DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

Università degli Studi di Padova

FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria dell'Automazione

corso di Progettazione di Sistemi di Controllo

Patrolling and tracking in 3D space

Antonio Balsemin

Francesco Scotton

Marco Todescato

Anno Accademico 2011-2012

Indice

Capitolo 1

Introduzione

Al giorno d'oggi la maggior parte dei sistemi di videosorveglianza sono basati sul controllo visivo gestito da operatori umani. Questa implementazione presenta una serie di svantaggi come ad esempio il fatto che l'operatore può non essere in grado di mantenere la concentrazione in situazioni di emergenza e di sorvegliare più di qualche monitor contemporaneamente. Nasce quindi la necessità di sfruttare sistemi dotati di intelligenza in grado di gestire autonomamente la sorveglianza [2]. E' da tener presente però come non sia facile sostituire l'operatore data la sua capacità di discriminare eventi di ogni tipo. Un sistema complementamente automatico risulta meno flessibile e potrebbe essere soggetto ad errori.

Effettuare il **patrolling** di un perimetro significa continuare a percorrere il percorso assegnato in modo tale da visitarne ogni punto il più spesso possibile, per proteggerlo o semplicemente per osservarne lo stato.

Per **tracking** si intende la capacità di un sistema di riconoscere ed inseguire un evento improvviso (target) [2],[3].

Lo scopo di questo progetto è l'implementazione di un algoritmo efficiente, robusto e scalabile che gestisca il patrolling e il tracking per un sistema di videosorveglianza dotato di videocamere PTZ (Pan Tilt Zoom) Ulisse (figura 1.1).

L'allestimento del laboratorio del dipartimento, costituito da videocamere Ulisse ciascuna dotata di un pc "personale", ha suggerito una struttura dell'algoritmo di tipo decentralizzato. Questo comporta dei vantaggi nella scalabilità dell'algoritmo: una struttura centralizzata avrebbe comportato una crescita della complessità al crescere del numero di videocamere.

1.1 Stato dell'arte

In questa sezione si vuole fornire al lettore una panoramica di quello che è già presente in letteratura riguardo il patrolling e il tracking, sia in ambito scientifico che in quello commerciale.

Ambito scientifico

Il nostro progetto inquadra due argomenti correlati ma diversi: il *multi-agent patrolling* e il *tracking*.

Figura 1.1: Videocamera Ulisse

Nel multi-agent patrolling un team di agenti (nel nostro caso le videocamere) si occupa di visitare ripetutamente un set di punti in modo da monitorare eventuali cambiamenti. Il patrolling non è quindi ristretto solo all'ambito della videosorveglianza, ma trova applicazione in molti altri campi (in cui cambia la definizione degli agenti) come: sistemi di difesa militari, veicoli sottomarini o robot mobili che monitorano particolari aree, industrie nucleari (per il controllo di perdite radioattive) e molti altri. In [1] viene effettuata un'analisi teorica del problema confrontando diverse strategie di patrolling e le loro complessità computazionali.

In [11] è descritta una soluzione di tipo distribuito per un problema di videosorveglianza multi-agent che prevede anche modifiche perimetrali e perdite di agenti.

In [12] sono proposte tre strategie di controllo distribuite per il patrolling in reti di videocamere che si differenziano in base al tipo di comunicazione tra gli agenti: sincrona, asincrona gossip simmetrica e asincrona gossip asimmetrica. Nei tre casi vengono proposte le relazioni per ricavare gli estremi di competenza di ciascuna videocamera supponendo che l'area di patrolling sia una linea 1D.

In [3] è stata proposta una soluzione per estendere l'approccio di [12] al caso 3D. Inoltre l'autore propone un metodo di controllo della velocità, uno dello zoom e un'architettura per l'algoritmo di patrolling. Vengono anche riportati i test dei risultati ottenuti che si limitano però solo al caso di due videocamere Ulisse PTZ che eseguono il patrolling lungo una retta nel pavimento.

In [2] è proposto un algoritmo in grado di gestire patrolling e tracking in una rete di videocamere: esso calcola le sezioni di copertura ottime anche in presenza di vincoli fisici di copertura e utilizza il filtro di Kalman per seguire un evento improvviso. Gli autori effettuano anche delle simulazioni in ambiente Simulink per verificare la bontà del metodo proposto. Inoltre si cerca di trattare il tracking solo come problema di inseguimento coordinato rimanendo lontani dal campo della visione.

In effetti anche nel nostro lavoro è stato seguito questo approccio: è stato analizzato il tracking più come problema di controllo per l'inseguimento di un determinato evento, trattando solo in maniera generale tutto ciò che rientrava nell'ambito della visione. E' stato ricercato quindi un algoritmo robusto che potesse essere da noi utilizzato e interfacciato con la parte relativa al controllo delle videocamere.

In [13] è sviluppato un algoritmo di tracking per telecamere mobili in grado di inseguire oggetti (target) di qualsiasi tipo e di reagire in maniera efficiente nel caso di cambi di posa e occlusioni. Esso prevede inoltre una fase di ricerca del target che si sta inseguendo qualora venga perso a causa di occlusioni totali.

In [4] è presentato un algoritmo di tracking per videocamere PTZ basato sulla combinazione del filtro di Kalman esteso (EKF) e del filtro particellare (PF). EKF è utilizzato per predirre la posizione futura del target mentre PF per cercarlo qualora venga perso. In questo articolo gli autori propongono inoltre un modello per le videocamere PTZ.

In [14] è proposto un algoritmo simile a quello di [13] applicato alle videocamere PTZ. Gli autori descrivono anche un semplice metodo di controllo delle videocamere.

Ambito commerciale

Le reti di videocamere più diffuse in ambito commerciale sono quelle che sfruttano la tecnologia video di rete. Essa consiste nell'uso di reti cablate o wireless con tecnolgia TCP/IP per la trasmissione di video e audio. In particolare, le telecamere che usano questa tecnologia vengono dette "telecamere IP": esse integrano in un unico dispositivo anche una sorta di minicomputer in grado di svolgere le operazioni necessarie per l'elaborazione delle immagini, la compressione, l'analisi video e le funzionalità di rete.
Il vantaggio principale delle telecamere IP è che possono essere collegate in un qualsiasi punto della rete e non presentano quindi il vincolo di essere poste in prossimità di sistemi di acquisizione o unità di controllo. In questo modo è possibile accedere tramite remoto, per esempio un semplice smartphone, ai flussi video delle telecamere.
Le telecamere IP sono presenti in commercio sia con la sola funzione di ripresa ma anche con funzioni molto più complesse come la rilevazione e l'inseguimento automatico di un evento. Interessante si è rivelata la guida di Axis Communication, uno dei leader in questo settore (vedere [15]).
Nel nostro lavoro non sono state utilizzate telecamere IP descritte sopra: ogni telecamera è provvista di un pc "personale" e i vari pc sono collegati tra di loro per mezzo di una rete LAN che permette lo scambio di dati utilizzando un protocollo di comunicazione di tipo TCP/IP.
Il controllo di perimetri con PTZ è in genere un *controllo di posizione*. Infatti viene fornita all'utente la possibilità di memorizzare una serie di posizioni fisse tra le quali le videocamere PTZ continuano a muoversi secondo un ordine prestabilito o in maniera casuale. Spesso è presente anche un'interfaccia grafica che consente di selezionare il perimetro direttamente da monitor, "cliccando" in corrispondenza dei punti che delimitano il perimetro.
Non si trovano soluzioni nelle quali è adottato un *controllo in velocità* come in questo lavoro (vedere capitolo 5).
Interessante il prodotto Siveillance SiteIQ Wide Area di Siemens, un sistema di videosorveglianza intelligente che consente di controllare un'ampia area tramite una serie di videocamere fisse e PTZ e altri sensori (come barriere virtuali) e di rilevare, classificare e seguire eventuali eventi anomali. Inoltre esso è in grado di fornire tutte le informazioni necessarie all'operatore tramite un unico monitor su cui si ha a disposizione una mappa generale dell'area controllata e vengono segnalati e classificati una vasta gamma di eventi.

Molto diffuso e sviluppato infatti è il tracking: numerosi sono i software in commercio che consentono un tracking preciso di eventi di ogni tipo e in ogni condizione dell'ambiente (vedere ad esempio il Vi-System di Agentvi). Le telecamere PTZ vengono generalmente utilizzate per seguire l'evento (entro l'area di copertura possibile) tramite comandi successivi "in posizione".
Interessante si è rivelato [16] in cui si può trovare un'ampia descrizione dello stato dell'arte commerciale in ambito di videosorveglianza intelligente. E' proposta infatti un'ottima classificazione dei vari sistemi esistenti di smart surveillance e di alcuni progetti di ricerca ongoing.

1.2 Il nostro algoritmo

L'algoritmo implementato è in grado di gestire il patrolling e il tracking per una rete di videocamere. Esso è di tipo decentralizzato nel senso che ciascuna videocamera della rete ha informazioni relative solo alle videocamere ad essa adiacenti.
In particolare, esso prevede una fase offline in cui viene definita una traiettoria arbitraria in 3D, interpolante una serie di punti dello spazio: le videocamere della rete effettueranno il pattugliamento di tale curva dividendosela in maniera ottima. E' questa la principale novità di questo progetto, dato che in genere il pattugliamento è effettuato seguendo una traiettoria 1D.
L'algoritmo risulta essere oltre che scalabile, grazie al fatto che è decentralizzato, anche molto robusto ai guasti. Esso reagisce in maniera ottima ad ogni tipo di guasto, perfino nel caso di rotture di più di una videocamera della rete.
Un ulteriore aspetto interessante dell'algoritmo è che ciascuna videocamera è controllata in velocità per seguire la traiettoria pianificata e non in posizione, come presentano molti dei sistemi di videosorveglianza attuali dotati di telecamere PTZ. Solitamente il movimento delle videocamere è infatti tra una serie di punti fissati e risulta molto discontinuo. Il controllo in velocità realizzato garantisce invece una maggior fluidità dei movimenti.
Seguendo questa linea, l'algoritmo implementa un'analoga soluzione di controllo in velocità anche per il tracking: la videocamera è infatti movimentata in maniera coerente agli spostamenti del target tramite comandi in velocità. In particolare, essa è in grado di adattare la propria velocità in funzione di quella del target e di inquadrarlo perfettamente al centro del monitor nel caso in cui si fermi. Inoltre, qualora una o più videocamere fossero impegnate nel tracking dello stesso evento o di eventi diversi, quelle rimanenti nella rete si ridividono la traiettoria da sorvegliare in maniera ottima, compatibilmente con i propri limiti di movimento.
L'algoritmo mette a disposizione della videocamera anche un controllo di zoom: è infatti possibile fissare le dimensioni con cui si vuole visualizzare un determinato oggetto che, grazie al controllo, vengono mantenute invariate anche al variare della distanza dalla telecamera.
Infine, l'algoritmo è stato ampiamente testato in una rete reale di 3 videocamere PTZ che eseguono il patrolling di una traiettoria 3D, a differenza di [3] in cui ci si è limitati a testare un algoritmo simile solo lungo una retta posta nel pavimento pattugliata da 2 videocamere PTZ.

1.3 Struttura della relazione

In questa sezione si riassume brevemente il contenuto di ciascun capitolo della relazione.

- **Capitolo 2, Traiettoria di patrolling:** viene descritto in dettaglio il nostro metodo per creare una traiettoria arbitraria dello spazio. In sostanza, viene descritto un metodo per ricavare una curva interpolante n punti fissati a priori, costruita concatenando $n-1$ polinomi cubici. Tale curva è parametrizzata in maniera intelligente in modo da poter ricondurre il problema al caso 1D, in cui è più facile gestire la suddivisione ottima della traiettoria. E' riportata alla fine del capitolo anche la ricostruzione nel laboratorio della spline 3D scelta.

- **Capitolo 3, Calibrazione:** è descritto il modello utilizzato per la videocamera e la procedura per ricavare i parametri ad esso relativi, detta calibrazione. La calibrazione è stata effettuata seguendo [7].

- **Capitolo 4, Cinematica:** dato che le videocamere accettano solo comandi di posizione (angolare) per i motori, in questo capitolo è descritto in dettaglio il metodo, basato su [4], che è stato utilizzato per poter muovere la videocamera in modo che inquadri un determinato punto dello spazio 3D. Vengono riportati alla fine del capitolo anche dei test per valutare la bontà del metodo sviluppato.

- **Capitolo 5, Controllo videocamere:** sono descritti il controllo di velocità, utilizzato per assicurare un movimento fluido della videocamera, e quello di zoom, che consente di inquadrare un oggetto sempre con stesse dimensioni, indipendentemente dalla distanza dello stesso dalla videocamera. Per entrambi i controlli vengono riportati dei test e un'analisi delle problematiche che si sono presentate.

- **Capitolo 6, Comunicazione videocamere e partitioning:** in questo capitolo viene descritta l'architettura della comunicazione tra le videocamere e la parte di algoritmo, basata su [12], relativo alla suddivisione ottima della traiettoria scelta, detto partitioning. Alla fine del capitolo viene anche spiegata l'interfaccia grafica che è stata sviluppata per rendere più intuitiva la comprensione del funzionamento del partitioning a chi usa l'algoritmo.

- **Capitolo 7, Tracking:** è analizzata la parte dell'algoritmo relativa al tracking che si basa su [13]. Vengono proposte due tecniche per inseguire il target, una basata sul controllo di posizione e l'altra su quello in velocità, evidenziandone gli aspetti positivi e negativi di ciascuno.

- **Capitolo 8, Conclusioni e sviluppi futuri:** vengono commentati i risultati ottenuti e proposti i possibili sviluppi futuri del nostro lavoro.

Capitolo 2

Traiettoria di patrolling

Uno dei principali obbiettivi del progetto è quello di riuscire ad implementare un algoritmo di patrolling lungo arbitrarie traiettorie 3D. Per poter sfruttare quanto di già implementato in letteratura su curve 1D, è stato necessario definire una curva 3D mappata in ascissa curvilinea (s) in modo da rettificarne l'andamento e trattarla proprio come una qualunque retta 1D. Per definizione di ascissa curvilinea è necessario avere a disposizione la lunghezza effettiva della curva, cosa a priori non possibile. Per questo motivo si è deciso di procedere in due passi. Per prima cosa creare una curva parametrizzata attraverso una generica variabile adimensionale. Successivamente, calcolando la lunghezza della curva creata, riparametrizzarla in ascissa curvilinea. Tra le diverse famiglie di curve utilizzabili abbiamo scelto le Spline e cioè particolari funzioni costituite da un insieme di polinomi raccordati tra loro.
A questo punto risulta doveroso inoltrarsi maggiormente nel problema partendo dal principio, al fine di garantire al lettore una comprensione più chiara e completa.

2.1 Definizione di una curva

Una semplice metodologia con la quale definire una curva è quella di darne dei punti di passaggio anche detti *punti di controllo*. In questo caso la curva viene detta *interpolante* i punti stessi.
Ovviamente definire i punti di passaggio non basta. E' necessario scegliere la famiglia di funzioni da utilizzare ed alla quale imporre il passaggio attraverso i punti desiderati.
Un esempio è rappresentato dalle curve polinomiali ovvero definite attraverso un polinomio di grado desiderato. Per riuscire ad identificare univocamente un polinomio di grado n è d'altronde necessario imporre esattamente $n+1$ vincoli indipendenti in modo da individuarne univocamente i coefficienti.
Comunemente la scelta ricade sui polinomi cubici del tipo

$$y = a + bx + cx^2 + dx^3$$

dal momento che richiedono un esiguo numero di vincoli per essere definiti, ma allo stesso tempo individuano curve poco oscillanti e contenenti flessi, i quali consentono alla funzione di assumere andamenti molto variegati.

Si riporta un semplice esempio che tornerà utile per la comprensione dei passi successivi.
Scegliamo un polinomio cubico da far passare attraverso i punti di controllo (-1,2),(0,0),(1,-2),(2,0). Per ricavare i coefficienti a, b, c, d impostiamo il seguente sistema:

- **passaggio per (-1,2):** a-b+c-d=2
- **passaggio per (0,0):** a = 0
- **passaggio per (1,-2):** a+b+c+d=-2
- **passaggio per (2,0):** a+2b+4c+8d=0

che in forma matriciale diventa

$$\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -2 \\ 0 \end{bmatrix}$$

Risolvendo il sistema si giunge ai valori dei coefficienti che risultano pari a

$$a = 0, b = -\frac{8}{3}, c = 0, d = \frac{2}{3}$$

2.2 Curve definite a tratti e parametrizzazione

Definite le curve polinomiali si tratta di:

- individuare curve definite a tratti;
- parametrizzare tali curve.

Curve definite a tratti

L'idea è semplice: dal momento che definire una curva interpolante n punti attraverso un unico polinomio risulta poco vantaggioso perché l'andamento risulta molto oscillante e sensibile a variazioni anche piccole nel valore dei punti, si decide di definire la curva a tratti. In sostanza questo significa collegare opportunamente uno di seguito all'altro un certo numero di polinomi di grado inferiore.
A questo scopo, si impone il passaggio di ciascun tratto per i punti di controllo. Tuttavia in questo modo il sistema di equazioni (del tipo analogo alla sezione 2.1) risulta indeterminato.
E' conveniente quindi imporre la continuità C^1 e C^2 ai tratti consecutivi e cioè forzare i polinomi ad avere ugual valore di derivate prima e seconda nei punti di collegamento. Purtroppo questo non è ancora sufficiente: infatti per $n-1$ polinomi cubici (individuati da n punti di controllo) sono necessarie $4(n-1)$ equazioni, mentre il passaggio per i punti di controllo e le continuità C^1 e C^2 portano rispettivamente a $2(n-1)$ e $2(n-2)$ equazioni e cioè in totale a $4n-6$ equazioni. Le ultime due equazioni vengono individuate imponendo due valori

fittizi alle derivate prime nei punti iniziale e finale le quali non erano ancora state definite.

Questo procedimento verrà chiarito nell'esempio seguente.
Supponiamo di voler interpolare i punti (0,1),(2,2),(5,0),(8,0) utilizzando 3 polinomi cubici $[y = a_i + b_i x + c_i x^2 + d_i x^3; i = 0, 1, 2]$ ognuno dei quali definito su una coppia consecutiva di punti. Come descritto sopra, il numero di equazioni (6) risulta inferiore al numero di parametri (3*4) da individuare: è necessario specificarne altri $3 * 4 - 6 = 6$.
Imponendo la continuità C^1 e C^2 ai tratti consecutivi si individuano 4 equazioni aggiuntive. Le ultime due le ricaviamo imponendo due valori fittizi alle derivate prime nei punti iniziale e finale. Il sistema che ne risulta è pari a:

polinomio tra (0,1) e (2,2) [tratto f_0]	
pendenza inziale	$b_0 = 2$
passaggio per (0,1)	$a_0 = 1$
passaggio per (2,2)	$a_0 + 2b_0 + 4c_0 + 8d_0 = 2$
continuità C^1	$b_0 + 4c_0 + 12d_0 - b_1 - 4c_1 - 12d_1 = 0$
continuità C^2	$2c_0 + 12d_0 - 2c_1 - 12d_1 = 0$

polinomio tra (2,2) e (5,0) [tratto f_1]	
passaggio per (2,2)	$a_1 + 2b_1 + 4c_1 + 8d_1 = 2$
passaggio per (5,0)	$a_1 + 5b_1 + 25c_1 + 125d_1 = 0$
continuità C^1	$b_1 + 10c_1 + 75d_1 - b_2 - 10c_2 - 75d_2 = 0$
continuità C^2	$2c_1 + 30d_1 - 2c_2 - 30d_2 = 0$

polinomio tra (5,0) e (8,0) [tratto f_2]	
passaggio per (5,0)	$a_2 + 5b_2 + 25c_2 + 125d_2 = 0$
passaggio per (8,0)	$a_2 + 8b_2 + 64c_2 + 512d_2 = 0$
pendenza finale	$b_2 + 8c_2 + 128d_2 = 1$

In maniera analoga all'esempio della sezione 2.1 è semplice mettere il tutto in forma matriciale e ottenere il valore dei parametri $a_i, b_i, c_i, d_i; i = 0, 1, 2$.

