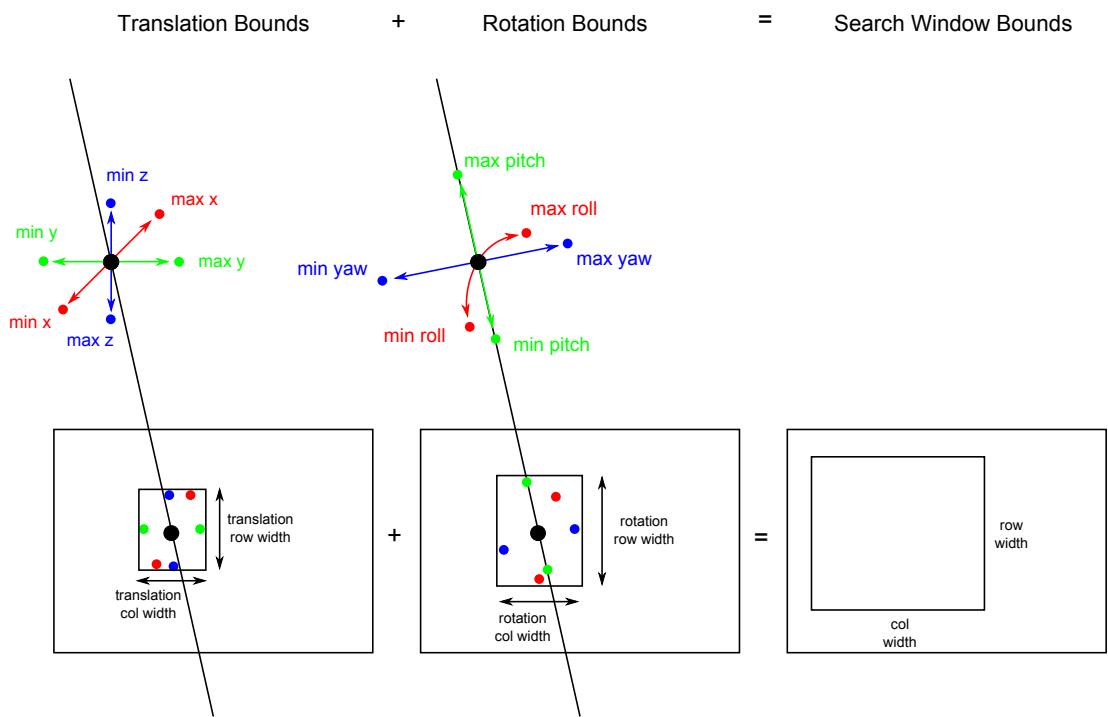


Figura 4.4: Costruzione delle finestre di ricerca per il tracking

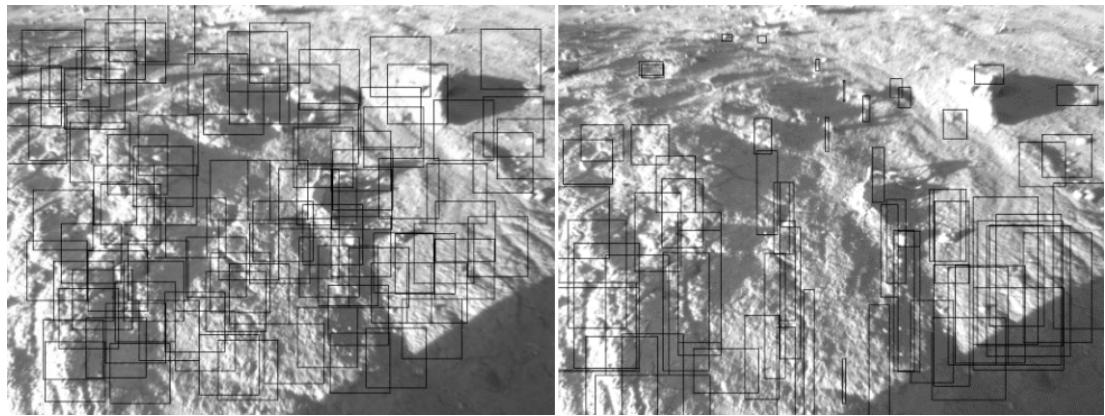


Figura 4.5: Differenza tra le finestre di tracking dei due algoritmi

modifiche apportate, il sistema riesce a tener traccia di un numero superiore di feature vicine al punto di vista, poichè in quelle posizioni sono necessarie finestre di dimensioni maggiori. Riassumendo, il risultato è una maggiore quantità di feature conservate combinata ad un minor carico computazionale richiesto da questo passo dell'algoritmo.

I benefici si mostrano in maniera più evidente ai livelli più bassi della piramide, poichè per ciascun sottolivello l'incertezza del movimento è basata sulla covarianza della stima risultante dal livello precedente e, durante l'attraversamento della piramide, tale incertezza dovrebbe diminuire causando un conseguente restringimento delle finestre di ricerca. Inoltre, abbinando l'impostazione delle dimensioni delle finestre all'incertezza della stima, la possibilità di eseguire un tracking non corretto a causa di rumore o ad un eventuale ambiguità dovrebbe decrescere ad ogni iterazione dell'algoritmo, rendendo a sua volta la stima sempre più accurata.

Per impostare i bound d'incertezza del movimento⁵ occorre utilizzare la covarianza della stima, ottenuta al livello precedente. Come spiegato nel capitolo 3, la covarianza C è una matrice 6×6 ordinata secondo (roll,pitch,yaw,x,y,z) data da:

$$C = \left[\sum_i H_j^T W_j H_j \right]^{-1}$$

$$H_j = [J_j \ I]$$
(4.1)

$$J_j = [R_x P_{Pj} \ R_y P_{Pj} \ R_z P_{Pj}]$$

dove R_x , R_y , R_z sono le derivate parziali di R rispetto ai tre angoli di rotazione e P_{Pj} è la posizione di un punto 3D nella prima coppia di immagini acquisita.

Calcolando attraverso la *Singular Value Decomposition* (SVD) una decomposizione di C , si ottengono sei autovettori e_i e altrettanti autovalori λ_i . Geometricamente i vettori e_i rappresentano le direzioni principali di un ellissoide a sei dimensioni, mentre i termini $\sqrt{\lambda_i}$ determinano le lunghezze dei suoi semiassi.

⁵Tali limiti consistono nel calcolare i valori minimi e massimi di errore sui sei gradi di libertà.

Per trasformare queste informazioni in bound che rappresentino l'incertezza del movimento stimato, si calcolano i limiti scalando gli assi principali di un fattore fisso (S) e proiettandoli sugli assi dello spazio del movimento. Il valore di S dovrebbe garantire le proiezioni dei punti sugli assi contengano completamente la covarianza del moto. I valori minimi e massimi di tali proiezioni sui sei assi principali definiscono i limiti del possibile movimento stimato.

$$\begin{aligned} \text{max_roll} &= S * \max_i(\sqrt{\lambda_i} e_i[0]) \\ \text{min_roll} &= S * \min_i(\sqrt{\lambda_i} e_i[0]) \\ &\vdots \\ \text{max_z} &= S * \max_i(\sqrt{\lambda_i} e_i[5]) \\ \text{min_z} &= S * \min_i(\sqrt{\lambda_i} e_i[5]) \end{aligned}$$

Tuttavia, dalle prove sperimentali si è verificato che con tale impostazione, le proiezioni degli ellissoidi non sempre contenevano completamente l'incertezza del movimento. Ragion per cui, nell'implementazione di questo progetto, si è scelto di calcolare le incertezze nel seguente modo:

$$\begin{aligned} \text{max_roll} &= S * \max_i |\sqrt{\lambda_i} e_i[0]| \\ \text{min_roll} &= -S * \max_i |\sqrt{\lambda_i} e_i[0]| \\ &\vdots \\ \text{max_z} &= S * \max_i |\sqrt{\lambda_i} e_i[5]| \\ \text{min_z} &= -S * \max_i |\sqrt{\lambda_i} e_i[5]| \end{aligned}$$

Implementazione del sistema

Come già accennato, il sistema precedente è stato ideato per un funzionamento in modalità *stop-and-go* ed inoltre si basa sulla possibilità di conoscere una stima del

movimento effettuato dall'integrazione delle informazioni provenienti dall'odometria meccanica e dall'IMU. L'obiettivo di questo progetto, invece, è di realizzare un algoritmo in grado di funzionare senza altre stime fornite da sorgenti esterne e in modo continuativo durante il moto.

