Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingeneria Informatica

tesi di laurea

Giroscopio visivo per telecamere omnidirezionali

Relatore: Emanuele Menegatti
Correlatore: Alberto Pretto

Laureando: Nicola Carlon

13 Dicembre 2011

A mio padre.

Se l'uomo saprà utilizzarla con spirito creativo,

la macchina sarà il servo e il liberatore dell'umanità.

Frank Lloyd Wright

Il vero mistero del mondo è il visibile, non l'invisibile.

Oscar Wilde

Dubbio, uno dei nomi dell'intelligenza.

Jorge Luis Borges

Prefazione

Fin dalle sue origini, l'uomo ha sempre cercato l'imitazione della Natura. Alcuni uomini hanno percorso questa ricerca come sfida verso di Essa, altri come un compagno di banco che cerca di copiare dal vicino più bravo. Alcuni l'hanno percorsa con lo scopo di compiacerLa, molti per contrastarLa ed altri per risolvere i problemi da Lei stessa posti.
Dobbiamo a questa ricerca le invenzioni che hanno cambiato la storia, anche se non subito riconducibili ad essa, esattamente come una TAC e il Web non evocano lo studio sulle particelle elementari, che invece le accomuna.

L'arte, con la rappresentazione della Natura nelle sue forme classiche di pittura, scultura e letteratura, è stata primogenita di questa aspirazione e sua regina incontrastata per millenni.
All'ispirazione artistica del teatro degli automi ellenico del -300, e ancor prima delle statuette snodate egizie del -2000 (tomba di Kelmis), possiamo far risalire la nascita dell'idea di automazione. Già presente nelle opere di Erone di Alessandria e poi di Archimede, sviluppò per la prima volta tutto il suo potenziale con Leonardo Da Vinci, che si spinse ad ideare il primo

automa raffigurante un cavaliere (1495-7).
Dopo Leonardo, oltre all'invenzione del motore a vapore, furono probabilmente le opere di Jacques de Vacounson e poi le bambole di Pierre Jaquet-Droz con l'influenza dei Bernoulli a porre le basi per la meccanica del lavoro industriale.
Negli ultimi due secoli abbiamo assistito ad una esponenziale specializzazione dell'automazione al fine di sostituire l'uomo nel processo industriale. In questo modo sono nate le numerose discipline ingegneristiche che oggi conosciamo.
Dagli anni '50, grazie all'avvento dei calcolatori, le ricerche sulla robotica subiscono una forte accelerazione.
Oggi le due cose che più si avvicinano all'imitazione della natura sono i *coevolved predator and prey robots* di Dario Floreano e gli *androidi* di Hiroshi Ishiguro.

Parallelamente all'automazione, anche la botanica prima e la biologia poi si sono interessate all'imitazione della Natura. Rimpicciolendo i loro strumenti fino a dar vita alla biologia molecolare e alla genetica si sono evolute, grazie anche al genio di Craig Wenter, al punto da farci supporre che una nuova disciplina possa proseguire questa ricerca da un punto di vista profondamente diverso rispetto all'automazione.

Noi siamo qui per sederci sulle spalle di questi giganti e fare un ulteriore passo avanti.

L'autore.

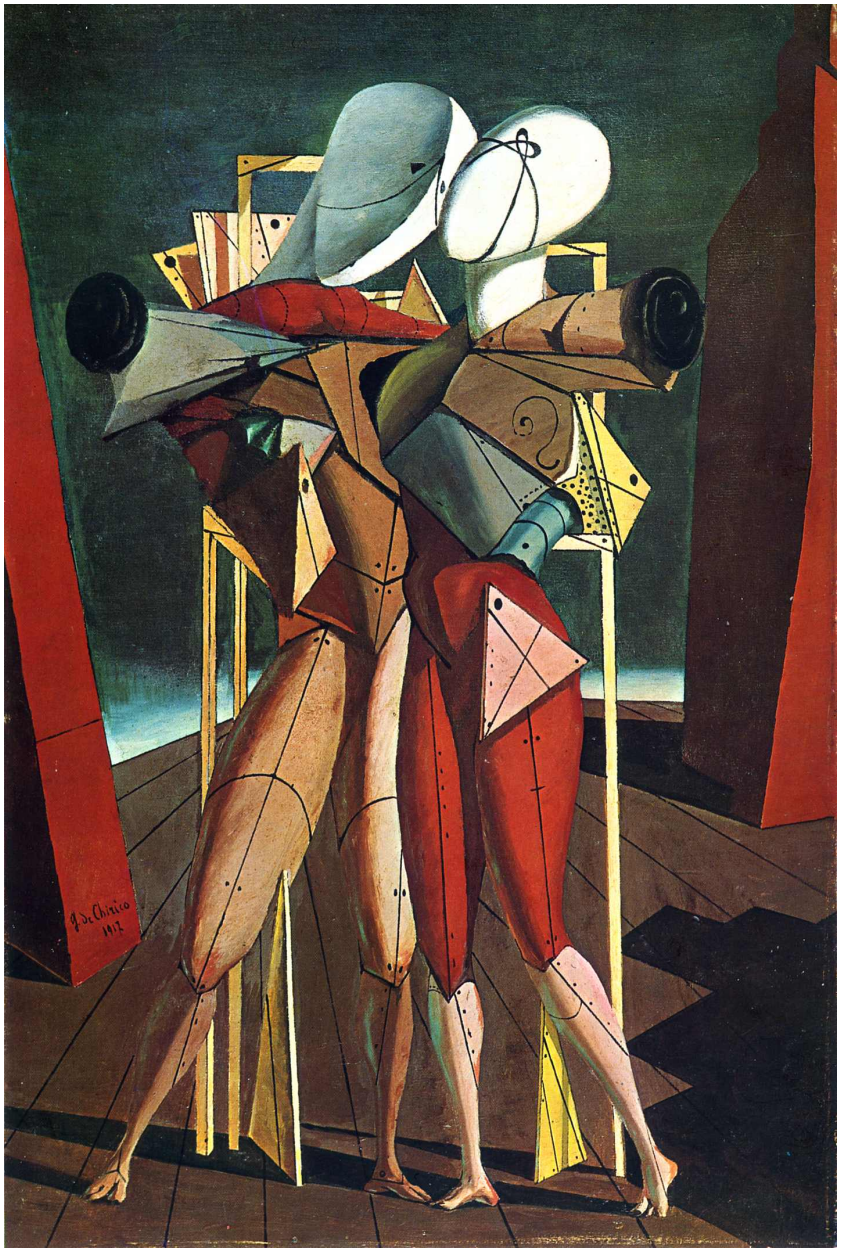


Figura 1. Ettore e Andromaca - De Chirico

Sommario

Questo lavoro si propone di fornire un supporto visivo ai sensori inerziali nella stima dell'assetto per la stabilizzazione dei robot umanoidi.
Prima di tutto si è effettuato un reverse engineering sul robot per predisporlo all'acquisizione di dati sull'assetto tramite protocollo USB.
Successivamente si è scelta una telecamera omnidirezionale posta sulla testa del robot per permettere un'analisi in ogni direzione.
Questa stima con una telecamera omnidirezionale è spesso vincolata al tipo di ambiente in cui si opera; si cerca quindi di eliminare la dipendenza dall'ambiente estraendo solo la direzione del movimento come vettore nello spazio 3D. Purtroppo la necessità di operare in real-time ha escluso soluzioni di tipo "structure from motion" con geometria epipolare impedendo una misura quantitativa dello spostamento.
Per stimare la direzione del movimento si esegue il tracking di corners con il metodo di Lucas-Kanade. Dopo la calibrazione della telecamera omnidirezionale per eseguire i calcoli sulla sfera unitaria, si considerano i Great Circles (GCs) passanti per gli estremi dei tracks. Si cerca quindi il *focus of expansion* dell'optical flow come fosse un punto di fuga delle rette nel mondo reale prodotte dal movimento.
Questo Motion Vanishing Point (MVP) corrisponde quindi al punto sulla

sfera in cui si concentrano il maggior numero di intersezioni tra i GCs. In caso di rototraslazione si ottengono due MVPs separati da un angolo dovuto alla rotazione.

Lo spostamento nel tempo del MVP porta l'informazione dello spostamento del robot rispetto alla posizione precedente e può essere usato per rendere più accurata la stima dell'assetto prodotta da dispositivi inerziali.

Indice

Capitolo 1

Introduzione

1.1 Robotica

La robotica negli ultimi 50 anni ha visto uno sviluppo notevole che l'ha portata a differenziarsi nello studio di cinque principali settori:

- Militare: missili teleguidati, droni, esoscheletri, tactical robots
- Industriale: automotive, manipolatori, cartesiani, CNC, imballaggio
- Medico: telechirurgia, simulatori, assistenza, riabilitazione
- Domestico: domotica, pulitori, sorveglianza
- Entrateinment: animatronics, giocattoli

L'ordine in cui sono stati citati questi settori non è casuale, in quanto bisogna sempre considerare che il settore trainante è quello gli armamenti. La robotica, inizialmente considerata una tecnica empirica, è da poco definibile come scienza ed è stata capace di unire discipline sia di natura umanistica, come linguistica e psicologia, sia di natura scientifica, come biologia, fisiologia, automazione, elettronica, fisica, informatica, matematica

e meccanica.

1.2 Biomimica

Al di là dell'aspetto meccanico, che ha visto un miglioramento con il progresso della tecnologia, è interessante concentrarsi sul paradigma della **biomimica**.

La Natura è la macchina ottimizzatrice per eccellenza, in quanto ha avuto milioni di anni per risolvere problemi NP.

La biomimica consiste nello sfruttare le soluzioni trovate dalla Natura e adattarle per risolvere problemi simili.

Questo paradigma può essere utilizzato ad alto livello sull'aspetto **hardware/ morfologico** dei robot per trovare il modo migliore di costruirli al fine di farli agire nell'ambiente voluto.

Se in un ambiente acquatico gli organismi non dotati di pinne e vincolati a muoversi nel 2D del fondale hanno una struttura a simmetria radiale, significa che un robot che deve muoversi su un terreno accidentato potrà farlo al meglio se olonomo (figura 1.1).

(a) Stella marina (b) ATHELETE (NASA)

Figura 1.1. Simmetria radiale

Se in un ambiente estremamente esteso come le steppe orientali dell'Asia il cavallo ha adattato il suo apparato locomotore per permettergli di percorrere lunghe distanze, un robot da cui si desiderano gli stessi risultati potrà farlo al meglio se dotato di arti simili (figura 1.2).

(a) Cavallo (b) AlphaDog (DARPA)

Figura 1.2. Locomozione

La biomimica si può anche applicare alla morfologia a basso livello, simulando in ambiente virtuale l'evoluzione della sua struttura fisica per essere in grado di risolvere un task al meglio.

Anche all'aspetto **sofware/ comportamentale** si può applicare la biomimica in due livelli. L'alto livello, ovvero l'insieme dei comportamenti che il robot attiva a seconda del task e degli stimoli esterni, può essere strutturato con un sistema *beaviour based* imitando comportamenti studiati in natura dall'etologia o dalle scienze cognitive.

A basso livello invece si può strutturare il "cervello" dei robot come una rete neurale unita agli algoritmi genetici in modo da ottenere un autoapprendimento, per ora solo di semplici task come fanno i robot di Floreano.

Anche i sensi del robot sono oggetto di biomimica in quanto giocano un ruolo cruciale per l'adattamento all'ambiente.

Se per esempio in natura i pipistrelli usano gli ultrasuoni per muoversi e cacciare in totale assenza di luce, un robot che deve operare in situazioni di visibilità critiche potrà evitare gli ostacoli al meglio se dotato di sensori agli ultrasuoni 1.3.

(a) Pipistrello

(b) Alpha Search and Rescue (DARPA)

Figura 1.3. Navigazione con ultrasuoni

Il senso più utilizzato e studiato è però la vista. L'acquisizione di informazioni su oggetti distanti viene ottimizzata in Natura dalla percezione della luce riflessa.

Essa, quando presente, è sempre preferibile agli ultrasuoni, per alcune loro limitazioni strutturali. Gli ultrasuoni viaggiano infatti a soli 360Km/s contro i 300000Km/s della luce e oltretutto necessitano di un'emissione da parte del soggetto, dimezzando ulteriormente il framerate di acquisizione. Per la luce, invece, il collo di bottiglia è il solo tempo di risposta delle cellule fotorecettrici.

Inoltre gli ultrasuoni sono più soggetti ad interferenze o a giochi di riflessi che possono ingannare il soggetto molto più frequentemente della luce che,

per ingannare allo stesso modo, necessiterebbe di un complicato labirinto di specchi. Per non parlare della possibilità, tutt'altro che remota di non ricevere gli ultrasuoni inviati, a causa della forma di un eventuale ostacolo anche molto comune, come mostrato in figura 1.4.

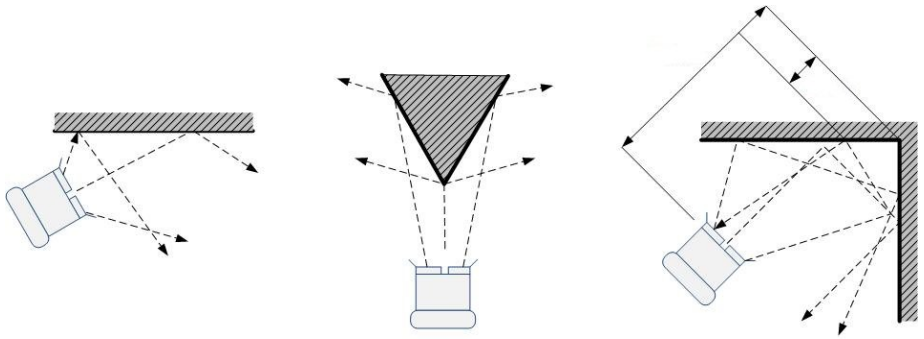


Figura 1.4. Possibili errori nella ricezione degli ultrasuoni

Dotare di una telecamera un robot ci risulta abbastanza facile dato il grandissimo sviluppo della fotografia in questo ultimo secolo.
Per quanto riguarda il framerate, i sensori odierni possono arrivare senza grossi problemi alle velocità di quelli visti in natura, spostando il collo di bottiglia dalle cellule fotorecettrici (elementi del CCD) all'elaborazione del flusso video.
In effetti la questione cruciale ed estremamente complessa nell'uso di una telecamera da parte del robot è proprio l'elaborazione, sia dal punto di vista dell'enorme carico computazionale, sia dal punto di vista algoritmico.

1.3 Computer vision

Lo scopo principale della visione artificiale è quello di riprodurre la vista umana. Vedere è inteso non solo come l'acquisizione di una fotografia

bidimensionale di un'area, ma soprattutto come l'interpretazione del contenuto di quell'area. L'informazione è intesa in questo caso come qualcosa che implica una decisione automatica.
Seguendo il percorso dell'informazione nei sistemi biologici dotati di vista si può meglio comprendere lo sforzo richiesto dalla computer vision.

Visione naturale	**Visione artificiale**
Luce visibile (300nm - 800nm)	Spettro elettromagnetico (1nm - $3\mu m$)
Cornea/cristallino	Lenti
Coni/bastoncelli	Sensore CMOS/CCD
Filtraggio nella retina	Digital image processing
Elaborazione nella corteccia visiva	Segmentation, object recognition, labeling...