Parametrizzazione

Avendo a disposizione la curva a tratti è necessario esprimere ogni tratto costituente la curva rispetto un parametro adimensionale generico, i.e. si tratta di parametrizzare la curva.
Si sceglie innanzitutto di parametrizzare ciascun tratto rispetto un parametro $u_i \in [0, 1], i = 1, 2, ..., n - 1$, dove n è il numero di punti di controllo e quindi $n - 1$ il numero dei tratti di curva. A questo scopo si pone:

$$u_i = \frac{x - x_i}{x_{i+1} - x_i} \qquad x \in [x_i \ x_{i+1}] \tag{2.1}$$

dove x_i e x_{i+1} sono i punti di controllo tra i quali è definito l'intervallo i-esimo. In seguito si capirà più in dettaglio il vantaggio di questa parametrizzazione. Considerando quindi il generico polinomio cubico nel parametro u_i

$$f_i(u_i) = a_i + b_i u_i + c_i u_i^2 + d_i u_i^3$$

si può imporre la continuità C^0 ,ovvero il passaggio per i punti di controllo, semplicemente facendo passare la curva nella y desiderata per $u_i = 0$ e $u_i = 1$ ovvero all'inizio e alla fine del segmento:

$$f_i(0) = a_i = y_i$$

$$f_i(1) = a_i + b_i + c_i + d_i = y_{i+1}$$

Per le continuità C^1 e C^2 si sfrutta la regola di derivazione composta e si ottiene rispettivamente:

$$D_x f_i = \frac{\partial f_i}{\partial u_i}\frac{du_i}{dx} = f_i' \frac{1}{x_{i+1} - x_i} \tag{2.2}$$

$$D_x^2 f_i = \left(\frac{\partial f_i'}{\partial u_i}\frac{du_i}{dx}\right)\frac{du_i}{dx} = f_i'' \frac{1}{(x_{i+1} - x_i)^2} \tag{2.3}$$

A questo punto si forza il match delle pendenze e delle curvature tra un tratto di curva e il successivo ovvero

$$(D_x f_i)(1) = (D_x f_{i+1})(0)$$

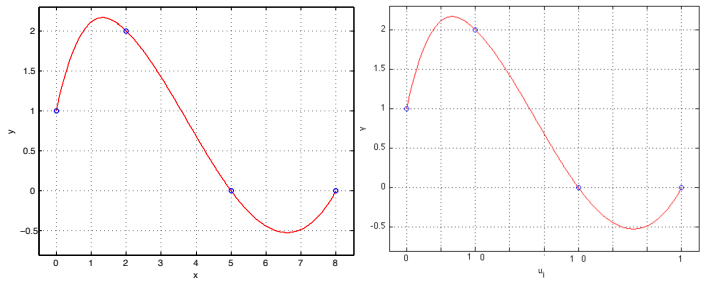
$$(D_x^2 f_i)(1) = (D_x^2 f_{i+1})(0)$$

Infine è necessario garantire i vincoli "extra" sulle pendenze dei punti iniziale e finale al fine di raggiungere il numero adeguato di vincoli:

$$\frac{f_0'(0)}{x_1 - x_0} = s_0$$

$$\frac{f_{n-1}'(1)}{x_n - x_{n-1}} = s_n$$

dove s_0 e s_n rappresentano i valori di pendenza desiderati.
Nella figura seguente (2.1) si nota sia la definizione a tratti che la parametrizzazione scelta.



(a) $f_0 : x \in (0, 2)$
$f_1 : x \in (2, 5)$
$f_2 : x \in (5, 8)$

(b) $f_0 : u_0 \in (0, 1)$
$f_1 : u_1 \in (0, 1)$
$f_2 : u_2 \in (0, 1)$

Figura 2.1: Esempio spline

2.2.1 Curve nello spazio

Il principio finora illustrato può essere applicato a generiche curve in 2 e 3D. L'idea di fondo è quella di avere tre funzioni parametrizzate rispetto lo stesso parametro ed ognuna delle quali identifica rispettivamente la coordinata spaziale x, y e z. In questo modo si possono sfruttare arbitrarie funzioni per ciascuna delle tre coordinate e, considerarle contemporaneamente per individuare un andamento nello spazio.

Nel nostro caso particolare si tratta di scegliere dei generici polinomi cubici

$$x_i = a_{x_i} + b_{x_i} u_i + c_{x_i} u_i^2 + d_{x_i} u_i^3$$

$$y_i = a_{y_i} + b_{y_i} u_i + c_{y_i} u_i^2 + d_{y_i} u_i^3$$

$$z_i = a_{z_i} + b_{z_i} u_i + c_{z_i} u_i^2 + d_{z_i} u_i^3$$

che per ogni tratto di curva esprimano l'andamento delle tre coordinate spaziali. I coefficienti dei polinomi si possono ricavare risolvendo un sistema matriciale simile a quello visto nella sezione precedente ottenuto imponendo gli opportuni vincoli visti.

2.3 Metodi di interpolazione

I metodi esposti rappresentano il classico metodo per costruire *splines naturali cubiche*. L'algoritmo è utile in teoria perché rappresenta le basi per la comprensione del metodo, ma in pratica non è molto efficiente dal momento che la dimensione delle matrici da risolvere aumentano all'aumentare dei punti di controllo e risultano, anche per pochi punti, di dimensione relativamente grande. Inoltre la forma della curva risulta molto dipendente dal valore dei punti di controllo ma, più di questo, una qualunque variazione nei punti causa necessariamente il ricalcolo di tutto il sistema: si ha in questo senso poco controllo locale sull'andamento dei punti.
Esistono metodi che sfruttano lo stesso principio visto ma in maniera più efficiente tra cui ricordiamo: *Hermite Splines*, *Catmull-Rom Splines* e *Cardinal Splines*. In particolare nel corso del progetto abbiamo sfruttato il metodo basato sulle *Catmull-Rom Splines* dal momento che permette di risolvere un sistema di dimensione fissa per ogni coppia di punti di controllo.

Catmul-Rom Splines

L'algoritmo di *Catmul-Rom* è di principio identico a quello di *Hermite* con la sola differenza nei vincoli da imporre a ciascun tratto di curva. Infatti per ciascuna coppia di punti p_i, p_{i+1} *Hermite* impone il passaggio per i punti e prevede che sia l'utente a decidere il valore della pendenza in ogni punto di raccordo tra tratti di curva successivi. Questo significa che oltre ad n punti di controllo devono anche essere passati in ingresso n valori di derivata prima.
Catmul-Rom invece prevede solo l'utilizzo dei punti di controllo e di due punti aggiuntivi, uno posizionato prima di p_0, ovvero prima dell'inizio della curva, ed uno dopo p_n, ovvero dopo la fine della curva.
Vediamo più in dettaglio il metodo di Catmul-Rom.

Esso, oltre ad imporre il passaggio per i punti di controllo allo stesso modo di come è stato descritto nella sezione 2.2, ne fissa le derivate prime, garantendo implicitamente la continuità C^1. In particolare, le pendenze Δx_i, Δy_i, Δz_i e Δx_{i+1}, Δy_{i+1}, Δz_{i+1} relative ai punti p_i e p_{i+1} vengono calcolate come:

vettori delle derivate prime in ogni punto

$$t_i = \tfrac{1}{2}(p_{i+1} - p_{i-1}) = [\Delta x_i \ \Delta y_i \ \Delta z_i]'$$

$$t_{i+1} = \tfrac{1}{2}(p_{i+2} - p_i) = [\Delta x_{i+1} \ \Delta y_{i+1} \ \Delta z_{i+1}]'$$

mentre le derivate rispetto il parametro (dalla derivazione della 2.1) sono pari:

$$\frac{\partial u_i}{\partial x} = \frac{1}{x_{i+1} - x_i}$$

$$\frac{\partial u_i}{\partial y} = \frac{1}{y_{i+1} - y_i}$$

$$\frac{\partial u_i}{\partial z} = \frac{1}{z_{i+1} - z_i}$$

Come si vede dalle relazioni riportate sopra, sono necessari anche p_{i-1} e p_{i+2} ed è proprio da questa esigenza che sono stati creati anche i due punti aggiuntivi prima di p_0 e dopo p_n in modo da riuscire a calcolare i vettori tangenti anche in tali punti.

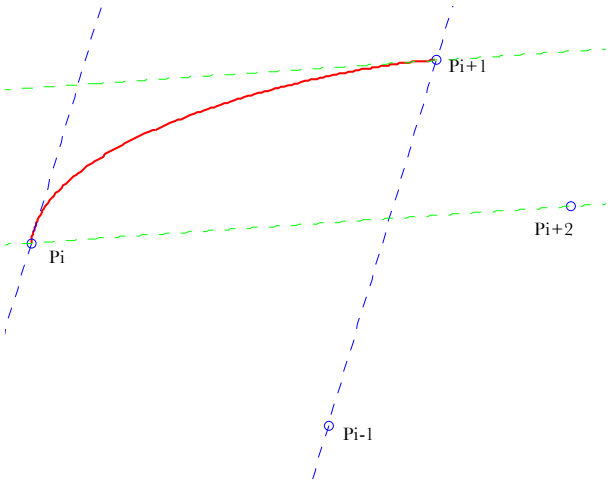


Figura 2.2: Metodo Catmul-Rom

Dalla costruzione di tali vincoli si nota come nell'algoritmo scelto siano stati in definitiva imposti solo i vincoli di continuità C^0 e C^1, i quali garantiscono un corretto raccordo nella pendenza dei tratti e quindi sono considerati prioritari rispetto alla continuità C^2 che viene esclusa.

A questo punto è possibile costruire i vettori delle tangenti in forma parametrica

sfruttando la (2.2):

$$Dp_i = [\Delta x_i \frac{\partial u_i}{\partial x} \ \Delta y_i \frac{\partial u_i}{\partial y} \ \Delta z_i \frac{\partial u_i}{\partial z}]'$$

$$Dp_{i+1} = [\Delta x_{i+1} \frac{\partial u_i}{\partial x} \ \Delta y_{i+1} \frac{\partial u_i}{\partial y} \ \Delta z_{i+1} \frac{\partial u_i}{\partial z}]'$$

Riscrivendo il tutto in forma matriciale si giunge ad un sistema simile a quello delle sezioni precedenti.

Un semplice esempio può essere utile a chiarire il procedimento: si considerino i punti $p_{i-1} = (1, -1)$; $p_i = (1, 1)$; $p_{i+1} = (4, 3)$; $p_{i+2} = (5, -2)$

Da semplici conti risulta

$$\frac{du_i}{dx} = \frac{1}{3}; \ \frac{du_i}{dy} = \frac{1}{2}$$

$$t_i = \frac{1}{2}(p_{i+1} - p_{i-1}) = \begin{bmatrix} \frac{3}{2} \\ 2 \end{bmatrix} = \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix}$$

$$t_{i+1} = \frac{1}{2}(p_{i+2} - p_i) = \begin{bmatrix} 2 \\ -\frac{3}{2} \end{bmatrix} = \begin{bmatrix} \Delta x_{i+1} \\ \Delta y_{i+1} \end{bmatrix}$$

da cui

$$Dp_i = [\Delta x_i \frac{\partial u_i}{\partial x} \ \Delta y_i \frac{\partial u_i}{\partial y}]' = \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$$

$$Dp_{i+1} = [\Delta x_{i+1} \frac{\partial u_i}{\partial x} \ \Delta y_{i+1} \frac{\partial u_i}{\partial y}]' = \begin{bmatrix} \frac{2}{3} \\ -\frac{3}{4} \end{bmatrix}$$

e quindi

---

passaggio per p_i **in** $u_i = 0$	$x(0) = a_x = 1$
	$y(0) = a_y = 1$
passaggio per p_{i+1} **in** $u_i = 1$	$x(1) = a_x + b_x + c_x + d_x = 4$
	$y(1) = a_y + b_y + c_y + d_y = 3$
pendenza in p_i	$x'(0) = b_x = \frac{1}{2}$
	$y'(0) = b_y = 1$
pendenza in p_{i+1}	$x'(1) = b_x + 2c_x + 3d_x = \frac{2}{3}$
	$y'(1) = b_y + 2c_y + 3d_y = -\frac{3}{4}$

ed in forma matriciale

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 4 & 3 \\ \frac{1}{2} & 1 \\ \frac{2}{3} & -\frac{3}{4} \end{bmatrix}$$

E' interessante porre l'attenzione sul fatto che, scegliendo la parametrizzazione considerata, per ogni tratto di curva la matrice del sistema lineare rimane sempre la stessa ed è quindi fissa: basta calcolare una sola volta l'inversa.
Inoltre il metodo è molto flessibile per la definizione dei punti di controllo: la modifica nel valore di uno di essi si ripercuote solo nei tratti di curva adiacenti ad esso ma non sugli altri. Non è quindi necessario ricalcolare tutta la spline.

2.4 Algoritmo implementato

La nostra implementazione è del tutto simile a quanto descritto finora. In sostanza abbiamo utilizzato il metodo di Catmul-Rom con la parametrizzazione proposta con 2 differenze:

- consideriamo sempre punti 3D (negli esempi abbiamo mostrato solo casi 1 e 2D);
- cambiamo la parametrizzazione per riportarci in ascissa curvilinea.

L'estensione a 3D è assolutamente intuitiva come riportato nella sezione 2.2.1. Si riporta invece la descrizione della parametrizzazione che è stata utilizzata.

Parametrizzazione in ascissa curvilinea

Da quanto visto finora ogni tratto risulta parametrizzato secondo un parametro $u_i \in [0, 1]$. Nel nostro caso sarebbe molto più comodo avere a disposizione un parametro che evolve linearmente lungo la curva in modo da identificarne univocamente ogni punto e, contemporaneamente, ricondurre il problema al caso 1D. Risulta particolarmente utile poter accedere ai punti 3D della curva semplicemente conoscendo la distanza percorsa lungo la stessa. Un parametro che garantisce entrambi gli obbiettivi è proprio l'ascissa curvilinea. Dato che matematicamente risulta abbastanza complicato eseguire un esplicito cambiamento di variabile per riportarsi in ascissa curvilinea s, facciamo in modo che la curva sia accessibile dall'esterno come se lo fosse, mantenendo in realtà la parametrizzazione individuata in precedenza.
Più precisamente, sfruttando MATLAB (o analoghe funzioni C++ per l'integrazione numerica), si possono calcolare le lunghezze di ogni tratto di curva compreso tra coppie di punti e quindi la lunghezza parziale della curva fino ad ogni punto di controllo. Il passaggio in ascissa curvilinea risulta immediato: grazie alle varie lunghezze valutate "offline" è possibile capire a quale tratto i-esimo di curva appartiene il valore desiderato $\overline{s}$ di ascissa e trovare il corrispondente valore del parametro u_i tramite la:

$$u_i = \frac{\overline{s} - L_i}{L_{i+1} - L_i}$$

dove L_i è la lunghezza della curva fino al punto di controllo i-esimo.
Come accennato sopra, questo procedimento mette in luce come la curva non sia realmente parametrizzata in ascissa curvilinea, ma tramite qualche confronto e traslazione è possibile accedere ai suoi punti come se la curva fosse effettivamente parametrizzata in s. Per effettuare queste operazioni di creazione e accesso alla curva in ascissa curvilinea automaticamente, sono state create due apposite funzioni MATLAB poi convertite in C++.
La funzione di creazione della spline (chiamata `splineinterpolante`) determina tutti i coefficienti dei polinomi cubici dei vari tratti. Essi definiscono completamente l'equazione della spline ed è quindi possibile ricavare le coordinate di un qualsiasi punto 3D della curva corrispondente ad un certo valore di ascissa curvilinea. In particolare la funzione `valutazionespline(s)` accetta come parametro di ingresso un particolare valore s e restituisce le coordinate del corrispondente punto 3D rispetto al sistema di riferimento globale (vedere capitolo 4).

2.5 La nostra spline

Si riportano nella figure seguenti rispettivamente il grafico 3D in MATLAB della spline che è stata utilizzata (figura 2.3) e le foto della sua ricostruzione (tramite il nastro segnaletico) nel Navlab in figura (2.4).

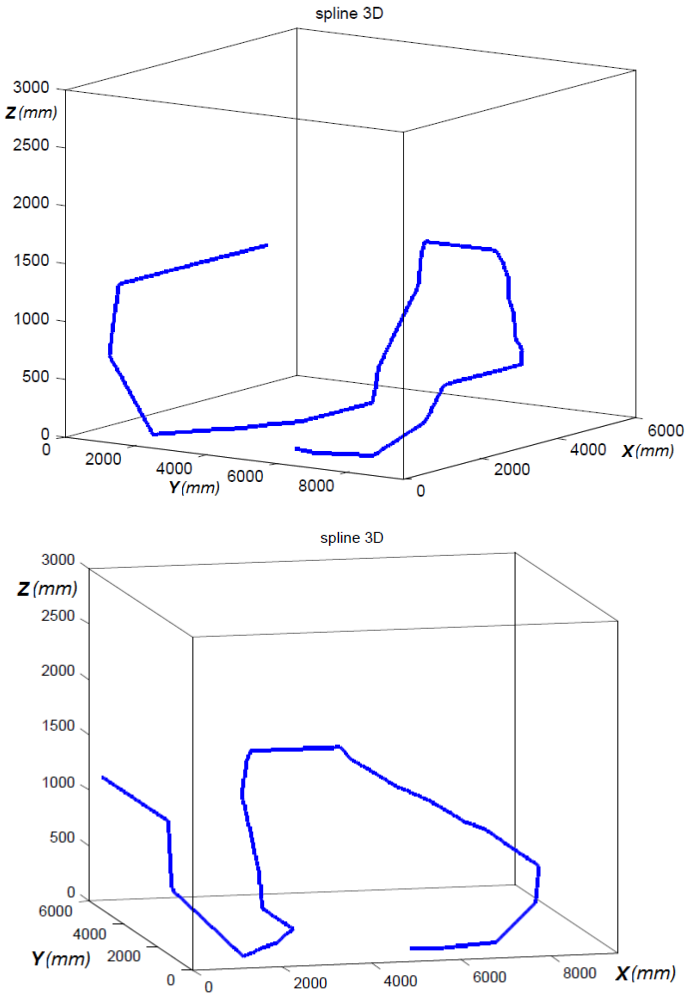


Figura 2.3: Grafico 3D della spline utilizzata

Figura 2.4: Ricostruzione spline nel Navlab

Capitolo 3

Calibrazione videocamere

Prima di spiegare come è stata svolta la calibrazione è opportuno descrivere il modello della videocamera che è stato utilizzato.

3.1 Modello telecamera

E' stato utilizzato il modello *pinhole* nel quale si suppone che i raggi di luce entrino nella videocamera da un piccolo foro di dimensioni infinitesime: in questo modo ogni punto della scena P_w viene proiettato nel piano immagine $\mathcal{R}$ (vedere figura 3.1) tramite un *unico* raggio che da luogo ad un unico punto P_{im}. Con

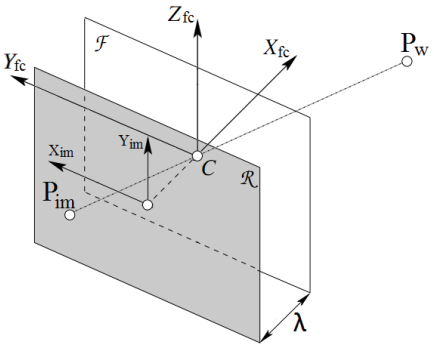


Figura 3.1: Modello pinhole

questa assunzione, l'immagine che si forma nel piano $\mathcal{R}$ risulta sempre a fuoco e con una dimensione che dipende unicamente dal parametro λ, detto *distanza focale*.
λ rappresenta la distanza tra il *piano immagine* $\mathcal{R}$ (che è in sostanza il sensore CCD all'interno della telecamera che acquisisce l'immagine) e il piano $\mathcal{F}$ contenente "il foro da cui passa la luce".
Dalla figura (3.1) si può osservare che il modello pinhole è costituito da un piano immagine $\mathcal{R}$ e da un punto C, distante λ da $\mathcal{R}$, detto *centro ottico*. Il piano $\mathcal{F}$ parallelo a $\mathcal{R}$ e contenente C è detto *piano focale*. La retta passante per C e ortogonale a $\mathcal{R}$ è detto *asse ottico* e il punto in cui esso interseca il piano $\mathcal{R}$ è

detto *punto principale*. Tutta questa terminologia introdotta farà comodo nelle prossime sezioni e nel capitolo 4.
Si vogliono ricavare a questo punto le relazioni tra le coordinate del punto P_w dello spazio e quelle della sua proiezione P_{im} nel piano immagine. Scegliendo i sistemi di riferimento orientati come in figura (3.1) si ottengono equazioni piuttosto semplici.
Consideriamo la vista laterale della situazione in figura (3.1): Dalla figura (3.2),

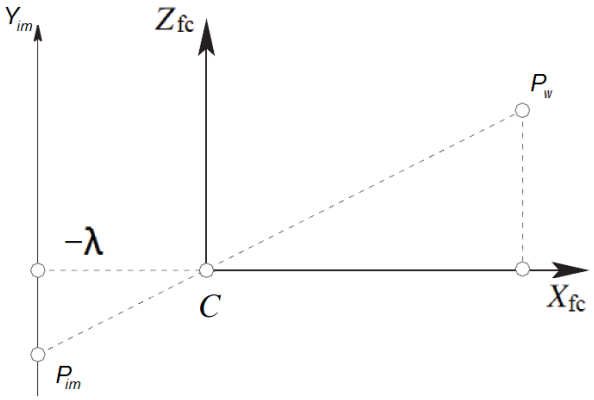


Figura 3.2: Vista laterale modello pinhole

sfruttando la similitudine dei due triangoli, si può ricavare la relazione:

$$\frac{-\lambda}{x_{fc}} = \frac{y_{im}}{z_{fc}}$$

Analogamente, considerando la vista dall'alto della figura (3.1),ovvero quella proiettata sul piano individuato da x_{fc} e y_{fc}, si ottiene:

$$\frac{-\lambda}{x_{fc}} = \frac{x_{im}}{y_{fc}}$$

Si possono dunque ricavare x_{im} e y_{im}:

$$x_{im} = -\lambda \frac{y_{fc}}{x_{fc}} \tag{3.1}$$

$$y_{im} = -\lambda \frac{z_{fc}}{x_{fc}} \tag{3.2}$$

3.2 Calibrazione

Prima di poter utilizzare le telecamere per i nostri scopi, è necessario effettuare una procedura preliminare di fondamentale importanza detta *calibrazione*. Tale procedura serve per individuare i parametri descrittivi della telecamera e per permetterci di sfruttare effettivamente il modello *pinhole* descritto nella sezione precedente.