Dopo aver implementato l'algoritmo sviluppato da MER con i miglioramenti proposti da MSL (seppur con qualche modifica), già dai primi test sperimentali si sono riscontrate alcune problematiche. Innanzitutto, in assenza di una stima preliminare, si è utilizzata come finestra di ricerca nel tracking l'intera immagine del livello più basso della piramide, e tuttavia, nonostante i tempi di calcolo in tale livello siano rimasti contenuti, si sono riscontrati numerosi match errati, a causa dell'eccessivo numero di punti confrontati. L'immediata conseguenza riscontrata consisteva nel fatto che alla stima iniziale del movimento effettuato tra i due frame, nella maggior parte dei casi, corrispondeva una covarianza elevata e, quindi, una conseguente impostazione di finestre di notevoli dimensioni nell'iterazione successiva (oltre ad un cospicuo dei tempi di calcolo necessari). Per questo motivo, tale approccio è stato cambiato ,imponendo forti vincoli sulla velocità massima consentita per il rover⁶, in modo da cercare di limitare le dimensioni delle finestre di tracking.

In questa versione dell'algoritmo, l'acquisizione delle immagini e la stima vengono eseguite il più velocemente possibile, senza alcuna pausa nel mezzo: ciò permette di avere stime molto ravvicinate fra loro, con piccoli spostamenti tra un'immagine e l'altra, soddisfacendo così il vincolo precedentemente imposto di uno spostamento massimo. Tuttavia, l'utilizzo di tali stime molto vicine fra loro ha l'inevitabile conseguenza che l'errore, accumulandosi nel tempo, aumenti in maniera significativa.

Per cercare di ridurre questo inconveniente, si è deciso di conservare in memo-

⁶Tale vincolo non comporta particolari problemi, poiché in ogni caso la velocità richiesta ai rover planetari risulta comunque molto bassa.

ria il primo frame utilizzato per le stime: dopo aver effettuato un certo spostamento, si riapplica l'intero algoritmo tra la prima immagine e l'ultima acquisita, avendo a disposizione, questa volta, una stima iniziale del movimento per poter limitare già dal primo livello le finestre di ricerca. Tale inizializzazione proviene dalla somma delle stime intermedie eseguite in precedenza.

Dopo aver testato tale implementazione con alcune prove, con spostamenti di pochi metri, si è subito notato l'inadeguatezza del sistema utilizzato: i tempi di calcolo sono risultati estremamente variabili e fortemente dipendenti dalla covarianza associata alla stima in ciascun livello. Infatti, nelle situazioni in cui l'incertezza sulla stima era elevata, il tempo necessario per ottenere il risultato aumentava eccessivamente, facendo cadere l'ipotesi assunta che la successiva immagine fosse acquisita prima di aver compiuto uno spostamento significativo.

Per tali motivi si è abbandonato questo tipo di approccio e si è pensato di affrontare il problema con un metodo diverso.

4.2 SURF-based visual odometry

Diversamente dal sistema realizzato in precedenza, in cui si è cercato di replicare per intero un algoritmo già sviluppato per poi adattarlo ai requisiti di questo progetto, questo secondo approccio non è basato su un particolare sistema esistente ma è un'implementazione originale, che parte dal lavoro di Nister[8] per poi rielaborarlo in maniere differente.

Il sistema citato utilizza l'Harris corner detector per estrarre le feature ma, a differenza di quello precedente, lo applica ad ogni immagine, sia sinistra che destra, e le fasi di matching e tracking vengono eseguite attraverso una ricerca di correlazione su finestre di dimensione 11×11 costruite solo attorno ai punti estratti.

A differenza dell'algoritmo elaborato da MER, il tracking viene eseguito su più immagini poichè, per garantire un buon matching e allo stesso tempo un buon tracking, l'acquisizione dei frame deve essere ravvicinata, essendo l'estrattore utilizzato poco robusto ai cambiamenti di scala.

Siccome in un sistema orientato ad applicazioni spaziali, troppe acquisizioni causerebbero un eccessivo sforzo computazionale, si è pensato di utilizzare un altro tipo di estrattore, che sia *scale invariant* e *rotation invariant*, in modo da poter catturare i frame in maniera più dilatata nel tempo. L'estrattore SURF è quello che soddisfa in modo più adeguato tali necessità.

4.2.1 Overview dell'algoritmo

Inizialmente era stato previsto che le estrazioni e il tracking fossero eseguite su tutte le immagini acquisite e che il matching, la triangolazione e la stima del movimento fossero svolte solo dopo un certo numero di frame, in modo da non accumulare troppi errori di natura integrativa. Partendo dalla prima coppia stereo, dalla quale si ricavava l'insieme iniziale di punti 3D, si eseguiva un inseguimento

delle feature in ognuno dei frame successivi: ciò si realizzava estraendo i punti d'interesse nelle viste di sinistra e eseguendo il tracking in un intorno del punto di cui si doveva tenere traccia. Dopo un numero predefinito di frame o nel caso in cui le feature inseguite fossero scese al di sotto di una determinata soglia, si realizzava il processo completo: venivano eseguite l'estrazione su entrambe le immagini, la ricerca dei match, la triangolazione ed infine la stima del movimento avvenuto tra il frame in questione e l'ultimo frame nel tempo in cui era stato calcolato l'insieme di punti tridimensionali.

Questa soluzione, pur avendo fornito risultati piuttosto promettenti in termini di accuratezza, si è dimostrata computazionalmente pesante: il tracking continuo su ogni frame acquisito comportava sia un rallentamento nel calcolo della stima del movimento (devono essere eseguite due estrazioni ad ogni passo di acquisizione), sia un eccessivo sovraccarico del processore.

Di conseguenza si è previsto un altro tipo di tracking, più snello dal punto di vista computazionale, che lascia il processore parzialmente a disposizione per utilizzi alternativi (durante l'intervallo di tempo in cui non deve eseguire calcoli per la stima del movimento).

Le immagini vengono così acquisite ad un *frame rate* fissato in base alla massima velocità prevista per il rover⁷ e memorizzate in un apposito buffer circolare. L'estrazione delle feature viene effettuata solo ogni k acquisizioni, dove k è definito come *frame-step* (mentre le immagini sulle quali viene eseguita la rilevazione dei punti vengono chiamate *keyframe*). Il *frame-step* ottimale risulta essere un compromesso tra lunghi step (minore carico computazionale e migliore accuratezza) e il bisogno di avere un numero sufficiente di feature fornite all'algoritmo di stima. Si parte da un valore iniziale predefinito (ad esempio 10 frame): se le feature rimaste per la stima del moto sono superiori ad una determinata soglia,

⁷Più la velocità è alta, più la frequenza di acquisizione deve aumentare, in quanto la scena inquadrata non deve variare eccessivamente.

viene mantenuto tale valore, altrimenti lo step viene dimezzato e vengono ripetute nuovamente le procedure di tracking e stima. Le situazioni più frequenti per cui ciò accade è quando il rover compie una rotazione: si noti che un rover autonomo conosce quali saranno i suoi prossimi comandi di guida e di conseguenza potrebbe preventivamente dimezzare il suo *frame-step* nel caso in cui debba ruotare.

4.2.2 L'estrazione delle feature

Come già accennato, l'estrattore prescelto è il SURF, i cui principi di funzionamento sono stati illustrati in modo esaustivo nei primi capitoli di questa tesi. Per la realizzazione di questo progetto è stata utilizzata la funzione di OpenCV, *cvExtractSURF*, sia nella sua versione originale, sia in quella che implementa l'*Upright SURF*, ottenuta partendo da tale funzione base e modificandola opportunamente per questo progetto.

L'U-SURF è più veloce del suo predecessore e mantiene comunque un alto fattore di distinguibilità delle feature. La sua implementazione non prevede il calcolo dell'orientazione del pattern, che è stato quindi eliminato dal codice originale, rendendo così il confronto fra i punti più sensibile alle rotazioni, rimanendo tuttavia abbastanza robusto per cambi di orientazione al di sotto di circa 15°[24].

In Tabella 4.1, a parità di feature rilevate, il tempo di estrazione risulta minore per l'algoritmo di U-SURF. Si può notare che aumentando il numero di punti estratti nell'immagine, la differenza fra i carichi computazionali dei due estrattori si fa sempre più marcata (il calcolatore sul quale sono stati eseguiti i test computazionali è dotato di un processore dual-core con frequenza di 3.16 *GHz*). Questa particolare implementazione ha semplicemente eliminato la parte di calcolo dell'orientazione della feature tralasciando ulteriori possibili accorgimenti che potrebbero diminuire ancor di più i tempi, anche perchè in questo progetto si è sfruttato il fatto di poter utilizzare in parallelo i due processori presenti sul

calcolatore utilizzato nei test, rendendo quindi già abbastanza rapida la versione originale.

