UNIVERSITÀ DEGLI STUDI DI PARMA

FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

REALIZZAZIONE DI UN SISTEMA STEREOSCOPICO IBRIDO OMNIDIREZIONALE/PIN-HOLE E TECNICHE DI VISIONE ARTIFICIALE PER LA ROBOTICA AUTONOMA

RELATORE:

Chiar.mo Prof. Ing. Giovanni Adorni

CORRELATORI:

Dott. Gerhard Kraetzschmar
Dott. Ing. Monica Mordonini

TESI DI LAUREA DI:

Marcello Carletti

ANNO ACCADEMICO 2001/2002

Indice

Indice delle figure

Introduzione

Gli apparati di visione sono, probabilmente, i sensori che offrono il miglior compromesso tra costo, flessibilità e ricchezza di informazioni. Le crescenti prestazioni dei calcolatori elettronici, al pari dei decrescenti costi delle apparecchiature video, permettono oggigiorno lo sviluppo di robot che si basano per lo più sulla visione per compiti di navigazione e di auto-localizzazione in ambienti semi-strutturati e non strutturati, anche se la fusione con dati forniti da altri tipi di sensori (come ultrasuoni, infrarossi, laser range-finder, …) può essere utile, o addirittura necessaria, in molte applicazioni pratiche.

Siccome i robot operano in ambienti sempre meno strutturati, la necessità di sistemi di visione che forniscano una descrizione globale, anche se talvolta approssimativa, dell'ambiente circostante è in crescita. Ciò è particolarmente vero quando i robot operano in ambienti dinamici e rapidamente variabili. Queste considerazioni hanno stimolato un crescente interesse in sistemi di visione omnidirezionale basati, di norma, su un sensore catadiottrico, composto da una telecamera rivolta verso l'alto che acquisisce l'immagine riflessa da uno specchio convesso appeso sopra di essa. La visione omnidirezionale sta diventando sempre più popolare nelle applicazioni in cui un robot autonomo deve reagire a stimoli visivi che possono giungere da qualsiasi direzione in ogni momento della sua attività, o per pianificare il suo comportamento in funzione di essi. Questi ed altri ambiti traggono vantaggio dall'ampiezza dei campi visivi generabili tramite catadiottro e dalla compatta organizzazione delle informazioni (seppur caratterizzata da una risoluzione limitata). Fornendo il più ampio campo visivo possibile, i sistemi di visione

omnidirezionale possono, inoltre, ovviare alla necessità di telecamere “attive”, che richiedono complesse strategie di controllo.

Questa tesi affronta diversi aspetti di un progetto di visione artificiale applicata alla robotica, centrato sull’impiego di un sensore visivo di tipo HOPS (*Hybrid Omnidirectional/Pin-hole Sensor*), che combina ad un catadiottro per la visione omnidirezionale una tradizionale telecamera a CCD posta frontalmente, con la quale condivide parte del campo visivo fornendo informazioni stereoscopiche. Questo permette di rafforzare le potenzialità del sensore con informazioni più dettagliate riguardanti una specifica regione spaziale, rendendo quindi più affidabile la ricostruzione dell’ambiente circostante.

Questo lavoro di tesi è stato svolto principalmente presso il Dipartimento di Elaborazione di Informazioni Neurali dell’Università di Ulm, in Germania [25], e poi terminato presso il Dipartimento di Ingegneria dell’Informazione dell’Università di Parma.

L’obiettivo finale è la creazione di un sistema di gestione per robot mobili che si avvalga, nel campo sensoriale, di un modulo di visione capace di sfruttare tutte le potenzialità del sensore HOPS. Oltre alla progettazione del sensore omnidirezionale, ciò riguarda lo studio e la realizzazione di un modulo software di gestione di un sistema di visione binoculare per robot mobili in tutti i suoi principali aspetti: a partire dalla calibrazione dei singoli sensori ottici, passando per la calibrazione stereo del sistema, per giungere a strumenti di analisi ed interpretazione delle immagini e ad un sottomodulo di comunicazione dei risultati ottenuti. Il modulo di visione può essere, infatti, visto come uno dei componenti del robot stesso in grado di comunicare (ad esempio ad un modulo di pianificazione ed azione) i propri risultati.

Durante la fase di progetto, si è seguito come principio fondamentale quello della generalità, ossia si è voluto creare un modulo in grado di adattarsi facilmente a diverse applicazioni e a diversi sensori visivi, dunque non solo ad uno specifico sistema binoculare e ad una specifica applicazione, anche se ha trovato il suo principale impiego nell’utilizzo con HOPS.

Il lavoro svolto può essere ricondotto a due diversi obiettivi di ricerca: una prima parte riguarda la messa a punto di tecniche per la calibrazione di sensori ottici,

con particolare riferimento al visore catadiottrico impiegato (capitolo 3); una seconda parte riguarda l'analisi delle immagini acquisite, ponendosi l'obiettivo generale di elaborare le informazioni stereoscopiche per rilevare la presenza di eventuali ostacoli al movimento del robot in ambienti semi-strutturati e non strutturati (capitolo 4). Questa seconda parte utilizza il modello dell'inversione prospettica, ricavando informazione sulla presenza di ostacoli dall'analisi delle differenze tra le immagini generate per inversione prospettica a partire da quelle acquisite dalle due telecamere.

In relazione a queste diverse parti della ricerca sono stati progettati e realizzati due differenti moduli software:

- il modulo per la calibrazione, che raccoglie l'insieme degli algoritmi sperimentati, ne permette l'applicazione a diversi tipi di sensori e situazioni, ed è stato strutturato per facilitare l'integrazione di nuovi algoritmi;
- il software per il controllo della visione binoculare, che risulta indipendente dalla calibrazione e dai tipi di sensore, e che fornisce delle routines generiche per l'identificazione e l'analisi di ostacoli e per l'estrazione di informazioni di basso livello, permettendo una facile integrazione di routines più specifiche per singole applicazioni. L'architettura realizzata propone, quindi, alcuni aspetti computazionali tipici dell'approccio gerarchico ed altri aspetti operativi più legati invece al paradigma delle *visual routines*.

Come ultimo aspetto, viene analizzata l'integrazione dei moduli software in due diverse architetture robotiche (capitolo 5), con riferimento, dal lato hardware, alla gestione di frame-grabber ed attuatori differenti e, dal lato software, ad ambienti di sviluppo di concezione profondamente diversa.

Allo scopo di facilitare l'integrazione di nuovi algoritmi nell'architettura dei moduli sopra descritta, si è deciso di progettare e realizzare il software con tecniche e strumenti orientati agli oggetti.

Capitolo 1

La visione artificiale

1.1 Generalità

L'apparato visivo è, senza dubbio, il più potente tra gli apparati sensoriali di cui l'evoluzione ha dotato la specie umana: fornisce una vasta quantità di informazioni sull'ambiente circostante e ci permette di interagire con esso in maniera intelligente, senza bisogno di un contatto fisico. Allo stesso tempo, però, la visione è un processo complicato e le nostre conoscenze sulla biologia di questo processo sono ancora limitate. Da una parte, quindi, non dobbiamo stupirci dell'estrema attenzione e degli sforzi che si sono fatti per dare alle macchine la capacità di vedere ma, dall'altra, dobbiamo anche prendere atto delle grandi difficoltà incontrate. La visione artificiale è un settore di ricerca ormai particolarmente attivo da oltre trent'anni. Un periodo, questo, nel quale si sono avvicendate molte teorie, si sono confrontati paradigmi alternativi e sono state sviluppate tecniche diverse. In alcuni settori di ricerca, come l'*image processing* e il *pattern recognition*, questo ha portato ad un rapido sviluppo ed alla creazione di molteplici sistemi utilizzabili nella pratica. Ma in altri settori, in particolare quelli che comportano operazioni di più alto livello, i progressi sono stati più limitati.

Si può dire che "*la visione artificiale si pone l'obiettivo di automatizzare ed integrare una vasta gamma di processi e rappresentazioni tipiche della percezione visiva*" [1], che è un processo che mette in relazione l'input visuale con modelli del mondo preesistenti. Data l'estrema diversità tra le immagini percepite e i modelli che ne descrivono ed astraggono le informazioni, i sistemi di visione artificiale sono, normalmente, costituiti da una gerarchia di livelli rappresentativi intermedi che

richiedono altrettanti processi per la loro interconnessione. Quindi, la visione artificiale è strettamente legata sia a problematiche di analisi di basso livello, dette anche di *early-processing*, sia a problematiche di più alto livello, che si possono definire *cognitive*.

I processi che costituiscono le fondamenta della percezione visiva, quelli di basso livello, forniscono la sensibilità a caratteristiche quali la luminosità, il colore, la distanza, o la percezione di oggetti e la distinzione tra terreno ed ostacoli. Queste sono capacità che i sistemi visivi biologici hanno sviluppato nel corso di milioni di anni di evoluzione e i cui meccanismi, come già osservato, risultano essere ancora parzialmente inaccessibili. La necessità da parte dei sistemi di visione artificiale di emulare queste capacità, richiede di inventare tecniche e strumenti per l'estrazione delle informazioni elementari. Questi processi di estrazione rappresentano le basi per elaborazioni di più alto livello, come la modellizzazione geometrica dell'ambiente circostante e la pianificazione, che hanno invece l'obiettivo di interpretare la scena circostante sulla base di un modello del mondo più o meno strutturato.

1.1.1 Il processo di formazione dell'immagine

Il processo di formazione dell'immagine di una scena su una superficie implica che, tra tutti i raggi luminosi riflessi o emessi dagli oggetti presenti nella scena, sia possibile eseguire una selezione in modo che, idealmente, la luce proiettata su diversi punti della superficie di formazione dell'immagine provenga da diversi punti dello spazio della scena e, viceversa, diversi punti dello spazio proiettino luce su diversi punti della superficie di formazione. Lo scopo del processo è, quindi, produrre sulla superficie un'immagine che sia una rappresentazione sufficientemente buona della scena. I fenomeni e le problematiche che intervengono in questo processo sono molteplici e hanno portato, nel corso della storia, a molteplici studi sia scientifici che tecnologici.

Per ottenere questo risultato di selezione, il sistema di filtraggio ideale, dal punto di vista della risoluzione, è lo stenoscopio, ossia una camera oscura dotata di un piccolo foro su un lato che determina la formazione dell'immagine sul lato opposto. Questo foro lascia entrare nella scatola, tra tutti i raggi luminosi propagati

da ogni singolo punto dello spazio oggetto (ossia lo spazio tra il dispositivo di filtraggio e la scena stessa), solo quelli che sono direzionati attraverso il foro.

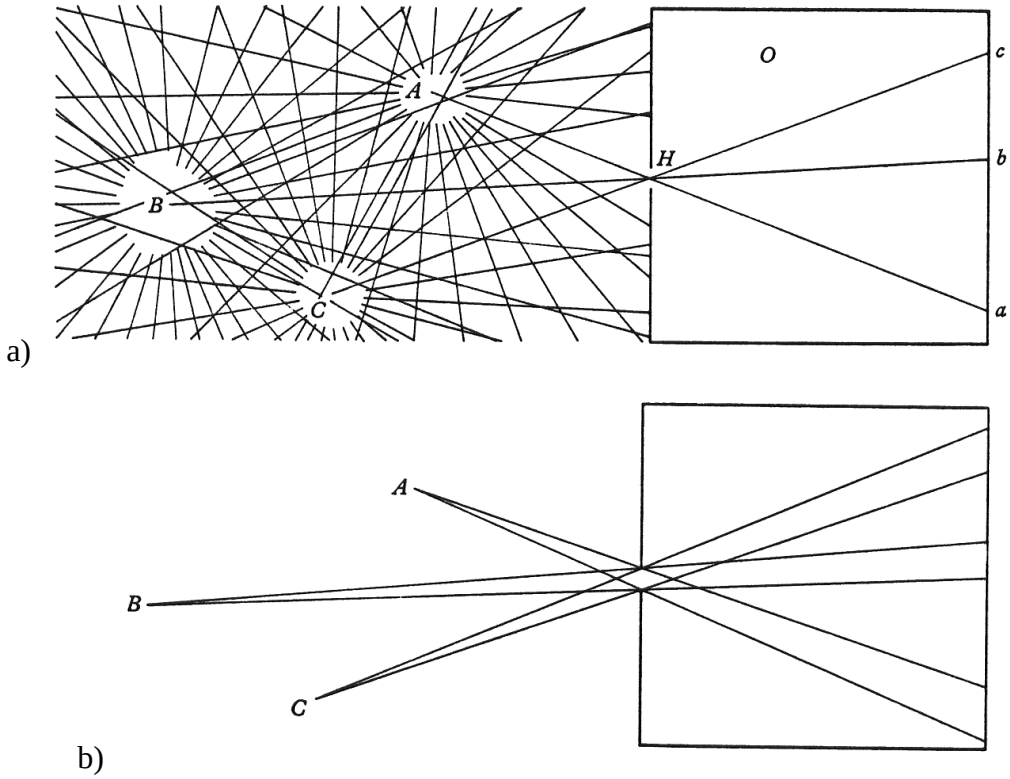


Fig. 1.1 Lo stenoscopio. Mentre ogni punto luminoso invia raggi divergenti nei dintorni (a)), lo stenoscopio seleziona certi coni di raggi (b)) tra tutti quelli che riempiono l'intero spazio, cosicché i raggi principali (AH, BH, CH in a)) di questi coni convergono verso il centro del foro stenopeico.

Quindi, ad ogni punto dello spazio oggetto corrisponde nello spazio immagine (ossia lo spazio tra il filtro e la superficie di formazione dell'immagine) un cono di luce, e quindi una piccola ellisse luminosa, detta "*disco di Airy*", sulla superficie come contributo all'immagine. In altre parole, il disco di Airy è la proiezione del foro sulla superficie, nella direzione della retta congiungente il centro del foro ad un punto dello spazio. La somma di tutte le ellissi prodotte da tutti i punti dello spazio genera l'immagine. Tanto più piccolo è il foro, tanto più alta sarà la risoluzione dell'immagine: infatti, dati due punti dello spazio molto vicini, i due dischi di Airy da essi prodotti avranno una percentuale di sovrapposizione delle aree tanto minore quanto minore sarà il diametro del foro. E' proprio in questi termini che si può misurare la risoluzione dell'immagine prodotta. Il principale limite alla risoluzione

dell'immagine è dovuto al fenomeno della diffrazione, che impedisce di ridurre oltre un certo limite ben preciso (pari ad un diametro di 0,38mm) le dimensioni del foro. Oltre questo limite, infatti, la diffrazione determina un deterioramento dell'immagine, sfocandola.

Il principale problema pratico dello stenoscopio, che lo rende inutilizzabile in casi reali, è il fatto che, tanto minore è la dimensione del foro, tanto minore è la quantità di luce che lo può attraversare per unità di tempo: per operare l'acquisizione di un'immagine in tempi brevi risulta necessario avere fori di dimensione elevata a scapito, quindi, della risoluzione. Da qui nasce l'esigenza, per recuperare la risoluzione persa, di sostituire il foro con un altro tipo di filtro, una lente, in grado di concentrare tutta la luce proveniente da un'area più ampia, focalizzandola sulla superficie di formazione dell'immagine. Questo vantaggio viene, però, pagato in termini di una proiezione non più ideale sulla superficie. L'ottica di tutte le telecamere, infatti, produce una distorsione delle immagini, generalmente distinta in una componente radiale e una tangenziale, di solito trascurabile.

1.2 La visione artificiale e la robotica

La visione artificiale trova nella robotica una delle sue più importanti fonti di applicazione al mondo reale. Se, come definita da Brady [2], "*la robotica è la connessione intelligente tra percezione e azione*", o come dice Arkin [3], "*un robot intelligente è una macchina capace di estrarre informazioni dal suo ambiente e usare la conoscenza sul suo mondo per muoversi in maniera sicura ed al fine di perseguire uno scopo*", indubbiamente una delle principali sorgenti di informazione sul mondo esterno, ossia uno degli strumenti di percezione migliori, non può che essere la visione.

L'applicazione delle tecniche di visione artificiale ad un ambito così fortemente a contatto con la complessità del mondo reale, quale la robotica, implica una serie di problematiche aggiuntive, legate all'integrazione del sistema visivo e delle informazioni da esso prodotte con le restanti componenti del sistema robotico (un modulo di ragionamento e di decisione, un modulo di implementazione delle decisioni tramite operazioni sul mondo esterno o sullo stato interno del sistema,

ecc.), nonché alla necessità di interagire con l'ambiente in tempo reale. Nel corso degli ultimi due decenni sono stati sperimentati innumerevoli approcci a queste problematiche, ma tutti possono essere ricondotti a tre diverse famiglie di soluzioni (e quindi anche a tre tipologie architetturali): l'*approccio gerarchico* (introdotto da Marr [4]), che può essere fatto risalire al paradigma della *percezione generalizzata*, la *visione attiva* (introdotta da Bajcsy [5]), più legata al paradigma della *percezione modulare*, e le *visual routines* (introdotte da Ullman [6]). In realtà, difficilmente si può ricondurre un sistema di visione artificiale reale ad uno solo di questi modelli, che rappresentano comunque importanti punti di riferimento teorici e metodologici per inquadrare il problema. Di seguito, sono analizzate brevemente le caratteristiche di questi tre approcci.

L'approccio gerarchico ha gettato le fondamenta di gran parte del lavoro svolto nel settore e ha influenzato anche gli altri due. È un approccio computazionale alla visione, in cui la percezione visiva è fondamentalmente un problema di elaborazione di informazione su più livelli gerarchici.

- L'input del sistema è l'immagine (o più immagini) intesa come matrice di valori (o insieme di matrici per le immagini a colori) di intensità luminosa e assunta come dato di partenza, la cui formazione è un processo indipendente dalle fasi successive.
- Un primo stadio elaborativo, che produce il cosiddetto *raw primal sketch*, è quello di estrazione di informazioni elementari dall'immagine: i contorni, le discontinuità di intensità, ecc..
- Il *raw primal sketch* contiene un'informazione ancora parziale, frammentaria, dipendente dal punto di vista della telecamera. Perciò, con varie tecniche legate al gradiente di luminosità (*shape from shading*, *shape from texture*, *shape from contour*, il metodo delle immagini intrinseche, che cerca di combinare le precedenti, ecc.) si cerca di completare questa informazione e, tramite la visione stereo, ottenerne di aggiuntiva sulla profondità spaziale.
- Si giunge, quindi, alla cosiddetta rappresentazione *2½D*, in cui il tipo di descrizione è sempre l'immagine ma, tramite la componente di profondità, si aggiunge informazione tridimensionale. Questa è, comunque, sempre legata al

punto di vista della telecamera: su essa non si è ancora giunti a introdurre concetti di regione, oggetto, parte.

- Infine, l'ultima fase di elaborazione, che porta alla rappresentazione *3D*, cerca di astrarsi dal punto di vista della telecamera e di interpretare la scena individuando oggetti, caratterizzandone la forma in termini di orientazione delle superfici elementari che li compongono e spostando il sistema di riferimento negli stessi oggetti individuati, al preciso scopo di renderne la rappresentazione indipendente dal punto di vista.

Quasi tutti i primi studi ed esperimenti riguardanti la percezione robotica si sono rifatti al paradigma della *percezione generalizzata*: il sistema visivo veniva considerato come un modulo a sé stante, indipendente dal resto, anche a causa delle difficoltà tecniche inizialmente incontrate ed il notevole peso elaborativo richiesto. Inoltre, per assicurare in ogni situazione l'acquisizione di sufficiente informazione, ci si poneva l'obiettivo di una ricostruzione tridimensionale completa della scena osservata. Questi due aspetti finivano per diventare limiti del sistema stesso: si determinava uno spreco di risorse elaborative e il sistema era troppo rigido per operare efficientemente in tempo reale.

Dalla rilevazione di questi limiti ebbe origine il secondo paradigma: quello di *percezione modulare*. In questo caso, il sistema sensoriale non è più una componente indipendente, ma interagisce e si mette al servizio degli altri moduli del robot. In pratica, di volta in volta, ricerca e rileva solamente le informazioni necessarie, senza andare oltre il bisogno di informazioni del robot.

Innumerevoli sono gli approcci nati a partire da questo paradigma di interazione sistema-percezione. Essi presentano diversi gradi di influenza dello stato interno e della conoscenza pregressa del robot sul processo visivo in quasi tutte le sue fasi. Tra gli esempi più importanti dobbiamo ricordare la *percezione attiva* (con controllo interattivo del robot sul processo percettivo), la percezione *action-oriented* (dove le ipotesi di azione e di goal influenzano il tipo di analisi e di elaborazione da eseguire), la percezione basata su aspettative e metodi *focus-of-attention* (dove le conoscenze immagazzinate sull'ambiente influenzano rispettivamente l'interpretazione e l'area di interesse delle immagini). Tutti questi approcci fanno

comunque riferimento alle due famiglie descritte sopra: la visione attiva e le visual routines.

La visione attiva introduce, rispetto al modello dell'approccio gerarchico, un importante grado di libertà: ogni livello elaborativo può influenzare i precedenti in una sorta di retroazione controllata dagli effettivi bisogni del sistema ed, in particolare, inserisce in questo loop anche la fase di acquisizione delle immagini, fase quindi non più passiva ma di ricerca attiva.

Infine, le *visual routines* costituiscono un approccio un po' più flessibile dei precedenti, che si fonda sulla realizzazione e l'impiego di un insieme di routines elementari che operano elaborazioni finalizzate ad un preciso obiettivo. Vi sono routines più generiche (dette *routines universali*), finalizzate ad una prima elaborazione delle immagini, cioè a far emergere le informazioni elementari, e quindi applicabili in molte situazioni; poi, vi è un insieme di routines più specifiche delle singole applicazioni, finalizzate ad elaborazioni generalmente di alto livello. Secondo quest'ultimo paradigma, quindi, ogni sistema visivo non può che essere dedicato a specifici compiti ed all'estrazione di informazioni legate alle applicazioni specifiche.

Oggi, la visione artificiale ha due principali aree di applicazione nella robotica: il riconoscimento e l'interpretazione degli oggetti presenti nella scena (al servizio di manipolatori e robot mobili) e la navigazione in ambienti indoor o outdoor (al servizio di veicoli, trasportatori, robot mobili). In molte di queste applicazioni, da un utilizzo combinato di sensori ottici e range-sensors, sempre più ricercatori stanno attualmente spostando la propria attenzione verso sistemi basati esclusivamente sulla visione. Oltre alla semplicità, flessibilità e relativa economicità di questi, l'abbandono di altri tipi di sensori quali laser e ultrasuoni, consente di superare le non trascurabili problematiche di confronto e fusione tra dati provenienti da sistemi sensoriali disomogenei.

1.3 Gli algoritmi di visione

In questo paragrafo vengono esaminati alcuni problemi di base relativi all'elaborazione delle immagini nell'ambito della robotica. Tra i processi visivi più comuni per un robot mobile ci sono la segmentazione dell'immagine, allo scopo di

riconoscere forme e possibili ostacoli, e l'individuazione dei contorni degli oggetti. Esistono varie tecniche, che nei prossimi paragrafi sono descritte sinteticamente per fornire una base teorica degli argomenti trattati in seguito.

1.3.1 Elaborazioni di basso, medio e alto livello

L'elaborazione delle immagini procede, di norma, per livelli. Quello più basso coinvolge i singoli pixel dell'immagine ed esegue operazioni elementari dalle quali nascono, in genere, altre immagini. In questo livello si eseguono solitamente operazioni puntuali o locali in cui ogni pixel viene considerato a sé stante. Nel livello intermedio si eseguono operazioni che considerano porzioni più ampie di immagini o l'immagine nel suo complesso, il cui risultato non è un'immagine elaborata, ma una struttura dati che rappresenta le caratteristiche di interesse della scena. Infine, nel livello superiore non si ha più nemmeno a che fare con immagini vere e proprie, ma con strutture dati che vengono elaborate al fine di descrivere la scena con un linguaggio formale.

La distinzione tra i livelli non è netta. Nella figura seguente si nota come, scendendo di livello, aumenta sia il carico per la CPU che l'ammontare di dati da elaborare. Inoltre, ai livelli più bassi è più alta la regolarità delle operazioni da compiere, nel senso che queste sono molto numerose ma ripetitive. Salendo di livello, invece, questa regolarità diminuisce e l'elaborazione si fa più “intelligente” e flessibile.

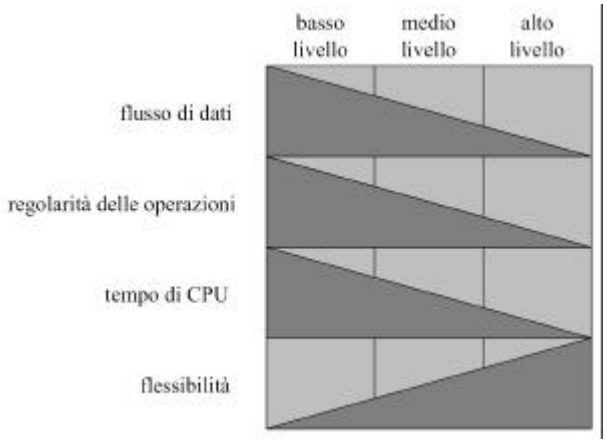


Fig. 1.2 Livelli di elaborazione.

1.3.2 Immagini digitali: risoluzione, densità e profondità

L'elaborazione visiva in un calcolatore avviene su immagini digitalizzate, ovvero su immagini che hanno subito un processo di campionamento e quantizzazione. Un'immagine naturale, quale quelle che percepisce l'occhio umano, può essere intesa come un segnale bidimensionale nel dominio dello spazio. Si supponga, per semplicità, di considerare solo la luminanza dell'immagine, così che questa appaia a toni di grigio, e possa essere intesa come un'applicazione $\Re^2 \rightarrow \Re$ che associa ad ogni punto dello spazio un valore di luminosità. Il processo di *campionamento* consiste nel prelevare i valori che la funzione assume sui nodi di un reticolo. Il numero di campioni presi nelle due dimensioni corrisponde alla *risoluzione* dell'immagine. La *densità* è, invece, il numero di campioni per unità di superficie e viene spesso indicata in *dpi* (Dots Per Inch). È evidente che maggiore è la risoluzione dell'immagine e maggiore è la sua qualità, cioè risulta più dettagliata. Questo è in accordo con il teorema del campionamento secondo cui, prendendo meno campioni, si riduce la massima frequenza (spaziale) rappresentabile: se un'immagine presenta componenti frequenziali alte, prendere un numero troppo basso di campioni comporta la perdita inevitabile di dettaglio (*aliasing*). Il concetto di *profondità*, invece, riguarda il processo di quantizzazione che ogni campione subisce. Per profondità si intende il numero di bit necessari a rappresentare tutti i livelli di quantizzazione. Nelle immagini a toni di grigio, spesso, la profondità è di 8 bit (256 livelli), mentre in quelle a colori può essere di 8, 16 o 24 bit.

1.3.3 Binarizzazione di immagini a toni di grigio

Le immagini a toni di grigio sono matrici di pixel in cui ogni elemento può assumere un numero compreso in un certo range, che rappresenta la gamma dei possibili livelli di grigio. Nella maggior parte dei casi, la gamma dei toni di grigio si estende da 0 a 255, quindi ad ogni pixel viene associato un byte. Binarizzare un'immagine di questo tipo significa degradare la gamma a due soli valori: nero (0) e bianco (255). Il criterio secondo il quale ogni pixel viene trasformato in nero o

bianco è semplice: se il pixel in esame ha un valore superiore ad una certa soglia viene trasformato in bianco, altrimenti in nero.

L'operazione più difficile è rappresentata dalla scelta della soglia. La binarizzazione si propone, solitamente, di isolare le zone dell'immagine relative ad un oggetto dallo sfondo, perciò la soglia va scelta in modo che cada a metà tra i livelli di grigio che competono all'oggetto e allo sfondo. Ciò, però, non è facile e spesso si procede per tentativi. Un metodo valido è quello che si appoggia all'istogramma dell'immagine [1], che è un grafico in cui viene rappresentato il numero di occorrenze di ogni livello di grigio nell'immagine. In questo modo, la scelta ottima per la soglia potrebbe essere quella del baricentro dell'istogramma.

a)  b) 

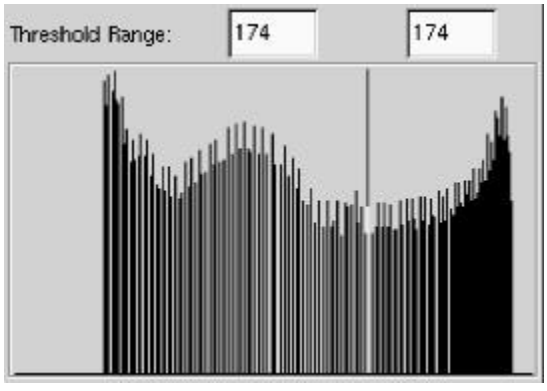
c) 

Fig. 1.3 a) Immagine a toni di grigio; b) immagine binarizzata; c) istogramma dell'immagine a toni di grigio.

1.3.4 I sistemi di rappresentazione del colore

Esistono vari sistemi per rappresentare un'immagine a colori. Quello *RGB* (Red-Green-Blue), senza dubbio tra i più conosciuti, è un sistema di riferimento in cui ogni colore viene individuato dalle sue componenti rossa, verde e blu. Queste rappresentano, infatti, i colori fondamentali grazie ai quali è possibile ottenerne qualunque altro, ovvero ogni colore può essere interpretato come una combinazione lineare dei tre fondamentali. Immaginando di prendere un sistema di riferimento cartesiano in tre dimensioni, ed associando ad ogni asse uno dei tre colori base, nasce il *cubo RGB.*

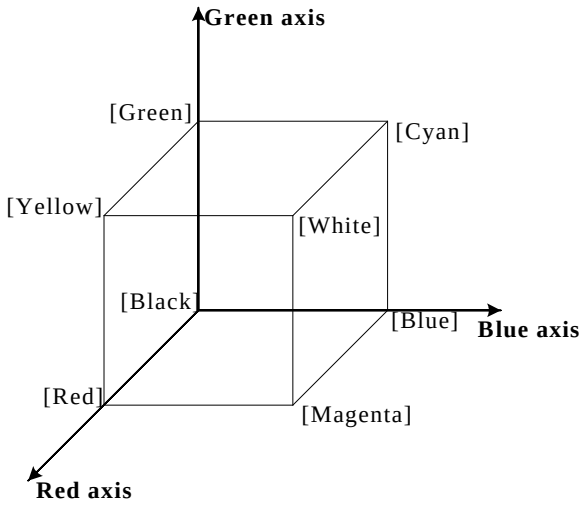


Fig. 1.4 Rappresentazione del cubo RGB.

