UNIVERSITÁ DEGLI STUDI DI PADOVA

FACOLTÁ DI INGEGNERIA

Corso di Laurea in Ingegneria dell'Automazione

Tesi di Laurea Specialistica

TRACKING PTZ IN TEMPO REALE PER VIDEOSORVEGLIANZA

Relatore: Prof. Emanuele Menegatti

Correlatore: Ing. Giambattista Gennari

Laureando: Bruno Lain

Anno Accademico 2009/2010

a Laura
e alla mia famiglia

Indice

Sommario

Il lavoro di tesi si inserisce in un progetto il cui scopo è la realizzazione di un sistema di tracking con telecamera PTZ. L'inseguimento automatico di un soggetto è una caratteristica interessante per un sistema di videosorveglianza e il suo scopo è quello di coadiuvare il lavoro delle persone addette al monitoraggio di zone ad accesso limitato. Il progetto, svolto in collaborazione con Videotec S.p.A, è partito dalla costruzione di un data set di video rappresentanti gli scenari di interesse per l'azienda. I video (organizzati in 8 categorie, per un totale di più di 23000 frame) sono abbinati ad un file contenente le coordinate del soggetto che si desidera inseguire, in modo da poter facilmente verificare l'efficacia del tracker. Si è implementato e messo a punto con soluzioni innovative un algoritmo di tracking basato su istogrammi e si è verificato il notevole miglioramento in prestazioni ottenuto rispetto ad un secondo algoritmo già sviluppato dall'azienda. Il funzionamento dell'algoritmo è stato ampiamente verificato in prove in tempo reale con la telecamera PTZ e sono stati sfruttati i vantaggi ottenibili dal controllo diretto del movimento della camera e dello zoom. Il particolare sistema di controllo della telecamera PTZ progettato consente di ottenere una ripresa fluida della scena e allo stesso tempo è in grado di operare al fine di favorire il funzionamento del tracker. Si è infine individuato un sistema di rilevazione di drift che segnala la maggioranza dei casi in cui il target viene perso o non è più visibile nella scena, a fronte di un esiguo numero di falsi allarmi.

Capitolo 1

Introduzione

La disponibilità sul mercato di hardware in grado di processare flussi video in tempo reale e la contemporanea riduzione dei prezzi delle telecamere ha consentito negli ultimi anni lo sfruttamento di algoritmi di visione artificiale anche in prodotti industriali. Anche la ricerca nel campo della visione computazionale sta crescendo moltissimo, diffondendosi nei settori più disparati. Contemporaneamente le tecniche più collaudate e affidabili stanno trovando grande spazio nel mercato, principalmente nel campo dell'automazione industriale, dell'intrattenimento e nella videosorveglianza.

In quest'ultimo ambito c'è una varietà di applicazioni consistente. Una prima distinzione si può fare tra il monitoraggio di aree interne (quali stazioni, areoporti, musei) ed aree esterne (piazze, perimetri di edifici, porti). Nelle aree affollate è di interesse l'analisi comportamentale delle persone, l'individuazione di persone sospette e il loro inseguimento all'interno dell'area inquadrata dalle telecamere. Altro campo di studi è la localizzazione di bagagli abbandonati.

La videosorveglianza perimetrale si rivolge generalmente a zone poco affollate, magari perchè ad accesso limitato. Basti pensare al controllo del perimetro di un edificio, a porti o a piattaforme offshore. In questo contesto è utile il rilevamento di eventi anomali, come l'ingresso di veicoli o persone nell'area sotto controllo. Con l'aumentare del numero di telecamere in una stessa installazione si fa sempre più sentire l'esigenza di un sistema intelligente di videosorveglianza, che possa aiutare gli operatori nel compito della vigilanza. Ciò è già possibile grazie a risorse hardware dedicate all'acquisizione e all'elaborazione delle grandi moli di dati proveniente da diversi flussi video. Possono essere eseguite in real-time, su hardware equivalente a quello di un personal computer moderno, operazioni quali il riconoscimento di oggetti, la classificazione, l'analisi di eventi e il tracking.

L'applicazione delle tecniche di computer vision alla videosorveglianza permette in primo luogo di far fronte ai cali di concentrazioni di un individuo incaricato di monitorare una zona e di coadiuvarlo nel suo compito. In secondo luogo consente, quando si ha bisogno di rivedere immagini già registrate, di trovare velocemente i momenti in cui è successo qualcosa di anomalo, o eventualmente solo gli eventi della categoria a cui si è interessati (entrata di veicoli, uscita di veicoli, soggetti veloci oppure lenti e così via). In collaborazione con l'algoritmo di event detection, si può pensare di avere una seconda registrazione che inquadri il soggetto di interesse con un angolo di ripresa più

ristretto, in modo da riuscire ad identificarlo e a seguirlo nei suoi movimenti. Questo è lo scopo del progetto di tracking in cui si inserisce il lavoro di tesi. Verranno studiati degli algoritmi adatti allo scopo, tramite simulazioni su un'insieme di video che rappresentano gli scenari d'interesse. Si faranno poi dei test su un sistema reale, composto da una telecamera PTZ (Pan, Tilt, Zoom) controllata in modo da ottenere un inseguimento fluido del soggetto.

Le potenzialità di una disciplina giovane come la computer vision hanno reso la videosorveglianza un terreno fertile per la ricerca scientifica. Ad esempio in [2] viene presentata una panoramica delle più comuni metodologie e delle possibili applicazioni, mentre in [3] viene presentato un possibile sistema di videosorveglianza completo. La visione computazionale si scontra però con l'imprevedibilità dell'ambiente esterno e con la varietà dei comportamenti degli individui. I settori in cui finora ha trovato terreno fertile sono difatti quelli in cui l'ambiente in cui si inserisce la telecamera è noto, così come le sue variazioni. È il caso questo di prodotti per l'assemblaggio di prodotti, del controllo di qualità, dell'intrattenimento. In altri campi la difficoltà a garantire delle prestazioni minime in un ampio scenario di situazioni ostacola tuttora il diffondersi di dispositivi intelligenti.

1.1 Contesto applicativo

Il lavoro di tesi è stato svolto presso l'azienda Videotec S.p.A situata in via Friuli 6, Schio (VI). L'azienda ha un'esperienza ventennale nella produzione di custodie e brandeggi per telecamere. Si è specializzata nella produzione di apparecchi in grado di tollerare condizioni di lavoro estreme, quali temperature elevate, ambienti con presenza di materiali esplosivi ed infiammabili, ampie escursioni termiche.

Di recente l'azienda ha sviluppato un nuovo prodotto hardware e software, chiamato *Albert*, che introduce importanti novità nell'utilizzo delle telecamere per la sorveglianza. Le sue capacità computazionali consentono di automatizzare il sistema di videosorveglianza, rendendolo intelligente, e fornendo quindi assistenza agli operatori. Albert è una piattaforma flessibile, che dispone di capacità computazionali che potranno permettere di eseguire un algoritmo di tracking PTZ in tempo reale.

Al momento le caratteristiche principali di questo strumento si possono riassumere in due punti:

- **Cooperazione tra unità**. Ad ogni telecamera viene connesso un Albert, ciascuno dei quali è a sua volta connesso ad un PC centrale unico controllato dall'operatore. Le varie unità sono collocate su una mappa virtuale che riproduce l'area sottoposta a videosorveglianza. Conoscendo la posizione delle varie unità e l'area inquadrata da ognuna di esse è possibile applicare degli algoritmi di cooperazione tra le camere al fine di massimizzare l'area inquadrata. Con gli algoritmi di cooperazione di cui è dotato Albert le telecamere sono coordinate in maniera ottimale, al fine di massimizzare l'area inquadrata e di garantire la copertura nei punti sensibili.

  È possibile inoltre individuare tutte le camere in grado di inquadrare una stessa zona, in modo da vedere da più punti di vista un soggetto d'interesse che si trovi in quel punto. Si può dunque scegliere di volta in volta la telecamera più adatta.

- **Algoritmi avanzati per la videosorveglianza intelligente**. L'Albert è in grado di acquisire e digitalizzare il flusso video analogico proveniente dalla telecamera. La versatilità dell'hardware permette di implementare algoritmi di visione in grado di funzionare in tempo reale. Attualmente è stato ideato ed implementato un algoritmo di rilevazione

di eventi [5] in grado di analizzare la scena inquadrata da una camera fissa, di crearsi molto velocemente un modello del background e di segnalare successivamente all'operatore eventuali eventi anomali.

Albert è inoltre in grado di registrare il flusso video compresso con il codec H.264. In questo modo è possibile rivedere le scene nelle quali sono stati rilevati eventi anomali, ad esempio in momenti in cui l'operatore non era fisicamente presente davanti ai monitor.

Nello scenario descritto è abbastanza facile immaginare un ampliamento delle funzioni di Albert. Per il mercato sarebbe molto interessante lo sviluppo di un sistema affidabile per l'inseguimento di un determinato soggetto all'interno della scena inquadrata da una telecamera, ovvero quello che in letteratura è chiamato *tracking*. L'obiettivo è l'inseguimento di un target, ovvero il soggetto di interesse nella scena (persona o veicolo). Il tracking avviene grazie ad una singola telecamera PTZ, che si muove fluidamente secondo i due gradi di libertà di cui dispone ed adattando lo zoom alla dimensione del target in modo che sia sempre ben visibile nell'inquadratura. L'algoritmo presentato in [6] è un primo approccio al problema del tracking, inteso come inseguimento a camera fissa. Prima che il target esca dall'immagine la telecamera va a riposizionarsi predicendo la posizione in base a un modello del moto.

Per ottenere un sistema completo e affidabile ci sono ancora molti problemi da affrontare. Indispensabile è l'algoritmo di rilevazione di eventi, che è già disponibile presso l'azienda. In secondo luogo andranno ricercati dei criteri che discriminino i target tra quelli interessanti e quelli che non vale la pena seguire. Un blocco che può essere d'aiuto è un identificatore di target, che permette di distinguere tra persone, automobili, camion o animali. In base a tali informazioni si può pensare di utilizzare diverse impostazioni (ad esempio il moto di un veicolo soggetto a vincoli non olonomi è più facilmente prevedibile rispetto ad una persona, tuttavia può muoversi a velocità maggiori). Il cuore del sistema sono però gli algoritmi di tracking in senso stretto, che è infatti la parte maggiormente studiata in questa tesi. Un blocco altrettanto importante è un algoritmo che sia in grado di decidere quando fermare il tracking e quindi disimpegnare la telecamera dal compito assegnatole. Da non trascurare è il controllo del brandeggio e dello zoom, che devono far sì che il video ripreso dalla telecamera risulti fluido e piacevole alla vista, e ovviamente permetta di leggere maggiori informazioni rispetto a quelle che si avrebbero con una camera fissa.

Infine se si dispone di più telecamere calibrate che inquadrano la stessa zona il tracking può essere notevolmente migliorato grazie alla collaborazione. Si può altresì pensare di cominciare l'inseguimento del target con una telecamera per poi passarlo ad altre quando questo si porti ai limiti dello spazio coperto dalla prima. Un'idea delle potenzialità di questo approccio si può avere in [4].

Con il termine tracking ci si riferisce in definitiva alla capacità di un elaboratore di individuare la posizione di un oggetto da inseguire all'interno di una sequenza d'immagini. Il mondo che ci circonda è però un ambiente dinamico, ricco di oggetti che possono essere animati o no in continuo movimento o in lenta trasformazione. Nell'ambiente vanno incluse anche le condizioni di luce e quelle atmosferiche, le cui variazioni si presentano a volta come processi molto lenti, altre come eventi improvvisi.

Le principali difficoltà che incontra un algoritmo di tracking si possono raggruppare in tre grandi categorie:

- le occlusioni, dovute a qualche oggetto che si interpone tra il target e la telecamera
- i cambiamenti dell'oggetto visti dalla telecamera, dovuti ad una varietà di fattori: cambi di posa, di illuminazione, allontanamento o avvicinamento alla telecamera ecc.
- distrazioni dovute a oggetti in qualche aspetto simili al target.

Gli algoritmi di tracking sono normalmente onerosi dal punto di vista computazionale. Molto spesso le approssimazioni usate per ridurre i tempi di calcolo portano ad un peggioramento delle prestazioni più che proporzionale rispetto al risparmio di cicli di cpu. Il tracking, nella sua accezione più generale, è dunque un problema difficile. In soccorso alla telecamere si può allora pensare di utilizzare altri sensori come laser, radar, sensori a infrarossi. Essi comportano però dei costi aggiuntivi non indifferenti, pertanto non verranno considerati nell'ambito di questo progetto.

La fase di individuazione del target da inseguire e la rilevazione della sua posizione ai fini della costruzione del modello per il tracking non è stata affrontata in questo lavoro, in quanto verrà affidata ad un algoritmo di event detection simile a quello già sviluppato dall'azienda. In seguito si supporrà pertanto di conoscere la posizione del target al primo frame del video.

1.2 Obiettivi

Lo scopo della tesi è l'analisi e l'implementazione di algoritmi di tracking adatti ad essere utilizzati con una telecamera mobile e su hardware limitato. L'analisi avrà il fine di individuare le condizioni nelle quali un determinato algoritmo funziona e quelle in cui, al contrario, non è robusto. Questo permetterà, nei lavori futuri, di studiare dei metodi per la combinazione di tecniche diverse, in modo da irrobustire e prolungare la durata del tracking. Parallelamente sarà utile ricercare degli indicatori di drift o delle metodologie atte a rilevare automaticamente la perdita del target da parte della telecamera. Un terzo punto della tesi sarà la costruzione di un data set di video da utilizzare per le simulazioni e per il confronto tra algoritmi diversi. Ad ogni video dovrà essere allegato un file con la ground truth, ovvero le coordinate del target annotate manualmente. Infine si presenterà un'implementazione su un sistema reale, che dimostrerà l'efficacia delle soluzioni trovate e metterà in luce le problematiche tecniche che andranno affrontate nella messa a punto di un sistema di tracking completo.

1.3 Lavoro di tesi

Il lavoro di tesi si concentra su due punti principali:

- Lo studio e la messa a punto di un algoritmo di tracking.
- Il controllo del brandeggio e dello zoom della camera.

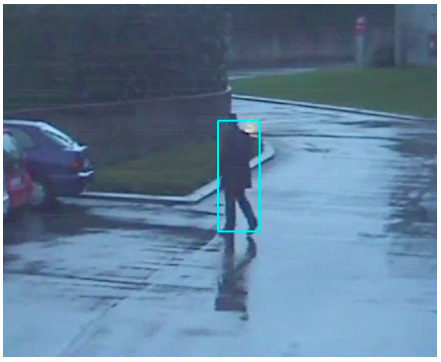
L'azienda dispone, oltre a [6], di un algoritmo di tracking basato sul template matching [7] sviluppato per lo stesso scopo del presente elaborato. Il confronto di diverse tecniche permette di studiare le situazioni dove una tecnica funziona meglio di un altra ed eventualmente di combinarle per riuscire a irrobustire e a prolungare la durata del tracking. In sintesi si può pensare di combinare diversi algoritmi facendoli girare in parallelo o in sequenza. Poi si tratta di fondere i risultati secondo indici che indicano l'affidabilità della stima nella particolare situazione, come presentato in [8] o in [9].

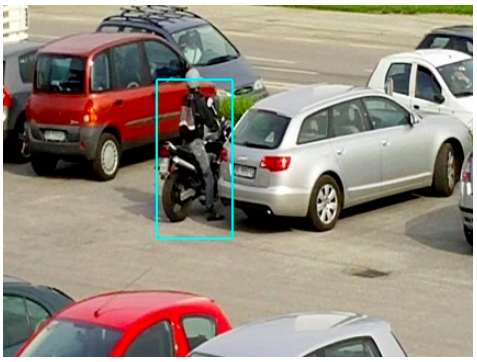
In questo senso lo studio e la messa a punto di diversi algoritmi diventa una necessità di fronte alla difficoltà di garantire buone prestazioni del sistema nel suo complesso. Di cruciale importanza è la raccolta di un numero elevato di video sui quali poter provare e confrontare diversi algoritmi. Tali video devono essere dotati di ground truth, ovvero un file che per ogni frame specifichi le coordinate del bounding box circostante il target che si desidera inseguire. La prima parte del lavoro di tesi è stato pertanto quella della costruzione di un discreto data set di video, come verrà spiegato nei dettagli nel capitolo ad esso dedicato.

Riguardo l'algoritmo di tracking la scelta è stata quella di studiare e mettere a punto la tecnica del mean-shift, in quanto è un approccio abbastanza diverso rispetto a quello utilizzato in [7], diversificando in questo modo gli algoritmi di tracking disponibili.

1.4 Ipotesi di progetto

In questo lavoro si suppone di conoscere la posizione del soggetto da tracciare nel primo frame. Il modello iniziale sarà costruito sulla base di questa informazione. L'obiettivo è quello di inseguire il target nei frame successivi, controllando la telecamera mobile al fine di portarlo e tenerlo in primo piano all'interno dell'inquadratura, possibilmente con movimenti fluidi, per il tempo più lungo possibile.





Le due caratteristiche fondamentali che deve avere l'algoritmo di tracking per essere di interesse nell'applicazione considerata sono:

- *Bassa complessità computazionale*, in modo da consentire l'esecuzione in real-time su hardware dedicato.
- *Scarsa sensibilità ai parametri*, in quanto si chiede di funzionare in molteplici condizioni ambientali senza bisogno di interventi esterni.

1.5 Materiale già presente

Ci sono a disposizione in azienda due algoritmi di tracking già menzionati. Il primo è basato sul *motion-detection*, il secondo sul *template-matching*.

- *Motion-detection*: tale approccio non soffre di problemi di drift, ma dà luogo ad un inseguimento a scatti, in quanto una volta spostata la telecamera è necessario reinizializzare il background con una breve fase di learning. Naturalmente in simulazione funziona solo su video a telecamera fissa.

  Sarà questo il metodo con il quale, nel sistema completo, verrà individuato il target da inseguire, e con il quale si inizierà il tracking. Tracking

che proseguirà in un secondo momento con altri metodi di inseguimento fluido.

- *Template matching*: è un metodo molto efficace in tracking di breve durata. Consente una buona precisione sulla posizione e la scala del target, chiaramente quando esso non è molto diverso dal modello del primo frame. Di contro è poco robusto alle occlusioni e molto sensibile ai cambi di posa del soggetto. L'algoritmo a disposizione sfrutta l'immagine in scala di grigi. Nel prossimo capitolo saranno presentati i concetti di base del suo funzionamento.

Di entrambi si possiede il codice Matlab e una buona documentazione sugli stessi. In particolare il secondo sarà utilizzato a titolo di confronto dell'algoritmo proposto in questa tesi.

Capitolo 2

Due approcci al tracking

Si presentano qui, in sintesi, due approcci al problema del tracking presenti in letteratura, che saranno confrontati nel seguito della tesi. Si è constatato che i lavori che sfruttano l'algoritmo di tracking vero e proprio per muovere una telecamera PTZ sono in numero esiguo rispetto al totale. In questa capitolo ci si concentrerà perciò sull'algoritmo di tracking vero e proprio, argomento affrontato in una moltitudine di articoli.

Va precisato che in tutti i lavori finora pubblicati la durata del tracking in condizioni difficili è sempre ridotta. Le capacità che ha un essere umano di inseguire un soggetto non sono replicabili da un algoritmo software. Oltre ad avere ottime capacità visive, gli uomini (così come gli animali) sfruttano una moltitudine di informazioni a priori sul soggetto, sull'ambiente circostante e presentano una flessibilità difficile da ripetere su un dispositivo elettronico. Una buona parte dei lavori presenti in letteratura sfrutta infatti delle informazioni a priori sul tipo di target da inseguire, come ad esempio mani, facce, veicoli, occhi. Nelle fasi iniziali di questo lavoro si vuole studiare una piattaforma il più possibile flessibile e adattabile ad un vasto campo di scenari, perciò si partirà dall'ipotesi di non conoscere il tipo di target di cui si vuole tener traccia.

In generale tutte le tecniche consistono nella creazione di un modello del target basato su delle *features*. Il modello può rimanere fisso, può venire aggiornato o può essere ricalcolato ad ogni iterazione. La seconda fase di ogni algoritmo è quella che permette di trovare la posizione del target nel nuovo frame (*matching*) in modo da massimizzare una certa misura rispetto al modello del target.

2.1 Template Matching

Storicamente i primi tentativi di tracking sono stati fatti mediante l'utilizzo di un template, ovvero un'immagine dell'oggetto da tracciare. Per ogni nuovo frame acquisito si cerca di individuare la posizione sull'immagine che minimizza una certa funzione. I primi a proporre questa tecnica furono Lucas e Kanade [10].

Le possibili traslazioni, rotazioni e deformazioni del template sono espresse dalla funzione $W(\mathbf{x};\mathbf{p})$ che è la mappatura del pixel $\mathbf{x}$ dal template T al sistema di coordinate dell'immagine I. $W(\mathbf{x};\mathbf{p})$ dipende dai parametri $\mathbf{p} = [p_1 p_2 \dots p_k]$. Il problema del template traking si riduce ad un problema di

minimizzazione nei parametri $\mathbf{p}$ della seguente funzione:

$$\sum_{textbfx} [T(\mathbf{x}) - I_n(W(\mathbf{x};\mathbf{p}))]^2$$

dove la somma è calcolata su tutti i pixel del template T($\mathbf{x}$). La formula precedente è la distanza che si vuole minimizzare ed è chiamata *SSD* (Sum of Squared Differences).

Nello scenario della videosorveglianza si è interessati principalmente alla traslazione del target e al cambio di scala, quindi il vettore $\mathbf{p}$ è composto da 3 parametri: due per la traslazione e uno per la dimensione. Introdurre un quarto parametro (ad esempio per indicare l'orientazione del bounding box) nello specifico aggiunge un grado di libertà in più che va a peggiorare le prestazioni, senza portare informazioni interessanti per l'inseguimento del soggetto. In altri applicazioni l'orientazione è invece un parametro da tenere in considerazione.