Hessian threshold	U-SURF (ms)	SURF (ms)	punti estratti
500	187	203	1503
400	203	234	1821
300	219	265	2267
200	266	312	2898
100	297	375	3822

Tabella 4.1: Tempi di estrazione del SURF e dell'U-SURF

A differenza dell'approccio precedente, si avranno quindi a disposizione un set di feature estratte nell'immagine di sinistra e uno in quella di destra con i relativi descrittori, cioè i vettori contenenti le informazioni necessarie per trovare le corrispondenze tra i vari punti d'interesse, sia tra le coppie di frame stereo (matching), sia tra immagini acquisite ad istanti temporali diversi (tracking).

4.2.3 Il matching e la triangolazione

Per la fase di matching si utilizzano estrazioni delle feature sia nell'immagine di sinistra che in quella di destra, in modo da eseguire una ricerca delle corrispondenze basata solo sui descrittori dei punti estratti, riducendo significativamente il tempo di questa procedura. Ciò è molto importante, poichè la doppia estrazione eseguita su entrambe le viste occupa già un tempo considerevole e quindi diventa fondamentale limitare il più possibile la durata delle fasi successive, per consentire all'algoritmo di operare in tempo reale.

Il confronto fra le feature di due immagini stereo, inoltre, sfrutta come in precedenza il vincolo geometrico epipolare dell'immagine rettificata e la positività

della disparità per restringere ulteriormente l’insieme di possibili punti candidati al matching. Questo procedimento viene fatto in maniera bidirezionale: ad ogni feature sulla vista di sinistra se ne associa una sull’immagine di destra, e per ciascuno dei punti estratti a destra se ne associa uno sulla vista di sinistra. Un match sarà valido solo se il punto assegnato come “miglior compagno” alla feature di sinistra ha trovato a sua volta come corrispondenza la feature stessa (*mutual consistency check*). Tutto ciò è computazionalmente fattibile poichè il numero di confronti da fare per ciascuna feature non è elevato, dato che solo i punti estratti nell’altra vista e in un intorno della stessa riga devono essere controllati. Tale tecnica di mutua consistenza aggiunge una buona robustezza allo step di matching, riducendo significativamente la possibilità di associazioni errate causate da occlusioni o dalla perdita di feature da una vista all’altra. Per esempio, difficilmente tale controllo permetterà che una feature estratta a sinistra venga erroneamente associata ad un punto differente nel caso in cui nell’immagine di destra il punto esatto sia andato fuori scena: infatti, il punto selezionato dovrebbe trovare a sua volta nell’immagine di sinistra un’altra feature (si suppone quella esatta), rendendo non valida l’associazione precedente.

Il confronto fra i descrittori delle feature viene fatto tramite la distanza euclidea: la feature scelta sarà quella che otterrà la distanza minore fra i vettori, così definita:

$$\overline{D_{F_1} D_{F_2}} = \sqrt{\sum_{i=0}^{63} (D_{F_1}[i] - D_{F_2}[i])^2} \quad (4.2)$$

dove D_{F_1} e D_{F_2} sono i descrittori associati rispettivamente alla feature F_1 e alla feature F_2 ⁸.

Inoltre, come illustrato nel capitolo 2, viene utilizzata la *nearest-neighbour-ratio*, per cui viene selezionata la feature che risulta “più vicina” solo se il rapporto tra la distanza dal descrittore di tal punto e quella dal secondo *neighbour* più

⁸L’indice i varia tra 0 e 63, poichè la lunghezza utilizzata per il descrittore è di 64 unità.

vicino non supera una soglia prefissata. Ad esempio, siano $dist_1$ la distanza del descrittore “più vicino” a quello della feature considerata e $dist_2$ la distanza del secondo descrittore “più vicino”, allora il punto corrispondente al vettore con distanza $dist_1$ verrà selezionato solo se $dist_1/dist_2 < threshold^9$.

Con un tale setup e sui calcolatori utilizzati in questo progetto, la fase di matching impiega un tempo computazionale che risulta essere trascurabile rispetto al passo totale di calcolo della stima da keyframe a keyframe. Inoltre dai risultati ottenuti, la procedura si è dimostrata efficiente ed accurata, con un tasso d’errore estremamente ridotto.

Una volta trovate le corrispondenze tra due coppie di immagini stereo per un determinato keyframe, si ottengono dunque due set di punti immagine opportunamente associati fra loro e da essi, tramite l’operazione di triangolazione, si trovano i punti tridimensionali, oggetto dell’algoritmo di stima.

4.2.4 Il tracking fra i keyframe

Sostanziali differenze con l’approccio precedente si individuano anche in questo step e risiedono principalmente nel fatto che nell’implementazione di questo tipo di tracking non viene utilizzata alcun tipo di stima proveniente da un qualche sistema esterno (i.e. odometria delle ruote) per indicare in che punto costruire la finestra di ricerca della feature da inseguire; inoltre, a differenza del sistema basato sulle piramidi, non esistono porzioni d’immagine su cui verrà eseguito il tracking ma solo punti fra cui bisogna stabilire la corrispondenza.

Passando dal keyframe i al keyframe $i+1$, l’obiettivo di questa fase è quello di trovare i punti appartenenti al frame $i+1$ associati ad ognuna delle feature estratte nella coppia di immagini i , per la quale sono state calcolate le corrispondenze fra le due viste ed è stata effettuata la triangolazione.

⁹In questo progetto la threshold è stata fissata a 0.7

Per trovare le varie corrispondenze fra i punti del frame precedente e di quello successivo al movimento si utilizzano le stesse tecniche su cui si basa la procedura di matching: si esegue la ricerca della feature associata al descrittore che si trova alla minima distanza euclidea (Equazione 4.2) da quello confrontato e che non sia troppo vicina al secondo descrittore più vicino (*Nearest-Neighbour-Ratio*). Si può notare che è stato possibile rendere questo sistema totalmente indipendente da una stima iniziale, poichè non c'è la necessità di conoscere in quale porzione d'immagine cercare il punto track.

Tale caratteristica è molto importante poichè garantisce il funzionamento del sistema anche in occasione di un eventuale fallimento da parte degli altri sistemi di odometria: per esempio, se in un certo tratto del percorso, il rover attraversa un terreno molto sdrucciolevole e di conseguenza l'odometria meccanica fornisce risultati inesatti, l'odometria visuale, essendo completamente indipendente, non viene influenzata ed elabora la stima corretta del movimento eseguito. Ciò non avviene invece nel sistema utilizzato da MER[36]: un errore rilevante da parte dell'odometria porterebbe al collocamento errato delle finestre in cui cercare il punto corretto e quindi al fallimento della procedura d'inseguimento delle feature ed al conseguente insuccesso del sistema di odometria visuale.

Una volta che i punti sono stati associati tra immagini temporalmente distanti fra loro e dopo che è stato generato dal processo di triangolazione un secondo insieme di punti tridimensionali, i due set di punti vengono forniti all'algoritmo di stima.

Capitolo 5

Prove Sperimentali

Le prove sono state svolte su entrambe le tipologie di algoritmi realizzate, ma, come già anticipato, a causa delle scarse performance ottenute dal primo, i test sono stati concentrati sul secondo metodo implementato. I risultati forniti quindi si riferiscono solamente a tale tecnica, che è stata testata preliminarmente in ambiente indoor; dopo aver verificato che operasse in modo sufficientemente accurato all'interno, si è provveduto a eseguire una serie di prove all'esterno, in modo da ottenere una stima sul suo eventuale funzionamento in un ambiente esterno, dato che l'obiettivo finale è la navigazione in un contesto spaziale.

Alcuni test sono stati effettuati online, per mezzo di un notebook collocato a bordo del rover, collegato direttamente al sistema di acquisizione: utilizzando una CPU Intel Core2 da 1.66 GHz , il tempo necessario per la doppia estrazione (sinistra e destra) è in media 0.5 secondi e quello che intercorre tra l'acquisizione dei keyframe e il calcolo della stima è in media 1.1 secondi. A seconda del frame rate con cui si acquisiscono le immagini e lo step fra i keyframe, il carico computazionale varia di conseguenza. Per esempio con uno step di 10 frame e una frequenza di acquisizione di circa 5 fps , la CPU sarà occupata circa per il 50% del tempo, mentre con lo stesso frame rate ma con uno step superiore l'occupazione del processore diminuirà.