3.2.1 Definizione

Calibrare la telecamera significa stimarne i parametri caratteristici ovvero i cosiddetti parametri *intrinseci* ed *estrinseci*. Questa procedura è necessaria dato che per il passaggio da coordinate espresse nel sistema di riferimento della telecamera a quelle nel piano immagine è necessaria la conoscenza della *lunghezza focale* λ (vedere relazioni 3.1 e 3.2).
Inoltre è importante tenere presente che il punto principale non si trova necessariamente al centro dell'immagine: questo è dovuto al fatto che il posizionamento del sensore CCD è effettuato con una certa tolleranza ed è quindi difficile che si trovi perfettamente centrato sull'asse ottico.
Questo aspetto verrà ripreso nel capitolo 4 relativo alla cinematica. Per ora ci limitiamo nel descrivere i parametri intrinseci ed estrinseci e come vengono ricavati utilizzando il toolbox [7].

3.2.2 Parametri intrinseci

I parametri intrinseci sono quelli che forniscono informazioni riguardanti la geometria della telecamera e la distorsione introdotta dalle sue lenti. Dal momento che si è deciso di utilizzare un modello pinhole per la telecamera, gli unici parametri intrinseci da considerare sono la distanza focale e la posizione del punto principale.
Tuttavia, il modello che viene utilizzato per la calibrazione effettuata con [7] è più accurato del pinhole e fornisce la stima dei seguenti parametri:

- lunghezza focale (vettore appartenente a $\mathbb{R}^{2\times1}$)
- coordinate punto principale (espresse in pixel rispetto a un sistema di riferimento centrato nell'angolo in alto a sinistra dell'immagine)
- coefficiente di skew
- vettore $K_c \in \mathbb{R}^{5\times1}$ relativo alla distorsione

La stima della lunghezza focale è restituita dal toolbox come vettore di due componenti espresse in pixel del tipo $f = [\lambda \cdot s_x \ \lambda \cdot s_y]$, dove λ è la lunghezza focale in mm ed s_x e s_y rappresentano il numero di pixel per millimetro rispettivamente lungo la direzione x e y dell'immagine, tenendo conto del fatto che in generale i pixel sono rettangolari e non quadrati. Il coefficiente di skew fornisce l'angolo tra gli assi x e y del sensore CCD (che nel modello per la telecamera da noi utilizzato è assunto pari a 90°). Il vettore K_c fornisce le informazioni relative alla distorsione radiale (dovuta alla forma delle lenti) e tangenziale (dovuta alle tolleranze nel montaggio delle lenti). Questi ultimi due parametri non sono stati considerati, dal momento che non se ne tiene conto nel modello da noi scelto.

E' interessante osservare che le telecamere PTZ consentono di impostare un determinato livello di zoom modificando la posizione delle lenti in modo da variare la distanza focale (dalle relazioni 3.1 e 3.2 si capisce che la dimensione degli oggetti nel piano immagine dipende dalla distanza focale). E' chiaro quindi che i parametri intrinseci variano al variare dello zoom.

3.2.3 Parametri estrinseci

Questi parametri individuano la posizione del sistema di riferimento di un qualsiasi oggetto rispetto il sistema di riferimento della telecamera e sono:

- il *vettore di traslazione* dal centro del sistema di riferimento della camera al centro di quello dell'oggetto;
- la *matrice di rotazione* relativa tra i due sistemi di riferimento.

Questi due parametri consentono di ricavare le coordinate di un punto espresso rispetto a un sistema di riferimento di un oggetto generico rispetto a quello della telecamera (vedere figura 3.3).

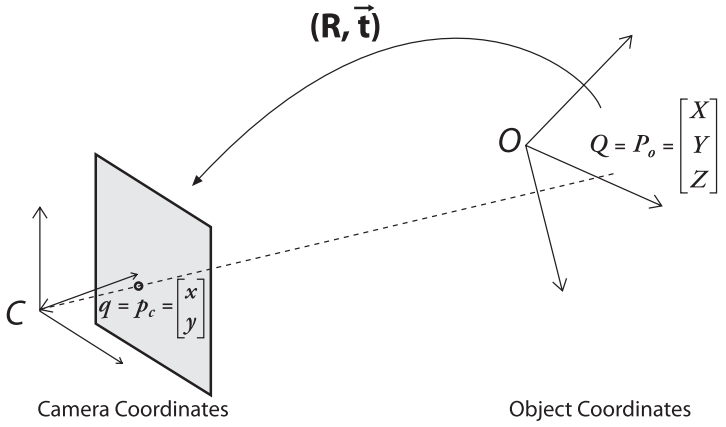


Figura 3.3: Conversione coordinate da sistema oggetto a sistema telecamera

3.2.4 Procedura di calibrazione

Per ricavare i parametri intrinseci ed estrinseci è stato sfruttato quindi il toolbox MATLAB sviluppato da Jean-Yves Bouguet di cui si riporta solo una traccia e si rimanda a [7] per i dettagli.
Il metodo di calibrazione consiste nell'acquisire dalla telecamera un set di foto di una scacchiera composta da quadrati di dimensioni note. La scacchiera deve essere posta in diverse orientazioni: le due figure seguenti rappresentano

rispettivamente la procedura in generale e un esempio delle immagini da noi acquisite.

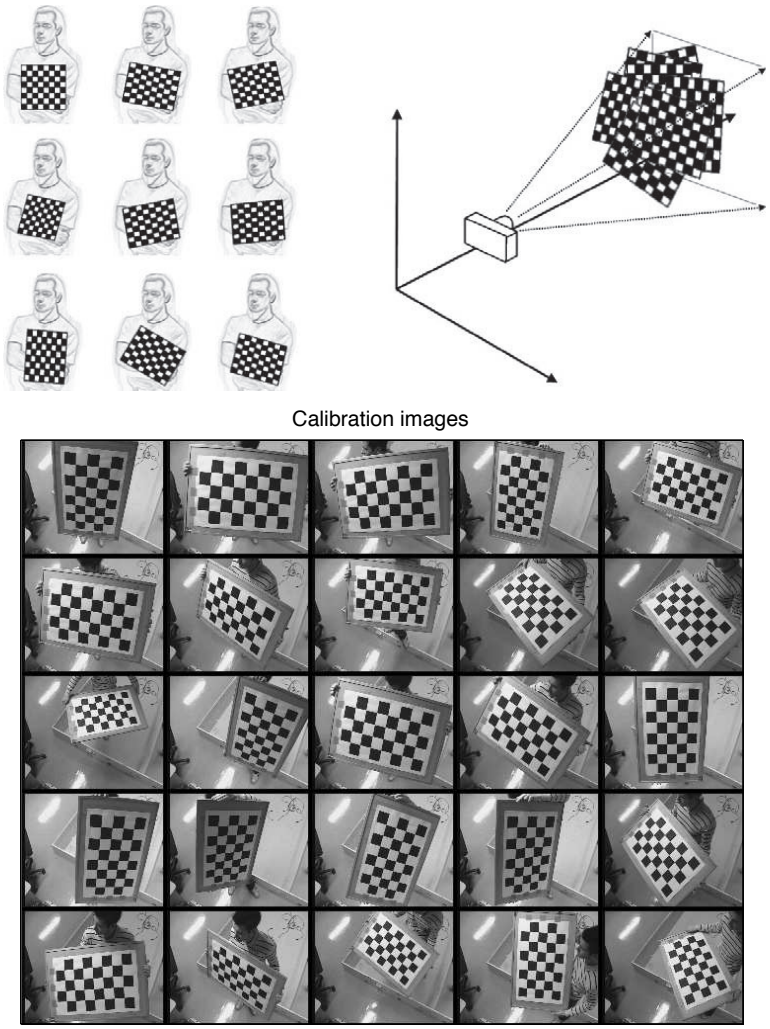


Figura 3.4: Calibrazione

Da questo set di immagini, conoscendo le dimensioni della scacchiera l'algoritmo in [7] è in grado di ricavare i parametri intrinseci descritti nelle sezioni precedenti.
A questo punto una volta "calibrati" gli intrinseci, è possibile ricavare i parametri estrinseci relativi a una determinata configurazione della videocamera PTZ "scattando" un'unica foto alla scacchiera. Più precisamente, si pone la scacchiera in una determinata posizione e si variano gli angoli di pan e tilt in modo che la telecamera possa inquadrarla ed acquisire una foto. Da questa foto il toolbox consente di ottenere la matrice di rotazione e il vettore di traslazione tra il sistema di riferimento della videocamera con quei particolari angoli di pan e tilt e la scacchiera in quella posizione. Si riporta in figura (3.5) un esempio di foto che può essere utilizzata per ricavare i parametri estrinseci.

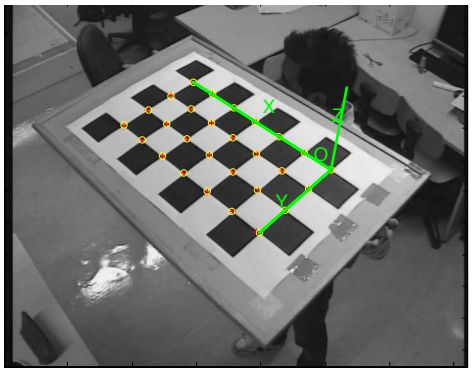


Figura 3.5: Sistema di riferimento della scacchiera

Dato che si utilizza un modello pinhole per la videocamera, è necessario però compensare la distorsione presente nella particolare immagine che si utilizza per ricavare i relativi parametri estrinseci. La compensazione avviene proprio sfruttando i parametri intrinseci precedentemente stimati attraverso un apposito comando messo a disposizione nel tool (indicato con Undistort image). Una volta ottenuta l'immagine senza distorsione si utilizza il comando del toolbox per il calcolo dei parametri estrinseci relativi alla particolare immagine (indicato con Compute Extrinsic).

Riepilogo passi calibrazione

E' utile a questo punto esporre sinteticamente i passi da compiere per una corretta calibrazione nell'ipotesi di modello pinhole.

1. procurarsi una scacchiera e misurare accuratamente le dimensioni dei quadrati (che verranno poi passate come parametri di ingresso al toolbox)
2. raccogliere un set di immagini di tale scacchiera in diverse posizioni (vedi figura 3.5);
3. tramite il toolbox di Bouguet [7] ricavare la stima dei parametri intrinseci;
4. raccogliere un secondo set di immagini della scacchiera nelle posizioni in cui si vogliono ricavare i relativi parametri estrinseci (vedere anche capitolo 4);
5. eliminare la distorsione da tale raccolta per riportarsi all'ipotesi di modello *pinhole*;
6. da tale set calcolare, per ciascuna delle immagini, i parametri estrinseci corrispondenti.

Capitolo 4

Cinematica

Dal momento che la posizione delle videocamere Ulisse può essere controllata solo cambiando il valore degli angoli di pan (θ in figura (4.1)) e tilt (ψ in figura (4.1)), è necessario creare un algoritmo che consenta di muovere la videocamera di modo che riesca ad inquadrare un punto arbitrario dello spazio.

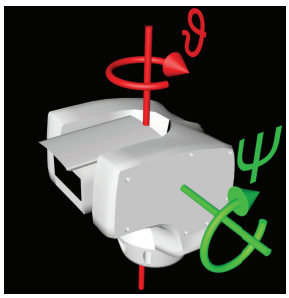


Figura 4.1: Angoli di pan e tilt

Solo in questo modo è possibile seguire una traiettoria arbitraria definita per punti nello spazio.
In particolare è necessario stabilire un sistema di riferimento globale rispetto al quale vengono espresse le coordinate dei punti 3D. Successivamente, scelto un opportuno modello per la videocamera, si devono ricavare delle equazioni in forma chiusa che forniscano gli opportuni angoli di pan e tilt per portare la videocamera ad inquadrare il punto di coordinate x, y e z.

4.1 Sistemi di riferimento e modello videocamera

E' stata seguita la tecnica proposta in [4] applicando qualche piccola modifica che verrà descritta in questo capitolo.
Le prove degli algoritmi implementati sono state svolte nel laboratorio Navlab del nostro dipartimento. Il laboratorio è provvisto di 3 videocamere Ulisse e per questo motivo l'algoritmo è stato sviluppato per 3 videocamere, ma può essere esteso a un numero N arbitrario di videocamere.
Nelle figura successiva (4.2) si riporta una ricostruzione schematica del laboratorio.

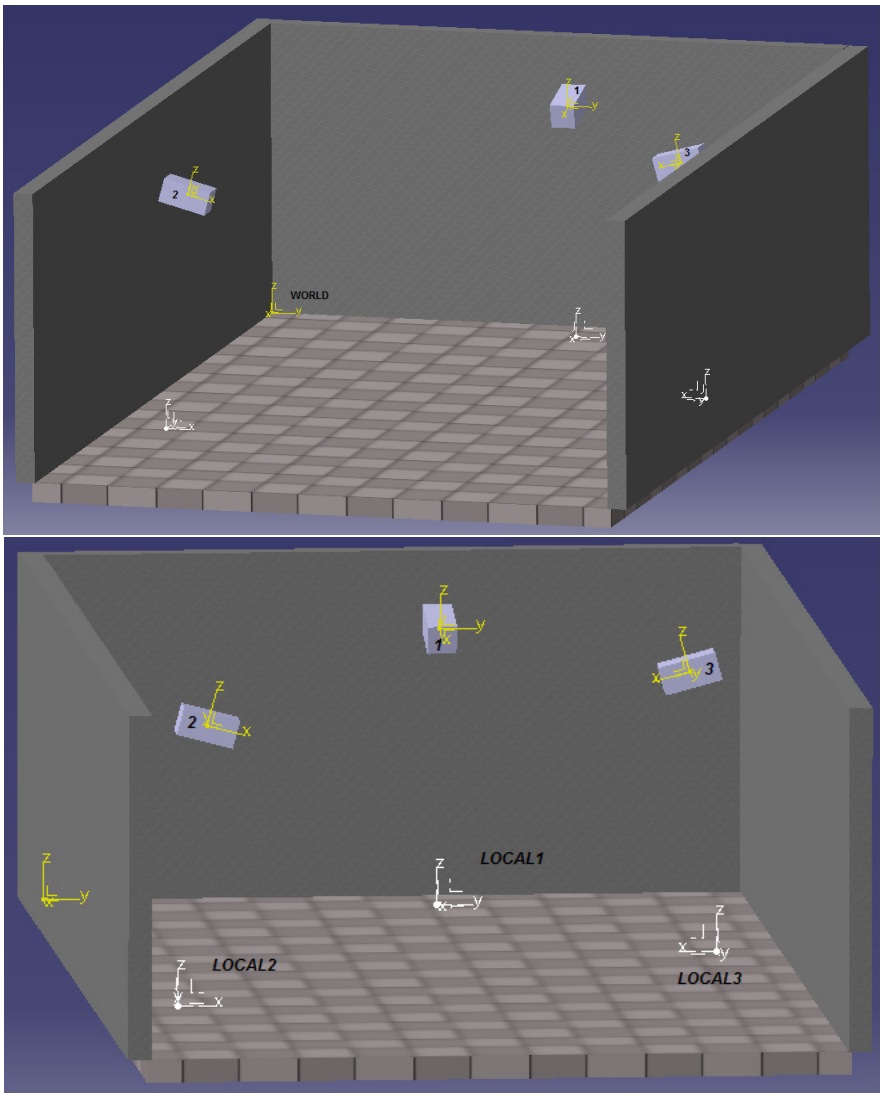


Figura 4.2: Schema 3D del Navlab

Come si può notare dalla figura (4.2), le videocamere sono "staccate" dal muro: questo è per tener conto del fatto che sono fissate ad esso per mezzo di un supporto che le mantiene a una certa distanza (vedi figura 1.1). In questo modo le videocamere sono in grado di effettuare una rotazione di 360° attorno all'asse di pan.
In figura (4.2) sono indicati vari sistemi di riferimento:

- il sistema di riferimento globale (in giallo) indicato con WORLD;
- i sistemi di riferimento nel centro ottico di ogni videocamera (in giallo);
- i sistemi di riferimento locali (in bianco) indicati con i simboli LOCAL1, LOCAL2 e LOCAL3 centrati rispettivamente nei punti in cui gli assi di rotazione di pan delle videocamere 1, 2 e 3 intersecano il pavimento;

Si può sottolineare da subito che i sistemi di riferimento nei centri ottici delle varie videocamere sono stati fissati in modo conforme a quelli in [4] e cioè con l'asse x posto sull'asse ottico della videocamera e l'asse z verso l'alto.
In questo modo le coordinate di un punto rispetto al sistema locale di ogni videocamera (LOCAL) si ottengono dalle coordinate (dello stesso punto) espresse rispetto al sistema di riferimento nel centro ottico della videocamera con le seguenti trasformazioni ([4]):

$$\begin{bmatrix} x_{wloc} \\ y_{wloc} \\ z_{wloc} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ H \end{bmatrix} + R_\theta \left(\begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} + R_\psi \left(\begin{bmatrix} x_{off} \\ y_{off} \\ z_{off} \end{bmatrix} + \begin{bmatrix} x_{oc} \\ y_{oc} \\ z_{oc} \end{bmatrix} \right) \right) \tag{4.1}$$

dove con R_θ e R_ψ si sono indicate rispettivamente le matrici di rotazione attorno all'asse z e all'asse y relative agli angoli θ e ψ. Le trasformazioni indicate dalla formula (4.1) sono rappresentate nella figura (4.3):

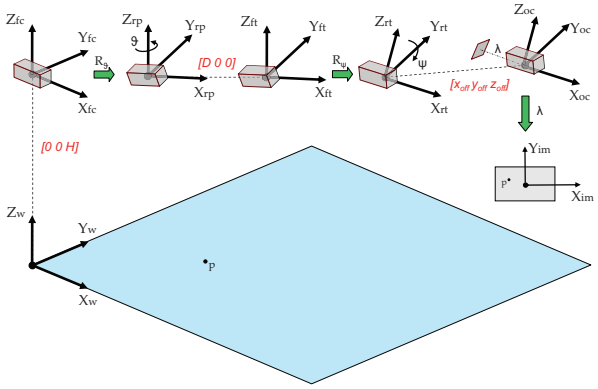


Figura 4.3: Trasformazioni camera sistema locale

Per riportare le coordinate dei sistemi locali delle videocamere rispetto al sistema globale scelto e viceversa basta applicare delle semplici traslazioni (vedere figura (4.2)) dopo aver misurato la posizione dei centri dei vari sistemi LOCAL rispetto al sistema WORLD.

4.2 Parametri del modello meccanico della videocamera

Per definire il modello di ciascuna videocamera è necessario ricavare i parametri H (altezza videocamera considerata rispetto l'asse di rotazione di tilt, vedere figura (4.3), D (distanza tra l'asse di rotazione di pan e quello di tilt), x_{off}, y_{off}, z_{off} (coordinate del centro ottico rispetto al sistema di riferimento centrato nell'asse di tilt, vedere figura 4.1).
Per trovare questi parametri è stato seguito l'approccio di [4]. E' opportuno riportare il procedimento completo dato che in [4] è presente solo una traccia e non è stato facile applicarla al nostro caso.
Dopo aver effettuato la calibrazione di tutte e 3 le videocamere si procede come segue:

1. si pone la scacchiera in diverse posizioni della stanza (noi ne abbiamo effettuate 20 per ogni videocamera)

2. per ogni posizionamento, si misurano le coordinate $[x_w\ y_w\ z_w]$ del centro del sistema di riferimento della scacchiera (che verrà indicato con P_{scacch}) rispetto al sistema globale scelto (WORLD nel nostro caso). Il centro del sistema scacchiera è il primo dei quattro punti che si selezionano per delimitare l'area utile della scacchiera durante la procedura di valutazione dei parametri estrinseci (vedere [7] per dettagli).