Ogni punto all'interno del cubo rappresenta un colore e le sue proiezioni sui tre assi indicano le sue componenti di rosso, verde e blu. Il vertice del cubo che coincide con l'origine degli assi corrisponde al colore con tutte le componenti nulle, cioè il nero. Percorrendo i tre spigoli del cubo che partono dall'origine, si osservano tutte le gradazioni dei tre colori fondamentali. Il vertice del cubo opposto all'origine corrisponde al bianco, che è il colore con tutte le tre componenti al massimo valore. Lungo gli spigoli del cubo che partono dal vertice che rappresenta il bianco, si possono osservare altri tre colori in tutte le loro gradazioni: il ciano, il magenta e il giallo.

Ogni colore si può, quindi, esprimere anche in funzione delle componenti ciano, magenta e giallo, che rappresentano così una seconda terna di colori fondamentali. Il sistema *CMY* (Cyan-Magenta-Yellow) è complementare a quello RGB, e viene spesso utilizzato in tipografia in quanto l'assenza di colore si identifica nel bianco, che è solitamente il colore del supporto tipografico. Al contrario, per realizzare il colore nero si devono utilizzare tutti i tre colori. Per motivi analoghi, il sistema RGB è idoneo per applicazioni in cui lo sfondo è nero.

In ambito informatico, tra i due sistemi menzionati, quello RGB è sicuramente il più utilizzato e, nel seguito, si farà sempre riferimento a questo. Inoltre, solitamente viene dedicato un byte ad ognuna delle tre componenti, quindi queste possono assumere un qualsiasi valore compreso tra 0 e 255 e il numero complessivo di combinazioni (cioè colori) ottenibili è 256^3=16.777.216. Questo numero ragguardevole si avvicina al massimo numero di colori che l'occhio umano può distinguere. Le immagini a colori di alta qualità hanno, quindi, una profondità di 24 bit (8 per ciascuna delle tre componenti) e vengono anche dette *true color*.

C'è, infine, un terzo sistema molto utilizzato nell'ambito dell'elaborazione delle immagini, il sistema *HSI* (Hue-Saturation-Intensity) o *HSV* (Hue-Saturation-Value), in cui ogni colore viene rappresentato univocamente da tre componenti dette *tinta*, *saturazione* e *luminosità*. Si può passare dalle coordinate RGB a quelle HSI tramite una funzione biunivoca $\Re^3 \to \Re^3$, cioè ad ogni punto dello spazio tridimensionale RGB viene associato uno ed un solo punto dello spazio HSI.

Il sistema HSI nasce, quindi, da quello RGB attraverso un cambio di sistema di riferimento:

$$I = 0.30 \cdot R + 0.59 \cdot G + 0.11 \cdot B$$
$$x_I = 0.60 \cdot R - 0.28 \cdot G - 0.32 \cdot B$$
$$x_Q = 0.21 \cdot R - 0.52 \cdot G + 0.31 \cdot B$$

La prima equazione esprime come calcolare il segnale di *luminanza*, ovvero la luminosità del pixel, come una media pesata delle componenti RGB. In particolare,

viene dato maggior peso al verde, meno al rosso e ancor meno al blu, per rispettare la fisiologia dell'occhio umano che è più sensibile al verde, meno al rosso e ancor meno al blu. In questa formula, R, G e B rappresentano i valori di rosso, verde e blu normalizzati, in modo che la luminanza vari da zero, nel caso del nero, a uno, nel caso del bianco. Quantizzando la luminanza a 256 livelli, si può ottenere un'immagine a toni di grigio.

Con le altre due equazioni è possibile calcolare due nuovi segnali x_I e x_Q, che rappresentano, rispettivamente, la parte in fase e quella in quadratura di un nuovo segnale complesso $C = x_I + j \cdot x_Q$, detto *crominanza*. Se, invece di ragionare in termini di fase e quadratura, si parla in termini di ampiezza e fase, alla fase di *C* viene dato il nome di tinta (Hue) ed alla sua ampiezza quello di saturazione (Saturation):

$$H = \arctan\left(\frac{x_Q}{x_I}\right)$$

$$S = \sqrt{x_I^2 + x_Q^2}$$

Pur trattandosi di un sistema derivato dall'RGB, l'HSI è comodo perché consente di ragionare sui colori in modo più intuitivo e con un gergo più vicino a quello quotidiano. La nascita dell'HSI è, però, anche tecnico-storica: con l'avvento della televisione a colori fu necessario garantire a chi era in possesso di un ricevitore in bianco e nero di riuscire comunque a ricevere la trasmissione a colori vedendola sempre in bianco e nero, senza dover cambiare ricevitore. Ecco quindi che, invece di trasmettere su bande disgiunte i tre segnali rosso, verde e blu, si è pensato di trasmettere i segnali di luminanza e crominanza. In questo modo, chi possiede un ricevitore in bianco e nero è sensibile al solo segnale di luminanza e continua a vedere in bianco e nero, mentre i ricevitori a colori hanno una circuiteria aggiuntiva che permette di demodulare anche il segnale di crominanza e riottenere da *I*, x_I e x_Q i segnali RGB da inviare ai tre cannoni elettronici del cinescopio a colori [7].

Così come per l'RGB, è possibile dare una rappresentazione grafica anche del sistema HSI. Rappresentando sul piano del foglio il segnale di crominanza per tutti i possibili valori di tinta e saturazione, tenendo fissa la luminanza ad un certo valore, si ottiene una figura nota come *cerchio dei colori*.

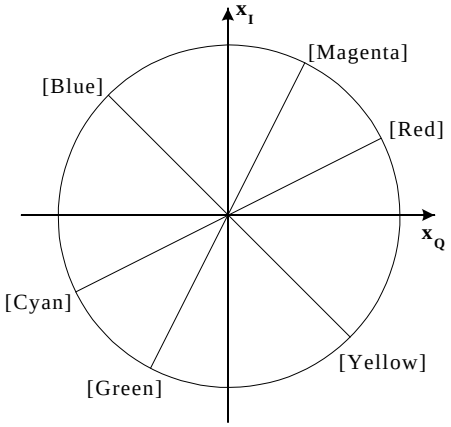


Fig. 1.5 Rappresentazione del cerchio dei colori.

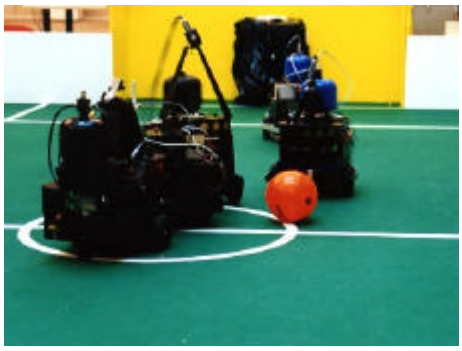
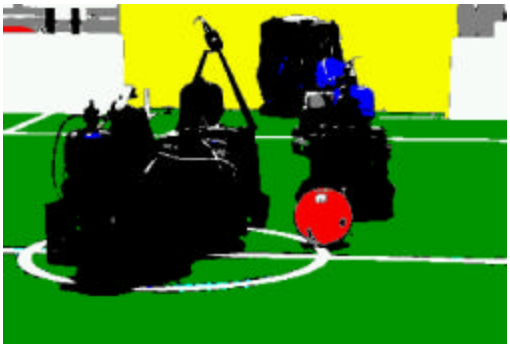
Muovendosi su cerchi concentrici si percorrono tutte le tinte possibili con la medesima saturazione. Il cerchio più esterno rappresenta i colori al massimo valore di saturazione, mentre il cerchio che degenera nel centro degli assi è un certo livello di grigio che dipende dal valore di luminanza scelto. Percorrendo invece una qualunque semiretta che parte dall'origine, si può osservare la variazione di saturazione di un determinato colore (tinta costante). Considerando anche la luminanza, si aggiunge una terza dimensione al diagramma: si può pensare di tracciare un cerchio dei colori per ogni valore di luminanza. Al variare di questa, varia anche la dimensione massima dei cerchi, perché fissando la luminanza si pone un limite al massimo valore che possono assumere le componenti RGB, e quindi alla saturazione del colore. In particolare, se la luminanza è massima (1 se R, G e B sono normalizzate) il cerchio dei colori raggiunge la sua massima dimensione e nel suo centro si osserva il colore bianco. Se, al contrario, la luminanza è nulla, il cerchio degenera in un punto di colore nero. Quindi, lo spazio HSI può essere immaginato come un cono in coordinate cilindriche, lungo il cui asse, partendo dal vertice, si può osservare tutta la gamma dei grigi.

1.3.5 Segmentazione di immagini a colori

Quello della segmentazione delle immagini a colori è un problema molto importante ed è, di norma, il primo processo di elaborazione ogni qual volta un robot mobile debba muoversi in ambienti strutturati in cui dominano i colori. Infatti, se è possibile distinguere gli elementi strutturali dell'ambiente tramite i colori, una buona segmentazione dell'immagine rappresenta, di fatto, una grossa percentuale del lavoro necessario per la classificazione degli elementi che compongono la scena.

Segmentare un'immagine significa etichettare con la stessa *label* tutti i pixel che si ritiene appartengano al medesimo elemento strutturale; spesso, poi, si usa associare ad ogni etichetta un colore, come retroazione visiva per valutare i risultati del processo. Un esempio tipico di applicazione è quello di un robot mobile che debba adempiere a compiti di navigazione in ambienti chiusi (uffici, laboratori, ecc.). In questo caso, la segmentazione delle immagini provenienti dal sistema di visione consisterà nell'etichettare in un certo modo i pixel appartenenti alle pareti, alle porte, al pavimento, ecc..

La segmentazione di un'immagine a colori nel sistema RGB si suddivide in due fasi. Nella prima, si partiziona il cubo RGB ricavando delle porzioni ad ognuna delle quali viene assegnata una label. Nella seconda, si scandiscono tutti i pixel dell'immagine e si assegna ad ognuno di essi la label della porzione del cubo RGB alla quale appartengono. La segmentazione va a buon fine ogni volta che è possibile distinguere popolazioni ben distinte di punti nel cubo RGB, come nel caso di immagini in cui esiste un elevato contrasto tra i colori dei vari oggetti. In questi casi è, infatti, agevole partizionare lo spazio, facendo in modo che ogni popolazione appartenga ad una sola porzione. Purtroppo, si ha spesso a che fare con immagini in cui, invece, esistono molte tinte intermedie. In questi casi, i pixel dell'immagine vanno ad occupare lo spazio nel cubo RGB in maniera più uniforme e la scelta delle soglie risulta molto più difficile. Immagini che presentano, quindi, molte sfumature, sono più difficili da segmentare e la bontà della segmentazione dipende molto dalla cura con la quale vengono scelte le soglie.

a)  b) 

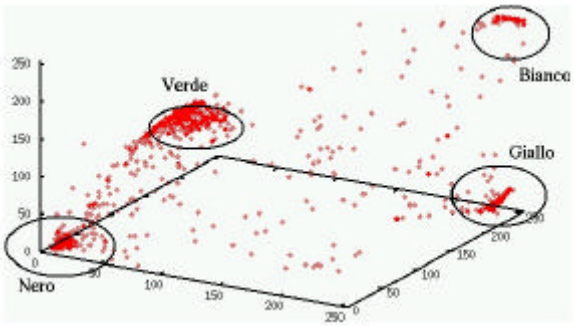
c) 

Fig. 1.6 a) Immagine acquisita; b) immagine segmentata; c) cubo RGB.

Il partizionamento dello spazio RGB andrebbe fatto tramite sottoinsiemi dalla forma adatta a quella della popolazione di punti a cui si riferiscono. Tuttavia, quello che solitamente si fa è stabilire dei semplici intervalli, contraddistinti quindi da un valore minimo e massimo, sulle componenti RGB che, in termini geometrici, consiste nel suddividere lo spazio in parallelepipedi.

Il concetto di segmentazione nel sistema HSI è simile a quello nel sistema RGB, tranne per il fatto che ad essere partizionato è lo spazio HSI. Le soglie vanno prese, quindi, in termini di intervalli sui valori di luminosità, tinta e saturazione. Il grande vantaggio della segmentazione nel sistema HSI è quello di poter lavorare su un parametro per volta: per trovare le soglie più adatte, si parte dalla scelta sull'intervallo della tinta, e ciò può avvenire senza preoccuparsi né della luminosità né della saturazione, si procede con la scelta dell'intervallo sulla saturazione e poi sulla luminosità.

1.3.6 Estrazione dei contorni

Quello del riconoscimento di forme in un'immagine è un problema molto comune e, a tal fine, la strada più efficace è quella dell'analisi dei contorni. Esistono varie tecniche per estrarre i bordi di un oggetto. La più semplice consiste nell'applicare all'immagine un filtraggio che esalti le discontinuità che si osservano in corrispondenza dei contorni degli oggetti. In definitiva, si effettua un filtraggio di tipo passa-alto: tutte le brusche variazioni di tinta e/o di luminosità nell'immagine danno luogo a contributi frequenziali alti. In questo modo, si ottiene una nuova immagine che presenta contributi più alti in corrispondenza proprio dei bordi degli oggetti. L'operazione di filtraggio può essere effettuata sia sul segnale di luminanza, quindi su un'immagine a toni di grigio, sia sulla tinta, quindi su un'immagine a colori.

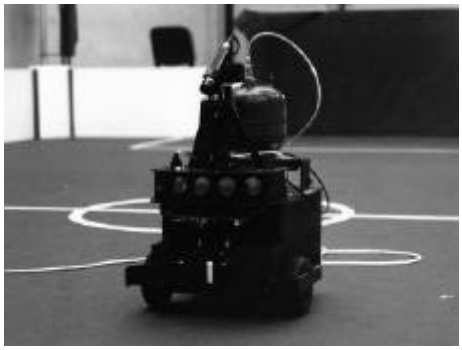
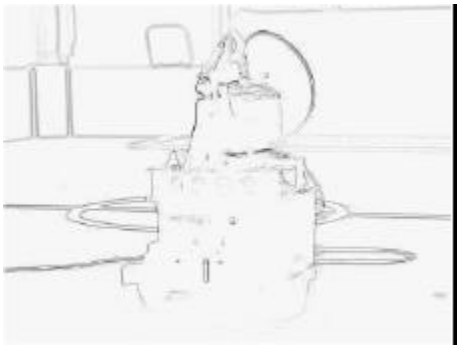
a)  b) 

Fig. 1.7 a) Immagine acquisita; b) immagine dei contorni in negativo.

Spesso, l'immagine dei contorni viene sottoposta ad un'operazione di binarizzazione, prima di essere passata ai successivi processi di elaborazione.

Il tipo di filtraggio utilizzato nella figura precedente evidenzia i bordi sia orizzontali che verticali, ma esistono altri tipi di filtro che rilevano i bordi in una sola direzione, oppure anche in diagonale. In pratica, in base alla forma della maschera vengono individuati certi contorni piuttosto che altri. Non solo, ma si può anche decidere di dare un peso diverso ai vari pixel. Nel caso più generale, quindi, il filtraggio avviene tramite una matrice in cui i vari elementi vengono pesati in modo opportuno.

1.3.7 La trasformata di Hough

Una volta in possesso dell'immagine dei contorni, risulta necessario elaborare opportunamente l'immagine al fine di estrarne le caratteristiche di interesse. La trasformata di Hough [8] è un metodo che consente di riconoscere, in un'immagine, curve esprimibili analiticamente. Sebbene sia possibile, in linea teorica, riconoscere anche curve complesse, di fatto la trasformata di Hough viene utilizzata prevalentemente per individuare rette.

Data un'immagine binaria di dimensioni (dim_x,dim_y), questa viene trasformata passando ad un nuovo spazio, detto trasformato. La trasformazione viene effettuata tracciando per ogni pixel acceso di coordinate (x,y) una curva nello spazio immagine:

$$r = x \cdot \cos t + y \cdot \sin t$$

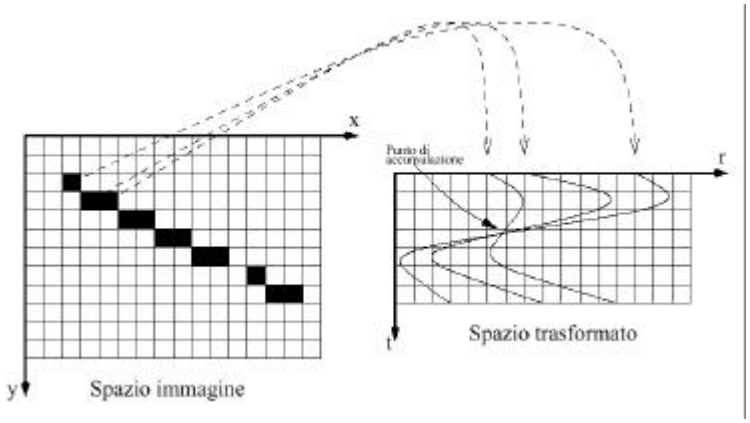


Fig. 1.8 Trasformata di Hough.

Lo spazio trasformato è un piano nelle variabili r e t, dove $t \in [0,\pi]$ e $r \in [0,dim_x]$. Ogni pixel dell'immagine corrisponde ad una curva nello spazio trasformato, mentre ogni punto dello spazio trasformato corrisponde ad una retta nell'immagine. La proprietà fondamentale della trasformata di Hough è che, tutti i pixel dell'immagine che appartengono ad una retta, danno luogo a curve nello spazio trasformato che si intersecano tutte nello stesso punto. La potenza della trasformata di Hough sta nella robustezza ai disturbi: nella figura seguente, si nota come la retta evidenziata sia interrotta dalla sagoma del robot; tuttavia, la si riesce ad individuare.

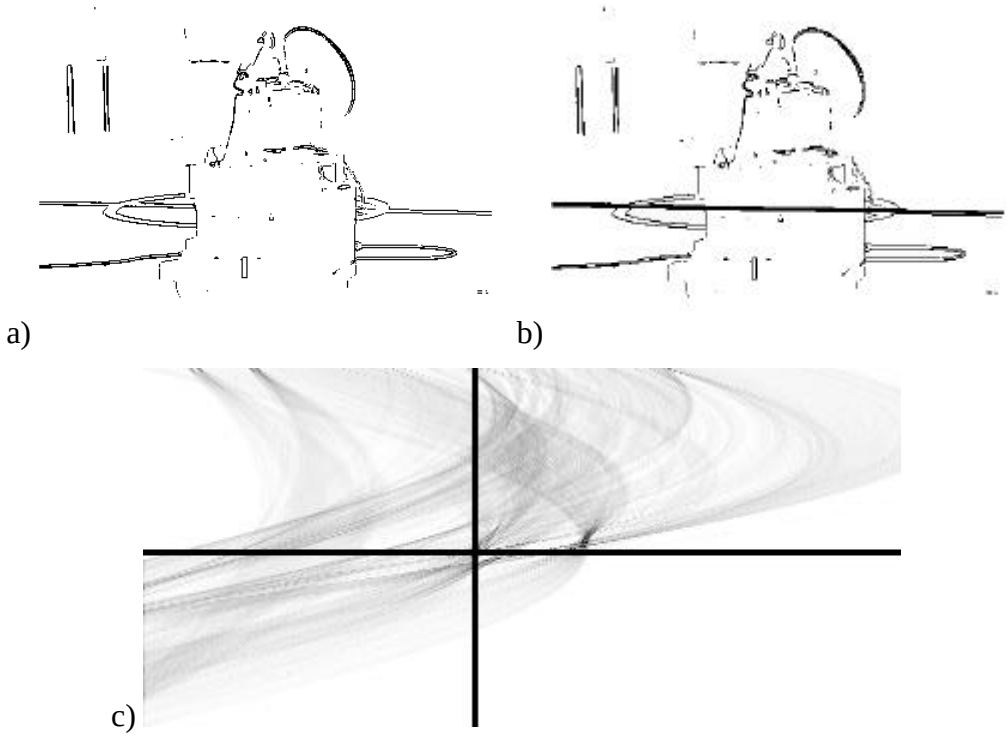


Fig. 1.9 a) Immagine dei contorni binarizzata; b) risultato della trasformata; c)spazio trasformato.

1.4 La calibrazione dei sensori visivi

In questo paragrafo viene preso in considerazione un problema molto comune nella robotica mobile, ovvero la stima delle distanze relative degli oggetti che circondano il robot. Per fare ciò, occorre implementare un procedimento di rimozione prospettica.

1.4.1 La proiezione prospettica

Un'immagine è una proiezione bidimensionale del mondo tridimensionale su un piano, che avviene sezionando, tramite il *piano di proiezione*, la piramide visiva il cui vertice risiede nel *punto di proiezione*. Un determinato punto del mondo tridimensionale corrisponde, quindi, all'intersezione della retta, detta *raggio di proiezione*, che unisce il punto in questione ed il centro di proiezione, con il piano di proiezione. In questa analisi si assume che i raggi di proiezione siano linee rette, anche se, in realtà, un'immagine acquisita da una telecamera nasce da un processo di

proiezione, in cui i raggi di proiezione vengono sottoposti a distorsione, per effetto delle lenti attraversate durante il loro cammino. Tuttavia, se la distorsione della lente è limitata, può essere trascurata e la proiezione può continuare ad essere immaginata tramite linee rette.

Nella figura seguente è rappresentato, in modo schematico, il fenomeno descritto. Variando la posizione del centro di proiezione relativamente all'oggetto, varia la proiezione e, di conseguenza l'immagine, dell'oggetto. La distanza tra il centro ed il piano di proiezione, è detta *distanza focale* ed influenza la dimensione dell'immagine dell'oggetto.

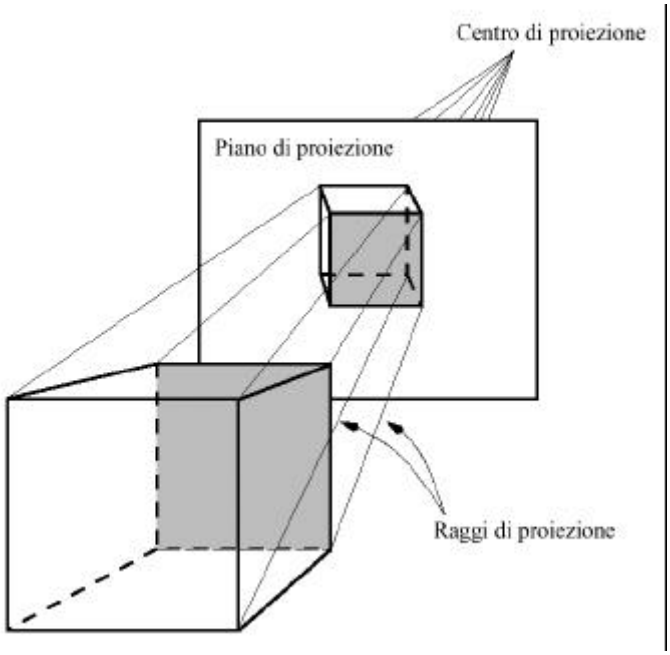


Fig. 1.10 La proiezione prospettica.

1.4.2 La rimozione prospettica

Come detto, la proiezione prospettica da luogo alla nascita di un'immagine bidimensionale, perciò fa perdere inevitabilmente una dimensione. Applicare la procedura inversa non porta, quindi, alla ricostruzione della scena in tre dimensioni: l'unica possibilità è di effettuare una proiezione inversa su un piano noto assumendo, anche in questo caso, che i raggi siano linee rette. Ogni pixel dell'immagine

corrisponde, sul piano di proiezione, all'intersezione della retta che unisce il centro di proiezione ed il pixel, con il piano stesso.

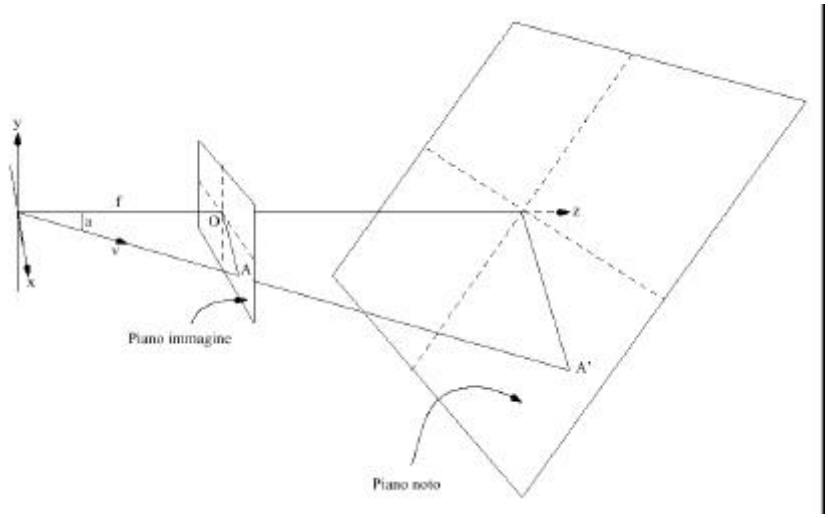


Fig. 1.11 La proiezione inversa.

Il punto *A* viene proiettato sul piano di proiezione nel punto *A'*. Per individuare le coordinate del punto *A'*, si deve quindi effettuare l'intersezione del raggio di proiezione passante per *A* con il piano.

La procedura di rimozione prospettica si suddivide nelle seguenti operazioni:

- si stabilisce un piano su cui effettuare la proiezione;
- si calcola la posizione e l'orientazione della telecamera rispetto a tale piano;
- si considerano, uno alla volta, tutti i pixel dell'immagine e si determina la retta che li individua;
- si effettua una roto-traslazione per posizionare la telecamera nel punto corretto;
- si calcola l'intersezione tra la retta e il piano di proiezione.

Ricostruendo correttamente la scena su piani noti, tutto ciò che non appartiene, nella scena reale, a questi piani, viene ricostruito erroneamente. Per esempio, si prenda in considerazione una telecamera installata su un veicolo. Questa riprende la sede stradale e l'immagine appare distorta per effetto della prospettiva, che è possibile rimuovere, proiettando l'immagine stessa su un piano e collocando il piano immagine in modo da rispettare la posizione e l'orientazione della telecamera rispetto al suolo [9]. In questo modo, tutto ciò che nell'immagine corrisponde al suolo viene

correttamente proiettato sul piano (in questo caso si tratta di un'operazione invertibile), però, tutto ciò che nell'immagine non corrisponde al suolo viene proiettato sul piano come se vi appartenesse. Ogni ostacolo che emerge dal suolo, quindi, compare nell'immagine corretta sotto forma di una sagoma deformata. Questa coincide con quella dell'ombra che l'oggetto proietterebbe sul suolo nel caso in cui vi fosse una sorgente di luce puntiforme posta nel punto in cui si trova la telecamera.

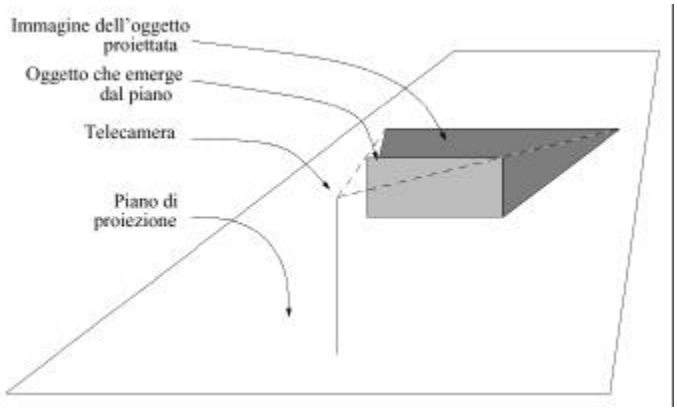


Fig. 1.12 Effetto della rimozione prospettica su oggetti non appartenenti al piano di ricostruzione.

1.4.3 La distorsione della lente

In precedenza, si è assunto che il processo di proiezione prospettica avvenga tramite raggi rettilinei. In realtà, esistono sempre delle incurvature dovute alle lenti della telecamera e talvolta non è possibile trascurarle. Il processo di proiezione inversa al fine di rimuovere la distorsione dall'immagine, quindi, andrebbe effettuato tenendo conto del fatto che i raggi sono curvilinei. In realtà, quello che si fa è applicare la proiezione inversa non sull'immagine acquisita, ma su un'immagine corretta rimuovendo la distorsione dovuta alla lente, in modo da riportarsi nella condizione di applicazione del modello pin-hole in cui la proiezione avviene tramite raggi rettilinei.

Per rimuovere l'effetto della distorsione dall'immagine è necessario conoscere come vengono deviati i raggi dalla lente della telecamera. Se l'apertura della telecamera non supera gli 80°, si può ipotizzare che la distorsione dovuta al dispositivo ottico sia di tipo sferico, ciò significa che ogni punto dell'immagine

risulta affetto da un errore che dipende dalla sua distanza dal centro dell'immagine. Il centro dell'immagine, che coincide con il prolungamento dell'asse ottico, è l'unico punto a non essere affetto dalla distorsione mentre, spostandosi verso gli angoli dell'immagine, la distorsione si fa sempre più pesante.

Per rimuovere questo tipo di distorsione da un'immagine, la posizione di ogni pixel va corretta di una quantità che dipende dalla sua distanza dal centro dell'immagine tramite un parametro *K*:

$$\begin{cases} dx = K \cdot x\left(x^2 + y^2\right) \\ dy = K \cdot y\left(x^2 + y^2\right) \end{cases}$$

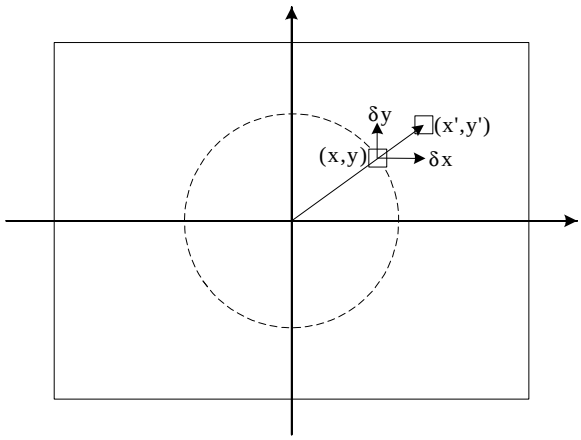


Fig. 1.13 Effetto della rimozione della distorsione radiale.