Per quanto riguarda la frequenza di acquisizione nelle prove effettuate si è utilizzato un rate di circa 5 *fps*, con una velocità massima del rover di 0.20 *m/s*. Si noti che, diminuendo la velocità massima consentita (i rover spaziali si muovono a velocità notevolmente più basse di quelle utilizzate), si potrebbe diminuire il frame rate affinchè lo spostamento effettuato tra due keyframe non sia troppo limitato¹, riducendo ulteriormente il carico computazionale della CPU.

Oltre ai test online, per valutare l'accuratezza dei risultati al variare di diversi parametri dell'algoritmo, sono state acquisite preventivamente alcune sequenze di immagini, poi salvate in file video.

Inoltre, non utilizzando i diversi livelli di immagini sfruttati invece nel primo metodo, anzichè operare alla risoluzione massima consentita dalle telecamere stereo a disposizione, si è scelto di acquisire immagini a 640×480 pixel, in modo da favorire l'utilizzo in tempo reale del SURF, il quale, come detto, è piuttosto oneroso per quanto riguarda i tempi di calcolo.

Per visualizzare i risultati ottenuti è stata realizzata un'interfaccia con librerie grafiche *Qt*, la quale, oltre a mostrare i vari valori di posizione e di orientazione stimati, disegna il movimento del rover in un piano cartesiano (X, Y), per facilitare la comprensione immediata di eventuali errori di stima dell'algoritmo e anche per dare all'interfaccia un aspetto più gradevole per l'utente.

5.1 Setup di riferimento

Per le prove sperimentali dell'algoritmo sono stati utilizzati due diversi rover, uno utilizzato per le prime prove indoor, l'altro sfruttato soprattutto per i test svolti outdoor.

¹Si ricorda che per avere una migliore accuratezza deve esistere un compromesso tra lunghi step, per non accumulare troppo errore di natura integrativa, e la necessità di conservare un numero di corrispondenze sufficienti tra le due acquisizioni.

Entrambi i robot mobili verranno brevemente descritti in modo da chiarire le loro caratteristiche e verrà anche illustrato il sistema di acquisizione adoperato.

5.1.1 Il sistema di telecamere stereo

Come già accennato, per questo progetto è stata utilizzata la telecamera stereo *Bumblebee2* della *Point Grey Research* (Figura 5.1); in Tabella 5.1 se ne riportano le principali caratteristiche.



Figura 5.1: Il sistema di telecamere stereo *Bumblebee2*

Grazie alle funzioni messe a disposizione dalle librerie *Triclops* e *FlyCapture* (che accompagnano il sistema stereo), l'utente è in grado di controllare le impostazioni della camera, migliorare la qualità dei frame acquisiti e accedere in real-time alle immagini di profondità elaborate dalla tecnologia di stereovisione inclusa.

Specification	Description
Imaging Sensor	Two Sony 1/3" progressive scan CCD ICX204 (1024 × 768 max pixels) 4.65 μm square pixels
Baseline	12cm
Lens Focal Length	3.8mm with 70° HFOV or 6mm with 50° HFOV
Video Data Output	8, 16 and 24-bit digital data
Frame Rates	18, 15, 7.5, 3.75, 1.875 FPS
Interfaces	6-pin IEEE-1394 for camera control and video data transmission 4 general-purpose digital input/output pins.
Voltage Requirements	8-32V
Power Consumption	Less than 3W
Signal To Noise Ratio	Greater than 60dB at 0dB gain
Dimensions	157mm × 36mm × 47.4mm
Mass	342 grams
Lens System	High quality microlenses protected by removable glass system
Accurate Pre-Calibration	For lens distortions and misalignments
Stereo Pair Alignment	Left and right images aligned to within 0.05' pixel RMS error
Calibration Retention	Minimizes loss of calibration due to shock and vibration

Tabella 5.1: Le caratteristiche tecniche della Bumblebee2

La telecamera *Bumblebee2* è in grado di fornire immagini sia ad una risoluzione 640×480 con un framerate di $48 \text{ } fps$, sia ad una risoluzione di 1024×768 con un framerate di $18 \text{ } fps$.

Per quanto riguarda la parte di gestione e di processing delle immagini acquisite dalle telecamere stereo, sono state utilizzate le librerie *OpenCV* realizzate da Intel.

5.1.2 Il rover utilizzato indoor

La piattaforma mobile *robuLAB 80* è un rover *multi-purpose*, realizzato per numerose applicazioni indoor. Il suo utilizzo può anche essere adattato per applicazioni specifiche e l'architettura meccanica permette di operare in modo stabile e affidabile con *payload* fino a 80 Kg .

Il rover presenta una struttura di dimensioni piuttosto contenute ($772 \times 590 \times 475 \text{ mm}$) con un peso di 125 Kg e può raggiungere una velocità massima di 2.6 m/s .

Il sistema di controllo è composto da un pentium M da 1.4 GHz ed è dotato di due uscite RS232, otto uscite USB, sedici I/O digitali e dieci analogici. Il sistema è alimentato da quattro batterie da 48 V .

L'architettura software è stata sviluppata ed è compatibile con Microsoft Robotics Studio. Essa implementa tutte le funzioni base necessarie ai rover autonomi come, per esempio, la gestione e l'astrazione del livello hardware e dei sensori, un sistema anti-collisione e la localizzazione.

Il sistema di acquisizione è stato collocato sulla parte anteriore del robot, per mezzo di una unità pan/tilt, necessaria allo scopo di fissare la *Bumblebee2* in una posizione opportuna per la navigazione del rover (Figura 5.3).



Figura 5.2: Il rover robuLAB 80



Figura 5.3: La Bumblebee2 montata sul rover

5.1.3 Il rover utilizzato outdoor

Il rover utilizzato per i test outdoor è una piattaforma mobile *tricycle-like*, realizzata nel laboratorio G.R.A.A.L.[38].

Il veicolo è costituito esternamente da un telaio di alluminio con un ingombro di $700 \times 849 \text{ mm}$, il quale costituisce l'involturo del rover (Figura 5.4).

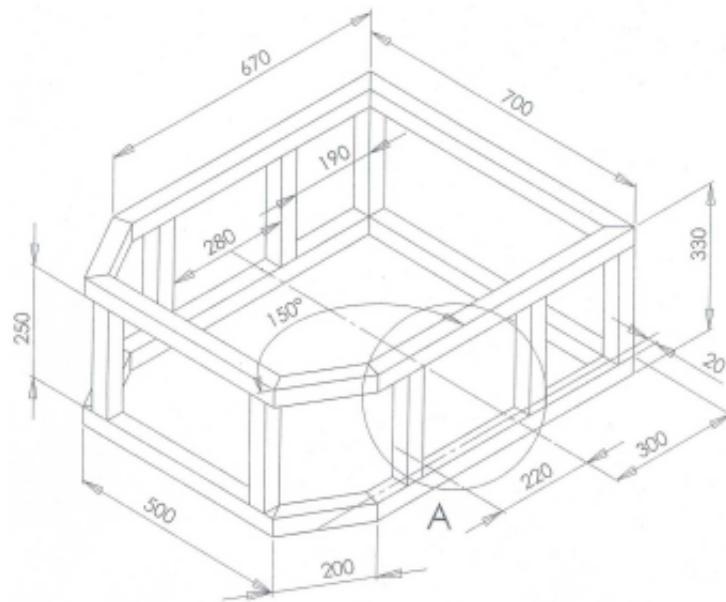


Figura 5.4: Vista prospettica del telaio, misure in mm

Le tre ruote utilizzate per il movimento sono elettromotoruote di diametro 205 mm , pesanti 17 Kg l'una, fornite di motori DC aventi una potenza di 250 W e alimentati a 24 V , grazie ai quali il rover può raggiungere una velocità massima di 4.7 Km/h .

La sensoristica a bordo del veicolo è rappresentata da encoder e potenziometri. Per ottenere la misura della posizione delle ruote sono utilizzati tre encoder digitali a 500 impulsi e 2 canali, tuttavia le letture delle loro uscite vengono effettuate durante la generazione sia dei fronti di salita che su quelli di discesa. In tal

modo, ad ogni giro del motore corrispondono 2000 impulsi anzichè solamente 500, aumentando così la risoluzione della misura e apprezzando variazioni dell'ordine di $3.14 \cdot 10^{-3} \text{ rad}$, con un errore di quantizzazione massimo $\epsilon_q \cong 1.57 \cdot 10^{-3} \text{ rad}$.

Tre potenziometri analogici invece misurano l'angolo di rotazione degli sterzi. Dal loro valore centrale di uscita è possibile apprezzare una variazione massima di $\pm 2.36 \text{ rad}$, a causa della presenza di una zona morta di circa 1.55 rad .