3. per ogni posizionamento, si acquisiscono gli angoli di pan e tilt della videocamera quando essa inquadra al centro dell'immagine il centro del sistema scacchiera del punto precedente. In questo modo si limitano gli effetti della distorsione che è maggiore ai bordi dell'immagine.

4. per ogni posizionamento si ricavano i parametri estrinseci della videocamera in questione (vedere sezione 3.2.4) e cioè la matrice di rotazione R e il vettore di traslazione T_c. Il vettore T_c contiene le coordinate del punto P_{scacch} espresse rispetto al sistema di riferimento nel centro ottico con le convenzioni del toolbox di Bouguet [7]. Per ricavare quindi le coordinate di P_{scacch} espresse rispetto al sistema nel centro ottico secondo le nostre convenzioni è necessario moltiplicare il vettore T_c per la matrice $R_{BougtoOur} = R_z(-90)R_x(-90)$, dato che il sistema di riferimento nel centro ottico considerato nel toolbox è ruotato rispetto a quello da noi utilizzato (ha l'asse z sull'asse ottico). Tali coordinate verrano indicate con il vettore $[x_{oc}\ y_{oc}\ z_{oc}]$

5. Ora si conoscono le coordinate del punto P_{scacch} rispetto a due sistemi di riferimento: quello globale e quello nel centro ottico, indicate rispettivamente con $[x_w\ y_w\ z_w]$ e $[x_{oc}\ y_{oc}\ z_{oc}]$. Da queste informazioni per ogni posizionamento si può considerare la relazione (ricavata riscrivendo in modo

diverso 4.1):

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} - R_\theta R_\psi \begin{bmatrix} x_{oc} \\ y_{oc} \\ z_{oc} \end{bmatrix} = \begin{bmatrix} 0 & R_\theta(1,1) & \\ 0 & R_\theta(2,1) & R_\theta R_\psi \\ 1 & R_\theta(3,1) & \end{bmatrix} \begin{bmatrix} H \\ D \\ x_{off} \\ y_{off} \\ z_{off} \end{bmatrix} \quad (4.2)$$

dove le matrici di rotazione sono state calcolate in base agli angoli di pan e tilt misurati al punto 3. La (4.2) può essere vista come un sistema del tipo $y(i) = X(i)\beta$ dove si è inserito l'indice i che indica la misura che si sta considerando (l'indice varia tra 1 e il numero di misure N). Giustapponendo le varie relazioni del tipo (4.2) per ogni misura, si ottiene il sistema complessivo $y = X\beta$ a cui si può applicare la formula dei minimi quadrati $\beta = (X^T X)^{-1} X^T y$ per ricavare una stima dei parametri desiderati.

Questo algoritmo è stato realizzato solo in MATLAB (e non in C++) dal momento che anche il toolbox utilizzato è implementato in MATLAB e i calcoli vengono effettuati una sola volta offline.

4.3 Cinematica diretta

E' stato implementato seguendo [4] un algoritmo che svolge la cinematica diretta: note le coordinate di un punto in pixel nel piano immagine (si veda il capitolo 3 per la terminologia e il modello della telecamera), l'algoritmo restituisce le coordinate corrispondenti rispetto al sistema WORLD.
Dato che si deve operare con coordinate nello spazio e nel piano immagine espresse rispetto alla stessa unità di misura, è necessario convertire le coordinate da pixel a mm. A questo scopo sono state utilizzate le relazioni $x_{im} = \frac{x_{pix}}{s_x}$ e $y_{im} = \frac{y_{pix}}{s_y}$ dove x_{pix} e y_{pix} sono le coordinate in pixel e s_x e s_y il numero di pixel per millimetro rispettivamente lungo la direzione x e y dell'immagine, ricavati dalle dimensioni orizzontali e verticali del singolo pixel del CCD indicate nel datasheet della SONY.
In generale per individuare un punto nello spazio sono necessarie 2 videocamere che lo inquadrano simultaneamente: in questo caso si assume invece che il punto sia sul pavimento (z=0 nel sistema WORLD) in modo da poter ricavare le sue coordinate anche solo con una videocamera che lo inquadra.
Si riporta per completezza il procedimento seguito in [4] con qualche dettaglio in più.
Le coordinate del centro ottico rispetto al sistema LOCAL (vedere figura (4.2)) si ricavano utilizzando la (4.1) applicata a $[x_{oc}\, y_{oc}\, z_{oc}] = [0\,0\,0]$, dato che il centro ottico è anche il centro del sistema di riferimento della videocamera e cioè:

$$\begin{bmatrix} x_{ocw} \\ y_{ocw} \\ z_{ocw} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ H \end{bmatrix} + R_\theta \left(\begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} + R_\psi \left(\begin{bmatrix} x_{off} \\ y_{off} \\ z_{off} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) \right) \quad (4.3)$$

A questo punto, dato che le coordinate nel piano immagine sono legate a quelle del sistema di riferimento nel centro ottico tramite le seguenti relazioni (sono le relazioni 3.1 e 3.2 dove x_{im} ha un segno diverso dato che è orientato in maniera opposta a y_{oc}, confrontare figura 3.1 e 4.3):

$$x_{im} = \lambda \frac{y_{oc}}{x_{oc}}$$

$$y_{im} = -\lambda \frac{z_{oc}}{x_{oc}}$$

in cui si è utilizzata la stessa notazione riportata in figura (4.3) (dove λ è la distanza focale), si possono ricavare le coordinate del punto in questione rispetto al sistema di riferimento della videocamera centrato nel centro ottico. In particolare, imponendo $x_{oc} = -\lambda$ (vedere figura (4.3) per convincersi del motivo) si ottiene il vettore $[x_{oc} \;\; y_{oc} \;\; z_{oc}] = [-\lambda \;\; -x_{im} \;\; y_{im}]$ che rappresenta le coordinate del punto di cui si sta svolgendo la cinematica diretta espresse nel sistema di riferimento della videocamera. Queste coordinate possono essere espresse nel sistema LOC della videocamera tramite la solita trasformazione (4.1):

$$\begin{bmatrix} x_{imw} \\ y_{imw} \\ z_{imw} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ H \end{bmatrix} + R_\theta \left(\begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} + R_\psi \left(\begin{bmatrix} x_{off} \\ y_{off} \\ z_{off} \end{bmatrix} + \begin{bmatrix} -\lambda \\ -x_{im} \\ y_{im} \end{bmatrix} \right) \right) \tag{4.4}$$

Ora si può ricavare l'equazione della retta (espressa nel sistema LOCAL in forma parametrica nel parametro $t \in \mathbb{R}$) che passa per i due punti ricavati nelle relazioni (4.3) e (4.4) come indica la figura (4.4):

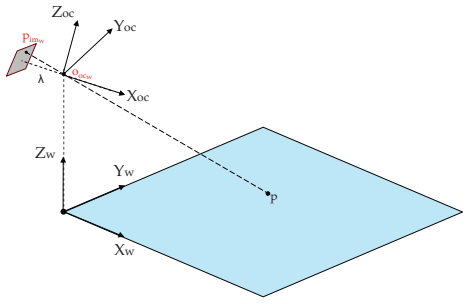


Figura 4.4: Cinematica diretta

$$\begin{cases} x_{wLOC}(t) = x_{imw} + (x_{ocw} - x_{imw})t \\ y_{wLOC}(t) = y_{imw} + (y_{ocw} - y_{imw})t \\ z_{wLOC}(t) = z_{imw} + (z_{ocw} - z_{imw})t \end{cases}$$

Imponendo poi $z_{wLOC}(t) = 0$, ricavando dalla terza equazione il valore di t e sostituendolo nella prima e la seconda si ottiene:

$$\begin{cases} x_{wLOC}(t) = \frac{-z_{imw}(x_{ocw}-x_{imw})}{z_{ocw}-z_{imw}} + x_{imw} \\ y_{wLOC}(t) = \frac{-z_{imw}(y_{ocw}-y_{imw})}{z_{ocw}-z_{imw}} + y_{imw} \\ z_{wLOC}(t) = 0 \end{cases}$$

che sono proprio le equazioni che permettono di ottenere le coordinate del punto nel piano immagine, espresse nel sistema di riferimento LOCAL. Per riportare il punto rispetto al sistema WORLD sono sufficienti delle traslazioni (vedere figura 4.2).
Il codice che sfrutta questo metodo è stato implementato in MATLAB e C++.

4.4 Cinematica inversa

Per poter muovere la videocamera lungo la traiettoria definita rispetto al sistema WORLD, è necessario sviluppare un algoritmo di cinematica inversa che, date le coordinate di un punto nello spazio, restituisca i valori degli angoli di pan e tilt che consentano alla videocamera di inquadrare quel punto.
Anche in questo caso è stato seguito [4] senza però restringersi al caso di essere ad altezza nulla. Vale la pena riportare una traccia del procedimento in modo da capire meglio dove sono entrate in gioco le nostre modifiche.
Si consideri un generico punto P di coordinate $[x_w\ y_w\ z_w]$ (rispetto a WORLD) che si vuole inquadrare con la videocamera. Innanzitutto si esprime P (tramite traslazioni) rispetto al sistema locale della videocamera che si sta considerando ottenendo le coordinate $[x_{wLOC}\ y_{wLOC}\ z_{wLOC}]$. Dal momento che si desidera centrare il punto da inquadrare perlomeno nel punto principale (non necessariamente uguale al centro del monitor), è necessario posizionare la videocamera in modo che l'asse ottico lo intersechi. Nel nostro caso questo corrisponde ad avere le coordinate di P rispetto al sistema nel centro ottico della videocamera del tipo $[x_{oc}\ \ 0\ \ 0]$ dato che l'asse x è proprio sull'asse ottico.
A questo punto la relazione tra i due punti è sempre quella che si ottiene sfruttando il modello di partenza e cioè:

$$\begin{bmatrix} x_{wLOC} \\ y_{wLOC} \\ z_{wLOC} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ H \end{bmatrix} + R_\theta \left(\begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} + R_\psi \left(\begin{bmatrix} x_{off} \\ y_{off} \\ z_{off} \end{bmatrix} + \begin{bmatrix} x_{oc} \\ 0 \\ 0 \end{bmatrix} \right) \right) \quad (4.5)$$

Dalla (4.5) si devono ricavare gli angoli di pan (θ) e tilt (ψ). A questo scopo seguendo le notazioni di [4] si ponga:

$$\Delta = \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \end{bmatrix} = \begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} + R_\psi \left(\begin{bmatrix} x_{off} \\ y_{off} \\ z_{off} \end{bmatrix} + \begin{bmatrix} x_{oc} \\ 0 \\ 0 \end{bmatrix} \right)$$

Sostituendo Δ nella (4.5) si ottiene:

$$\begin{cases} \delta_1 cos(\theta) - \delta_2 sin(\theta) = x_{wLOC} \\ \delta_1 sin(\theta) + \delta_2 cos(\theta) = y_{wLOC} \\ H + \delta_3 = z_{wLOC} \end{cases} \tag{4.6}$$

Moltiplicando la prima equazione per $sin(\theta)$ e la seconda per $cos(\theta)$ (tenendo presente che questo è possibile per $\theta \neq n\frac{\pi}{2}$ con $n \in \mathbb{Z}$) dopo qualche passaggio si arriva a:

$$cos(\theta) = \frac{\delta_2 y_{wLOC} \pm \sqrt{x_{wLOC}^4 - x_{wLOC}^2 \delta_2^2 + x_{wLOC}^2 y_{wLOC}^2}}{x_{wLOC}^2 + y_{wLOC}^2} \tag{4.7}$$

Per capire quale delle due soluzioni fosse il valore di coseno da considerare, la (4.7) è stata confrontata con un suo valore approssimato:

$$cos(\theta)_{appr} = \frac{x_{wLOC}}{\xi}$$

dove $\xi = \sqrt{x_{wLOC}^2 + y_{wLOC}^2} - D$. In sostanza si considera il triangolo rettangolo sul piano ad altezza H parallelo al pavimento con i cateti e l'ipotenusa pari rispettivamente a x_{wLOC}, y_{wLOC} e ξ. Il segno corretto di θ (una volta applicata la funzione $arccos$) si determina in base al segno di y_{wLOC}.
Se $\theta = \frac{\pi}{2} + n\pi$ per qualche n allora $cos(\theta) = 0$ e $sin(\theta) = \pm 1$ e risulta $x_{wLOC} = \pm\delta_2 = \pm y_{off}$ (l'ultima uguaglianza si ottiene sviluppando i calcoli). Se quindi è verificata l'uguaglianza $x_{wLOC} = \pm y_{off}$ si può determinare θ sapendo di essere in questa condizione "singolare". Il valore di n è stabilito a seconda del quadrante in cui si trovano x_{wLOC} e y_{wLOC}. E' possibile effettuare un ragionamento analogo per $\theta = n\pi$.
Dalla terza equazione della (4.6) si ottiene:

$$z_{off} cos(\psi) - (x_{off} + x_{oc}) sin(\psi) = -H_s \tag{4.8}$$

Elevando poi al quadrato ambo i membri dopo qualche passaggio si ottiene:

$$sin(\psi) = \frac{H_s(x_{off} + x_{oc}) \pm \sqrt{z_{off}^4 - H_s^2 z_{off}^2 + (x_{off} + x_{oc})^2 z_{off}^2}}{(x_{off} + x_{oc})^2 + z_{off}^2} \tag{4.9}$$

dove $H_s = H - z_{wLOC}$. In questa formula è ancora incognito x_{oc}. Questo si ricava mediante un'approssimazione tramite la:

$$x_{oc} \approx \sqrt{\xi^2 + H_s^2}$$

in cui in sostanza si approssima la distanza tra il punto P nello spazio e il centro ottico come distanza tra il punto ad altezza H_s sull'asse di rotazione di pan e il punto P.
Per la scelta del valore del $sin(\psi)$ di tilt corretto si vanno a sostituire i due valori trovati nella (4.8) e si vede quale tra i due si avvicina di più al valore corretto di H_s.
Da queste equazioni si ricavano quindi gli angoli di pan (θ) e tilt (ψ).

4.5 Osservazioni sul metodo utilizzato

E' opportuno riportare qualche osservazione sul metodo appena illustrato.
La (4.7) ha il denominatore che si annulla se x_{wLOC} e y_{wLOC} sono nulle. In effetti questa condizione rappresenta i punti che si trovano sull'asse di rotazione di pan della videocamera (vedere figura 4.3) in cui si ha una singolarità e l'angolo θ non è definito (potrebbe essere qualsiasi). Il problema si ha anche per i punti per cui $x^2_{wLOC} + y^2_{wLOC}$ è piccolo: in sostanza non è possibile ricavare il valore dell'angolo di pan quando la relazione 4.7 del $cos(\theta)$ risulta maggiore di 1 in modulo. Questo limite tuttavia non è risultato un grosso problema: i punti sull'asse di pan si trovano proprio sotto la videocamera dove il campo visivo è limitato. Sono stati stabiliti dei vincoli in modo che la videocamera non potesse mai portarsi nella configurazione di singolarità.
L'approssimazione relativa a x_{oc} porta a un errore sul calcolo dell'angolo di tilt che è maggiore per punti vicini alla videocamera cioè dove l'errore su x_{oc} è più marcato. Tuttavia anche questo non è poi stato troppo limitante: infatti per sfruttare al meglio il campo visivo della videocamera conviene farle inquadrare punti che non le sono troppo vicini, nei quali l'angolo di tilt presenta meno errore.
Si riporta quindi un esempio di 4 punti per testare la cinematica inversa della videocamera 2 del Navlab (vedere figura 4.2 per numerazioni telecamere) nella tabella seguente.

x (mm)	y (mm)	z (mm)	$\lvert\Delta\theta\rvert$ (°)	$\lvert\Delta\psi\rvert$ (°)	Δe (cm)
4638	1727	220	0.7	5.3	35
0	3341	0	0.7	0.2	10
7908	1727	0	0.6	4.1	28
3426	6020	0	0.4	0.4	9

Tabella 4.1: tabella prove cinematica

Nella tabella 4.1 sono riportate le coordinate dei 4 punti di esempio rispetto al sistema WORLD, gli scostamenti $|\Delta\theta|$ e $|\Delta\psi|$ degli angoli di pan e tilt (rispetto al valore corretto) in cui si porta la videocamera quando si calcola la cinematica inversa per quelle coordinate e la distanza Δe tra il punto corretto e quello che corrisponde al punto principale nel monitor. Per riportare le coordinate nel sistema LOCAL2 e valutare la distanza (perlomeno nel piano) tra telecamera e punto è sufficiente ricalcolare i valori di ascissa come $x_{LOCAL2} = y - dx_2$ e di ordinata $y_{LOCAL2} = L_2 - x$, dove $dx_2 = 546mm$ e $L_2 = 6023mm$ sono gli offset del sistema LOCAL2 rispetto al sistema WORLD (vedere figura 4.2). I primi due punti della tabella 4.1 sono riportati con una "x" rossa nella figura (4.5) mentre il "quadratino" bianco rappresenta la posizione del punto principale con la sua incertezza.
Come si può vedere dai valori riportati in tabella 4.1 e dalla figura 4.5, l'errore sull'angolo di tilt, che si ripercuote sulla distanza Δe dal punto "vero", è tanto più marcato quanto i punti sono vicini alla videocamera. Sono state effettuate

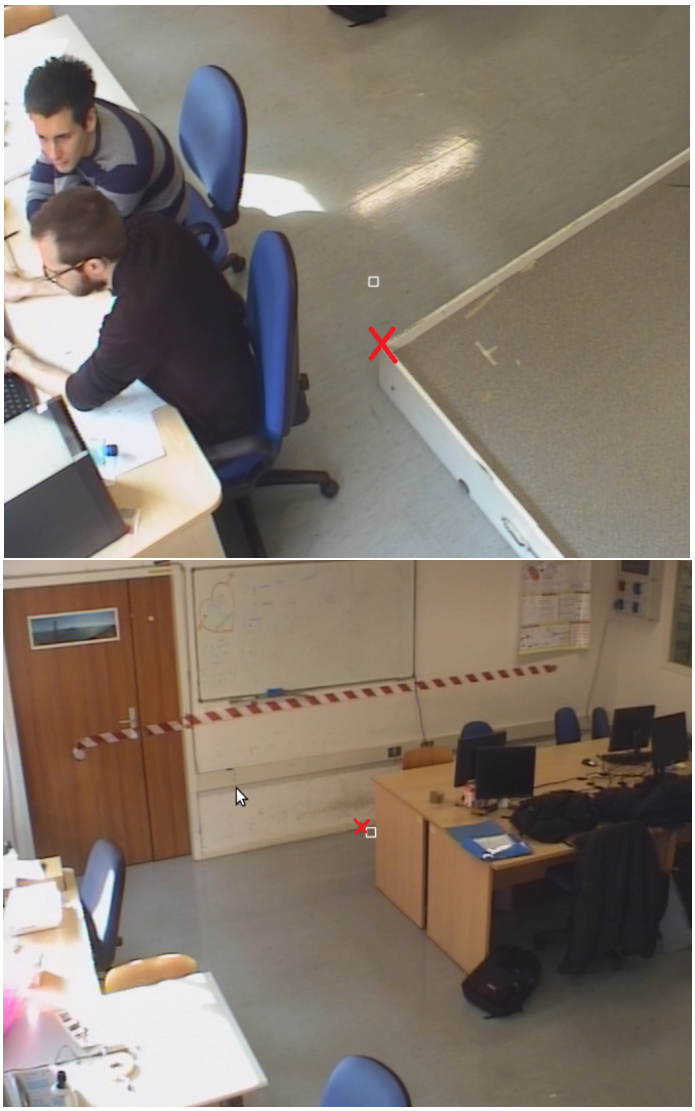


Figura 4.5: Test cinematica inversa

anche prove sulle altre due videocamere del Navlab che hanno presentato un comportamento analogo. Inizialmente con questo metodo non erano stati ottenuti risultati molto soddisfacenti. Dopo vari test si è capito che l'angolo di tilt presentava un offset "intrinseco". Più precisamente, quando si impostava il valore di tilt nullo in realtà la videocamera presentava un angolo ψ di qualche grado, che influiva decisamente sulla bontà della cinematica.
E' necessario tener presente inoltre che, nella cinematica, si ripercuotono tutte le approssimazioni effettuate a monte: la semplificazione a modello pinhole per la videocamera, gli errori sulle misure dei punti per ricavare i parametri del modello meccanico della videocamera (vedere sezione 4.2) e il fatto che tali parametri costituiscono in ogni caso una stima, che è tanto migliore quanto maggiore è il numero delle misure.
E' per questo che ci si può ritenere soddisfatti dei risultati ottenuti: infatti queste imprecisioni sono molto più evidenti in ambienti di dimensioni limitate come il nostro laboratorio e lo sarebbero molto meno in ambienti più vasti, tipici delle applicazioni della videosorveglianza.