La prima digitalizzazione di immagine risale al 1957, ma è a partire dagli anni '70 e '80 che sono emerse le prime applicazioni di image processing grazie all'aumento delle capacità di calcolo degli elaboratori.
I settori che hanno beneficiato dell'image processing sono moltissimi e gli scopi molto simili a quelli della robotica:

- Militare: immagini satellitari, engaging, face detection, bodyscanner, visori notturni
- Astronomico: image enhancement, fotografia infrarossa
- Medico: diagnostica per immagini, X-Ray, TAC, PET, MRI, analisi funzionale, creazione di atlanti
- Industriale: grasping, verifica strutturale, TAC, imballaggio

- Entrateinment: cinema, animazione, fotografia, fotogiornalismo

Con l'aumento della potenza di calcolo e della parallelizzazione dei processori si è potuta gestire una mole di dati sempre maggiore con un beneficio sia in termini di risoluzione delle singole immagini sia in termini di video processing.
La frontiera attuale per la robotica è il real-time video processing che consente di elaborare il flusso video e utilizzarlo come fonte di informazione sull'ambiente proprio come fa la Natura.
Attualmente il real-time video processing è utilizzato per la videosorveglianza, nei droni e nella robotica industriale ma si può trovare anche in prodotti commerciali come le fotocamere con face-detection.

1.4 Robot umanoidi

Una delle linee di ricerca della robotica moderna è rappresentata dalla robotica umanoide che si pone l'obiettivo di costruire dei robot dalle forme umane, dando loro la capacità di interagire con strumenti e in ambienti creati appositamente per l'uomo. Essa costituisce attualmente una delle sfide più ambiziose sia per la complessità a livello meccanico e di percezione, sia per la difficoltà di ricreare alcune funzioni o attività proprie dell'uomo, quali il coordinamento motorio e la sua intelligenza.
Il primo problema che si incontra quando si ha a che fare con un robot umanoide è la stabilità.
L'uomo è una delle poche specie che passa la maggioranza del tempo in una condizione di equilibrio instabile. Una vaga idea della difficoltà nel-

l'apparentemente semplice task della deambulazione lo può dare il fatto che impariamo a farlo sui due piedi dai 9 ai 12 mesi, mentre i nostri antenati quadrupedi solo dopo 3 o 4 mesi.
Per risolvere questo problema l'uomo utilizza due sensori molto diversi:

- **il labirinto**, che è formato da tre canali semicircolari posti ortogonali tra loro nei tre piani dello spazio e possiede recettori cigliati sui quali è presente un gel che, grazie al flusso dell'endolinfa provocato dal movimento corporeo, permette la percezione del movimento stesso.

- **gli occhi**, che informano il cervello sulla situazione spaziale nella quale si trova il corpo, permettendoci di valutare la posizione del corpo rispetto all'ambiente circostante e la direzione del movimento.

Per il principio della biomimica può essere vincente associare nei robot ai sensori inerziali, fondamentalmente paragonabili al labirinto, un sistema di visione artificiale che tra i tanti task che gli sono richiesti può anche occuparsi della propriocezione.

1.5 Ego-motion

L'egomotion in robotica è riferito al movimento nello spazio 3D di una telecamera. La sua stima utilizzando solo le informazioni acquisite dalla telecamera stessa fa parte della *visual odometry*.
Lo strumento principale della visual odometry consiste nello studio dell'**optical flow** (figura 1.5).
Dotata di solo un milione di neuroni e solo 3000 pixel per ciascun occhio,

la mosca domestica, per esempio, raggiunge la navigazione 3D a un'impressionante velocità di 700bodylength/s.

(a) Occhi di una mosca

(b) Telecamera singola

Figura 1.5. Optical Flow

La Natura ha raggiunto in questo caso quello che si sta ricercando nel campo della robotica aerea: stabilizzazione dinamica, navigazione autonoma 3D, collision avoidance, monitoraggio, attracco, decollo e atterraggio autonomo, ecc... Gli ultimi sette decenni hanno dimostrato che gli insetti volanti si guidano attraverso gli ambienti con l'elaborazione del flusso ottico (OF) che viene generato sui loro occhi come una conseguenza della loro locomozione.

Nella computer vision la visual odometry (figura 1.6) è così strutturata:

1. **Aquisizione delle immagini**: con telecamera singola, stereo o omnidirezionale.

2. **Calibrazione**: per rimuovere le distorsioni dovute all'ottica

3. **Feature detection**: selezione degli operatori che saranno cercati nei frame acquisiti per formare l'optical flow

4. **Tracking**: ricerca degli operatori corrispondenti nei frame successivi

5. **Rimozione outliers**: controllo degli errori durante il tracking analizzando i vettori dell'optical flow.

6. **Stima del movimento**: attraverso l'analisi dell'optical flow.

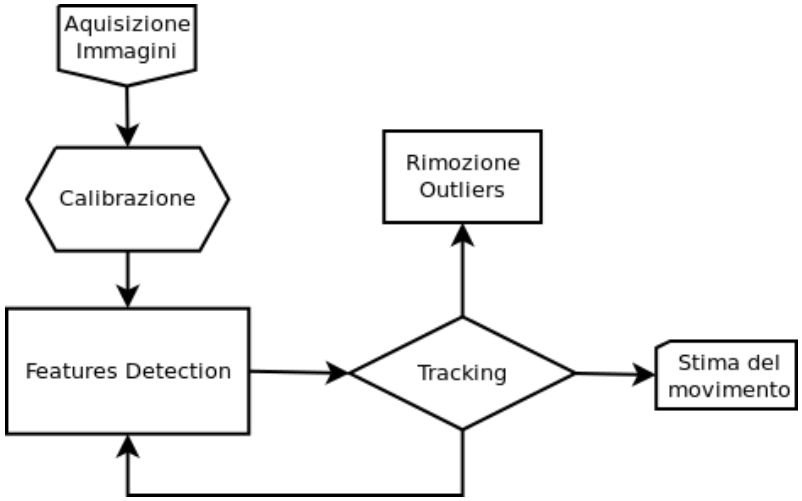


Figura 1.6. Visual odometry

Capitolo 2

Il robot: ROBOVIE-X

Lo scopo di questa tesi è fornire un supporto al robot nella stima dell'assetto usando la visione omnidirezionale e per questo si è iniziato con lo studiarne l'integrazione e l'interfacciamento.

Il punto di partenza è la tesi precedente di Carlo Tavian [41] che ha sviluppato un algoritmo di camminata statica con offset dinamici ricavati dai sensori inerziali. Alcuni passi sulla descrizione del robot e sul reverse engineering sono presi da essa.

La camminata è stata programmata tramite il software proprietario da Tavian e prevede l'adattamento grazie a dei valori letti da alcune celle di memoria, aggiornate tramite collegamento USB durante la camminata stessa.

La scoperta di queste celle e la loro scrittura sono frutto del reverse engineering iniziato da Riccardo Bonetto e proseguito con questa tesi e con quella di Tavian utilizzando parti del codice di Fabio Dalla Libera.

2.1 Descrizione

Il modello di robot utilizzato in questa tesi è lo Standard Robovie-X (KT-X), prodotto dalla V-Stone [38] figura 2.1.

Il robot è di dimensioni $343 \times 180 \times 71mm$ per un peso di 1.3kg. Possiede

Figura 2.1. Robovie-X

17 gradi di libertà, 3 per ogni braccio, 5 per ogni gamba e 1 per la testa. Ogni grado di libertà è attuato da un servomotore analogico modello VS-S092J capace di sviluppare una coppia di 9.03N/m e velocità massima di 9.52rad/s. La scheda di elaborazione on-board è fornita di un cpu a 60MHz (modello H8) con 64kB di ram e 512kB di ROM dove vengono caricati i movimenti e la mappatura del controller.

I servomotori vengono controllati dalla scheda attraverso 17 uscite in PWM. Il robot è fornito anche di un altoparlante per riprodurre dei suoni wav anch'essi caricati nella ROM della scheda.

La scheda può comunicare con alcuni tipi di controller wireless, tra cui i controller PlayStation 2 compatibili; e può venire programmata (mediante il software RobovieMaker2 figura 2.2) attraverso un'interfaccia USB2.0.

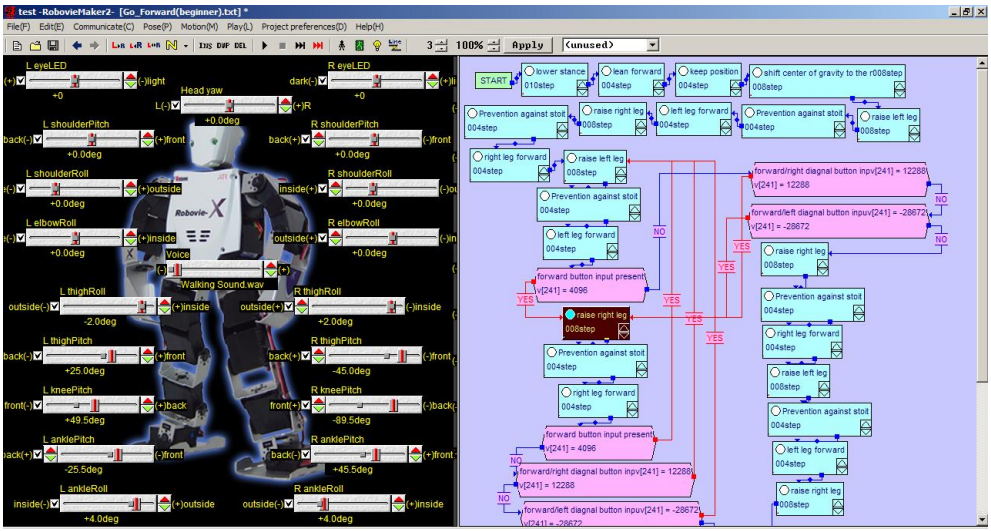


Figura 2.2. RobovieMaker 2

Robovie-X possiede inoltre un selettore a nove posizioni che permette di scegliere tra nove diversi set di movimenti e configurazioni di controller senza dover riprogrammare il robot.

Completa l'equipaggiamento del robot la scheda opzionale VS-IX001 contente il sensore inerziale, formato da un accelerometro a tre assi e un giroscopio a due assi (manca l'asse del giroscopio corrispondente all'asse longitudinale del robot).

Il robot è alimentato da un pacco batterie ricaricabile NiMh 6V composto da 5 celle con capacità di 1600mAh

2.2 Il software di fabbrica

Il software fornito a corredo del robot è chiamato RobovieMaker 2 ed è disponibile unicamente per Windows.

L'interfaccia consiste principalmente in due pannelli. Nel pannello di sinistra (Pose Area) sono presenti delle slidebar che consentono di controllare il valore di ogni variabile di giunto, oltre che la luminosità dei led ed altri parametri di minore importanza.

Nel pannello di destra (Motion Area) c'è uno spazio in cui è possibile programmare dei movimenti attraverso un semplice linguaggio a blocchi, che permette di memorizzare le pose definite nella Pose Area e di impostare le transizioni tra le varie pose.

È possibile utilizzare anche strutture condizionali (if-then-else) ed effettuare semplici operazioni tra un numero limitato di variabili. È importante sottolineare come le pose all'interno della Posa Area non possano essere definite in funzione di alcun parametro: l'unico modo di variare la posizione di un giunto è agire attraverso la relativa slidebar. Inoltre non è possibile in alcun modo interrompere, accelerare, variare l'esecuzione di una posa in corso, poiché tutte le operazioni sulle variabili vengono effettuate tra il termine di una posa e la successiva, e dunque non è neppure possibile iniziare un'altra posa o elaborare variabili nel contempo.

L'unico modo per variare la posizione del motore al di fuori del programma creato nella Motion Area è rappresentato dalla possibilità di sommare, al comando di ciascun motore, una variabile.

Questa variabile però sarà la stessa per ciascuna posa e non è possibile

modificarla durante l'esecuzione di una posa, ma solo attraverso i blocchi di calcolo nell'intervallo tra il termine di una posa e l'inizio della successiva.

Una volta creato il programma di movimento del robot nella Motion Area è possibile caricarlo nella memoria del robot ed associare un evento per l'esecuzione di tale programma. L'evento può essere la pressione di un tasto o di una combinazione di tasti sul controller, o la variazione di una variabile interna del robot (ad esempio la variabile che rappresenta il valore letto dall'accelerometro.)

Durante l'esecuzione di un programma, se il robot è collegato via USB al computer è possibile accedere alla lettura delle variabili interne attraverso il programma.

2.3 Limitazioni

Per quanto riguarda la scheda, la V-Stone non fornisce alcun modo di programmarla all'infuori del citato RobovieMaker2 con le limitazioni precedentemente esposte quali la semplicità delle strutture condizionali presenti, l'impossibilità di interrompere o variare una posa, l'impossibilità di definire una posa in modo parametrico.

Questo implica la necessità di effettuare le elaborazioni su un computer esterno alla scheda del robot stesso.

Purtroppo la V-Stone non ha reso pubblico il protocollo di comunicazione dei suoi robot, rendendo necessario il reverse engineering sull'apparato sperimentale.

Il protocollo USB2.0 non consente la comunicazione Full-Duplex, non per-

mette cioè di trasmettere e ricevere contemporaneamente.
In più questo protocollo non garantisce la costanza della velocità di trasmissione dei pacchetti. Questo rende difficoltoso garantire il funzionamento dell'algoritmo della camminata qualora si decidesse di inviare costantemente i valori di ogni singolo motore attraverso la porta USB.

2.4 Configurazione

Nella tesi di Carlo Tavian [41] si è posta la necessità di scegliere dove effettuare l'elaborazione dell'algoritmo per la camminata.
Date le limitazioni del protocollo USB2.0 esposte nel paragrafo precedente, effettuare tutta l'elaborazione sul PC esterno ed inviare solamente i comandi ai motori del robot avrebbe potuto rivelarsi problematico, specialmente nel caso di una camminata dinamica.
D'altra parte, l'impossibilità di effettuare operazioni complesse tramite il linguaggio a blocchi del RobovieMaker2 e l'impossibilità di programmare la scheda in altro modo, rende improponibile l'implementazione dell'algoritmo di una camminata "intelligente" (cioè che tenga conto dei dati rilevati dai sensori inerziali, azioni esterne, ecc) interamente sull'unità di elaborazione del robot.
Dato che poi il flusso video della telecamera non può essere gestito onboard per problemi di carico computazionale, l'interfacciamento con un elaboratore esterno è indispensabile.
Dal momento che esistono già delle camminate performanti fornite dalla V-Stone attraverso il suo software, Tavian ha deciso di procedere progettando degli algoritmi di "correzione" della camminata. Si è trattato quin-

di di programmare il robot per effettuare una camminata deterministica, cioè programmata staticamente sulla scheda del Robovie-X, ed elaborarne le correzioni dinamiche in risposta a stimoli ambientali su un PC esterno, comunicante via USB con la scheda del robot.
L'elaboratore esterno avrà accesso esclusivamente alle variabili interne della scheda, e non potrà comandare direttamente i motori, ma solo impostarne degli offset attraverso la scrittura di alcuni variabile interne al robot.