L'alimentazione all'intero veicolo è fornita da quattro batterie a 12 V , sistamate in un apposito vano ricavato nella parte centrale del rover in modo da poter ripartirne il peso e ricaricabili attraverso una presa esterna presente sul lato destro del telaio.

La parte di controllo prevede l'utilizzo di convertitori *Microspeed 60 DC* bidirezionali a quattro quadranti come azionamenti. Lo stadio di potenza a Power Mosfet è pilotato in *PWM* con una frequenza di modulazione di 22 KHz . I possibili modi di pilotaggio sono in velocità o in coppia: il controllo in velocità delle ruote si ottiene attraverso una scheda di reazione da encoder inserita nell'azionamento, mentre per gli sterzi tramite reazione di armatura. Tuttavia, quest'ultima possibilità conduce ad una perdita di precisione ed una riduzione della coppia.

Per quanto riguarda il processore che controlla il veicolo si tratta di un *VIA Eden 400 MHz* a basso consumo, con sistema operativo *Linux/RTAI*, montato su una scheda madre *Advantech PCA-6772F-J1A2 Fanless*, la quale possiede un collegamento *Ethernet 10/100 Base-T*. Il sistema di controllo è inoltre dotato di una scheda di *I/O Servo To Go Model 2*, di un router wireless, per poter effettuare un eventuale controllo remoto, e un hard disk sul quale risiede tutto il software di controllo.

Per realizzare le prove sperimentali è stata montata la telecamera stereo *Bumblebee2* sulla parte anteriore del rover per mezzo di una staffa avvitata al telaio (Figura 5.6).



Figura 5.5: Il rover utilizzato per le prove outdoor

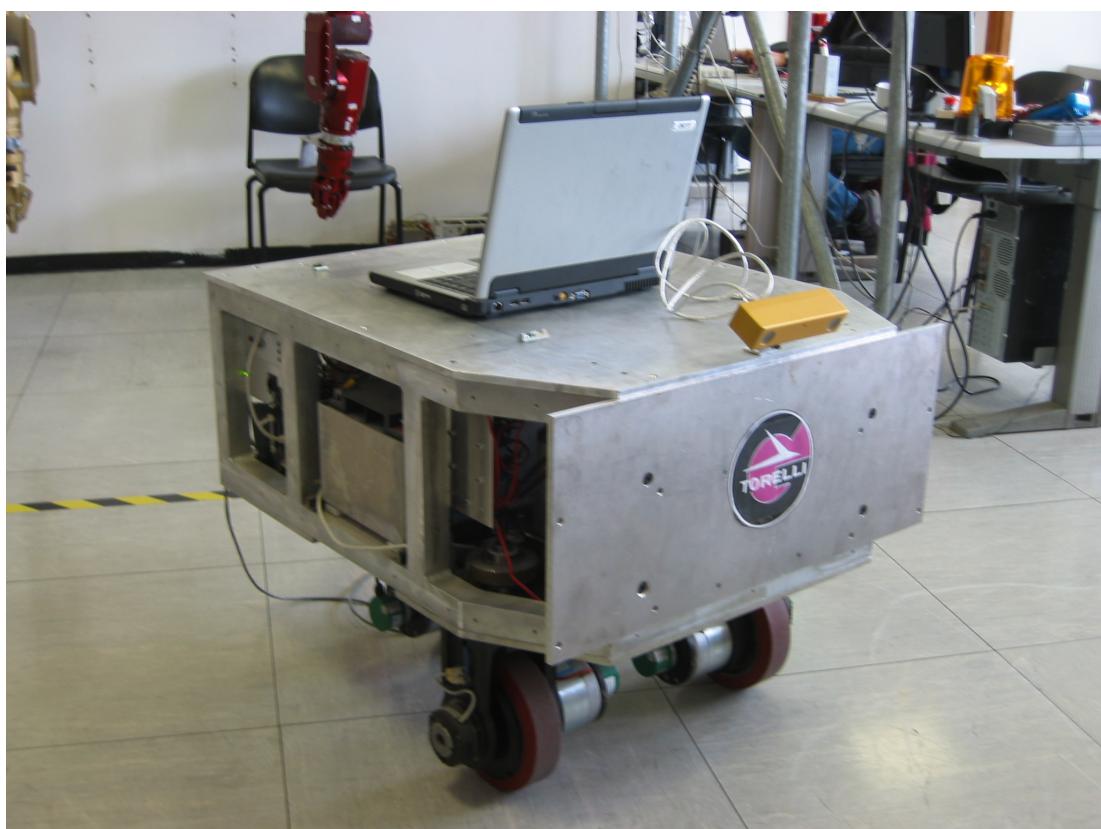


Figura 5.6: Il rover con il sistema di acquisizione stereo

5.2 I test indoor

I primi test effettuati sull’algoritmo sono stati realizzati per necessità in ambiente indoor², in modo da verificare il funzionamento preliminare del sistema e per poter calibrare adeguatamente i suoi parametri.

La Figura 5.7 mostra i risultati di un test svolto all’interno dei laboratori del *SIIT*. Per verificare l’accuratezza della stima, il rover è stato fermato e la sua posizione è stata misurata quando passava vicino al punto iniziale, opportunamente marcato.

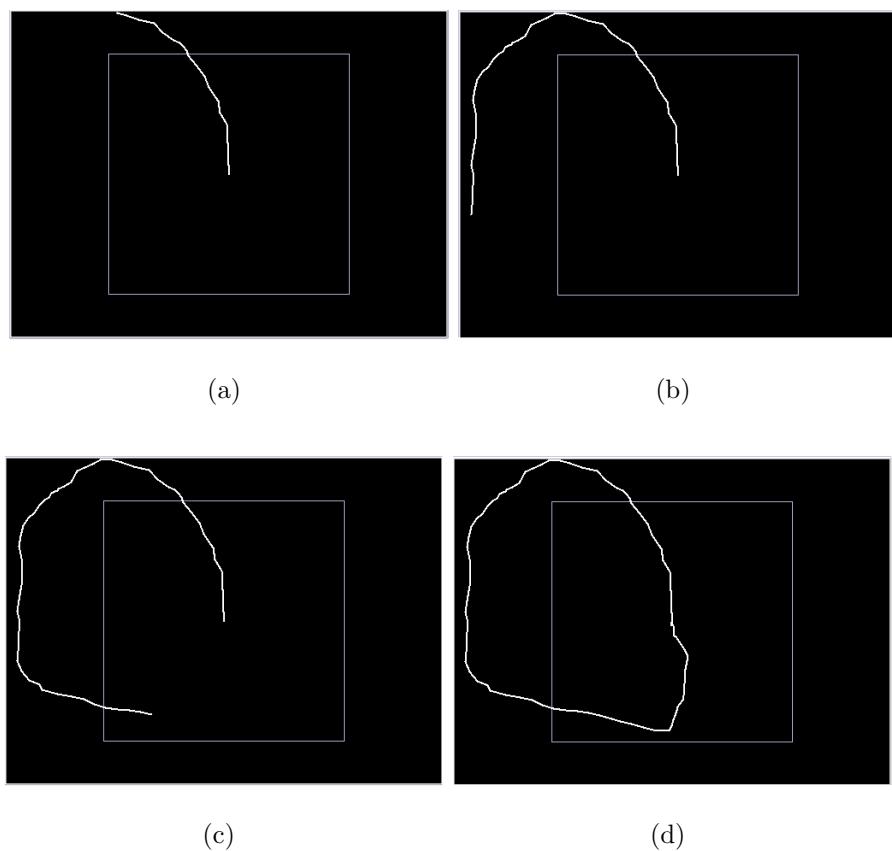


Figura 5.7: Un percorso chiuso stimato dall’algoritmo

²Nel laboratorio *G.R.A.A.L.* e al distretto tecnologico *SIIT*

Anzichè confrontare il risultato con misure di riferimento prese a mano, si è adoperata una tecnica differente, già utilizzata in letteratura. Si è provveduto a far passare il rover, durante la sua navigazione, varie volte sul punto di partenza iniziale e si sono confrontate attraverso l'algoritmo stesso le coppie di frame rilevate in tali occasioni direttamente rispetto alle immagini riferite all'istante di partenza, in modo da ottenere l'esatta differenza di posizione rispetto a quella iniziale (tale valore di riferimento è indicato con R in tabella). Confrontando tale risultato con quello fornito dalla stima dell'algoritmo (indicato con A) su tutto il percorso effettuato fino all'istante di riferimento si può ottenere un'approssimazione dell'errore intrinseco del sistema di odometria visuale. Questa tecnica si basa sul fatto che il processo di stima è molto accurato su frame piuttosto simili fra loro, mentre l'errore accumulato dovrebbe essere più che altro di natura integrativa.