La matrice di *warping* $W(\mathbf{x};\mathbf{p})$ porta, nel caso di traslazione e cambio di scala, ad una trasformazione matriciale del tipo:

$$\begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+p_3 & 0 & p_1 \\ 0 & 1+p_3 & p_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

L'algoritmo procede nel seguente modo:

- Trovare, usando un qualunque metodo per la minimizzazione di funzioni quadratiche, $\Delta\mathbf{p}$ che minimizzi:

$$\sum_{\mathbf{x}} [T(\mathbf{x}) - I_n(W(\mathbf{x};\mathbf{p}+\Delta\mathbf{p}))]^2$$

- Aggiornare il vettore $\mathbf{p}$:

$$\mathbf{p} + \Delta\mathbf{p} \rightarrow \mathbf{p}$$

- Eseguire i due passi precedenti finché $\mathbf{p}$ non converge o fino a quando non si raggiunge un numero massimo di iterazioni prefissato.

Al fine di migliorare la complessità computazionale dell'algoritmo sono state proposte molte estensioni, ad esempio cambiando la funzione di warping o introducendo delle approssimazioni nel calcolo del minimo.

2.2 Tracking basato su istogrammi

Un'ampia parte degli studi sul tracking sono dedicate alle tecniche che si basano su una descrizione della regione dell'immagine in cui il soggetto si trova. In genere queste usano come modello un'istogramma, ad esempio l'istogramma delle orientazioni degli edge o delle texture, ma quello che ha dato ottimi risultati è l'istogramma del colore. L'istogramma dà una descrizione statistica globale del soggetto da inseguire, perdendo così ogni informazione spaziale. L'istogramma inoltre è veloce da calcolare per un algoritmo che deve girare in tempo reale.

Come è facilmente intuibile un tracker basata su questo tipo di feature sarà molto resistente alle deformazioni o ai movimenti del target, mentre non potrà distinguere tra due oggetti con stesso aspetto globale ma diversa composizione spaziale. Un possibile approccio è quello presentato in [17]. Il metodo che tuttavia ha reso interessante il tracking basato su istogrammi è il *Mean Shift*. Si tratta di un metodo di stima non parametrica che consente di trovare un massimo locale di una densità di probabilità dati alcuni campioni della stessa (l'istogramma nel caso specifico). Il metodo, introdotto nel 1975 in [14], è molto flessibile ed ha trovato numerose applicazioni nel campo dell'elaborazione delle immagini come nel clustering e nel visual tracking. Il mean-shift tracking verrà trattato in dettaglio nel prossimo capitolo.

Capitolo 3

Mean shift tracking

In questo lavoro di tesi si è implementato un algoritmo di tracking basato su istogrammi. La tecnica utilizzata è quella del *mean shift*, originariamente proposta da Comaniciu [12]. La densità di probabilità del soggetto (ovvero l'istogramma) è stimata utilizzando un kernel monotono decrescente. Il kernel deve soddisfare alcune proprietà generali, come discusso in [14] e [15].

È conveniente utilizzare il kernel di Epanechnikov, definito come:

$$K_E(x) = \frac{1}{2}c_d^{-1}(d+2)(1+||x||^2) \qquad \text{se } ||x|| < 1, \text{ 0 altrimenti}$$

dove c_d^{-1} è il volume della sfera unitaria d-dimensionale e x indica le coordinate normalizzate dei pixel del target, riferite al suo centro.

Un frame di un video è un'immagine a 2 dimensioni, pertanto il kernel di Epanechnikov utilizzato è:

$$K_E(x) = \frac{2}{\pi}(1-||x||^2) \qquad \text{se } ||x|| < 1, \text{ 0 altrimenti}$$

Come distanza da minimizzare è stata scelta in [12] quella di Bhattacharyya. Per stimare la posizione del target si cerca il massimo del coefficiente di Bhattacharyya, così definito:

$$\rho[p(\mathbf{y}), q] = \sum_z \sqrt{p_z(\mathbf{y})q_z}$$

dove con q_z si è indicato l'istogramma del target e con p_z l'istogramma della porzione d'immagine candidata.

La stima di densità con kernel $K(x)$ e raggio della finestra h, calcolata in un punto $\mathbf{x}$ dell'immagine è data da:

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\frac{1}{h^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

Indichiamo con $\hat{q}_u$ e $\hat{p}_u$ i valori del bin u dell'istogramma del target e del candidato rispettivamente. Con $\hat{\mathbf{y}}_0$ indichiamo la posizione corrente del centro del candidato.

Si dimostra che massimizzare il coefficiente di Bhataccharyya equivale a massimizzare il termine

$$\sum_{i=1}^{n} \omega_i K\left(\left|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right|^2\right)$$

dove h è un parametro che indica la larghezza di banda del kernel, e gli ω_i sono dati dalla formula:

$$\omega_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \delta[b(\mathbf{x}_i) - u]$$

dove δ è la funzione di Kronecker, ovvero vale 1 nel bin u dell'istogramma.

Il vettore traslazione che viene calcolato ad ogni iterazione è chiamato *mean shift* ed indica la direzione che dà con maggior probabilità la posizione del target nel frame corrente. Esso viene calcolato come:

$$\hat{\mathbf{y}} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \omega_i g\left(\left|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right|^2\right)}{\sum_{i=1}^{n} \omega_i g\left(\left|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right|^2\right)}$$

dove con $g()$ si è indicata la funzione derivata. Dal momento che la derivata del kernel di Epanechnikov è costante, l'espressione si semplifica notevolmente, riducendosi al rapporto che è la distanza media pesata in base ai coefficienti ω_i:

$$\hat{\mathbf{y}} = \frac{\sum_{i=1}^{n} \mathbf{x}_i \omega_i}{\sum_{i=1}^{n} \omega_i}$$

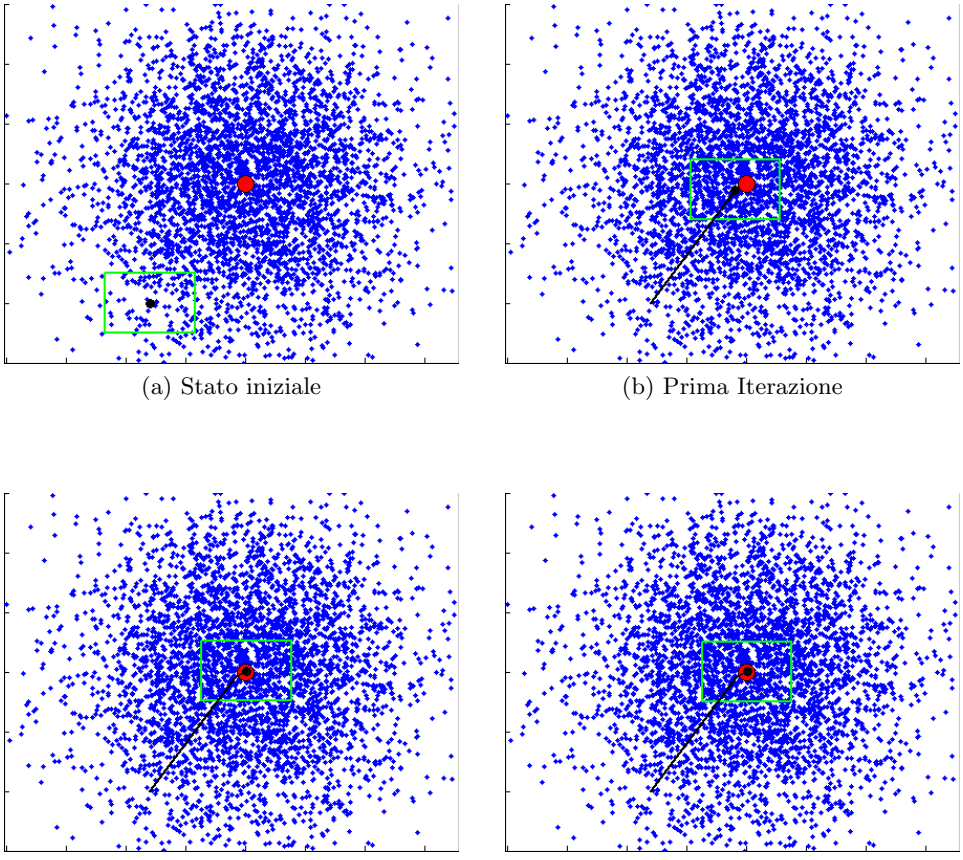
Per ulteriori dettagli si rimanda a [12] e [13], dove si dimostra che il vettore mean shift ha sempre la stessa direzione del massimo incremento di densità. È possibile perciò, attraverso più iterazioni, portarsi in un massimo locale. La proprietà interessante che ha questo vettore è che ha modulo grande quando $\mathbf{x}$ è lontano dal massimo locale, e decresce mano a mano che ci si avvicina. Questa è una condizione necessaria per la convergenza dell'algoritmo. Per la dimostrazione della convergenza dell'algoritmo si rimanda a [15].

La procedura del mean shift, che porta ad un massimo locale a partire da un punto $\mathbf{x}$ può essere così schematizzata:

1. Calcolo del vettore Mean Shift nel punto $\mathbf{x}$.
2. Traslazione della finestra di ricerca della quantità data dal vettore Mean Shift.
3. Se il vettore Mean Shift ha modulo minore di una soglia prefissata l'algoritmo termina, altrimenti si ritorna al punto 1.

3.1 Esempio su una distribuzione di punti casuale

Si riporta un esempio grafico di come funziona l'algoritmo su dei campioni di una distribuzione normale, con una finestra di ricerca di dimensioni fissate arbitrariamente, così come il punto iniziale. Si può vedere che già alla prima iterazione il vettore mean shift consente di avvicinarsi notevolmente al punto di massima densità (ovvero la media della distribuzione, segnalata con un cerchio al centro della figura). Alla seconda iterazione si è già sufficientemente vicini al punto desiderato. Nella terza iterazione il vettore mean shift ha modulo molto piccolo, pertanto la procedura termina.



(a) Stato iniziale

(b) Prima Iterazione

(a) Seconda Iterazione

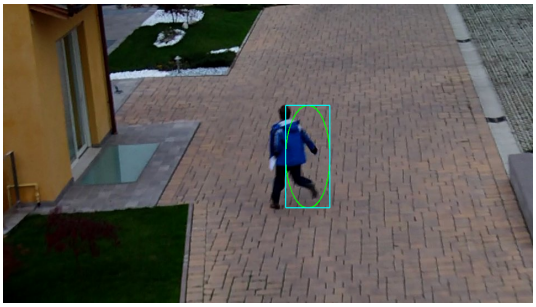
(b) Terza Iterazione e terminazione dell'algoritmo

Figura 3.1: Esempio di funzionamento della procedura del mean shift

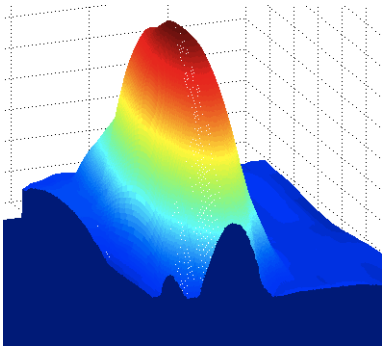
3.2 Esempio su un frame

Prima di addentrarsi nei dettagli implementativi, si propone un'esempio di mean shift applicato ad un'immagine. Si tratta di un frame di un video, e il target che si vuole individuare è la persona. Il modello è stato ricavato manualmente sull'immagine della persona ripresa 5 frame prima.

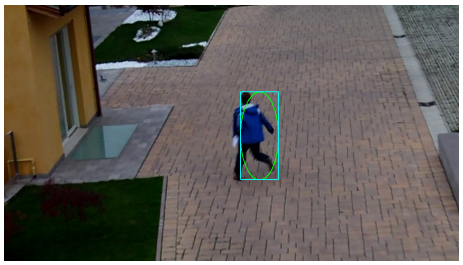
Innanzitutto vediamo in figura (b) come varia il coefficiente di Bhatacharyya sull'immagine (in un intorno della persona). Si nota come il picco esiste ed è in corrispondenza del target nel nuovo frame, pertanto la ricerca che stiamo eseguendo ha senso. Anche qui, come nell'esempio precedente, la convergenza è molto veloce, e di norma nel giro di 4-5 iterazioni la procedura termina. Questo aspetto rende particolarmente adatta la tecnica del mean-shift ad applicazioni in real-time.



(a) Frame nel quale si vuole determinare la posizione del soggetto. La finestra di ricerca parte dall'ultima posizione nota



(b) Andamento del coefficiente di Bhataccharyya sul frame (in un intorno del soggetto)



(c) Prima iterazione

(d) Seconda iterazione

(e) Terza iterazione

(f) Quarta iterazione

Capitolo 4

Video data set

Per poter studiare un algoritmo di tracking è fondamentale avere a disposizione un buon numero di video sui quali poter verificarne il funzionamento. I video dovranno sì rispecchiare l'ambito a cui il sistema è destinato, ma al contempo dovranno comprendere scenari e soggetti diversi tra loro. Altrimenti il rischio è di ottimizzare un sistema per un piccolo sottoinsieme delle possibili condizioni di luce, background, problematiche e trovarsi con un algoritmo poco flessibile e robusto.

Un buon numero dei lavori presenti in letteratura dedicano poco spazio alla fase di test di un algoritmo, e spesso presentano i risultati solamente su un paio di video. Altre volte vengono proposte delle modifiche, o affinamenti e varianti pensate appositamente per migliorare le prestazioni in un unico video. Il tipo di sistema che è oggetto di questa tesi è tanto più interessante quanto più ridotto è l'intervento di un operatore umano. Non è conveniente pensare (e in alcune installazioni non è nemmeno fattibile) che possano essere definite a priori le condizioni di funzionamento, o il tipo di soggetti. Eventualmente, se in possesso di tecniche affidabili per farlo, si può pensare di adattare automaticamente il comportamento di un algoritmo di tracking in base al soggetto o all'ambiente in cui si sta muovendo. Anche in questo caso è comunque di fondamentale importanza avere un'insieme di video diversificati, dunque una parte importante della tesi è stata dedicata alla costruzione del *data set di video*.

In relazione all'applicazione considerata, si possono definire alcune linee guida nella scelta dei video da inserire nel data set, come:

- Persone o veicoli come soggetti
- Ambienti generalmente poco affollati (si presume che il soggetto da tracciare sia in un ambiente ad accesso limitato)
- Occlusioni parziali, cambi di luce e posa come problematiche principali (eventuali occlusioni totali dovrebbero essere rilevate dal drift detector).
- Soggetti diversi e più o meno contrastati rispetto allo sfondo.
- Veicoli veloci e lenti, persone che corrono e persone che camminano.
- Durata compresa tra i 10 e i 60 secondi.

Figura 4.1: *Esempi di video presenti nel data set*

Al fine di facilitare l'analisi degli algoritmi e di organizzare in maniera organica il data set, lo si è suddiviso in *categorie*. Ciò consente di vedere in maniera veloce le problematiche a cui l'algoritmo di tracking è più sensibile e quelle in cui è più robusto. Le categorie sono state scelte in maniera da raggruppare le situazioni più frequenti nei scenari di videosorveglianza, cercando nello stesso tempo di non frammentare eccessivamente l'insieme. Si precisa che, in base anche a quanto specificato nel paragrafo precedente, non si è al momento interessati ad analizzare il tracking per tipologia di soggetti, bensì per tipo di problematiche.

Si sono raccolti in tutto 38 video, per un totale di 23105 frame. Tutti i video del data set sono affiancati ad un file contenente la ground truth del soggetto da tracciare. Questo lavoro è indispensabile per poter studiare rigorosamente le prestazioni, senza dover affidarsi ad un giudizio soggettivo di chi sta guardando. Le categorie scelte contengono ognuna circa 5 video e sono le seguenti:

1. *Casi facili*
2. *Cambi di scala*
3. *Occlusioni*
4. *Cambi di posa*
5. *Sfondo complesso*
6. *Basso contrasto*
7. *Cambi di luce*
8. *Casi difficili*

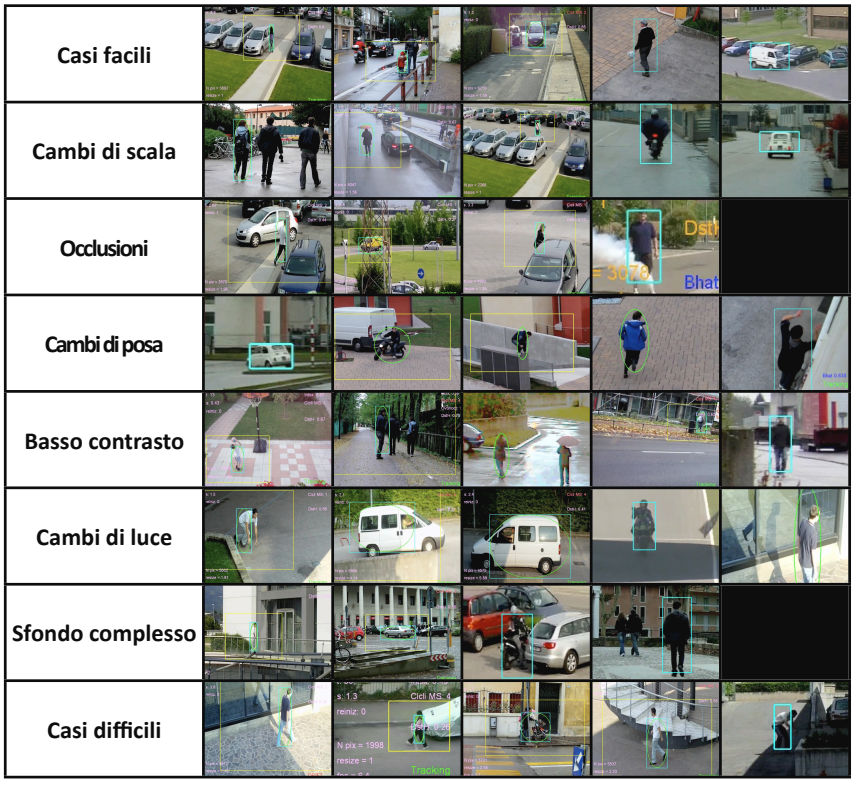
Nella categoria *casi facili* si sono inseriti quei video in cui ci si aspetta che un qualunque algoritmo di tracking dia discreti risultati. Sono le situazioni in cui si dovrebbe riuscire a garantire la robustezza del sistema.

Nella categoria *casi difficili* si sono inseriti quei video dove, a causa del sovrapporsi di più problematiche, non si richiede necessariamente un buon funzionamento dell'algoritmo. Sono dunque un buon campo per provare i limiti delle tecniche utilizzate.

La categoria *cambi di scala* è utile per verificare l'adeguatezza del sistema di aggiornamento della scala, in quanto presenta i casi tipici in cui il soggetto si allontano o si avvicina alla telecamera. Nella categoria *basso contrasto* sono raccolti i video nei quali il target è simile al background mentre con *sfondo complesso* si intendono i video nei quali l'ambiente è ricco di dettagli, corner o oggetti che possono creare disturbo ad alcune tipologie di algoritmi di tracking. La categoria *cambi di luce* contiene video brevi, idonei a valutare la resistenza di un algoritmo ad un netto cambio di illuminazione.

Le coordinate del bounding box, memorizzate nel file allegato contenente la ground truth del soggetto, sono state segnate manualmente attraverso il tool sviluppato da Piotr Dollàr, liberamente scaricabile presso [11]. Ciò permette, oltre alla possibilità di studiare il comportamento del tracker, di reinizializzare il tracking nel caso il target venga perso. In tal modo si possono sfruttare anche video molto lunghi.

Un'altra interessante opportunità è quella di analizzare l'andamento di alcuni valori lungo il percorso seguito dal target. Si può ad esempio studiare quanto velocemente varia la scala nel data set, si possono valutare indici che misurino la differenza tra background e foreground, o la differenza del target tra un frame e il successivo o molto altro ancora.



Capitolo 5

L'algoritmo di tracking

Si è deciso di studiare, mettere a punto e valutare le prestazioni di un algoritmo di tracking basato su Mean-Shift, in quanto utilizza informazioni molto diverse rispetto al template matching, ed è già sfruttato con buoni risultati in varie applicazioni [21] [22] [23] [20]. In particolare si fa uso di tutti e tre i canali dell'immagine RGB andando a creare un modello del target basato sul colore, a differenza di molte altre tecniche che si basano solamente sull'immagine in scala di grigi.

Si è partiti dalla soluzione proposta nell'articolo [12], che è stato il primo a proporre la tecnica ed è quello al quale tutti i lavori successivi fanno riferimento. Un'implementazione base dell'articolo si può trovare in [19].

5.1 Lo spazio colore

La maggioranza dei dispositivi di acquisizione di immagini registrano il colore dell'immagine grazie al filtro di Bayer posto sopra al sensore. Il filtro di bayer è una matrice di filtri colorati con i colori primari additivi: rosso, verde, blu (RGB). In questo modo ad ogni cella fotosensibile arriva la luce filtrata da uno dei tre colori primari. Tramite algoritmi di demosaicizzazione i valori vengono interpolati per ricostruire in tempo reale l'immagine a risoluzione piena. Trascurando tutti gli algoritmi di elaborazione dell'immagine si può affermare che l'RGB rappresenta direttamente la misura effettuata dal sensore.

L'RGB è anche lo spazio colore utilizzato per memorizzare le immagini digitali, quindi è immediato iniziare lo studio utilizzando l'informazione disponibile direttamente. Esistono diversi altri modi di rappresentare un'immagine a colori: molto utilizzato nel campo della visione è lo spazio HSB (Hue, Saturation, Brightness, chiamato anche HSV). In alcune applicazioni è usato lo spazio Lab, considerato percettivamente uniforme, in quanto approssima meglio la visione umana. Le installazioni delle telecamere di videosorveglianza sono nella maggioranza dei casi installazioni analogiche, e il video acquisito è in formato composito (CVBS), e viene trasmesso a distanza da cavi coassiali. Per motivi di retrocompatibilità con gli impianti esistenti, di isolamento galvanico tra interno ed esterno e di costo sono tuttora il sistema preferito nell'installazione di impianti.