Il valore di *K* è molto importante, in quanto esprime l'entità della distorsione: quanto più è grande e tanto più è elevata la distorsione dell'immagine. Spesso, tale parametro va determinato sperimentalmente.

Capitolo 2

Progettazione di un sensore omnidirezionale

2.1 Perché un sensore omnidirezionale

Molte delle applicazioni nel campo della visione artificiale richiedono (o comunque possono trarre beneficio da) un ampio campo visivo. Esistono molteplici esempi a questo riguardo: la sorveglianza, la teleconferenza, l'acquisizione di modelli per la realtà virtuale, l'operazione in grandi spazi di lavoro, e chiaramente il trasporto, l'auto-localizzazione, la stima del proprio movimento, la pianificazione.

Purtroppo, le telecamere convenzionali hanno, nella maggior parte dei casi, un campo visivo piuttosto limitato, e questo risulta essere spesso restrittivo per l'applicazione. Per cercare di ovviare a questa mancanza, sono state introdotte numerose varianti ai sensori visivi classici. Fra le soluzioni proposte, le più immediate riguardano l'utilizzo di più telecamere o di telecamere mobili (*active vision*). Un approccio alternativo molto efficace, che permette di semplificare la gestione dell'input, è di accrescere il campo visivo utilizzando specchi in congiunzione alle tradizionali telecamere. Per *sensore catadiottrico* (o omnidirezionale) si intende, appunto, la combinazione di lenti e specchi, posizionati in configurazioni appropriatamente studiate, per ottenere un campo visivo molto superiore rispetto a quello del sensore effettivamente utilizzato. Il termine catadiottrico deriva da "diottrica", la disciplina degli elementi rifrangenti (le lenti), e "catottrica", la disciplina delle superfici riflettenti (gli specchi).

Se si vuole espandere il campo visivo in maniera isotropa, la migliore delle

soluzioni è, probabilmente, quella di adottare specchi convessi con un asse di simmetria centrale. Posizionando l'asse ottico della telecamera in verticale e facendolo coincidere con l'asse di simmetria dello specchio, si ottiene un ampio campo visivo in tutte le direzioni.

Nel campo della robotica, in particolare, ci si sta muovendo verso ambienti di lavoro sempre meno strutturati e sempre più dinamici, che propongono numerosi problemi legati all'individuazione di caratteristiche generali dell'ambiente circostante, come nell'auto-localizzazione o nella ricerca di oggetti di interesse. Questo richiede l'utilizzo di sensori visivi che diano un'informazione più "globale" (anche se meno precisa e particolareggiata) sulla scena in cui il robot è immerso, e permettano di ottenere una descrizione veloce, magari sommaria, ma più direttamente traducibile in azione.

L'impiego di catadiottri ha, quindi, il vantaggio di raccogliere in una sola immagine un'informazione riguardante una vasta area, evitando i complessi meccanismi di controllo generalmente impiegati nella visione attiva; tuttavia, vanno anche evidenziati due nuovi problemi introdotti:

- la riflessione della telecamera e del robot sullo specchio fa sì che la parte centrale dell'immagine, ossia quella caratterizzata da maggiore risoluzione e minori distorsioni, sia generalmente occupata dall'immagine riflessa del robot e non sia, quindi, utilizzabile per descrivere la scena circostante;
- l'utilizzo congiunto di una telecamera e di uno specchio convesso determina il sommarsi di effetti distorcenti di cui, spesso, risulta difficile trovare la legge e quindi compensare l'effetto.

Tutti questi aspetti fanno sì che, tramite il catadiottro, si possano ottenere informazioni adeguate per un'analisi di massima di una scena ampia (facilitando, ad esempio, l'auto-localizzazione, l'individuazione di oggetti ed elementi di possibile interesse per l'applicazione, ecc.), ma non sufficientemente dettagliate per un'analisi approfondita dei particolari (ad esempio, per l'individuazione di ostacoli). Ecco perché una buona soluzione è quella di associare al catadiottro un tipo differente di sensore, in modo da accrescere la qualità e la risoluzione dell'informazione legata ad una specifica parte del campo visivo e rendere più robusta l'analisi della scena.

Anche nel progetto in esame è stato introdotto un secondo sensore, in questo caso una telecamera a CCD, usato per inquadrare la regione frontale del robot.

2.2 Definizione delle specifiche

Obiettivo di questo capitolo è la descrizione della progettazione di un sensore omnidirezionale per un robot autonomo, costituito da uno specchio a simmetria rotazionale e da una telecamera il cui asse principale sia allineato con l'asse di simmetria dello specchio stesso. Il punto nodale di questo tipo di sensore è la forma dello specchio, che può essere modellato in modo opportuno e con la precisione necessaria per ottenere dalla telecamera le informazioni desiderate.

In letteratura esistono diversi articoli relativi alla progettazione di visori a 360°, in particolare [10] rappresenta una delle prime pubblicazioni sulla visione omnidirezionale, e i tipi di specchio più utilizzati sono conici, conici con vertice sferico e isometrici. Di seguito è riportata una tabella che riassume pregi e difetti di ognuna di queste tipologie.

Tipo di specchio	Pregi	Difetti
Conico	- Scegliendo opportunamente i parametri geometrici è possibile eliminare dall'immagine il corpo del robot - Errore relativo basso nelle misure delle distanze sul piano di riferimento	- Distanza massima misurabile molto ridotta - Scarsa risoluzione nella misura di distanze piccole
Conico con vertice sferico	- Buona risoluzione nella misura di distanze piccole - Errore relativo basso nelle misure delle distanze sul piano di riferimento	- Una parte dell'immagine è inutilizzabile, perché occupata dal corpo del robot - Distanza massima misurabile molto ridotta
Isometrico	- Errore assoluto basso nelle misure delle distanze sul piano di riferimento - Conversione da distanze sul piano immagine a distanze sul piano di riferimento con una semplice operazione di moltiplicazione	- Errore relativo elevato nella misura di distanze piccole - Parte dell'immagine risulta inutilizzabile perché occupata dal corpo del robot

I primi due tipi introducono una distorsione che può, comunque, essere corretta utilizzando informazioni sulla struttura geometrica dello specchio ma, allo stesso tempo, riduce la precisione ottenibile nelle misurazioni. Il terzo tipo è una soluzione molto interessante, perché fornisce un'immagine del piano di riferimento senza distorsioni, perciò facilmente interpretabile dall'occhio umano, però mantiene un errore assoluto costante (ad esempio, pari a 5cm). Questo porta a commettere errori relativi troppo grandi nel valutare distanze piccole (ad esempio del 20% su 25cm) ed errori relativi eccessivamente piccoli nel valutare distanze grandi (ad esempio dell'1% su 500cm); questa caratteristica è esattamente l'opposto di ciò a cui si è normalmente interessati in applicazioni robotiche. Per questo motivo, l'obiettivo di questo lavoro consiste nel progetto di un profilo per lo specchio che permetta misure di distanze sul pavimento (assunto come piano di riferimento) con un errore relativo costante. Inoltre, la proprietà di non distorcere di uno specchio isometrico vale solo se lo stesso viene posizionato ad un'altezza dal pavimento esattamente pari a quella per cui è stato progettato, perciò non vale su diversi tipi di robot. D'altro canto, la soluzione qui presentata distorce l'immagine del pavimento ma, come descritto nel prossimo capitolo, questo effetto può essere rimosso utilizzando una tabella di look-up che può essere generata automaticamente in qualsiasi configurazione d'interesse. Per questo motivo, ha validità più generale; inoltre, la velocità con cui si risale alle distanze sul piano di riferimento è preservata, in quanto si sostituisce, alla semplice operazione di moltiplicazione, un altrettanto semplice accesso ad una tabella che associa a ciascun punto sul pavimento un pixel dell'immagine acquisita.

È ora opportuno cercare di capire la natura dell'errore commesso qualora si voglia misurare la distanza dal robot di un punto P giacente sul pavimento; questo può essere fatto sfruttando l'immagine P' che il punto stesso produce sul CCD del sensore visivo. Per semplicità e senza perdita di generalità, data la simmetria del problema, si considera il caso in cui un punto P giacente sul pavimento si muove radialmente rispetto ad un punto O, proiezione sul pavimento dell'origine del sistema di riferimento, coincidente con il centro della base del robot.

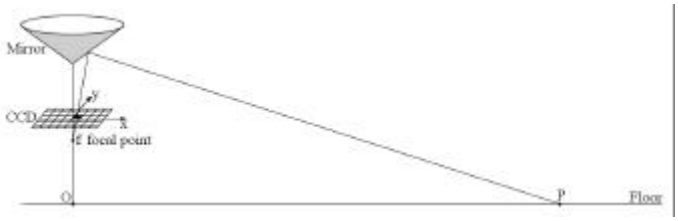


Fig. 2.1 Schema della procedura di stima della distanza tra *O* e *P*.

Nota la geometria del sistema specchio-telecamera e la posizione del punto *P*, è possibile risalire alle coordinate (x,y) della proiezione *P'* del punto *P* sul piano del CCD della telecamera (modellata secondo il *pin-hole camera model*); viceversa, note le coordinate (x,y) e la geometria dello sensore, è possibile risalire alla distanza del punto *P* dal punto *O* (d_{OP}).

Nella realtà non si dispone esattamente delle coordinate (x,y), ma di una loro discretizzazione (x_d,y_d) ottenibile a partire dalle coordinate del pixel in cui *P* viene proiettato; nota la coppia (x_d,y_d), si può risalire ad una stima della distanza d_{OP}, anch'essa discretizzata. L'errore inevitabilmente commesso in questa procedura di discretizzazione dipende dalla geometria dello specchio, ed il suo valore massimo in funzione della posizione del punto *P* è determinabile una volta nota la funzione $d_{OP}(x_d,y_d)$. Sfruttando queste considerazioni, l'errore associato al calcolo della distanza d_{OP} attraverso uno specchio conico ed uno isometrico è illustrato nelle seguenti figure.

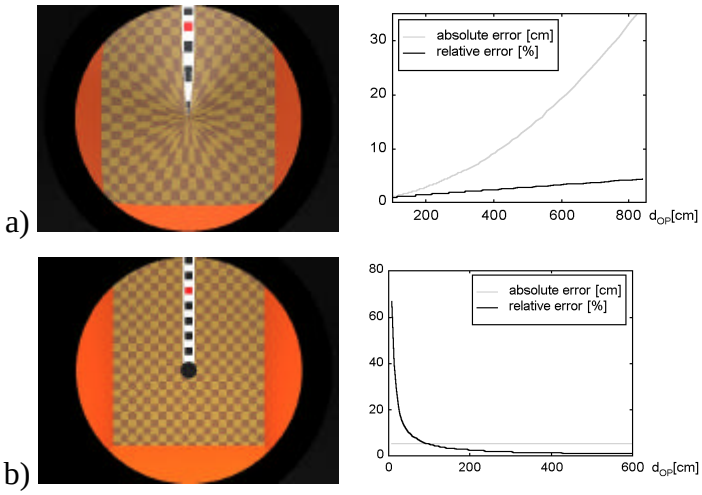


Fig. 2.2 Immagine prodotta da uno specchio (320x240) ed errore associato alla misura di d_{OP} nel caso di uno specchio conico (a)) e di uno isometrico (b)).

Aumentando la risoluzione della telecamera si può ottenere, a parità di profilo dello specchio utilizzato, una diminuzione dell'errore commesso sulla misura di d_{OP}, come diretta conseguenza del raffinamento della discretizzazione delle coordinate (x,y).

Come si può notare dalla figura precedente, lo specchio conico, a cui quello qui descritto si avvicina abbastanza (dato che la forma conica porta ad un errore relativo quasi costante), permette, scelta opportunamente l'inclinazione del profilo, di eliminare dal centro dell'immagine il corpo del robot. Questa proprietà, ovviamente non ottenibile con uno specchio isometrico, che non deve distorcere ciò che si trova sul piano di riferimento, evita lo spreco della parte centrale dell'immagine, così da avere nella zona a maggior risoluzione e minor distorsione le informazioni più importanti, ovvero quelle relative alle vicinanze del robot stesso. La controindicazione di questa scelta sta nel fatto che, proiettando l'intera circonferenza che si decide di rimuovere nel centro dell'immagine, si perde "risoluzione" nell'individuazione di punti prossimi a tale circonferenza. Il principale problema è rappresentato dal fatto che l'immagine degli oggetti si assottiglia con l'avvicinarsi degli stessi al robot, e questo può renderne complessa l'individuazione. In passato sono state adottate diverse soluzioni nel tentativo di evitare questo tipo di problemi ed, in pratica, tutte rinunciavano all'eliminazione del corpo del robot dall'immagine. A questo proposito, si citano gli specchi conici a vertice sferico, progettati per rimediare alla discontinuità dell'immagine derivante dagli specchi semplicemente conici: questi ultimi presentano, per costruzione, una discontinuità di curvatura in corrispondenza del vertice dello specchio e quindi mappano in un unico punto dell'immagine un'intera circonferenza attorno al centro del robot; imponendo la continuità della curvatura in corrispondenza del vertice, ad esempio utilizzando un vertice sferico, si evita questo problema ma si porta nell'immagine una certa porzione del corpo del robot. Se non si vuole utilizzare la soluzione del vertice sferico, per non rinunciare ai benefici introdotti dalla rimozione del corpo del robot dall'immagine, si deve porre un'attenzione particolare alla scelta della dimensione della circonferenza da far collassate nel centro dell'immagine. La soluzione adottata è presentata nel paragrafo 2.4.

Un altro spunto interessante proposto in [11] è di aggiungere allo specchio altre due parti a quella descritta, utilizzata per misure di distanze sul piano di riferimento. Progettare uno specchio che consenta buone risoluzioni sul piano di riferimento porta inevitabilmente a riscontrare problemi nell'individuazione di oggetti che stanno al di sopra del piano stesso, ed in particolare nell'individuazione di landmark sulle pareti, su ostacoli o altri robot. Si può, allora, pensare di rinunciare alla precisione nella misura e risolvere il problema costruendo una seconda parte dello specchio dedicata alla visione degli oggetti che non stanno sul piano di riferimento. Inoltre, per ovviare al problema della perdita di "risoluzione" discussa, è stata introdotta una terza parte più esterna, detta specchio di prossimità, che proietta la zona più prossima al robot nei pixel più esterni dell'immagine, ottenendo così due vantaggi: gli oggetti non degenerano in un punto quando si avvicinano al robot ma appaiono grandi, e si diminuisce la zona non visibile coperta dal corpo del robot.

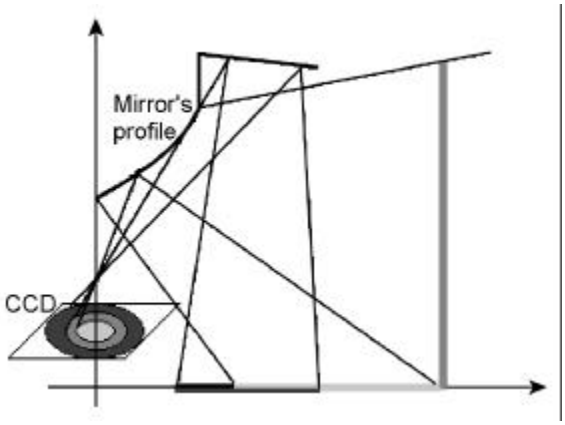


Fig. 2.3 Profilo complessivo di uno specchio tipo quello proposto in [11] e raffigurazione delle zone visibili con tutte le tre parti di cui è composto.

Entrambe queste soluzioni sono molto interessanti, ma non sono state ritenute utili ai nostri scopi, per evitare di perdere in generalità nell'utilizzo del sensore omnidirezionale. Perciò, si è deciso di costruire uno specchio costituito solo dalla parte per le misure sul piano di riferimento.

2.3 Descrizione della metodologia di progetto

Nella progettazione dello specchio è stata utilizzata una tecnica già sviluppata in [11]: data una generica funzione $f:\Re^2 \to \Re^2$, si può sintetizzare uno specchio che proietti ciascun punto (*X*,*Y*) appartenente ad un regione limitata del piano di riferimento in un punto (*x*,*y*) sul piano del CCD, in modo tale che (*x*,*y*)=*f*(*X*,*Y*). Data la simmetria rotazionale dello specchio, in realtà interessa sintetizzare un profilo radiale che proietti i punti a distanza *D* ($0 \le D \le D_{MAX}$) sul piano di riferimento in punti a distanza *d* ($0 \le d \le d_{MAX}$) sul piano del CCD, dal centro del sensore. La proiezione segue una relazione $d=f^*(D)$, dove f^* è una generica funzione $f^*:[0, D_{MAX}] \to [0, d_{MAX}]$.

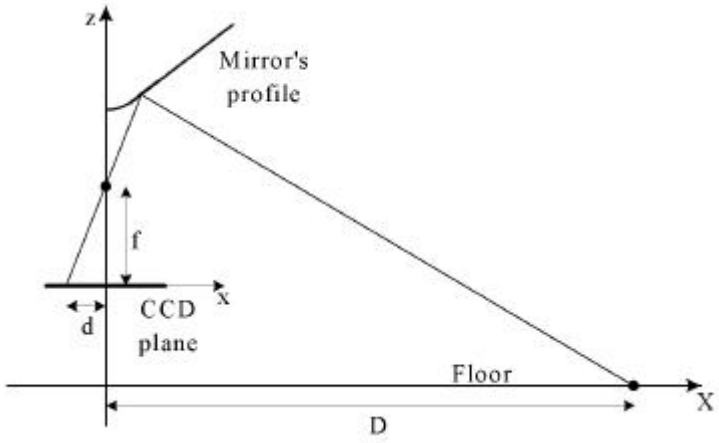


Fig. 2.4 Mappatura dei punti sul piano di riferimento in punti sul piano del CCD, in funzione del profilo dello specchio.

La soluzione esatta di questo tipo di problemi richiede la risoluzione di un'equazione differenziale piuttosto complessa; in [12] viene proposto un metodo che permette di ottenere il risultato di tale equazione nel semplice caso $d=f^*(D)=kD$, cui corrisponde uno specchio isometrico. [11], invece, propone una tecnica di risoluzione basata su un'approssimazione locale lineare del profilo dello specchio, ottenuta sostituendo in ogni punto il profilo stesso con la sua tangente. Di seguito sono riportati i passi principali di questa tecnica considerando, per ora, il caso in cui $f^*(0)=0$, ovvero senza eliminazione del corpo del robot dall'immagine.

1. Si discretizza l'intervallo $[0,D_{MAX}]$ in un insieme del tipo $\{0,D_1,D_2,\ldots, D_{N-1}=D_{MAX}\}$, che viene mappato, dalla funzione f^*, nell'insieme $\{0,d_1,d_2,\ldots, d_{N-1}=d_{MAX}\}$.
2. [i=0] Il primo tratto del profilo passa per il punto $P_0=(0,z_0)$ e ha tangente orizzontale: in questo modo, D=0 viene mappato in d=0. Il valore z_0, fissato dal progettista, coincide con l'altezza del vertice dello specchio da terra; inoltre, si definisce r_2 come la retta passante per P_0 ed avente inclinazione nulla.
3. [$i=i$+1] Si costruisce r_1 come la retta passante per il punto focale $(0,f)$ ed il punto sul CCD $(-d_i,h)$, dove h è l'altezza del CCD da terra.

   Si determina P_i come punto di intersezione tra le rette r_1 e r_2. Si costruisce r_3 come la retta passante per i punti P_i e $(D_i,0)$. Dato che r_1 e r_3 individuano il percorso che la luce deve seguire per congiungere il punto a distanza d_i sul CCD con il punto a distanza D_i sul piano di riferimento, si può calcolare la tangente del profilo nel punto P_i utilizzando le leggi dell'ottica.

   Si costruisce la retta r_2 passante per P_i e tangente al profilo in P_i stesso.

   Se $i\neq(N$-1) si ripete il punto 3.
4. Si costruisce il profilo dello specchio congiungendo i punti P_i, i=0,1,2,…,N-1.

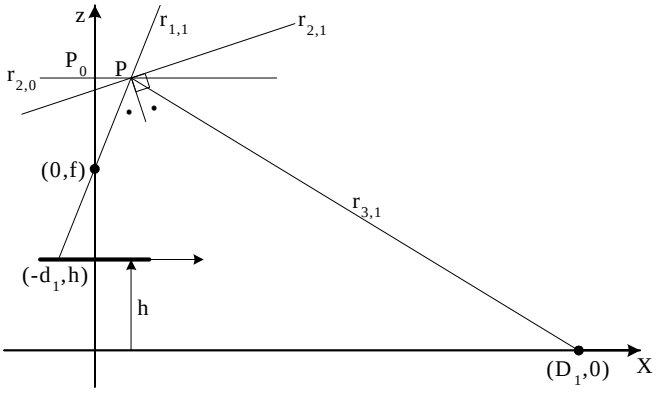
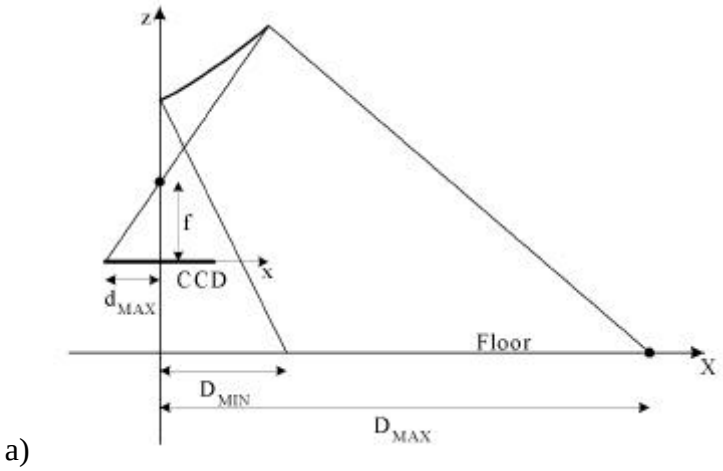


Fig. 2.5 Schema per la costruzione del profilo dello specchio ($r_{*,i}$ indica r_* all'iterazione i-esima).

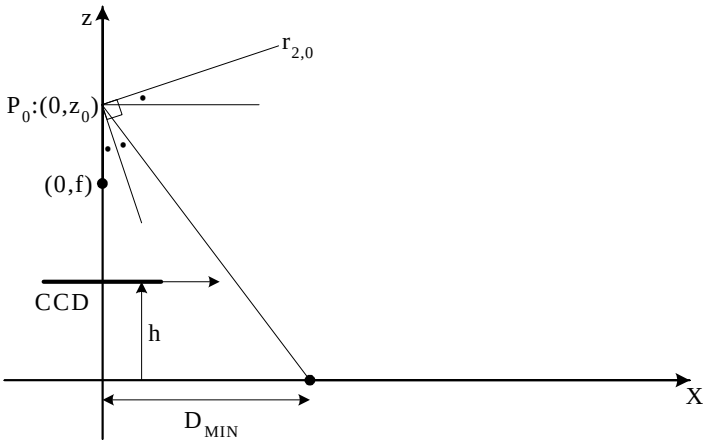
Apportando una piccola modifica ai primi due passi dell'algoritmo di base, si può considerare anche il caso $f^*:[D_{MIN},D_{MAX}]\rightarrow[0,d_{MAX}]$, con $f^*(D_{MIN})$=0, in modo da eliminare dal centro dell'immagine l'area occupata dal corpo del robot.

1'. Si discretizza l'intervallo $[D_{MIN},D_{MAX}]$ in un insieme del tipo $\{D_0=D_{MIN},D_1,D_2,\ldots,D_{N-1}=D_{MAX}\}$, che viene mappato, dalla funzione f^*, nell'insieme $\{0,d_1,d_2,\ldots,d_{N-1}=d_{MAX}\}$.

2'. [i=0] Il primo tratto del profilo passa per il punto $P_0=(0,z_0)$ e ha tangente pari a $\tan\left(\frac{\arctan\left(\frac{D_{MIN}}{z_0}\right)}{2}\right)$: in questo modo, $D=D_{MIN}$ viene mappato in d=0. Il valore z_0, fissato dal progettista, coincide con l'altezza del vertice dello specchio da terra; inoltre, si definisce r_2 come la retta che rappresenta questo primo tratto.



a)



b)

Fig. 2.6 a) Mappatura dell'intervallo $[D_{MIN},D_{MAX}]$ in punti sul piano del CCD; b) schema per il calcolo di r_2 di cui al passo 2'.

Come detto in precedenza, l'obiettivo di questo lavoro consiste nel costruire un profilo dello specchio che consenta misurazioni delle distanze sul pavimento commettendo un errore relativo costante: in questo modo, gli oggetti vicini vengono rilevati con sufficiente precisione, mentre possono essere commessi errori assoluti piuttosto elevati nella stima di grandi distanze. Si supponga di voler utilizzare la parte centrale dell'immagine per misurare la distanza degli oggetti che giacciono sul piano di riferimento commettendo un errore relativo pari ad a; si supponga, inoltre, che queste distanze siano più grandi del valore minimo D_{MIN}. Nel centro dell'immagine sono mappati i punti a distanza D_{MIN}; nei pixel più prossimi al centro dell'immagine (distanti d_1 dal centro del CCD) sono mappati i punti sul pavimento che si trovano ad una distanza $D_1=D_{MIN}+a\cdot D_{MIN}$; nei pixel successivi (distanti d_2 dal centro del CCD) sono mappati i punti a distanza $D_2=D_1+a\cdot D_1$, e così via. In questo modo, vengono costruiti per punti gli insiemi $\{D_0=D_{MIN},D_1,D_2,\ldots,D_{N-1}=D_{MAX}\}$ e $\{0,d_1,d_2,\ldots,d_{N-1}=d_{MAX}\}$ con le seguenti posizioni:

1. d_i=distanza dell'i-esimo pixel dal centro del CCD, con $i=1,2,\ldots,N-1$;
2. $D_i=D_{i-1}+a\cdot D_{i-1}$, con $i=1,2,\ldots,N-1$ e $D_0=D_{MIN}$.

Questo modo di procedere permette di costruire immediatamente la tabella di look-up citata al paragrafo 2.2, che consente di associare a ciascun pixel p dell'immagine il valore D_p, coincidente con la stima della distanza di un punto P giacente sul pavimento ed avente proiezione $P'\in p$ sul CCD. Se d_i è la distanza di p dal centro del CCD, si ponga $D_p=D_i$. Per come è costruito il profilo dello specchio, un punto P che disti D dalla base del robot ($D_{i-1}\le D\le D_i$) viene proiettato in un pixel che dista d_{i-1} o d_i dal centro del CCD. Nel caso peggiore (che, comunque, non si può mai realizzare) $D=D_{i-1}$ ma il punto P è proiettato in un pixel che dista d_i: in questo caso, la distanza stimata è $D_{est}=D_i$, quindi si commette un errore assoluto $e=D_{est}-D=D_i-D_{i-1}=a\cdot D_{i-1}=a\cdot D$ e un errore relativo pari ad a.

A questo punto, non resta che scegliere una funzione f^* che tenga conto delle esigenze di misura. Prima, però, è opportuno capire come la scelta della funzione f^* influisca sulla precisione della misura delle distanze degli oggetti dal robot.

2.3.1 Calcolo dell'errore commesso nella misura di distanze

Per valutare la posizione rispetto al robot di un punto *P* giacente sul piano di riferimento sono necessarie due informazioni: l'azimut j e la distanza d_{OP} di tale punto dal robot.

In [10] viene dimostrato come la prima informazione possa essere ottenuta, data la simmetria rotazionale dello specchio, dalle coordinate (*x*,*y*) della proiezione *P'* del punto *P* sul piano del CCD; in particolare si ha:

$$j = \arctan\left(\frac{y}{x}\right)$$

Come già notato, non si dispone delle coordinate (*x*,*y*) ma solamente delle coordinate (x_d,y_d) del pixel p tale che $P' \in$ p. L'errore e_j commesso nella stima di j, a causa di questa discretizzazione, ha un valore massimo che diminuisce all'aumentare della distanza d_{pixel} di p dal centro del CCD.

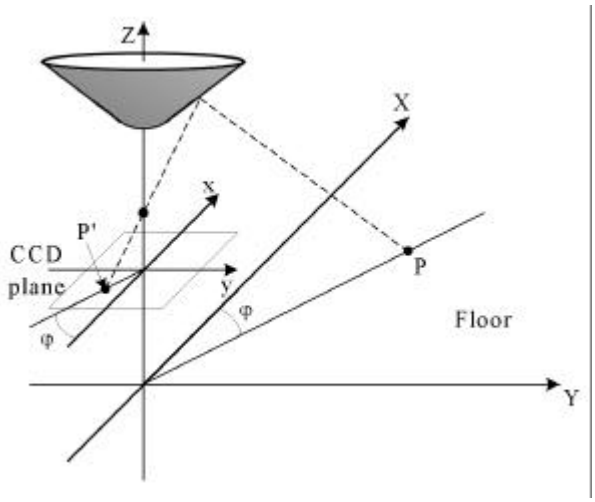


Fig. 2.7 Schema per il calcolo dell'azimut del punto *P* rispetto al robot.