La tabelle di seguito elencano gli errori di posizione (distanza tra la posizione stimata dall'algoritmo e quella misurata) negli istanti di stop, sullo stesso percorso con diverse variazioni dei parametri³.

Come si può notare dai risultati ottenuti (Tabelle 5.4 e 5.5), il filtro di LoG (*Laplacian of Gaussian*) applicato alle immagini in ingresso garantisce una migliore accuratezza, rendendo i frame meno sensibili alle variazioni di luminosità. Anche se le stime sembrano migliorare all'aumentare della soglia dell'estrattore (Tabelle 5.2 e 5.3), le differenze fra i risultati nelle varie prove effettuate sono ridotte, perciò, in generale, si può concludere che il valore di tale parametro non influenza l'accuratezza del sistema in modo sostanziale nei test indoor.

Occorre sottolineare che l'errore più grande commesso dal sistema è spesso quello sull'asse Z . Ciò potrebbe essere dovuto alla presenza delle pareti nelle

³Nelle prove effettuate gli assi di riferimento sono posizionati nel seguente modo, per evitare un problema di singolarità di notazione: asse X parallelo al terreno e perpendicolare all'asse ottico, asse Y parallelo all'asse ottico, asse Z perpendicolare al terreno.

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]		[m]	[gradi]	[gradi]	[m]
664	R	0.001 -0.001 -0.000		(1.10%)	-0.06 0.01 -1.17		
	A	-0.066 0.109 -0.039		0.134	-0.75 -1.05 1.33	2.80	12.24
1056	R	0.000 -0.006 -0.001		(0.91%)	-0.05 0.04 -0.28		
	A	-0.074 -0.111 -0.198		0.235	-1.82 0.10 3.22	3.92	25.64
1455	R	0.000 -0.007 -0.001		(1.12%)	-0.03 0.03 0.52		
	A	-0.109 -0.202 -0.369		0.430	-2.80 0.07 5.37	5.58	38.21

Tabella 5.2: Gli errori commessi dall'algoritmo con LOG e Hessian threshold 350

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]		[m]	[gradi]	[gradi]	[m]
664	R	0.015 0.000 0.001		(2.88%)	-0.01 0.00 -0.93		
	A	0.116 -0.293 0.169		0.352	0.75 -6.04 1.61	6.59	12.24
1056	R	0.000 -0.007 -0.003		(2.99%)	-0.02 0.00 -0.93		
	A	0.191 -0.577 0.473		0.766	4.29 -9.85 6.30	12.95	25.64
1455	R	0.002 -0.008 0.002		(2.80%)	0.00 -0.02 0.57		
	A	0.287 -0.782 0.687		1.072	6.06 -14.08 10.82	18.42	38.21

Tabella 5.3: Gli errori commessi dall'algoritmo senza LOG e Hessian threshold 350

frame	posizione		$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]	[m]	[gradi]	[gradi]	[m]
664	R	0.001 -0.001 -0.000	(1.19%)	-0.06 0.01 -1.17		
	A	-0.020 -0.012 -0.144	0.145	-0.18 -0.73 1.06	2.35	12.24
1056	R	0.000 -0.006 -0.001	(1.10%)	-0.05 0.04 -0.28		
	A	-0.077 -0.051 -0.271	0.284	-0.70 -0.44 2.81	3.19	25.64
1455	R	0.000 -0.007 -0.001	(1.22%)	-0.03 0.03 0.52		
	A	-0.139 -0.107 -0.437	0.468	-2.04 0.00 5.19	5.08	38.21

Tabella 5.4: Gli errori commessi dall'algoritmo con LOG e Hessian threshold 500

frame	posizione		$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]	[m]	[gradi]	[gradi]	[m]
664	R	0.001 -0.001 -0.000	(0.95%)	-0.06 0.01 -1.17		
	A	-0.008 -0.061 -0.100	0.116	-0.35 -0.96 1.92	3.25	12.24
1056	R	0.000 -0.006 -0.001	(0.94%)	-0.05 0.04 -0.28		
	A	-0.009 -0.128 -0.208	0.241	-0.96 -0.79 3.77	4.23	25.64
1455	R	0.000 -0.007 -0.001	(0.98%)	-0.03 0.03 0.52		
	A	-0.020 -0.154 -0.347	0.376	-2.55 -1.39 4.92	5.26	38.21

Tabella 5.5: Gli errori commessi dall'algoritmo con LOG e Hessian threshold 700

stanze, le quali potrebbero provocare un apparente movimento delle feature in questa direzione.

5.3 I test outdoor

Dopo aver riscontrato preliminarmente che, in ambiente indoor, il sistema operasse in modo sufficientemente adeguato, si è deciso di testare il sistema outdoor, per testarne il comportamento in una situazione più simile a quella di una superficie da esplorare. Quindi, si sono effettuate varie prove all'esterno dell'università, su un terreno che, nonostante sia asfaltato, presenta molti buchi e asperità, risultando sufficientemente disconnesso per testare l'algoritmo su una superficie che in qualche modo somigliasse ad un terreno di tipo spaziale.

Come nelle prove indoor, non avendo a disposizione un sistema che fornisse l'esatta posizione del rover, si sono confrontate le stime ottenute con le misure rilevate in posizioni fissate, in cui il rover è stato fermato (tali posizioni sono state riconosciute attraverso segni disegnati sul terreno). Purtroppo il metodo migliore per misurare con accuratezza l'errore commesso sarebbe stato quello di confrontare le misure ottenute con una tecnica di localizzazione GPS, che fornisse l'effettiva posizione globale del rover da confrontare con le stime ottenute dall'algoritmo. Tuttavia, in mancanza di un setup del genere, si è approntato appunto questo tipo di confronto, il quale almeno fornisce una misura qualitativa della bontà del sistema.

Sono state effettuate diverse tipologie di prove: il rover è stato dapprima guidato solo con un movimento puramente traslazionale, per poi fargli eseguire un percorso chiuso piuttosto lungo; infine si è compiuto un tragitto molto più lungo dei primi due, in cui però la piattaforma mobile ad ogni giro veniva fatta passare sempre dallo stesso punto di riferimento iniziale. In tal modo è stato possibile confrontare i risultati su un percorso abbastanza ampio, con la possibilità di

avere dei riferimenti intermedi in cui calcolare l'errore tra la posizione misurata e la stima ottenuta dall'algoritmo.

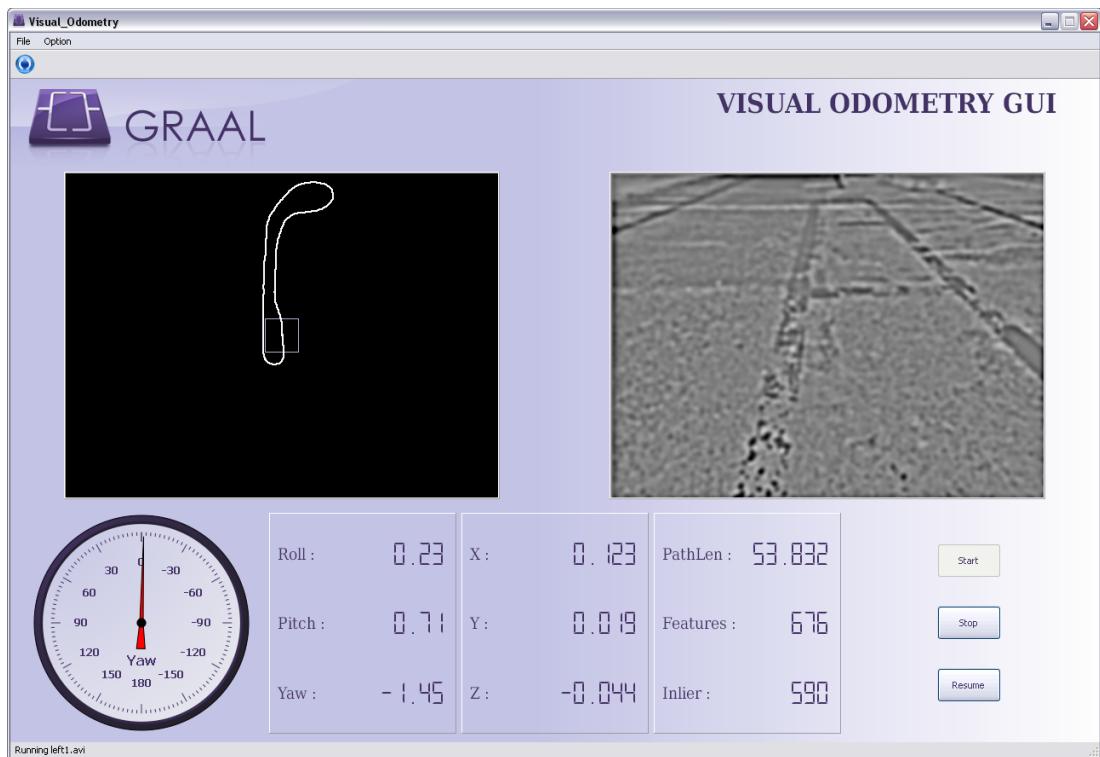


Figura 5.8: Percorso chiuso del rover outdoor

Analizzando i risultati ottenuti (riportati nelle Tabelle dalla 5.6 alla 5.11), si può innanzitutto notare che l'errore percentuale di stima rimane ben al di sotto dell'1%, dimostrando una migliore accuratezza del sistema di odometria visuale rispetto ai risultati ottenuti in ambiente indoor. L'incremento dell'errore assoluto all'aumentare della lunghezza percorsa è dovuto alla natura integrativa di questo sistema di localizzazione, anche se si può tuttavia considerare piuttosto contenuto.