Il video composito consiste della somma elettrica dei due canali di luminanza (Y) e di crominanza (C), quest'ultimo modulato in ampiezza su una

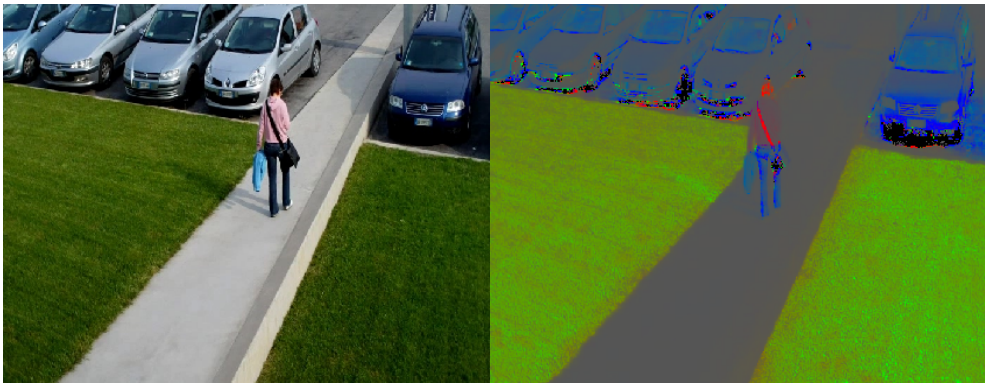
sottoportante a 4.433 MHz (nello standard televisivo PAL). In Albert il video viene acquisito e digitalizzato in formato Y'UV, composto da una componente di luminanza (Y) e due di crominanza (U e V).

In alcuni lavori in letteratura consigliano come spazio colore l'*RGB normalizzato* (in base all'intensità del pixel). La trasformazione è la seguente:

$$R_n = \frac{R}{R+G+B},$$

$$G_n = \frac{G}{R+G+B},$$

$$B_n = \frac{B}{R+G+B},$$



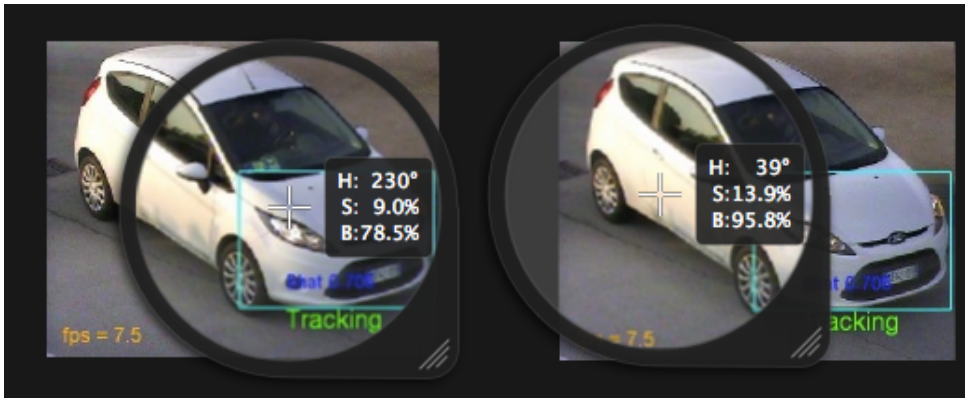
(a) Immagine in RGB (b) Immagine in RGB normalizzato

Figura 5.1: *Effetto della trasformazione del frame in RGB normalizzato.*

Questo dovrebbe rendere l'algoritmo robusto ai cambi di luminosità. Si può notare facilmente che tutte le tonalità della scala dei grigi sono mappate nello stesso punto $(R_n, G_n, B_n) = (1/3, 1/3, 1/3)$ e quindi si va a perdere completamente la differenza tra pixel di colore neutro. Infatti si è visto che nella maggior parte dei video si ottengono prestazioni migliori utilizzando lo spazio RGB semplice, e i cambi di luminosità conviene affrontarli con altre tecniche.

Si sono fatte delle prove con lo spazio HSB, tralasciando la componente B in modo da rendere meno sensibile l'algoritmo ai cambi di luminosità. In sostanza si è provato a sfruttare le sole componenti tinta e saturazione. La variazione dell'immagine che viene registrata quando c'è un cambio di luce non va a influire solamente sulla luminosità del colore, ma anche sulla tinta e la saturazione. In particolare questo accade nel passaggio da una zona illuminata dal sole ad una in ombra, e viceversa (figura 5.2). Perciò i vantaggi sperati non erano sufficienti a giustificare la conversione di ogni frame nello spazio HSB.

Si può affermare dunque che, in generale, non esiste uno spazio colore che dà in assoluto prestazioni migliori, ma va scelto in base all'applicazione e alle conoscenze a priori sul tipo di target. Ad esempio normalizzare lo spazio RGB consente di discriminare notevolmente le diverse tinte, ma non distingue tra loro le varie tonalità di grigio (tra l'altro molto presenti in scenari di



(a) Valori HSB, parte in ombra (b) Valori HSB, parte in luce

Figura 5.2: *Valori delle componenti HSB in un veicolo nella parte in ombra (prima immagine) e nella parte al sole (seconda immagine). Oltre all'ovvio cambio di luminosità (componente B) variano contemporaneamente anche le componenti H e S. Non c'è quindi un valore che, rimanendo costante, potrebbe permettere al modello di rimanere valido nonostante il cambio luce. Lo stesso discorso vale per le componenti dello spazio Lab.*

videosorveglianza). In altri casi si può supporre che il target abbia un colore molto diverso dal background e allora può essere sufficiente l'informazione contenuta dal canale H dello spazio HSB.

Alla luce delle considerazioni fatte lo spazio scelto è dunque l'RGB standard, con il quale i risultati ottenuti sono comunque buoni e non necessita di alcuna conversione che andrebbe a penalizzare il tempo di esecuzione.

5.2 Il modello del target

Nel mean shift tracking il modello è costituito da un'istogramma, di dimensione pari al numero di componenti dello spazio colore utilizzato. In questo caso l'istogramma è a 3 dimensioni, pertanto è memorizzato tramite una matrice $N \times N \times N$. Un numero N elevato consente di modellare più selettivamente il target e di essere più robusti nei casi in cui il background sia simile al foreground. Tuttavia N grande comporta un notevole aumento sia dei tempi di calcolo che dell'occupazione di memoria. Un numero N non molto grande dà invece più flessibilità nel caso di piccole variazioni dell'immagine, ma può non essere adatto a descrivere sufficientemente il target.

Dalle simulazioni effettuate sul data set si è visto che un buon compromesso è dato da $N = 24$. Perciò l'istogramma utilizzato ha dimensione $24 \times 24 \times 24$. Nelle prove in real-time si preferisce un framerate elevato, si è utilizzato dunque $N = 20$. In [12] e [19] viene usato $N = 16$.

Un istogramma tridimensionale può essere visualizzato tramite un cubo ma per maggior chiarezza si riporta in figura 5.3 (e in tutte le successive rappresentanti istogrammi), una rappresentazione in due dimensioni. La figura è data dalla somma lungo la terza dimensione.

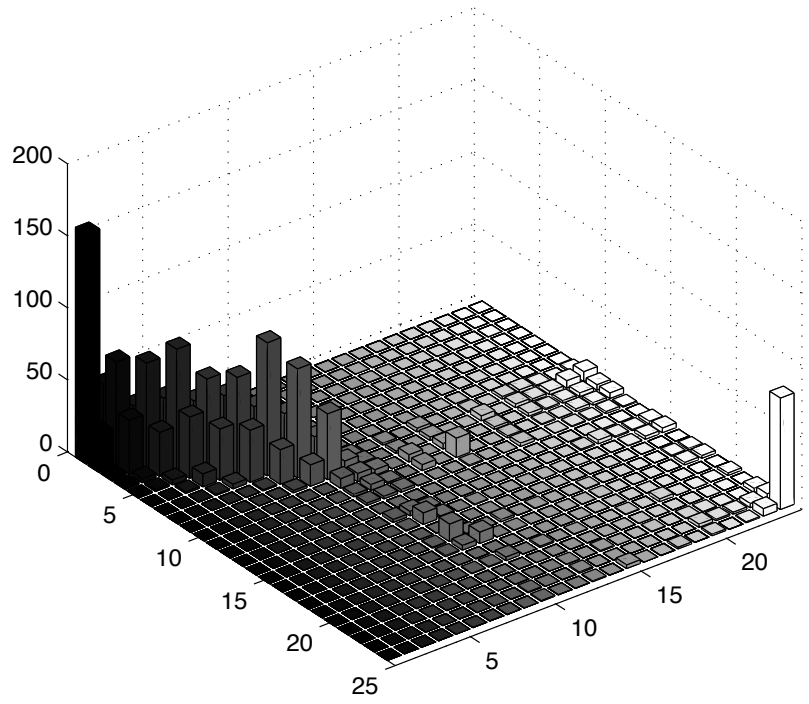


Figura 5.3: *Visualizzazione della somma lungo la terza dimensione delle componenti dell'istogramma $N \times N \times N$*

5.3 Inizializzazione del modello

Un buon modello iniziale è una condizione cruciale per l'andamento del tracking. Il modello più corretto è quello costruito sui soli pixel appartenenti al target, e che non contiene alcuno dei pixel del background. Questo problema è fondamentale in ogni algoritmo di tracking in quanto un modello errato, anche solo in parte, porta più velocemente al drift. In un'applicazione reale il modello dovrà essere costruito in maniera automatica e nel più breve tempo possibile, in modo da non ritardare l'inizio del tracking. Si può pensare di avere a disposizione un algoritmo di event detection che sia in grado di effettuare una scelta binaria tra i pixel appartenenti al background e quelli appartenenti al solo target che si desidera inseguire. Solitamente gli algoritmi in grado di far ciò allarmano anche le zone dove c'è l'ombra del soggetto, quindi è necessario rimuovere le ombre con algoritmi ad hoc. Oppure si può pensare di segmentare l'area allarmata e, cercando di identificare il tipo di soggetto, scegliere solo le parti che più probabilmente ne fanno parte.

In questo lavoro non si è trattato il problema di come individuare il target da seguire, ma si suppone di conoscere il buonding box che lo contiene strettamente nell'istante in cui si vuole cominciare il tracking. In pratica per creare il modello si sfrutta la ground truth del solo primo fotogramma. All'interno di questo riquadro sono comunque presenti dei pixel appartenenti allo sfondo, perciò si rende necessario selezionare i pixel appartenenti al foreground. Si è pensato di confrontare la distribuzione di probabilità dei pixel all'interno del bounding box con quella dei pixel appartenenti ad una cornice esterna ad esso. La distribuzione di probabilità è data dagli istogrammi e, sfruttando la regola di Bayes, si è in grado di assegnare un peso ad ogni pixel del riquadro (peso che indica la probabilità che un pixel appartenga al target dato il valore delle sue componenti RGB).

Si introducono i seguenti simboli:

- $p(f)$ è la probabilità a priori che un pixel appartenga al foreground, definita per comodità dal *kernel di Epanechnikov* (è una matrice già in memoria, non vi è pertanto la necessità di effettuare ulteriori calcoli). Serve a modellare l'ipotesi che i pixel al centro hanno una maggior probabilità di appartenere al target rispetto a quelli periferici.

- $p(x)$ è data dalla distribuzione complessiva dei pixel all'interno del bounding box e nella cornice circostante.

- $p(x|f)$ è data dalla distribuzione complessiva dei pixel all'interno del bounding box

- $p(x|b)$ è data dalla distribuzione complessiva dei pixel appartenenti alla sola cornice esterna.

La regola di Bayes è:

$$p(f|x) = \frac{p(x|f) \cdot p(f)}{p(x)}$$

$p(x|f)$ e $p(x|b)$ sono due istogrammi (non normalizzati) già calcolati per altri scopi, pertanto per trovare $p(x)$ è sufficiente sommarli e poi normalizzare

l'istogramma[1]. In figura 5.4 si riporta il kernel di Epanechnikov, utilizzato come $p(f)$.

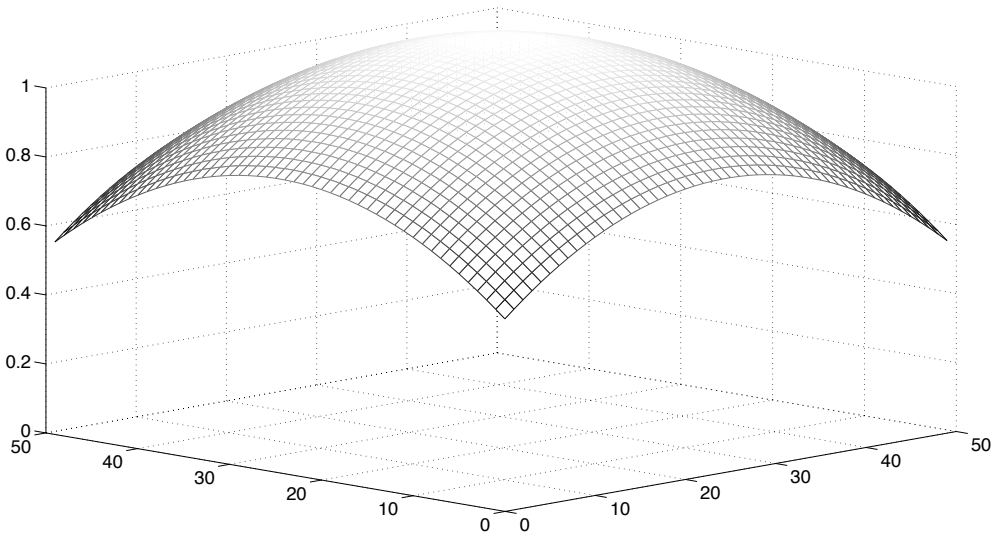


Figura 5.4: *Kernel di Epanechnikov, utilizzato come p(f)*

Una volta trovata la matrice dei pesi con la regola di Bayes viene applicata una soglia alla probabilità trovata: $p(f|x)_{min} = 0.6$ in modo da evitare che i pixel con probabilità più bassa vadano ad influire nel modello. I pixel con peso superiore a $p(f|x)_{min}$ contribuiscono in maniera proporzionale al rispettivo $p(f|x)$ nella costruzione dell'istogramma.

Si è scelto il metodo presentato perché permette di eliminare efficacemente dall'immagine iniziale i pixel appartenenti al background, e contemporaneamente è un'operazione molto veloce che consente di cominciare il tracking in pochi istanti. Un vantaggio sarebbe la possibilità di fare affidamento su più di un fotogramma per costruire il modello. Utile è anche la conoscenza del background data dagli istanti precedenti.

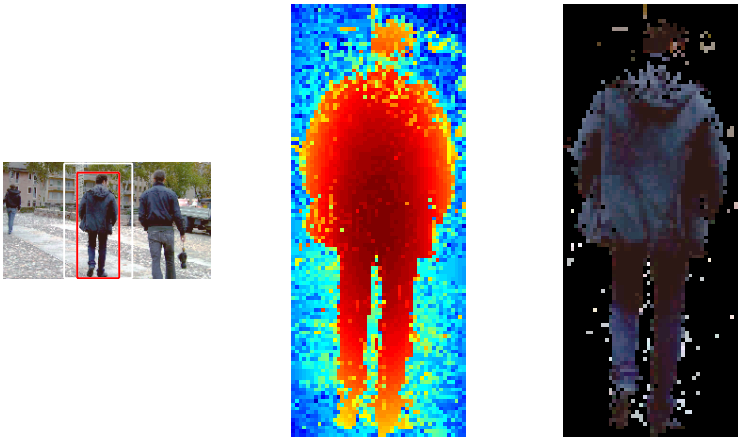
In aggiunta si è provato un diverso tipo di inizializzazione, simile a quella proposta in [12]. In questo caso i pixel del bounding box vengono pesati mediante il **rapporto bin a bin** tra l'istogramma all'interno della finestra di ricerca e quello della cornice esterna. In tal modo i pixel presenti solo all'interno avranno peso massimo, gli altri un valore intermedio compreso tra 0 e 1.

$$v(u) = min\left(\frac{b_{back}(u)}{b_{fore}(u)},\ 1\right)$$

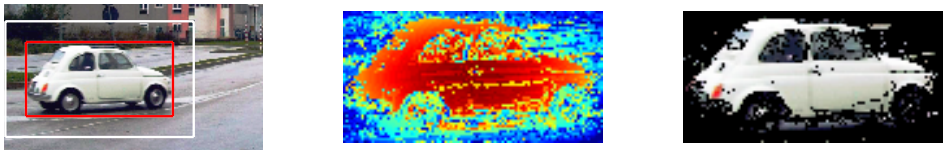
La tecnica ha portato tuttavia a risultati troppo selettivi, e di conseguenza a risultati peggiori rispetto alla soluzione proposta.

Si riportano in figura 5.5 e 5.6 degli esempi per alcuni video del data set. Si ricorda che il modello è l'istogramma tridimensionale e non l'immagine del target.

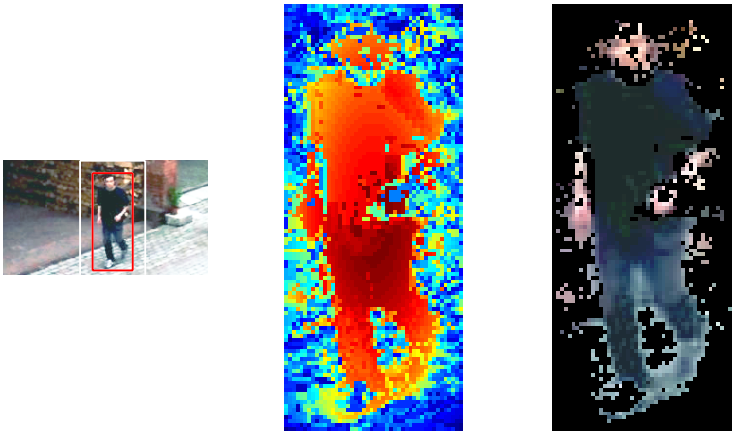
[1]Con normalizzare l'istogramma si intende dividere ogni suo bin per la numerosità dello stesso. In questo modo l'istogramma può essere inteso come una distribuzione discreta di probabilità.



(a) Portello

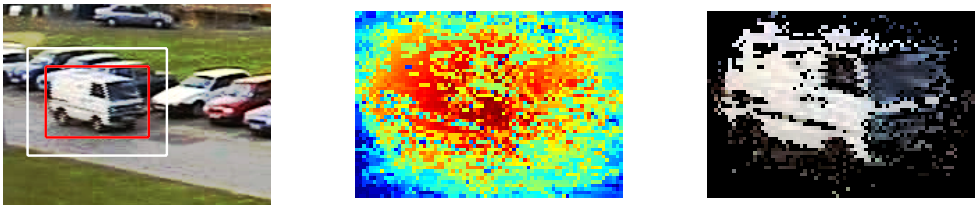


(b) LuigiRetro

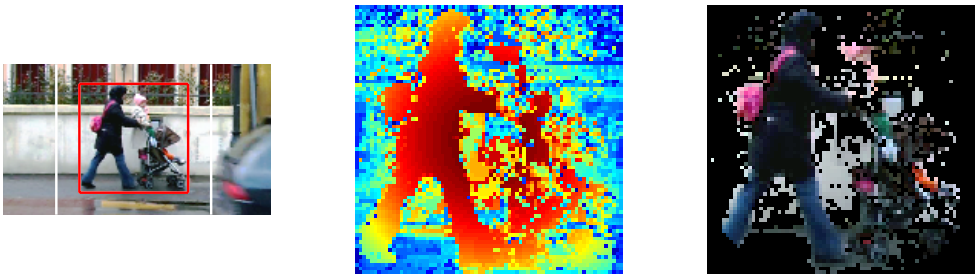


(c) BrunoIntrusione

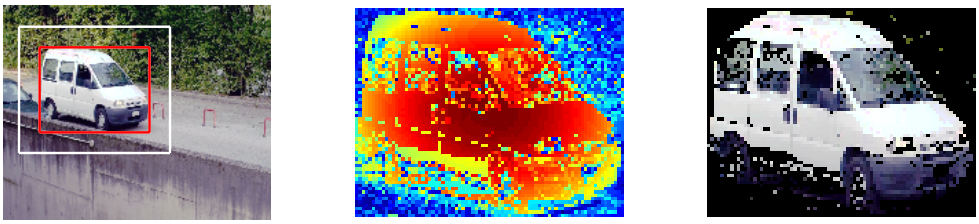
Figura 5.5: *Inizializzazione del modello.*
Nella prima immagine di ogni serie c'è il frame con indicato il bounding box (in rosso) e la cornice esterna (in bianco). Nella seconda è visualizzata la matrice dei pesi $p(f|x)$, dove in rosso vi sono i pixel con probabilità più alta e in blu quelli con probabilità più bassa. La terza immagine sono visualizzati i soli pixel che dopo la sogliatura, entrano a far parte del modello, ognuno con un peso pari a quello della seconda immagine.



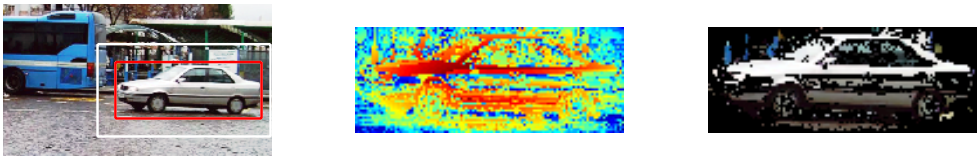
(a) Van



(b) Passeggino



(c) ScudoOmbraLuce



(d) StazioneAutoGrigia

Figura 5.6: Inizializzazione del modello. Altri esempi.

5.4 Equalizzazione dell'immagine

Accade spesso negli scenari esaminati che il soggetto sia molto scuro o molto chiaro. Ciò comporta che i valori di intensità dei pixel siano concentrati agli estremi della scala 0-255 con la quale sono memorizzati i valori (0 equivale ad un pixel nero, 255 ad un pixel bianco). Questo avviene perché l'esposizione della telecamera, che regola appunto la luminosità dell'immagine, viene calcolata sulla base di tutta l'immagine inquadrata. Il target però solitamente occupa una porzione ridotta del frame, tale da essere in pratica ininfluente per il sistema di esposizione automatica. Tipicamente nel caso di un target scuro (ma lo stesso discorso vale in generale) ci si trova con un istogramma di questo tipo:

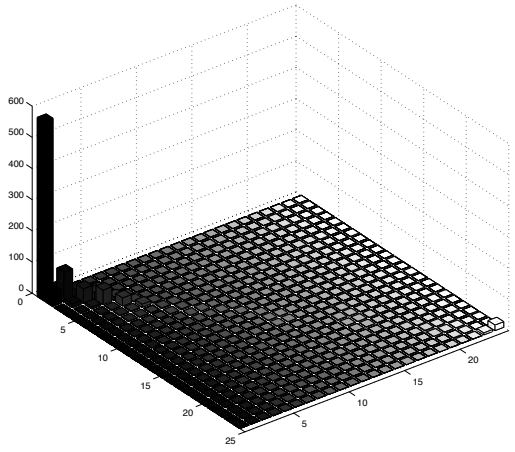


Figura 5.7: *Istogramma di un target scuro*

Si vede come la maggior parte dei pixel appartenga a pochissimi bin e questo accade non solo nel caso il sensore sia saturo, ma in genere quando il target presenta vaste aree di colore uniforme. La descrizione del target è quindi molto selettiva e in una situazione simile si va incontro ai seguenti problemi:

- Qualsiasi altra porzione del background il cui istogramma ha lo stesso bin con valore elevato può distrarre facilmente il bounding-box, portando al drift dell'algoritmo. Nella medesima situazione si possono verificare problemi nella scelta della scala corretta.