Capitolo 5

Controllo videocamere

5.1 Controllo di velocità

Data la tipologia dei comandi con cui è possibile interagire con le camere il controllo più naturale da implementare risulterebbe essere quello di posizione, considerata anche la presenza di encoder piuttosto precisi ai motori. Tale implementazione presenta però un inconveniente abbastanza sgradevole: con un movimento per punti la camera risulta muoversi "a salti".
Dal punto di vista percettivo un movimento così discontinuo risulta estremamente fastidioso e perciò è stato deciso di implementare un controllo di velocità che permetta alle camere di muoversi fluidamente lungo la traiettoria. Non avendo a disposizione alcun sensore di velocità è stata sfruttata l'informazione di posizione fornitaci dagli encoder. Le ipotesi dalle quali partiamo sono:

- movimento dell'ascissa curvilinea s a velocità costante (vel) lungo la traiettoria 3D desiderata (questo si traduce in comandi di velocità ai motori tutt'altro che costanti, dal momento che le velocità nello spazio dei motori e quelle nello spazio operativo 3D sono legate da trasformazioni non lineari)
- istanti di controllo (dt) non inferiori ad una certa soglia a causa dal tempo di attesa da garantire per permettere una corretta comunicazione sulla seriale tra PC e telecamera PTZ.

Queste due ipotesi individuano un limite inferiore (per una velocità fissata vel) sulla scelta del passo (ds) di discretizzazione lungo la traiettoria. Esso infatti dipende dal valore di dt secondo la relazione:

$$ds = vel * dt$$

ds rappresenta dunque il passo (di ascissa curvilinea) che scegliamo di considerare lungo la spline.

5.1.1 Contributo di feed-forward

L'algoritmo del controllo di velocità consta di una procedura che viene iterata ad ogni istante di controllo dt. Tra un istante e l'altro il programma è posto

in attesa (sleep) in modo da aspettare che il comando venga effettivamente eseguito. Il procedimento è costituito dai seguenti punti:

1. Supponendo che la videocamera inquadri il punto *s1* della spline, si ricavano, tramite la funzione `valutazionespline(s)` descritta nella sezione 2.4, le corrispondenti coordinate *x1* nel sistema di riferimento globale e quelle del punto successivo *x2*, che corrisponde al valore di ascissa curvilinea *s1+ds* lungo la spline (posizione da raggiungere);

2. utilizzando l'algoritmo di cinematica inversa si ricavano i valori di angoli di pan (θ_1 e θ_2) e tilt (ψ_1 e ψ_2) necessari per posizionare la camera rispettivamente sui punti $x1$ e $x2$;

3. dai valori degli angoli, per differenza, si ricava l'incremento sui valori di pan e tilt:
$$dp = \theta_2 - \theta_1$$
$$df = \psi_2 - \psi_1$$

4. si ricavano i valori di velocità approssimate da mantenere durante il relativo intervallo *dt*:
$$v_{pan-ff} = \frac{dp}{dt}$$
$$v_{tilt-ff} = \frac{df}{dt}$$

5. si invia alle videocamere tramite scrittura su porta seriale il comando per fissare le velocità di pan e tilt ai valori calcolati.

Tali valori rappresentano solo una componente di *feedforward*. Con questo si intende che tali valori potrebbero essere completamente calcolati offline e quindi sono noti a priori.
Considerando solo tale contributo (completamente in catena aperta), il movimento risulta fluido, ma difficilmente si riesce a rimanere in traiettoria a causa di forti derive dovute a:

- imperfetta sincronia tra comando ed esecuzione dello stesso
- approssimazione della velocità applicata (vedere ipotesi nella sezione 5.1)
- perdita sporadica di un comando da parte della porta seriale.

Essendo il controllo in catena aperta ogni perdita od imperfezione causano uno scostamento irrecuperabile dalla traiettoria desiderata.

5.1.2 Contributo Proporzionale

Per migliorare le prestazione è stato deciso di correggere tali valori di controllo con un contributo di tipo P (proporzionale) ottenuto da una *retroazione di posizione*. In particolare sfruttando un comando ("get") della camera è possibile accedere alle posizioni angolari dei motori e quindi implementare un controllo del tipo:

$$v_{pan-prop} = Kp_{pan} * (\theta_{ff} - \theta_{mis})$$

$$v_{tilt-prop} = Kp_{tilt} * (\psi_{ff} - \psi_{mis})$$

dove indichiamo con Kp_{pan} e Kp_{tilt} i valori del guadagno proporzionale espresso in s^{-1} (che può anche essere differente tra pan e tilt), con θ_{mis} e ψ_{mis} gli angoli che restituisce il comando "get" e con θ_{ff} e ψ_{ff} gli angoli teorici in cui dovrebbe essere la telecamera secondo i calcoli effettuati precedentemente in feedforward.

5.1.3 Controllo complessivo

Il comando di controllo di velocità è costituito dai 2 contributi considerati e cioè:

$$v_{pan} = v_{pan-ff} + Kp_{pan} * (\theta_{ff} - \theta_{mis})$$
$$v_{tilt} = v_{tilt-ff} + Kp_{tilt} * (\psi_{ff} - \psi_{mis})$$

dove Kp_{pan} e Kp_{tilt} sono stati opportunamente tarati attraverso una serie di prove sperimentali.
I comandi in velocità vengono aggiornati ad ogni intervallo di tempo dt, ovvero ogni volta che la camera compie uno spostamento da s a $s+ds$ e percorre quindi lo spazio necessario per raggiungere il punto $x2$ partendo da $x1$.

5.1.4 Problematiche e soluzioni

Le principali problematiche riscontrate sono:

1. perdita di comandi da parte della seriale;
2. mancanza di sincronismo nei comandi;
3. tempo di percorrenza del tratto di curva desiderato ben superiore a quello previsto considerando una movimentazione a velocità costante dell'ascissa curvilinea lungo la spline;
4. misure erronee degli angoli di pan e tilt dovute a "corruzione dei dati" da parte della seriale.

Di seguito si cerca di analizzarle e si spiega la soluzione trovata.

Linea seriale

Il filo condutture per le 4 cause di errore è la linea di comunicazione seriale.
Ogni volta che si accede ad essa è richiesto un tempo minimo di attesa (70ms) per permettere un corretto scambio dei pacchetti inviati. Se la comunicazione richiede una trasmissione bidirezionale (come la richiesta delle misure degli encoder) tale tempo di attesa raddoppia. Questo ha una serie di conseguenze. In particolare:

- fissata una certa velocità di movimento dell'ascissa curvilinea, come accennato all'inizio del capitolo, non è possibile eseguire una discretizzazione troppo fitta della traiettoria da seguire, dal momento che questo comporterebbe uno scambio di informazione troppo frequente, con conseguente perdita dei pacchetti inviati (problematica 1);

- ogni volta che si accede alla linea, per permettere il rispetto del tempo di attesa richiesto è introdotto un ritardo (sleep del programma). Questo significa che durante tale intervallo di tempo non viene eseguita nessun'altra istruzione e dunque ogni volta che si accede alla seriale i comandi vengono ritardati di un tempo pari al tempo di attesa richiesto (problematica 2).
- ogni volta che si accede alla linea seriale per ottenere le misure degli angoli di pan e tilt attuali, viene introdotto un ritardo. Il numero di ritardi dipende quindi dal numero di punti con cui si discretizza la traiettoria. E' chiaro quindi che il tempo di percorrenza effettivo si scosterà tanto più da quello calcolato offline come:

  $$T = \frac{L_{tot}}{vel}$$

  dove L_{tot} è la lunghezza totale della spline scelta e vel la velocità di percorrenza prevista, quanto minore è il passo di discretizzazione ds scelto (problematica 3);
- pur rispettando il tempo di attesa, la linea è afflitta da perdita sporadica di pacchetti. Questo può portare a una restituzione di valori di pan e tilt misurati completamente diversi da quelli reali, con conseguenze disastrose sui comandi in velocità calcolati col controllo proposto nella sezione 5.1.3 (problematica 4);

Per risolvere i primi 3 problemi, fissata una certa velocità di percorrenza dell'ascissa curvilinea lungo la curva, è stato scelto un passo ds in modo da garantire un buon compromesso per l'intervallo di campionamento dt. Verrebbe infatti da prendere ds in modo che dt sia maggiore del tempo di attesa della seriale e grande abbastanza in modo da evitare una discretizzazione troppo fitta che richiederebbe troppe misure degli angoli di pan e tilt. Tuttavia, un valore di dt troppo grande porta ad un controllo poco frequente a cui consegue uno scostamento maggiore dalla traiettoria pianificata. Nella sezione 5.1.5 verrà descritto più in dettaglio come è stato scelto il passo ds.
Per risolvere la problematica 4 è stato invece deciso di disabilitare il controllo quando la comunicazione seriale non è andata a buon fine e di applicare quindi solo la componente di feedforward. In particolare, la comunicazione errata viene diagnosticata da un particolare carattere di controllo che fa parte del messaggio inviato "dalla PTZ al pc".

5.1.5 Tuning del controllo

Per effettuare una corretta scelta dei parametri di controllo è stato deciso di procedere in 3 passi:

- scelta della velocità di movimentazione dell'ascissa lungo la spline;
- scelta del passo di discretizzazione;
- scelta dei valori delle costanti di controllo.

La prima scelta è stata effettuata basandosi più che altro su criteri percettivi per far si che la traiettoria fosse percorsa con una velocità adeguata tenendo presente che lo spazio da percorrere è abbastanza ridotto. E' stato scelto $vel = 450\ mm/s$.

Osservazione

Poniamo per un attimo l'attenzione sulla movimentazione della camera per evitare di fraintendere i risultati ottenuti. Si è sempre parlato di movimenti a velocità costante lungo la spline. Con questo si intende che l'ascissa curvilinea si muove lungo la traiettora con tale velocità. Non ci si deve quindi aspettare che un tale movimento corrisponda ad un altrettanto movimento a velocità costante da parte dell'inquadratura della telecamera. Questo dipende infatti dalla prospettiva con la quale viene vista la traiettoria dalla telecamera che, ricordiamo, "vede" in 2D e non in 3D. In particolare si rifletta su un movimento dell'ascissa s a velocità costante lungo una traiettoria che cade esattamente lungo l'asse ottico della videocamera. Anche se l'ascissa procede a velocità costante, la videocamera non cambia la sua posizione dal momento che non percepisce alcuno spostamento dell'ascissa stessa all'interno del suo campo visivo.

Per riuscire ad effettuare una corretta scelta dei parametri in gioco sono state fissate alcune *metriche* in modo da poter confrontare l'effetto di parametri diversi.

Metriche di riferimento

Le *metriche* scelte sono:

- **tempo effettivo di percorrenza lungo la curva**: questa misura deve risultare il più vicino possibile al tempo di percorrenza teorico dell'ascissa curvilinea;
- **media campionaria di pan e tilt** calcolata sui valori di ritorno dai "get" degli angoli di pan e tilt nei punti di discretizzazione della traiettoria secondo le:

$$m_{pan} = \sum_i \frac{(\theta_{iff} - \theta_{imis})}{N_{correct}}$$

$$m_{tilt} = \sum_i \frac{(\psi_{iff} - \psi_{imis})}{N_{correct}}$$

  dove θ_{iff} e ψ_{iff} sono i valori degli angoli di pan e tilt valutati in feedforward relativi al punto di controllo i-esimo , θ_{imis} e ψ_{imis} i valori misurati degli angoli di pan e tilt relativi al punto di controllo i-esimo. Più precisamente vengono considerati per il calcolo solo gli $N_{correct}$ punti di controllo in cui le misure degli angoli di pan e tilt sono andate a buon fine, dato che sui rimanenti il confronto perde di significato. Ci si aspetta che i valori di media oscillino attorno a zero dato che almeno nei punti di controllo la traiettoria reale deve essere molto vicina a quella pianificata;
- **varianza campionaria di pan e tilt** calcolata sugli stessi valori di controllo considerati sopra tramite le:

$$var_{pan} = \sum_i \frac{(\theta_{iff} - \theta_{imis} - m_{pan})^2}{N_{correct}}$$

$$var_{tilt} = \sum_i \frac{(\psi_{iff} - \psi_{imis} - m_{tilt})^2}{N_{correct}}$$

Anche in questo caso ci si aspetta che i valori oscillino attorno allo zero dal momento che la dispersione almeno attorno ai punti di controllo dovrebbe rimanere contenuta.

Gli esempi che si riportano per l'analisi fanno riferimento ad un tratto di curva lungo $L = 7m$ avendo impostato la velocità pari a $vel = 450mm/s$.
Dalla tabella 5.1 (ottenuta per valori di $Kp_{pan} = Kp_{tilt} = K_p = 0.5$) si vede come, per un valore del passo troppo piccolo, i tempi effettivi di percorrenza T_{eff} si discostano troppo dal valore teorico pari a

$$T_{teorico} = \frac{L}{vel} \simeq 15.6s$$

In particolare valori troppo piccoli del passo causano eccessivi ritardi dal momento che il numero di letture dalla seriale diventa eccessivo; d'altro canto valori troppo elevati invece, che sembrerebbero ben rispettare il tempo di percorrenza (ad es. $ds = 800mm$) portano a una scarsa efficacia del controllo che agisce poco frequentemente con un conseguente allontanamento eccessivo dalla traiettoria ideale.

passo ds (mm)	250	350	500	600	**650**	700	800
T_{eff} (s)	19.91	18.68	17.74	16.38	**16.03**	17.10	15.47

Tabella 5.1: tabella scelta passo

Infatti per i vari passi, è stato verificato anche come la telecamera seguisse la spline (realizzata con il nastro segnaletico nel Navlab) e come variavano la media e la varianza campionarie, in modo tale da trovare un compremesso tra ritardi introdotti ed efficacia del controllo. Fissato il passo $ds = 650mm$ si può passare al tuning dei guadagni di controllo confrontando i vari valori di media e varianza campionarie. E' stato considerato $Kp_{pan} = Kp_{tilt} = K_p$, ma in generale possono essere diversi.

Kp	m_{pan}	m_{tilt}	var_{pan}	var_{tilt}
2	0.71	0.5	25	15
1.7	0.3	0.1	5	1.4
0.8	0.25	0.03	0.45	0.03
0.5	0.05	0.02	0.42	0.06

Tabella 5.2: tabella scelta guadagni controllo

Dai valori riportati in tabella 5.2 risulta evidente come per valori troppo grandi delle costanti di controllo media e varianza crescono e in particolare il movimento reale della camera risulta molto oscillante e lontano dal comportamento desiderato. In realtà per valori attorno a $K_p = 0.5$ le prestazioni numeriche e visive non cambiano moltissimo e sono stati quindi mantenuti questi valori.

5.2 Controllo di zoom

5.2.1 Zoom continuo

Le funzioni già implementate a nostra disposizione prevedevano solo uno zoom "discretizzato" in quanto consentivano di impostare esclusivamente valori interi. Per riuscire ad imporre un livello di zoom che non fosse limitato a 1x, 2x,...,10x è stato necessario implementare una funzione che permettesse di scegliere anche valori intermedi. Le videocamere Ulisse da noi utilizzate dispongono di un comando per la regolazione dello zoom in 16384 livelli diversi. Esso accetta come parametro un certo valore del livello (tra 0 e 16534) ed imposta di conseguenza un valore di zoom: il valore 0 corrisponde a uno zoom di 1x mentre 16384 a uno zoom di 10x.
Purtroppo però, all'aumentare del parametro il valore di zoom (in x) non cresce linearmente. E' stato quindi necessario ricavare una funzione che calcolasse il livello da utilizzare per impostare il valore di zoom desiderato.
A questo scopo sono stati rilevati, per 20 valori del parametro, i valori corrispondenti di zoom tramite il comando "get" che restituisce, oltre ai valori di pan e tilt, anche lo zoom attuale. In particolare i primi 10 parametri sono stati scelti in modo da ottenere valori di zoom nelle vicinanze dell'intervallo [1x 2.4x], mentre i secondi 10 in quelle dell'intervallo [2.4x 4.8x]. Si è scelto di non considerare valori di zoom superiori in quanto risultavano inadeguati nel nostro ambiente piuttosto ridotto. Nella tabella (5.3) seguente sono riportati i valori ottenuti. Tramite questi valori sono state ottenute due rette interpolanti dal momento che una sola avrebbe introdotto degli errori troppo elevati.
Infatti utilizzando Matlab ed in particolare le funzioni *polyfit* e *polyval* sono

parametro	zoom (x)	parametro	zoom (x)
0	1	9000	2.68
1000	1.26	10000	3.11
2000	1.45	10250	3.23
3000	1.6	10500	3.37
4000	1.72	10750	3.51
5000	1.83	11000	3.66
6000	1.96	11500	3.99
7000	2.13	12000	4.37
8000	2.37	12500	4.79

Tabella 5.3: tabella misure zoom

state ricavate le equazioni delle rette interpolanti che più si avvicinavano ai dati, cioè alle coppie parametro-valore di zoom. In figura (5.1) è riportato il risultato.
Attraverso questa procedura è stata creata una funzione (denominata `movezoom(z)`) che accetta come parametro di ingresso il valore di zoom (in x) e imposta il valore effettivo di zoom della videocamera a uno molto prossimo a quello "passato in ingresso".
Questa mappa consente di ottenere piccoli errori tra zoom desiderato ed impostato nell'intervallo [1x 4.7x], dato che i dati utilizzati sono interni a questo intervallo. In particolare il metodo introduce un errore massimo tra lo zoom

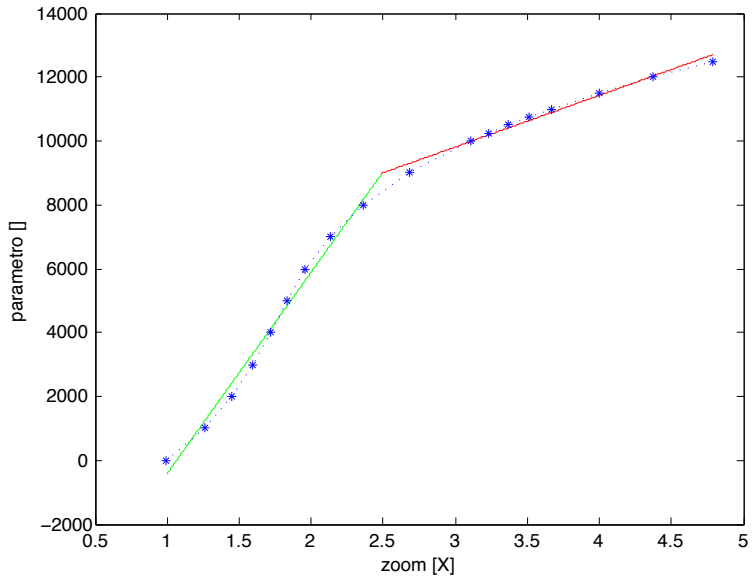


Figura 5.1: Retta interpolante e punti utilizzati per il calcolo della retta.

calcolato e quello impostato dell'ordine del decimo di x.
Se si volesse utilizzare un range di zoom maggiore si dovrebbe ripetere la procedura ricavando un'ulteriore retta interpolante per l'intervallo di interesse.

5.2.2 Rapporto di zoom

A questo punto è necessario stabilire "quanto grande si vuole inquadrare un oggetto". A tal scopo si sceglie un particolare valore del *rapporto di zoom* definito dalla

$$R_{zoom} = \frac{d_0}{z_0}$$

dove con d_0 si intende la distanza di un oggetto campione dalla videocamera e con z_0 il valore di zoom (in x) con cui si decide di inquadrare tale oggetto (per visualizzarlo alla dimensione desiderata).
Fissato il rapporto R_{zoom} è immediato quindi calcolare, sfruttando la medesima relazione, il valore di zoom necessario ad inquadrare un oggetto con le stesse dimensioni al variare della distanza (calcolabile attraverso la funzione di cinematica diretta del capitolo 4). Infatti supponendo che l'oggetto da inquadrare sia a distanza d, il valore di zoom da impostare per mantenere il rapporto di zoom costante, sarà pari a:

$$z = \frac{d}{R_{zoom}} \tag{5.1}$$

Dal momento che il valore z sarà in generale un numero non intero la procedura di linearizzazione vista nella sezione precedente risulta necessaria.

5.2.3 Risultati

Come si può vedere in Figura 5.2 il pallone viene mantenuto pressochè sempre delle stesse dimensioni anche se si allontana di qualche metro (da una foto all'altra) dalla telecamera.