Dai risultati precedenti inoltre si possono trarre un altro paio di conclusioni. La stima sembra essere leggermente migliore quando le immagini, prima di essere sottoposte all'algoritmo, vengono prefiltrate con LoG (*Laplacian of Gaus-*

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
	[m]			[m]	[gradi]	[gradi]	[m]
1410	R	0.013	-0.018	-0.010	(0.51%)	1.32 -0.17 -2.55	
	A	0.172	0.196	0.074	0.279	1.37 0.87 -2.45	1.04 53.76
2850	R	0.031	-0.013	-0.008	(0.31%)	1.52 -0.09 -3.15	
	A	0.369	0.049	-0.016	0.343	0.12 0.98 -0.09	3.53 108.52
4240	R	0.031	-0.011	-0.007	(0.32%)	0.24 -0.20 -0.85	
	A	0.528	0.077	0.018	0.505	-2.36 0.92 3.24	4.97 158.60

Tabella 5.6: Gli errori commessi dall'algoritmo con LOG e Hessian threshold 350

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
	[m]			[m]	[gradi]	[gradi]	[m]
1410	R	0.013	-0.018	-0.010	(0.48%)	1.32 -0.17 -2.55	
	A	0.112	-0.108	0.212	0.259	0.27 -0.70 -1.55	1.54 53.76
2850	R	0.031	-0.013	-0.008	(0.49%)	1.52 -0.09 -3.15	
	A	0.384	-0.153	0.367	0.533	-1.22 -1.18 -0.75	3.80 108.52
4240	R	0.032	-0.011	-0.007	(0.46%)	0.26 -0.20 -0.84	
	A	0.595	-0.152	0.449	0.738	-3.45 -1.83 3.32	5.80 158.60

Tabella 5.7: Gli errori commessi dall'algoritmo senza LOG e Hessian threshold 350

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]		[m]	[gradi]	[gradi]	[m]
1410	R	0.013 -0.018 -0.010		(0.22%)	1.32 -0.17 -2.55		
	A	0.123 0.019 -0.044		0.120	0.23 0.71 -1.45	1.78	53.76
2850	R	0.031 -0.013 -0.008		(0.28%)	1.52 -0.09 -3.15		
	A	0.331 0.067 -0.041		0.312	-0.02 1.21 -0.19	3.58	108.52
4240	R	0.031 -0.011 -0.007		(0.30%)	0.24 -0.20 -0.85		
	A	0.506 0.088 -0.002		0.485	-2.46 1.17 3.36	5.18	158.60

Tabella 5.8: Gli errori commessi dall'algoritmo con LOG e Hessian threshold 400

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]		[m]	[gradi]	[gradi]	[m]
1410	R	0.013 -0.018 -0.010		(0.47%)	1.32 -0.17 -2.55		
	A	0.146 -0.098 0.192		0.254	0.19 -0.63 -1.47	1.62	53.76
2850	R	0.031 -0.013 -0.008		(0.52%)	1.52 -0.09 -3.15		
	A	0.446 -0.146 0.352		0.565	-1.33 -1.16 -0.48	4.04	108.52
4240	R	0.032 -0.011 -0.007		(0.55%)	0.26 -0.20 -0.84		
	A	0.686 -0.187 0.561		0.883	-3.79 -2.37 3.95	6.63	158.6

Tabella 5.9: Gli errori commessi dall'algoritmo senza LOG e Hessian threshold 400

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]		[m]	[gradi]	[gradi]	[m]
1410	R	0.013 -0.018 -0.010		(0.17%)	1.32 -0.17 -2.55		
	A	0.104 0.001 -0.009		0.092	0.24 0.40 -1.46	1.63	53.76
2850	R	0.031 -0.013 -0.008		(0.22%)	1.52 -0.09 -3.15		
	A	0.270 0.051 -0.020		0.247	-0.40 1.17 -0.58	3.44	108.52
4240	R	0.031 -0.011 -0.007		(0.23%)	0.24 -0.20 -0.85		
	A	0.397 0.078 -0.003		0.376	-2.59 1.35 2.81	4.87	158.60

Tabella 5.10: Gli errori commessi dall’algoritmo con LOG e Hessian threshold 500

frame	posizione			$\ \Delta pos\ $	angolo	$\ \Delta ang\ $	path
		[m]		[m]	[gradi]	[gradi]	[m]
1410	R	0.013 -0.018 -0.010		(0.21%)	1.32 -0.17 -2.55		
	A	0.067 -0.057 0.087		0.117	0.16 -0.10 -1.93	1.31	53.76
2850	R	0.031 -0.013 -0.008		(0.13%)	1.52 -0.09 -3.15		
	A	0.130 -0.039 0.099		0.148	-0.23 0.75 -1.51	2.35	108.52
4240	R	0.032 -0.011 -0.007		(0.14%)	0.26 -0.20 -0.84		
	A	0.212 -0.053 0.127		0.228	-1.93 1.03 1.55	3.46	158.6

Tabella 5.11: Gli errori commessi dall’algoritmo con LOG e Hessian threshold 700

sian) (Tabelle dalla 5.6 alla 5.9), il quale rende le acquisizioni più robuste alle variazioni d'illuminazione, anche se, tuttavia, a differenza delle prove indoor, i risultati rimangono comunque comparabili. Infine, dalle Tabelle 5.10 e 5.11 si evince come l'accuratezza della stima migliori in maniera significativa all'aumentare della soglia Hessiana, con cui vengono scelti i punti d'interesse dall'estrattore nell'immagine: ciò è in parte dovuto al fatto che con soglie molto basse (i.e. 350) l'estrattore individua un numero considerevole di feature, causando probabilmente qualche difficoltà ulteriore nell'associazione dei punti nelle fasi di matching e di tracking.

In conclusione, il sistema sembra essere abbastanza accurato (errore minore dell'1% della distanza percorsa) e piuttosto robusto (le prove sono tutte andate a buon fine), anche su un terreno outdoor piuttosto sconnesso.

Capitolo 6

Conclusioni e sviluppi futuri

In questa tesi sono stati presentati due diversi algoritmi di odometria visuale. Il primo sistema realizzato, partendo da quello utilizzato nella missione MER[36], non ha prodotto risultati soddisfacenti per i requisiti inizialmente fissati. In particolare, rendendo l'algoritmo indipendente da una stima iniziale (uno degli obiettivi del progetto), i calcoli dovevano essere eseguiti tra frame molto ravvicinati fra loro e, quindi, causavano un sovraccarico eccessivo della CPU (caratteristica non desiderata per un rover planetario). D'altro canto, utilizzando l'informazione sul movimento fornita da una sorgente esterna (i.e. odometria meccanica), nelle prove preliminari si è riscontrato che un errore proveniente da tale fonte causava il conseguente fallimento dell'odometria visuale. Inoltre, talvolta, i tempi di calcolo si allungavano eccessivamente, a causa della imprevedibilità della covarianza associata alle stime nei primi livelli della piramide, non consentendo così un'utilizzo in tempo reale.

Il secondo algoritmo, invece, basato sul lavoro di Nister[8] (che prevede l'estrazione di feature su tutte le immagini acquisite e realizza le fasi di matching e di tracking associando tra loro solo i punti estratti), si è rivelato sufficientemente accurato, robusto, e adeguato alle specifiche del progetto. La sua realizzazione, anzichè sull'Harris corner detector, ha sfruttato l'estrattore di feature SURF, il

quale garantisce che le fasi di matching e tracking siano molto affidabili, grazie ai descrittori associati ai punti rilevati, e permette di confrontare frame acquisiti anche dopo spostamenti relativamente ampi, per merito della sua invarianza alle trasformazioni¹ tra le immagini (a differenza di Nister, che necessita di frame molto ravvicinati nel tempo e comporta quindi un conseguente sovraccarico della CPU).