- Un piccolo cambio di luminosità, sufficiente a traslare l'istogramma di un bin, porta velocemente al drift. Si ricorda che, in questo lavoro, il confronto tra due istogrammi (per motivi di complessità computazionale) avviene solo tra bin corrispondenti, senza quindi considerare i bin limitrofi.

- L'indice di Bhattacharyya, essendo influenzato da pochi bin, risulta poco indicativo sull'affidabilità della stima trovata. È difficile dunque fissare delle soglie che, ad esempio, consentano di dichiarare perso il target.

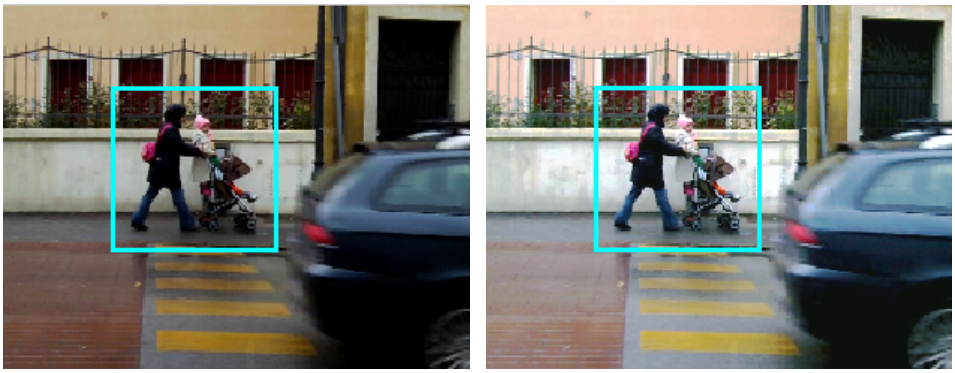
Le soluzioni prese in considerazione sono due. La prima è quella di usare istogrammi con bin a dimensione non più fissa ma variabile. In particolare se si riscontra che il soggetto ha una certa intensità prevalente si può definire un'istogramma con dei bin più stretti in quell'intorno, in modo da avere una descrizione più dettagliata, e più larghi altrove. Avendo a che fare con istogrammi tridimensionali la cosa diventa però complessa da gestire e il tutto sarebbe poco flessibile, in quanto non sarebbe conveniente variare la larghezza dei bin durante il tracking.

La seconda soluzione consiste nell'equalizzare l'intorno dell'immagine del target in modo da sfruttare tutta la scala di valori 0-255. Ciò avviene considerando singolarmente i tre canali dell'immagine RGB, e calcolando una trasformazione T che permetta di aumentare il contrasto nell'intorno del target. Il contrasto dell'immagine monocromatica aumenta in quanto si vanno a modificare i valori di intensità in maniera tale che l'istogramma (mono dimensionale) dell'immagine in uscita sia simile ad un dato istogramma, ad esempio piatto. Indicando con c_0 l'istogramma dell'immagine in ingresso e con c_1 l'istogramma desiderato, si vuole trovare la trasformazione T che minimizza la sommatoria sui bin:

$$\sum_{1}^{N_{bin}} |c_1(T(k)) - c_0(k)|$$

dove T è vincolata dal fatto di essere monotona crescente. Si ricavano separatamente le tre funzioni T_R, T_G, T_B relative ai tre canali luminosi[2] del primo frame.

Il calcolo di T non avviene sull'intero frame, ma solo su un'area equivalente a quella del bounding box iniziale aumentata del 50% e centrata in esso. In figura 5.8 si vede a sinistra l'immagine così com'è stata acquisita, a destra l'immagine dopo l'equalizzazione. Si noti com'è aumentato il contrasto globale, in particolare quello tra background e foreground. Anche i colori sono stati alterati (parete della casa).



(a) Immagine senza equalizzazione (b) Immagine equalizzata

Figura 5.8: *Fotogramma prima e dopo l'equalizzazione. In ciano è indicata la porzione di immagine sulla quale si è calcolata la trasformazione T. Si noti come sia più grande del target, in modo da includere una parte di background.*

[2]Non essendo le tre trasformazioni identiche, l'immagine elaborata presenta i colori alterati. All'utente viene comunque mostrata l'immagine acquisita, senza nessuna elaborazione.

In figura 5.9 è visualizzata, a titolo di esempio, la T_G relativa al canale verde.

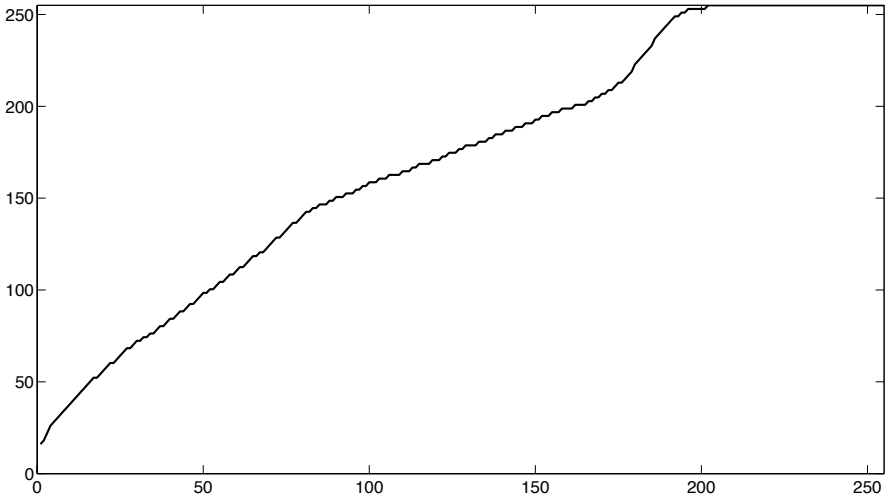
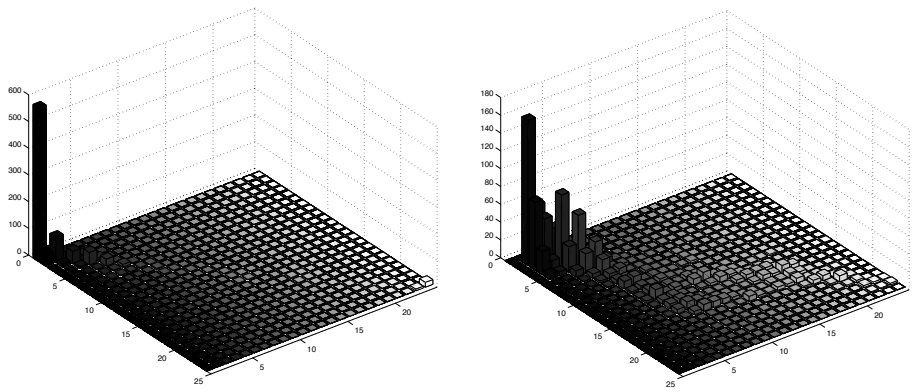


Figura 5.9: *Trasformazione T_G, che verrà applicata a tutti i frame successivi.*

In figura 5.10 è rappresentato l'istogramma di un target prima e dopo l'equalizzazione. Essendo il modello del soggetto, i pixel del background non sono considerati. Per questo motivo l'istogramma non è piatto, ma mantiene la forma precedente pur interessando un numero di bin molto più grande rispetto a prima.



(a) Istogramma senza equalizzazione

(b) Istogramma dopo l'equalizzazione dell'immagine

Figura 5.10: *Istogramma di un target scuro prima e dopo l'equalizzazione dell'immagine*

5.5 Modello del Background

Per modificare il comportamento dell'algoritmo di tracking in base all'ambiente in cui si muove il target è utile avere un modello del background. Si vedrà nei capitoli successivi come questa informazione sia sfruttata in momenti diversi e come diventi molto utile per rafforzare la robustezza del tracking.

In un algoritmo basato su istogrammi, è conveniente modellare il background allo stesso modo. Si costruisce a tal fine attorno al target una cornice (figura 5.11), in modo tale che la sua area sia 3 volte quella del bounding box.

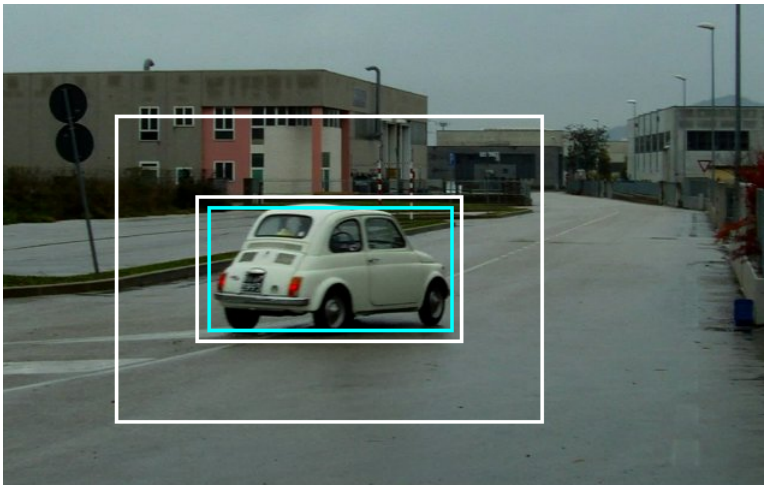


Figura 5.11: *L'area compresa tra i due riquadri bianchi è quella usata per modellare il background durante il tracking. La sua area è pari a 3 volte quella del bounding box in azzurro.*

L'aggiornamento del modello avviene attraverso una combinazione lineare convessa tra l'istogramma del background al frame precedente e l'istogramma della cornice al frame corrente. Allo stesso modo verranno aggiornate altre informazioni durante l'esecuzione dell'algoritmo. La combinazione lineare convessa è infatti in questi casi il metodo più veloce e con la minore occupazione di memoria.

$$hist_{BG}(t) = (1 - \gamma_{BG}) \cdot \frac{hist_{BG}(t-1)}{N(hist_{BG}(t-1))} + \gamma_{BG} \cdot \frac{hist_{curr}}{N(hist_{curr})}$$

Si è indicato con $N(hist)$ la somma di tutti i bin dell'istogramma (la combinazione lineare convessa ha senso solo se fatta sugli istogrammi normalizzati). L'aggiornamento del background deve essere abbastanza veloce, per adattarsi velocemente ai cambi di sfondo o ai cambi di illuminazione. Si è perciò utilizzato $\gamma_{BG} = 0.3$.

Si introducono ora due valori che verranno richiamati spesso nel seguito dell'elaborato.

Definizione 1 ($Bhat_{IntExt}$)
Si definisce $Bhat_{IntExt}$ l'indice di Bhattacharyya derivante dal confronto tra l'istogramma $hist_{BG}$ e l'istogramma dell'immagine racchiusa dal bounding box dopo l'esecuzione dell'algoritmo sul frame corrente.

Grazie all'equalizzazione dell'immagine questo valore diventa molto significativo, e sarà sfruttato sia dal drift detector che per variare alcuni parametri dell'algoritmo.

Definizione 2 ($Bhat_{ModInt}$)
Si definisce $Bhat_{ModInt}$ l'indice di Bhattacharyya derivante dal confronto tra l'istogramma $hist_{TG}$ del modello del target e l'istogramma dell'immagine racchiusa dal bounding box dopo l'esecuzione dell'algoritmo sul frame corrente.

5.6 Aggiornamento della scala

La stima della scala corretta di un target è un punto cruciale per ogni algoritmo di tracking che sia d'interesse pratico. Prima di tutto perché è il parametro di base per regolare in maniera opportuna lo zoom lungo tutto il percorso compiuto dal target. In secondo luogo migliora le prestazioni del tracker in quanto permette di fissare adeguatamente la finestra di ricerca. Se questa è troppo grande infatti aumentano le possibilità di distrazione da parte di oggetti del background, se troppo piccola rischia di muoversi all'interno del target, riducendo anche la precisione sulla posizione.

L'indice di Bhattacharyya, e in generale gli indici basati su istogrammi, non sono molto significativi come metodi per confrontare scale diverse, come notato ad esempio in [1]. Il come adattare la scala efficacemente nel mean shift tracking è stato ed è tuttora oggetto di studi e rimane il principale tra i punti deboli dell'algoritmo.

In [12] e [18] sono proposti a tal fine due metodi. Il secondo è presentato come più efficace, tuttavia costringe ad effettuare per ogni frame 3 cicli dell'algoritmo. Il primo invece è più semplice e flessibile e può essere adattato per funzionare con un notevole risparmio nei tempi di calcolo.

La tecnica proposta in questa tesi prende spunto dunque da [12]: l'algoritmo di mean shift viene usato due volte per ogni frame su due scale diverse. Nei frame pari il controllo è alla scala corrente e ad una scala minore del 10%, nei frame dispari alla scala corrente e alla scala superiore del 10%. Trovati i due bounding box candidati, si calcola l'indice di Bhattacharyya tra il loro istogramma e quello del modello. La nuova dimensione del target sarà quella che dà un'indice più alto[3]. La nuova scala è data da una combinazione lineare tra la scala precedente e quella ottima nuova.

$$scala_{new} = (1 - \lambda_{scala}) \cdot scala_{prev} + \lambda_{scala} \cdot scala_{opt}$$

[3]Precisamente le soglie sono diverse nel caso si stia cercando la scala più grande o quella più piccola, in modo da favorire l'aumento di scala. Solitamente la scala più piccola è favorita in quanto include meno background e fornisce un indice di Bhattacharyya maggiore.

Nelle simulazioni sul data set si è utilizzato $\lambda = 0.5$. La ricerca su una finestra aumentata o diminuita del 10% si rende necessaria per avere una differenza di indice apprezzabile tra le due scale. Se vi è la necessità di limitare la velocità con cui variano le dimensioni del bounding box conviene ridurre il parametro λ_{scala} piuttosto che la percentuale del 10%.

La ricerca sulla seconda scala parte dal punto trovato dalla prima, in questo modo si riesce a velocizzare la convergenza dell'algoritmo riducendo il numero di iterazioni necessarie.

5.7 Mascheratura durante il tracking

La scelta della scala è in particolar modo critica perché, basandosi su istogrammi, la scala più piccola tende ad avere un indice maggiore in quanto contiene meno background. I bin dell'istogramma normalizzato relativi ai colori del target avranno infatti un valore più grande, di conseguenza l'indice di Bhattacharyya sarà più elevato.

Si è perciò pensato di mascherare le porzioni di immagini candidate con un sistema simile a quello utilizzato nella fase di inizializzazione, dove come $p(x|b)$ si sfrutta l'istogramma del background $hist_{BG}$. Si riportano in sintesi i passi compiuti per ogni singola scala:

1. Esecuzione dell'algoritmo di mean shift fino a convergenza

2. Sul bounding box candidato si calcola la matrice dei pesi, come nella fase di inizializzazione.

3. Mascheratura dei pixel con peso minore di $peso_{min} = 0.15$. Essi pertanto non influiranno sull'istogramma del candidato[4].

4. Costruzione dell'istogramma del candidato, basato sui pesi trovati al punto 2.

5. Valutazione dell'indice di Bhattacharyya tra l'istogramma del candidato e l'istogramma del modello del target.

Grazie a questa tecnica il miglioramento sulla scelta della scala è notevole, nonostante la tecnica di base rimanga quella del confronto tra istogrammi. Si è visto invece che la precisione e la robustezza nell'inseguimento del target sono poco influenzate dalla mascheratura dei pixel del background se questa viene effettuata durante il tracking.

Non si usa lo stesso metodo proposto in [12], in quanto troppo selettivo e si è visto che può portare a casi critici in situazioni in apparenza non problematiche. Anche in [16] si evidenzia il fatto che modellare il background permette una maggior accuratezza nella scelta della scala. Un altro vantaggio è la maggior robustezza nei cambi di luminosità.

[4] Il parametro $peso_{min}$ è chiave nella scelta della scala: se piccolo o nullo viene favorita la scala inferiore e il bounding box tende ad essere contenuto nel target. Se elevato viene favorita la scala superiore e il bounding box tende a contenere il target.

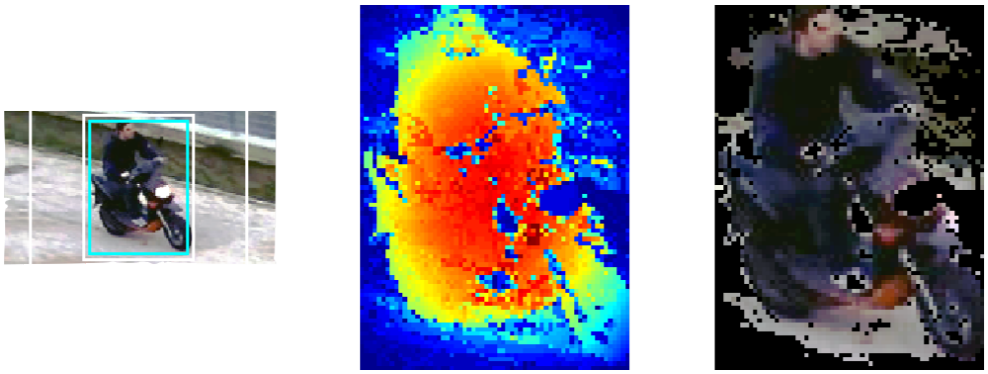


Figura 5.12: *Nella prima immagine vi è disegnato il frame corrente. Nella seconda la matrice dei pesi. Nella terza vi sono i pixel che hanno un peso maggiore di* $peso_{min}$ *e pertanto andranno a costituire l'istogramma. In questo ambito l'interesse non è mascherare precisamente tutti i pixel del background. Se infatti fosse possibile oscurarli completamente, ci si ritroverebbe nella situazione che gli istogrammi di ogni bounding box contenente il target coincidono, perdendo la capacità di scegliere la dimensione migliore.*

5.8 Altri accorgimenti

Ridimensionamento del frame

Al fine di mantenere costante la complessità computazionale si è adattato il codice in modo da ridimensionare il frame acquisito sulla base del numero di pixel del target al frame precedente. In questa maniera il codice lavora su un target composto da un numero di pixel circa costante durante l'intera durata del tracking. In più si elimina il problema di confrontare un istogramma costruito su un numero di pixel limitato con uno costruito su un numero di pixel di qualche ordine di grandezza superiore, situazione che potrebbe fornire un indice poco significativo e che si verifica quando c'è un cambio di scala molto veloce.

Le dimensioni dell'immagine vengono dunque scalate di un fattore pari a *resize*, così definito:

$$resize = max\left(1,\ \sqrt{\frac{height_{BB} * width_B B}{NumMaxPixel}}\right)$$

con $NumMaxPixel$ pari a 5000. Si è constatato che quando il target è molto piccolo (composto da meno di 1500 pixel) l'algoritmo diventa molto impreciso e poco efficace.

Aggiornamento del modello

Al fine di prolungare la durata del tracking è di fondamentale importanza l'aggiornamento del modello del target. In questo modo l'algoritmo è in grado di far fronte a lenti cambi d'aspetto del target dovuti alla luce, alla posizione e alla variazione dello zoom.

L'istogramma del modello del target viene aggiornato tramite una combinazione lineare convessa, allo stesso modo con cui viene aggiornato $hist_{BG}$:

$$p(t) = (1 - \gamma_{mod}) \cdot \frac{p(t-1)}{N(p(t-1))} + \gamma_{mod} \cdot \frac{hist_{BB}}{N(hist_{BB})}$$

dove $\gamma_{mod} = 0.006$ che, seppur bassa al fine di evitare che errori su una minoranza di frame possano corrompere il modello, è in grado di portare evidenti miglioramenti sulla correttezza e sulla durata media del tracking.

I pixel utilizzati per costruire $hist_{BB}$ non potranno chiaramente essere tutti quelli all'interno del bounding box, in quanto una parte di questi appartiene al background. Per fare in modo che non entrino nel modello si sfrutta la tecnica del paragrafo 5.7. In particolare la stessa matrice di pesi già calcolata per l'aggiornamento della scala viene qui usata per mascherare i pixel che hanno un peso minore di $peso_{min} = 0.5$. Successivamente si usa un kernel di quarto grado per fare in modo che i pixel centrali diano un contributo elevato e quelli periferici un contributo trascurabile (figura 5.13).

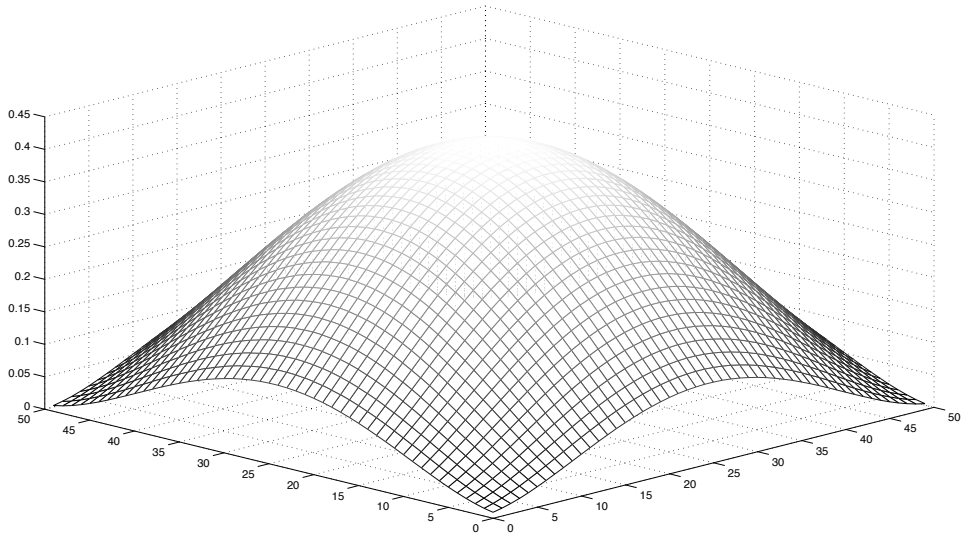


Figura 5.13: *Kernel di quarto grado utilizzato per l'istogramma dell'aggiornamento del modello.*

Questa selezione molto restrittiva sui pixel è necessaria per evitare drift causati dall'inserimento nel modello, anche in piccole quantità, di pixel appartenenti al background.