2.5 Reverse engineering

2.5.1 Protocollo USB

La prima fase del lavoro è stata progettare una infrastruttura di comunicazione tra la scheda del robot e l'elaboratore esterno collegato via USB.
Dal momento che l'azienda produttrice non fornisce alcuna specifica del protocollo di comunicazione tra il software e la scheda on-board, è necessario prima di tutto comprendere il protocollo di comunicazione attraverso tecniche di reverse engineering.
Gran parte del lavoro è stato effettuato da R. Bonetto [9], con il reverse engineering del software RobovieMaker2 e lo sniffing dei pacchetti USB, comprendendo la struttura base del pacchetto e i metodi per il comando dei motori del robot RB2000, anch'esso prodotto dalla V-Stone.
Partendo da questo lavoro, N. Frezzato e A.Casasola [12] hanno adattato e implementato in linguaggio C i metodi necessari al comando dei motori del

robot Robovie-X. Date le librerie utilizzate, il software creato era disponibile unicamente per Windows.
Agli scopi del lavoro di Tavian e di questo, rimanevano da implementare alcune funzioni:

- Lettura giroscopi ed accelerometri
- Lettura variabili interne robot
- Scrittura su variabili interne robot

Parte di questo lavoro è consistito nel proseguire il reverse engineering del protocollo USB per scoprire come effettuare ed implementare tali operazioni.
Per la cattura dei pacchetti USB è stato usato il software USBTrace; le operazioni di cattura ed analisi sono state effettuate in collaborazione con Tavian.

2.5.2 Struttura del pacchetto USB

Viene riportata ora la descrizione della struttura comune ad ogni pacchetto, determinata da R. Bonetto.
Ogni pacchetto USB, sia in ingresso che in uscita, è formato da 64 byte, organizzati in 32 Word. Ogni coppia rappresenta un valore, con la convenzione Little Endian , cioè con il byte più significativo trasmesso dopo il byte meno significativo. In risposta ad un singolo pacchetto inviato dal PC si possono ricevere uno o tre pacchetti. Tutti i pacchetti hanno la seguente struttura, indipendentemente dalla loro funzione:

- byte 00: **0x55**
- byte 01: rappresenta la quantità di pacchetti inviati/ricevuti consecutivamente. La prima cifra è l'indice del pacchetto corrente, la seconda è la quantità totale di pacchetti. Ad esempio, se il byte 01 vale "13", vuol dire che il pacchetto corrente è il secondo di tre pacchetti (il precedente avrebbe il byte 01 posto a "03" ed il successivo a "23")
- byte 02, 03, 05: rappresentano un codice che identifica il tipo di trasmissione.
- byte 04: solitamente usato quando il pacchetto tratta di un indice.

2.5.3 Lettura e scrittura delle variabili interne

La struttura di memoria della scheda di elaborazione del robot è costituita da 256 variabili, ciascuna indicata da un indice da 1 a 256. Molte di queste sono riservate per l'utilizzo esclusivo del robot, altre sono disponibili esclusivamente in lettura o in scrittura. Sono stati determinati dei range di alcune variabili:

- Le variabili da 66 a 128 sono disponibili in lettura e scrittura per l'utente
- Le variabili 129, 130 e 131 rappresentano il valore degli accelerometri (nell'ordine X, Y, Z) e sono di sola lettura
- Le variabili 132 e 133 rappresentano il valore dei giroscopi X e Y e sono di sola lettura

- Le variabili immediatamente successive sono di sola lettura, probabilmente per schede diverse dalla VS-IX001 che ospitino più sensori

Non sono stati indagati i significati di altri range di variabili.

2.5.4 Scrittura variabile interna

Per la scrittura di una generica variabile R/W si invia un pacchetto USB al robot così strutturato:

- byte 00: **0x55**
- byte 01: **0x01** (singolo pacchetto)
- byte 02, 03, 05:**0xE0,0x08,0x02** (codice della richiesta)
- byte 04: Indice della variabile moltiplicato per 2
- byte 06: Parte bassa (8 bit meno significativi) del valore da assegnare alla variabile
- byte 07: Parte alta (8 bit più significativi) del valore da assegnare alla variabile

Tutti i byte non specificati possono essere posti a zero.
Per effettuare la lettura di una variabile interna R/W è necessario inviare un apposito pacchetto e poi monitorare in pacchetti in entrata per leggere la risposta inviata dalla scheda del robot. Il pacchetto di richiesta di lettura è così composto:

- byte 00: **0x55**

- byte 01: **0x01** (singolo pacchetto)
- byte 02, 03, 05:**0xE0,0x08,0x02** (codice della richiesta)
- byte 06: Indice della variabile moltiplicato per 2

Tutti i byte non specificati possono essere posti a zero.
Una volta inviato tale pacchetto, il robot risponderà inviando un pacchetto contenente il valore richiesto; è quindi necessario leggere i pacchetti in entrata fintantoché non si riceverà il pacchetto di risposta, che verrà riconosciuto analizzando i primi 4 byte:

- byte 01:**0x01** (singolo pacchetto)
- byte 02,03:**0xE0,0x08**
- byte 04: Indice della variabile (moltiplicato per 2) di cui era stata richiesta la lettura
- byte 06: Valore della variabile, 8 bit meno significativi
- byte 07: Valore della variabile, 8 bit più significativi

2.5.5 Lettura dei sensori inerziali

La lettura dei sensori inerziali (indici da 129 a 133) viene fatta in maniera diversa dalle variabili R/W con indirizzo da 65 a 128.
Infatti utilizzando il precedente tipo di pacchetti che utilizza il solo byte 06 per specificare l'indice della variabile da leggere moltiplicato per due non è possibile accedere ad indirizzi superiori a 127 (in quanto $127 \cdot 2$ = **00xFE**, mentre $128 \cdot 2$ = **0x100** e necessiterebbe di due byte). Si è

ipotizzato che questa differenza del metodo di lettura sia dovuta la fatto che le variabili successive alla 127 siano da considerarsi, nella intenzioni dei programmatori, variabili riservate al sistema e quindi non normalmente accessibili agli utenti.
Sempre mediante il medesimo procedimento di cattura dei pacchetti USB, si è scoperto che il software RobovieMaker 2 accede a queste cinque variabili contemporaneamente inviando al robot un pacchetto così strutturato:

- byte 00: 0x55
- byte 01: 0x01 (singolo pacchetto)
- byte da 02 a 12:
- byte 13: **0xE0,0x00,0x10,0x04,0x00,0x0B,0x20,0x00, 0x8A,0xEC,0x06** (codice della richiesta) **0x09** indice del pacchetto, la risposta conterrà questo indice per identicare il pacchetto
- byte 14,15,16: **0xE9,0X02,0X0A**

il resto dei byte possono essere posti a zero. Una volta spedito tale pacchetto viene immediatamente spedito il pacchetto di risposta, contente:

- byte 04,05: Accelerometro, asse X
- byte 06,07: Accelerometro, asse Y
- byte 08,09: Accelerometro, asse Z
- byte 10,11: Giroscopio, asse X
- byte 12,13: Giroscopio, asse Y

Ciascun valore è da interpretare con la convenzione Little Endian come per i valori precedentemente mostrati.

2.5.6 Sviluppi futuri

Grazie all'implementazione di Tavian è possibile utilizzare i valori correttivi ottenuti dalla telecamera omnidirezionale con il metodo proposto in questo lavoro per migliorare la stabilità della camminata.
Il software proposto, anch'esso per Linux, utilizza le stesse librerie di quello di Tavian proprio per facilitarne l'integrazione futura.

Capitolo 3

La visione omnidirezionale

Il punto di partenza è la tesi precedente di Matteo Finotto [16] che ha sviluppato un algoritmo di visual giroscope tramite vanishing points, estraendo le linee dall'ambiente visto da una telecamera omnidirezionale. Alcuni passi sulla descrizione dell'apparato telecamera/specchio sono presi da essa.

Sebbene una telecamera omnidirezionale non segua propriamente i principi della biomimica, motivo per cui non è ammessa nelle competizioni RoboCUP, può essere una notevole facilitazione soprattutto per l'egomotion.

Le telecamere omnidirezionali hanno molte applicazioni in robotica: per il tracking di oggetti, per lo SLAM, per la stima dell'assetto negli UAV ecc...

Come è intuibile, una telecamera frontale con un limitato campo visivo sarebbe stata restrittiva per la stima dell'assetto, quindi si è deciso di ampliare la regione visibile affiancandola ad uno specchio iperbolico per ottenere un'immagine omnidirezionale, ossia con un campo visivo orizzontale di 360°.

Un sistema così formato viene detto catadiottrico, unione delle parole

greche *katoptrikè* (composta da *katà*, contro, e *optomai*, vedere), ossia la scienza della luce riflessa, e *diaptrikè* (composta a sua volta da *dià*, attraverso, e *optomai*), ossia la scienza che concerne i fenomeni della luce rifratta.

Per aumentare il campo visivo sono comunque possibili altre varianti: l'uso di più telecamere, l'uso di una telecamera mobile, l'uso di lenti fisheye, ognuna con i propri pregi e i propri difetti come descritto in figura 3.1.

Sensore omnidirezionale	Registrazione dell'immagine	Isotropico	Dinamico	Risoluzione dell'immagine	Note
Telecamera mobile/panoramica	Problemi meccanici	Solo per i pixel centrali perpendicolari alla rotazione	No, ritardo nel tempo	Alta per ogni angolo	Non adatta per scene dinamiche, costosa
Insieme di telecamere con angoli di visuale diversi	Complicata	No, ogni telecamera ha il suo angolo	Sì, se le telecamere sono sincronizzate	Alta per ogni angolo	Consumo di banda, costi, tutte le telecamere devono essere calibrate e sincronizzate
Catadiottrico	Ok, soprattutto con un solo punto di proiezione	Sì	Sì	Bassa	Unica immagine, invariante alla rotazione, distorsione
Fisheye	Facile ma le lenti richiedono calibrazione e correzione	Sì	Sì	Bassa in periferia, da media a alta al centro	Vista emisferica, distorsione

Figura 3.1. Confronto tra i vari sensori omnidirezionali. (Adattata da http://cswww.essex.ac.uk/mv/capture.html)

Il sensore catadiottrico offre numerosi benefici. Innanzitutto l'immagine è invariante alla rotazione e quindi non dipende dalla direzione della telecamera (isotropia): l'oggetto viene sempre visto anche se non si trova di fronte al robot.

Un campo visivo così ampio rende l'immagine robusta alle occlusioni.

Inoltre questo tipo di visione induce una semplice geometria non limitata dallo stretto campo visivo di una telecamera tradizionale [25]: tutte le rette verticali vengono trasformate in rette radiali, le altre linee rette diventano circonferenze nell'immagine omnidirezionale e quindi i punti all'infinito di rette parallele corrispondono alle intersezioni di circonferenze nella sfera equivalente [15] [42].
Infine tutto l'ambiente che circonda il robot viene catturato mediante un unico scatto, a scapito però di un'elevata risoluzione.
Un esempio di immagine omnidirezionale è riportato in figura 3.2(a) nella quale si vede l'intero laboratorio di Robotica Autonoma così come lo osserva il robot.
Questa immagine però non è molto comprensibile per l'occhio umano. Ecco che può essere quindi "srotolata" andando a formare il cosiddetto cilindro panoramico (figura 3.2(b)).

3.1 La telecamera

Sulla testa del robot è posta in orizzontale una webcam USB Philips SPC1000NC [36] con uno specchio iperbolico [24]
La telecamera sarà puntata verso l'alto, dove cattura l'immagine riflessa dallo specchio omnidirezionale.
La webcam misura 40 mm x 82 mm x 88 mm e pesa 110 grammi, monta un sensore CMOS con risoluzione massima di 1280 x 1024 (1.3 MP). Altre caratteristiche degne di nota sono i 30 frame al secondo e un campo visivo di 80° .

(a)

(b)

Figura 3.2. (a) Esempio di immagine omnidirezionale e (b) relativo cilindro panoramico

3.2 Lo specchio

Baker e Nayar [4] hanno classificato i sensori catadiottrici in due categorie in base al numero di punti di vista (o centri di proiezione).

Il caso più favorevole è quello di avere un unico centro di proiezione. Ciò significa che ogni raggio luminoso che attraversa l'apertura dell'ottica, e che si è precedentemente riflesso sullo specchio, passerebbe per un singolo punto (punto di vista) se non ci fosse lo specchio stesso (punto O in figura 3.3) [18].

In questo modo è possibile generare immagini prospettiche geometricamente corrette e non distorte a partire da quelle omnidirezionali. Gli specchi con questa caratteristica maggiormente utilizzati sono quelli parabolici e quelli iperbolici.

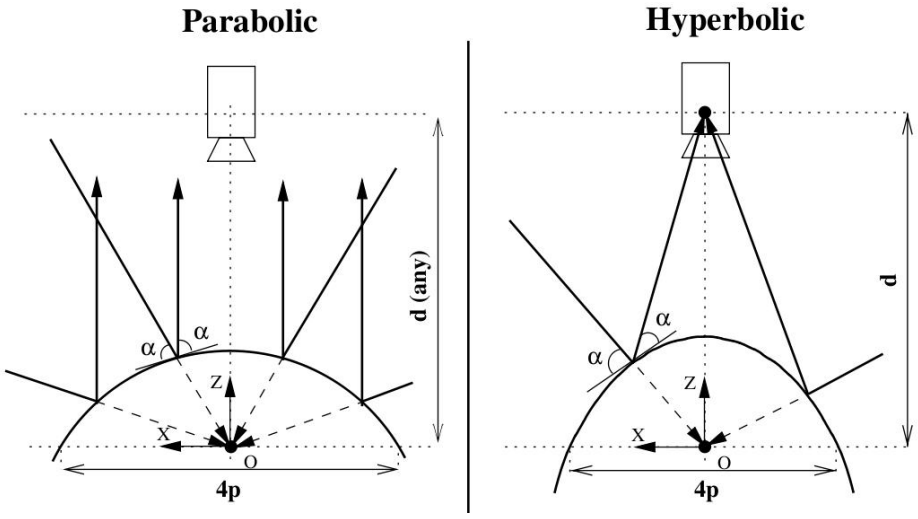


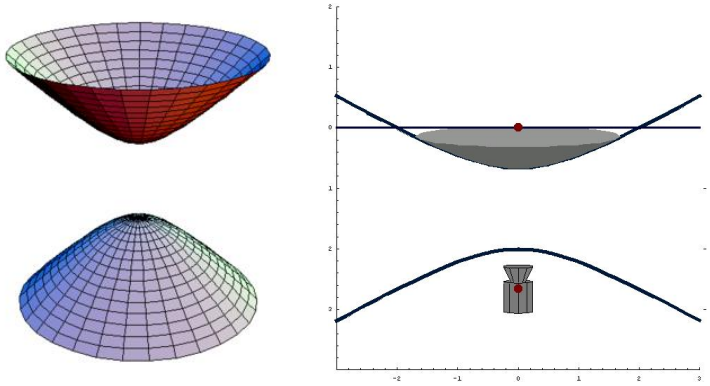
Figura 3.3. Differenza tra specchio parabolico e specchio iperbolico

In questo lavoro l'utilizzo di una telecamera prospetticha ci ha vincolato alla scelta dello specchio iperbolico, in particolare il modello VS-C14 prodotto dalla Vstone [24] e visibile in figura 3.4.