I risultati ottenuti dalle prove effettuate hanno evidenziato errori di stima minori dell'1% rispetto alla lunghezza totale del tragitto compiuto, con distanze percorse fino a 180 m. Inoltre, il sistema ha dato dimostrazione di operare correttamente in modo continuativo nel tempo, con una limitazione più che accettabile sulla velocità consentita al rover (0.20 m/s); poiché nelle applicazioni spaziali le piattaforme robotiche si muovono a velocità nettamente inferiori, il frame rate di acquisizione delle immagini potrà essere ridotto ulteriormente, diminuendo così la percentuale di utilizzo della CPU da parte dell'algoritmo.

Tuttavia, per poter essere definitivamente utilizzato come sistema di volo, considerando il fatto che l'hardware di bordo dei rover spaziali è considerevolmente inferiore dal punto di vista delle prestazioni rispetto a quello adoperato in questo progetto, l'algoritmo necessita di ulteriori ottimizzazioni. In maniera particolare, gli sforzi si dovranno concentrare sulla fase di estrazione, che risulta il collo di bottiglia, impiegando circa la metà del tempo necessario all'esecuzione dell'intero algoritmo.

Una strada che in futuro potrebbe essere percorsa è quella di cercare di integrare il sistema realizzato sulle ultime FPGA dichiarate *space-qualified*, che potrebbero consentire all'algoritmo di odometria visuale di funzionare anche sull'hardware di volo con buoni risultati e, allo stesso tempo, di lasciare la CPU del rover a disposizione degli altri importanti task da svolgere nella missione.

Inoltre, in accordo con *Thales Alenia Space*, è sorto l'interesse per lo sviluppo

¹Rotazioni e cambiamenti di scala.

di un modulo di *Data Fusion*, atto all'integrazione dei risultati forniti dal sistema realizzato con le informazioni provenienti dagli altri sensori (IMU, odometria meccanica) presenti a bordo del rover utilizzato dall'azienda come *test-bench*, in modo da ottenere un'accuratezza migliore nel processo di localizzazione.

Bibliografia

- [1] Moravec, H. (1980). *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford University, Stanford, CA.
- [2] Matthies, L. e Shafer, S. (1987). Error modelling in stereo navigation. *IEEE Journal of Robotics and Automation*, RA-3(3).
- [3] Matthies, L. (1989). *Dynamic Stereo Vision*. PhD thesis, Carnegie Mellon University Computer Science Department. CMU-CS-89-195.
- [4] Zhang, Z., Faugeras, O., e Ayache, N. (1988). Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints. In *International Conference on Computer Vision*, pp. 177-186, Tampa, Florida. Computer Society of the IEEE, IEEE Computer Society Press.
- [5] Lacroix, S., Mallet, A., Chatila, R., e Gallo, L. (1999). Rover self localization in planetary-like environments. In *International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, pp. 433-440, Noordwijk, Netherlands.
- [6] Hirschmüller, H., Innocent, P., e Garibaldi, J. (2002). Real-Time Correlation-Based Stereo Vision with Reduced Border Errors. In *International Journal of Computer Vision*, Volume 47 (1/2/3), April-June 2002, pp. 229-246.

- [7] Nister, D., Naroditsky, O., e Bergen, J. (2004). Visual Odometry. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652-659. IEEE Computer Society Press.
- [8] Nister, D., Naroditsky, O., e Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3-20.
- [9] C. McCarthy e N. Barnes. (2004). Performance of optical flow techniques for indoor navigation with a mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, (ICRA2004), pp. 5093-5098, New Orleans, USA, 2004.
- [10] Vassallo, R. F., Santos-Victor, J., e Schneebeli, J. (2002). A general approach for egomotion estimation with omnidirectional images. In *OMNIVIS*, pp 97-103.
- [11] Gluckman, J. e Nayar, S. K. (1998). Ego-motion and omnidirectional cameras. In *ICCV*, pp. 999-1005.
- [12] Corke, P. I., Strelow, D., e Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. In *Proceedings of IROS 2004*.
- [13] Zereik, E. (2010). *Space Robotics Supporting Exploration Missions: Vision, Force Control and Coordination Strategies*. PhD Thesis, Course in Electronic and Computer Engineering, Robotics and Telecommunications, University of Genoa, 2010.
- [14] Konolige, K., Agrawal, M. (2007). Rough terrain visual odometry. In *Proc. International Conference on Advanced Robotics (ICAR)*, Agosto 2007.
- [15] Beauchemin, S. S., Barron, J. L. (1995). The computation of optical flow. In *ACM Computing Surveys*, 27(3), 1995.

- [16] Comport, A., Malis, E., Rives, P. (2007). Accurate quadrifocal tracking for robust 3d visual odometry. In *ICRA*, 2007.
- [17] Farin, D. (2005). *Automatic Video Segmentation Employing Object/Camera Modeling Techniques*. PhD thesis, Eindhoven University. 90-386-2381-X.
- [18] Moravec, H. (1979). Visual mapping by a robot rover. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pp. 599-601, Agosto 1979.
- [19] Shi, J., Tomasi, C. (1994). Good feature to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593-600, 1994.
- [20] Harris, C., Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference, Manchester*, pp. 147-151, 1988.
- [21] Smith, S. M., Brady, J. M. (1997). SUSAN - a new approach to low level image processing. In *International Journal of Computer Vision (IJCV)*, 23(1):45-78, Maggio 1997.
- [22] Trajkovi, M., Hedley, M. (1998). Fast corner detection. In *Image and Vision Computing*, Elsevier, 1998.
- [23] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints, cascade filtering approach. In *IJCV*, 60(2):91-110, Gennaio, 2004.
- [24] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L. (2008). Speeded-Up Robust Feature (SURF). In *ECCV*, 2006.
- [25] Lindeberg, T. (1990). Scale-space for discrete signals. In *PAMI*, 12(3):234-254, 1990.

- [26] Bay, H. (2006) *From Wide-baseline Point and Line Correspondances to 3D*. PhD thesis, ETH Zurigo, 2006.
- [27] Lindeberg, T., Bretzner, L. (2003). Real-time scale selection in hybrid multi-scale representations. In *Scale-Space*, pp. 148-163, 2003.
- [28] Nishihara, H. K. (1984). PRISM, a Practical Real-Time Imaging Stereo Matcher. Technical Report A.I. Memo 780, MIT, Cambridge, MA, 1984.
- [29] Gennery, D. B. (1980). *Modelling the environment of an exploring vehicle by means of stereo vision*. PhD thesis, Stanford University, Stanford, CA. Giugno 1980.
- [30] Anandan, P., Weiss, R. (1985). Introducing a smoothness constraint in a matching approach for the computation of displacement fields. In *Proc. ARPA IUS Workshop, SAIC*, pp. 186-197, Dicembre 1985.
- [31] Lucas, B. D., Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proc. of Imaging Understanding Workshop*, pp. 121-130, 1981.
- [32] Tomasi, C., Kanade, T. (1991). Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Aprile 1991.
- [33] Schonemann, P. H. (1966). A generalized solution of the orthogonal procrustes problem. In *Psychometrika*, 31(1):1-10, Marzo 1966.
- [34] Schonemann, P. H., Carroll, R. M. (1970). Fitting one matrix to another under choice of a central dilation and a rigid motion. In *Psychometrika*, 35(2):245-255, Giugno 1970.
- [35] Graybill, F. A. (1983). *Matrices with Applications in Statistics*. Wadsworth International Group, Belmont, CA, 1983.

- [36] Maimone, M., Cheng, Y., Matthies, L. (2007). Two years of Visual Odometry on the Mars Exploration Rovers. In *Journal of Field Robotics*, 24(3):169-186, Marzo 2007.
- [37] Johnson, A. E., Goldberg, S. B., Cheng, Y., Matthies, L. (2008). Robust and Efficient Stereo Feature Tracking for Visual Odometry. In *IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, Maggio 2008.
- [38] Simetti, E., Vecchio, M. (2005). *Realizzazione del sistema di controllo per la movimentazione di un manipolatore mobile anolomo*. Thesis, Genoa Robotic And Automation Laboratory, Novembre 2005.
- [39] Alismail, H. (2009). *Exploring Visual Odometry for Mobile Robots*. Thesis, Carnegie Mellon, Qatar, Maggio 2009.