Velocità di aggiornamento variabile

L'indice $Bhat_{IntExt}$ precedentemente introdotto è un indicatore molto affidabile della complessità del tracking[5]. Ai fini dell'aggiornamento del modello del target è evidente che se tra interno ed esterno del bounding box c'è poca differenza (ovvero $Bhat_{IntExt}$ è alto) la selezione dei pixel appartenenti al target sarà meno efficace.

Si è deciso perciò di variare la velocità di aggiornamento del target in funzione di $Bhat_{IntExt}$. Dunque γ_{mod} sarà massima per valori bassi di $Bhat_{IntExt}$ e decrescerà per valori alti fino ad annullarsi (figura 5.14).

[5]L'argomento verrà discusso più in dettaglio nel paragrafo 6.1.

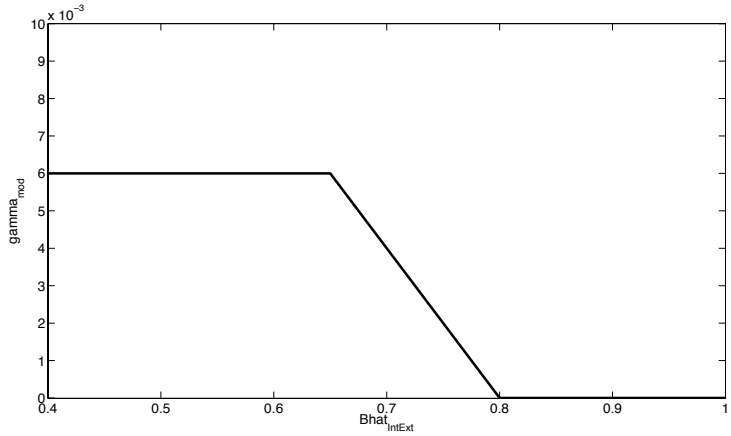
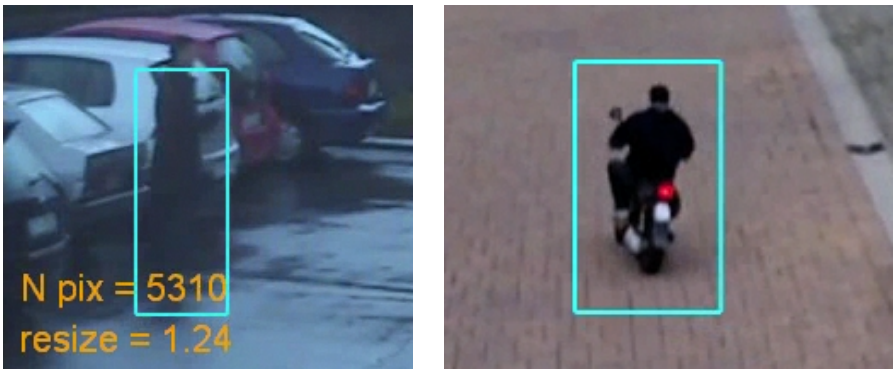


Figura 5.14: *Variazione della velocità di aggiornamento del modello in funzione di* $Bhat_{IntExt}$

Frequenza di aggiornamento della scala

Nella categoria del data set *Basso contrasto* si è notato che in alcune situazioni il bounding box tende ad espandersi velocemente a causa della somiglianza tra il colore del background e quello del target. In questi casi l'errore sulla scala porta inevitabilmente al drift. In altre situazioni, a causa di una veloce riduzione della scala del target, si ha che l'algoritmo trova correttamente la posizione del target, ma la scala del bounding box non viene ridotta abbastanza velocemente, determinando all'interno del bounding box vaste aree contenenti background.

Entrambi i casi si possono identificare da un indice $Bhat_{IntExt}$ molto alto. La soluzione, che si è dimostrata molto efficace, è quella di dimezzare la frequenza con cui viene eseguito l'algoritmo mean shift alla scala più grande. Perciò, invece di effettuare la ricerca alla scala corrente e a quella maggiore del 10%, viene cercata solamente la scala corrente. Questo fintanto che $Bhat_{IntExt}$ non torna sotto la soglia (fissata in 0.76).



(a) Caso in cui il target è poco contrastato rispetto al background.

(b) Caso in cui una veloce riduzione delle dimensioni del target ha impedito al bounding box di ridursi altrettanto velocemente.

Figura 5.15: *Situazioni nelle quali viene ridotta la frequenza con cui è eseguito l'algoritmo mean shift alla scala più grande.*

5.9 Passi dell'algoritmo

Si riportano infine in sintesi tutti i passi dell'algoritmo.

1. Acquisizione del primo frame e costruzione del modello del target che è dato dall'istogramma tridimensionale calcolato sulla base dei pixel appartenenti al bounding box iniziale.

2. Acquisizione del nuovo frame.

3. Inizializzazione del candidato target con l'ultima posizione trovata al frame precedente e confronto dell'istogramma con quello del modello iniziale.

4. Calcolo dei pesi:

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}(u)}{\hat{p}(u, \hat{\mathbf{y}}_0)}}\, \delta[b(\mathbf{x}_i^*) - u]$$

   dove $\hat{q}$ è l'istogramma del modello da tracciare, costruito sul frame iniziale, e $\hat{p}$ è l'istogramma del candidato al frame k. $b(\mathbf{x_i})$ indica l'indice del bin relativo al pixel $\mathbf{x_i}$.

5. Calcolo del vettore di Mean Shift e della prossima posizione del candidato target:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n} \mathbf{x}_i w_i g\left(\| \frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h} \|^2\right)}{\sum_{i=1}^{n} w_i g\left(\| \frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h} \|^2\right)}$$

   dove $g(x) = -k'(x)$ e $k'(x)$ è la derivata del kernel.

6. Valutazione del coefficiente di Bhattacharyya tra l'istogramma della nuova finestra candidata e quello del modello del target. Se il valore è diminuito siamo nel caso di overshoot, si procede dunque dimezzando il vettore del mean shift, fintanto che il coefficiente di Bhattacharyya non è più grande di quello calcolato al punto 3.

7. Se il vettore mean shift è minore di una certa soglia l'algoritmo termina, altrimenti si ritorna al passo 4.

8. I passi da 4 a 6 vengono ripetuti per un diverso valore della scala.

9. Scelta della scala ottima, aggiornamento del background. Calcolo dell'indice $Bhat_{IntExt}$. Aggiornamento del modello del target sulla base di $Bhat_{IntExt}$.

10. Ritorno al passo 2.

Capitolo 6

Il drift detector

In un contesto di videosorveglianza perimetrale la capacità di rilevare la perdita del target assume un'importanza fondamentale. Vista l'imprevedibilità dell'ambiente ripreso dalla telecamera, si presume che qualunque algoritmo di tracking possa fallire nell'inseguimento. In ogni caso, anche in presenza di un sistema affidabile, prima o poi il target può uscire dall'area sorvegliata, può essere nascosto da altri oggetti o può andare oltre le possibilità dello zoom della telecamera.

Con il termine *drift detector* si indicano un insieme di tecniche che permettono di dichiarare perso il target o di segnalare una situazione incerta in cui si riconosce l'inaffidabilità della stima della sua posizione. È dunque importante che la ricerca di indicatori di drift segua pari passo lo studio di algoritmi di tracking.

Considerando un sistema di tracking a telecamera singola tali *indicatori* si possono distinguere sostanzialmente in due categorie:

- *Interni*: sono tutte le informazioni che si possono ricavare dal funzionamento dell'algoritmo e che dipendono direttamente da esso. Possono essere considerate tali tutte le distanze tra il modello del target e il candidato attuale, la velocità di convergenza dell'algoritmo verso la nuova posizione del target, tutti i casi in cui non è possibile trovare una stima della posizione, stati particolari in cui giunge l'algoritmo in situazioni critiche.

- *Esterni*: sono tutte le informazioni che si possono dedurre analizzando il video acquisito e il bounding box individuato dal tracker. Si considerano tali tutti i confronti che si possono fare tra le varie immagini del target durante l'inseguimento, il confronto tra il foreground e il background, algoritmi di stima del moto, la stima della traiettoria e delle dimensioni del target sullo spazio ambiente, l'individuazione di problematiche note come riflessi o zone d'ombra, gli algoritmi di rilevazione di persone o di veicoli e infine anche tutti gli algoritmi di tracking utilizzabili come confronto con quello principale.

Si è potuto verificare che gli indicatori interni sono in grado di segnalare solo una piccola parte dei casi di drift. Diventa necessario dunque sfruttare più informazioni possibili, usando gli indicatori esterni per tutti quei casi in cui l'algoritmo di tracking non fornisce alcuna informazione.

Anche l'analisi del moto della telecamera può, come ultima possibilità, dare informazioni sulla correttezza del tracking. In caso di errata individuazione del target il moto tende infatti a diventare instabile. L'obiettivo del drift detector è però quello di intervenire tempestivamente in modo da reinizializzare l'algoritmo o, se ciò non è possibile, di far tornare la telecamera nella posizione predefinita, prima che il video visto dall'operatore diventi poco piacevole.

Per completezza si specifica che nel caso il sistema di videosorveglianza sia composto da più telecamere calibrate, un aiuto fondamentale (sia al tracker che al drift detector) può esser dato dal riscontro di una seconda telecamera che inquadra la stessa scena.

Concludendo, il drift detector per essere efficace dovrà essere una combinazione di molti indicatori, i quali permettano di coprire efficacemente almeno i casi più frequenti di drift.

6.1 Indicatori di Drift per l'algoritmo proposto

Il drift detector implementato è stato pensato come un'insieme di condizioni sufficienti per segnalare la perdita del target. Appena uno degli indicatori di drift segnala l'allarme il drift detector viene disattivato. Lo stato di drift è perciò bloccante e l'allarme viene azzerato solamente quando si reinizializza l'algoritmo.

Nell'implementazione del drift detector si sono sfruttati quattro possibili indicatori, due interni e uno esterni.

- Il primo indicatore di drift interno è dato dall'impossibilità di calcolare il vettore mean shift. Questo avviene quando la finestra di ricerca si trova in una zona in cui l'intersezione tra l'istogramma locale e quello del target è nulla: il vettore mean shift in questi casi non è definito. Si tratta di un evento raro quando si utilizzano video già registrati, ma è molto più frequente quando si controlla la telecamera PTZ in real time. Infatti un errore nel controllo o un ritardo di comunicazione o uno zoom troppo spinto o un oggetto che si interpone tra l'obiettivo e il target possono far si che l'immagine diventi improvvisamente molto diversa dal modello. In questi casi il tempo di reazione del drift detector è immediato.

- Il secondo indicatore di drift interno è l'indice dato dal confronto tra l'istogramma iniziale del target e quello del modello del target attuale ($Bhat_{ModInt}$). Sono state effettuate delle prove sulla ground truth dei video del dataset, notando come tale valore[1] decresca parecchio, anche nei casi in cui il tracking è corretto (ci sono situazioni in cui il tracker lavora senza problemi con indici molto bassi, ad esempio attorno a 0.3). Non è in definitiva un indice affidabile per descrivere la qualità del tracking. Si tratta di un problema generale dei metodi basati sugli istogrammi, fatto già notato da molti lavori in letteratura. $Bhat_{ModInt}$ però può essere utile per rilevare i casi di occlusioni totali o cambi di luce consistenti, in quanto in tali casi l'indice decresce velocemente verso

[1] L'indice di Bhattacharyya è un valore compreso tra 0 e 1, vale 1 quando i due istogrammi sono identici

valori inferiori a 0.25. Segnalando drift quando $Bhat_{ModInt} < 0.25$ per 3 frame consecutivi, si è in grado di rilevare la maggioranza delle occlusioni totali dei target (figura 6.1).

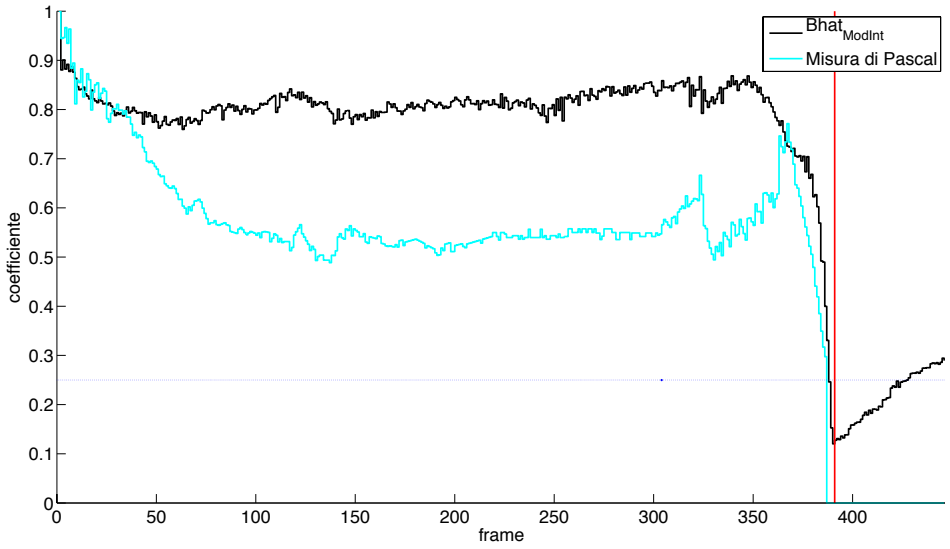


Figura 6.1: *Andamento dell'indice di Pascal (in azzurro) e l'indice* $Bhat_{ModInt}$ *(in nero) sull'intero video* LuigiRetro. *Al frame 387 il target scompare dalla visuale (occlusione totale). L'indice di pascal è infatti zero perché il riquadro della ground truth, non essendo definito, ha area nulla. Dal frame 388* $Bhat_{ModInt}$ *diventa minore di 0.25 e al frame 391 (linea rossa) viene segnalato il drift. La linea blu tratteggiata indica la soglia* $Bhat_{ModInt} = 0.25$. *Si noti come l'indice* $Bhat_{ModInt}$ *rimanga prossimo a 1 solo nei primi istanti di tracking per poi scendere a valori inferiori.*

- Il terzo indicatore può essere considerato esterno, ed è dato dal confronto tra l'istogramma all'interno del bounding box e quello del modello del background, ovvero l'indice definito nel capitolo precedente $Bhat_{IntExt}$. Grazie all'equalizzazione dell'immagine diventa un buon indice delle prestazioni generali dell'algoritmo. Si è deciso di segnalare il drift quando

$$Bhat_{IntExt} > Max_{IntExt} \qquad \text{per 5 frame consecutivi}$$

  $Bhat_{IntExt}$ è prossimo a 1 quando gli istogrammi del bounding box e quello del background sono molto simili. Questo avviene quando il bounding box arriva a contenere molto background, ad esempio a causa di una scala molto maggiore di quella corretta o a causa di intersezione nulla tra bounding box e target. La scelta del parametro Max_{IntExt} sarà discussa nel prossimo paragrafo.
  La tecnica è molto utile a segnalare anche i casi in cui il target è troppo simile al background e pertanto il risultato dell'algoritmo mean shift non è affidabile perché può essere facilmente distratto dagli oggetti dell'ambiente.

Un ultimo indicatore esterno di drift preso in esame è dato dalla saturazione del sensore sulle alte luci. Ad esempio nel passaggio dall'ombra alla luce un

target di colore chiaro avrà una o più componenti (R, G, B) saturate (figura 6.2). Quando la percentuale di questi pixel è rilevante si perde l'informazione sul colore (e contemporaneamente l'aspetto del target è cambiato molto rispetto al modello iniziale). È opportuno in questi casi fermare il tracker. In alternativa si potrebbe pensare ad un riaggiornamento veloce del modello in contemporanea ad un controllo sull'esposizione della telecamera, in modo da eliminare la saturazione del Target. Non avendo però la possibilità di studiare questo tipo di eventi la metodologia è stata esclusa dal drift detector.

(a) Frame attuale

(b) Saturazione dell'immagine nella scala RGB

Figura 6.2: *Problema della saturazione del sensore nel passaggio dall'ombra alla luce. I pixel colorati sono in saturazione per una o più componenti RGB*

Per completezza bisognerebbe considerare la saturazione del sensore verso le basse luci. Tuttavia nei video del data set alcuni target sono molto scuri (quasi completamente neri), pertanto non è da considerarsi come situazione anomala.

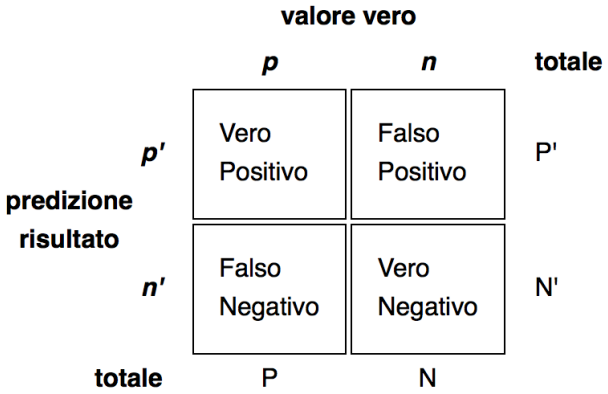
6.2 Caratterizzazione del Drift detector

Il drift detector ideale è quello che segnala il drift non appena la stima non è corretta e lo fa solo in questi casi. In una disciplina empirica come la videosorveglianza chiaramente ciò non è possibile. Ci saranno infatti sempre dei falsi positivi (FP), ovvero il caso in cui il detector segnala la perdita del target ma questo è ancora individuato correttamente dal tracker, e dei falsi negativi (FN), ovvero il caso in cui il target è stato perso ma il detector non segnala nulla.

Al fine di analizzare il comportamento del drift detector si sfrutterà lo strumento delle curve ROC (Receiver Operating Characteristic) che sono uno schema grafico per valutare un classificatore binario. Lungo l'asse delle ascisse si trova il *False Alarm Rate* (FAR) e sull'asse delle ordinate il *True Positive Rate* (TPR). La curva ROC rappresenta dunque la relazione tra allarmi corretti e falsi allarmi. Il punto ideale è $(0, 1)$, cioè tutti gli eventi sono stati rilevati e nessun falso allarme. Un algoritmo è migliore di un altro tanto più la sua curva è vicina al punto $(0, 1)$.

Si consideri dunque un classificatore binario che si vuole studiare al variare di un parametro. Fissato un valore per il parametro sono possibili quattro risultati:

- Vero Positivo (TP): il risultato e il valore vero sono entrambi positivi.
- Falso Positivo (FP): il risultato è positivo mentre il valore vero è negativo.
- Vero Negativo (TN): il risultato è il valore vero sono entrambi negativi.
- Falso Negativo (FN): il risultato è negativo e il valore vero è positivo.



Gli indici TPR e FAR sono definiti come segue:

$$TPR = \frac{TP}{TP + FN}$$

$$FAR = \frac{FP}{FP + TN}$$

Nel nostro caso si vogliono valutare le prestazioni del drift detector al variare della soglia Max_{IntExt}. Prima di specificare cosa si intende per TP, FP, TN, FN riguardo agli interessi dell'applicazione, si presentano alcune definizioni.

Definizione 3 (Indice di Pascal)
Siano rispettivamente $R_{GT}(t)$ e $\tilde{R}(t)$ gli insiemi di pixel che costituiscono il target secondo la ground truth e secondo la stima proposta dall'algoritmo. L'indice di Pascal tra i due insiemi al frame t è definito come:

$$P(t) = \frac{Area\left(R_{GT}(t) \bigcap \tilde{R}(t)\right)}{Area\left(R_{GT}(t) \bigcup \tilde{R}(t)\right)}$$

Definizione 4 (Frame non valido)
In questo lavoro un frame si considera non valido *quando l'indice di Pascal tra il bounding box e la ground truth scende sotto la soglia di* $1/3$*. La stima dell'algoritmo al frame t è quindi considerata corretta se e solo se:*

$$P(t) \geqslant \frac{1}{3}$$

Definizione 5 (Target perso)
Il target è considerato perso *se la stima dell'algoritmo non è corretta per 10 frame consecutivi.*

Dato che non è pensabile di avere un drift detector che segnali il drift nell'istante in cui questo si verifica, si possono rilassare le specifiche. Ci si può infatti accontentare che il drift venga segnalato nell'arco di una finestra temporale centrata nell'istante in cui si considera perso il target.

Per caratterizzare il comportamento del drift detector rimane a questo punto necessario definire le situazioni di *False Negative* (FN), *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), tenendo conto della finestra temporale.

Si fissi una finestra temporale di larghezza L_f frame.

- *FN*: si ha quando il target è considerato perso al frame t^*, ovvero
$$P(t) < 1/3 \quad \forall\ t \in [t^* - 9,\ t^*]$$
e il drift detector non segnala nulla nella finestra temporale centrata in t^*.

- *TP*: si ha quando il drift detector segnala la perdita del target al frame t^* e nella finestra centrata in t^* c'è almeno un frame non valido:
$$\exists\ \bar{t}, \quad t^* - \frac{L_f}{2} < \bar{t} \leqslant t^* + \frac{L_f}{2}, \quad \text{tale che} \quad P(\bar{t}) < 1/3$$

- *FP*: si ha quando il drift detector segnala la perdita del target al frame t^* e in tutta la finestra centrata in t^* si ha $P(t) \geqslant 1/3$:
$$\forall\ t, \quad t^* - \frac{L_f}{2} < t \leqslant t^* + \frac{L_f}{2} \quad \text{si ha} \quad P(t) \geqslant 1/3$$

- *TN*: si ha in tutti i frame nei quali il drift detector non segnala la perdita del target e in tutti gli $\frac{L_f}{2}$ frame precedenti e in tutti gli $\frac{L_f}{2}$ frame successivi non ci sono stati 10 frame non validi consecutivi.