Nell'algoritmo finale si è comunque deciso di disabilitare il controllo dello zoom dal momento che l'inquadratura risultava muoversi a "salti" e l'effetto visivo era poco piacevole. Questo è dovuto al fatto che lo zoom viene impostato attraverso comandi di "posizione" ovvero, la videocamera imposta il livello di zoom richiesto muovendolo a velocità costante, indipendente dal valore desiderato. Non avendo a disposizione alcun comando relativo alla velocità di movimentazione dello zoom non è stato possibile implementare un controllo che garantisse un movimento fluido, al contrario di quanto fatto per la movimentazione di pan e tilt.

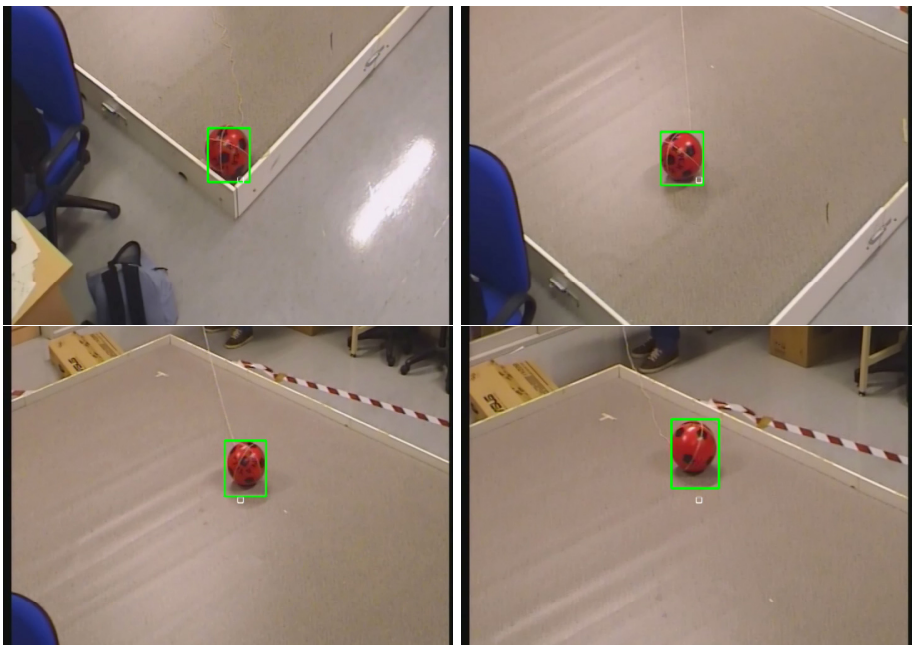


Figura 5.2: Controllo zoom

5.2.4 Problematiche

Le principali problematiche incontrate durante l'implementazione del controllo dello zoom sono state:

- calibrazione dei parametri intrinseci ed estrinseci della videocamera;
- calcolo della distanza dell'oggetto;
- approssimazione migliore dei livelli di zoom.

Calibrazione parametri intrinseci ed estrinseci della videocamera

Come è stato accennato nella sezione 3.2, i parametri intrinseci sono costanti solamente fissato un certo livello di zoom. Questo significa che variando quest'ultimo, tali parametri cambiano e dovrebbero essere ricalcolati per ogni livello di zoom. Un semplice modo per migliorare i risultati sarebbe quello di calibrare le camere per un set di livelli di zoom e cercarne quindi un interpolazione per ottenere i parametri relativi i livelli di zoom intermedi.
Da notare che il cambiamento di tali parametri si ripercuote inoltre sui parametri estrinseci della videocamera dal momento che per ottenerli bisogna sfruttare la calibrazione di quelli intrinseci (vedere capitoli 3 e 4).

Il fatto di utilizzare parametri diversi da quelli effettivi (per un dato livello di zoom) della telecamera si ripercuote sul posizionamento dell'oggetto all'interno dell'inquadratura dal momento che, da quanto detto, i parametri della cinematica dovrebbero essere differenti. È del resto vero che per valori di zoom non troppo elevati l'oggetto continua a rimanere inquadrato (questo è il nostro caso, si vedano figura 5.3) ma non si trova più centrato sul punto principale della videocamera.

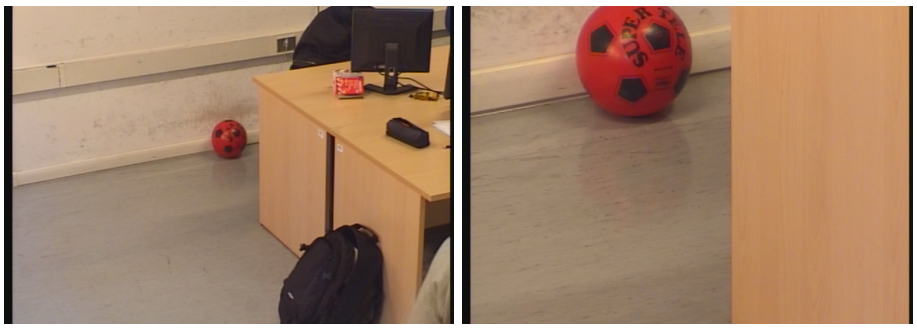


Figura 5.3: Posizionamento dell'oggetto "zoomato" rispettivamente a 2x e 6x

Calcolo della distanza dell'oggetto

Come visto nella sezione 5.2.2, per mantenere un oggetto a dimensione costante è necessario conoscerne la distanza tra il suo centro e quello ottico della videocamera in modo da utilizzare correttamente la relazione 5.1. Dal momento che con una videocamera è possibile conoscere solo la posizione di un oggetto sul piano $z = 0$ (vedere sezione 4.3), qualora volessimo conoscere la distanza di un oggetto ad altezza diversa da 0 otterremmo un valore erroneo dal momento che il raggio ottico che interseca l'oggetto, individuerebbe nel piano $z = 0$ un punto tanto più distante quanto più alto è l'oggetto da terra.

Approssimazione migliore dei livelli di zoom

Nella sezione 5.2.1 è stato esposto un metodo per settare valori di zoom in un intervallo continuo, utilizzando un'interpolazione lineare che introduce inevitabilmente degli errori. Per ottenere un errore inferiore tra valore di zoom calcolato e valore di zoom effettivamente impostato si potrebbe per esempio:

- raccogliere un set di valori più fitto;
- eseguire un'interpolazione non lineare, ad esempio cubica in modo da seguire meglio l'andamento dello zoom.

Capitolo 6

Comunicazione videocamere e partitioning

6.1 Partitioning Distribuito

L'obiettivo che si vuole raggiungere è quello di riuscire ad implementare un algoritmo per il *partitioning distribuito*.
Nella nostra applicazione nell'ambito della videosorveglianza, questo significa ricavare un algoritmo in grado di suddividere in maniera ottima l'area da pattugliare quando ogni telecamera ha a disposizione solo informazioni che gli arrivano dalle telecamere vicine.
Dal momento che il nostro scopo è quello di riuscire ad effettuare il patrolling di una linea arbitraria dello spazio, il partitioning si riconduce al problema di "partizionare" in maniera ottima la linea, in modo da minimizzare il tempo tra una visita e quella successiva di ciascun punto.
La linea, o traiettoria, deve essere suddivisa in segmenti non sovrapposti ognuno dei quali viene sorvegliato da una videocamera.
Vengono tenuti in considerazione anche i range fisici di movimento delle videocamere (pan e tilt) e i limiti imposti dalla topologia dell'area che contiene la traiettoria da sorvegliare (ad esempio ostacoli che impediscono a una videocamera di inquadrarne una parte).
Ad ogni videocamera sono assegnati due estremi sulla traiettoria (*estremi di patrolling*) che definiscono l'intervallo che essa deve sorvegliare.

6.1.1 Il problema del patrolling

La nostra traiettoria, come visto nel capitolo 2, è una spline parametrizzata in ascissa curvilinea di lunghezza fissata e quindi, anche se la traiettoria si trova "immersa" nello spazio 3D, si può considerare il problema come nel caso 1D. Infatti tramite l'ascissa curvilinea si può pensare a una versione "rettificata" della traiettoria in 3D.
E' stato seguito l'approccio proposto in [12]: se ne riportano per completezza le parti fondamentali e i nostri adattamenti.
Sia $\mathcal{L} = [0, L]$, $L > 0$ la traiettoria da sorvegliare espressa in ascissa curvilinea e

sia N il numero di videocamere utilizzate (nel nostro caso $N = 3$). Ognuna di queste è identificata da un numero che va da 1 a N.

Ipotesi:

- le videocamere non hanno un field of view (f.o.v.) puntiforme. Tuttavia noi consideriamo solo il punto che la videocamera inquadra nella direzione dell'asse ottico e ci riconduciamo al caso di f.o.v. puntiforme;
- il movimento delle videocamere non è limitato solo al pan come in [12]: l'ascissa curvilinea può individuare generici punti dello spazio i quali possono essere centrati sul punto principale, utilizzando la funzione di cinematica inversa descritta nella sezione 4.4;
- le videocamere hanno un intervallo di copertura costante;

Nel seguito si farà uso delle seguenti variabili:

1. $D_i = [D_{i,inf}, D_{i,sup}] \subset \mathcal{L}$ è l'intervallo di copertura della videocamera i fissato in base alla topologia dell'area che contiene la nostra traiettoria, alla configurazione e ai limiti della videocamera.
2. $z_i(t) : \mathbb{R}^+ \to D_i$ è la funzione continua che mappa la posizione dell'ascissa curvilinea s_i relativa alla videocamera i come funzione del tempo.
3. $v_i \in [-V_{i,max}, +V_{i,max}]$ è la velocità con cui $z_i(t)$ si muove lungo la traiettoria. Nel nostro algoritmo abbiamo assunto $v_i = vel \quad \forall i \in \{1, 2, ..., N\}$, ma manteniamo la simbologia di [12] per considerare un caso più generale.
4. $A_i = [a_{i,l}, a_{i,r}]$ rappresenta l'intervallo della traiettoria assegnato alla videocamera i. Ovviamente devono essere soddisfatte le condizioni $A_i \subseteq D_i$, $\forall i \in \{1, 2, ..., N\}$.

Nel seguito si assume che gli intervalli di copertura D_i, $i \in \{1, 2, ..., N\}$, soddisfino i seguenti vincoli di *collegamento*,

$$D_{i,inf} \leq D_{i+1,inf}, \qquad D_{i,sup} \leq D_{i+1,sup}$$

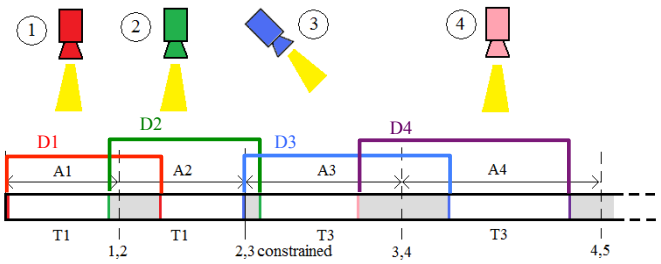


Figura 6.1: Patrolling di una traiettoria rettilinea (caso 1D) in cui sono evidenziati gli intervalli D_i e la soluzione ottima del partitioning con vincoli rappresentata dagli intervalli A_i.

Si assume che all'istante iniziale $t = 0$ ad ogni videocamera sia assegnata una porzione della traiettoria definita da $A_i(0) = [a_{i,l}(0), a_{i,r}(0)]$ per $i \in \{1, 2, ..., N\}$.

$a_{i,l}(t)$ e $a_{i,r}(t)$ verranno chiamati estremi di patrolling e rappresentano rispettivamente l'estremo sinistro e l'estremo destro dell'intervallo A_i. Si assume che $\{A_1(0), ..., A_N(0)\}$ soddisfino i seguenti tre vincoli:

- vincolo fisico: $A_i \subseteq D_i$, $\forall i \in \{1, 2, ..., N\}$
- vincolo di copertura: $\bigcup_{i=\{1,...,N\}} A_i(0) = \mathcal{L}$
- vincolo di collegamento: $a_{i,l}(0) \leq a_{i+1,l}(0)$ $\quad$ $a_{i,r}(0) \leq a_{i+1,r}(0)$

Si fa notare che i vincoli di collegamento e di copertura implicano che $a_{1,l}(0) = 0$ e che $a_{N,r}(0) = L$.
L'algoritmo che abbiamo implementato prevede l'utilizzo delle relazioni dimostrate nell'articolo [12], nel quale è stato realizzato un algoritmo che permette alle videocamere di aggiornare iterativamente i loro estremi di patrolling, basandosi solo su informazioni di tipo locale (provenienti dalle videocamere adiacenti). Nell'articolo [12] vengono considerati 3 possibili casi: un sistema con comunicazione sincrona, comunicazione asincrona di tipo *gossip* simmetrico e comunicazione asincrona di tipo *gossip* asimmetrico. Inoltre la trattazione del problema è stata impostata risolvendo prima il problema senza vincoli fisici e poi aggiungendo i vincoli.

6.2 Algoritmo di partitioning ottimo con comunicazione sincrona

Nel nostro caso, visto il tipo di comunicazione già presente in laboratorio (si veda il paragrafo 6.3), è stato utilizzato l'algoritmo per sistemi con comunicazione sincrona. Tale algoritmo assicura che:

- Ad ogni iterazione i vincoli di copertura e di collegamento siano soddisfatti
- L'insieme degli intervalli A_i converge alla partizione ottima della traiettoria

6.2.1 Algoritmo senza limiti fisici

Si assume che le videocamere siano inizializzate con delle partizioni $A_i(0)$ della traiettoria che non coincidono necessariamente con quelle ottime e che queste soddisfino le condizioni di copertura e di collegamento.
Ad ogni iterazione ciascuna videocamera trasmette i propri estremi a quelle adiacenti, riceve da queste i loro estremi di patrolling e aggiorna gli estremi basandosi sulle informazioni ricevute.
Verranno indicate con v_i le velocità delle ascisse curvilinee s_i, assunte nel nostro caso tutte uguali a *vel* dal momento che si vuole mantenere la stessa velocità di percorrenza della traiettoria per ogni videocamera.
Al tempo t per $i \in \{2, ..., N-1\}$ la videocamera i aggiorna il valore del suo estremo attuale $a_{i,l}(t)$ al valore $a_{i,l}(t+1) \in [a_{i-1,l}(t), a_{i,r}(t)]$ utilizzando il seguente criterio:

il tempo richiesto affinchè l'ascissa curvilinea s_i si muova lungo la traiettoria alla velocità v_i dall'estremo $a_{i,l}(t+1)$ all'estremo $a_{i,r}(t)$ è uguale al tempo richiesto affinchè l'ascissa curvilinea s_{i-1} si muova lungo la traiettoria alla velocità v_{i-1} dall'estremo $a_{i-1,l}(t)$ all'estremo $a_{i,l}(t+1)$.

Allo stesso modo la videocamera i aggiorna il valore dell'estremo destro $a_{i,r}(t)$ al valore $a_{i,r}(t+1) \in [a_{i,l}(t), a_{i+1,r}(t)]$ utilizzando il seguente criterio:

il tempo richiesto affinchè l'ascissa curvilinea s_i si muova lungo la traiettoria alla velocità v_i dall'estremo $a_{i,l}(t)$ all'estremo $a_{i,r}(t+1)$ è uguale al tempo richiesto affinchè l'ascissa curvilinea s_{i+1} si muova lungo la traiettoria alla velocità v_{i+1} dall'estremo $a_{i,r}(t+1)$ all'estremo $a_{i+1,r}(t)$.

Formalmente questi criteri si possono descrivere con le seguenti equazioni:

$$\frac{a_{i,l}(t+1) - a_{i-1,l}(t)}{v_{i-1}} = \frac{a_{i,r}(t) - a_{i,l}(t+1)}{v_i} \tag{6.1}$$

$$\frac{a_{i+1,r}(t) - a_{i,r}(t+1)}{v_{i+1}} = \frac{a_{i,r}(t+1) - a_{i,l}(t)}{v_i} \tag{6.2}$$

Da queste equazioni è possibile quindi ricavare le equazioni di aggiornamento

$$a_{i,l}(t+1) = \frac{a_{i,r}(t)v_{i-1} + a_{i-1,l}(t)v_i}{v_{i-1} + v_i} \tag{6.3}$$

$$a_{i,r}(t+1) = \frac{a_{i+1,r}(t)v_i + a_{i,l}(t)v_{i+1}}{v_i + v_{i+1}} \tag{6.4}$$

per $i = 1$ e $i = N$ abbiamo inoltre che:

$$a_{1,l}(t) = 0 \qquad a_{N,r}(t) = L \qquad \forall t \in \mathbb{N}$$

Si fa notare che (6.3) e (6.4) implicano

$$a_{i,r}(t) = a_{i+1,l}(t) \qquad \forall i \in \{1, ..., N-1\} \tag{6.5}$$

Queste formule sono state implementate in C++ nella funzione `boundrefresh()`.

6.2.2 Algoritmo con limiti fisici

Nel seguito si assume che le videocamere memorizzino $a_{i,l}$, $a_{i,r}$, v_{i-1}, v_i, v_{i+1}, D_i, D_{i-1} e D_{i+1}.
In presenza di limiti fisici, i valori calcolati con le formule (6.3) e (6.4) potrebbero essere al di fuori dell'intervallo dei limiti fisici D_i: ad esempio si potrebbe ottenere che $a_{i,l}(t+1) < D_{i,inf}$ oppure che $a_{i,r}(t+1) > D_{i,sup}$.
Consideriamo ora l'aggiornamento degli estremi $a_{i,r}$ e $a_{i+1,l}$ in presenza di vincoli.
Distinguiamo quindi tre casi (tenendo presente la relazione 6.5):

1. $a_{i,r}(t+1) > D_{i,sup}$
2. $a_{i,r}(t+1) < D_{i+1,inf}$
3. $a_{i,r}(t+1) \in [D_{i+1,inf}, D_{i,sup}]$

Nel caso in cui si verifichi il caso (1) gli estremi vengono imposti pari a:

$$a_{i,r}(t+1) = a_{i+1,l}(t+1) := D_{i,sup}$$

nel caso (2) invece:

$$a_{i,r}(t+1) = a_{i+1,l}(t+1) := D_{i+1,inf}$$

infine nel caso (3):

$$a_{i,r}(t+1) = a_{i+1,l}(t+1) := \frac{a_{i+1,r}(t)v_i + a_{i,l}(t)v_{i+1}}{v_i + v_{i+1}}$$

Si può osservare che queste condizioni consentono di soddisfare ad ogni iterazione il vincolo di copertura (per rendersene conto si veda la figura 6.1).

Si procede in modo analogo per l'aggiornamento degli estremi $a_{i,l}$ e $a_{i-1,r}$ distinguendo i seguenti tre casi:

1. $a_{i,l}(t+1) < D_{i,inf}$
2. $a_{i,l}(t+1) > D_{i-1,sup}$
3. $a_{i,l}(t+1) \in [D_{i,inf}, D_{i-1,sup}]$

Nel caso in cui si verifichi il caso (1) gli estremi vengono imposti pari a:

$$a_{i,l}(t+1) = a_{i-1,r}(t+1) := D_{i,inf}$$

nel caso (2) invece:

$$a_{i,l}(t+1) = a_{i-1,r}(t+1) := D_{i-1,sup}$$

infine nel caso (3):

$$a_{i,l}(t+1) = a_{i-1,r}(t+1) := \frac{a_{i,r}(t)v_{i-1} + a_{i-1,l}(t)v_i}{v_{i-1} + v_i}$$

6.3 Comunicazione tra le videocamere

Il sistema a nostra disposizione è costituito da 3 videocamere ognuna delle quali è connessa ad un PC per mezzo di un'interfaccia seriale. I PC sono connessi ad una rete locale LAN che permette loro lo scambio di dati utilizzando un protocollo di comunicazione di tipo TCP/IP. Tutti i computer connessi mediante TCP/IP, possiedono un loro indirizzo unico, detto indirizzo IP (Internet Protocol) che si compone di quattro numeri delimitati da un punto, ognuno dei quali è compreso tra 0 e 255.
La parte riguardante lo sviluppo del software in C++ per implementare questo tipo di comunicazione era già presente nei PC del laboratorio, ma è stata leggermente modificata per adattarla alla nostra applicazione.