Più complesso risulta, invece, ricavare la distanza d_{OP} a partire da *P'*, supponendo ovviamente di conoscere la geometria dello specchio. Però, dal procedimento di progetto descritto, la funzione f^* è nota, perciò:

$$d = f^{*}(d_{OP}), \quad \text{posto } d = \sqrt{x^2 + y^2} \quad P' : (x, y)$$

Se tale funzione risulta invertibile, si può porre:

$$d_{OP} = f^{*-1}(d)$$

Il problema principale di questo tipo di misurazioni è legato al fatto che, dall'immagine fornita dalla telecamera, non si possono ricavare le coordinate (x,y) di P', ma solo la posizione del pixel p tale che $P' \in \mathsf{p}$. Poiché tale pixel ha una dimensione finita A, si può risalire ad un insieme di possibili distanze D ottenibili dalla relazione

$$\mathsf{D} = f^{*-1}(\mathsf{d}), \quad \text{posto } \mathsf{d} = \left\{ \sqrt{x^2 + y^2} \mid (x, y) \in \mathsf{A} \right\}$$

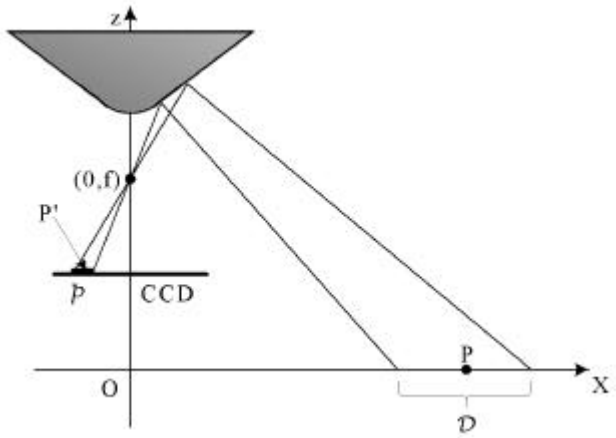


Fig. 2.8 Illustrazione di come l'insieme di punti D venga mappato in un unico pixel p.

Per avere una stima della distanza d_{OP}, si deve costruire una tabella che associ a ciascun pixel p dell'immagine una distanza D_{p}: per ogni punto P tale che $P' \in \mathsf{p}$, si approssima $d_{OP} \cong D_{\mathsf{p}}$. In questo modo, si commette un errore $e_{d_{OP}}$ il cui valore massimo è dato da:

$$e_{d_{OP}} = \max_{d \in \mathsf{D}} \left(d - D_{\mathsf{p}} \right)$$

In conclusione, conviene scegliere D_{p} in modo da minimizzare $\forall \mathsf{p}$ il valore massimo dell'errore commesso; questo può essere facilmente ottenuto fissando D_{p} pari al valore medio dell'insieme D.

A questo punto, può essere utile capire con che precisione si possono stimare le coordinate degli oggetti osservati, rispetto ad un sistema di riferimento solidale al robot e giacente sul piano di riferimento. A partire dalla distanza stimata D del punto P dal robot e dalla stima dell'azimut j si possono ricavare le coordinate P:($x=D\cdot\cos j$,$y=D\cdot\operatorname{sen} j$). Sfruttando la relazione dell'errore e_x, commesso sulla stima della variabile $x=f(x_1,\ldots,x_n)$, a partire da e_{x_1},...,e_{x_n}

$$e_x = \frac{\partial f}{\partial x_1} \cdot e_{x_1} + \ldots + \frac{\partial f}{\partial x_n} \cdot e_{x_n}$$

si può trovare che:

$$e_x = \cos j \cdot e_D - D \cdot \sin j \cdot e_j$$
$$e_y = \sin j \cdot e_D - D \cdot \cos j \cdot e_j$$

2.3.2 Descrizione dettagliata dell'algoritmo di progettazione

Per la progettazione dello specchio secondo la tecnica descritta all'inizio di questo paragrafo è stato realizzato uno script Matlab, programma scelto per la sua efficienza e versatilità in applicazioni matematiche.

Per un corretto funzionamento, prima dell'esecuzione del programma occorre che l'utente inserisca alcuni parametri geometrici del sistema, riguardanti sia lo specchio da progettare sia la telecamera impiegata per inquadrarlo. Questi parametri sono, di seguito, descritti in dettaglio.

Parametri geometrici del robot:

- `robotMaxDim` – diametro del cerchio che circoscrive il robot, in mm; qualora il robot avesse base circolare è sufficiente inserire il diametro della stessa, mentre se il robot è a base rettangolare si devono fornire le misure dei due lati
- `robotHeight` – altezza del corpo del robot, in mm

Queste due misure servono per determinare quale sia, a livello del pavimento, il diametro del cerchio che la base del robot proietta verso lo specchio, ovvero il diametro del cerchio che lo specchio dovrà far collassare nel centro dell'immagine.

Parametri geometrici della telecamera:

- `f` – lunghezza focale della lente, in mm
- `h` – altezza da terra del fuoco della lente, in mm
- `CCDWidth` – dimensione maggiore del CCD, in mm
- `CCDHeight` – dimensione minore del CCD, in mm

Parametri dell'immagine da acquisire:

- `imageWidth` – dimensione orizzontale dell'immagine, in pixel
- `imageHeight` – dimensione verticale dell'immagine, in pixel
- `distMax` – massima distanza sul pavimento, in mm, che si vuole vedere attraverso il sensore omnidirezionale

Parametri geometrici dello specchio:

- `diamMirror` – diametro dello specchio, in mm
- `step` – passo di integrazione sul CCD; questo parametro incide sul numero *N* di elementi di cui sono composti gli insiemi *d* e *D* e, di conseguenza, sulla "risoluzione" del profilo, ovvero sulla distanza media tra due punti successivi del profilo stesso. Questo valore è, di norma, imposto da vincoli tecnologici del costruttore

Il profilo dello specchio viene rappresentato da una serie di punti (`xProfile,zProfile`) nel sistema di riferimento (*x*,*z*) illustrato nella prossima figura; questi punti, interpolati linearmente, forniscono il profilo dello specchio, la cui rotazione attorno all'asse *z* fornisce lo specchio vero e proprio.

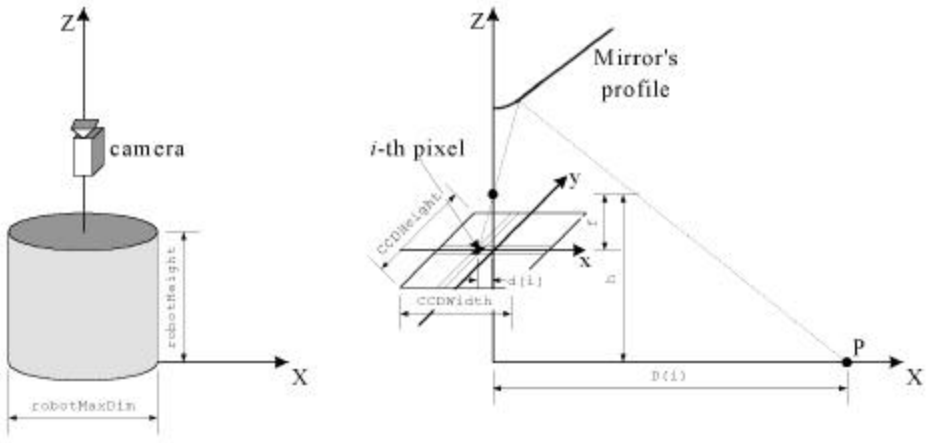


Fig. 2.9 Rappresentazione di alcuni dei parametri geometrici utilizzati dall'algoritmo.

Per la simmetria rotazionale del visore, ci si può limitare alla risoluzione di un problema bidimensionale, considerando una sezione verticale. La prima cosa da fare è costruire i vettori *d* e *D*, entrambi con cardinalità $N = \left(\frac{\texttt{imageHeight}}{2} - 1 \right) \cdot \texttt{step} + 1$. Il valore *d*(*i*) coincide con la distanza dal centro del CCD dell'*i*-esimo pixel (*i*=1,…,*N*), mentre il valore *D*(*i*) (*i*=1,…,*N*) rappresenta la distanza del punto *P*, sul piano di riferimento, che si desidera venga mappato dall'insieme telecamera-specchio nell'*i*-esimo pixel dell'immagine.

Lo specchio deve proiettare la corona circolare, sul piano di riferimento, di raggio interno *D*(1) e raggio esterno *D*(*N*) in un cerchio di raggio `imageHeight`/2 sul piano immagine. *D*(1) viene calcolato in modo tale da evitare la visione del corpo del robot, mentre i valori successivi si costruiscono ricorsivamente: `D(i)=D(i-1)+D(i-1)*alpha`, fintanto che non si raggiunge `distMax` (per `i`=`imageHeight`/2); all'interno di questi punti si va a riempire, con distribuzione uniforme, fino a raggiungere la cardinalità *N*. Il coefficiente moltiplicativo `alpha`,

ovvero il massimo errore relativo tollerato nelle misure di distanze sul piano di riferimento, viene determinato grazie alla seguente relazione:

$$a = \frac{\texttt{imageHeight}}{2}\sqrt{\frac{D_{MAX}}{D_{MIN}}} - 1$$

Ora si dispone di tutte le informazioni per progettare il profilo dello specchio di misura grazie al seguente algoritmo.

1. [i=0] Il primo tratto del profilo passa per il punto P_0=(0,z_0) ed ha tangente pari a $\tan\left(\frac{\arctan\left(\frac{D(1)}{z_0}\right)}{2}\right)$.

   Si definisce r_2:`a_r2*x+b_r2` come la retta che rappresenta questo primo tratto.

2. [i=i+1] Si costruisce r_1:`a_r1*x+b_r1` come la retta passante per il punto di fuoco (0,`f`) e per il punto (-d(i),`h-f`) sul CCD.

   Si determina P_i:(`xProfile(i)`,`zProfile(i)`) come punto di intersezione tra le rette r_1 e r_2.

   Si costruisce r_3:`a_r3*x+b_r3` come la retta passante per i punti P_i e (D(i),0).

   Dato che r_1 e r_3 individuano il percorso che deve seguire la luce per congiungere il punto a distanza d(i) sul CCD con il punto a distanza D(i) sul piano di gioco, si può calcolare la tangente del profilo nel punto P_i utilizzando le leggi dell'ottica.

   Si costruisce la retta r_2 passante per P_i e tangente al profilo nello stesso punto.

   Se $i \neq N$, si ripete il punto 2.

Una volta calcolati tutti i punti del profilo, questi vengono salvati su file in due formati: uno adatto per essere inserito in un modello POV-Ray al fine di poterne simulare il funzionamento (paragrafo 2.4), e l'altro per l'utensile a controllo numerico utilizzato per la realizzazione vera e propria dello specchio (paragrafo 2.5).

Infine, vengono calcolati i dati relativi alla risoluzione con cui è stato calcolato il profilo e ne viene presentata a video una rappresentazione in figura.

2.4 Verifiche in simulazione

A questo punto della progettazione, è giunto il momento di decidere i valori dei parametri dell'algoritmo descritti, sulla base delle caratteristiche e del task del robot che deve utilizzare lo specchio.

Prima, però, è opportuno osservare che la scelta dei parametri non è completamente libera. In particolare, non è possibile porre lo specchio ad un'altezza troppo elevata dalla base superiore del corpo del robot, e quindi da terra, perché la rigidità del supporto di plexiglas su cui è fissato lo specchio diminuisce all'aumentare della sua altezza, con conseguente aumento delle traslazioni relative tra telecamera (solidale al robot) e specchio; inoltre, siccome lo specchio può risultare piuttosto pesante, si rischierebbe di alzare troppo il baricentro del robot, a scapito della sua stabilità. D'altra parte, non risulta nemmeno possibile porre lo specchio troppo vicino al corpo del robot, perché questo impedirebbe la visione di oggetti molto vicini allo stesso. Per questi motivi, la scelta di h è limitata entro un certo intervallo, siccome l'altezza dello specchio da terra è data da f (che determina l'angolo visuale della telecamera, ed è fissato dall'ottica scelta), da h e dal diametro di cui si vuole lo specchio.

L'obiettivo finale è quello di costruire uno specchio di validità più generale possibile, ovvero che possa funzionare in modo ottimale su diversi tipi di robot e, in particolare, su un B21 della RWI (dell'Università di Ulm), con una base cilindrica di 25cm di raggio e alta 97cm, e su un robot auto-costruito dell'Università di Parma, con una base rettangolare di lati 37cm e 27cm e alta 40cm. Per questo motivo, si è pensato di calibrare i parametri dell'algoritmo su un ipotetico robot di misure intermedie (base cilindrica di raggio 20cm e alta 55cm), in modo da non penalizzare nessuna delle due architetture.

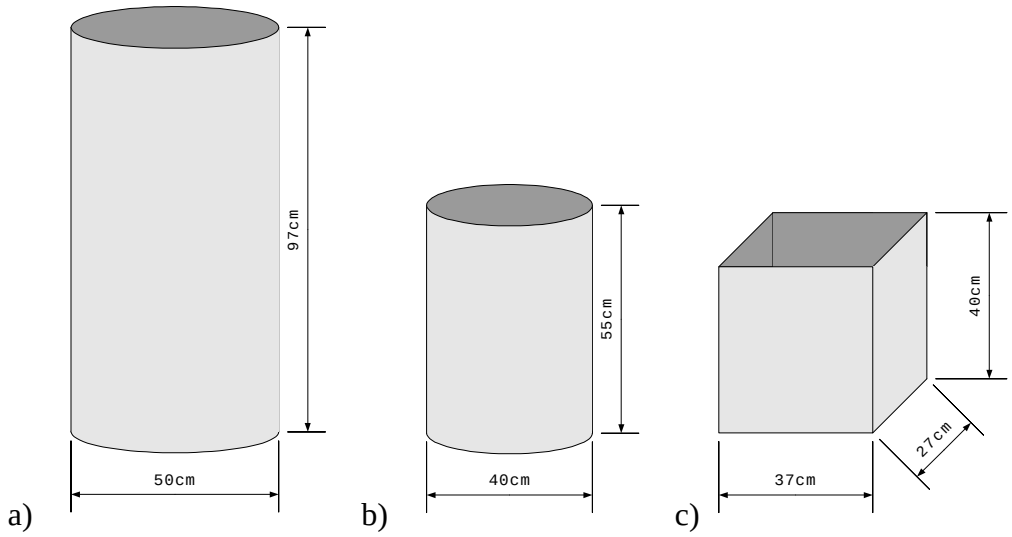


Fig. 2.10 Strutture e dimensioni dei tre robot utilizzati nella simulazione: a) B21; b) robot ipotetico; c) robot auto-costruito.

Ciò precisato, per la determinazione dei parametri ottimali, che garantiscano il miglior compromesso tra le varie architetture, si è scelto di valutare le diverse combinazioni attraverso una simulazione tridimensionale del sistema robot-sensore all'interno di un ambiente generico (una stanza di $9\text{m} \times 4\text{m}$, alta 3m). Questa simulazione, realizzata grazie al programma di ray-tracing POV-Ray in ambiente Linux, ha portato alla scelta dei parametri riportati nelle tabelle seguenti.

Parametri vincolati dal sistema:

`f`	3.8mm
`CCDWidth`	5.08mm
`CCDHeight`	3.81mm
`imageWidth`	512 pixel
`imageHeight`	384 pixel

Parametri scelti:

`robotMaxDim`	350mm
`robotHeight`	550mm
`h`	703.8mm
`diamMirror`	140mm
`distMax`	4000mm

Si può notare che `robotMaxDim` non corrisponde al diametro dell'ipotetico robot utilizzato in simulazione, ma è un po' inferiore, per ridurre l'effetto di perdita di risoluzione, dovuta all'assottigliamento degli oggetti, nel centro dell'immagine, ovvero vicino al robot.

Il parametro `step` non è stato ripreso perché, come detto, è dato da vincoli tecnologici del costruttore, perciò viene rimandata al paragrafo 2.5 una descrizione di come influisca sul profilo dello specchio.

Con queste scelte, si ottiene che la circonferenza sul piano di riferimento che viene fatta collassare nel centro dell'immagine ha raggio pari a

$$D_{MIN} = \frac{\texttt{robotHeight} \cdot \texttt{robotMaxDim}}{2 \cdot (z_0 - \texttt{robotHeight})} + \frac{\texttt{robotMaxDim}}{2} = 539.5mm\,.$$

Dalla formula per a vista al paragrafo 2.3.2, una volta nota la geometria del robot e quindi la minima distanza visibile, si può ricavare l'andamento di a al variare della massima distanza misurabile D_{MAX}.

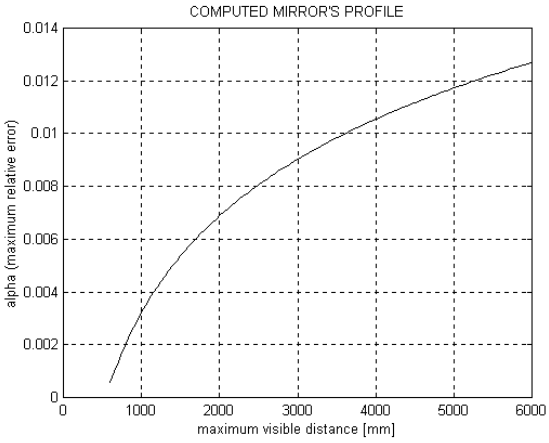


Fig. 2.11 Andamento di $a(D_{MAX})$, noto D_{MIN}.

Con il valore di D_{MAX} scelto, l'errore relativo nelle misure sul piano di riferimento vale a=0.0105.

Il profilo dello specchio corrispondente a questi parametri, così come il suo comportamento in simulazione sui tre diversi tipi di robot descritti, è visualizzato nelle seguenti figure.

Fig. 2.12 Profilo dello specchio ottenuto.

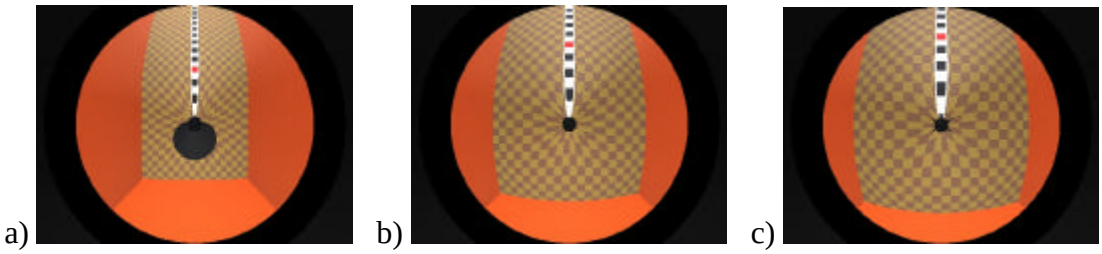


Fig. 2.13 Simulazione delle immagini ottenibili dai tre robot utilizzati: a) B21; b) robot ipotetico; c) robot auto-costruito.

Per avere un'indicazione delle misure e delle distanze ottenibili, si consideri che gli scacchi del pavimento hanno lati di 20cm, così come i quadrati neri posti in fila sulla striscia bianca, larga 30cm. Questi ultimi vengono utilizzati per la calibrazione prospettica del sensore (capitolo 3), motivo per cui sono stati introdotti nel modello *3D* dell'ambiente.

2.5 Realizzazione del prototipo

Dopo aver ottenuto buoni risultati in simulazione, si è potuti passare alla realizzazione effettiva dello specchio. Di questa fase si è occupata l'officina meccanica (Werkstatt) dell'Università di Ulm, che ha utilizzato un utensile a

controllo numerico per “scolpire” un blocco di metallo cilindrico, al fine di ottenere la forma dello specchio desiderata. L’utensile è stato programmato immettendone in memoria tutti i punti di cui è costituito il profilo dello specchio, con la seguente sintassi:

```
G1 X 0.000 Z -0.000
G1 X 0.288 Z -0.043
G1 X 0.360 Z -0.054
G1 X 0.432 Z -0.065
G1 X 0.504 Z -0.076
...
```

Ogni riga inizia con il comando `G1`, che indica alla macchina di applicare un’interpolazione lineare tra due punti successivi, e prosegue con le coordinate *x* e *z* del punto, considerando che il valore di *x* va indicato raddoppiato ed il valore di *z* deve decrescere a partire dal vertice dello specchio. Entrambe le coordinate (espresse in mm) possono essere date con una precisione massima di 3 cifre decimali e possono assumere valori sia positivi che negativi.

Come già anticipato, in questa fase è stato formalizzato anche il vincolo relativo al parametro `step`, vincolo derivante dalla dimensione della memoria dell’utensile, e quindi dal numero di punti che può contenere. Il valore massimo utilizzabile è 8, che significa avere un profilo di 1530 punti con una distanza media tra due successivi di 49.8μm e una varianza di 0.065μm. Ovviamente, potendo usufruire di una macchina con più memoria, si potrebbe aumentare il valore di `step` e quindi avere una maggiore precisione nella realizzazione dello specchio, ma quello utilizzato ha dato, comunque, risultati più che buoni.

Un primo prototipo dello specchio è stato realizzato in alluminio, materiale scelto per la sua morbidezza e la facilità di lavorazione, ma non era soddisfacente in quanto troppo poco riflettente. Il risultato era un’immagine con un fastidioso effetto nebbia, che rendeva difficile l’elaborazione dell’immagine. Per questi motivi è stata prodotta una seconda versione in ottone ricoperto galvanicamente con nichel che, anche se di più difficile lavorazione e più pesante, garantisce una riflessione molto

migliore e più luminosa. Questo secondo modello è quello che è stato impiegato per lo studio e la realizzazione degli algoritmi di visione descritti nei prossimi capitoli.

Il montaggio del sistema visivo è stato completato fissando questo nuovo specchio sulla sommità di una struttura già esistente, costituita fondamentalmente da un cilindro in plexiglas, un ripiano circolare interno ad esso e due telecamere a CCD [13]. Nel prototipo, per la parte che costituisce il catadiottro, una delle due telecamere è fissata al centro del ripiano circolare ed è orientata verticalmente verso la specchio, in modo da far coincidere il suo asse ottico con l'asse dello specchio. Inoltre, il ripiano circolare interno possiede un grado di libertà, che gli permette di essere posizionato a diverse altezze. La seconda telecamera è, invece, posizionata all'esterno del cilindro di plexiglas ed orientata in modo da inquadrare l'area di fronte al robot. Sia la base del sistema che il ripiano dello specchio sono dotati di un metodo per l'orientamento della struttura costituito da quattro coppie vite-molla, utilizzate come regolatori di posizione tra due ripiani di plexiglas, in modo che risultino paralleli al piano del pavimento, ottenendo anche la verticalità dell'asse ottico del catadiottro.

La seconda telecamera, infine, è posizionata lateralmente rispetto all'asse ottico del catadiottro, in modo da ottenere una maggiore disparità stereo e quindi una più facile estrazione delle informazioni tridimensionali contenute nella coppia di immagini.

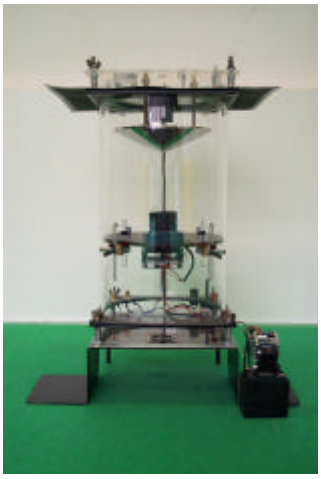


Fig. 2.14 Il sistema visivo completo.

Capitolo 3

Calibrazione prospettica di un sensore visivo

3.1 Descrizione del metodo di calibrazione prospettica

Le due principali non-idealità che affliggono le immagini acquisite da telecamere tradizionali sono la distorsione delle lenti e l'effetto prospettico.

L'ottica di tutte le telecamere produce una distorsione delle immagini, generalmente distinta in una componente radiale e una tangenziale, quest'ultima di norma trascurabile. In quasi tutte le telecamere con angoli di apertura non superiori agli 80°, questo effetto di distorsione è comunque molto limitato e trascurabile in un primo modello grossolano dell'ottica.

Tralasciando la distorsione delle lenti (adottando quindi il cosiddetto modello pin-hole per l'ottica della telecamera), l'effetto prospettico è il processo di proiezione degli oggetti tridimensionali della scena reale nelle corrispondenti forme sulla superficie di formazione dell'immagine. In particolare, questa proiezione si chiama geometrica planare, ed è caratterizzata dal fatto che avviene su un piano e le linee di proiezione sono delle rette. La retta di proiezione di ogni punto della scena dello spazio oggetto o passa per uno specifico punto detto centro di proiezione e da qui interseca il piano di proiezione (*proiezione prospettica*), o è parallela ad un asse di proiezione unico per tutti i punti (*proiezione ortografica*).

La proiezione prospettica è un processo che determina la perdita di molte informazioni: da un mondo tridimensionale si passa ad una sua descrizione bidimensionale, quindi si perdono le informazioni su dimensione, profondità e

posizione relativa degli oggetti. Si tratta di un fenomeno non lineare e non reversibile, inoltre va a sommarsi a quello della distorsione introdotta dall'ottica.

Sempre considerando un sistema ottico non distorcente ed una superficie di formazione dell'immagine planare, la relazione matematica tra le coordinate di un punto nello spazio osservato e quelle del suo corrispondente nell'immagine generata (ossia le equazioni della proiezione prospettica) è piuttosto elementare:

$$x = f \cdot X/Z$$
$$y = f \cdot Y/Z$$

In questa, X, Y e Z sono le coordinate nello spazio oggetto riferite ad un sistema di riferimento centrato nel centro ottico O, x e y le coordinate sul piano dell'immagine F, e f la distanza focale del sensore, cioè la distanza tra il centro ottico del sistema di lenti e il piano di formazione dell'immagine (oO). Questa relazione permette, conoscendo la distanza focale, di determinare come ogni punto dello spazio oggetto vada a generare un punto dell'immagine ma, come osservato precedentemente, questa relazione non è invertibile.

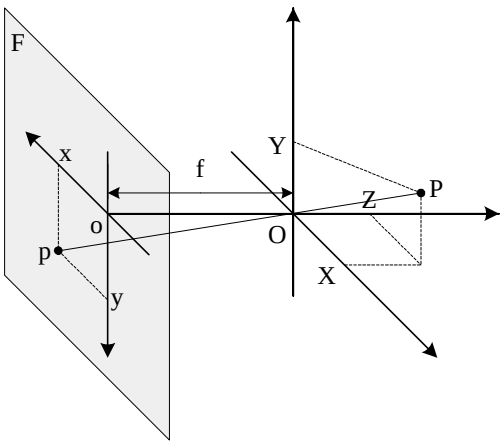


Fig. 3.1 Sistema di coordinate utilizzato nelle equazioni della proiezione prospettica.

Nel processo descritto, l'immagine generata sulla superficie di formazione è continua, quindi, per ottenere un'immagine digitale, è necessario effettuarne un

campionamento, e questo generalmente avviene tramite un CCD. Questo campionamento determina un ulteriore deterioramento della qualità dell'immagine, in termini sia di risoluzione, che di quantizzazione dei livelli di intensità luminosa.

Un passo fondamentale verso il recupero di almeno una parte delle informazioni perse durante il processo di formazione dell'immagine è la cosiddetta calibrazione del sensore visivo. Nel caso più generale e classico, questa consiste nel ricavare le caratteristiche ottico-geometriche interne della telecamera, inerenti al modello adottato per la telecamera stessa (dette *parametri intrinseci*), e quelle esterne, cioè l'orientazione ed il posizionamento della telecamera nello spazio (dette *parametri estrinseci*). A partire da una precisa conoscenza di questi parametri e dall'utilizzo di informazioni sull'ambiente di lavoro, o derivanti da sistemi di visione stereoscopica, o derivanti, ad esempio, da sequenze di immagini, risulta possibile recuperare informazioni sulla profondità, la distanza relativa e la forma degli oggetti. Insomma, si può cercare di aggiungere tridimensionalità alla conoscenza che si ha della scena osservata. Tra le innumerevoli tecniche proposte in letteratura per la calibrazione di telecamere, le principali sono probabilmente quelle di Tsai [14] e Faugeras [15].

Un modo di impiegare le informazioni sul sistema ottico ricavate dalla sua calibrazione è quello che va sotto il nome di prospettiva inversa (IPM, *Inverse Perspective Mapping*). Come descritto nello studio del processo di formazione dell'immagine, le immagini fornite da una qualsiasi telecamera danno una rappresentazione molto distorta della realtà: si ha una componente di distorsione legata all'ottica della telecamera, e soprattutto una distorsione geometrica legata al processo di proiezione prospettica. La prospettiva inversa è una tecnica che vuol rimappare le informazioni presenti nell'immagine I su una nuova immagine rettificata R, che dia una rappresentazione non distorta di ciò che appare su un qualunque piano P preso come riferimento. Questa mappatura può essere descritta come segue.

Date le coordinate (x_I,y_I) del pixel dell'immagine I che rappresenta il punto (X,Y) del piano P, la sua intensità $I(x_I,y_I)$, le coordinate (x_R,y_R) del pixel

dell'immagine rettificata *R*, che rappresenta lo stesso punto (*X*,*Y*) del piano *P*, e la sua intensità $R(x_R,y_R)$, a parte traslazioni dei sistemi di riferimento,

$$x_R = round\left(X \middle/ sens\right)$$
$$y_R = round\left(Y \middle/ sens\right)$$
$$R(x_R(X), y_R(Y)) = I(x_I(X,Y), y_I(X,Y))$$

dove *sens* è la lunghezza del lato del quadrato del piano *P* rappresentata da un pixel dell'immagine *R*. La relazione che la calibrazione deve individuare è $(x_I(X,Y),y_I(X,Y))$.

Nel caso della navigazione di robot, in cui il piano di riferimento è tipicamente il pavimento, il risultato di questa operazione è la generazione di una mappa dell'ambiente a livello pavimento: questa immagine sembra una fotografia scattata dall'alto verso il basso ad una altezza considerevole. Tutti gli elementi della scena che non si trovano sul piano di riferimento risultano ulteriormente distorti. E' chiaro però che, riuscendo a distinguere (sia tramite conoscenza a priori sull'ambiente di lavoro sia tramite l'impiego di visione stereo) tra ciò che giace e ciò che non giace su questo piano, si ottiene informazione su profondità, posizione relativa e distanza fra gli oggetti nello spazio osservato.

In relazione alla specifica applicazione, il piano di riferimento può essere diverso: oltre a quello del pavimento, si può, ad esempio, pensare di avere piani orizzontali a diverse altezze per studiare la posizione di oggetti specifici dalle dimensioni note, o piani verticali per l'analisi di pareti [16], ecc.. Le applicazioni possono essere molteplici: dalla navigazione indoor e outdoor, alla ricostruzione ed analisi di specifici piani di una scena (ad esempio pareti di palazzi), all'auto-localizzazione in ambienti strutturati. I vantaggi di ottenere una tale descrizione dell'ambiente sono ovvi e, infatti, negli ultimi dieci anni si sono sviluppate molte applicazioni sfruttando questo strumento.