Figura 3.4. Lo specchio omnidirezionale

Lo specchio è modellato dalla superficie di una delle due parti di un iperboloide a due falde (figura 3.5(a)), descritto dalla formula $\frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$ ottenuto ruotando un'iperbole attorno al proprio asse focale. I raggi ottici si intersecano dapprima nel fuoco di questa falda dell'iperboloide, per poi

convergere nel punto focale dell'altra falda, che coincide con il centro ottico della telecamera (figura 3.5(b)).



(a) (b)

Figura 3.5. (a) Esempio di immagine omnidirezionale e (b) relativo cilindro panoramico

In questo modo il sistema telecamera-lente-specchio può essere considerato un tutt'uno creando un sistema con un solo punto di proiezione.

Al vertice dello specchio è attaccato uno stelo plastico brevettato che serve, oltre a centrare lo specchio sulla telecamera, ad evitare un auto riflesso della telecamera stessa.

Lo specchio ha un diametro di 30 mm ed è alloggiato in un cilindro di plastica trasparente alto 48 mm, con un diametro di 41 mm e un peso totale di 40 grammi. Può quindi essere montato facilmente sulla sommità di un robot senza creare significativi spostamenti di baricentro.

Il campo visivo verticale à mostrato in figura 3.6: fino a 15° sopra il piano orizzontale dello specchio e fino a 60° sotto.

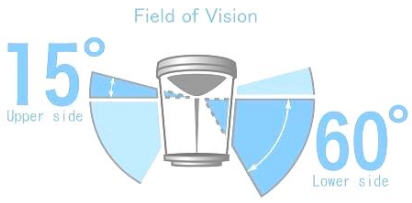


Figura 3.6. Campo visivo dello specchio V-Stone

3.3 La sfera equivalente

Con l'unificazione teorica di tutti i tipi di telecamere omnidirezionali [5] è stato possibile introdurre il concetto di sfera unitaria che, dopo opportuna calibrazione, permette di effettuare calcoli su uno spazio SO(3) senza distorsioni geometriche.

Geyer e Daniilidis [18] hanno dimostrato come la proiezione di un punto del mondo reale sul piano immagine attraverso un sensore a singolo punto di vista, possa essere modellata con un mapping a due fasi: il punto reale viene mandato sulla superficie della sfera e da qui sul piano immagine partendo da un punto sull'asse ottico della telecamera. La posizione di questo punto di proiezione dipende dalla forma dello specchio.

Interessante è notare come, una volta dimostrata questa equivalenza, il modello della sfera sia valido per ogni tipo di specchio a singolo punto di vista, andando a costituire le basi per una teoria unificativa per tutti i sensori.

Si prenda per semplicità uno specchio parabolico. Il centro di proiezione coincide col fuoco della parabola. Un raggio di luce incidente nel fuoco della parabola viene riflesso dallo specchio in un raggio parallelo all'asse

della parabola come in figura 3.7. In questo modo ogni punto nell'immagine è in corrispondenza biunivoca con il raggio passante per il fuoco.

Vediamo ora l'equivalenza tra la proiezione fatta attraverso lo specchio

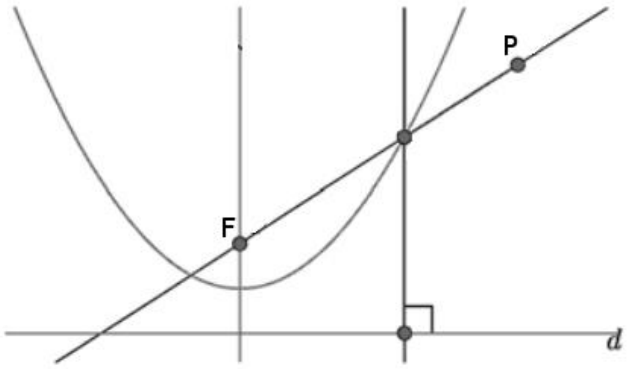


Figura 3.7. Un raggio incidente con il fuoco della parabola viene riflesso in un raggio parallelo all'asse della parabola

parabolico e la proiezione che coinvolge la sfera. Poichè lo specchio è un paraboloide regolare e poichè il mapping biunivoco è preservato in ogni piano che passa per l'asse della parabola, è sufficiente considerare una sezione del paraboloide.

Sia P una parabola, F il fuoco e d la direttrice. Sia l la retta parallela a d e passante per F . Si consideri la seguente definizione della proiezione Q del punto P riflesso dalla parabola sulla retta.

Definizione 1 *Q è la proiezione del punto R sulla retta l, dove R è l'intersezione della parabola P con la retta FP (figura 3.8 a sinistra).*

Può essere data una definizione alternativa. Sia C una circonferenza con centro F e raggio pari a $2p$, dove p è la distanza focale della parabola. La circonferenza e la parabola P si intersecano in due punti lungo la retta

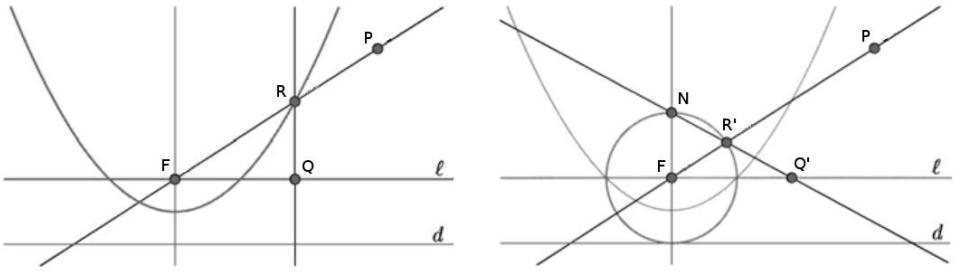


Figura 3.8. Rappresentazione della Definizione 1 (a sinistra) e della Definizione 2 (a destra). (Da [18])

l, e la direttrice d è tangente alla circonferenza. Sia N il polo nord della circonferenza, ossia il punto diametralmente opposto all'intersezione della circonferenza con la direttrice.

Definizione 2 *Q è la proiezione del punto R sulla retta l dal punto N , dove R è l'intersezione della retta FP con la circonferenza C (figura 3.8 a destra).*

Da queste due definizioni si ricava il seguente Lemma.

Lemma 1 *I punti Q e Q ottenuti dalla proiezione del punto P come descritto rispettivamente nelle definizioni 1 e 2 sono coincidenti.*

Lo stesso ragionamento si può facilmente estendere al caso 3D, concludendo che la proiezione attraverso uno specchio parabolico è equivalente alla proiezione su una sfera seguita dalla proiezione stereografica sul piano immagine.
Per la dimostrazione del Lemma 1 e dell'estensione al caso tridimensionale, si faccia riferimento a [18].

Ci si può chiedere adesso se lo stesso discorso possa valere per specchi diversi da quello parabolico (sempre ad unico centro di proiezione).
La prima proiezione del punto P sulla sfera non dipende dalla forma dello specchio dato che si fa sempre nello stesso modo tramite la retta FP . La differenza nella seconda proiezione consiste solamente nella posizione del punto sull'asse della sfera dal quale proiettare verso il piano immagine. Tale posizione dipende dall'eccentricità della conica. Come appena visto, uno dei due casi limite è rappresentato dallo specchio parabolico dove il punto coincide con il polo nord della sfera. L'altro caso limite è rappresentato da uno specchio piano, caso degenere di un sistema catadiottrico con eccentricità infinita. In questo caso il secondo centro di proiezione coincide con il centro della sfera. Nel caso di uno specchio iperbolico, come quello utilizzato in questa tesi, il centro della seconda proiezione cade tra i due estremi, come si può vedere in figura 3.9. Si può calcolare come la

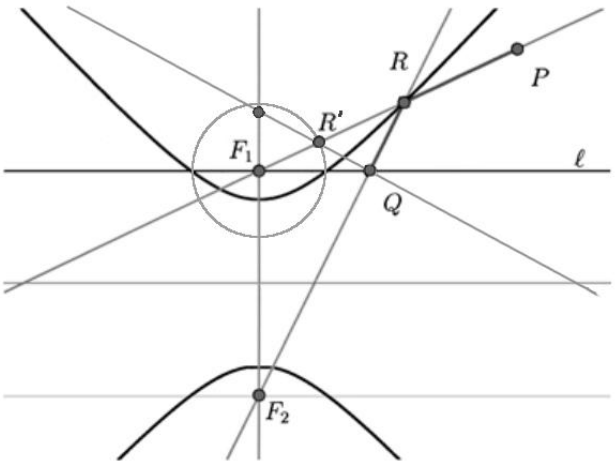


Figura 3.9. Proiezione nel caso di specchio iperbolico. (Adattato da [18])

distanza di questo punto dal centro della circonferenza sia pari a

$\frac{d}{\sqrt{d^2+4p^2}}$

dove d è la distanza tra i due fuochi e $4p$ è il lato retto (corda che passa per il fuoco $F1$ parallela a d) [7]. In questo modo è stata fornita una teoria che unifica tutti gli specchi ad un singolo punto di vista, permettendo di lavorare in un framework generale indipendentemente dal fatto che lo specchio usato sia parabolico o iperbolico. Inoltre, calcolare la proiezione di un punto del mondo sulla sfera e da qui al piano immagine è semplice e, poichè la funzione è biiettiva, si può riproiettare ogni punto dell'immagine 2D nuovamente sulla sfera e da qui ottenere informazioni particolari come vedremo nei capitoli successivi.

Bisogna comunque notare che la seconda proiezione inversa, quella dalla sfera al mondo reale non è univoca perchè si può determinare la direzione ma non è possibile ricavare la distanza dell'oggetto dal fuoco dello specchio.

La sfera ha poi alcune proprietà molto interessanti per quanto riguarda la proiezione: le linee rette del mondo reale vengono mappate in circonferenze con centro e raggio uguali a quelli della sfera.

3.4 La calibrazione del sensore

La calibrazione della configurazione specchio iperbolico e telecamera prospettica è stata effettuata con OcamClib Toolbox per Matlab [32]

Per questo lavoro è stata particolarmente utile la funzione TOFILE implementata da Finotto [16] che, una volta convertite le coordinate dei pixel dell'immagine omnidirezionale nei corrispondenti punti della sfera unitaria, li salva in un file di testo.

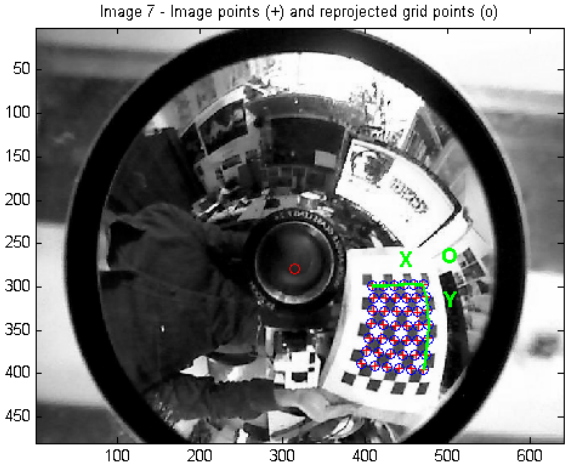


Figura 3.10. Calibrazione del sensore a risoluzione 640x480 pixel

In questo modo la calibrazione del sensore viene eseguita una sola volta, e la conversione delle coordinate da 2D a 3D si riduce all'estrazione di un elemento dall'array di coordinate, con chiari benefici computazionali.

Capitolo 4

Il giroscopio visivo

4.1 L'optical flow

Il flusso ottico citato nell'introduzione è una approssimazione del movimento dell'immagine, definita come la proiezione delle velocità dei punti di superficie 3D del mondo reale sul piano immagine di un sensore visivo [8]. Secondo la formulazione di Horn e Schunck [23], esso si può esprimere usando l'informazione locale. Si considera la funzione di intensità I, in funzione della posizione e del tempo, approssimata con le serie di Taylor:

$$I(x+dx, y+dy, t+dt) = I(x,y,t) + \frac{\delta I}{\delta x}dx + \frac{\delta I}{\delta y}dy + \frac{\delta I}{\delta t}dt + H.O.T \quad (4.1)$$

dove si possono trascurare gli $H.O.T.$ assumendo movimenti piccoli.
Da questa equazione possiamo ricavare l'equazione del flusso ottico:

$$\frac{\delta I}{\delta x}V_x + \frac{\delta I}{\delta y}V_y + \frac{\delta I}{\delta t} = 0 \quad (4.2)$$

dove V_x e V_y sono le velocità delle componenti x e y; e $\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}$ e $\frac{\delta I}{\delta t}$ le derivate parziali della funzione intensità nelle direzioni corrispondenti.
In letteratura l'equazione del flusso ottico 4.2 si trova spesso nella forma

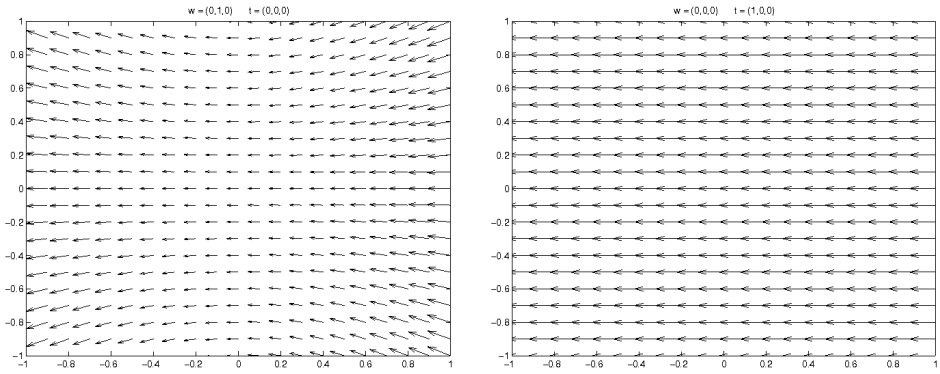
matriciale:

$$\bigtriangledown I \cdot \vec{V} = -\frac{\delta I}{\delta t} \tag{4.3}$$

La presenza di due incognite ci costringe a cercare un vincolo ulteriore (*aperture problem*).

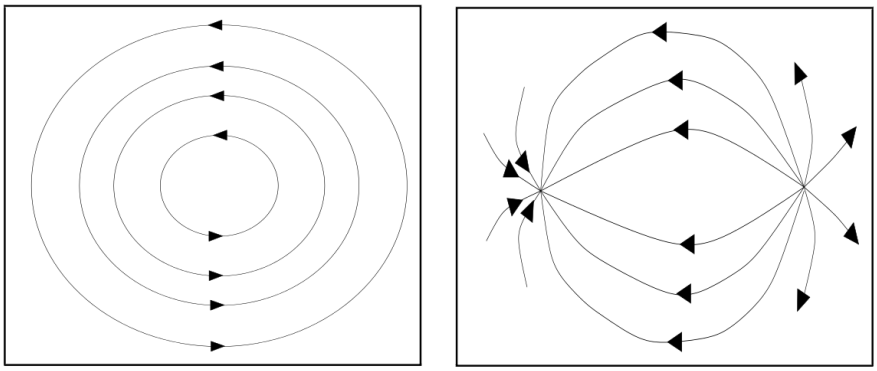
Nel paragrafo 4.4.4 vedremo come il metodo di Lucas-Kanade [28] risolve questo problema.

Per questo lavoro è importante notare che l'uso di una telecamera omnidirezionale risolve i problemi di ambiguità tra rotazioni e traslazioni che si porrebbero usando una telecamera convenzionale, in quanto riceve informazioni da tutte le direzioni [40] e [14] (figura 4.1 e 4.2).