6.3.1 Architettura del sistema di comunicazione

Ognuno dei tre PC collegati alla rete implementa delle funzioni C++ per svolgere le tipiche operazioni di Server/Client. Ogni PC può trasmettere agli altri PC connessi in rete un messaggio riguardante il proprio *Stato* e allo stesso tempo è in ascolto in attesa di ricevere gli *Stati* degli altri PC .
Lo *Stato* contiene informazioni relative alla connessione in rete della relativa videocamera, la sua modalità di funzionamento(patrolling, tracking o altro), gli estremi di patrolling e la sua velocità.
In particolare quando viene ricevuto un dato (*Stato* di una delle videocamere connesse in rete), questo viene memorizzato in un'apposita area di memoria che denominiamo *Stato Attuale*. E' importante sottolineare che ad ogni PC è associata una struttura di tipo *Stato Attuale* la quale contiene dei puntatori a delle altre strutture di tipo *Stato* (una per ogni PC connesso alla rete). E' in quest'ultime che ogni PC memorizza il proprio *Stato* e quello ricevuto dagli altri PC. Dalla lettura di quest'area di memoria si possono conoscere gli *Stati* di ogni videocamera connessa in rete.
Quando viene eseguito il programma su un PC, questo trascrive nel proprio *Stato Attuale* tutte le informazioni relative alla rispettiva videocamera ed inizializza a zero lo spazio riservato agli *Stati* delle altre videocamere. Essi vengono aggiornati nello *Stato Attuale* quando vengono ricevuti i dati inviati dalle altre.
Si riportano di seguito delle figure che schematizzano la struttura dello *Stato Attuale*.

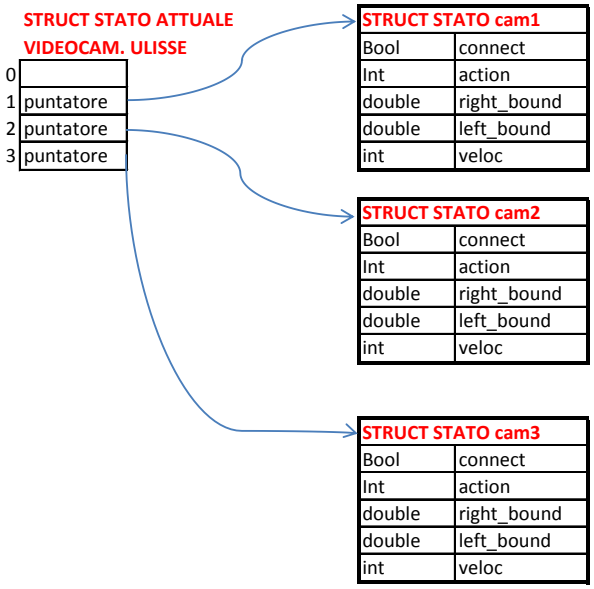


Figura 6.2: Schematizzazione della struct di tipo Stato Attuale

Per il nostro obbiettivo, che è quello di implementare un algoritmo distribuito, alcune informazioni sono superflue: ad ogni videocamera sono sufficienti le informazioni ricevute da quelle adiacenti, mentre questo tipo di sistema di comunicazione permetterebbe ad ogni videocamera di conoscere lo *Stato* di tutte le altre. Per simulare l'approccio distribuito si è quindi semplicemente evitato di andare a leggere dallo *Stato Attuale* le informazioni non necessarie (vedere figura 6.3).

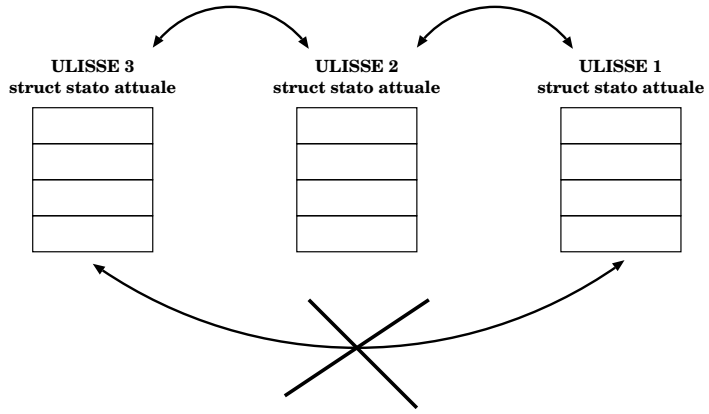


Figura 6.3: Schema di comunicazione

Nello schema in Figura 6.3 si riporta la cancellazione "ideale" del ramo di comunicazione tra le videocamere non adiacenti.

6.3.2 Messaggio Scambiato

Per poter utilizzare le relazioni riportate nella sezione 6.2.2 è necessario che a ciscuna videocamera vengano inviati gli estremi e le velocità di quelle adiacenti. Ecco che si spiega la struttura attribuita allo *Stato* che costituirà parte del nostro messaggio (fatta eccezione per la variabile connect che non viene inviata). Lo *Stato* infatti indica se la telecamera relativa è connessa in rete e la sua azione (nel nostro caso patrolling indicata con 2, tracking con 1, condizione di riposo con 0), oltre agli estremi attuali di patrolling e alla velocità (vedere figura 6.2). Il messaggio scambiato tra le videocamere dovrà inoltre contenere l'identificativo della videocamera che ha inviato il messaggio (nel nostro caso è un numero da 1 a 3). Il PC che riceve il messaggio andrà quindi ad aggiornare nello *Stato Attuale* lo *Stato* relativo al PC che ha inviato tale messaggio. Il messaggio scambiato tra i PC è costituito da una stringa. Riportiamo di seguito un esempio:

1.2100000.0020000.00500

Questa stringa ci dice che il PC1 ha spedito il messaggio, la videocamera connessa al PC1 sta svolgendo un'azione di tipo azione 2 cioè patrolling (le azioni possibili sono patrolling, tracking e condizione di riposo), il suo estremo sinistro di patrolling è 10000.00, l'estremo destro è 20000.00 e che la velocità dell'ascissa curvilinea lungo la traiettoria è pari a 500 mm/s.

6.4 Il nostro algoritmo di partitioning

Prima di descrivere più in dettaglio il nostro algoritmo è importante evidenziare che è necessario stabilire un ordine delle videocamere proprio per dare senso al concetto di precedente e successivo. Conviene quindi riguardare la disposizione delle videocamere nel laboratorio Navlab riportata in figura (4.2). La nostra scelta è stata quella di prendere come prima videocamera la 3, come seconda la 2 e per ultima la 1, secondo la numerazione riportata in figura (4.2).
Questa scelta non comporta perdita di generalità: infatti il nostro algoritmo consente di fissare offline in maniera semplice l'ordine delle videocamere. Si può quindi pensare che l'ordine delle videocamere sia arbitrario, ma fissato all'inizio.
Oltre ad implementare le relazioni descritte in 6.2.2, il nostro algoritmo risulta molto robusto ai guasti, reagendo in maniera ottima perfino nel caso in cui si rompano 2 telecamere *qualsiasi*. Infatti esso è in grado di ristabilire una nuova rete tra le videocamere "sane" mantenendo l'ordine scelto all'inizio.
Nella tabella che segue sono riportate tutte le situazioni che si possono verificare. In particolare, vengono rappresentati gli "adattamenti della rete" nel caso in cui una o più telecamere non siano connesse alla rete, pensando a questa situazione come un guasto (che potrebbe essere solo un problema di connessione oppure un'effettiva rottura della videocamera). In questo contesto con "adattamenti della rete" si indica il settaggio delle telecamere adiacenti (precedente e successiva) per ogni telecamera che fa parte della rete. Viene riportato 0 quando una videocamera non può avere un precedente o un successivo in base all'ordinamento scelto all'inizio (3-2-1).

	Numero cam connesse	PREV	NEXT
Cam 1	1	0	0
	2	2 connessa: prev = 2 altrimenti: prev = 3	0
	3	2	0
Cam 2	1	0	0
	2	3 connessa: prev = 3 altrimenti prev = 0	1 connessa: next = 1 altrimenti next = 0
	3	3	1
Cam 3	1	0	0
	2	0	2 connessa: next = 2 altrimenti next = 1
	3	0	2

Si possono sottolineare due aspetti:

- se c'è nella rete una sola videocamera a fare patrolling vengono fissati gli estremi di patrolling pari ai suoi limiti fisici, in modo che sorvegli più area possibile;

- se rimangono due telecamere viene creata una "nuova rete" utilizzando le relazioni della sezione 6.2.2 per N=2;
- nel caso rimangano nella rete solo la telecamera 3 e la 1 viene ristabilito l'arco di comunicazione idealmente cancellato (vedere figura 6.3) in modo da poter continuare ad effettuare il patrolling (il ragionamento si può ripetere in maniera analoga se cambia l'ordine iniziale);

E' importante evidenziare che nel caso in cui rimangano connesse alla rete solo due telecamere sia necessario applicare le relazioni di 6.2.2 con una certa attenzione. Infatti non è detto che il vincolo di copertura di cui si è parlato in 6.1.1 sia soddisfatto: dipende dai limiti fisici di ogni telecamera. Nel caso in cui i limiti fisici delle telecamere rimaste attive nella rete non consentano di coprire tutta la traiettoria, gli estremi di patrolling di ogni telecamera vengono fissati a quelli fisici in modo da sorvegliare la maggior area possibile.
Per capire meglio le situazioni in cui rimangono nella rete solo una o due telecamere (secondo sempre il nostro ordine 3-2-1) si riportano i vari casi in maniera schematica nella figura seguente.

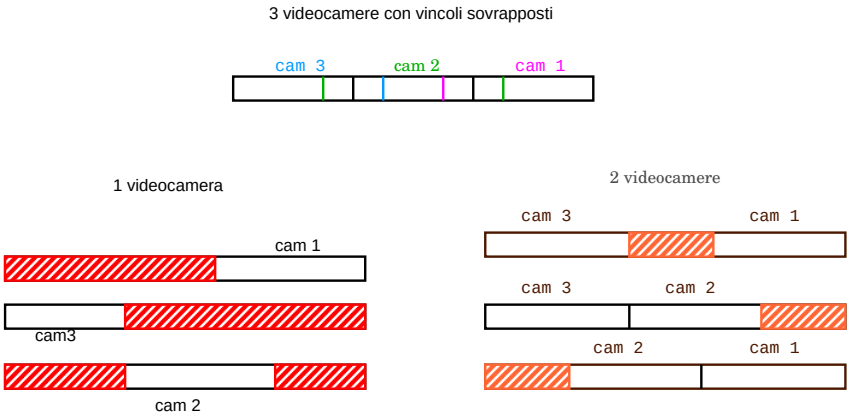


Figura 6.4: In alto sono evidenziati nei vari colori i limiti fisici e nelle figure sottostanti in rosso si evidenziano le aree scoperte.

6.5 Interfaccia grafica

Per rendere più intuitiva la comprensione del funzionamento dell'algoritmo di partitioning implementato, è stata creata un'interfaccia grafica. Si riporta un esempio nella figura seguente (6.5).

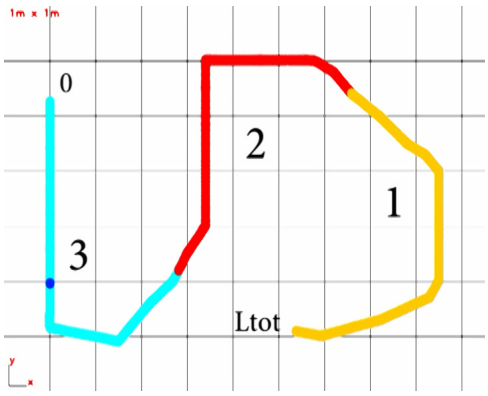


Figura 6.5: Interfaccia grafica per l'algoritmo di partitioning

La (6.5) rappresenta una vista dall'alto della stanza del Navlab (vedere anche figura 4.2) su cui è riportata una griglia in cui ogni quadrato rappresenta $1m^2$. Nell'angolo in basso a sinistra si possono notare gli assi x e y del sistema di riferimento globale della stanza.
La curva colorata è la proiezione della spline scelta (vedere figure 2.3 e 2.4) sul pavimento: la parte turchese indica la zona di competenza della prima telecamera, la rossa quella della seconda e la gialla quella della terza.
Il "cerchietto blu" indica la posizione attuale della videocamera su cui si sta guardando il video. In questo caso il punto blu si muoverà nella zona rossa dato che l'immagine è relativa alla seconda videocamera.
Più precisamente il "cerchietto blu" rappresenta l'ultimo punto in cui ha agito il controllo sulla spline: in questo modo si riesce comunque ad avere un'ottima idea su dove sta guardando la videocamera in questione.
Per realizzare l'interfaccia sono state utilizzate la funzione `valutazionespline(s)` descritta nella sezione 2.4 e una funzione C++ "già pronta" in grado di disegnare un punto, conoscendo le sue coordinate x e y, nella finestra riportata in figura 6.5.
In sostanza, note le aree di competenza di ciascuna videocamera in ascissa curvilinea, si possono calcolare le coordinate di un certo numero di punti tra gli estremi di patrolling e "passarli" alla funzione per l'interfaccia. Il numero di punti è stato scelto in modo da non rallentare troppo l'algoritmo (che dovrebbe "disegnarne" troppi), cercando allo stesso tempo di ottenere una curva piuttosto continua.
Ogni volta che la videocamera raggiunge o passa vicino ad un punto di controllo, vengono valutate le sue coordinate così da disegnarlo nella finestra riportata in 6.2 (cerchietto blu), seguendo lo stesso procedimento riportato sopra.
Nel caso in cui ci siano aree scoperte, vengono "colorate" di nero nella curva. Si riporta un esempio in figura (6.6):

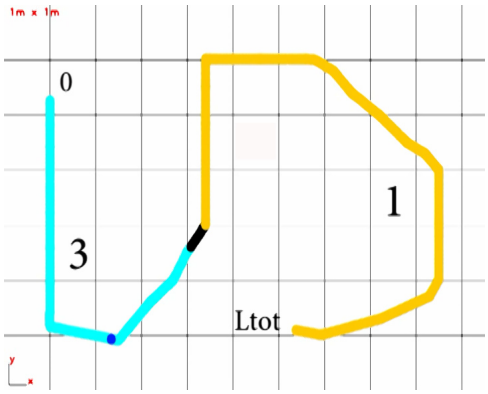


Figura 6.6: Esempio vincolo di copertura non soddisfatto

6.6 Test sperimentali

In questa sezione è riportato un esempio di convergenza degli estremi di patrolling. Di seguito (figura 6.7) si mostrano i limiti fisici imposti alle nostre videocamere e la situazione di regime a cui si devono portare le telecamere. Infatti, dal momento che le velocità delle ascissa curvilinee relative ad ogni videocamera sono le stesse ed è soddisfatto il vincolo di copertura (vedi 6.1.1), la situazione di convergenza è la suddivisione in 3 parti uguali della traiettoria. Nella figura (6.8) è raffigurato un esempio di alcune iterazioni dell'algoritmo di

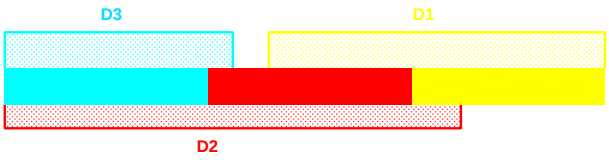


Figura 6.7: Esempio convergenza partitioning dove: D_1(mm)=[9000 24000], D_2(mm)=[0 18000], D_3(mm)=[0 8500]

partitioning quando sono presenti nella rete tutte e tre le videocamere.
La (6.8) è costituita da una serie di tabelle, ciascuna delle quali rappresenta lo *Stato Attuale* della videocamera 3 del Navlab. La prima colonna indica il numero della videocamera (secondo la numerazione del Navlab, vedere 4.2), la seconda se la telecamera è connessa, la terza l'azione svolta dalla telecamera (2 è il patrolling) e le ultime due gli estremi di patrolling ($a_{i,l}(\cdot)$ e $a_{i,r}(\cdot)$).
Queste tabelle vengono visualizzate durante l'esecuzione dell'algoritmo di partitioning. Dalla figura (6.8) si osserva come, dopo una serie di iterazioni, si arrivi alla situazione di convergenza che corrisponde alla suddivisione della traiettoria (lunga $24m$) in tre parti uguali come spiegato sopra.

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	14250	24000
2	1	2	8125	14250
3	1	2	0	8125

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16062.5	24000
2	1	2	7125	16062.5
3	1	2	0	7125

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15562.5	24000
2	1	2	8031.25	15562.5
3	1	2	0	8031.25

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16015.6	24000
2	1	2	7781.25	16015.6
3	1	2	0	7781.25

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15890.6	24000
2	1	2	8007.81	15890.6
3	1	2	0	8007.81

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16003.9	24000
2	1	2	7945.31	16003.9
3	1	2	0	7945.31

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15972.7	24000
2	1	2	8001.95	15972.7
3	1	2	0	8001.95

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16001	24000
2	1	2	7986.33	16001
3	1	2	0	7986.33

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15993.2	24000
2	1	2	8000.49	15993.2
3	1	2	0	8000.49

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16000.2	24000
2	1	2	7996.58	16000.2
3	1	2	0	7996.58

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15998.3	24000
2	1	2	8000.12	15998.3
3	1	2	0	8000.12

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16000.1	24000
2	1	2	7999.15	16000.1
3	1	2	0	7999.15

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15998.3	24000
2	1	2	8000.03	15999.6
3	1	2	0	8000.03

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16000	24000
2	1	2	7999.79	16000
3	1	2	0	7999.79

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15999.9	24000
2	1	2	8000.01	15999.9
3	1	2	0	8000.01

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16000	24000
2	1	2	7999.95	16000
3	1	2	0	7999.95

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	15999.9	24000
2	1	2	8000	15999.9
3	1	2	0	8000

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16000	24000
2	1	2	7999.99	16000
3	1	2	0	7999.99

Cam numb	Connected	Status	Left Bound	Right Bound
1	1	2	16000	24000
2	1	2	8000	16000
3	1	2	0	8000

Figura 6.8: Esempio iterazioni partitioning

Capitolo 7

Tracking

Oltre a un buon sistema di controllo per il pattugliamento di un determinato perimetro, nei sistemi di videosorveglianza è molto importante anche il *tracking* di un evento. Il sistema deve infatti essere in grado di accorgersi dell'evento all'interno della zona che sta controllando e di inseguirlo nel modo più efficiente possibile.
In questo lavoro si è cercato di mantenere le distanze dall'ambito della visione e di trattare il tracking solo come un problema di inseguimento di un oggetto. In sostanza, si è cercato di implementare una tecnica, utilizzando un algoritmo di tracking "già pronto", per controllare le videocamere in modo da poter sempre inquadrare al centro del monitor l'oggetto identificato e seguirlo anche quando si sposta velocemente e tende ad uscire dal field of view.
L'algoritmo che si è scelto per il tracking è quello proposto in [13]: esso è adatto a telecamere mobili, è in grado di inseguire oggetti qualsiasi, è robusto a cambi di posa ed occlusioni ed è in grado di effettuare una ricerca dell'oggetto nel field of view qualora venga perso. Esso prevede una fase di inizializzazione in cui si seleziona "col mouse" un rettangolo all'interno del quale si trova l'oggetto da inseguire (target). A questo punto l'algoritmo fa in modo di "rimanere agganciato" al target e cioè sposta il rettangolo (tracker) visualizzato nel video in base ai movimenti dell'oggetto. Il tracker può variare le sue dimensioni in seguito a cambi di posa del target.
Potrebbe sembrare limitativo per la nostra applicazione questo tipo di algoritmo. Infatti esso consente di inseguire un qualsiasi oggetto che però deve essere selezionato "manualmente" dall'operatore, mentre l'ideale per un sistema di videosorveglianza intelligente sarebbe di avere a disposizione un tracking automatico.
Tuttavia questo dipende solo dall'algoritmo di tracking: la tecnica da noi proposta in questo capitolo per il controllo della videocamera in base ai movimenti del target può essere utilizzata con un qualsiasi altro algoritmo. Basterebbe quindi un approfondimento nell'ambito della visione per implementare un algoritmo di tracking automatico. Questo aspetto però va al di fuori degli obbiettivi di questo lavoro. Infatti, in questo progetto si è interessati più al controllo della videocamera in base alla dinamica del target e all'interfacciamento dell'algoritmo di tracking con quello di patrolling esposto nei capitoli precedenti, piuttosto che alla realizzazione di un tracking automatico.
Dalla figura (7.1) si può vedere che è stato selezionato come target una palla

rossa. L'algoritmo di tracking provvede a inseguire il movimento della palla nel field of view ma non provvede a muovere la videocamera, funzionalità necessaria per poter seguire il target in un qualsiasi punto.
In figura è riportato anche (in arancio) il sistema di riferimento scelto per il piano immagine.