Si è studiato il comportamento del drift detector al variare della soglia Max_{IntExt}, in quanto l'unica che richiede un'attenta definizione. Per ogni valore del parametro si sono conteggiati il numero di falsi negativi, di veri positivi e di falsi positivi. L'algoritmo di tracking è stato reinizializzato ogni qual volta il target è stato perso oppure quando il drift detector ha segnalato allarme (dopo aver atteso un numero di frame pari a $L_f/2$).

Per avere dei valori corretti si è moltiplicato il numero di eventi rilevati per la larghezza della finestra L_f. Nello specifico:

$$\begin{aligned} TP &= (\text{numero di veri positivi}) \cdot L_f \\ FP &= (\text{numero di falsi positivi}) \cdot L_f \\ FN &= (\text{numero di falsi negativi}) \cdot L_f \\ TP &= (\text{numero di frame analizzati}) - TP - FP - FN \end{aligned}$$

I valori di Max_{IntExt} utilizzati sono compresi nell'intervallo $[0.65, 0.95]$. La finestra di ricerca ha larghezza $L_f = 80$ frame, il che equivale a dichiarare efficace il drift detector se anticipa il drift di 40 frame o se lo segnala con un ritardo di 40 frame. In base a come si vuole impostare il sistema di tracking questa finestra può essere spostata più verso i frame futuri o più verso i frame passati. Il primo caso è utile quando si vuole semplicemente riportare il brandeggio nella posizione predefinita. Nel secondo caso si pensa ad un sistema più prudente per poter individuare nuovamente il target e continuare così il tracking.

La curva ROC risultante dalle simulazioni sull'intero data set è mostrata in figura 6.3.

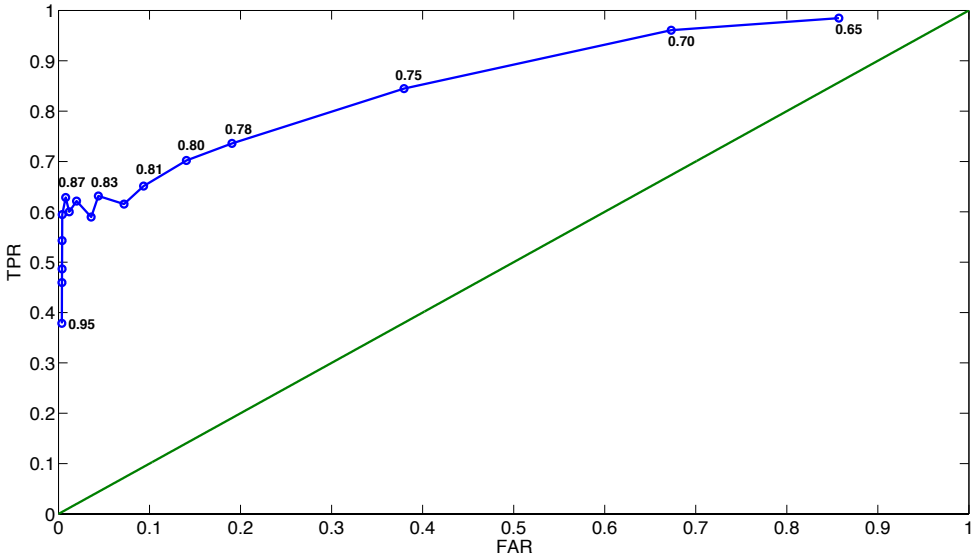


Figura 6.3: *Curva ROC del drift detector al variare del parametro* Max_{IntExt}.

Si è pensato al drift detector come ad una procedura in grado di segnalare una buona percentuale di drift, contenendo allo stesso tempo il numero di falsi allarmi. Si è scelta perciò come soglia $Max_{IntExt} = 0.87$, che permette di individuare il 63% dei drift al costo di un FAR uguale all'1%. Per aumentare il numero di drift rilevato bisogna accettare un FAR via via crescente.

L'idea del presente lavoro è quella di cercare di tenere contenuti i falsi allarmi, e lasciar spazio ad altre tecniche che consentano di individuare i drift non rilevabili attualmente. Conviene inoltre integrare il drift detector con altre metodologie come in primo luogo l'analisi della traiettoria del target nello spazio. Ciò richiede però l'utilizzo del brandeggio e un'opportuno ambiente di test che permetta di eseguire prove ripetibili.

In alcune applicazioni una finestra di 80 frame può essere troppo lunga, si è perciò rifatta la stessa analisi sull'intero data set con una finestra di 50 frame (un intervallo di ± 1 secondo centrato nell'istante del drift). Il confronto tra le due curve si può vedere in figura 6.4. Il rapporto tra TPR e FAR è diminuito e in tal caso, considerando $Max_{IntExt} = 0.87$, solamente il 48% dei drift viene segnalato correttamente.

La figura evidenzia un problema intrinseco a questo tipo di analisi, ovvero la forte dipendenza dei risultati dalle definizioni che si sono date di TP, TN, FP, FN. Esse dovranno essere adattate in base alle specifiche dell'applicazione in cui andrà inserito il drift detector. A sua volta anche il drift detector andrà tarato per avere una risposta più o meno veloce in base alle richieste.

Il drift detector progettato ha un costo computazionale irrilevante rispetto al tracker, perché sfrutta in gran parte dati e indici già in memoria per l'algoritmo di tracking. Questo vantaggio consente di sfruttarlo anche nelle prove in tempo reale con il brandeggio, dove si è dimostrato efficace in particolar modo nelle occlusioni (figura 6.5).

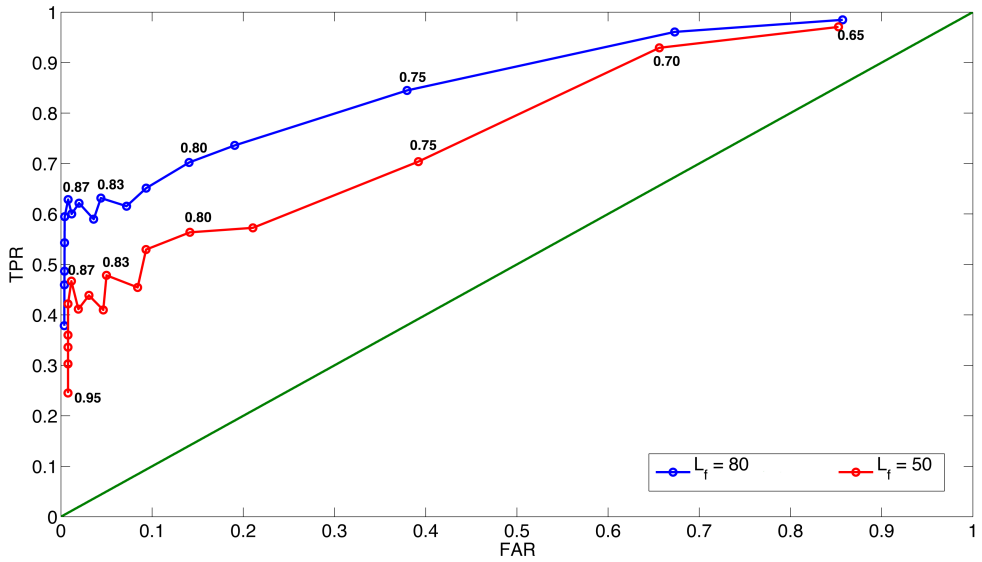


Figura 6.4: *Confronto tra le curve ROC del drift detector usando una finestra di larghezza $L_f = 80$ (in blu) e una di larghezza $L_f = 50$ (in rosso), al variare del parametro Max_{IntExt}.*

Figura 6.5: *Sequenza di tracking in tempo reale con brandeggio. Il drift è prontamente segnalato nel giro di 8 frame dall'occlusione del soggetto. In blu è riportato l'indice $Bhat_{IntExt}$. Si vede come nella seconda immagine esso assuma un valore elevato (0.92) causando la segnalazione del drift detector.*

Capitolo 7

Valutazione delle prestazioni

In questo capitolo si valuteranno le prestazioni dell'algoritmo di tracking proposto (MS) sulle categorie del data set. I risultati saranno confrontati con quelli ottenuti tramite l'algoritmo basato sul template matching (SSD). Inoltre si è provato a combinare le due tecniche in modo da usarle contemporaneamente sullo stesso video (MSSD).

L'algoritmo MSSD è stato pensato per unire i pregi di MS e SSD in modo da avvantaggiarsi nell'utilizzo con il brandeggio in real-time. Non è però ottimizzato per il data set, pertanto viene presentato solo a titolo di confronto. In MSSD l'algoritmo basato su SSD lavora in cascata all'algoritmo basato su mean-shift, sfruttandone il maggior bacino di attrazione. Se l'algoritmo SSD converge nel giro di poche iterazioni si presume che la stima sia affidabile, in tal caso esso andrà ad influenzare la scala trovata con il MS. Quando SSD non converge nel giro di qualche decina di iterazioni viene reinizializzato sulla base del bounding box del MS. L'output dell'algoritmo MSSD è la media tra le due stime. L'idea di base è di fare in modo che gli algoritmi si influenzino a vicenda solo quando la stima di uno è ritenuta più affidabile della stima dell'altro. MSSD ha dato buoni risultati nelle prove in real-time.

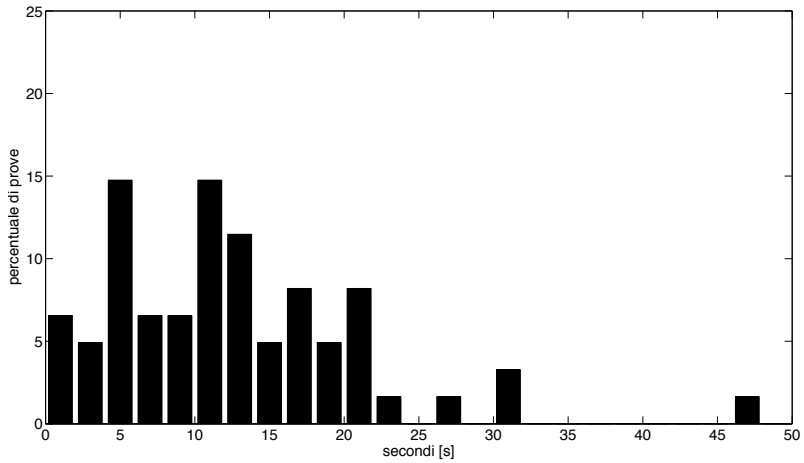
Nelle simulazioni sul data set ogni qual volta il target è perso il video viene reinizializzato in base alla ground truth del frame successivo alla perdita del target e il tracking riparte da zero. In questa fase il drift detector non ha influenza sul tracking. Di seguito si indicherà con la sigla BB il bounding box stimato dall'algoritmo di tracking e con GT il bounding box della ground truth.

7.1 Test quantitativo sull'intero data set

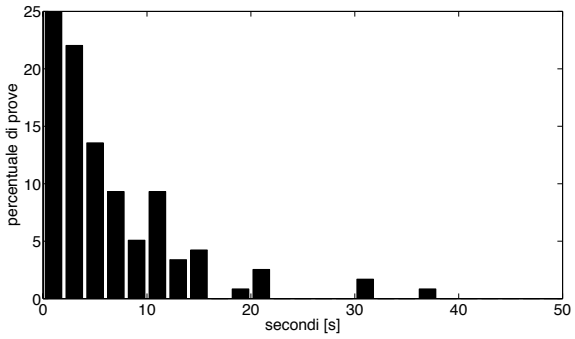
I tre algoritmi MS, SSD, MSSD sono stati valutati nelle medesime condizioni sulle diverse categorie del data set. La durata media dei tracking per l'algoritmo proposto in questa tesi è di 12.5 secondi. Per l'SSD la durata media è di 6.4 secondi, mentre per l'MSSD è di 10.2 secondi. La distribuzione delle durate però non è uniforme, pertanto la media non è un valore molto adatto a misurare le performance dell'algoritmo. Visivamente è più utile vedere come sono distribuite le durate. Nei grafici seguenti troviamo sull'asse delle ordinate la percentuale di tracking che hanno avuto una durata compresa nell'intervallo indicato nell'asse delle ascisse. Tanto più frequenti i tracking

di breve durata, tanto maggiore sarà l'altezza delle barre sulla sinistra. Le barre sulla destra rappresentano i tracking più lunghi.

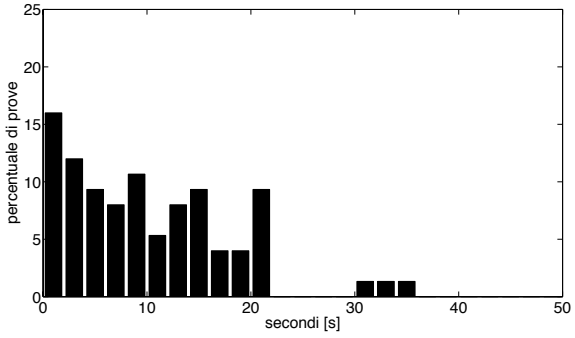
L'algoritmo MS proposto ha un numero ridotto di durate inferiori ai 5 secondi e la massima concentrazione si ha nell'intervallo tra i 5 e i 20 secondi. Nell'SSD sono invece concentrate nell'intervallo 0-10, evidenziando la scarsa capacità di questo algoritmo a essere robusto su periodi medio-lunghi. L'MS-SD presenta una situazione intermedia tra i due, con un numero però ancora elevato di durate minori di 5 secondi.



(a) Algoritmo MS



(b) Algoritmo SSD



(c) Algoritmo MSSD

Figura 7.1: *Frequenza delle durate dei tracking.*

7.2 Indici

Per analizzare più in dettaglio le performance dei tracker bisogna disporre di indici che non descrivano solo la durata del tracking, ma soprattutto la bontà della stima.

L'indice di Pascal, utilizzato nel capitolo relativo al drift detector, è utile per descrivere con un solo parametro la validità o meno della stima della posizione del target. Non dà però alcuna informazione sul tipo di errore: un basso indice di Pascal può essere causato sia da un errore sulla posizione, sia da un errore sulla scala ma con il centro del target corretto. Per distinguere i casi si è dovuto introdurre due nuovi indici, uno che misura l'errore sulla posizione e l'altro l'errore sulla dimensione. Il confronto tra il bounding box e la ground truth avviene in definitiva attraverso i tre seguenti valori:

- *Numero di frame validi*: è dato dal numero di frame tali per cui

$$P(t) \geqslant \frac{1}{3}$$

- *Errore di posizione*: è dato dalla distanza euclidea tra il centro della stima del target e il centro della ground truth. Il tutto è normalizzato secondo la diagonale del riquadro della ground truth.

$$E_{pos}(t) = \frac{\|center_{GT}(t) - center_{BB}(t)\|_2}{diag_{GT}(t)}$$

  con ovvio significato dei simboli.

- *Errore di dimensione* è dato dalla distanza euclidea tra i due vettori (relativi a GT e a BB) formati dall'altezza e dalla larghezza del rispettivo riquadro. Il tutto è normalizzato secondo la diagonale del riquadro della ground truth.

$$E_{dim}(t) = \frac{\|[h_{GT}(t) \quad w_{GT}(t)] - [h_{BB}(t) \quad w_{BB}(t)]\|_2}{diag_{GT}(t)}$$

  dove con h si è indicata l'altezza e con w la larghezza del riquadro.

Tali valori sono stati mediati sui video per ogni categoria e i risultati sono riportati nella tabella 7.2. Il valore migliore per ogni indice è segnalato in grassetto.

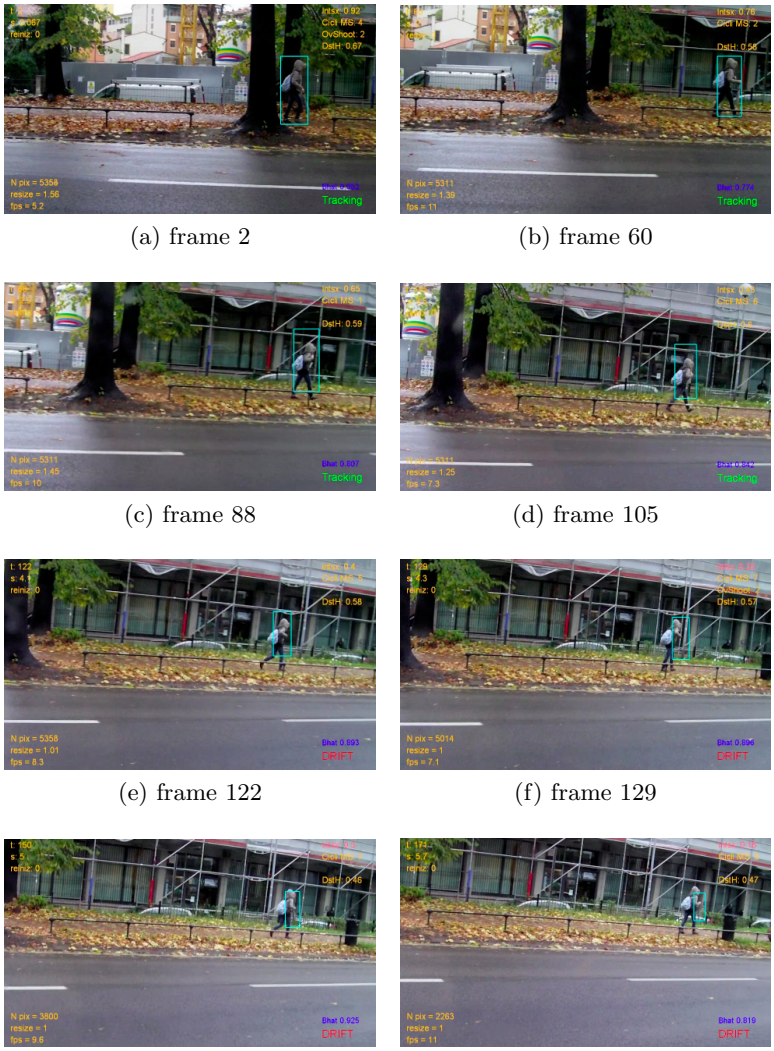
Risultati quantitativi sulle categorie del data set

Categoria del data set	Frame totali	Numero di frame validi			Reinizializzazioni			Errore di posizione			Errore di dimensione			Frame al secondo		
		MS	SSD	MSSD	MS	SSD	MSSD	MS	SSD	MSSD	MS	SSD	MSSD	MS	SSD	MSSD
Casi Facili	3727	3696 (99.2%)	**3704 (99.4%)**	3667 (98.7%)	2	**1**	3	0.076	0.091	**0.064**	0.153	0.181	**0.129**	**8.6**	3.4	3.1
Cambi di scala	3441	3392 (98.6%)	3356 (97.5%)	**3400 (98.8%)**	3	5	**2**	0.091	**0.062**	**0.062**	0.210	**0.178**	0.191	**9.3**	5.1	2.6
Occlusioni	3073	**2954 (96.1%)**	2878 (93.7%)	2902 (94.4%)	**7**	17	12	0.119	**0.095**	**0.095**	0.260	**0.164**	0.197	**11.5**	7.3	5.2
Cambi di posa	2672	**2648 (99.1%)**	2474 (92.6%)	2563 (95.9%)	**2**	14	7	**0.070**	0.458	0.086	**0.185**	0.225	0.212	**9.7**	4.4	3.9
Sfondo complesso	2253	**2174 (96.5%)**	2147 (95.3%)	2170 (96.3%)	**3**	8	5	0.105	0.096	**0.092**	0.178	**0.097**	0.142	**8.0**	3.9	2.8
Basso contrasto	2643	**2615 (98.9%)**	2643 (90.4%)	2597 (98.3%)	**1**	20	2	**0.130**	0.134	0.152	**0.137**	0.157	0.199	**8.5**	4.4	3.5
Cambi di luce	688	**670 (97.4%)**	642 (93.3%)	669 (97.2%)	**1**	2	**1**	0.085	0.060	**0.058**	0.151	0.142	**0.127**	**8.8**	6.9	4.8
Casi Difficili	4608	**4431 (96.2%)**	4218 (91.5%)	4354 (94.5%)	**10**	27	12	0.086	0.100	**0.081**	0.187	**0.171**	0.184	**8.9**	4.9	3.9
TOTALE	23105	**22580 (97.7%)**	21807 (94.4%)	22332 (96.7%)	**29**	94	44	0.094	0.136	**0.086**	0.186	**0.169**	0.177	**9.1**	4.9	3.6

Figura 7.2: *Tabella dei risultati*

7.3 Risultati

Si mostrano i risultati dell'algoritmo MS su alcuni video del data set.



(a) frame 2 (b) frame 60

(c) frame 88 (d) frame 105

(e) frame 122 (f) frame 129

(g) frame 150 (h) frame 171

Figura 7.3: Algoritmo MS, categoria **sfondo complesso**.
Il video mostra una criticità dell'algoritmo proposto. Il soggetto, a partire dal frame 88, è poco contrastato rispetto al background. Inoltre lo sfondo non è uniforme. Al frame 105 l'errore di posizione è evidente e al frame 122 viene segnalato correttamente il drift. Infatti dal frame 129 il bounding box, pur rimanendo nei pressi del target, ha un errore sia di posizione che di scala elevato, non adeguato alle necessità di un sistema di tracking PTZ.

(a) frame 2 (b) frame 128 (c) frame 179

(d) frame 218 (e) frame 288 (f) frame 397

(g) frame 471 (h) frame 530 (i) frame 595

(j) frame 699 (k) frame 731 (l) frame 757

(m) frame 798 (n) frame 850 (o) frame 914

Figura 7.4: Algoritmo MS, categoria **cambio posa**.
In questo video lungo più di 900 frame è messa in evidenza la robustezza dell'algoritmo nei cambi di posa. Il soggetto è inquadrato ripetutamente di fronte, di lato, da dietro e da sopra (frame 731 - 757). Ci sono anche cambi di scala veloci dovuti sia allo zoom che al movimento del target (dal frame 179 al 288, dal 595 al 699). Le proporzioni del bounding box rimangono costanti (frame 757), ma ciò non impedisce un corretto inseguimento del target.
Il drift detector (in basso a destra, in verde) segnala sempre un tracking corretto, quindi nessun falso allarme è stato lanciato.