(a) Rotazione lungo Y (b) Traslazione lungo X

Figura 4.1. Ambiguità di rotazione e traslazione in una telecamera convenzionale



(a) Rotazione lungo Y (b) Traslazione lungo X

Figura 4.2. Rotazione e traslazione in una telecamera omnidirezionale [40]

4.2 Il focus of expansion

Percepiamo il movimento nostro e del nostro sguardo all'interno di un ambiente con oggetti statici come fluire della proiezione sulla retina di tale ambiente.

Per un dato movimento questo ambiente sembra uscire da un punto preciso sulla retina: il focus of expansion (FOE).

Questo punto corrisponde al punto dove il vettore (3D) che specifica la direzione istantanea del movimento (il vettore tangente al motion-path descritto dal'istante iniziale a quello corrente) interseca il piano immagine.

Come spiegato in [6], utilizzando il semplice modello pinhole abbiamo:

$$x' = \frac{x}{z} \tag{4.4}$$

$$y' = \frac{y}{z} \tag{4.5}$$

Considerando u,v e w come le velocità dei punti del mondo x,y,z e un punto $P = (x_0, y_0, z_0)$ al tempo iniziale, dopo un intervallo di tempo t la sua immagine sarà:

$$(x', y') = \left[\frac{x_0 + ut}{z_0 + wt}, \frac{y_0 + vt}{z_0 + wt}\right] \tag{4.6}$$

Al variare di t questa equazione del "flow-path" non è altro che quella di una retta. Per $t \to -\infty$ il punto sul piano immagine diventa:

$$FOE = \left[\frac{u}{w}, \frac{v}{w}\right] \tag{4.7}$$

Il FOE viene usato spesso nella dynamic scene analysis per un'analisi qualitativa delle traiettorie degli oggetti in movimento.

In [20] invece, si utilizza un'immagine omnidirezionale che presenta due

FOE antipodali: uno da cui l'ambiente sembra uscire, l'altro in cui l'ambiente sembra entrare.

Sfruttando questa proprietà ed estraendo i FOEs, viene calcolata la posizione di un robot che si muove in 2D.

Nell'articolo sopracitato si utilizza però un'immagine omnidirezionale prodotta da una sequenza di immagini catturate da una telecamera in rotazione attorno ad un punto che non coincide con il punto nodale dell'ottica.

In questo modo si sfrutta la mancanza del vincolo SVP [4] per ricostruire informazioni sulla profondità, ottenendo quello che nell'articolo chiamano *Monocular Omnidirectional Stereo*.

In questo lavoro, invece, si sfrutta la calibrazione nella sfera unitaria per calcolare il versore $\overrightarrow{OF}$ dove O è il centro della sfera e F è il FOE estratto. $\overrightarrow{OF}$ fornisce quindi l'informazione sulla direzione del movimento della telecamera e lo studio della sua variazione nel tempo sarà sufficiente per l'implementazione di un visual gyroscope.

4.3 L'idea

La maggioranza dei progetti per l'ego-motion che fanno uso di telecamere omnidirezionali [19] [33] [2] cercano di ricostruire la scena in 3D utilizzando la geometria epipolare (figura 4.3), con metodi come l'*8-point algorithm* [21]

Per lavorare in real-time sono invece consigliate soluzioni computazionalmente meno pesanti, in vista anche di un'implementazione embedded.

Gli algoritmi più performanti di attitude estimation sono infatti quelli utilizzati negli UAV [15] che si basano sull'estrazione e sul tracking della linea

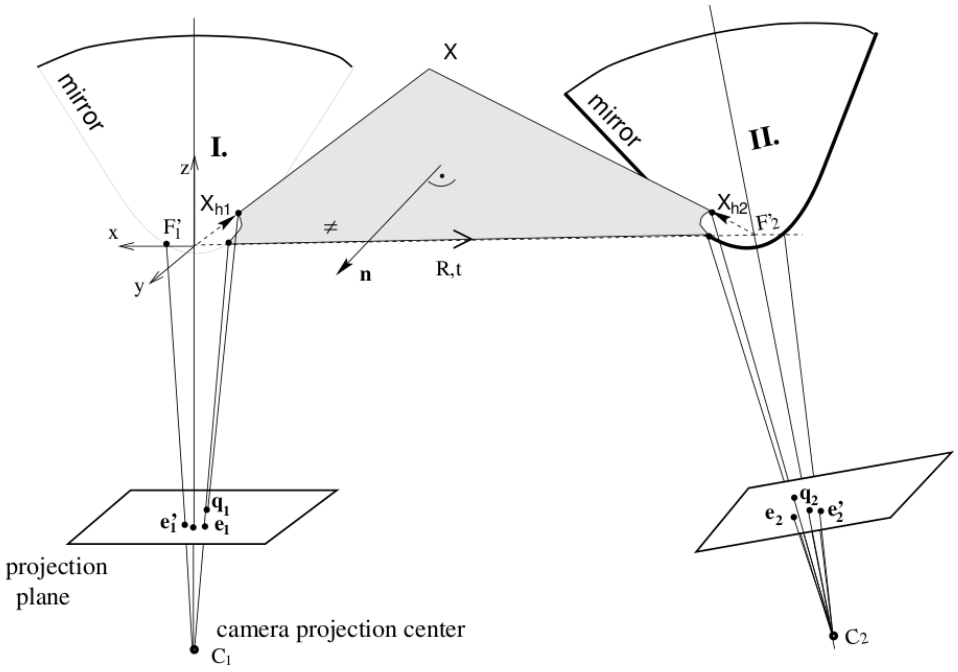


Figura 4.3. Geometria epipolare di due telecamere omnidirezionali con specchio parabolico [40]

dell'orizzonte; questa è infatti relativamente semplice da estrarre dato che si lavora in ambienti esterni e in quota.

Esistono anche algoritmi [17] [25] [29] che sfruttano metodi di *central catadrioptic line detection* [44] per individuare i punti di fuga (VPs) dell'immagine ed effettuarne il tracking, oppure per il *visual servoing* come in [45] [1]. Questi algoritmi sono solitamente usati in ambienti indoor dotati di oggetti che facilitano l'individuazione dei punti di fuga come tavoli, librerie, travi, scale, ecc.. e non sono particolarmente robusti, come viene fatto notare in come viene fatto notare in [30], perchè gruppi di oggetti posizionati in modo non coerente alle direzioni principali, come dei libri inclinati negli scaffali, travi a 45°, ecc.., possono incidere sul risultato.

In [42] si utilizza invece l'estrazione dei VPs da un'immagine omnidirezionale per la registrazione di immagini off-line allo scopo di ricostruire scene urbane.

Date le problematiche dell'approccio spaziale, per ampliare al massimo i possibili scenari d'uso si è scelto un approccio temporale, in modo da non avere la necessità di trovare features particolari come rette e segmenti per estrarre i punti di fuga, ma limitandosi al tracking dei corners dell'immagine.

Nella maggioranza degli algoritmi citati viene usato prima un metodo spaziale, per individuare i punti di fuga tramite proiezioni di rette del mondo 3D sul piano immagine, e poi un metodo temporale, per il loro tracking e conseguente calcolo dell'assetto.

In questo lavoro invece si usa un metodo temporale per il tracking di features generiche e successivamente si considera il tempo come una dimesione spaziale calcolando un *vanishing point del movimento*, ovvero il punto al quale tutti i tracks estratti tendono, cioè il *Focus of Expansion* dell'optical flow.

4.4 Il software

Il software è stato sviluppato in C++ per Linux utilizzando le librerie OpenCV.

A differenza della visual odometry esposta nell'introduzione, un visual giroscope richiede solo il calcolo dell'assetto del robot, in quanto la mancanza di informazioni sulla profondità impedisce una stima quantitativa dello spostamento.

Seguendo comunque il paradigma della visual odometry, si può dividere il software in tre aspetti fondamentali:

- **Acquisizione delle immagini**: mediante telecamera omnidirezionale
- **Calibrazione**: cambio delle coordinate dal piano immagine alla sfera unitaria
- **Features Detection**: scelta automatica degli oggetti da seguire nei frame per ottenere l'optical flow
- **Tracking**: individuazione dello spostamento nel tempo degli oggetti scelti al passo precedente durante tutta l'esecuzione
- **Rimozione outliers**: studio dei tracks e clustering
- **Stima dell'assetto**: dai tracks e dal FOE.

Qui di seguito verrano esposti i singoli passaggi indicati anche in figura 4.4.

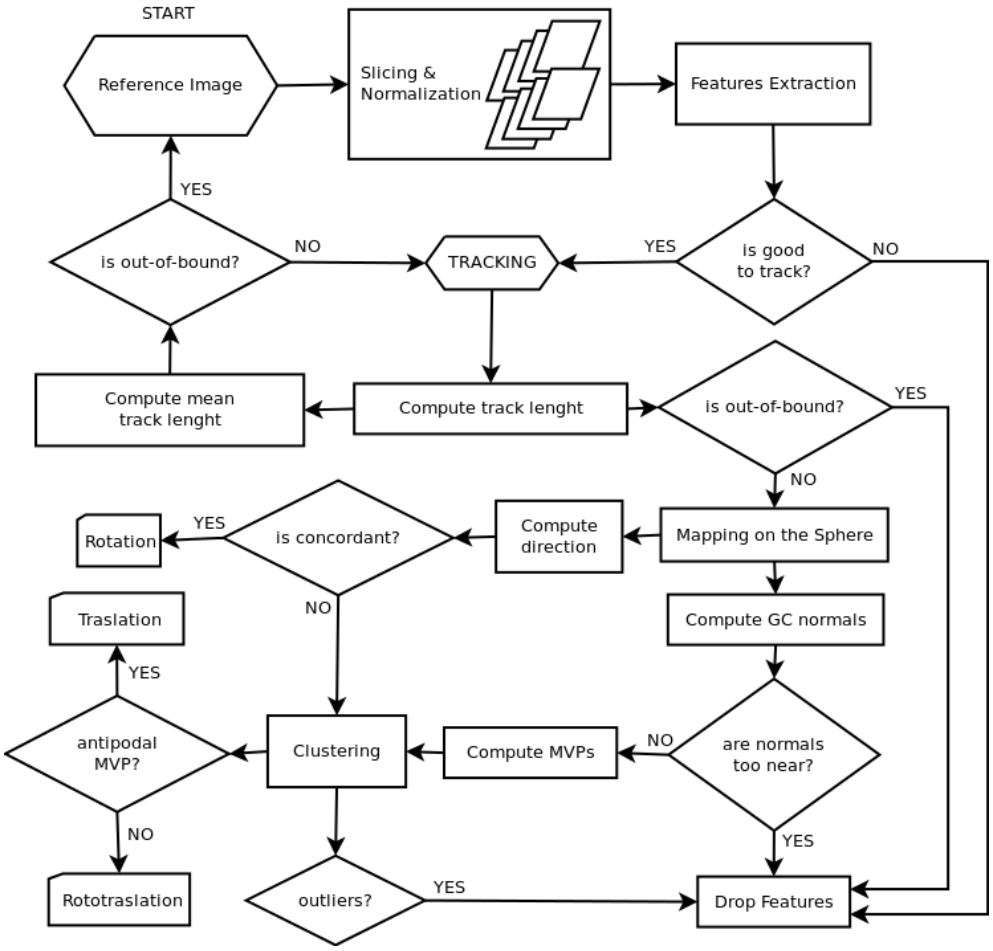


Figura 4.4. System overview

4.4.1 Acquisizione delle immagini

La telecamera omnidirezionale è facilmente accessibile grazie Video4Linux. Purtroppo però alcune impostazioni essenziali per il buon funzionamento del programma non sono accessibili tramite le OpenCV e per questo la telecamera va configurata inizalmente tramite un qualsiasi software di gestione delle webcam come *uvcdynctrl* e *guvcview* vedi (Appendice A).
Nello specifico è importante disabilitare l'**Auto Exposure** in quanto produrrebbe un vistoso calo nel framerate per cercare di ottenere un'immagine più luminosa in caso di necessità.
Una volta impostata la telecamera il software può iniziare l'acquisizione dei fotogrammi.

4.4.2 Calibrazione

Sfruttando il file generato dalla calibrazione effettuata con oCamCalib si possono rimappare le posizioni in pixel dell'immagine in posizioni 3D sulla superficie della sfera unitaria.
In questo modo si potranno effettuare i calcoli dei tracks con il minimo sforzo.

4.4.3 Features Detection

La selezione delle features è cruciale per la robustezza del metodo, per questo l'immagine omnidirezionale è stata divisa in 8 settori utilizzando delle maschere per definire le Region Of Interest (ROI), come in figura 4.5.

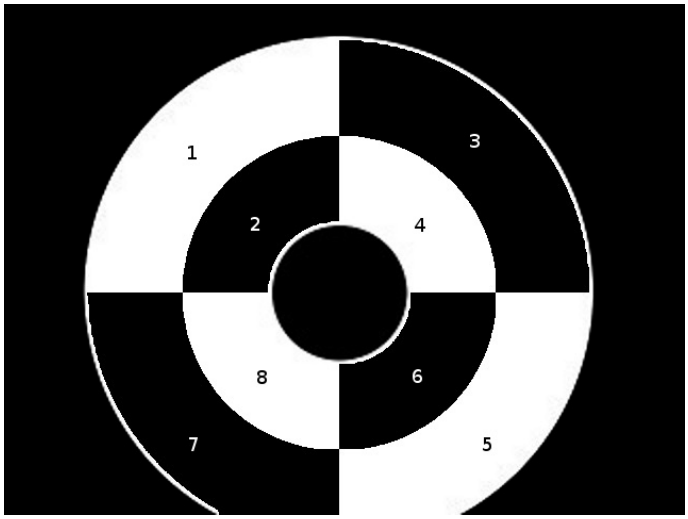


Figura 4.5. Suddivisione dell'immagine.

Ad ogni settore viene applicato un *contrast stretching* con il metodo *cvNormalize* di OpenCV [34] per evitare che zone particolarmente luminose come finestre o lampade impediscano una corretta features detection nelle aree più scure.

All'inizializzazione vengono scelte $N = 25$ features per ogni settore utilizzando il metodo *cvGoodFeaturesToTrack* di OpenCV. Questo metodo seleziona i corners nell'immagine calcolando il minimo autovalore della matrice gradiente [39].

Le maschere vengono utilizzate solo durante l'acquisizione in modo che il tracking avvenga anche se le features escono dai loro settori.

L'uso delle ROI unito al contrast stretching è molto importante per essere sicuri di estrarre features in tutte le direzioni. Se così non fosse si potrebbero avere ampie regioni senza alcuna feature e non si potrebbero sfruttare i vantaggi dell'uso di una telecamera omnidirezionale, distinguendo tra rotazioni su un asse e traslazioni lungo un suo ortogonale, come presentato

nel paragrafo 4.1.

Per ogni feature estratta si applica la funzione *cvFindCornerSubPix* (figura 4.6) che è molto importante per avere una stima accurata della posizione della feature a risoluzione sub-pixel.