L'applicazione di questa tecnica a campi quali il riconoscimento di ostacoli e la determinazione dello spazio libero trova le sue origini in un progetto sviluppato nel

1991 da Mallot [17]. Da allora, questa tecnica è stata utilizzata in numerosi progetti nel campo sia della navigazione indoor, sia di veicoli su strada.

Si deve, però, notare come non sia sempre possibile conoscere con precisione tutti i parametri intrinseci ed estrinseci del sensore. D'altro canto, per applicare l'IPM, la conoscenza di tutti questi non è strettamente necessaria. L'obiettivo da raggiungere è la determinazione di una relazione tra la posizione dei pixel nell'immagine originale e la relativa posizione nell'immagine rettificata (quindi, sul piano preso come riferimento nello spazio oggetto). Conoscendo tutti i parametri intrinseci ed estrinseci del sensore ottico, questa relazione è una trasformazione geometrica relativamente semplice (si veda a riguardo il paragrafo 3.2.1). In caso contrario, è possibile cercare questa relazione attraverso tecniche di calibrazione più empiriche che, come descritto nei successivi paragrafi, presentano alcuni vantaggi operativi in sede di calibrazione, ma anche svantaggi dovuti alla necessità di operare una nuova calibrazione completa ad ogni modifica, strutturale o di componentistica, del sensore visivo.

3.2 Tecniche di calibrazione del sensore omnidirezionale

Un problema centrale affrontato nella prima parte della sperimentazione effettuata è quello della calibrazione del sensore catadiottrico.

Questa fase, come visto, ha l'obiettivo finale di ottenere una corrispondenza tra la posizione dei punti del piano di riferimento (in questo caso il pavimento) e la posizione dei pixel nell'immagine acquisita dal catadiottro stesso. In questo modo, sapendo da quale pixel ogni punto del piano viene rappresentato, è immediata l'operazione di inversione della prospettiva, per ottenere una visione di tipo bird-eye.

Come già precedentemente notato, un obiettivo primario di tutto il lavoro svolto è la generalità, ossia l'applicabilità delle routines e degli algoritmi a nuove e diverse situazioni. Per questo motivo, il software realizzato è strutturato in modo da poter essere esteso facilmente con nuovi algoritmi e soluzioni al problema della calibrazione, permettendo un facile adattamento e impiego con i più svariati tipi di sensori ottici.

Nel corso della ricerca svolta sul sensore catadiottrico sono state progettate ed impiegate, principalmente, tre tecniche di calibrazione differenti: una basata su un modello geometrico del sistema, una seconda, più empirica, basata sull'impiego di campioni di corrispondenza punto-pixel e sull'interpolazione tra di essi, una terza che è una versione ridotta della seconda, quindi applicabile in una gamma più ristretta di situazioni.

3.2.1 Tecnica di calibrazione geometrica

Questa tecnica di calibrazione utilizza un modello molto simile a quello classico impiegato da Tsai [18]. Tramite la costruzione di un modello geometrico del sensore visivo, fondato sulla conoscenza della posizione e dell'orientazione reciproca della telecamera e dello specchio, dei parametri intrinseci della telecamera, della forma dello specchio, della posizione e dell'orientazione del piano di riferimento per l'inversione prospettica, si può calcolare la relazione tra pixel dell'immagine e punti del piano di riferimento. Per la telecamera si utilizza un modello pin-hole, ipotizzando che la distorsione introdotta dall'ottica sia sferica e trascurando un'eventuale componente tangenziale. In particolare, tra i parametri intrinseci della telecamera, solo gli angoli di apertura e il coefficiente di distorsione radiale dell'ottica risultano essere rilevanti ai fini del modello.

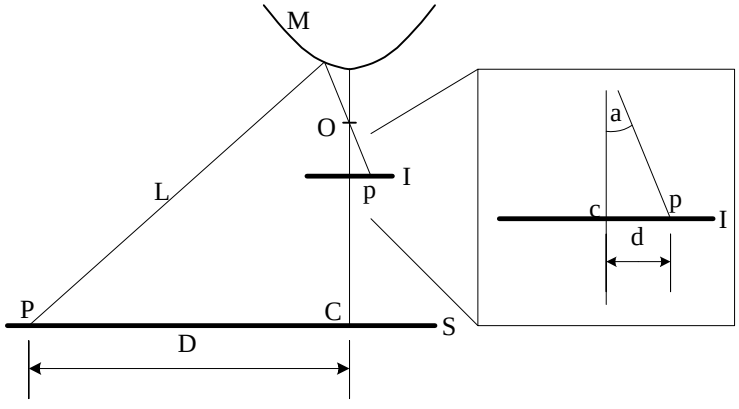


Fig. 3.2 Modello utilizzato nella calibrazione geometrica.

Con riferimento all'immagine precedente, il modello utilizzato permette di

ricostruire il percorso che il fascio di luce *L* (rappresentato, per semplicità, con una linea), incidente sull'apertura dell'ottica *O*, segue per giungere sul piano di formazione dell'immagine *I*, passando attraverso la sua incidenza e riflessione sullo specchio *M*, fino a risalire al punto della superficie del piano di riferimento *S* da cui esso è stato emesso. Si crea, quindi, una relazione tra il punto in posizione *p* sul piano di formazione dell'immagine (legato alla direzione di incidenza del fascio luminoso sull'ottica ed al coefficiente di distorsione radiale di questa) ed il punto *P* del piano di riferimento. Questo metodo si basa sul presupposto che lo specchio, e più in generale il catadiottro, abbia un asse di simmetria verticale coincidente con l'asse ottico. In questo modo, ad ogni punto in posizione *p* distante un valore *d* specifico dall'incidenza dell'asse ottico sul piano *I*, corrisponde un punto *P* distante un valore *D* specifico dal punto di incidenza dell'asse ottico sul piano *S* (perpendicolare a questo). Si crea quindi una funzione *D(d)* che permette di rimappare sul piano *S* l'immagine generata su *I*.

Di seguito, è illustrato nel dettaglio l'algoritmo utilizzato.

1. Dato un punto *p* sul piano di formazione dell'immagine (con riferimento alla figura precedente) di coordinate (x_p,y_p) riferite ad un sistema cartesiano *xoy* centrato nel punto *c* di incidenza dell'asse ottico della telecamera sul piano *I* ed espresse in unità di pixel, se ne calcola la distanza *d* da *c* stesso:

$$d = \sqrt{x_p^2 + y_p^2}\ .$$

2. Si calcola la direzione di incidenza sul centro ottico del raggio luminoso che ha generato, nell'immagine, il punto *p*. Per far questo, si deve prima recuperare l'effetto della distorsione radiale (di coefficiente *K*) sul valore di *d*, calcolando d_1:

$$d_1 = d \cdot \left(1 + K \cdot d^2\right).$$

3. Si calcola l'angolo di incidenza *a*:

$$a = \arctan\left(d_1 / f\right)$$

dove f è la distanza focale effettiva della telecamera (ossia il segmento cO) espressa in unità di pixel e definita come:

$$f = \frac{d_I / 2}{\tan(a_o / 2)}$$

dove a_o è l'apertura angolare orizzontale dell'ottica e d_I la dimensione orizzontale in pixel dell'immagine (ma potrebbero essere anche quelle verticali).

4. Conoscendo l'altezza h del punto O dal piano di riferimento, si calcola l'equazione $f(X,Y)$ della retta passante per O con direzione indicata dall'angolo a e riferita ad un sistema di riferimento $XYZC$ centrato in C con asse Z verticale e coincidente con l'asse ottico della telecamera ed assi X e Y paralleli a x e y, ed orientati opportunamente. Dato che, in realtà, lo specchio è un solido di rotazione e quindi il sistema è simmetrico rispetto al suo asse centrale, ci si può limitare allo studio dell'equazione $f(D)$, con D distanza sul piano di riferimento dal punto C:

$$f(D) = h + \tan(a) \cdot D.$$

5. Conoscendo il profilo esatto dello specchio, ovvero la funzione radiale $Z=g(D)$, dalla soluzione del sistema delle due equazioni $f(D)$ e $g(D)$ si può ricavare il valore D_P di D tale che $f(D_P)=g(D_P)$, che rappresenta la distanza dall'asse ottico dell'incidenza del fascio luminoso L sullo specchio M. A partire dal valore della derivata $g'(D)$ di $g(D)$ in D_P si può ottenere l'angolo di inclinazione $b = \arctan(g'(D_P))$ dello specchio nel punto di incidenza, e si può calcolare l'angolo di salita s con cui il fascio luminoso è giunto allo specchio a partire dal punto P:

$$s = \frac{p}{2} - a - 2b.$$

6. Conoscendo *s*, si può calcolare la distanza *D* del punto *P* dal centro *C*:

$$D = D_P + \frac{g(D_P)}{\tan(s)}.$$

7. Infine, le coordinate (X_P,Y_P) di *P* sono semplicemente:

$$X_P = x_p \cdot \frac{D}{d}$$
$$Y_P = y_p \cdot \frac{D}{d}$$

8. La tabella di look-up di inversione prospettica associa al pixel dell'immagine rettificata che descrive il punto del piano di riferimento di coordinate (X_P,Y_P), il pixel dell'immagine acquisita di coordinate (round(x_p),round(y_p)).

In conclusione, ciò che si deve conoscere per applicare l'algoritmo è: l'equazione radiale del profilo dello specchio $g(D)$ (che include anche l'altezza di questo dal piano di riferimento), il coefficiente di distorsione radiale dell'ottica *K*, la dimensione in pixel delle immagini acquisite d_I, l'angolo di apertura dell'ottica a_o, l'altezza *h* del centro ottico *O* dal piano di riferimento.

Questo algoritmo possiede il vantaggio di suddividere tutti i componenti del catadiottro e di analizzarli separatamente, in modo tale che la modifica di uno di essi o la riorganizzazione del sistema nello spazio comporti solamente la necessità di misurare i nuovi valori assunti dai parametri del modello (ad esempio l'altezza dello specchio dal suolo, il coefficiente di distorsione dell'ottica, l'equazione della forma dello specchio) per ottenere una nuova calibrazione del sistema. D'altro canto, non è sempre possibile conoscere con precisione tutti i parametri, inoltre il sistema è molto

sensibile anche a piccoli errori in alcuni di questi.

Le difficoltà incontrate sono, sostanzialmente, legate al fatto di non conoscere con sufficiente precisione la forma dello specchio impiegato. Il primo prototipo di specchio su cui sono stati applicati gli algoritmi di calibrazione descritti in questo paragrafo non è quello di cui è stata trattata la progettazione nel capitolo precedente, ma quello descritto in [19], già in dotazione al laboratorio. Questo è conico a vertice sferico: il cono ha un angolo di apertura di 117° ed una base di raggio 8.9cm, mentre la sfera ha un raggio di 6.6cm. I due solidi s'incontrano ad una distanza di 3.5cm dall'asse del cono, perciò lo specchio raggiunge un'altezza complessiva di 4.3cm. Basandosi sul profilo deciso per questo specchio in fase di progetto, si è visto che questo corrisponde solo parzialmente alla sua forma effettiva. Non solo questo non risulta, a causa dell'imprecisione della tecnologia di realizzazione impiegata, esattamente rispondente al progetto, ma la sua superficie risulta abbastanza irregolare. In realtà, la tecnologia di lavorazione dell'acciaio, utilizzata per la realizzazione dello specchio, ha introdotto delle imprecisioni, sia per quanto riguarda il profilo dello specchio stesso che per la sua regolarità superficiale. Si sono potute, inoltre, verificare imprecisioni dal punto di vista della simmetria assiale (ossia la sezione perpendicolare all'asse può non essere perfettamente circolare ma piuttosto leggermente ellittica) e delle irregolarità locali a livello della superficie riflettente.

Si è, perciò, pensato alla sostituzione dello specchio con uno di diversa concezione, quale quello descritto nel precedente capitolo, sia per poter personalizzare il suo profilo nel modo desiderato, sia perché la differente tecnologia di realizzazione ha permesso di eliminare molte delle cause di distorsione citate. Rimane comunque il problema che il profilo non è definito da una funzione continua, ma da una sua discretizzazione; inoltre, l'utensile impiegato nella tornitura del blocco di metallo grezzo introduce, comunque, un certo livello di imprecisione e non è garantito che la copertura galvanica abbia lo stesso spessore ovunque. Dunque, anche in questo caso, il profilo dello specchio non è esattamente uguale a quello previsto in fase di progetto, anche se vi si discosta molto meno rispetto a quello iniziale.

Questi problemi richiedono, per la calibrazione del catadiottro, l'utilizzo di una

tecnica almeno parzialmente empirica per il recupero delle distorsioni legate alle irregolarità osservate. Questo algoritmo rimane pur sempre un buono strumento, efficace nel caso di utilizzo di specchi realizzati con tecnologie ad alta precisione.

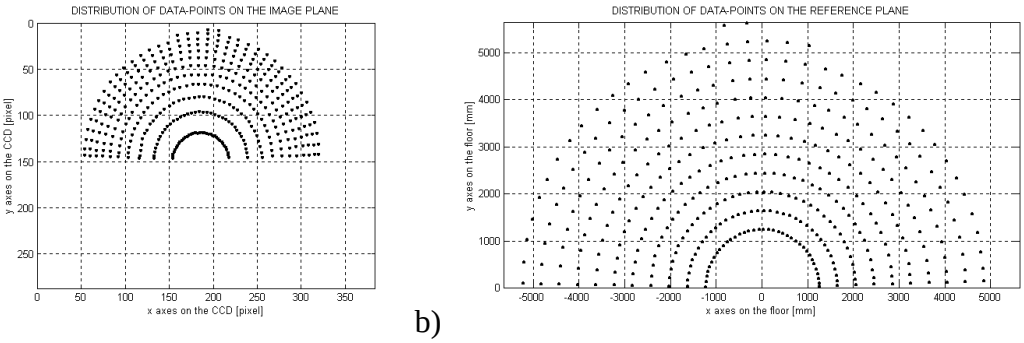
3.2.2 Tecnica di calibrazione empirica completa

Invece di creare un modello articolato che descrive separatamente tutte le parti del sistema fisico (la telecamera con i suoi parametri intrinseci, lo specchio e la sua forma, il posizionamento reciproco di questi, il piano di riferimento, ecc.), per cercare di ovviare ai problemi visti nel paragrafo precedente si è sviluppato un algoritmo che, anziché basarsi su un modello analitico, si fonda su basi strettamente empiriche. I passi principali di questo algoritmo, nella versione più generale, sono riportati di seguito.

- Si definiscono due sistemi di riferimento cartesiani bidimensionali: uno (*xoy*) posizionato sul piano dell'immagine *F*, con unità di misura in pixel, l'altro (*XOY*) posizionato sul piano di riferimento per l'inversione prospettica *S*, con unità di misura pari a *sens*, che corrisponde alla lunghezza (espressa in mm) del lato del quadrato sul piano di riferimento rappresentato da un pixel nell'immagine con prospettiva rimossa.
- Si raccoglie un insieme W di N campioni empirici costituiti da coppie di punti (p_i,P_i), con p_i appartenente al piano dell'immagine F e P_i al piano di riferimento S, tali che il pixel p_i dell'immagine sia una rappresentazione di un intorno del punto P_i. Le coppie di campioni in W devono essere rappresentativi di tutte le regioni di interesse del campo visivo, ed essere concentrate più densamente nelle regioni di maggior importanza: la precisione locale dell'inversione prospettica è, infatti, proporzionale a questa densità.
- Per ognuno dei punti G della porzione del piano S che viene rappresentata dall'immagine sul piano F, tale che $G \notin P_i$ per ogni $(p_i,P_i) \in W$, si opera un'interpolazione bidimensionale attraverso spline bicubiche per trovare la posizione del punto g da associare a G. In questo modo si riesce a rimappare tutta l'area di interesse dell'immagine sulla superficie S.

L'implementazione di questa tecnica può essere semplificata se i punti P_i dei campioni di *W* vengono scelti su un reticolo regolare applicato al piano *S*, quale può essere un reticolo a maglie quadrate o rettangolari o, come nel caso in questione, lungo linee direttrici radiali passanti per il punto di incidenza dell'asse ottico sul piano *S*. Scegliere in questo modo i punti campione risulta particolarmente efficace quando il sistema presenta un asse di simmetria radiale, incidente perpendicolarmente nel centro *O* del piano *S*.

Questo posizionamento dei punti campione P_i ne determina una densità decrescente dalla regione centrale verso l'esterno, e ciò consente di ottenere una inversione prospettica con precisione massima nelle vicinanze del robot e decrescente allontanandosi. Il numero di rette direttrici dalle quali si estraggono i punti campione dipende, chiaramente, dalla precisione con cui si vuole calibrare il sistema: all'aumentare del loro numero, la calibrazione risulta più robusta nell'affrontare eventuali imperfezioni della simmetria rotazionale. In questo caso, si è scelto di utilizzare campioni presi da 37 direttrici equidistanti angolarmente, per calibrare la metà anteriore dell'immagine omnidirezionale, ovvero si è campionata una retta ogni 5°, ma in casi normali può essere sufficiente anche un numero inferiore di direzioni.



a) b)

Fig. 3.3 Distribuzione dei punti campione raccolti sul piano immagine (a)) e dei corrispondenti punti sul piano di riferimento (b)).

Non in tutte le direzioni si è potuto rilevare lo stesso numero di punti in quanto, nella configurazione a cui ci si riferisce, l'asse ottico del sensore catadiottrico non corrisponde all'asse centrale della base del robot, ma è leggermente spostato, in virtù

della conformazione del robot stesso. Inoltre, dell'immagine omnidirezionale è stata raddrizzata solo la metà anteriore, perché, nella configurazione d'installazione, gran parte della metà posteriore dell'immagine omnidirezionale risulta coperta da una torretta con vari pulsanti e connettori necessaria al funzionamento del robot. Per questo motivo, la calibrazione di questa metà dell'immagine risulta impossibile e, in ogni caso, totalmente inutile perché non potrebbe dare contributi significativi a qualsiasi tipo di elaborazione.

Per quanto riguarda la rilevazione delle coppie di campioni, questa è un'operazione che viene fatta con l'ausilio di un pattern di quadrati neri alternati regolarmente e allineati su una striscia di colore bianco.

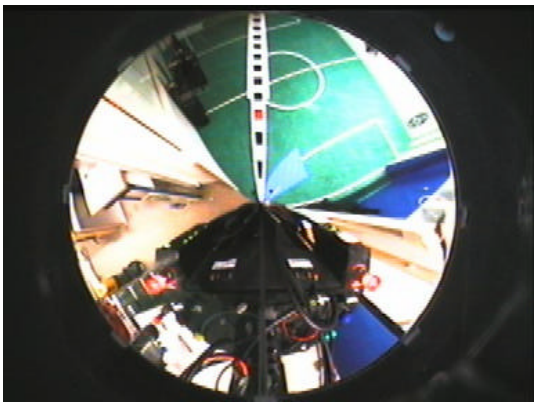


Fig. 3.4 Pattern utilizzato per la calibrazione empirica del catadiottro.

Questa rilevazione viene fatta automaticamente attraverso l'elaborazione di immagini che visualizzano il pattern in posizioni specifiche, come descritto nel paragrafo 3.4. Ciò rende più veloce ed efficiente il processo di calibrazione, caratteristica molto importante in quanto, indipendentemente dal posizionamento dei campioni, questa tecnica ha lo svantaggio di richiedere una nuova calibrazione completa ad ogni modifica del sistema visivo (ad esempio, della sua posizione rispetto al piano di riferimento).

Una volta rilevati i campioni empirici, come detto, per poter rimappare tutta l'area di interesse dell'immagine sulla superficie S si opera un'interpolazione bidimensionale [20].

Il problema dell'interpolazione consiste nello stimare il valore di una funzione $f(x)$ per un qualunque valore arbitrario x, "disegnando" una curva liscia che passi attraverso tutti i punti campione x_i. Nelle situazioni in cui è importante fornire la continuità delle derivate della funzione interpolante, conviene utilizzare le *spline* che garantiscono una buona regolarità. Una spline è un polinomio tra ogni coppia di punti tabulati, i cui coefficienti sono determinati, in parte, non localmente, a differenza delle funzioni interpolanti polinomiali. La non-località è necessaria per garantire la regolarità globale della funzione fino ad un certo ordine di derivata: le spline cubiche, che sono le più popolari, sono continue fino alla derivata seconda. Le spline, inoltre, tendono ad essere più stabili delle polinomiali, con meno possibilità di oscillazioni strane tra i punti tabulati. L'interpolazione può essere effettuata, come nel caso di questa calibrazione, su più di una dimensione, come sequenza di più interpolazioni uni-dimensionali.

Nel caso specifico, l'interpolazione è su funzioni bidimensionali, e si è deciso di applicare le spline bicubiche per ottenere una buona regolarità. Questo tipo di interpolazione prevede che i punti campione siano posizionati in una griglia rettangolare completa e non disposti casualmente, e viene ottenuta applicando globalmente spline cubiche uni-dimensionali: per interpolare un valore della funzione, si eseguono m spline uni-dimensionali lungo le m righe della tabella, seguite dall'applicazione di un'ulteriore spline uni-dimensionale lungo la nuova colonna creata. Per una migliore comprensione del metodo, si riporta anche la tecnica di interpolazione uni-dimensionale attraverso spline cubiche.

Dati i valori tabulati $y_i=y(x_i)$, $i=1,...,N$, si focalizzi l'attenzione sull'intervallo tra x_j e x_{j+1}. Un'interpolazione lineare nell'intervallo dà la funzione

$$y = A \cdot y_j + B \cdot y_{j+1}, \text{ dove } A = \frac{x_{j+1} - x}{x_{j+1} - x_j} \text{ e } B = 1 - A = \frac{x - x_j}{x_{j+1} - x_j}$$

La derivata seconda di questa è nulla negli intervalli tra i campioni ed indefinita o infinita in corrispondenza delle ascisse x_j. L'obiettivo delle spline cubiche è di fornire una funzione interpolante con derivata prima liscia e derivata

seconda continua sia all'interno di ogni intervallo che ai loro estremi. Supponendo di avere tabulati, oltre ai valori y_i, anche i campioni della derivata seconda della funzione (y_i"), all'interno di ogni intervallo si può aggiungere, al secondo membro dell'equazione precedente, un polinomio cubico la cui derivata seconda varia linearmente dal valore y_j" al valore y_{j+1}", ovvero è continua. Se questo polinomio cubico viene costruito in modo che abbia valore nullo in corrispondenza di x_j e x_{j+1}, aggiungendolo all'equazione precedente mantiene la corrispondenza dei valori tabulati:

$$y = A \cdot y_j + B \cdot y_{j+1} + C \cdot y''_j + D \cdot y''_{j+1},$$

dove A e B sono uguali a quelli dell'equazione precedente, $C = \frac{1}{6}\left(A^3 - A\right)\left(x_{j+1} - x_j\right)^2$ e $D = \frac{1}{6}\left(B^3 - B\right)\left(x_{j+1} - x_j\right)^2$. Le derivate prima e seconda di questa funzione valgono:

$$\frac{dy}{dx} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{3A^2 - 1}{6}\left(x_{j+1} - x_j\right)y''_j + \frac{3B^2 - 1}{6}\left(x_{j+1} - x_j\right)y''_{j+1}$$

$$\frac{d^2y}{dx^2} = A \cdot y''_j + B \cdot y''_{j+1}$$

L'unico problema è che si è supposto di conoscere i valori y_i" ma, nella realtà, non è così. Però, non si è ancora richiesto che la derivata prima sia continua agli estremi degli intervalli. L'idea è di imporre questa continuità ed utilizzarla per calcolare i valori della derivata seconda in corrispondenza dei punti campione.

Le equazioni richieste sono ottenute ponendo la derivata prima calcolata in x_j con riferimento all'intervallo (x_{j-1},x_j) uguale alla stessa calcolata sempre in x_j, ma con riferimento all'intervallo (x_j,x_{j+1}):

$$\frac{x_j - x_{j-1}}{6} y''_{j-1} + \frac{x_{j+1} - x_{j-1}}{3} y''_j + \frac{x_{j+1} - x_j}{6} y''_{j+1} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}$$

Ci sono N-2 equazioni (per j=2,...,N-1) nelle N incognite y_i", per i=1,...,N. Per avere un'unica soluzione, si devono specificare altre due condizioni, tipicamente prese come condizioni al contorno in x_1 e x_N. I modi più comuni sono:

- porre una o entrambe y_1" e y_N" uguali a 0, ottenendo la cosiddetta *spline cubica naturale*, che ha derivata seconda nulla in uno o entrambi i suoi estremi;
- porre y_1" e y_N" uguali a valori calcolati dall'equazione della derivata prima, in modo che la derivata prima della funzione interpolante abbia i valori specificati agli estremi della funzione.

Nel caso in esame, si è scelto di utilizzare le condizioni di spline naturale.

Per ottenere una griglia di interpolazione rettangolare completa, come richiesto dalla tecnica descritta, si è scelto di memorizzare i punti campione P_i non con le coordinate (X_i,Y_i), ma in termini di distanza e orientazione dal centro della base del robot, punto di riferimento dell'odometria dello stesso, preoccupandosi di cancellare le righe non complete per i motivi già citati.

Inoltre, l'interpolazione bidimensionale attraverso spline bicubiche permette di interpolare funzioni $\Re^2 \to \Re$, e non $\Re^2 \to \Re^2$ come necessario per rimappare il piano immagine nel piano di riferimento per l'inversione prospettica. Per ovviare a questo problema, si è scelto di interpolare due funzioni $\Re^2 \to \Re$, che permettono di rimappare una la coordinata x del piano immagine sul piano di riferimento, l'altra la coordinata y.

Il risultato della tecnica di calibrazione descritta, sull'immagine di riferimento riportata in precedenza, è raffigurato nella figura seguente.

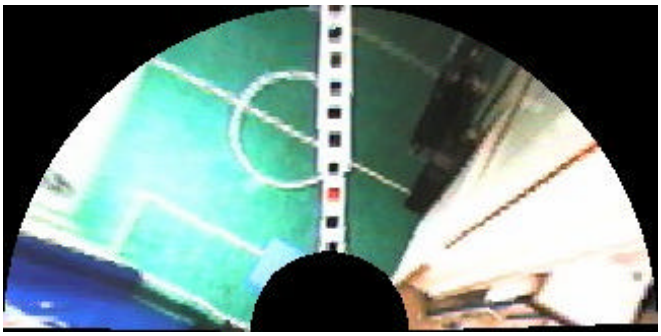


Fig. 3.5 Risultato della calibrazione empirica del sensore omnidirezionale.

Questa tecnica è, in assoluto, la più generale per la calibrazione di un sensore omnidirezionale: permette di gestire non-idealità nella forma dello specchio, scostamenti dell'asse ottico rispetto a quello della base del robot, ecc. Come requisito fondamentale, però ha che si possa conoscere, con buona precisione, la posizione angolare del robot al momento dell'acquisizione di ognuna delle immagini utilizzate per raccogliere i punti campione. Se questo non accade, l'interpolazione non può funzionare correttamente e la ricostruzione risulta imprecisa.

3.2.3 Tecnica di calibrazione empirica radiale

Nel caso non si possa conoscere con esattezza la posizione angolare del robot durante l'acquisizione delle immagini, ma potendo confidare nella perfetta simmetria rotazionale dello specchio, si può applicare una tecnica di calibrazione empirica ridotta. Questa è del tutto analoga alla precedente: l'unica differenza è che i punti campione vengono estratti lungo una sola linea direttrice, per semplicità presa di fronte al robot. Perciò, per associare un punto sul piano di riferimento ad un punto del piano immagine si procede nel modo seguente: la distanza del pixel corrispondente dal centro del CCD sul piano immagine si calcola per interpolazione attraverso spline cubica uni-dimensionale a partire dalla distanza del punto stesso dal centro della base del robot, mentre l'orientazione rimane inalterata, ma con segno opposto, in quanto lo specchio inverte la sinistra con la destra.

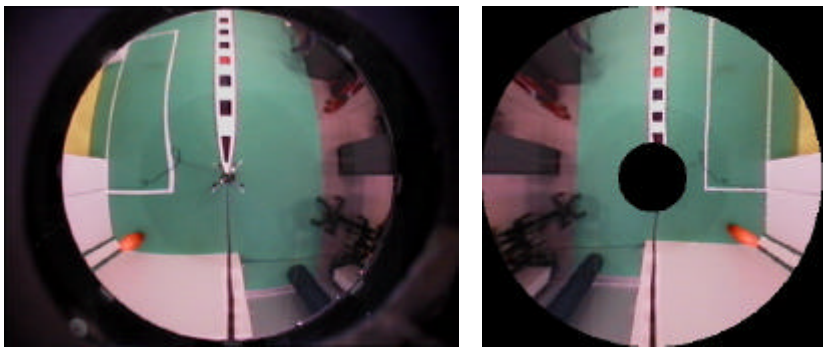


Fig. 3.6 Risultato della calibrazione empirica radiale del sensore omnidirezionale.

Un requisito di questo metodo è che l'asse ottico del catadiottro coincida con l'asse verticale della base del robot, altrimenti non vale più che l'orientazione non cambia passando da piano immagine a pavimento.

3.3 Tecniche di calibrazione del sensore frontale

La calibrazione di una telecamera a CCD tradizionale consiste nella ricerca dei parametri che regolano il modello e che non sempre sono noti o facili da ottenere. Nel lavoro descritto sono state sperimentate due tecniche empiriche, analoghe a quelle già descritte per il sensore omnidirezionale.

La prima tecnica è del tutto identica a quella empirica completa illustrata nel paragrafo 3.2.2. L'unica differenza è un campo visivo più ristretto, e quindi meno rette direttrici da analizzare per l'estrazione dei punti campione.

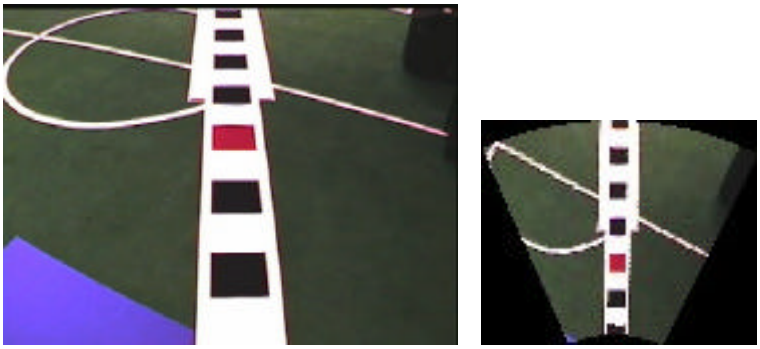


Fig. 3.7 Risultato della calibrazione empirica completa del sensore tradizionale.