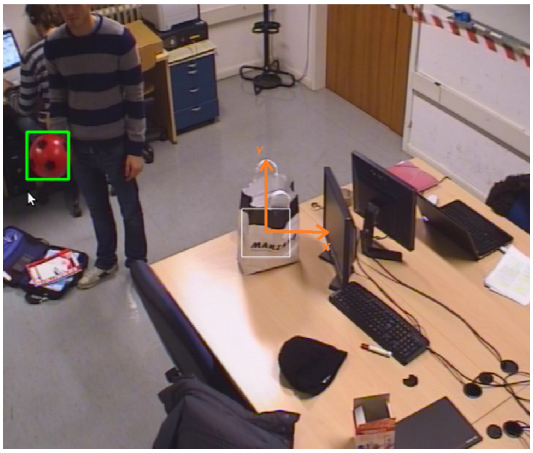


Figura 7.1: Tracking

7.1 Inseguimento oggetto

In questa sezione verrano esposti due semplici metodi di controllo che consentono alla videocamera di inseguire il target e cercare di mantenerlo al centro del monitor. La videocamera infatti deve essere in grado di muoversi in maniera coerente agli spostamenti del target, in modo da non perderlo quando tende a uscire dal field of view in cui è stato "catturato".
Dal momento che la comunicazione tra telecamera e rispettivo PC avviene tramite porta seriale, è stato deciso di realizzare due processi che operassero in parallelo, uno per il movimento della telecamera e l'altro per l'elaborazione dei frame e la visualizzazione del video per il tracking. Questa scelta è stata dettata dal fatto che la trasmissione seriale introduce dei ritardi non trascurabili e il tracking risultava perennemente in ritardo rispetto ai movimenti reali.
Nelle prossime sezioni descriveremo le tecniche utilizzate per l'inseguimento del target.

7.1.1 Primo metodo

Il primo metodo che è stato implementato è basato sulle funzioni di cinematica del capitolo 4.
Dal momento che l'algoritmo di tracking fornisce le coordinate in pixel del centro del rettangolo in cui si trova il target, si vuole cercare di muovere la telecamera in modo che il centro del monitor coincida (o sia perlomeno vicino) a quello del target.
Il metodo più naturale è quello di utilizzare le funzioni di cinematica diretta e cinematica inversa di cui si è parlato nelle sezioni 4.3 e 4.4.

E' stata quindi implementata una nuova funzione, chiamata

$$\texttt{move2pixel}(x_{pix},y_{pix},\texttt{pan},\texttt{tilt})$$

che accetta come parametri di ingresso le coordinate in pixel del centro del target (rispetto al centro dell'immagine) e gli angoli di pan e tilt in cui si trova la videocamera (che si ricavano con la funzione get che ritorna i valori letti dagli encoder). La funzione opera come segue:

- calcola tramite la funzione cinematica diretta della sezione 4.3 le coordinate rispetto al sistema di riferimento globale della proiezione del centro del target sul pavimento (vedere figura 4.4). Nelle formule vengono utilizzati come θ e ψ gli angoli di pan e tilt in cui si trova la videocamera
- ricava gli angoli di pan e tilt in cui si deve portare tramite la funzione cinematica inversa di cui si è parlato nella sezione 4.4
- muove la telecamera negli angoli di pan e tilt ricavati

Questa funzione verrà poi richiamata ripetutamente a seconda dei cambiamenti delle coordinate in pixel del centro del target. Ogni volta che la funzione viene richiamata è necessario utilizzare i valori di pan e tilt attuali e dunque si devono acquisire i relativi valori dagli encoder.
E' stata prevista anche un'area sufficientemente grande intorno al centro del monitor (quadrato bianco in figura 7.1 e 7.2), che indicheremo come *saturation box*, in cui il controllo viene disabilitato e la telecamera sta ferma. In questo modo, si evita che quando il target si ferma la videocamera continui ad eseguire piccoli movimenti correttivi intorno al centro del target. Questi sono dovuti all'offset del punto principale rispetto al centro del monitor (vedere capitolo 4) e alle imprecisioni della cinematica. Infatti, per quanto sia accurata la cinematica, la telecamera si muove in modo da portare il punto principale, che in generale non si trova perfettamente al centro dell'immagine, a coincidere col centro del target: è quindi necessaria la *saturation box* per evitare di continuare a "saltare" in posizioni diverse intorno al centro del target. Si riporta di seguito una figura che rappresenta il movimento della telecamera in modo da centrare il monitor sul target.

Figura 7.2: Tracking + controllo videocamera in posizione

Questa soluzione presenta tuttavia qualche difetto:

- ad ogni passo di campionamento è necessario accedere ai valori degli angoli misurati dagli encoder introducendo dei ritardi
- il controllo del movimento della telecamera è in posizione e risulta molto "scattoso"
- si ha un controllo che dipende dalla cinematica e quindi dalle sue imprecisioni (vedere capitolo 4)
- la videocamera porta il centro del monitor in un punto generico della *saturation box* e quindi in genere non perfettamente al centro

Si è deciso quindi di abbandonare questa soluzione per una concettualmente molto semplice ma efficace.

7.1.2 Secondo metodo

Dal momento che risulta veramente fastidioso vedere la videocamera che si muove "a scatti" nel tracking, perchè non utilizzare anche in questo caso un controllo in velocità?

E' stata quindi implementata una soluzione abbastanza simile a quella seguita nella sezione 5.1. Note le coordinate in pixel del centro del target (x_{pix} e y_{pix}) le velocità di pan e tilt imposte alla videocamera vengono calcolate come:

$$v_{pan} = K_{pan} \cdot (x_{centro} - x_{pix}) = -K_{pan} \cdot x_{pix}$$

$$v_{tilt} = K_{pan} \cdot (y_{centro} - y_{pix}) = -K_{tilt} \cdot y_{pix}$$

dove x_{centro} e y_{centro}, coordinate del centro dell'immagine, sono entrambe nulle dato che il sistema di riferimento scelto per il piano immagine è proprio al centro dell'immagine. K_{pan} e K_{tilt} sono guadagni (positivi) per le velocità di pan e tilt scelti dopo qualche prova sperimentale.
Con questo tipo di controllo si guida la telecamera verso il centro del target tanto più velocemente quanto maggiore è lo scostamento del target dal centro dell'immagine (vedere figura 7.3: la videocamera si muove nel verso della freccia gialla). Inoltre in questo modo le velocità di pan e tilt rispettano le convenzioni

Figura 7.3: Tracking + controllo videocamera in velocità

scelte. Infatti supponiamo di essere nella situazione di figura (7.3): col controllo proposto si applica una velocità negativa sia di pan e che di tilt che è in accordo con la nostra convenzione secondo cui lo spostamento positivo di pan è verso sinistra e quello di tilt verso il basso.
Qualora il target si fermi, il controllo proposto consente inoltre di muovere il centro del monitor verso quello del target, riducendo progressivamente i valori di velocità mano a mano che ci si avvicina fino a fermarsi quando i due punti coincidono perfettamente. In questo modo la telecamera rimane ferma ad inquadrare il target al centro del monitor quando esso è fermo.
Questo controllo che a prima vista risulta banale si è rivelato molto efficace:

- non si deve più accedere alla seriale per acquisire gli angoli di pan e tilt attuali e quindi si introducono meno ritardi
- la telecamera riceve un comando di velocità ad intervalli regolari (400 ms) senza mai fermarsi quando il target è in movimento. In questo modo vengono eliminati gli "scatti" del metodo precedente e il movimento risulta fluido
- il metodo non risente delle imprecisioni della cinematica
- la videocamera varia la propria velocità in base a quella del target mentre nel primo metodo era fissata nel comando per il movimento in posizione
- quando il target è fermo, non è necessario definire una regione all'interno della quale si disabilita il controllo come nel primo metodo. Infatti, come osservato sopra, il secondo metodo consente di fermare la videocamera con il target inquadrato al centro dell'immagine qualora esso sia fermo.

Si riporta nelle figure seguenti (figura 7.4) una sequenza di immagini relative all'inseguimento della palla rossa quando ci si muove "da un estremo all'altro" del laboratorio Navlab del nostro dipartimento.

7.2 Tracking e patrolling

A questo punto resta da interfacciare la parte di gestione del tracking di un evento con quella di patrolling illustrata nei capitoli precedenti. E' evidente che nei sistemi di videosorveglianza sia più prioritario l'inseguimento di un evento piuttosto che lasciare le videocamere nello stato di pattugliamento. In sostanza, quando una delle videocamere rileva un evento anomalo, essa deve abbandonare il compito di controllo della zona assegnata e dedicarsi completamente all'inseguimento dell'evento.
Le altre videocamere, rimaste nello stato di patrolling, devono dividersi nuovamente le zone da pattugliare considerando, nel rispetto dei limiti fisici imposti, anche quella della videocamera impegnata nel tracking. Questa strategia deve potersi ripetere anche nel caso di più videocamere impegnate contemporaneamente nel tracking dello stesso o di diversi target.
Lo stato di tracking è trattato in un modo simile ad un guasto (vedere capitolo 6): quando una o più telecamere è in tracking le altre se ne accorgono e cercano di dividersi in modo ottimo il perimetro da controllare.
Infine, se il tracker perde il target, dopo un certo timeout (da noi fissato a $5s$) la videocamera torna in patrolling.

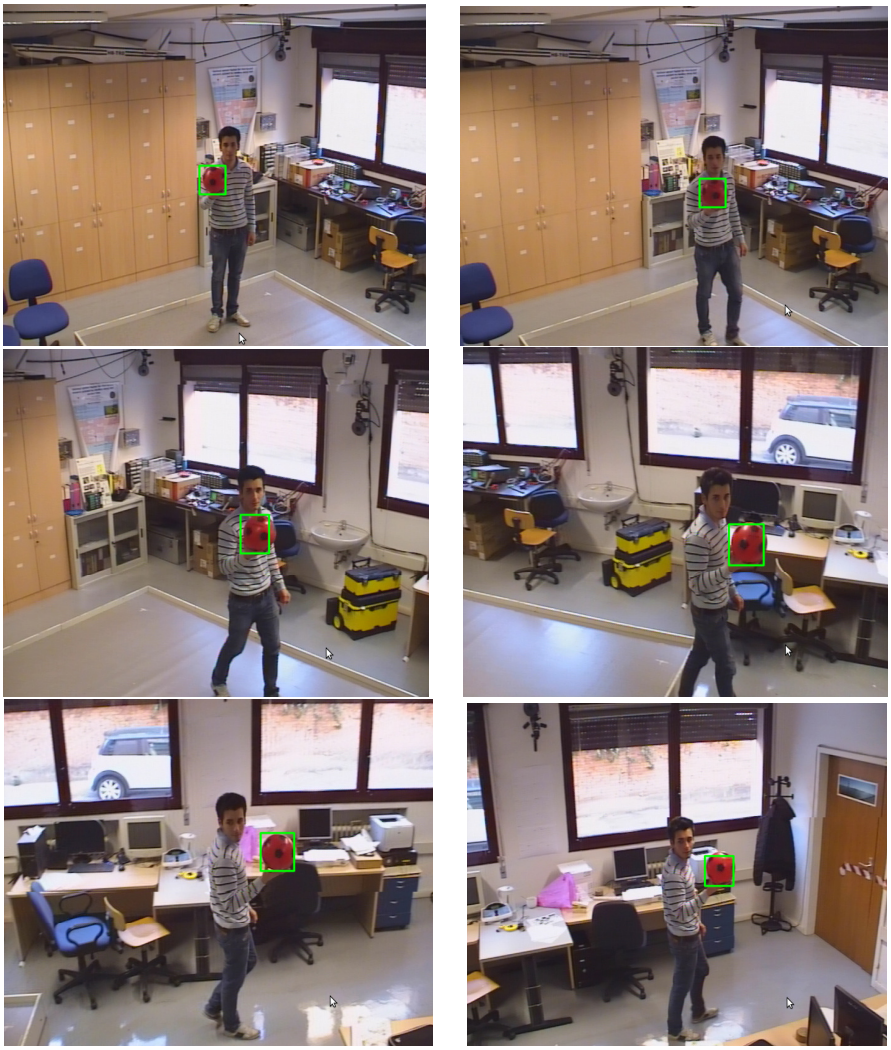


Figura 7.4: Sequenza di immagini tracking

Capitolo 8

Conclusioni e sviluppi futuri

8.1 Conclusioni

L'obbiettivo di questo progetto era quello di realizzare un algoritmo efficiente, robusto e scalabile che gestisse il patrolling e il tracking in una rete di videocamere PTZ.
Questo scopo è stato pienamente raggiunto: infatti, il nostro algoritmo è adatto per un numero arbitrario di telecamere, è in grado di pattugliare una traiettoria arbitraria 3D partizionandola in maniera ottima, gestisce guasti di ogni tipo ed effettua il tracking di un target selezionato manualmente. Inoltre esso prevede che, se una o più telecamere si stanno occupando del tracking di un evento, le altre gestiscano in maniera ottima le zone lasciate scoperte.
Tuttavia esso soffre ancora di alcune limitazioni. In particolare è necessario eseguire alcune procedure offline prima di poter utilizzare effettivamente l'algoritmo: la creazione della traiettoria, la calibrazione e le misure per ricavare i parametri del modello meccanico della videocamera possono risultare troppo onerose per l'utente finale o anche per chi esegue l'installazione del sistema di videosorveglianza. Inoltre, la selezione del target manuale non è adatta nel caso in cui si voglia rendere il sistema completamente automatizzato, anche se un tale sistema potrebbe risultare meno flessibile.
Riteniamo quindi opportuno proporre alcuni miglioramenti e possibili sviluppi futuri del nostro sistema, per far fronte alle limitazioni ancora presenti.

8.2 Proposte e Sviluppi futuri

Il nostro lavoro consta, come accenato nella sezione precedente, di una serie di procedure manuali da dover eseguire, soprattutto in fase di messa in servizio, per permettere al sistema di videosorveglianza di eseguire correttamente le sue mansioni.
Un possibile sviluppo futuro interessante sarebbe quello di creare un pacchetto software ben integrato con l'hardware a disposizione che permetta all'utente finale di installare autonomamente, o comunque con un minimo aiuto, l'apparato di sorveglianza e soprattutto sia in grado, una volta messa in funzione la struttura, di poterne controllare alcune variabili fondamentali come:

- la velocità di movimentazione delle camere oppure il tempo di patrolling;

- la forma del perimetro e dell'area da sorvegliare;
- la possibilità di discriminare eventi od oggetti di diversa tipologia a seconda dell'applicazione desiderata.

Il tutto ovviamente dovrebbe essere offerto nella maniera più trasparente possibile, ovvero mettendo nelle mani dell'utente un unico applicativo software da quale, tramite interfaccia grafica, interagire con il sistema.
Quanto da noi sviluppato nel corso di questo progetto non arriva a tanto, ma rappresenta un buon punto di partenza per i singoli pezzi da mettere insieme per dar vita ad un tale strumento. Vogliamo cercare di dare uno spunto per tale progetto.
Calibrazione delle videocamere: tramite l'uso di librerie *OpenCv*, che prendono spunto dal toolbox MATLAB di Bouguet descritto in [7], è possibile effettuare una procedura semiautomatica di calibrazione. Infatti gli strumenti offerti dalle librerie prevedono che le immagini necessarie per la calibrazione vengano scattate dall'utente tramite la scacchiera, ma la procedura affrontata manualmente nel toolbox che equivale alla vera e propria estrazione dei parametri intrinseci della camera viene eseguita automaticamente.

Modello di camere comprendenti della distorsione: nella nostra trattazione abbiamo supposto di avere a che fare con modelli delle videocamere PTZ abbastanza semplificati ed in particolare con un modello *pinhole*. Considerare anche la distorsione della videocamera può risultare estremamente utile nel caso si desiderino prestazioni ad alta precisione come suggerisce [8] in cui viene spiegato un metodo per ottenere i risultati desiderati.

Definizione on-line del path di patrolling: un utile caratteristica sarebbe quella di poter permettere all'utente di inserire, tramite interfaccia grafica, i punti di controllo che definiscono la traiettoria di patrol.
Nel nostro lavoro è stata implementata una soluzione statica: la traiettoria è nota alle telecamera essendo salvata in un'apposita area di memoria. Tale curva potrebbe in ogni caso essere facilmente modificata accedendo all'area di memoria in questione, impostando le coordinate dei punti di controllo desiderati.
Dal momento che tale soluzione pregiudica molto la flessibilità nell'uso dello strumento si potrebbe pensare ad una soluzione più maneggevole che consenta proprio di inserire i punti di controllo tramite *point and click* all'interno di una semplice ricostruzione schematica dell'area da videosorvegliare.

Cambiamento delle velocità di movimentazione delle camera e dei limiti fisici: nel corso del lavoro è stato assunto che le videocamere potessero muoversi tutte alla stessa velocità (dal momento che sono identiche) ed inoltre sono stati imposti dei particolari limiti fisici per dimostrare la convergenza e il partitioning da parte dell'algoritmo implementato. Nella realtà ci si potrebbe imbattere in limiti fisici che variano oppure in un set di videocamere di diversa tipologia. Oppure si potrebbe semplicemente richiedere che una particolare area venga sorvegliata in un tempo maggiore/minore o ad una velocità diversa. L'algoritmo prevede già questa seconda alternativa dal momento che permette alle videocamere di scambiarsi, in maniera distribuita, i valori di velocità di percorrenza della curva. Per quanto riguarda la modifica dei limiti fisici invece basterebbe permettere l'accesso all'area di memoria nella quale tali limiti ven-

gono imposti/verificati.

Controllo di zoom in velocità: è stato sviluppato un controllo di zoom che consente di visualizzare nel monitor a dimensione costante un determinato oggetto che cambia la sua distanza dalla videocamera. Dal momento che il controllo è in posizione lo zoom viene aggiornato "a step" e non in maniera graduale. Visivamente un tale tipo di controllo si traduce in uno zoom discontinuo e a salti. Risulterebbe molto più piacevole che lo zoom modificasse il suo valore con una certa velocità in modo del tutto equivalente a quanto fatto nel controllo del movimento di pan e tilt delle camere. La mancanza di un comando che ci permettesse di far muovere lo zoom ad una desiderata velocità ci ha impedito di terminare tale implementazione.

Tracking specifico e ricostruzione 3D della posizione di un oggetto: attraverso specifici algoritmi di tracking sarebbe possibile permettere l'individuazione e l'insegumento automatici di oggetti specifici a scelta dell'utente, senza la selezione manuale. Inoltre potrebbe risultare interessante la ricostruzione della posizione 3D dell'oggetto inquadrato. Le singole camere vedono infatti il mondo in 2D e non possono quindi conoscere l'effettiva distanza di un oggetto che non stia sul pavimento o a un'altezza nota. Tramite l'informazione combinata di due o più videocamere potrebbe però essere possibile ricostruirne la posizione corretta.

Bibliografia

[1] Yann Chevaleyre, *Theoretical Analysis of the Multi-agent Patrolling Problem*, Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology(IAT'04)

[2] Baseggio M., Merlo P., Pozzi M., *Coordinazione distribuita di telecamere per patrolling di perimetri e tracking*, relazione progetto anno accademico 2009/10

[3] Tamai G., *Implementation of a coordinated multi-camera perimeter patrolling system*, tesi di laurea magistrale anno accademico 2010/11

[4] D. M. Raimondo, S. Gasparella, D. Sturzenegger, J. Lygeros, M. Morari, *A tracking algorithm for PTZ cameras*, 2010

[5] Wei Liu, Y ongtian Wang, Jing Chen, Junwei Guo,Yongtian Wang *An Efficient Zoom Tracking Method for Pan-Tilt-Zoom Camera*, 2010

[6] Prof. Donald H. House *Spline Curves*, Clemson University, http://www.cs.clemson.edu/ $\sim$ dhouse/courses/405/notes/splines.pdf

[7] Bouguet (2008), *Camera calibration toolbox for Matlab*, http://www.vision.caltech.edu/bouguetj/calib_doc/

[8] Meine liebe Freunde, *A method for patrolling path definition in multi-camera systems*, VideoTec S.p.A.

[9] Fusiello A., *Visione computazionale*, http://profs.sci.univr.it/ fusiello/teaching/visione/appunti/

[10] Bradsky G., Kaehler A., *Learning OpenCV*, O'Reilly, 1st edition

[11] Derek Kingston, Ryan Holt, Randal Beardy, Timothy McLain, and David Casbeer, *Decentralized Perimeter Surveillance Using a Team of UAVs*, Brigham Young University, Provo, UT, 84602

[12] Ruggero Carli, Angelo Cenedese, Luca Schenato, *Distributed Partitioning Strategies for Perimeter patrolling*, 2011 American Control Conference on O'Farrell Street, San Francisco, CA, USA June 29 - July 01, 2011

[13] D. Zabotto, *Algoritmo basato su istogrammi di colore per l'inseguimento visivo di oggetti in reti di telecamere*, tesi di laurea magistrale anno accademico 2010/2011

[14] Pankaj Kumar, Anthony Dick, Tan Soo Sheng, *Real Time Target Tracking with Pan Tilt Zoom Camera*, 2009 Digital Image Computing: Techniques and Applications

[15] AXIS Communication, *Guida tecnica ai sistemi video di rete*, http://www.axis.com/

[16] *Review of existing smart video surveillance systems capable of being integrated with ADDPRIV project*, http://www.addpriv.eu