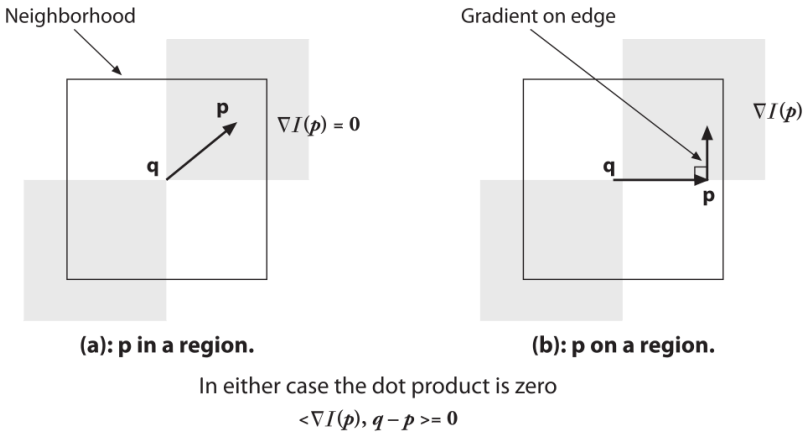


Figura 4.6. Accuratezza subpixel: (a) l'area attorno al punto p è uniforme e il suo gradiente è 0; (b) il gradiente sul bordo è ortogonale al vettore $\vec{qp}$ lungo il bordo; in entrambi i casi, il prodotto scalare tra il gradiente in p e il vettore $\vec{qp}$ è 0, [11]

Grazie alla calibrazione i punti sul piano immagine possono essere proiettati sulla sfera unitaria e si può calcolare la direzione dei singoli tracks come segno dell'angolo di colatitudine o, se pari a zero, dell'angolo azimutale.

4.4.4 Tracking

Il tracking di Lucas-Kanade [28] risolve il problema del calcolo dell'optical flow presumendo che esso sia sostanzialmente costante in un'area locale del pixel in esame. In questo modo si risolve l'equazione differenziale del flusso (4.3) per tutti i pixel in quell'area, ottenendo un sistema sovradimensionato

e applicando il criterio dei minimi quadrati.

$$\begin{cases} \sum_{x,y} W(x,y) I_x I_y u + \sum_{x,y} W(x,y) I_y^2 v = -\sum_{x,y} W(x,y) I_y I_t \\ \sum_{x,y} W(x,y) I_x^2 u + \sum_{x,y} W(x,y) I_x I_y v = -\sum_{x,y} W(x,y) I_x I_t \end{cases}$$

in cui $W(x,y)$ è la finestra Gaussiana usata per raggruppare pixel, mentre I_x, I_y e I_t sono le derivate parziali della funzione intensità nelle direzioni corrispondenti.

Per questo lavoro si è scelto di utilizzare il metodo di Lucas-Kanade piramidale [10] (figura 4.7). Il flusso ottico viene calcolato al livello più alto della piramide; successivamente la stima ottenuta viene propagata verso il basso fino a che non si arriva all'immagine originale.

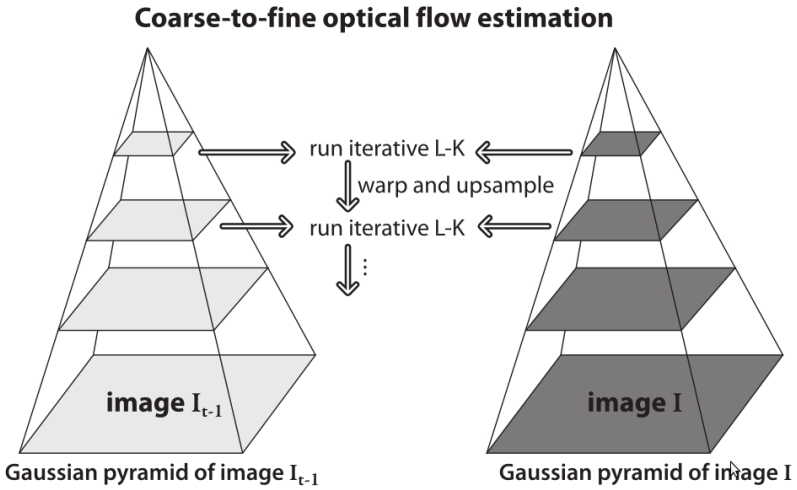


Figura 4.7. Suddivisione dell'immagine.

4.4.5 Rimozione outliers

Durante l'esecuzione dell'algoritmo si ha un continuo ricambio di features. Alcune vengono scartate dall'algoritmo di tracking perchè escono dal cam-

po visivo o perchè non vengono più considerate affidabili dal metodo *cvCalcOpticalFlowPyrLK* di OpenCV.
Un'ulteriore scrematura si ha in base alla lunghezza dei tracks in modo da rimuovere quelli troppo lunghi, prodotti da errori di matching, e quelli troppo corti, dovuti al rumore o da oggetti troppo lontani.
Infine si eliminano le features che hanno generato MPVs lontani dai cluster considerati, come spiegato più dettagliatamente nel prossimo paragrafo.
Ad ogni nuova esecuzione gli outliers trovati vengono rimpiazzati da nuove features in modo che il loro numero totale su ogni settore resti invariato.
Prima della ricerca di nuove features, tutte le features precedenti vengono oscurate insieme ai loro 8-neighbours creando delle aree circolari nere sull'immagine. Questo è necessario per evitare di selezionare due volte le stesse features oppure quelle appena rimosse. Ad ogni esecuzione si effettuano dei calcoli statistici sui tracks e si è valutato di riaggiornare l'inizializzazione in base alla lunghezza media di essi: quando la maggioranza dei tracks supera una certa soglia significa che il robot si è spostato dalla posizione precedente e si può aggiornare la reference image.

4.4.6 Stima dell'assetto

L'idea di base è quella di ricavare le variazioni dell'assetto dai tracks dell'optical flow e dallo spostamento del FOE.
Come dimostrato in [18] la proiezione di ogni retta del mondo reale sulla sfera forma una circonferenza con centro e raggio coincidenti con quelli della sfera, chiamata Great Circle (GC). In questo caso al posto di avere una retta nello spazio si considera la retta nel tempo passante per gli estremi

del track.

Esattamente come in [17], siano $P_i = (X_i, Y_i, Z_i)$ la posizione dell'i-esima feature al tempo $t = 0$ e $P'_i = (X'_i, Y'_i, Z'_i)$ al tempo $t = 1$.

Questi due punti definiscono sulla sfera un'unica GC $\mathcal{C}$ che giace sul piano $\mathcal{P}$ passante per il centro della sfera O, la cui normale è $\vec{n} = \overrightarrow{OP_i} \times \overrightarrow{OP'_i}$. Siano $\overrightarrow{n_i}$ e $\overrightarrow{n_j}$ le normali di due GCs sulla sfera. Il vettore $\overrightarrow{u_{ij}}$ passante per le intersezioni I_{ij} e I'_{ij} delle due circonferenze si calcola attraverso il prodotto vettoriale $\overrightarrow{u_{ij}} = \overrightarrow{n_i} \times \overrightarrow{n_j}$. Considerando tutti i tracks si ottengono fasci di circonferenze che si intersecano in due punti opposti, che chiameremo Motion Vanishing Points (MVPs).

Il FOE dell'optical flow è l'MVP a cui puntano le freccie dei tracks.

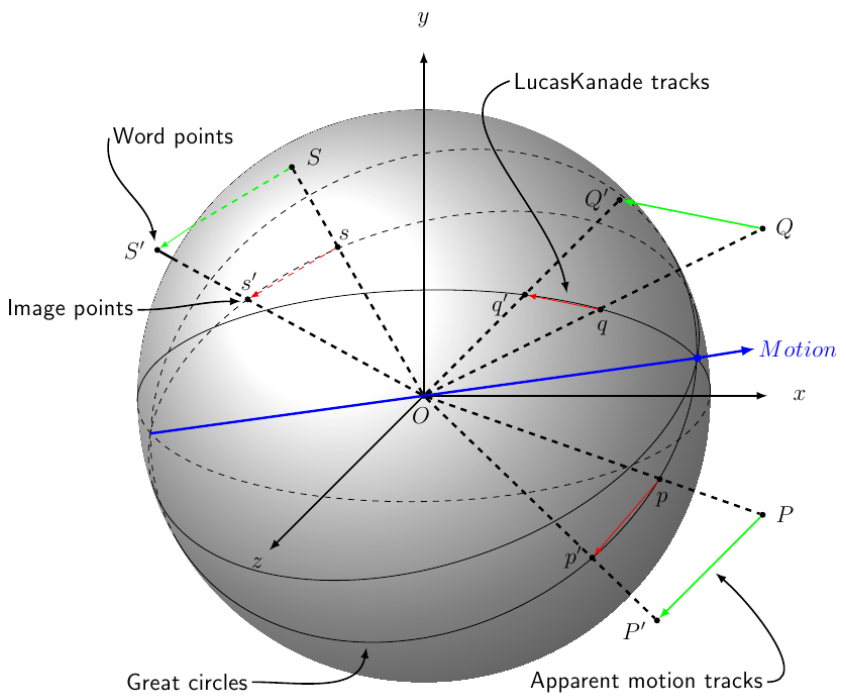


Figura 4.8. La direzione di due rette parallele è la stessa di quella della retta passante per i punti di intersezione tra le circonferenze corrispondenti

Come analizzato in [31] e in [26], se si ha una traslazione pura si avranno solo due MVPs antipodali: il focus of expansion (FOE) e il focus of contraction (FOC).

Se invece si tratta di una rototraslazione si otterranno quattro MVPs a due a due antipodali. Due degli MVPs non antipodali, separati da un angolo dovuto proprio alla rotazione, saranno il FOE e il FOC cercati, come indicato in figura 4.9.

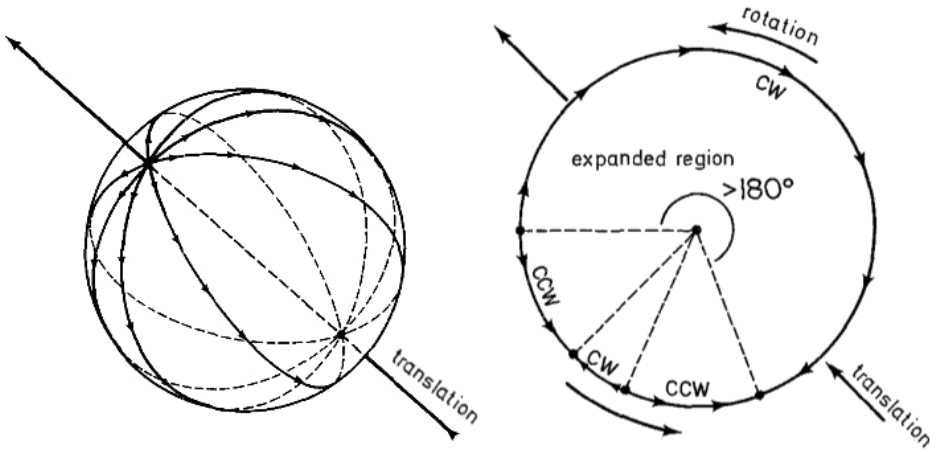


Figura 4.9. Traslazione pura e rototraslazione

A causa del rumore, dell'approssimazione, della differenza di risoluzione tra punti vicini e punti lontani, di possibili oggetti in movimento nella scena e di errori nel calcolo della corrispondenza tra features, alcuni gruppi di tracks possono produrre fasci di GCs che si intersecano in punti diversi. Per questo i punti antipodali di intersezione di ogni coppia di GCs vengono raggruppati in base alla loro posizione. Come eseguito in [17] si calcola il baricentro del cluster di MVPs quando la distanza dal baricentro precedente ed il nuovo punto è minore di una certa soglia, altrimenti si crea

un'altra lista con un nuovo baricentro.
Il baricentro che si riferisce al cluster con più elementi sarà l'MVP cercato.
Se si trovano due cluster non antipodali con numero di elementi simile si tratterà di una rototraslazione.
In presenza di rotazioni pure il FOE non ha significato, per questo vengono calcolati prima pitch, roll e yaw in base ai tracks:

- si calcolano gli angoli pitch, roll e yaw dalle proiezioni di ogni singolo track sui piani XZ, YX e XY rispettivamente.

- si calcola la direzione del singolo track (come segno dell'angolo), e se si rileva una direzione predominante (in base ad una soglia) su tutti i tracks significa che la telecamera sta ruotando in quella direzione

- si calcola l'angolo approsimato (vedi Capitolo 4.5) facendo una media dell'angolo prodotto dai tracks con direzione concorde.

Se non vengono rilevate direzioni predominanti significa che la telecamera sta tralsando o roto-traslando. Si procede quindi con il calcolo del baricentro e se si rilevano due MVPs non antipodali si calcola pitch, roll e yaw dall'angolo compreso tra i versori di movimento che puntano ai due MVPs.
Contemporaneamente) si calcola pitch, roll e yaw dall'angolo compreso tra i due versori di movimento determinati dai due MVPs non antipodali dovuti alla roto-traslazione (vedi figura 4.9). In questo modo si ottiene una stima dell'errore dovuto alla presenza di features vicine e può servire per migliorare la stima della rotazione in caso di rotazioni pure, sebbene

dipenda dalle features presenti al momento della roto-traslazione.
Oltre alla stima dell'assetto si riesce in questo modo ad avere anche una misura qualitativa dello spostamento del robot (versore di movimento) che può essere usata per migliorare anche la visual odometry.

4.4.7 GUI

Il software ha una GUI basata su QT [37] e OpenGL [35] per la visualizzazione della sfera unitaria e dell'MVP calcolato.

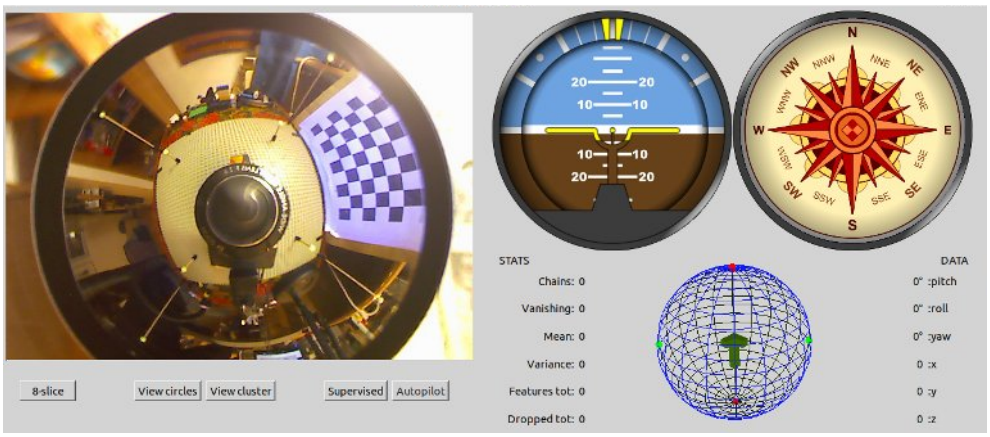


Figura 4.10. GUI

É prevista anche una funzione di debug per visualizzare sulla sfera unitaria le features e le circonferenze corrispondenti. Inoltre si possono generare e visualizzare con gnuplot file dati contenenti i cluster di MVPs, i tracks e un path di movimento generato dai MVPs rilevati durante lo spostamento della telecamera.