La seconda tecnica è intermedia tra quella empirica completa ed una geometrica; in questo senso, è simile alla tecnica radiale descritta nel paragrafo 3.2.3.

Questo metodo si fonda su:

- la proprietà per cui, se la distorsione radiale dell'ottica fosse nulla, ad ogni retta nell'immagine corrisponderebbe una retta sul piano del pavimento;
- la raccolta di un insieme W di campioni (p_i,P_i), dove p_i è il pixel dell'immagine senza distorsione radiale e P_i la cella del piano di riferimento rappresentata nell'immagine dal pixel p_i (o da un insieme di pixel tra cui p_i, a seconda del valore di *sens*).

In realtà, la distorsione dell'ottica non è nulla, ma si può ipotizzare che sia di tipo sferico e rilevante esclusivamente nella sua componente radiale (ipotesi valida dato che la telecamera utilizzata presenta un angolo di apertura inferiore agli 80°, quindi piuttosto basso), e che quindi sia possibile identificare un coefficiente di distorsione radiale (K), che sia costante in ogni punto dell'immagine. Grazie all'indipendenza dell'effetto della distorsione radiale dall'effetto prospettico, che permette di vederli come due blocchi indipendenti posti in cascata, è possibile recuperare la distorsione dell'ottica tramite la semplice espansione dell'immagine attraverso le seguenti formule, da applicare a tutti i pixel dell'immagine distorta,

$$\begin{aligned} x_u &= x_d\left(1+K\left(x_d^2+y_d^2\right)\right) \\ y_u &= y_d\left(1+K\left(x_d^2+y_d^2\right)\right) \end{aligned},$$

dove (x_d,y_d) sono le coordinate del pixel dell'immagine originale distorta, riferite ad un sistema cartesiano con origine nel centro dell'immagine, mentre (x_u,y_u) sono le coordinate riferite allo stesso sistema cartesiano alle quali il pixel deve essere spostato.

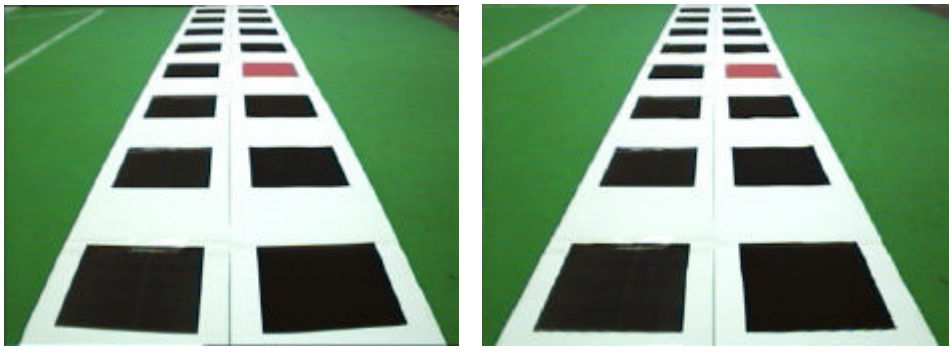


Fig. 3.8 Risultato della rimozione della distorsione radiale dall'immagine acquisita dal sensore tradizionale.

L'immagine a cui è stata rimossa la distorsione radiale viene analizzata per determinare i punti campione, come i punti centrali dei quadrati posizionati sulle due strisce parallele. Per ogni striscia viene calcolata la retta ai minimi quadrati di coefficiente angolare m e termine noto q:

$$m = \frac{N \cdot \sum_i x_i y_i - \left(\sum_i x_i\right)\left(\sum_i y_i\right)}{N \cdot \sum_i x_i^2 - \left(\sum_i x_i\right)^2}$$

$$q = \frac{\left(\sum_i y_i\right)\left(\sum_i x_i^2\right) - \left(\sum_i x_i\right)\left(\sum_i x_i y_i\right)}{N \cdot \sum_i x_i^2 - \left(\sum_i x_i\right)^2}$$

Questo è importante perché, essendo le due strisce di quadrati parallele sul piano di riferimento, lo devono essere anche nell'immagine raddrizzata, quindi l'equazione delle due rette approssimanti può essere impiegata per calcolare di quanto deve essere stirata (allungata o accorciata, in funzione di *sens*) ogni riga dell'immagine dedistorta. Inoltre, la distanza del centro di ogni quadrato dalla cima dell'immagine permette, attraverso un'interpolazione con spline cubiche uni-dimensionali, di calcolare l'allungamento in verticale dell'immagine stessa.

In conclusione, per rimappare un pixel dell'immagine in un punto sul piano di riferimento, la nuova coordinata *y* viene determinata per interpolazione a partire dalla distanza del pixel dalla cima dell'immagine dedistorta, mentre la nuova coordinata *x* viene calcolata attraverso il coefficiente di stiramento della riga dell'immagine estratta per interpolazione, in funzione della distanza tra le due rette approssimanti in corrispondenza della riga stessa.

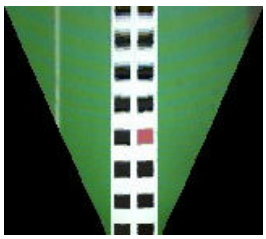


Fig. 3.9 Risultato della calibrazione descritta per il sensore tradizionale.

Questo metodo ha come requisito che le strisce di quadrati siano parallele tra loro e che l'asse centrale tra le due sia il più possibile coincidente con l'asse verticale dell'immagine acquisita.

3.4 Metodo di calibrazione autonoma del sistema

Tutte le tecniche di calibrazione almeno parzialmente empiriche, descritte nei paragrafi precedenti, prevedono l'estrazione di punti campione da un certo numero di immagini acquisite. Questo processo viene svolto automaticamente, rilevando pattern di quadrati neri alternati regolarmente ed allineati su una striscia di colore bianco.

I passi dell'algoritmo di riconoscimento dei quadrati sono riportati di seguito.

- Si parte dal centro dell'immagine nel caso del sensore omnidirezionale o dal centro del fondo dell'immagine nel caso del sensore tradizionale e si inizia a cercare, lungo la direzione indicata dalla posizione angolare del robot salvata in fase di acquisizione, il primo tratto visibile della striscia bianca; una volta trovato, se ne individua il centro in larghezza.
- Si continua a scandire, lungo la stessa direzione, alla ricerca dell'inizio di un quadrato, identificato da un pixel del colore del quadrato stesso con almeno un certo numero di vicini dello stesso colore ed un certo numero di vicini bianchi.
- Una volta trovato questo pixel iniziale, ci si estende a tutto il quadrato con una tecnica di *blob colouring*. A questo punto, si determina il baricentro del quadrato trovato, che diventa uno dei punti campione per l'interpolazione.
- Per trovare i quadrati successivi, la direzione in cui muoversi viene corretta come quella che congiunge gli ultimi due quadrati trovati o, per il secondo, quella che congiunge il centro del primo con il centro in larghezza della striscia bianca, individuata all'inizio dell'algoritmo.
- Questo metodo si ripete per ogni direzione di interesse.

I vari passi di elaborazione delle immagini sono riportati nella figura seguente, con riferimento al sensore tradizionale per ragioni di visibilità:

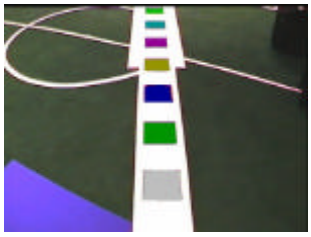
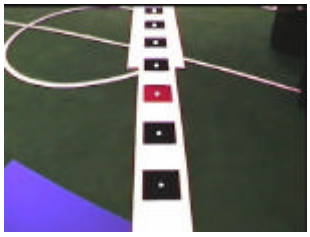
a)  b)  c) 

Fig. 3.10 Passi dell'algoritmo per l'estrazione dei punti campione: a) immagine acquisita; b) rilevazione dei quadrati costituenti il pattern; c) determinazione del baricentro dei quadrati.

Lungo la striscia bianca, si nota la presenza anche di un quadrato rosso. Questo, che viene rilevato in modo automatico esattamente come quelli neri, serve da riferimento per associare al pixel che rappresenta il centro di un quadrato la sua esatta distanza sul piano di riferimento dal centro del robot.

Lo stesso procedimento viene applicato anche nella calibrazione empirico-geometrica della telecamera frontale, anche se si devono rilevare due strisce invece di una: basta eseguire due volte lo stesso algoritmo, partendo da due punti, i centri delle strisce, differenti.

Un altro passo che viene eseguito automaticamente, ma alla fine di tutto il processo di calibrazione, è il calcolo del punto di riferimento dell'immagine raddrizzata, ovvero del pixel a cui corrisponde il centro della base del robot sul piano di riferimento. Questo viene fatto semplicemente dall'analisi dei campi visivi dei sensori, a partire dall'insieme dei punti campione estratti. Questo punto è di cruciale importanza, perché permette di mettere in relazione le immagini raddrizzate dei due sensori, al fine di trovare l'area di campo visivo comune.

Capitolo 4

Gli algoritmi di visione

4.1 Impiego del sistema ibrido per applicazioni di robotica mobile

L'uso congiunto delle informazioni provenienti dalle due telecamere è solo una delle possibilità d'impiego di HOPS. Il sistema prevede da una parte la possibilità d'impiego del catadiottro per un'analisi poco particolareggiata ma globale della scena circostante: applicazioni quali l'auto-localizzazione e la ricerca, su tutta la regione circostante il sistema, di eventuali elementi di interesse trovano grossi vantaggi nell'uso di un sensore di quel tipo. Dall'altra parte, la telecamera tradizionale può essere utilizzata per un'analisi a maggiore risoluzione della scena presente di fronte al robot.

Detto ciò, l'obiettivo primario del lavoro svolto nell'analisi delle immagini è, comunque, quello di rilevare la presenza di ostacoli nello spazio visivo comune ai due sensori di cui HOPS si avvale, quindi un utilizzo da sistema binoculare. Questa è, chiaramente, un'applicazione centrale per la navigazione di robot mobili e rappresenta una componente fondamentale per la costruzione di qualsiasi sistema di visione completo.

Il meccanismo di rilevazione degli ostacoli proposto si basa sulla tecnica della prospettiva inversa, analizzata nel paragrafo 3.1. Allo scopo di ottenere informazioni riguardanti la presenza di ostacoli e, quindi, l'entità dell'area camminabile, una valida alternativa allo studio delle disparità stereo su linee epipolari, è quella di rettificare entrambe le immagini e, grazie ad una calibrazione del sistema binoculare, di confrontarle direttamente. L'operazione di recupero della prospettiva viene

effettuata rispetto al piano del pavimento, quindi entrambe le immagini rettificate sono identiche nella loro rappresentazione di ciò che risiede sul pavimento stesso, mentre tutto ciò che emerge da questo piano (gli ostacoli) subisce una distorsione dipendente dalla prospettiva dalla quale si osserva la scena, cioè una distorsione che risulta essere diversa per le due diverse telecamere. Una semplice ricerca delle differenze tra le due immagini pixel a pixel fa, quindi, emergere informazioni sugli ostacoli presenti. Si tratta, a quel punto, di interpretare le informazioni emerse per giungere all'identificazione degli ostacoli veri e propri.

4.2 Descrizione degli algoritmi

Obiettivo di questa fase di elaborazione delle immagini è ricavare, a partire dai frame acquisiti dalle telecamere a bordo del robot, un'immagine congiunta in cui risultino evidenziate le regioni in cui le due immagini rettificate sono diverse, a causa della presenza di ostacoli (o meglio, oggetti che emergono dal pavimento). Le due telecamere operano in condizioni diverse sia di illuminazione, che di area inquadrata. Inoltre, nel prototipo realizzato, la situazione è aggravata dal fatto che i due sensori CCD hanno curve di sensibilità alla luce diverse ed entrambe utilizzano un sistema di bilanciamento automatico del bianco, problemi che, comunque, possono essere eliminati o limitati al momento della realizzazione dell'apparato definitivo. Per questi motivi, non può bastare un semplice ed immediato confronto tra le due immagini pixel a pixel, ma sono necessari alcuni passi di elaborazione, per ridurre il rumore e le differenze cromatiche con cui gli stessi oggetti vengono visti dalle due telecamere.

Di seguito è riportato uno schema a blocchi raffigurante i vari passi di questa fase di *pre-processing*:

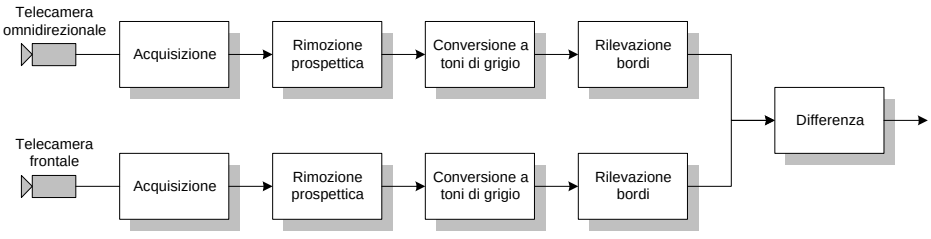


Fig. 4.1 Diagramma a blocchi delle fasi di elaborazione.

Per spiegare meglio tutti i passi, si fa riferimento ad un caso sperimentale pratico, relativo ad un set di immagini riprese nella prima fase di prove effettuata all'Università di Ulm. Si consideri la seguente coppia di immagini acquisite contemporaneamente e riproducenti la stessa scena:

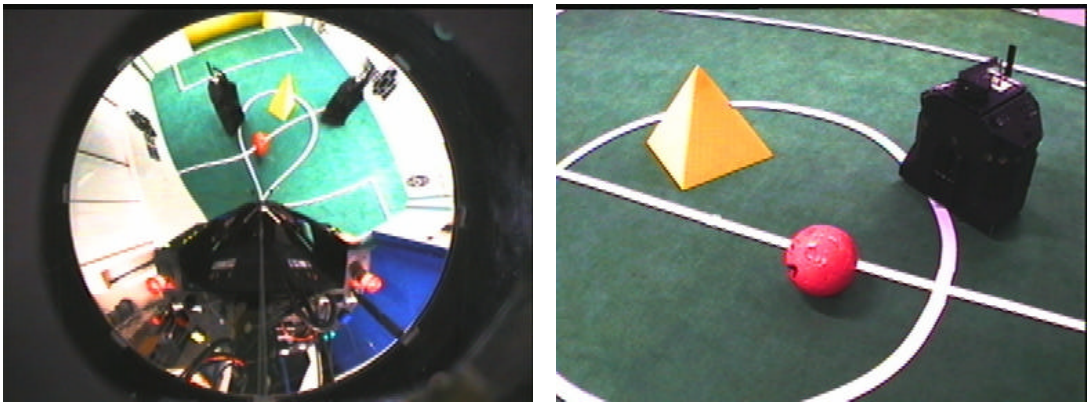


Fig. 4.2 Immagini acquisite dalle telecamere.

La prima fase (che è anche la più critica) è la rimozione prospettica. Come già visto, questa consente di trasformare le immagini acquisite in modo che diano una visione a volo d'uccello della scena, come se fossero state riprese da telecamere sul soffitto. Questo è possibile grazie all'applicazione di una tabella di look-up, generata durante la fase di calibrazione del sistema visivo, che mette in corrispondenza ogni punto del piano di riferimento con un pixel dell'immagine acquisita. Il risultato di questa fase, nel caso di riferimento, è il seguente:

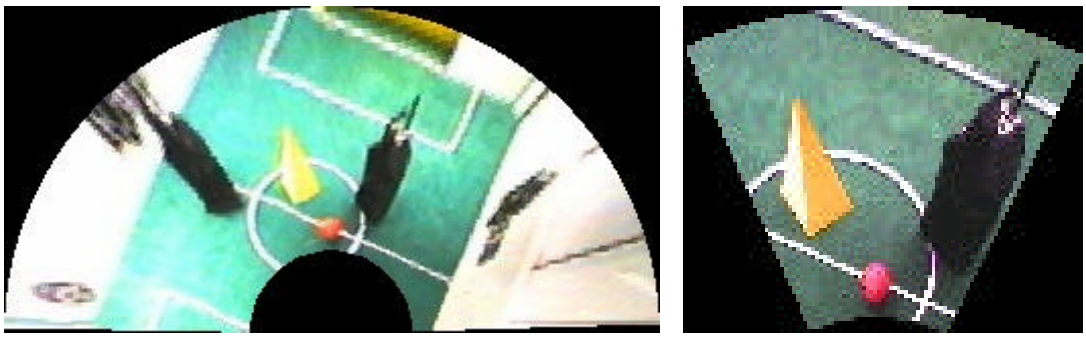


Fig. 4.3 Immagini raddrizzate.

Le immagini raddrizzate non vengono utilizzate nella loro interezza, ma solo nella regione di piano in cui si sovrappongono.

Fig. 4.4 Visualizzazione della regione di sovrapposizione delle immagini.

Su queste, sono stati provati vari tipi di filtraggi e trasformazioni. Il problema principale presentatosi è dovuto al fatto che, attraverso il visore omnidirezionale, le immagini risultano molto più sature che attraverso la telecamera frontale. Queste differenze cromatiche con cui gli stessi oggetti sono rappresentati nelle due immagini, che rendono difficile la loro comparazione, sono dovute sia ai differenti campi visivi delle due telecamere, e quindi alla differente scena rappresentata, sia al fatto che, mentre la telecamera frontale osserva direttamente la scena che rappresenta, il catadiottro riceve luce indiretta, per la riflessione dello specchio, e filtrata dal cilindro di plexiglas della struttura. Perciò, per poter confrontare in modo corretto le immagini, è necessario trovare un modo per “equalizzarle”.

Una prima prova, piuttosto intuitiva, potrebbe consistere nel variare i parametri dei frame-grabber, per cercare di avvicinare le caratteristiche cromatiche delle due immagini, ma questo non è possibile in tutte le configurazioni di un sistema robotico (ad esempio, quella dell’Università di Ulm).

Una seconda soluzione esplorata è la trasformazione dell’istogramma dell’immagine proveniente dal sensore catadiottrico, allo scopo di renderlo il più simile possibile a quello dell’immagine proveniente dall’altra telecamera (supposta di riferimento).

Di seguito vengono descritti brevemente i passi dell’elaborazione eseguita.

- L'istogramma dell'immagine omnidirezionale iniziale viene trasformato indipendentemente sulle tre bande R, G, B, ipotizzando che la correlazione tra i loro andamenti sia bassa. Nel seguito, si fa riferimento alle operazioni da svolgere su una sola delle tre bande (ad esempio quella del rosso R), ma il tutto è identico anche per le altre due.
- Per ogni valore p tra 0 e 255 (ossia, tutti i valori di intensità che un pixel può assumere) l'istogramma dell'immagine trasformata deve avere, come risultato finale, che il suo integrale da 0 a p sia il più vicino possibile all'integrale, sullo stesso intervallo, dell'istogramma dell'immagine di riferimento.
- Per ottenere ciò, si deve operare questa integrazione da 0 a p (ottenendo un valore T) sull'immagine di riferimento, e poi andare a cercare, nell'istogramma dell'immagine iniziale, per quale valore q si ottiene che l'integrale nell'intervallo $[0,q]$ ha il valore più vicino a T.
- Grazie ad un'integrazione su intervalli di dimensione crescente (da 0 a 255), si ottiene, quindi, una funzione $q(p)$. Ecco che la trasformazione dell'istogramma consiste semplicemente nel modificare le intensità dei pixel dell'immagine iniziale, sulla base della relazione $q(p)$ ottenuta: da essa si può, infatti, suddividere l'istogramma in intervalli delimitati dai valori $q(i)$ con $i=0,\ldots,255$, ossia il primo intervallo ($i=0$) è $[0,q(0)]$, l'n-esimo ($n=i+1$) è $[q(n-2),q(n-1)]$. Ad ogni pixel con intensità, sulla banda in questione, compresa nell'intervallo n-esimo, viene assegnato un nuovo valore di intensità pari a $i=n-1$.

Questa operazione rende le due immagini (quella di riferimento e quella trasformata) il più simili possibile, sia dal punto di vista cromatico che del contrasto. Si noti che questa elaborazione viene eseguita tra due immagini che, a parte piccole differenze legate alla distorsione della prospettiva inversa sugli ostacoli, inquadrano la stessa porzione di una scena e quindi rappresentano gli stessi oggetti e lo stesso sfondo, condizione necessaria per poter considerare fondato l'algoritmo stesso. Il risultato dell'applicazione della trasformazione dell'istogramma sull'immagine omnidirezionale precedente è rappresentato nella figura seguente:

Fig. 4.5 Immagine omnidirezionale con istogramma trasformato.

Si può notare che le zone bianche non sono state trasformate in modo corretto, e questo accade perché l'immagine iniziale, essendo molto satura, presenta sulle varie bande, in particolare su quella del verde, valori molto elevati all'estremità superiore dell'istogramma. Questo rende poco fedele la trasformazione perché, comunque si spostino i livelli con valori così elevati, non si riesce mai ad ottenere un avvicinamento soddisfacente all'istogramma dell'immagine di riferimento.

Risultati migliori sono stati ottenuti con algoritmi di *edge-detection* ([26],[27]). Tra questi ne sono stati sviluppati due: il primo si basa su una maschera di Sobel 3x3, mentre il secondo sull'applicazione di un filtraggio passa-basso gaussiano e, successivamente, di una maschera di Sobel, entrambi 5x5.

L'operatore di Sobel misura il gradiente bidimensionale su un'immagine, in modo da enfatizzare le regioni ad alta frequenza spaziale che corrispondono a bordi. Tipicamente, viene usato per trovare il modulo del gradiente approssimato in ogni punto di un'immagine a toni di grigio. Perciò, prima di poterlo applicare, occorre effettuare una conversione dal formato true-color (24 bit per pixel) a quello a livelli di grigio (8 bit per pixel).

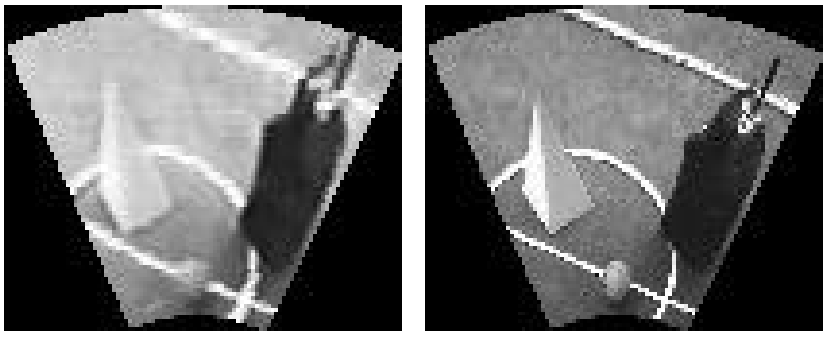


Fig. 4.6 Immagini convertite a toni di grigio.

Questo non è sempre necessario, perché si potrebbe anche essere interessati al calcolo del gradiente su ogni banda di colore, ma per questo progetto si è scelto un approccio più classico, al fine di mantenere una buona generalità e tempi di calcolo ridotti.

Il metodo di Sobel consiste nell'applicare ad ogni pixel dell'immagine due maschere, una per rilevare i gradienti orizzontali e l'altra per quelli verticali:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Queste matrici possono essere applicate separatamente, per avere una misura del gradiente in ogni direzione, ma anche combinate per trovare il modulo e l'orientazione del gradiente:

$$|G| = \sqrt{G_x^2 + G_y^2} \cong |G_x| + |G_y|$$

$$J = \arctan\left(\frac{G_y}{G_x}\right)$$

Tipicamente, per il calcolo del modulo si utilizza l'approssimazione indicata, in quanto molto più veloce. Orientazione pari a 0 significa che la direzione di massimo contrasto è orizzontale e il verso è da sinistra a destra, mentre angoli superiori sono misurati in senso antiorario. Spesso, comunque, viene utilizzata solo l'informazione relativa al modulo. Siccome, poi, l'operatore di Sobel è lineare, ogni matrice può essere scomposta come prodotto di un vettore colonna per un vettore riga:

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} = \begin{bmatrix} +1 \\ +2 \\ +1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & +1 \end{bmatrix} \quad ; \quad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} +1 & +2 & +1 \end{bmatrix}$$

Questo consente di ridurre il numero di moltiplicazioni, e quindi i tempi di calcolo.

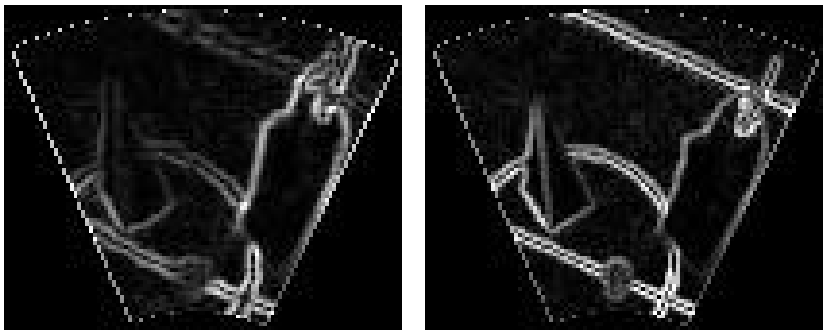


Fig. 4.7 Risultati dell'applicazione dell'operatore di Sobel (3x3).

Come anticipato, come seconda soluzione si è pensato di applicare, prima della maschera di Sobel, un filtraggio passa-basso gaussiano, che è un operatore bidimensionale che permette di rimuovere dettagli e rumore; in questo senso, è simile ad un filtraggio a media, ma utilizza una maschera diversa, in particolare raffigurante la forma di una campana gaussiana. In teoria, la distribuzione gaussiana è non nulla ovunque, il che richiederebbe una maschera di convoluzione di larghezza infinita ma, in pratica, è effettivamente nulla ad una distanza dal valor medio pari a circa tre volte la deviazione standard, perciò la maschera può essere troncata a quel punto.

Per maschere di piccole dimensioni (fino a 5x5) una buona approssimazione della gaussiana con s =1 si può ottenere a partire dai coefficienti binomiali. Nel caso in esame, in cui si è interessati ad un nucleo di convoluzione 5x5, si ha:

1						1	4	6	4	1
4						4	16	24	16	4
6	·	1	4	6	4	1	= 6	24	36	24	6
4						4	16	24	16	4
1						1	4	6	4	1

Convoluendo questo operatore, scomposto in vettore colonna e vettore riga, con una maschera di Sobel generalizzata in formato 5x5, si ottiene:

$$G_x = \begin{bmatrix} -1 & -2 & 0 & +2 & +1 \\ -4 & -8 & 0 & +8 & +4 \\ -6 & -12 & 0 & +12 & +6 \\ -4 & -8 & 0 & +8 & +4 \\ -1 & -2 & 0 & +2 & +1 \end{bmatrix} = \begin{bmatrix} +1 \\ +4 \\ +6 \\ +4 \\ +1 \end{bmatrix} \cdot \begin{bmatrix} -1 & -2 & 0 & +2 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} +1 & +4 & +6 & +4 & +1 \\ +2 & +8 & +12 & +8 & +2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix} = \begin{bmatrix} +1 \\ +2 \\ 0 \\ -2 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} +1 & +4 & +6 & +4 & +1 \end{bmatrix}$$

Applicando questi operatori, le immagini presentano bordi più larghi, per la maggiore profondità delle maschere applicate, e sono più insensibili a pixel di rumore isolati, rispetto a quelle a cui era stato applicato il gradiente di Sobel classico.

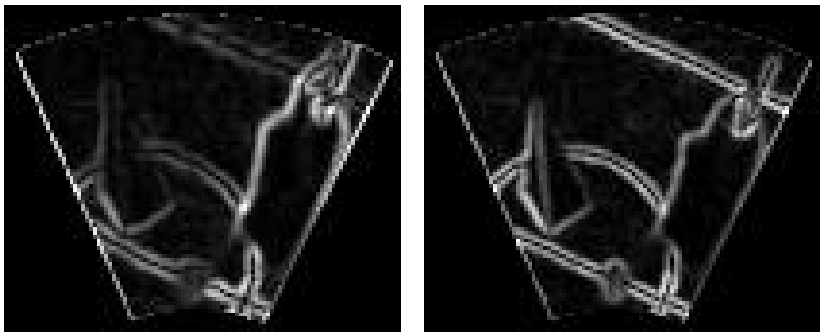


Fig. 4.8 Risultati dell'applicazione dell'operatore gaussiano+Sobel (5x5).

Qualunque filtro a maschera, come quelli visti, pone il problema di come trattare i bordi dell'immagine. Le soluzioni possibili sono due: o si replicano le prime e le ultime righe e le prime e le ultime colonne all'esterno dell'immagine, in modo che non diano contributo al calcolo del valore filtrato, oppure si evita di applicare la maschera nei bordi stessi. In questo progetto si è scelto di impiegare il secondo metodo.

La soluzione adottata per "equalizzare" al meglio le immagini, derivata dall'operatore di Canny, è di trovare le creste dei bordi estratti, per eliminare le

differenze dovute ad un livello di saturazione molto diverso tra le due immagini. Per fare questo si utilizza un sistema a due soglie: si scandisce l'immagine pixel a pixel e, appena se ne trova uno che supera la soglia superiore, lo si porta al massimo livello di intensità; a questo punto, se ne individua il vicino con maggiore intensità, lo si porta al livello massimo e gli altri vicini al livello minimo, e così via finché il vicino maggiore non scende sotto la soglia minima. In questo modo si ottiene un'immagine binarizzata e costituita da cammini di varia lunghezza.