(a) frame 2 (b) frame 44 (c) frame 78

(d) frame 94 (e) frame 114 (f) frame 210

(g) frame 270 (h) frame 297 (i) frame 313

(j) frame 326 (k) frame 333 (l) frame 413

(m) frame 583 (n) frame 619 (o) frame 626

Figura 7.5: Algoritmo MS, categoria **basso contrasto**.
Video caratterizzato da diversi incroci tra persone. Al frame 94 il soggetto del video passa vicino ad una seconda persona con una ridotta differenza tra i colori dei due. Il tracking prosegue correttamente anche nei seguenti incroci (frame 297 - 326). In particolare al frame 326 è evidenziata una incertezza sulla posizione, risolta una volta che le due persone non sono più sovrapposte (frame 333).
Anche in questa situazione il drift detector segnala correttamente che il tracking sta avvenendo con successo.

7.4 Considerazioni

L'algoritmo qui proposto e basato su mean shift ha dato prestazioni molto buone sull'intero data set. Il numero di reinizializzazioni causate dalla perdita dei target è basso e la velocità con cui l'algoritmo viene eseguito è molto buona (tenendo conto che gira in ambiente Matlab). L'errore di posizione è contenuto mentre quello sulla dimensione è il più alto tra i tre algoritmi, cosa che ci si aspettava da un tracker basato su istogrammi. L'algoritmo SSD, grazie alle sue caratteristiche, è il più preciso nell'individuare la scala corretta. La scarsa robustezza ai cambi di posa unita al fatto che il modello iniziale non viene mai abbandonato non consentono però lunghe durate del tracking[1].

L'unione dei due algoritmi (MSSD) non ha permesso di migliorare le prestazioni dell'algoritmo MS. Questo è dovuto principalmente al fatto che manca un indice significativo sull'attendibilità della stima dell'algoritmo SSD (mentre per MS $Bhat_{IntExt}$ è un buon indice). Un comportamento simile si può osservare anche in [9].

Analizzando la tabella per categorie si nota che che nei *video facili* tutti e tre gli algoritmi si comportano bene, infatti qui c'è la percentuale più alta di frame validi. Guardando i video si vede come i target siano inseguiti correttamente, e le reinizializzazioni siano tutte concentrate in un video molto lungo (Van). Nei *cambi di scala* è nettamente favorito l'SSD, in termini sia di errore di dimensione che di posizione. La categoria delle *occlusioni* è invece tra le più problematiche per tutti e 3 gli algoritmi. Nel MS la difficoltà maggiore si ha quando il target da occluso torna libero. Nell'occlusione infatti il bounding box si restringe per circondare la sola parte visibile del soggetto. Quando questo torna visibile completamente il bounding box non cresce di dimensione abbastanza velocemente, dunque il coefficiente di Pascal diventa piccolo e il tracker viene reinizializzato. Un altro problema, che spiega l'elevato numero di reinizializzazioni nella categoria occlusioni, è il momento in cui il target viene reinizializzato. Se in questo istante il target è occluso, il modello è poco significativo e nel giro di qualche frame si ritorna in una situazione di drift.

Nei *cambi di posa* il più robusto è chiaramente MS. La sua forza sta proprio in quello che può anche diventare un punto critico, ovvero la perdita dell'informazione spaziale. SSD ha un errore di posizione notevole perchè tende a seguire la parte di target visibile nel primo frame. Comunque in alcuni tratti il tracking funziona in maniera soddisfacente.

Lo *sfondo complesso* crea molto disturbo a MS, in quanto è più probabile che ci siano pixel simili al target che distraggono l'algoritmo. Lo si vede in tabella dalla percentuale di frame validi, che è più bassa rispetto alle altre categorie e simile a quella dei video difficili.

Nei video con *basso contrasto* tra target e background, MS ha dato ottimi risultati. Il merito è da attribuire agli accorgimenti presi in fase di implementazione, in primis l'equalizzazione dell'immagine e la variazione di alcuni parametri in funzione di $Bhat_{IntExt}$. Questa è la categoria dove SSD trova maggiori difficoltà.

[1] La ricerca della nuova posizione avviene sempre confrontando sia il modello iniziale, sia l'immagine al frame precedente. Il modello basato sul primo frame non viene mai aggiornato, in quanto unico frame certo.

I video della categoria *cambi luce* sono da intendersi più che altro come dei test[2], il cui risultato può essere negativo o positivo. Positivo se il tracking continua senza troppi disturbi anche dopo il cambio luce, negativo se il cambio luce causa drift. C'è da precisare che per MS molto dipende dal background in cui si muove il soggetto: se è complesso il cambio luce causerà molto probabilmente un drift. Se invece è uniforme in genere il test è superato. Molti cambi luce sono presenti anche nei video difficili e si è potuto constatare che questo rimane uno dei punti più critici per un sistema di tracking.

I *casi difficili* sono per loro natura molto vari, e riassumono le difficoltà delle altre categorie. MS rimane l'algoritmo che consente tracking più prolungati rispetto a SSD e MSSD. L'alto numeri di reinizializzazioni (in tutti e tre i casi) indica che ci sono ancora molte criticità difficilmente superabili ed evidenzia la difficoltà di avere un tracking robusto in una grande varietà di situazioni.

Le ultime considerazioni sono in merito all'algoritmo MSSD. Combinando MS con SSD non si è riusciti a ridurre la frequenza di drift di MS. Il motivo principale è quello spiegato precedentemente. Si ha che nei casi critici SSD va ad influire negativamente su MS, accelerando il drift. Chiaramente avviene anche il contrario, ma il valore $Bhat_{IntExt}$ è molto più affidabile e consente di regolare meglio il rapporto tra i due.

La combinazione dei due algoritmi ha ridotto il numero di reinizializzazioni rispetto ai singoli nei cambi scala, mantenendo basso l'errore di posizione come nell'SSD. Inoltre, nonostante la frequenza di drift sia maggiore rispetto a quella di MS, l'errore di posizione medio dell'MSSD è il minore dei tre nel data set. Ciò sta ad indicare che lo sviluppo e l'affinamento delle tecniche di combinazione di due algoritmi, nonchè il modo in cui essi si influenzano, possono portare vantaggi ad un sistema di tracking.

[2]I video di questa categoria sono difatti molto corti, come si vede dal numero di frame in tabella.

Capitolo 8

Prove in RT con camera PTZ

L'algoritmo proposto è stato pensato e implementato per funzionare anche in real time su telecamera PTZ. Il brandeggio utilizzato nelle prove è denominato *Ulisse Compact*. Esso è prodotto direttamente dall'azienda Videotec. La possibilità di interfacciare il brandeggio con un normale PC tramite la porta seriale ha permesso l'esecuzione di test sfruttando direttamente il codice Matlab scritto per le simulazioni sul data set.

Si è implementato un sistema di controllo del brandeggio e della telecamera che consente un tracking fluido e una visione piacevole. Sono state compiute prove sia in interni con oggetti di piccole dimensioni, sia in esterni con persone e veicoli.

8.1 Strumentazione

Il PC è dotato di un processore Intel Q9400 a 2.66 GHz equipaggiato con 4 GB di RAM. Il video analogico della telecamera è acquisito tramite un frame grabber e grazie all'*Image Acquisition Toolbox* di Matlab. Il tempo richiesto per eseguire l'acquisizione di un frame è di qualche centesimo di secondo.

Il brandeggio utilizzato è il modello Ulisse Compact. La rotazione sull'angolo di pan è continua, quella di tilt è compresa tra -90° e 90°. Per entrambi la velocità minima è di 0.1°/s e la massima di 200°/s. Tramite comandi seriali è possibile impostare la velocità desiderata o definire la posizione assoluta del brandeggio. Una logica di controllo interna imposta una curva di accelerazione adeguata in modo da non danneggiare i motori passo-passo.

Come si vede dalla figura 8.1 gli assi di rotazione pan e tilt non sono coincidenti. Anche il centro ottico della telecamera non è attraversato da nessuno dei due assi. Questo fatto può essere trascurato per soggetti sufficientemente distanti dal centro ottico, pertanto da qui in poi si assumerà che gli assi di pan e tilt si intersecano in un punto C coincidente col centro ottico della telecamera.

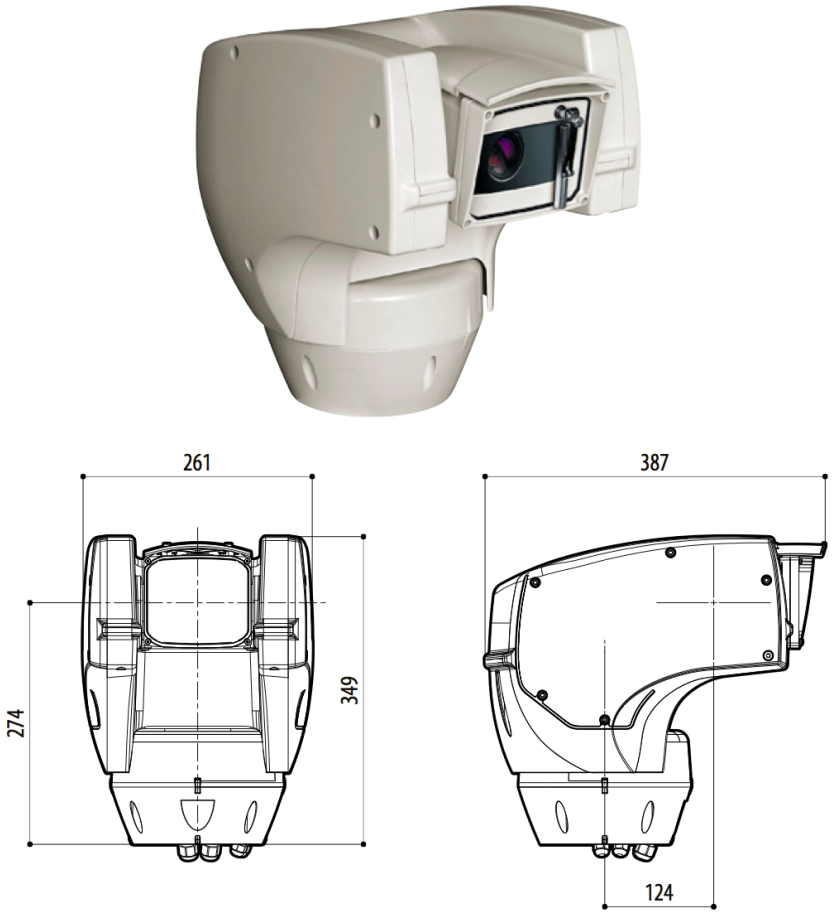


Figura 8.1: *Il brandeggio Ulisse Compact*

La telecamera SONY è a colori ed ha risoluzione 720×576. Il sensore CCD ha dimensione 1/3 di pollice. L'obiettivo di cui è dotata è uno zoom $36\times$ con lunghezza focale variabile da 3.4 mm a 122.4 mm, che consente un angolo di campo massimo di 57.8° e minimo di 1.7°. L'apertura massima varia da f/1.6 a f/4.5. Una serie di comandi permettono di modificare i parametri del bilanciamento del bianco, dell'esposizione e altro. Lo zoom si può muovere assegnando un valore e questo verrà raggiunto nel più breve tempo possibile. Altrimenti si comanda lo zoom di muoversi in avanti o indietro, e lo si può fermare con un terzo comando. Altre modalità non sono previste.

La connessione seriale utilizza lo standard EIA RS 485 half duplex. Protocollo Baud Rate 38400 bps 8-N-1. Ad ogni comando corrisponde una stringa di caratteri di lunghezza variabile. È possibile interrogare il brandeggio per avere la sua posizione attuale, il livello di zoom o la velocità istantanea. Le comunicazioni sulla seriale possono essere massimo 12 al secondo.

8.2 Modello della camera

Per il controllo del brandeggio è sufficiente usare un modello pinhole della telecamera (figura 8.2). Il centro del sistema assoluto (X_a, Y_a, Z_a) coincide con il centro ottico C della camera. Sul piano immagine si fissa il riferimento (X_c, Y_c) per cui ogni punto $P = (X, Y, Z)$ nel sistema di riferimento assoluto sarà proiettato in un punto dell'immagine di coordinate $p = (x, y)$.

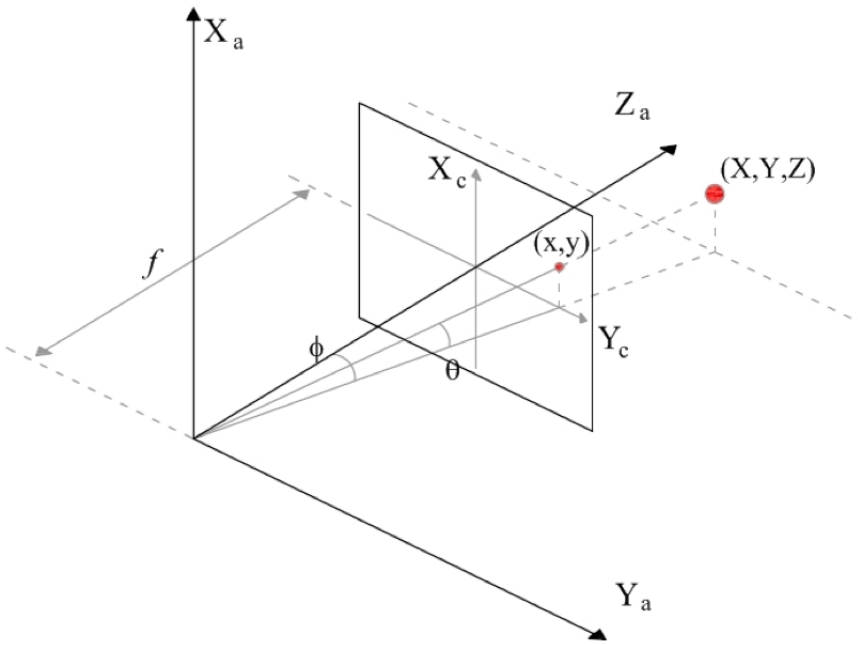


Figura 8.2: *Modello pinhole della telecamera*

La telecamera è modellata come una proiezione

$$\Pi \ : \ \mathbb{R}^3 \ \rightarrow \ \mathbb{R}^2 : \ (X, Y, Z) \rightarrow \ (x, y)$$

le cui equazioni sono:

$$x = f\frac{X}{Z} \tag{8.1}$$

$$y = f\frac{Y}{Z} \tag{8.2}$$

dove f è la lunghezza focale dell'obiettivo. Le lunghezze focali dell'obiettivo ai vari livelli di zoom sono state dedotte da test su alcuni valori (i.e. 1x, 2x, 4x, 8x, 16x) e poi interpolati. Una calibrazione precisa sarebbe auspicabile.

Per quanto riguarda il brandeggio, si possono associare ad ogni punto del piano immagine le coordinate di pan ϕ e tilt θ relative. Si definisce la mappa:

$$\Gamma \ : \ \mathbb{R}^2 \ \rightarrow \ \mathbb{R}^2 : \ (x, y) \rightarrow \ (\phi, \theta)$$

Con riferimento alla figura 8.2, le coordinate di pan e tilt relative al punto (x, y) si trovano con le seguenti formule:

$$\phi = \phi_0 + \arctan\left(\frac{y}{f}\right) \tag{8.3}$$

$$\theta = \theta_0 + \arctan\left(\frac{x}{f}\cos(\phi)\right) \tag{8.4}$$

$$\tag{8.5}$$

La posizione assoluta della telecamera è nota solamente negli istanti precedenti all'inizio del tracking. Come si vedrà nel prossimo paragrafo la posizione assoluta della telecamera non è necessaria ai fini del controllo e del funzionamento dell'algoritmo. La traiettoria del brandeggio potrà eventualmente essere stimata a posteriori sulla base delle velocità impresse ai motori e all'intervallo di tempo che è intercorso tra un comando e il successivo.

8.3 Controllo in velocità

Una volta acquisito il frame e trovata la nuova posizione del target si tratta di assegnare al brandeggio una nuova velocità, che consenta di tenere il target in prossimità del centro dell'immagine. Lo scopo del progetto è quello di avere un tracking fluido. A tal fine è sufficiente il controllo in velocità della telecamera tramite un controllore PD. Chiaramente più è alto il framerate, migliore sarà l'inseguimento del soggetto.

Il movimento del brandeggio si basa sulla distanza del centro del target dal centro del piano immagine. La lettura della posizione della PTZ tramite la porta seriale richiede circa un decimo di secondo, pertanto conviene non effettuarla, ma fare affidamento solamente all'errore di posizione sul piano immagine e non all'errore sulle coordinate (ϕ, θ).

Siano $u_\phi(t)$ e $u_\theta(t)$ gli ingressi di controllo del sistema. Il controllore è così definito:

$$u_\phi(t) = K_p \cdot \frac{x(t)}{f(t)} + K_d \cdot \left(\frac{x(t)}{f(t)} - \frac{x(t-1)}{f(t-1)}\right) \tag{8.6}$$

$$u_\theta(t) = K_p \cdot \frac{y(t)}{f(t)} + K_d \cdot \left(\frac{y(t)}{f(t)} - \frac{y(t-1)}{f(t-1)}\right) \tag{8.7}$$

Non avendo a disposizione un modello dinamico della telecamera l'unica possibilità è stata la taratura manuale dei parametri del controllore PD. Al fine di rendere piacevole la visione, oltre alla fluidità del movimento del brandeggio, è fondamentale che l'immagine non abbia oscillazioni, anche a bassa frequenza[1]. Questo equivale a richiedere che la risposta del controllore non presenti sovraelongazioni. Per ottenere tale risultato si è limitata l'azione della componente derivativa nei soli casi in cui il suo intervento fosse nella stessa direzione della componente proporzionale. Così si può tenere un guadagno K_d alto che consente, nei casi in cui un target fermo si rimette in moto, una veloce accelerazione del brandeggio. Quando invece un target in moto rallenta interviene solamente la componente proporzionale, consentendo un inseguimento dolce ed evitando il movimento oscillatorio della telecamera.

[1] Non è sono stato rilevato invece nessun problema relativo a vibrazioni del brandeggio.

Affinché gli algoritmi di tracking lavorino bene, la distanza che percorre il target sul piano immagine tra un frame e il successivo deve essere ridotta. Questo equivale a imporre dei vincoli sulla massima accelerazione della telecamera. Si fissi $\ddot{u}^*_{MAX}$ uguale al modulo della massima accelerazione angolare, riferita ad un target in primo piano nell'inquadratura con $zoom = 1$. Tale valore dipenderà dall'algoritmo utilizzato e dal numero di frame al secondo a cui gira. L'accelerazione massima imposta dipenderà dalla lunghezza focale utilizzata e dal rapporto tra le dimensioni del target e l'altezza del frame:

$$\dot{u}_{max}(t) = \frac{\dot{u}^*_{max}}{zoom(t)} \cdot \frac{diag_{BB}(t)}{N_{px}}$$

dove N_{px} è il numero di pixel del lato corto dell'immagine acquisita. Il vincolo imposto è importante nella fase iniziale del tracking, dove da una visione in grandangolo si passa ad uno zoom spinto sul soggetto.

Quando il target si ferma non ha senso inseguire con la telecamera i piccoli spostamenti rilevati dal tracker, in quanto provocherebbero solo disturbo all'utente. Perciò se il centro del target entra in un circonferenza di raggio $r_1 = \frac{N_{Px}}{30}$ e concentrica al centro ottico, si fermano i motori del brandeggio. Lo zoom viene bloccato nel momento in cui il target raggiunge le proporzioni prefissate. Il tracking riprende poi normalmente quando il centro del target esce da un cerchio di raggio $r_2 > r_1$, $r_2 = \frac{N_{PX}}{12}$ (figura 8.3).

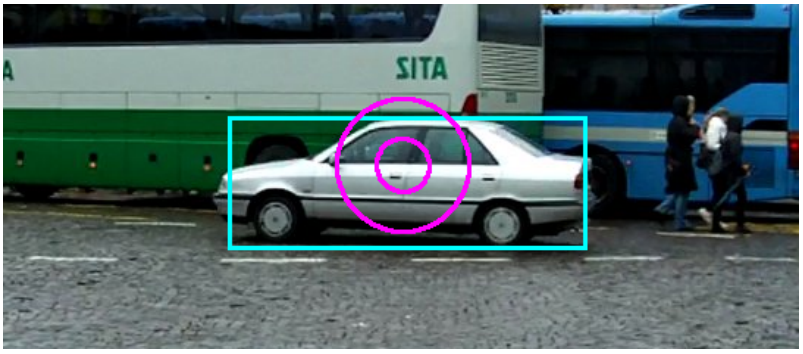


Figura 8.3: *Circonferenze che permettono di bloccare il movimento del brandeggio quando il centro del target entra in quella più piccola di raggio r_1. Quando esce dalla circonferenza più grande il movimento riprende.*

8.4 Regolazione dello zoom

Il modo più semplice per regolare il livello di zoom è imporre che il lato vincolante del bounding box sia ξ_0 volte la dimensione dell'immagine. Con *lato vincolante* si intende la dimensione più lunga del bounding box rapportata alla proporzione dell'immagine acquisita (4/3 nel caso di video PAL). Un valore di $\xi_0 = \frac{1}{3}$ pare un adeguato compromesso tra la necessità di inquadrare da vicino il soggetto e la necessità di avere margine attorno, sia per i movimenti del target, sia per compensare eventuali errori di scala.

Senza perdita di generalità ipotizziamo che il lato vincolante sia quello verticale. Il nuovo valore dello zoom sarà determinato dalla formula:

$$\begin{aligned} zoom(t) &= zoom(t-1) \cdot \delta_{zoom}(t) \\ &= zoom(t-1) \cdot \frac{N_{px}\ \xi_0}{h_{BB}(tt)} \end{aligned}$$

La variazione dello zoom nell'intervallo tra un frame e il successivo deve essere limitata, altrimenti il tempo di campionamento può non essere sufficiente a raggiungere il valore desiderato. Un secondo problema è che se il target non si trova nel centro dell'immagine, una variazione di zoom molto veloce potrebbe portarlo ai bordi del frame, fino a tagliarlo. Questo pericolo non c'è quando si riduce lo zoom, quindi è sufficiente impostare un limite superiore a $\delta_{zoom}(t)$:

$$\delta_{zoom}(t) \leqslant \delta_{MAX}$$

In ultima analisi δ_{MAX} può essere variabile in funzione della distanza del target dal centro del frame: se quest'ultimo si trova ai bordi del frame δ_{MAX} deve essere piccolo per evitare di tagliarlo. Se invece il target si trova nel centro dell'immagine, δ_{MAX} può essere meno restrittivo, consentendo movimenti più veloci dello zoom compatibilmente con il tempo di campionamento.