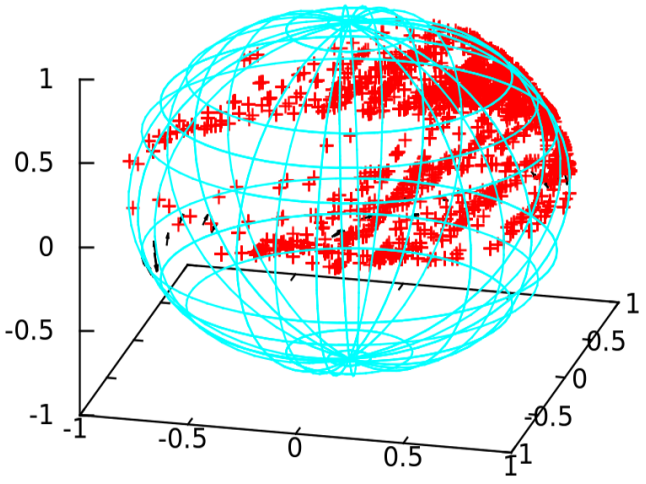


Figura 4.11. Cluster di MVPs in ambiente non strutturato

4.4.8 Ottimizzazione

L'utilizzo delle OpenCV richiede particolare attenzione al *memory leckage*, soprattutto se si tratta di applicazioni real-time che fanno uso del flusso video.

Per questo motivo è stato utilizzato il software **Valgrind** che permette un'analisi dell'utilizzo di risorse da parte del software. In questo modo si può evitare che il software accumuli sprechi di memoria che lo porterebbero a terminarla dopo aver eseguito un numero anche cospiquo di calcoli.

Mantenere un alto framerate è l'altro vincolo importante per il buon funzionamento dell'apparato.

Oltre alle considerazioni già fatte nel paragrafo 4.4.1 sull'esposizione automatica, è stato necessario scegliere gli algoritmi anche in base alle loro prestazioni.

Nello specifico si è pensato di optare per un clustering semplice come K-Means per poterlo effettuare on-line, durante l'inserimento di nuovi elementi del cluster, come indicato anche in [22] per il vanishing point detection.

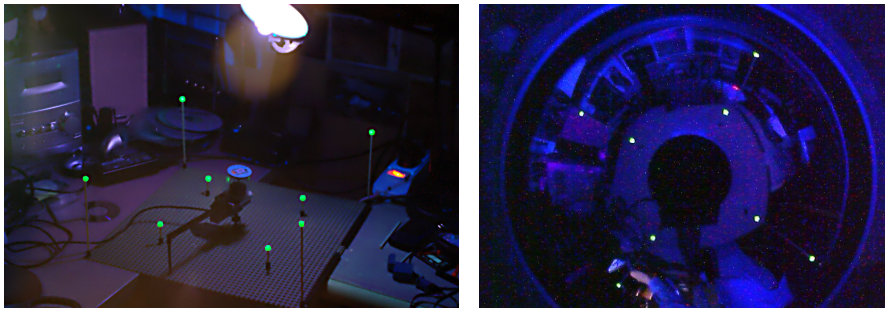
4.5 I risultati

4.5.1 Framework

Per studiare le disposizioni dei cluster di MPVs si è introdotta la possibilità di sostituire l'estrazione automatica delle features con un tool di selezione supervisionata che cerca le features in un intorno del punto cliccato dall'utente. Il framework è composto da una serie di punti luminosi nello spazio a diverse distanze e altezze come visibile in figura 4.12.

I punti sono dei marker passivi di materiale fluorescente, eccitato da una lampada UV posta sopra il framework.

Per traslare la telecamera si è scelto di far traslare il framework mantenendo la telecamere fissa su una staffa per poter ridurre l'errore di misura.



(a) (b)

Figura 4.12. Per massimizzare il contrasto si sono utilizzati marker passivi fluorescenti. (a) Fotografia UV (b) Vista dalla telecamera omnidirezionale

Per ruotare la telecamera è stato usato un servomotore *Hitec HS422* pilotato dalla scheda *Adafruit Motor/Stepper/Servo Shield* per Arduino [3]. Tramite il software viene inviato il valore in gradi dell'angolo di cui si vuol far ruotare la telecamera. Per questo si è resa necessaria una caratterizzazione dell'errore del servomotore (Figura 4.13) e solo dopo una sua

compensazione si è potuto valutare correttamente l'errore di rotazione.

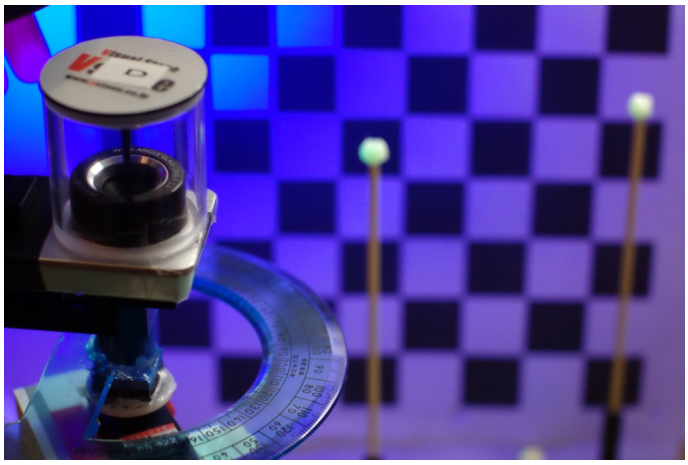


Figura 4.13. Stima dell'errore del servomotore e della telecamera

4.5.2 Traslazione

Per studiare il comportamento nella stima dell'assetto si sono utilizzati 8 markers disposti al centro delle relative sezioni in cui viene suddivisa l'immagine (vedi figura 4.5). Lavorando su una sfera la traslazione della telecamera in una qualunque direzione produrrà gli stessi effetti sull'optical flow, quindi per semplicità viene considerata la traslazione lungo l'asse X che coincide con l'asse sagittale del robot su cui è posta la telecamera.

Dopo 100 test effettuati traslando la telecamera di $10cm$ lungo X è emerso che l'asse effettivo è leggermente scostato rispetto a quello indicato dopo la calibrazione (figura 4.15).
Si è calcolato il baricentro Q del cluster delle intersezioni tra il versore movimento, che dovrebbe coincidere con l'asse X, e la sfera unitaria (ovvero

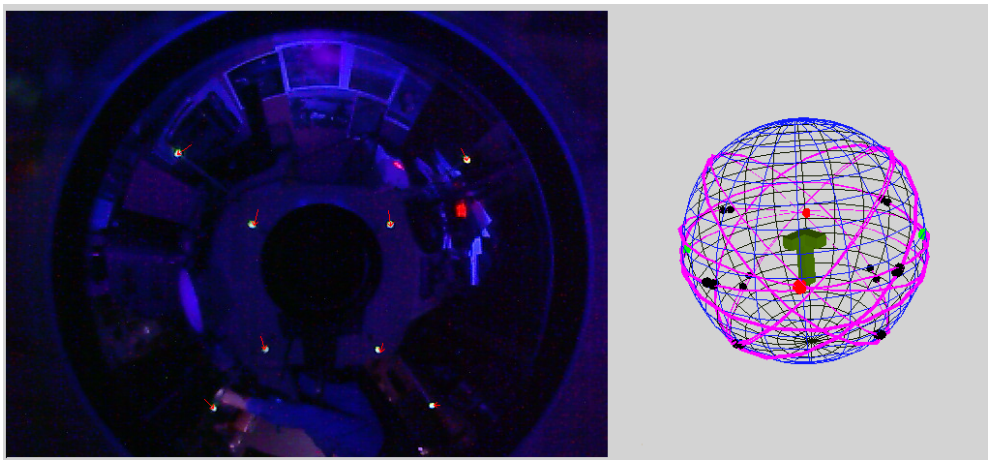


Figura 4.14. Disposizione dei tracks e delle GCs in caso di traslazione

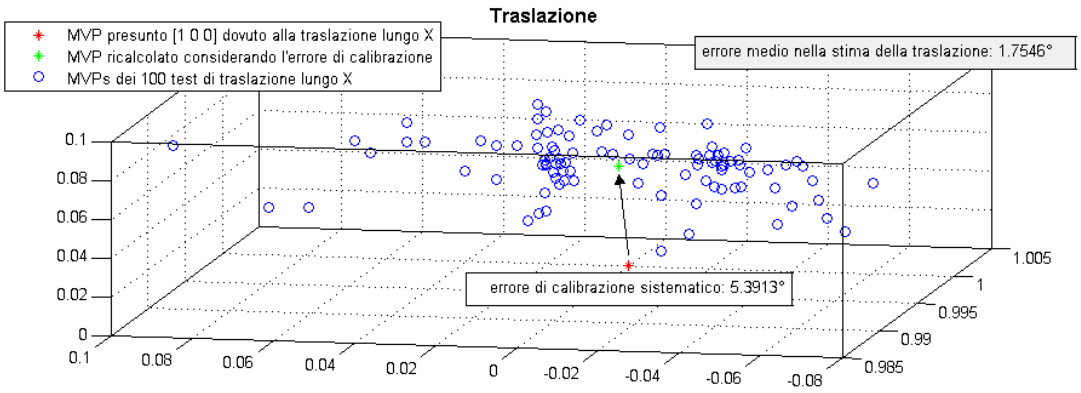


Figura 4.15. Errore nella stima della traslazione lungo X in ambiente strutturato

il FOE). Lo scostamento tra questo baricentro e il punto $P = (1, 0, 0)$ è un errore sistematico dovuto alla mappatura dell'immagine nella sfera unitaria.

Trattandosi di un errore sistematico è stato possibile compensarlo durante il calcolo del versore di traslazione.

La mancanza di informazioni sulla profondità impedisce una stima quantitativa del path di movimento, ma permette ugualmente di rilevare direzione e verso dello spostamento.

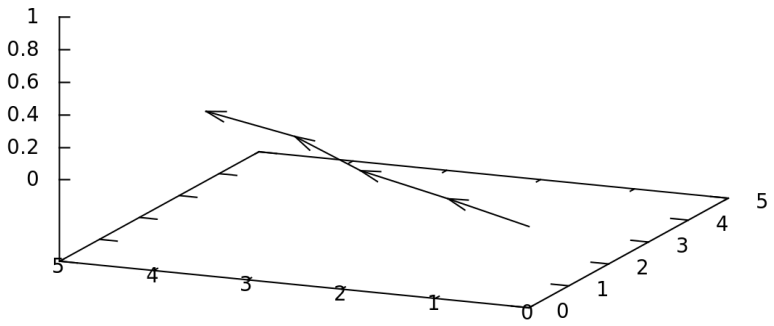


Figura 4.16. Path di movimento

4.5.3 Rotazione

Anche nel caso della rotazione si può sfruttare la mappatura sferica e considerare la rotazione lungo un unico asse.

In questo caso l'ambiente strutturato è formato da 4 markers giacenti sul piano ortogonale all'asse di rotazione in modo da valutare il drift dovuto all'aggiornamento dell'immagine di riferimento.

Bisogna considerare che ogni volta che la lunghezza media dei tracks supera il valore prestabilito l'immagine di riferimento viene aggiornata.

Questo aggiornamento durante rotazioni continue produce un errore sistematico che raddoppia se la rotazione ha un cambio di direzione in quanto si inizia a misurare la rotazione dopo che i tracks assumono un valore maggiore di una soglia minima.

Questo errore è stato misurato dopo una serie di test in ambiente strutturato e la sua correzione è stata verificata anche in ambiente non strutturato

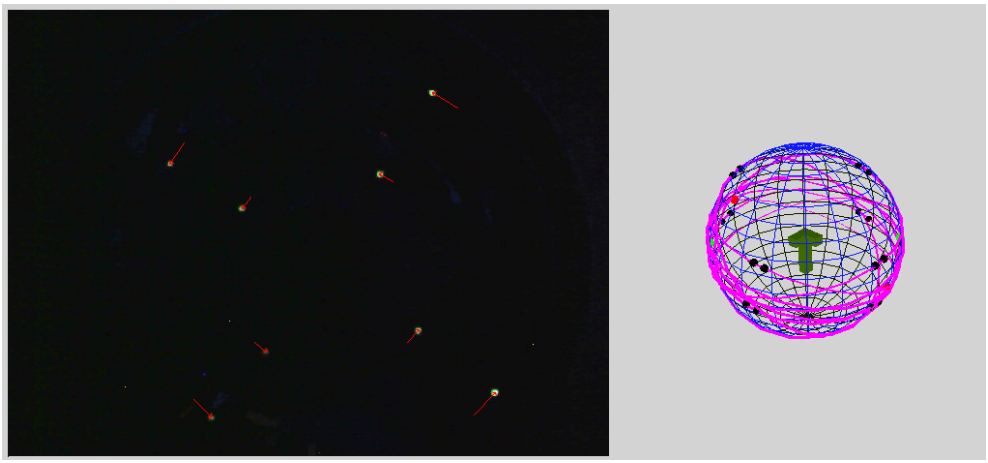


Figura 4.17. Disposizione dei tracks e delle GCs in caso di rotazione

(figura 4.18).

Trattandosi di un errore sistematico è stato possibile compensarlo durante

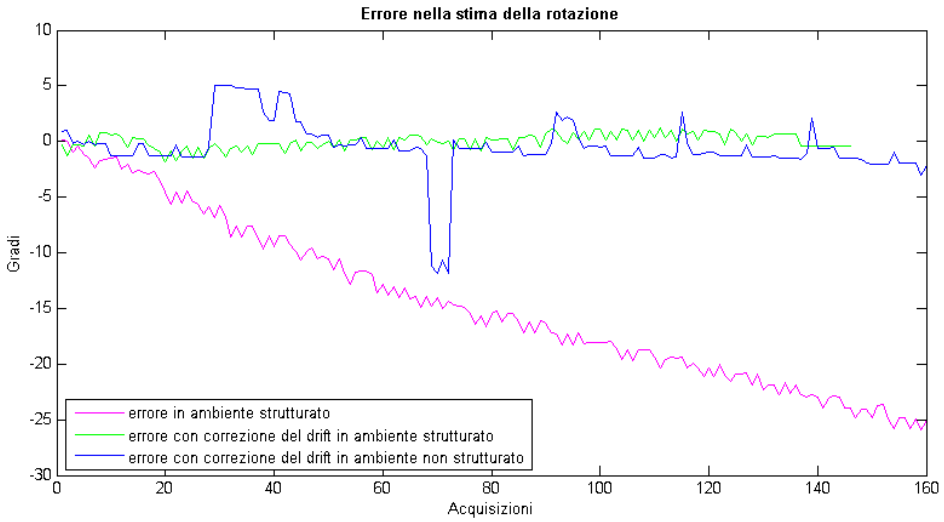


Figura 4.18. Errore di rotazione

il calcolo della rotazione.

4.5.4 Roto-Traslazione

Per avere una stima dell'errore nel calcolo della rototraslazione si è utilizzato lo stesso framework usato per la rotazione, decentrando però la telecamera rispetto al servomotore, come visibile in figura 4.19.