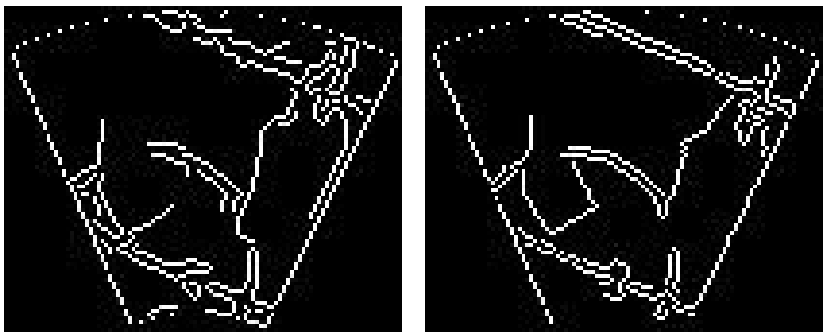


Fig. 4.9 Immagini ottenute dall'estrazione delle creste dei bordi.

Le immagini così ricavate possono, infine, essere differenziate, così da ottenere un'immagine congiunta su cui si potrà applicare la parte di visione vera e propria, ovvero il riconoscimento degli ostacoli.

Fig. 4.10 Immagine congiunta.

4.3 Parallelizzazione degli algoritmi con architettura MMX

Vista l'applicazione primaria del sistema visivo, l'indicatore di bontà principale del software prodotto è il tempo di esecuzione. Per ridurlo si è, quindi, pensato alla parallelizzazione di alcune delle sue parti.

La scelta della metodologia di parallelizzazione è caduta sull'architettura MMX (*MultiMedia eXtension*) ([21],[28],[29]), principalmente perché la maggior parte dei robot mobili monta un solo processore, e non è perciò possibile una tecnica multi-chip basata, ad esempio, su un cluster di PC. L'architettura MMX, che è un'estensione di tipo SIMD (*Single Instruction – Multiple Data*) dell'*Intel Architecture* (IA) di base, è particolarmente adatta per implementare alcuni algoritmi di visione, come i filtri impiegati nella rilevazione dei bordi e nel filtraggio passa-basso gaussiano, che prevedono l'applicazione dello stesso operatore (o della stessa sequenza di operatori) a tutti i pixel dell'immagine.

La tecnologia MMX è stata mappata all'interno della già esistente architettura floating-point e dei suoi registri. Questo ha consentito di mantenere compatibilità all'indietro con software esistenti, sia sistemi operativi che applicativi, senza il bisogno di modificare tutto per l'introduzione della nuova estensione.

Le applicazioni per cui è stata pensata sono, in particolare, l'elaborazione d'immagini, il texture mapping e la decompressione audio e video. All'interno di queste, le sequenze di codice hanno alcune caratteristiche comuni:

- tipi di dato di piccole dimensioni (ad esempio, pixel di 8 bit o campioni audio di 16 bit);
- pattern di accesso alla memoria regolari e ricorrenti;
- operazioni ripetitive eseguite sui dati;
- algoritmi compute-intensive.

Le principali caratteristiche della tecnologia MMX sono le seguenti.

- Quattro nuovi tipi di dato, pensati per impacchettare elementi di dati indipendenti in un registro. Ogni elemento in uno di questi tipi è un intero fixed-point; l'architettura non specifica la posizione del punto decimale all'interno degli

elementi, ma è compito dell'utente curarsene durante l'elaborazione. I nuovi tipi di dato definiti sono:

- Packed byte → 8 byte impacchettati in 64 bit
- Packed word → 4 word impacchettati in 64 bit
- Packed double-word → 2 double-word impacchettati in 64 bit
- Packed quad-word → 64 bit

- Un set di 57 istruzioni ottimizzate per operare su tutti gli elementi di dati indipendenti presenti in un registro, impiegando un parallelismo SIMD. Tutte le istruzioni, ad eccezione delle moltiplicazioni, vengono eseguite in un solo ciclo macchina. Le moltiplicazioni prevedono tempi di latenza di tre cicli, ma l'unità moltiplicatrice permette l'inizio del calcolo di un nuovo prodotto ogni ciclo macchina; questo, però, è vantaggioso solo se si devono eseguire più moltiplicazioni in sequenza, altrimenti il tempo di latenza è piuttosto penalizzante.
- Otto nuovi registri MMX a 64 bit mappati sui registri floating-point dell'IA, nei bit meno significativi (quelli della mantissa); il campo dell'esponente e il bit di segno sono settati a 1, generando un valore che non sarebbe accettabile come dato in virgola mobile. Questa tecnica è chiamata *aliasing*, perché i registri MMX e floating-point sono condivisi. Per questo motivo, all'interno delle applicazioni non si possono utilizzare dati ed istruzioni MMX e floating-point contemporaneamente, ma le parti di codice relative alle due tecnologie devono essere incapsulate in sequenze separate.
- Completa compatibilità con l'IA.

I passi dell'elaborazione, illustrata nel paragrafo precedente, che meglio si prestano ad una parallelizzazione mediante un'architettura SIMD come MMX sono la conversione a toni di grigio dell'immagine RGB, l'estrazione dei bordi, sia con l'operatore di Sobel classico che con la convoluzione di gaussiana e filtro di Sobel generalizzato, e la differenziazione delle due immagini.

La rimozione prospettica consiste in un accesso non completamente ordinato ad una tabella, perciò non può essere parallelizzata in modo efficiente. Anche la ricerca delle creste dei bordi non è un algoritmo adatto, in quanto l'analisi

dell'immagine non segue un percorso determinato a priori, e questo rende impossibile trovare una tecnica che permetta di caricare più pixel contemporaneamente in un registro MMX. Per questo motivo conviene mantenere un'implementazione sequenziale dell'algoritmo.

Nelle funzioni che, invece, si è pensato di parallelizzare, l'immagine va scandita pixel per pixel, e su ognuno di questi si deve applicare lo stesso operatore, perciò il flusso del programma è perfettamente noto a priori.

Per memorizzare le immagini, vengono utilizzati dei puntatori a caratteri: negli stream di byte corrispondenti, per le immagini a colori si memorizzano tre byte per pixel (uno per la componente rossa, uno per la verde e uno per la blu), mentre per quelle a toni di grigio un solo byte per pixel che rappresenta la sua intensità.

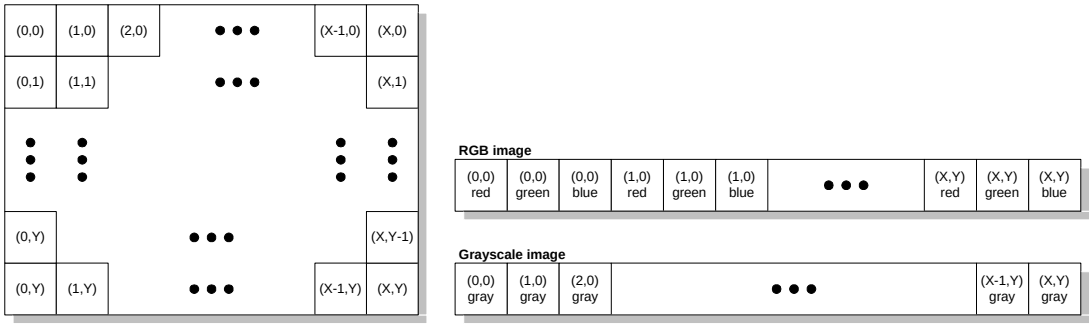


Fig. 4.11 Rappresentazione delle immagini in memoria.

Conversione da RGB a toni di grigio

Per convertire un'immagine dal formato true-color, in cui i pixel sono rappresentati da 24 bit (8 per banda di colore), a quello a toni di grigio, in cui i pixel sono rappresentati da 8 bit, si deve calcolare, per ogni pixel, una media aritmetica delle intensità sulle tre bande e memorizzare questo valore come livello di grigio. Perciò, per ogni pixel di coordinata (x,y), l'operazione da compiere è:

$$level_{out}(x,y) = \frac{r_{in}(x,y) + g_{in}(x,y) + b_{in}(x,y)}{3}$$

Essendo la parte di calcolo piuttosto semplice, la fetta di funzione più consistente è quella riguardante la riorganizzazione dei dati in ingresso, che consente la parallelizzazione del calcolo della media.

Si è pensato di lavorare su tre registri a 64 bit, ognuno contenente una banda di colore di 4 pixel. Teoricamente, sarebbe possibile lavorare con 8 pixel contemporaneamente e non solo 4, ma la somma delle tre componenti di colore può dare un risultato molto maggiore del massimo valore rappresentabile con 8 bit; inoltre, le funzioni MMX di shift (usate per moltiplicazioni e divisioni) non lavorano su dati impacchettati a byte, ma almeno a word. La figura seguente mostra come sono stati riorganizzati i byte dell'immagine in ingresso.

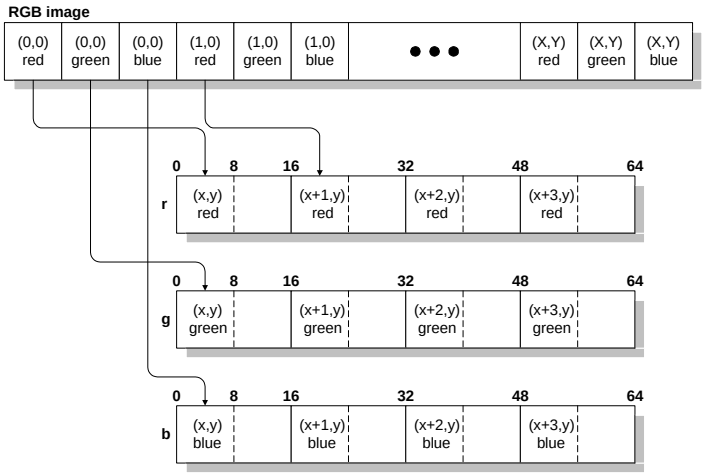


Fig. 4.12 Riorganizzazione dei byte per la conversione a toni di grigio.

A questo punto, i tre registri possono essere sommati, ma il risultato non può essere diviso per 3, in quanto l'architettura MMX non supporta né operazioni di divisione né moltiplicazioni floating-point. La soluzione consiste nel lavorare per approssimazioni successive: dividere per 3 significa moltiplicare per 0.33, che in binario è 0.01010101…; considerando solo i due bit più significativi della parte decimale, che corrispondono a 1/4 e 1/16, si ha $1/4 + 1/16 = 0.3125 = 1/3.2$ invece di 1/3. Utilizzando questa approssimazione, che è molto vantaggiosa in quanto permette di convertire una divisione in due operazioni di shift e una somma, si ottiene un'immagine leggermente più scura che con la media aritmetica.

Un'altra approssimazione rivelatasi molto buona è $1/2 - 1/8 = 0.375 = 1/2.67$, che invece porta ad un'immagine in uscita più chiara del dovuto.

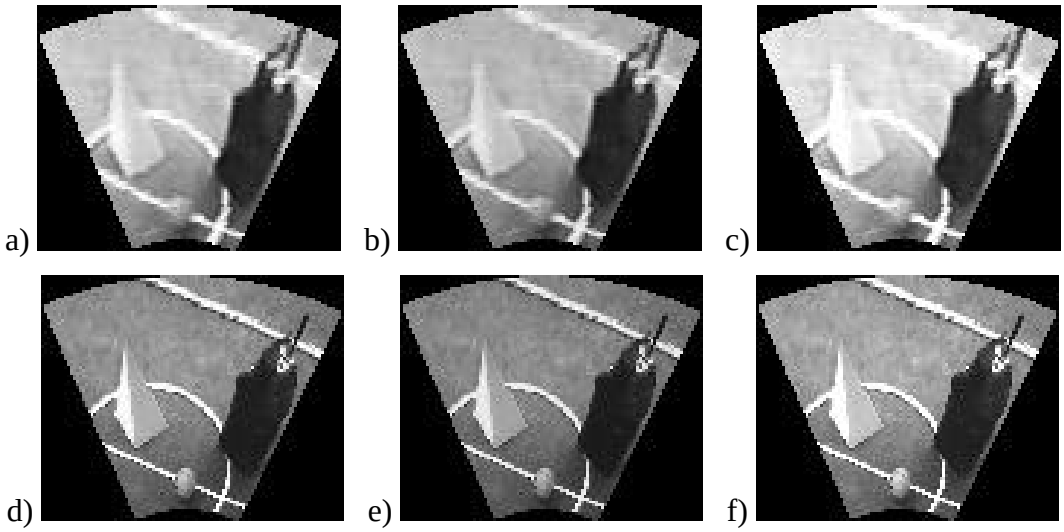


Fig. 4.13 Output della funzione di conversione da RGB a toni di grigio: a)-c) immagini riferite al visore omnidirezionale convertite con la media esatta (C++) e con le approssimazioni illustrate (MMX); d)-f) risultati ottenuti dalle immagini riferite alla telecamera tradizionale.

Dalla figura precedente, che riporta i risultati delle conversioni sulle immagini di riferimento presentate nel paragrafo precedente, si nota che la differenza tra le elaborazioni è molto limitata, perciò non incide sulle funzioni che seguono questa.

Estrazione dei bordi con l'operatore di Sobel

Con riferimento a quanto detto nel paragrafo 4.2, per l'estrazione dei bordi mediante l'operatore di Sobel non vengono impiegate direttamente le maschere 3x3 G_x e G_y, ma la loro scomposizione in vettore riga e vettore colonna.

Il parallelismo adottato, anche in questo caso, è di 4 pixel, per gli stessi motivi citati con riferimento alla conversione a toni di grigio. Così, mentre nell'immagine iniziale e finale ogni pixel è rappresentato con 8 bit, nelle variabili temporanee utilizzate per l'elaborazione, ogni pixel è rappresentato con 16 bit perciò, in fase di caricamento, i dati devono essere spacchettati, ed in fase di memorizzazione rimpacchettati.

L'algoritmo consiste, innanzitutto, nel calcolo di G_y attraverso l'applicazione del vettore riga all'immagine in ingresso, ottenendone così una intermedia, e del vettore colonna su questa, ricavando il gradiente lungo *y* dell'immagine originaria. Analogamente si calcola G_x, ovvero il gradiente lungo *x*. Infine si sommano i due contributi in modulo, per ottenere l'immagine desiderata. Entrambi questi contributi però, prima di poter essere addizionati, devono essere normalizzati, ovvero divisi per 4 (somma degli elementi positivi di ciascuna matrice), per riportare l'intensità dei pixel entro il massimo valore consentito.

Per l'elaborazione dei pixel in parallelo si usa la tecnica illustrata nella figura seguente:

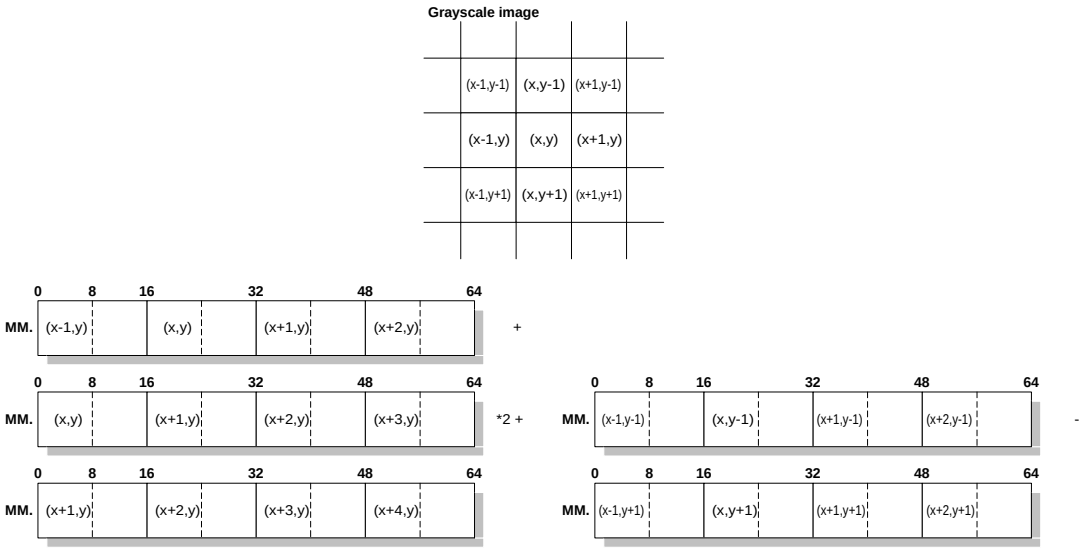


Fig. 4.14 Raffigurazione della tecnica di applicazione di vettore riga e vettore colonna all'immagine.

Come si può notare, tutte le moltiplicazioni e le divisioni hanno come fattore o divisore un multiplo di 2, perciò possono essere agevolmente calcolate mediante operazioni di shift logico.

Per il calcolo del valore assoluto di un numero si è utilizzato l'algoritmo riportato di seguito, corredato da esempi; per semplicità, questi si riferiscono a valori ad 8 bit e non a 16 come nel programma, ma ciò non altera la validità della tecnica.

CODICE	MODULO DI MM0 = 23 (00010111)	MODULO DI MM0 = -23 (11101001)
`MOVQ MM1,MM0`	MM0 = 00010111 MM1 = 00010111	MM0 = 11101001 MM1 = 11101001
`PSRAW MM0,07h`	MM0 = 00000000 MM1 = 00010111	MM0 = 11111111 MM1 = 11101001
`PXOR MM1,MM0`	MM0 = 00000000 MM1 = 00010111	MM0 = 11111111 MM1 = 00010110
`PSRLW MM0,07h`	MM0 = 00000000 MM1 = 00010111	MM0 = 00000001 MM1 = 00010110
`PADDW MM0,MM1`	MM0 = 00010111 MM1 = 00010111	MM0 = 00010111 MM1 = 00010110

L'esecuzione della funzione di edge-detection illustrata, sulle immagini di riferimento ottenute dalla funzione di conversione a toni di grigio, ha dato i seguenti risultati.

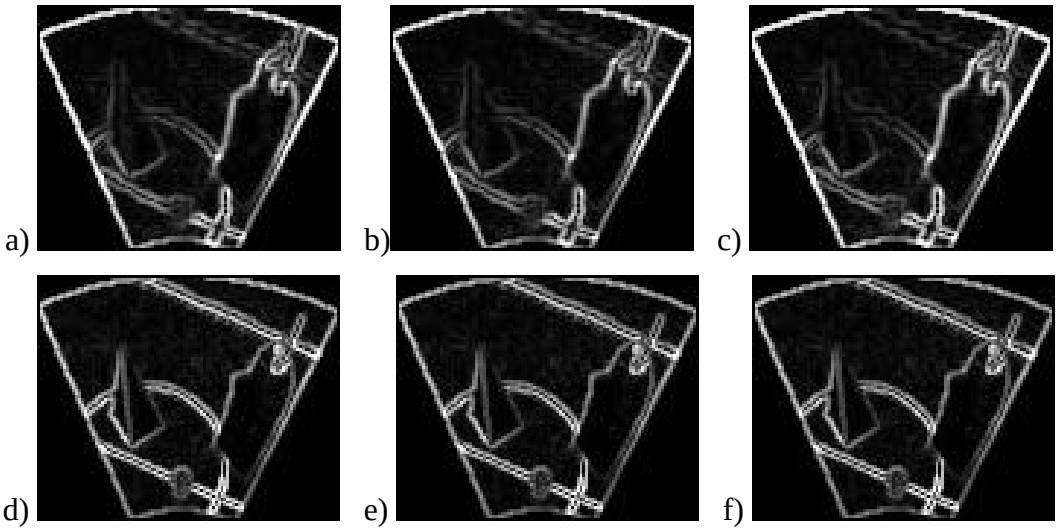


Fig. 4.15 Output della funzione di rilevamento dei bordi sulle corrispondenti immagini a toni di grigio.

Estrazione dei bordi con filtro gaussiano e operatore di Sobel

Questo secondo metodo di estrazione dei bordi è del tutto analogo al precedente, l'unica differenza è che non si utilizza solo l'operatore di Sobel, ma un filtro passa-basso gaussiano convoluto con una maschera di Sobel generalizzata, entrambi di dimensione 5x5. Anche in questo caso, si utilizza la scomposizione degli

operatori matriciali in vettori riga e vettori colonna illustrata nel paragrafo precedente, la stessa rappresentazione dei dati in memoria, ed un parallelismo a 4 pixel.

La differenza principale tra questo algoritmo e quello precedente, che deriva dalla differente dimensione delle maschere, riguarda l'accesso ai pixel dell'immagine originaria e la loro elaborazione:

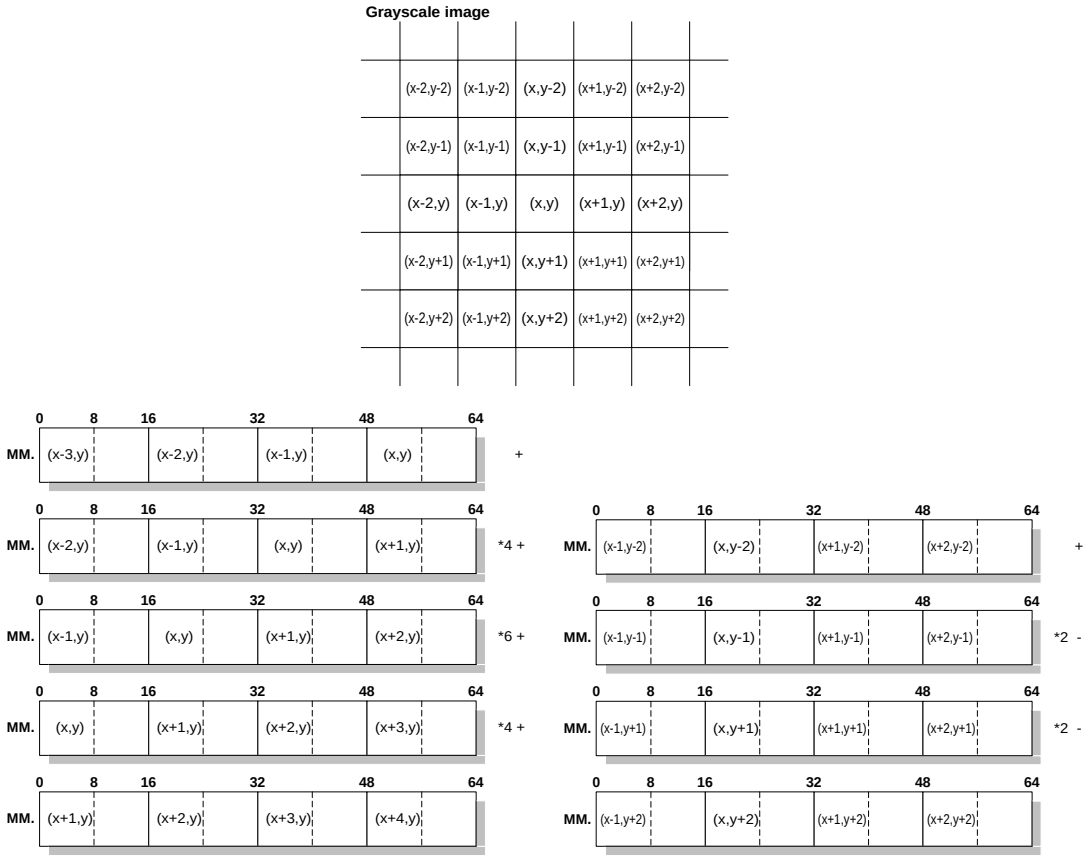


Fig. 4.16 Raffigurazione della tecnica di applicazione di vettore riga e vettore colonna all'immagine.

In questo caso, non tutte le moltiplicazioni e divisioni coinvolgono multipli di 2, ma tutte sono state svolte, comunque, mediante solo shift logici e addizioni o sottrazioni. La moltiplicazione per 6 si ottiene raddoppiando il numero da moltiplicare e sommandolo, poi, al quadruplo del numero stesso; questo risparmia comunque qualche ciclo macchina durante l'esecuzione. La divisione per il coefficiente di normalizzazione, che questa volta è pari a 48, si ottiene, con una

tecnica analoga a quella vista per la conversione a toni di grigio, attraverso l'approssimazione $1/32 - 1/128 = 1/42.67$.

L'esecuzione della funzione di edge-detection illustrata sulle immagini di riferimento a toni di grigio, ha dato i seguenti risultati.

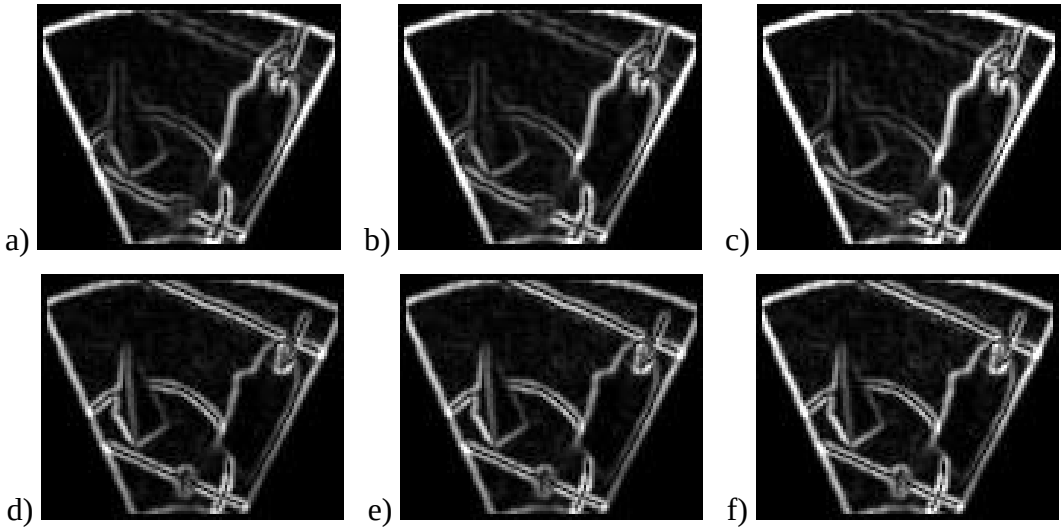


Fig. 4.17 Output della funzione di rilevamento dei bordi sulle corrispondenti immagini a toni di grigio.

Creazione dell'immagine congiunta

L'ultima funzione parallelizzata si occupa di costruire un'immagine congiunta, calcolando la differenza in modulo tra le due immagini pre-elaborate. In questo caso, si può impiegare un parallelismo a 8 pixel, in quanto non si ha bisogno di sommare numeri o di compiere operazioni di shift.

L'algoritmo impiegato è riportato nella tabella seguente, corredato da esempi:

CODICE	DIFFERENZA TRA MM0 = 95 (01011111) MM1 = 23 (00010111)	DIFFERENZA TRA MM0 = 23 (00010111) MM1 = 95 (01011111)
`MOVQ MM2,MM0`	MM0 = 01011111 MM1 = 00010111 MM2 = 01011111	MM0 = 00010111 MM1 = 01011111 MM2 = 00010111
`PSUBUSB MM0,MM1`	MM0 = 01001000 MM1 = 00010111 MM2 = 01011111	MM0 = 00000000 MM1 = 01011111 MM2 = 00010111
`PSUBUSB MM1,MM2`	MM0 = 01001000 MM1 = 00000000 MM2 = 01011111	MM0 = 00000000 MM1 = 01001000 MM2 = 00010111
`POR MM0,MM1`	MM0 = 01001000 MM1 = 00000000 MM2 = 01011111	MM0 = 01001000 MM1 = 01001000 MM2 = 00010111

Qualunque funzione di rilevazione dei bordi si decida di utilizzare, dato che i risultati sono molto simili, anche l'estrazione delle creste dei bordi stessi non darà grosse diversità, così come la differenziazione tra le immagini elaborate delle due telecamere. Per questo motivo, i risultati illustrati nel paragrafo precedente sono del tutto validi anche per le funzioni parallelizzate.

Analisi delle prestazioni

Le prestazioni degli algoritmi sono state valutate su due diverse architetture. La prima possiede un processore Intel Pentium MMX a 200Mhz con una cache solamente di primo livello contenente 16kB per le istruzioni e 16kB per i dati; la frequenza di bus, ovvero la frequenza di trasferimento delle informazioni tra CPU e memoria, per questo processore è pari a 66MHz. La seconda architettura è basata su una CPU AMD Duron Model-3 a 800MHz, con una cache L1 (divisa tra dati e istruzioni) di 128kB e una cache L2 di 64kB; la frequenza di bus, in questo caso, è pari a 200MHz.

I tempi di elaborazione ottenuti, in µs, sono riportati nella tabella seguente.

		Pentium MMX		Duron Model-3	
		C++	MMX	C++	MMX
`emptyFunction` (16*100 esecuzioni)	Prima esecuzione	3	5	1	3
	Media successive	2	4	0	1
`RGB2Gray` (16*100 esecuzioni)	Prima esecuzione	3908	2138	247	282
	Media successive	3546	1716	244	256
`detectEdgesSobel` (8*100 esecuzioni)	Prima esecuzione	28390	17491	1054	272
	Media successive	15887	2785	988	172
`detectEdgesSmoothSobel` (8*100 esecuzioni)	Prima esecuzione	40356	20383	1504	310
	Media successive	28244	3754	1506	234
`createCombined` (16*100 esecuzioni)	Prima esecuzione	3042	456	236	63
	Media successive	2824	324	243	53

La prima funzione, che è vuota, è stata inserita nella tabella per permettere il confronto tra i tempi di latenza delle chiamate a funzioni C++ e Assembly. La differenza non è considerevole, perciò può essere ignorata nel confronto tra le altre funzioni.

Dalla tabella, si nota che la parallelizzazione ha dato buoni risultati in tutte le funzioni, tranne in quella di conversione a toni di grigio, i cui tempi sono calati meno che per le altre o sono, addirittura, immutati. Una spiegazione può essere che, mentre le ultime tre accedono alla memoria a pacchi di 32 o 64 bit (4 o 8 pixel), la seconda deve preoccuparsi della riorganizzazione delle componenti colorate, sia per ottimizzare la sua esecuzione sia per impacchettare i dati per le funzioni successive, perciò può accedere ad un solo byte di memoria alla volta. Come noto, nei calcolatori moderni l'accesso alla memoria è la fase più critica in termini di tempi di esecuzione, e per questo motivo non si riesce a sfruttare la parallelizzazione della funzione di conversione a toni di grigio.

4.4 Risultati e sviluppi

L'elaborazione del set di immagini presentato ha evidenziato lati sia positivi che negativi. Da una parte, la strada seguita ha portato al "quasi" riconoscimento degli oggetti reali ed alla scomparsa dei falsi oggetti (ad esempio, le righe bianche) in tempi più che accettabili per applicazioni robotiche indoor, dall'altra ha evidenziato due problematiche di rilievo.

In primo luogo, gli oggetti, per essere rilevati, devono avere un buon contrasto rispetto al piano di riferimento: infatti, la piramide gialla su sfondo verde e bianco viene solo parzialmente rilevata nel lato in cui non sono presenti ombre. Questo problema è solo in parte dovuto al fatto che nella configurazione con cui sono state effettuate le prove non è stato possibile calibrare i parametri dei frame-grabber. Una possibile soluzione consiste nell'elaborare le immagini direttamente a colori. Si è deciso di non percorrere questa strada principalmente perché, operare sui colori, implica un aumento della quantità di dati da elaborare, che passa da una matrice a 256 livelli a 3 matrici, sempre a 256 livelli, e quindi del tempo di calcolo. Per ovviare al problema del tempo, si potrebbero segmentare i colori nelle immagini attraverso l'uso di reti neurali, ma questo è possibile solo in ambienti almeno parzialmente strutturati, in cui sia previsto un numero limitato di colori ben definiti. Questo, però, farebbe perdere in generalità.

In secondo luogo, si nota che i bordi degli ostacoli non hanno inclinazioni sensibilmente diverse tra le due immagini, ed in alcuni punti rischiano di annullarsi. Il motivo più probabile è che la configurazione delle due telecamere non è ottimale per evidenziare le disparità stereo, in particolare gli assi delle due telecamere non sono sufficientemente distanti lateralmente.

Nei laboratori dell'Università di Parma sono state testate sia una configurazione stereo più efficace, che la possibilità di variare i parametri dei frame-grabber in funzione dell'ambiente in cui si trova ad operare il robot. I risultati di questa sperimentazione sono riportati nelle seguenti figure.

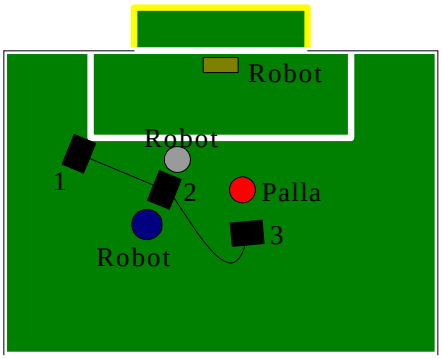


Fig. 4.18 Mappa dell'ambiente di sperimentazione.

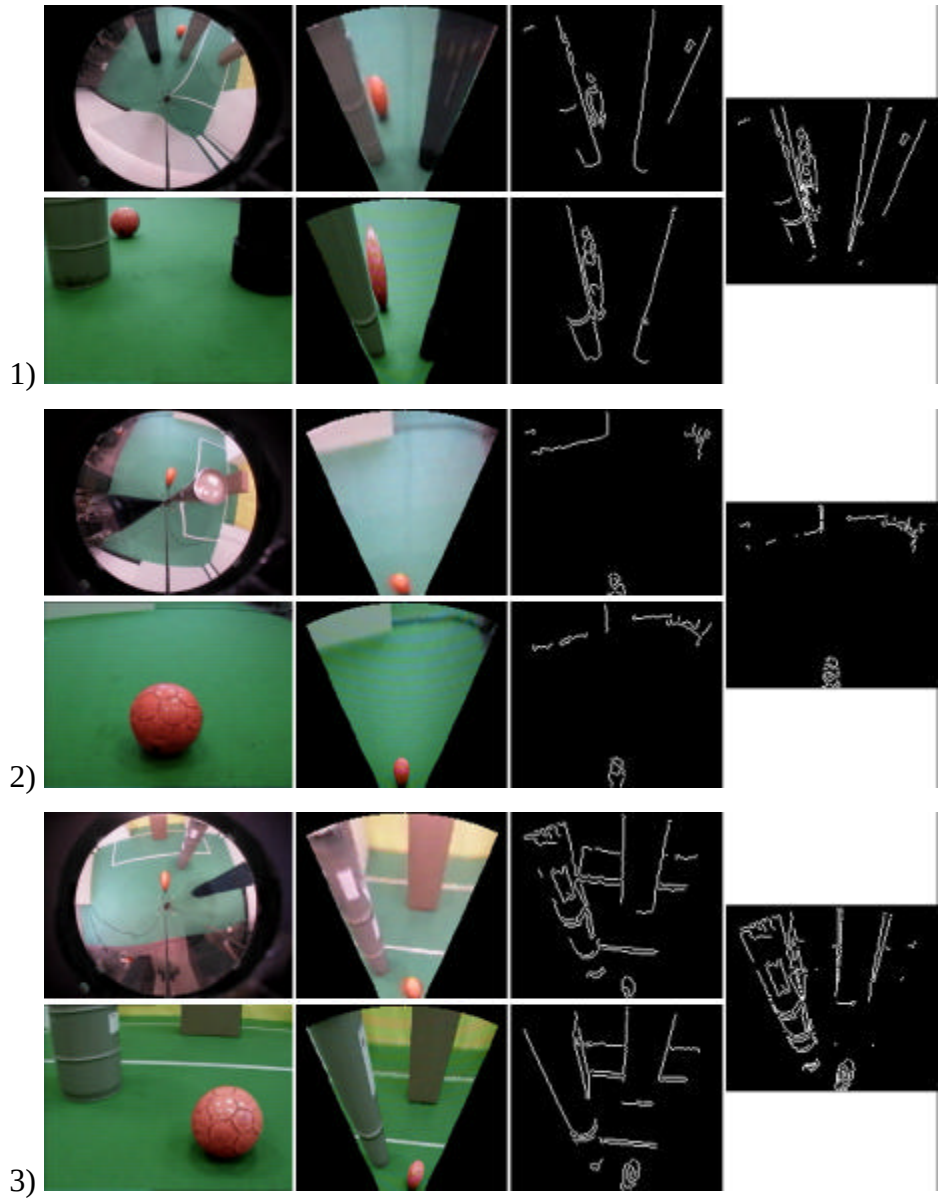


Fig. 4.19 Risultati della sperimentazione effettuata nella configurazione della figura precedente.

Come si può notare, le immagini del sensore omnidirezionale sono meno sature, e questo facilita il funzionamento degli algoritmi di pre-processing illustrati. Inoltre, è migliorata la disparità stereo tra le due immagini, quindi gli oggetti sono più semplicemente riconoscibili.

Una volta ottenuti buoni risultati da questa fase di elaborazione, si dovranno integrare questi moduli di calibrazione prospettica automatica e di pre-processing con un modulo per l'effettivo riconoscimento degli ostacoli in modo autonomo. Un esempio di tecnica di interpretazione di questo tipo di risultati è descritto in [22].

Capitolo 5

Impiego del sistema visivo in due architetture robotiche reali

5.1 Definizione dei requisiti software

Tutti gli algoritmi fin qui visti sono indipendenti dall'architettura robotica, rispondendo al criterio di generalità prefissato all'inizio del progetto. Questo non può valere, però, per gli algoritmi che consentono di interfacciare i moduli software descritti con l'hardware del robot, che sono necessariamente specifici per ogni piattaforma d'interesse.

Le funzionalità di questo tipo di cui si può avere bisogno sono le seguenti:

- acquisizione di immagini statiche da una o entrambe le telecamere;
- acquisizione di sequenze di immagini da una o entrambe le telecamere con un certo frame-rate;
- acquisizione di sequenze di immagini in movimento;
- acquisizione dei set di immagini necessarie per la calibrazione dei sensori.

In questo capitolo viene mostrata la soluzione studiata per questi problemi sulle due architetture software utilizzate.

Mentre i primi due servizi implicano l'utilizzo dei soli frame-grabber per l'acquisizione delle immagini, gli ultimi due prevedono anche la movimentazione del robot e quindi l'utilizzo dei motori. Il primo e l'ultimo sono più rivolti verso algoritmi sequenziali, mentre il secondo ed il terzo necessitano di funzionalità di temporizzazione.

5.2 Integrazione del modulo software di visione in un'architettura basata su CORBA

L'architettura Miró (*MIddleware for scalable mobile RObot application*) [23], sviluppata dall'Università di Ulm, fornisce due tipi di funzionalità: un insieme di servizi per gestire sensori ed attuatori, ed una rete di classi che implementa pattern di progetto per il controllo di robot mobili. Sensori ed attuatori sono modellati come oggetti, che possono essere controllati ed interrogati da loro metodi. In tal modo, i robot possono essere visti come aggregazioni di sensori, attuatori ed oggetti cognitivi, che possono scambiare informazioni e servizi, come gli agenti.

La struttura generale di Miró è illustrata in figura.

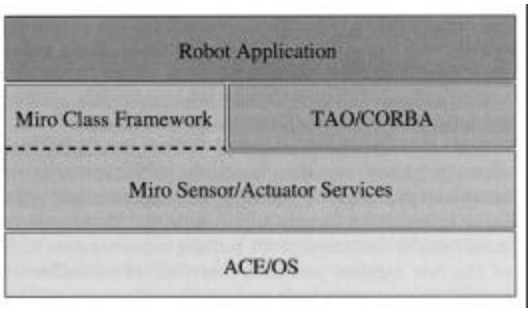


Fig. 5.1 La struttura di Miró.

Una delle astrazioni essenziali introdotte in Miró è che gli oggetti forniscono servizi, non solo interfacce astratte. Questo si rivela molto importante per ottenere piattaforme robotiche indipendenti: ad esempio, questo permette di fornire un servizio di movimento che possa essere implementato in modo consistente e portato su diverse piattaforme robotiche mobili. Inoltre, l'architettura è stata progettata ed implementata applicando rigorosamente tecniche object-oriented, per favorirne la riusabilità, un debug strutturato, scalabilità, mantenibilità e portabilità.

Miró fa un uso estensivo di librerie multi-piattaforma per favorirne la portabilità. L'ambiente ACE (*Adaptive Communications Environment*) fornisce livelli di astrazione orientati agli oggetti per molti sistemi operativi e primitive di comunicazione. L'implementazione di CORBA basata su ACE è rappresentata dal pacchetto TAO. Per la costruzione di interfacce grafiche, come strumento di

monitoraggio e visualizzazione, sono state utilizzate le QT, disponibili per vari sistemi operativi derivati da Unix, come Linux, e per Windows.

La disponibilità di sorgenti per tutte queste librerie aiuta ad aderire a standard aperti ed a superare la dipendenza dalla piattaforma. Per andare oltre le dipendenze dai linguaggi di programmazione, tutti i servizi dei sensori e degli attuatori esportano le loro interfacce come oggetti CORBA network transparent, che possono essere indirizzati da qualsiasi linguaggio e piattaforma per cui esistono binding ed implementazioni CORBA. Questo consente l'integrazione di sottosistemi di alto livello per il controllo di robot, come pianificatori basati su Lisp o interfacce utente basate su Java.

Una chiamata a funzione tradizionale è la via più comoda di interazione tra oggetti, ma non scala bene per tutti i dispositivi sensoriali. Per dare al programmatore più flessibilità, tutti i servizi sensoriali di Miró forniscono anche un modello di comunicazione event-triggered per l'elaborazione data-driven dei dati sensoriali. Questo approccio è abbastanza comune in ambienti real-time. L'implementazione di Miró utilizza il CORBA Notification Service per fornire questa funzionalità.

Il supporto di base per il controllo di più robot discende naturalmente da un ambiente di sviluppo software distribuito. Piccoli gruppi di robot possono indirizzarsi tra loro scambiandosi i riferimenti agli oggetti, se le loro rispettive configurazioni di sensori ed attuatori sono note a tutti. Questo approccio è facilitato dalla disponibilità di una funzionalità di Naming Service, che mette a disposizione un diverso namespace per ogni robot. Per gruppi più numerosi di robot, deve essere fornito un middleware aggiuntivo per supportarne l'interazione: ad esempio, per permettere una collaborazione spontanea tra robot che, inizialmente, non si conoscono.

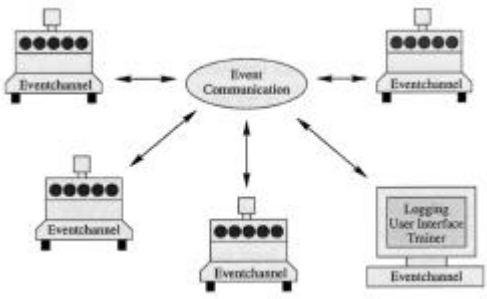


Fig. 5.2 Elaborazione ad eventi per la condivisione di informazioni in uno scenario multi-robot.

Grazie all'ambiente di sviluppo appena descritto, sono stati realizzati gli algoritmi di cui si necessita su un robot commerciale B21 dell'RWI, in dotazione all'Università di Ulm. Questo è equipaggiato con bumper, infrarossi, sonar ed un laser range finder; inoltre, monta due PC a bordo, uno dedicato al controllo di sensori ed attuatori, l'altro all'elaborazione d'immagini.

L'algoritmo per acquisire immagini statiche necessita solamente dei servizi messi a disposizione dai frame-grabber: deve acquisire le risorse hardware, riprendere le immagini dalle due telecamere, salvarle su file e, infine, rilasciare le risorse.

Il programma scritto per acquisire sequenze di immagini con un certo frame-rate prende quanto fatto per l'acquisizione di una sola coppia, e lo periodicizza con la frequenza richiesta. Siccome Miró non fornisce funzionalità per la temporizzazione e la concorrenza, ma permette la scrittura di soli programmi sequenziali, si è deciso di sfruttare un servizio dell'ambiente ACE, come detto di supporto a Miró, che si chiama *Reactor*. Questo gestisce eventi temporali, segnali, eventi basati sul monitoraggio di porte di I/O, oppure notifiche definite dall'utente; nel caso specifico, viene utilizzata la funzionalità che permette di definire le parti di codice da eseguire periodicamente e con che frequenza attivarle. Il programma sequenziale di appoggio continua ad essere eseguito in background, e viene interrotto ogni volta che si scatena l'evento temporale che fa partire il Reactor. In questo caso, il programma di background non fa nulla, restando in attesa di acquisire i prossimi frame. Nel caso del terzo servizio richiesto, quello dell'acquisizione di sequenze di immagini con robot in movimento, l'unica differenza sta nel processo di background, che ora si occupa del controllo degli attuatori, per definire le traiettorie che il robot deve seguire.

Il programma che serve per acquisire le sequenze di immagini necessarie per la calibrazione empirica del sistema visivo è, come il primo, semplicemente sequenziale: ruota il robot di un angolo dipendente dal numero di immagini di cui è composta la sequenza, acquisisce, salva su file la direzione in cui si è effettivamente fermato, e così via finché non ha completato il giro.

5.3 Integrazione del modulo software di visione in un'architettura real-time basata su ETHNOS

L'architettura ETHNOS (Expert Tribe in a Hybrid Network Operating System) [24], sviluppata dall'Università di Genova, è nata per risolvere i seguenti problemi:

- facilitare lo sviluppo in parallelo di diversi progetti, garantendo la loro interoperabilità o, perlomeno, fornendo un prefissato protocollo di comunicazione tra i processi;
- facilitare la progettazione di sistemi in tempo reale, gestendo la schedulazione dei processi;
- gestire processi di diverso tipo: procedurali e simbolici;
- distribuire il carico computazionale su diversi calcolatori, in particolare su una workstation e un robot, senza dover gestire direttamente la comunicazione.

Nel sistema ETHNOS si possono individuare due entità fondamentali: gli *esperti* ed il *kernel*.

Un esperto è un processo che esegue ripetutamente una stessa porzione di codice preposta a risolvere un compito specifico. Il sistema permette la schedulazione di quattro tipi di esperti:

- esperti periodici, che eseguono il proprio compito secondo una scadenza temporale prefissata;
- esperti sporadici, che eseguono il proprio compito in relazione al verificarsi di determinati eventi;
- esperti di background, che eseguono il proprio compito solo quando il processore non è occupato nell'esecuzione di esperti periodici o sporadici;
- esperto di subscheduling, con il compito di schedulare, a sua volta, esperti periodici o sporadici.

Ad esempio, in un eventuale sistema di navigazione per un robot mobile autonomo, il programma che si occupa di gestire le librerie per il controllo dei motori è un esperto, come pure il programma che effettua una ricostruzione sensoriale della topologia dell'ambiente esterno.

Il kernel è un processo con un duplice scopo: si interfaccia con lo scheduler di sistema gestendo sia le priorità degli esperti che le comunicazioni tra di loro. Il kernel, una volta attivato, effettua un'analisi di schedulabilità dell'insieme di esperti e ne imposta le rispettive priorità per ottenere la schedulazione secondo l'algoritmo *Rate Monotonic* (priorità inversamente proporzionale al periodo). L'analisi di schedulabilità non tiene conto delle caratteristiche specifiche di un particolare insieme di esperti (eventuali relazioni armoniche tra i periodi degli esperti che aumentano l'efficienza del rate monotonic) e pone come condizione per la schedulabilità un fattore di utilizzo del processore

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} < 0.69$$

dove C_i è il tempo di esecuzione dell'i-esimo esperto, T_i il suo periodo di attivazione ed n il numero di esperti da schedulare. È anche possibile specificare la priorità che si vuole assegnare ad un dato esperto, estromettendolo dal test di schedulabilità (operazione che viene solitamente effettuata con gli esperti di background e subscheduling).

Ogni volta che ricevono il controllo, gli esperti eseguono in modo sequenziale il codice specificato nel proprio comportamento (in modalità pre-emptive), dopodiché restituiscono il controllo. Per quanto riguarda la gestione della comunicazione, il kernel si occupa di gestire in maniera affidabile e trasparente lo scambio di messaggi tra esperti, effettuando un broadcasting efficiente dell'informazione (ogni messaggio è disponibile per la lettura a tutti gli esperti che desiderano leggerlo) con una tecnica di tipo *publish/subscribe*.

Il principio di comunicazione è il seguente:

- ogni esperto che vuole ricevere un determinato tipo di messaggio ne fa richiesta al kernel;
- ogni esperto che decide di produrre e condividere un messaggio lo passa al kernel, senza preoccuparsi di conoscerne la destinazione.

Il kernel garantisce che il messaggio prodotto sia ricevuto da tutti gli esperti che ne hanno fatto richiesta. Perché questo avvenga in maniera efficiente, è necessario suddividere i messaggi in 'tipi'. A titolo di esempio, si consideri il sistema di navigazione illustrato in figura:

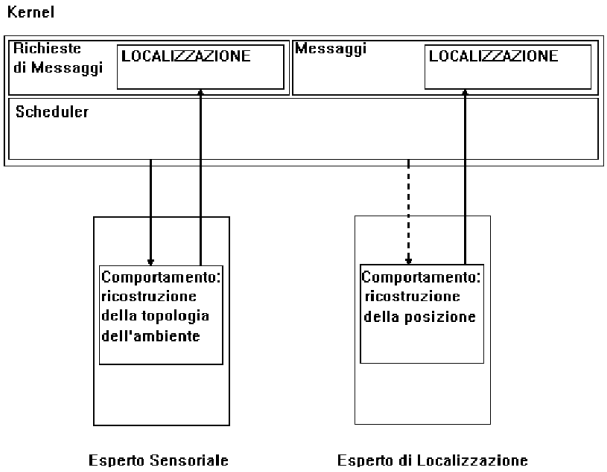


Fig. 5.3 Esempio di comunicazione. L'Esperto Sensoriale ha fatto richiesta di messaggi di tipo *localizzazione*. L'Esperto di Localizzazione produce, come risultato del suo comportamento, messaggi di tipo *localizzazione*. Ogni volta che il Kernel riceve messaggi dall'Esperto di Localizzazione li rende disponibili per la lettura all'Esperto Sensoriale. Ogni comunicazione avviene tramite la mediazione del Kernel, che effettua un broadcasting dell'informazione.

Inoltre, ETHNOS permette di costruire un sistema distribuito istanziando diversi kernel su differenti elaboratori, ognuno con il compito di gestire un certo numero di esperti; può succedere che un esperto faccia richiesta di messaggi che vengono prodotti da un esperto posto su un elaboratore remoto. Il sistema di comunicazione è completamente trasparente per il programmatore. Anche in questo caso, quindi, è il kernel stesso, comunicando con il kernel del sistema remoto, a preoccuparsi che ogni messaggio venga recapitato a tutti gli esperti che ne hanno fatto richiesta.

Grazie a questo ambiente di sviluppo, sono stati realizzati gli algoritmi di cui si necessita su un robot auto-costruito dell'Università di Parma. Questo è equipaggiato con bumper ed infrarossi; inoltre, monta un PC a bordo che esegue tutta l'elaborazione.

Questo ambiente è ottimale per servizi come l'acquisizione di sequenze di immagini con un dato frame-rate, con robot sia fermo che in movimento. Si tratta di creare un esperto periodico di acquisizione delle immagini, con periodo di attivazione pari all'inverso del frame-rate richiesto, ed, eventualmente, un altro esperto periodico che gestisce il movimento del robot interfacciandosi agli attuatori.

Il programma per l'acquisizione delle sequenze di immagini necessarie per la calibrazione empirica del sistema è stato realizzato grazie a due esperti: uno sporadico, che gestisce l'acquisizione delle immagini, ed uno periodico, che controlla la rotazione del robot ed il salvataggio su file delle posizioni a cui si è fermato per acquisire. Questi due esperti si sincronizzano a vicenda: a robot fermo, l'esperto di movimento manda un messaggio a quello di acquisizione per sbloccarlo ed, a fine acquisizione, questo invia un altro messaggio a quello di movimento indicandogli che può far ripartire la rotazione del robot verso l'obiettivo successivo.

Un'ultima differenza da notare tra le due architetture software provate è che nella prima i frame-grabber sono gestiti attraverso i driver delle periferiche stesse, mentre nella seconda viene utilizzata la libreria Video4Linux, che permette un buon livello di astrazione, essendo indipendente dalla periferica di acquisizione utilizzata.

Conclusioni

In questa tesi è stato presentato il lavoro svolto nella progettazione, realizzazione ed impiego di un sistema visivo di tipo HOPS (*Hybrid Omnidirectional/Pin-hole Sensor*), composto da un catadiottro per la visione omnidirezionale ed una telecamera a CCD, per associare, ai vantaggi di una visione foveale sulla regione frontale al sistema, quelli di un'ampia visione periferica fornita dal catadiottro. Il sistema è stato pensato per l'impiego in robot mobili autonomi: infatti, è applicabile a problemi di auto-localizzazione in ambienti semi-strutturati, nella ricerca ad ampio raggio di oggetti di interesse, nell'analisi di percorsi, ed in tutte quelle applicazioni che richiedono un ampio campo visivo.

Il lavoro svolto è stato suddiviso in tre parti. La prima consiste nella progettazione e nella realizzazione del sistema visivo, con particolare riferimento al profilo dello specchio di cui è costituito il sensore catadiottrico. Il risultato ottenuto è un sensore facilmente replicabile e modificabile, che può essere impiegato in diverse configurazioni. La seconda parte è focalizzata sulla calibrazione del sistema visivo in modo robusto ed automatico. Questo processo è stato adattato a due diverse configurazioni del sistema ottenendo, in entrambi i casi, buoni risultati in termini di precisione. La terza ed ultima parte riguarda la realizzazione, tramite il modello della prospettiva inversa, della funzionalità di pre-processing per l'individuazione di ostacoli presenti nel campo visivo comune alle due telecamere. Questo ha dato risultati soddisfacenti sia in termini di flessibilità al cambiamento dell'ambiente di lavoro, che in termini di tempi di elaborazione. Il test del sistema è stato effettuato in svariati ambienti indoor, con diverse configurazioni del sistema visivo e su due architetture robotiche differenti.

Gli sviluppi futuri si concentreranno prevalentemente in due direzioni: quella dell'integrazione del modulo di pre-processing con uno di individuazione ostacoli autonomo, e quella della realizzazione di un set di funzionalità visive completo che sfrutti tutte le potenzialità di HOPS prendendo le basi da quelle già sviluppate.

Ringraziamenti

Desidero ringraziare, innanzitutto, il Prof. Giovanni Adorni per avermi dato la possibilità di svolgere questo lavoro, che mi ha consentito di affrontare molti problemi della visione artificiale attraverso un progetto articolato, comprendente diversi aspetti teorici ed applicativi, e per tutti i progetti a cui mi ha permesso di partecipare negli ultimi due anni e mezzo. Ringrazio, inoltre, l'Ing. Stefano Cagnoni per la disponibilità e la pazienza con cui ha seguito i miei lavori, e l'Ing. Monica Mordonini, non solo per la presenza costante ed i preziosi consigli, ma soprattutto per l'amicizia sempre dimostrata in questi anni di "avventure".

Un ringraziamento particolare va a tutto il Dipartimento di Elaborazione di Informazioni Neurali e all'officina meccanica dell'Università di Ulm per avermi ospitato e per avermi sempre reso disponibile tutto ciò di cui ho avuto bisogno nei cinque mesi di mia permanenza. Ricordo il Dott. Gerhard Kraetzschmar, il mio "relatore" tedesco, la segretaria del dipartimento, che ha semplificato molte delle pratiche burocratiche che ho dovuto espletare ed, in particolare, il Dott. Stefan Sablatnög, per il quotidiano e preziosissimo aiuto.

Vorrei anche ringraziare tutti i ragazzi del gruppo storico del laboratorio di robotica mobile (Carlo Bernardi, Cristiano Rota, Enzo Bocelli e Giovanni Remondini) e gli ultimi arrivati (Giovanna Caruso e Gianluca Contesso) per aver contribuito a creare un ambiente di lavoro sempre piacevole e stimolante, senza dimenticare i preziosi momenti di svago.

Un ringraziamento speciale va alla mia famiglia e ad Elisa, perché sono riusciti ad essermi sempre vicini, soprattutto nei momenti più difficili, e per avermi sempre sopportato.

Bibliografia

[1] Ballard D.H., Brown C.M. *Computer Vision*. Prentice-Hall, Englewood Cliffs, 1982.

[2] Brady M. *Artificial Intelligence and Robotics*. Artificial Intelligence and Robotics, Vol. 26, pp. 79-121, 1985.

[3] Arkin R.C. *Behavior-Based Robotics*. The MIT Press, Cambridge MA, 1998.

[4] Marr D. *Vision: a Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman, New York, 1982.

[5] Bajcsy R. *Active Perception*. Proc. IEEE, Vol. 76, N.8, pp. 996-1005, 1988.

[6] Ullman S. *Visual Cognition*. Ed. S. Pinker, The MIT Press, Cambridge MA, 1985.

[7] Carlson A.B. *Communication Systems: an Introduction to Signals and Noise in Electrical Communication*. 3rd ed. McGraw-Hill International Editions (Electrical & Electronic Engineering Series), Singapore, 1986.

[8] Guil N., Villalba J., Zappata E.L. *A Fast Hough Transform for Segment Detection*, IEEE Trans. on Image Processing, Vol. 11, N. 4, pp. 1541-1548, 1995.

[9] Bertozzi M., Broggi A. *Real-Time Lane and Obstacle Detection on the GOLD System*, IEEE Computer Society – Proc. of the IEEE Intelligent Vehicles '96 (Tokyo), pp. 213-218, 1996.

[10] Yagi Y., Nishizawa Y., Yachida M. *Map-Based Navigation for a Mobile Robot with Omnidirectional Image Sensor COPIS*. IEEE Trans. on Robotics and Automation, Vol. 11, N. 5, pp. 634-648, 1995.

[11] Marchese F., Sorrenti D.G. *Omni-Directional Vision with a Multi-Part Mirror*. 4th International Workshop on RoboCup, pp. 289-298, 2000.

[12] Hicks R.A., Bajcsy R. *Reflective Surfaces as Computational Sensors*. IEEE Workshop on Perception for Mobile Agents – Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR99), pp. 82-86, 1999.

[13] Adorni G., Cagnoni S., Carletti M., Mordonini M., Sgorbissa A. *Designing Omnidirectional Vision Sensors*. AI*IA Notizie, Vol. 15, N. 1, pp. 27-30, 2002.

[14] Tsai R.Y. *A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses*. IEEE Journal of Robotics and Automation, Vol. RA-3, N. 4, pp. 323-344, 1997.

[15] Faugeras O. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. The MIT Press, Cambridge MA, 1993.

[16] Adorni G., Cagnoni S., Mordonini M. *An Efficient Perspective Effect Removal Technique for Scene Interpretation*. Proc. of the Asian Conference on Computer Vision, pp. 601-605, 2000.

[17] Mallot H.A., Bulthoff H.H., Little J.J. Bohrer S. *Inverse Perspective Mapping Simplifies Optical Flow Computation and Obstacle Detection*. Biological Cybernetics, Vol. 64, pp. 177-185, 1991.

[18] Tsai R.Y. *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision*. Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR96), 1996.

[19] Bonarini A., Aliverti P., Lucioni M. *An Omnidirectional Vision System for Fast Tracking for Mobile Robots*. IEEE Trans. on Instrumentation and Measurement, Vol. 49, N. 3, pp. 509-512, 2000.

[20] Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. *Numerical Recipes in C: The Art of Scientific Computing*. 2nd ed. Cambridge University Press, Cambridge, 1992.

[21] Mittal M., Peleg A., Weiser U. *MMX™ Technology Architecture Overview*. Intel Technology Journal Q3, 1997.

[22] Adorni G., Bolognini L., Cagnoni S., Mordonini M. *A Non-Traditional Omnidirectional Vision System with Stereo Capabilities for Autonomous*

Robots, in F. Esposito (ed.) AI*IA 2001: Advances in Artificial Intelligence – 7th Congress of the Italian Association for AI, Bari, Italy, September 2001. Proceedings (LNAI 2175), Springer, pp. 344-355, 2001.

[23] Utz H., Sablatnög S., Enderle S., Kraetzschmar G., Palm G. *Miró – Middleware for Mobile Robot Applications*. NeuroInformatik Abteilung (Università di Ulm), 2001.

[24] Piaggio M., Sgorbissa A., Tricerri L. *ETHNOS Kernel User Manual*. Ver. 4.2.1. DIST (Università di Genova), 1999.

Link utili:

[25] http://www.informatik.uni-ulm.de/ni/index_e.html

[26] http://www.dai.ed.ac.uk/Cvonline/

[27] http://www.dai.ed.ac.uk/HIPR2/

[28] http://cedar.intel.com/cgi-bin/ids.dll/topic.jsp?catCode=BMJ

[29] http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/20726.pdf