Il calcolo dell'errore è stato possibile grazie al fatto che l'angolo di rotazione

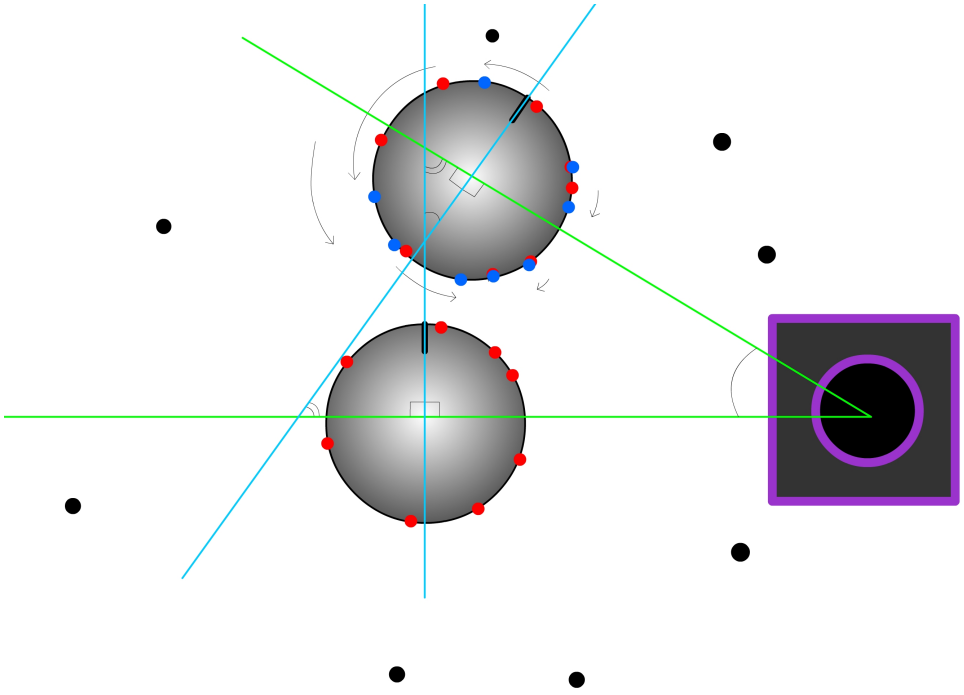


Figura 4.19. Configurazione del framework per la rototraslazione

del servomotore corrisponde all'angolo di rotazione della telecamera.

In questo caso l'errore è maggiore perchè il calcolo dell'angolo è basato sull'angolo tra i due MVPs non antipodali, rilevati attraverso il clustering e non calcolato direttamente dai tracks.

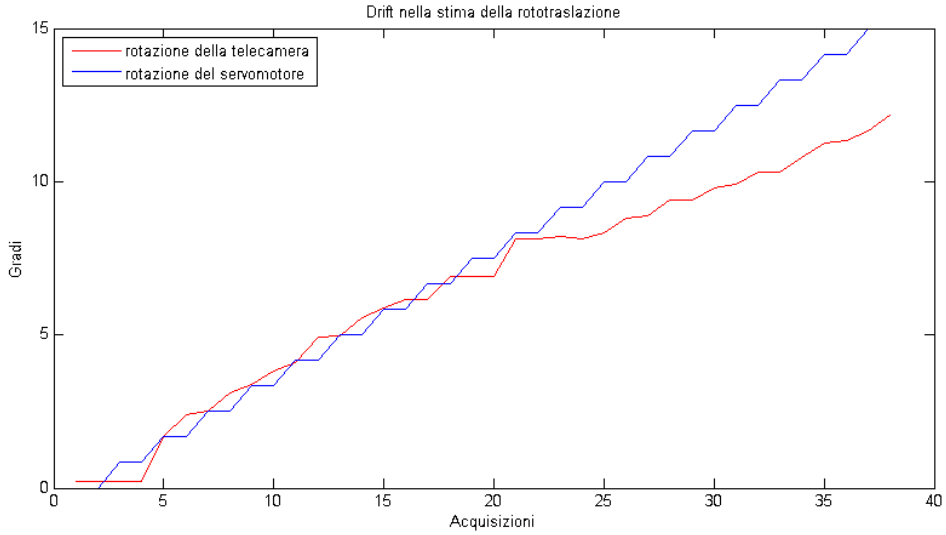


Figura 4.20. Errore di rototraslazione

Capitolo 5

Sviluppi futuri

Per migliorare la robustezza è necessario un tuning dei parametri in ambienti diversi per trovare la configurazione migliore.
Resta aperto il problema del motion blur che invece può essere addirittura sfruttato per migliorare la robustezza del calcolo della rotazione utilizzando i risultati di [27].

Studiando i risultati sperimentali riguardo al cluster di MVPs si è notata una corrispondenza con la Bingham distribution, come analizzato anche in [13], e non si esclude un possibile miglioramento nel clustering dei vanishing points e nel disaccoppiamento della rototraslazione, valutando il carico computazionale di un'implementazione di [43] per stimarne i coefficienti.
Inoltre, data la necessità di lavorare in real-time, si potranno aumentare le performance parallelizzando l'algoritmo, sfruttando la suddivisione in slice dell'immagine.

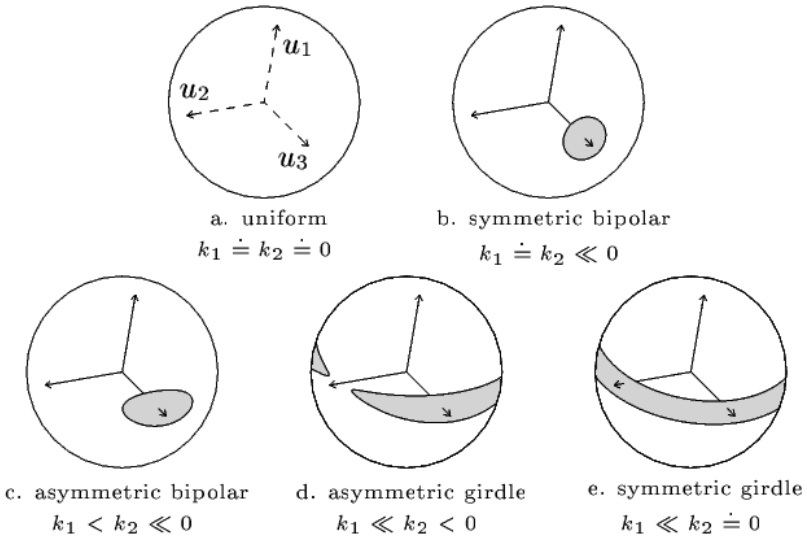


Figura 5.1. Possibili cluster di MVPs.

Ringraziamenti

Ringrazio il mio più caro amico Alberto Busetto (PHd student at ETH Zurich, Department of Computer Science) per avermi aiutato ed indirizzato più volte verso la strada giusta.

Impostazioni webcam

Comando che stampa la lista dei settings della telecamera:

```
uvcdynctrl -c -d /dev/video1 -v
```

I campi utili sono:

Exposure, Auto

ID :	0x0000000f,
Type :	Dword,
Flags :	{ CAN_READ, CAN_WRITE },
Values :	{ 'Auto Mode'[0], 'Manual Mode'[1], 'Shutter Priority Mode'[2], 'Aperture Priority Mode'[3] } ,
Default :	3

Exposure (Absolute)

ID :	0x00000011,
Type :	Dword,
Flags :	{ CAN_READ, CAN_WRITE },
Values :	[0 .. 5000, step size: 1],
Default :	1

Comando per disabilitare l'esposizione automatica:

```
uvcdynctrl -d /dev/video1 -s 'Exposure, Auto' -- 1
```

Comando per impostare l'esposizione in modo che il collo di bottiglia sia l'elaborazione da parte del software

```
uvcdynctrl -d /dev/video1 -s 'Exposure (Absolute)' -- 600
```

Bibliografia

[1] H.H. Abdelkader et al. "2 1/2 D visual servoing with central catadioptric cameras". In: *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*. 2005, pp. 3572–3577. DOI: 10.1109/IROS.2005.1545327.

[2] Matthew E. Antone. "Robust camera pose recovery using stochastic geometry". AAI0802578. Tesi di dott. 2001.

[3] *Arduino*. 2011. URL: http://www.arduino.cc/.

[4] S. Baker e S.K. Nayar. "A theory of catadioptric image formation". In: *Computer Vision, 1998. Sixth International Conference on*. 1998, pp. 35 –42. DOI: 10.1109/ICCV.1998.710698.

[5] Simon Baker e Shree K. Nayar. "Single Viewpoint Catadioptric Cameras". In: *In Panoramic Imaging: Sensors, Theory, and Applications*. 2001, pp. 1–44.

[6] D. H. Ballard e C. M. Brown. *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[7] J.P. Barreto e H. Araujo. "Geometric properties of central catadioptric line images and their application in calibration". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.8 (2005), pp. 1327 –1333. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2005.163.

[8] S. S. Beauchemin e J. L. Barron. "The computation of optical flow". In: *ACM Comput. Surv.* 27.3 (set. 1995), pp. 433–466. ISSN: 0360-0300. DOI: 10.1145/212094.212141.

[9] R. Bonetto. "Applicativo open source per il controllo del robot RB2000". Università degli studi di Padova, 2009.

[10] Jean-Yves Bouguet. *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm*. 2000. URL: http://robots.stanford.edu/cs223b04/algo_tracking.pdf.

[11] Adrian Bradski. *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees]*. 1. ed. Gary Bradski and Adrian Kaehler. O'Reilly Media, 2008. ISBN: 0-596-51613-4.

[12] A. Casasola. "Realizzazione di un traduttore da un linguaggio pseudo-naturale ad un file di comandi per il robot umanoide Robovie-X". Università degli studi di Padova, 2010.

[13] R.T. Collins e R.S. Weiss. "Vanishing point calculation as a statistical inference on the unit sphere". In: *Computer Vision, 1990. Proceedings, Third International Conference on*. 1990, pp. 400 –403. DOI: 10.1109/ICCV.1990.139560.

[14] Peter Corke e Robert Mahony. "Sensing and control on the sphere". In: *14th International Symposium on Robotics Research (ISRR 2009)*. A cura di Cedric Pradalier, Roland Siegwart e Gerhard Hirzinger. Lucerne, Switzerland: Springer-Verlag Berlin Heidelberg, 2009, pp. 71–85. URL: http://eprints.qut.edu.au/33737/.

[15] C. Demonceaux, P. Vasseur e C. Pegard. "UAV Attitude Computation by Omnidirectional Vision in Urban Environment". In: *Robotics and Automation, 2007 IEEE International Conference on*. 2007, pp. 2017 –2022. DOI: 10.1109/ROBOT.2007.363618.

[16] Matteo Finotto. "Visione Omnidirezionale su Robot Umanoide". Università degli studi di Padova, 2009.

[17] Matteo Finotto e Emanuele Menegatti. "Humanoid Gait Stabilization based on Omnidirectional Visual Gyroscope". In: *Department of Information Engineering (DEI)* (2009).

[18] Christopher Geyer e Kostas Daniilidis. "Catadioptric Projective Geometry". In: *International Journal of Computer Vision*. Vol. 45. 3. 2001, pp. 223–243.

[19] Joshua Gluckman e Shree K. Nayar. *Ego-Motion and Omnidirectional Cameras*.

[20] K. Ueda H. Ishiguro e S. Tsuji. "Omnidirectional visual information for navigating a mobile robot". In: *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. 1993, 799 – 804 vol.1. DOI: 10.1109/ROBOT.1993.292075.

[21] Richard Hartley e Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2ª ed. New York, NY, USA: Cambridge University Press, 2003. ISBN: 0521540518.

[22] Qiang He e Chee-Hung Henry Chu. *An Efficient Vanishing Point Detection by Clustering on the Normalized Unit Sphere.*

[23] Berthold K.P. Horn e Brian G. Schunck. *Determining Optical Flow.* Rapp. tecn. Cambridge, MA, USA, 1980.

[24] *Hyperbolic Mirror, V-Stone.* 2008. URL: `http://www.vstone.co.jp/english/products/sensor_camera/`.

[25] C. Demonceaux J.C. Bazin Inso Kweon e P. Vasseur. "Rectangle Extraction in Catadioptric Images". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* 2007, pp. 1 –7. DOI: `10.1109/ICCV.2007.4409208`.

[26] Jongcheol Kim e Yasuo Suga. "An Omnidirectional Vision-Based Moving Obstacle Detection in Mobile Robot". In: *International Journal of Control, Automation, and Systems.* Vol. 6. Dic. 2007, pp. 663–673.

[27] Georg Klein e Tom Drummond. "A single-frame visual gyroscope". In: *Proc. British Machine Vision Conference (BMVC'05).* Vol. 2. BMVA. Oxford, 2005, pp. 529–538.

[28] Bruce D. Lucas e Takeo Kanade. "An iterative image registration technique with an application to stereo vision". In: *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2.* Vancouver, BC, Canada: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679. URL: `http://dl.acm.org/citation.cfm?id=1623264.1623280`.

[29] Sang Ly, C. Demonceaux e P. Vasseur. "Translation estimation for single viewpoint cameras using lines". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on.* 2010, pp. 1928–1933. DOI: `10.1109/ROBOT.2010.5509555`.

[30] Christopher MEI e Ezio MALIS. "Fast central catadioptric line extraction, estimation,tracking and structure from motion". In: *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems Beijing, China.* INRIA Icare Project-team, Sophia-Antipolis, France, 2006.

[31] R. C. Nelson e J. Aloimonos. "Finding motion parameters from spherical motion fields (or the advantages of having eyes in the back of your head)". In: *Biological Cybernetics* 58 (4 1988). 10.1007/BF00364131, pp. 261–273. ISSN: 0340-1200. URL: `http://dx.doi.org/10.1007/BF00364131`.

[32] *oCamCalib, tutorial e download.* 2007. URL: http://asl.epfl.ch/scaramuz/research/DavideScaramuzzafiles/Research/OcamCalibTutorial.htm.

[33] Renè Vidal Omid Shakernia e Shankar Sastry. *Omnidirectional Egomotion Estimation From Back-projection Flow.*

[34] *OpenCV, documentazione.* 2011. URL: http://www.cs.indiana.edu/cgi-pub/oleykin/website/OpenCVHelp/.

[35] *OpenGL, documentazione.* 2011. URL: http://www.opengl.org/documentation/.

[36] *Philips.* 207. URL: http://www.philips.it/.

[37] *QT, documentazione.* 2011. URL: http://doc.qt.nokia.com/.

[38] *Robovie-X, V-Stone.* 2008. URL: http://www.vstone.co.jp/english/products/robovie_x/.

[39] Jianbo Shi e C. Tomasi. "Good features to track". In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on.* 1994, pp. 593 –600. DOI: 10.1109/CVPR.1994.323794.

[40] Tomas Svoboda, Tomas Pajdla e Vaclav Hlavac. "Motion Estimation using Central Panoramic Cameras". In: *IEEE Conf. on Intelligent Vehicles.* 1998, pp. 335–340.

[41] C. Tavian. "Algoritmi di stabilizzazione della camminata di robot umanoide". Università degli studi di Padova, 2011.

[42] Matthew E. Antone Seth Teller. *Automatic Recovery of Relative Camera Rotations for Urban.* 2000.

[43] Jhon T.Kent. "Asymptotic Expansion for the Bingham distribution". In: *Appl. Statist.* Vol. 36. University of Leeds, UK. 1987, pp. 139–144.

[44] Pascal Vasseur e El Mustapha Mouaddib. *Central Catadioptric Line Detection.* 1999.

[45] H. Haj Abdelkader Y. Mezouar e P. Martinet. "Central Catadioptric Visual Servoing From 3D Straight Lines". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems September 28 -October 2,2004, Sendai. Japan.* France, 2004.