Una volta trovato il valore $zoom(t)$ questo viene combinato linearmente con il valore precedente $zoom(t-1)$ al fine di renderne fluido il movimento. Infatti lo zoom non deve seguire prontamente le variazioni di dimensione del bounding box, ma deve piuttosto filtrarne gli errori sulla dimensione del target:

$$zoom(t) = \gamma_z\ zoom(t) + (1-\gamma_z)\ zoom(t-1)$$

con $\gamma_z = 0.1$.

Il parametro ξ_0 dipende in realtà da molti fattori. A parità di brandeggio, algoritmo e suo framerate, ξ_0 dovrebbe essere direttamente proporzionale al modulo della velocità angolare del target nello spazio ambiente (riferita al punto C della telecamera) e inversamente proporzionale alle dimensioni reali del soggetto. Per non appesantire la notazione consideriamo solamente la velocità angolare misurata rispetto all'asse X_a, ovvero nella stessa direzione della velocità di pan:

$$\xi(t) \propto \frac{\dot{\phi}_{target}(t)}{width_{target}}$$

in cui $width_{target}$ può essere stimata solo se si dispone di un riconoscitore di persone o veicoli oppure se si sa a priori il tipo di target che entreranno nell'area sorvegliata. $\dot{\phi}_{target}(t)$ può essere facilmente calcolata durante il tracking.

In questa tesi, non potendo sapere le dimensioni del target che si sta inseguendo, si è imposto uno $\xi(t)$ funzione della distanza tra il centro del target e il centro del frame. L'ipotesi è che più il target è distante dal centro e più la sua velocità sarà elevata.

$$\xi(t) = \xi_0\ \cdot f\left(\|(x(t),\ y(t))\|\right)$$

con ξ_0 che ora può essere scelto più piccolo, ottenendo così un maggior ingrandimento quando il target è fermo ($\xi_0 = 2.5$). La funzione scelta è la cubica di

figura 8.4, la cui formula è

$$f\left(\|(x(t),\ y(t))\|\right) = 1 + 3\left(\frac{\|(x(t),\ y(t))\|_2}{\frac{N_{px}}{2}}\right)^3$$

dove il fattore $\frac{N_{px}}{2}$ serve per normalizzare la distanza.

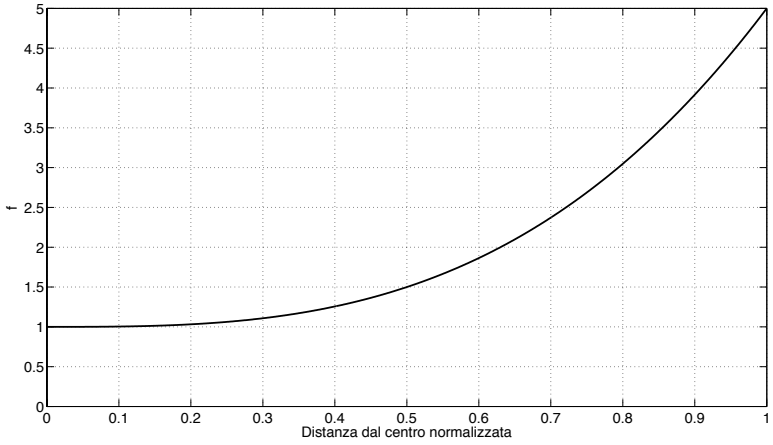


Figura 8.4: *La funzione proposta consente di mantenere basso il rapporto tra la dimensione del frame e quella del target quando questo si trova nelle vicinanze del centro, per salire velocemente quando il target si trova lontano dal centro perché ha accelerato. La curva passa per il punto (0.5, 1.5), quindi se il target si trova a metà strada tra il centro dell'immagine e il bordo avremo* $\xi(t) = \xi_0 \cdot 1.5 = 3.5$.

La tecnica qui discussa consente di ridurre le possibilità che un target troppo veloce esca dall'area inquadrata. In questo modo infatti si sfrutta la velocità di movimento dello zoom, più veloce rispetto all'accelerazione che si può imprimere al brandeggio.

8.5 Rapporto tra telecamera e algoritmo

Usare l'algoritmo con una telecamera PTZ reale porta alcuni vantaggi dati dalla conoscenza dei parametri della telecamera, altrimenti ignoti nei video del data set. In primo luogo si conosce la lunghezza focale e si conoscono le variazioni di zoom imposte dall'algoritmo di controllo della telecamera. È possibile dunque aumentare o diminuire la scala del target in modo da compensarne in anticipo le variazioni dovute alla differenza di lunghezza focale. In questo modo la scala da cui partirà l'algoritmo di tracking al frame t risulta:

$$scala_{prev}(t) = scala(t-1) \cdot \delta_{zoom}(t-1)$$

Si conosce inoltre la velocità angolare della telecamera, e si può misurare il tempo T_{elab} intercorso tra l'acquisizione del frame e il momento in cui viene dato il comando di velocità. In questo modo il punto di partenza per l'algoritmo di tracking al frame successivo può essere traslato per compensare

il movimento del brandeggio. Invertendo le formule (8.3) e considerando che $\Delta\phi = u_\phi(t-1)T_{elab}$ e $\Delta\theta = u_\theta(t-1)T_{elab}$ si trova:

$$\Delta y = f \tan \Delta\phi$$
$$\Delta x = \frac{f \tan \Delta\theta}{\cos \Delta\phi}$$

Fondamentale ai fini della buona riuscita del tracking è il blocco dell'esposizione automatica della telecamera. Con *esposizione* si intende la quantità di luce totale che viene fatta raggiungere al sensore. Essa è descritta da tre fattori, modificabili manualmente tramite un apposito comando:

- *la velocità dell'otturatore*: è il tempo durante il quale l'otturatore rimane aperto. Si può impostare un valore compreso tra 1/10000 s e 1 s.
- *l'apertura del diaframma*: è il rapporto tra la lunghezza focale e l'area dell'apertura del diaframma. L'apertura massima varia da f/1.6 a per $f = 3.4\ mm$ a f/4.5 per $f = 122,4$. L'apertura minima è f/28.
- *la sensibilità del sensore (guadagno)*: è l'amplificazione del segnale rilevato dagli elementi fotosensibili del sensore. È possibile assegnare un guadagno compreso tra -3 dB e 28 dB.

I tre valori possono essere letti interrogando la telecamera quando questa si trova in modalità esposizione automatica. Nell'istante in cui comincia il tracking tali valori vengono fissati, in modo tale che durante l'inseguimento l'esposizione non vari. Così facendo la luminosità del soggetto non cambia in base al background in cui si sta muovendo, riuscendo in questo modo ad aumentare l'affidabilità del tracking.
Per lo stesso motivo è opportuno bloccare il bilanciamento del bianco. La telecamera torna in modalità automatica una volta concluso il tracking.

La prospettiva varia in base alla distanza del soggetto dalla telecamera. Se il soggetto si trova vicino alla telecamera esso dovrà essere ripreso con angolo grande (zoom = 1, grandangolo). Se il soggetto si trova lontano dalla telecamera servirà un zoom spinto (teleobiettivo). La diversa prospettiva che si ha nei due casi comporta, a parità di velocità e dimensione del soggetto, una diversa velocità di variazione della dimensione del target nel frame.

L'angolo di campo β inquadrato dalla telecamera è dato da:

$$\beta(f) = 2 \arctan\left(\frac{d}{2f}\right)$$

dove d è la diagonale del sensore. Se è dato l'angolo che si desidera inquadrare, si trova che la lunghezza focale corrispondente è:

$$f = \frac{d}{2 \tan\left(\frac{\beta}{2}\right)}$$

Si consideri ora un target in movimento rispetto alla telecamera, in una direzione parallela all'asse ottico della stessa. Sia L la larghezza del target e h

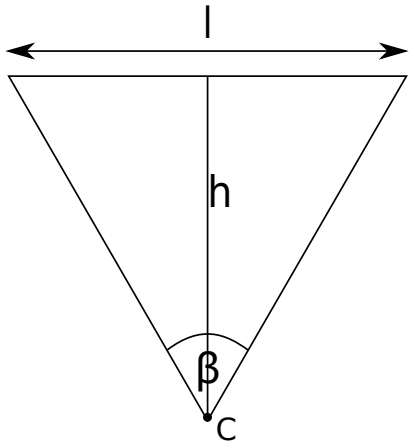


Figura 8.5: *Visualizzazione dell'angolo di campo della telecamera*

la sua distanza dal centro ottico all'istante t. Con riferimento alla figura 8.5 l'angolo necessario per inquadrare esattamente il soggetto è dato da:

$$\beta = 2 \arctan\left(\frac{L}{2h}\right)$$

La lunghezza focale voluta è:

$$f^* = \frac{d}{2}\frac{h}{L} \tag{8.8}$$

quindi la lunghezza focale necessaria ad inquadrare un target è direttamente proporzionale alla sua distanza dalla telecamera.

Si vede ora come varia invece la dimensione del target visto dalla telecamera in funzione della distanza h. Con riferimento alla figura 8.6, si supponga che un target di lunghezza L, che al frame $t = 1$ si trova a distanza h dalla telecamera, si porti al frame $t = 2$ ad una distanza $h + \Delta h$. Si supponga inoltre che al frame $t = 1$ la lunghezza focale della telecamera fosse regolata in modo tale da contenerlo esattamente nell'immagine e che questa rimanga costante nel frame successivo. Il sensore in figura è il tratto nero più spesso, mentre il target è rappresentato nelle due posizioni dalle linee rosse.
La dimensione l_1 del target sul sensore al frame $t = 1$ è data da:

$$l_1 = \frac{Lf}{h}$$

mentre la dimensione l_2 del target sul sensore al frame $t = 2$ è data da:

$$l_2 = \frac{Lf}{h + \Delta h}$$

La variazione Δx delle dimensioni del target sul piano immagine sarà:

$$\Delta l = l_2 - l_1 = -\frac{Lf\Delta h}{h^2 + h\Delta h}$$

e sostituendo la 8.8:

$$\Delta l = -\frac{\frac{d}{2}h\Delta h}{h^2 + h\Delta h}$$

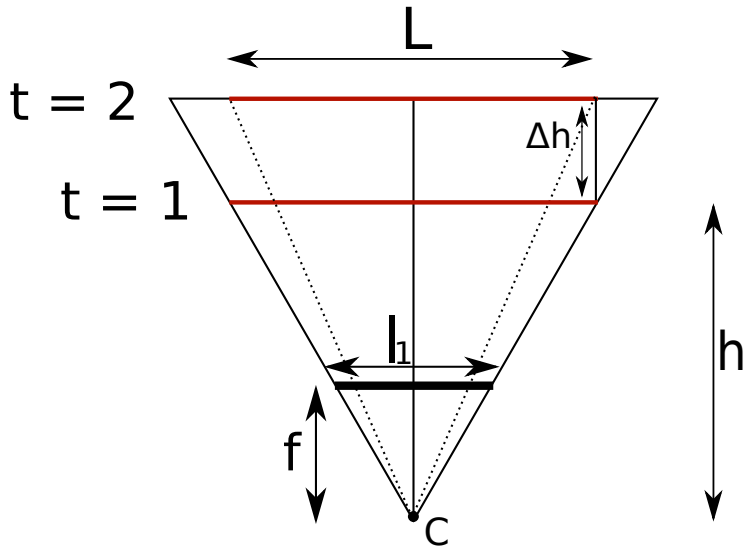


Figura 8.6: *Variazione della scala di un target in allontanamento dalla telecamera.*

In definitiva, fissata la velocità del target (e quindi Δh) si ha che, dal punto di vista della telecamera, esso avrà una scala che varia con un velocità funzione della sola distanza h. In figura 8.7 è raffigurato l'andamento di Δl in funzione della distanza h, ipotizzando una velocità costante pari a 60 km/h.

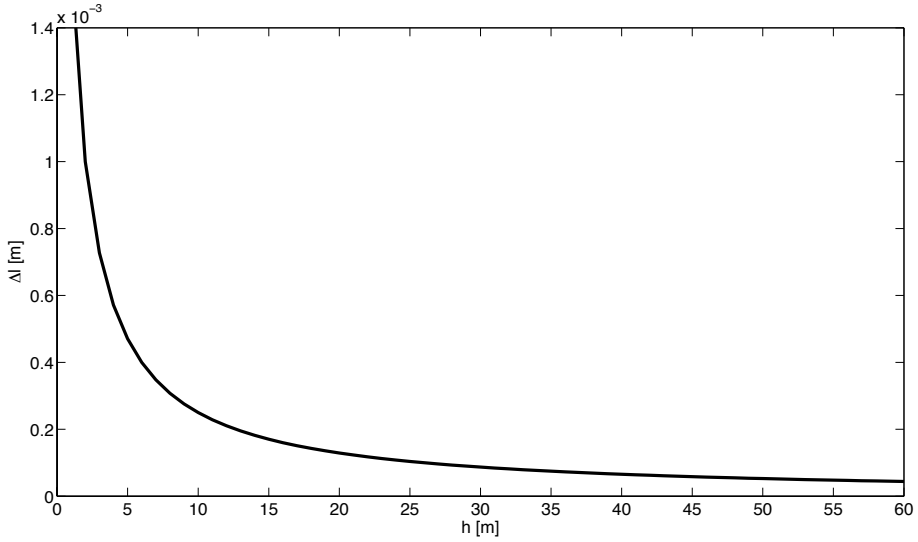
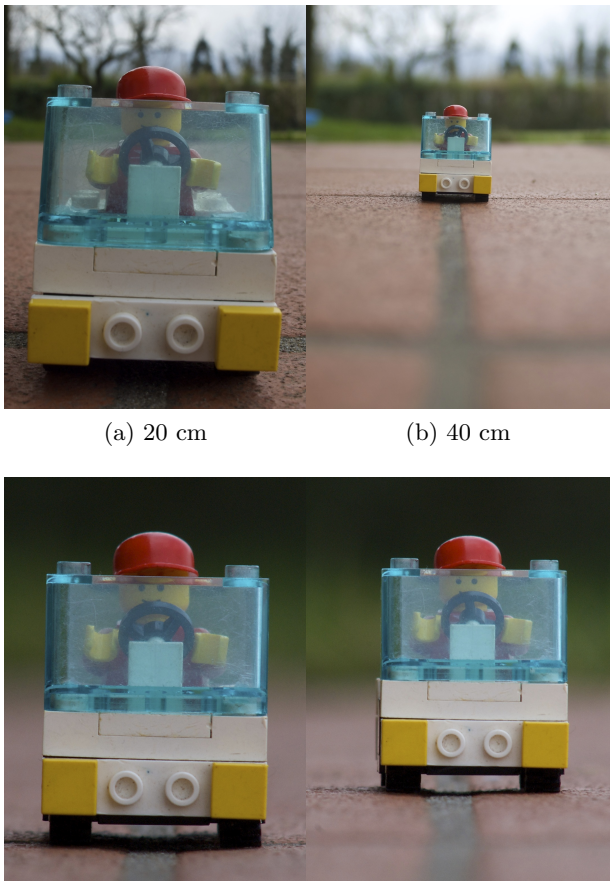


Figura 8.7: *Variazione della dimensioni di un target sul sensore in funzione della sua distanza dalla telecamera. Il Δt considerato è pari a $\frac{1}{25}$ s.*

In figura 8.8 vi è un esempio del fatto appena discusso. È stato ripreso un modellino in allontanamento dalla telecamera, in alto a sinistra ad una distanza iniziale di 20 cm dal piano immagine e in basso a sinistra ad una distanza iniziale di 150 cm. La lunghezza focale è mantenuta costante tra

le foto di destra e quelle di sinistra, e scelta in maniera tale da inquadrare inizialmente il target allo stesso modo. Lo spostamento del target tra le immagini di sinistra e quelle di destra è in entrambi i casi di $\Delta h = 20cm$. È evidente come (a parità di spostamento Δh) le dimensioni del target nell'immagine, inizialmente identiche, variano considerevolmente nel primo caso e molto poco nel secondo.



(a) 20 cm (b) 40 cm

(c) 150 cm (d) 170 cm

Figura 8.8: *Differenza di variazione delle dimensioni di un target ripreso prima ad una distanza iniziale di 20 cm (in alto) e di 150 cm (in basso) e dopo che si è allontanato di $\Delta h = 20$ cm*

Questo risultato consente di modulare l'aggiornamento della scala in funzione del livello di zoom. In dettaglio conviene variare il parametro λ in funzione del livello di zoom: da $\lambda_{max} = 1$ per $zoom = 1$ fino a $\lambda_{min} = 0.1$ per $zoom \geqslant 18$. I valori fissati sono arbitrari ma ragionevoli. Anche in questo ambito avere un algoritmo di stima delle dimensioni del target permetterebbe di trovare dei parametri più precisi.

Non si sono però variate le dimensioni delle finestre di ricerca ($\pm 10\%$ rispetto alla scala precedente) perchè cambierebbe il comportamento dell'algoritmo. Ad esempio cercare ad una scala del $\pm 1\%$ comporterebbe una differenza molto ridotta tra i tre istogrammi, pertanto la la scala migliore rimarrebbe quella precedente. Dall'altra parte aumentando la percentuale al $\pm 20\%$ si inserirebbe un margine di incertezza troppo elevato.

Un ultima considerazione si può fare sulla crescente difficoltà che si deve affrontare in un tracking PTZ di un target molto vicino alla telecamera. Infatti in questo caso la velocità con cui si muove il brandeggio è più elevata (con maggiori problemi in fase di accelerazione e decelerazione) e allo stesso tempo aumenta notevolmente l'incertezza sulle dimensioni.

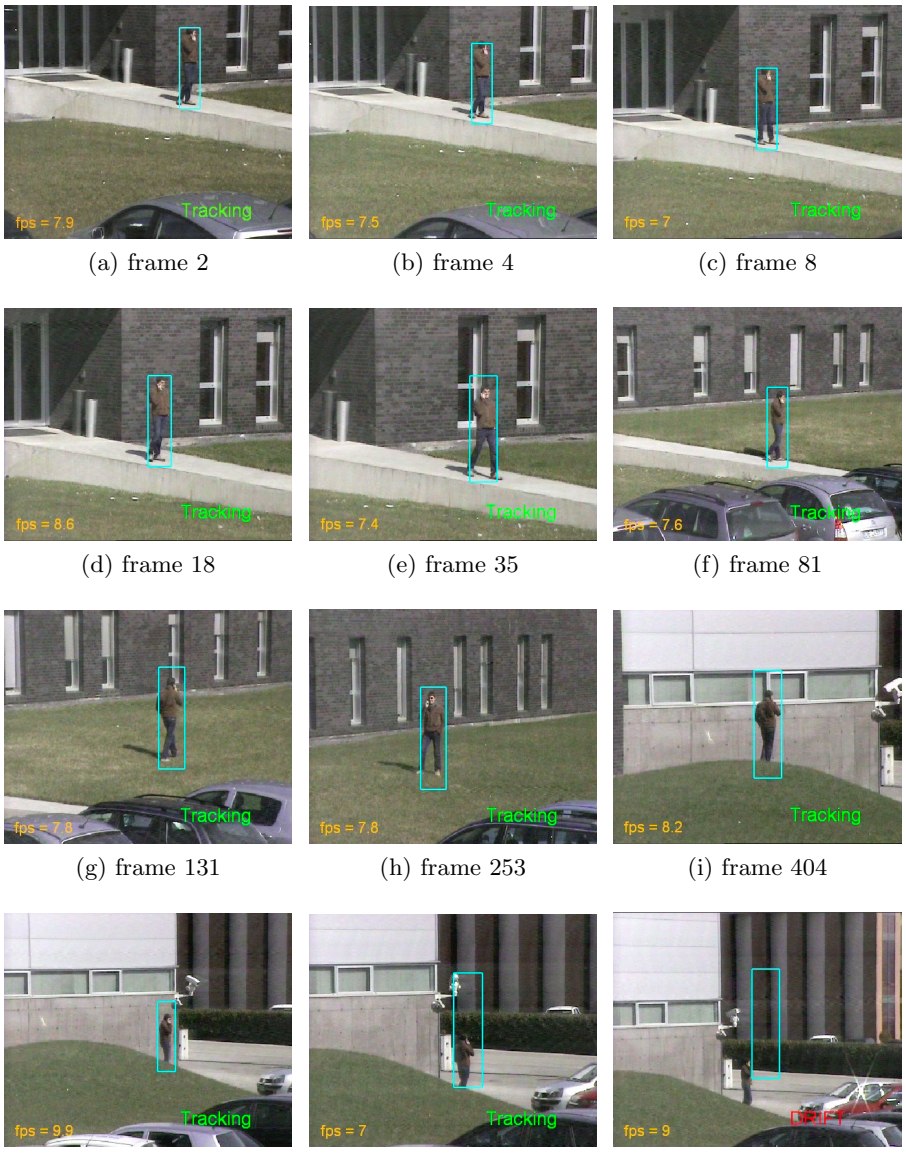
8.6 Risultati

L'algoritmo si adatta molto facilmente all'utilizzo in un sistema reale, come il brandeggio Ulisse Compact. La selezione del target da inseguire è effettuata a mano al primo frame, quindi in questo tipo di test è indispensabile che il target sia fermo nel momento in cui comincia il tracking. Il framerate raggiunto è di circa 8 fps, comprensivo del tempo di acquisizione del frame e del tempo necessario a scrivere sulla porta seriale.

I risultati ottenuti sono molto buoni quando il background è abbastanza uniforme, riuscendo a inseguire il target per decine di secondi. Anche il controllo della telecamera si è dimostrato adatto ad ottenere un tracking fluido e piacevole da vedere. Si è potuto constare la robustezza dell'algoritmo ai cambi di posa e l'efficacia delle scelte effettuate per il controllo del brandeggio.

Maggiori difficoltà rispetto al data set si sono riscontrate dove lo sfondo è complesso. Ad esempio si sono avuti casi di drift con una persona che cammina vicino a delle auto parcheggiate. L'utilizzo dell'algoritmo MSSD ha consentito, nonostante il framerate più basso, di ridurre la frequenza di drift dovuti a questo problema. Lo sfondo complesso rende anche meno efficace il drift detector in quanto si trovano con più facilità aree simili al target (che quindi attirano il bounding box) ma di dimensioni non tali da allarmare il drift detector.

Nelle figure nelle pagine seguenti sono illustrati in dettaglio due test di tracking in tempo reale effettuati con il brandeggio Ulisse Compact.



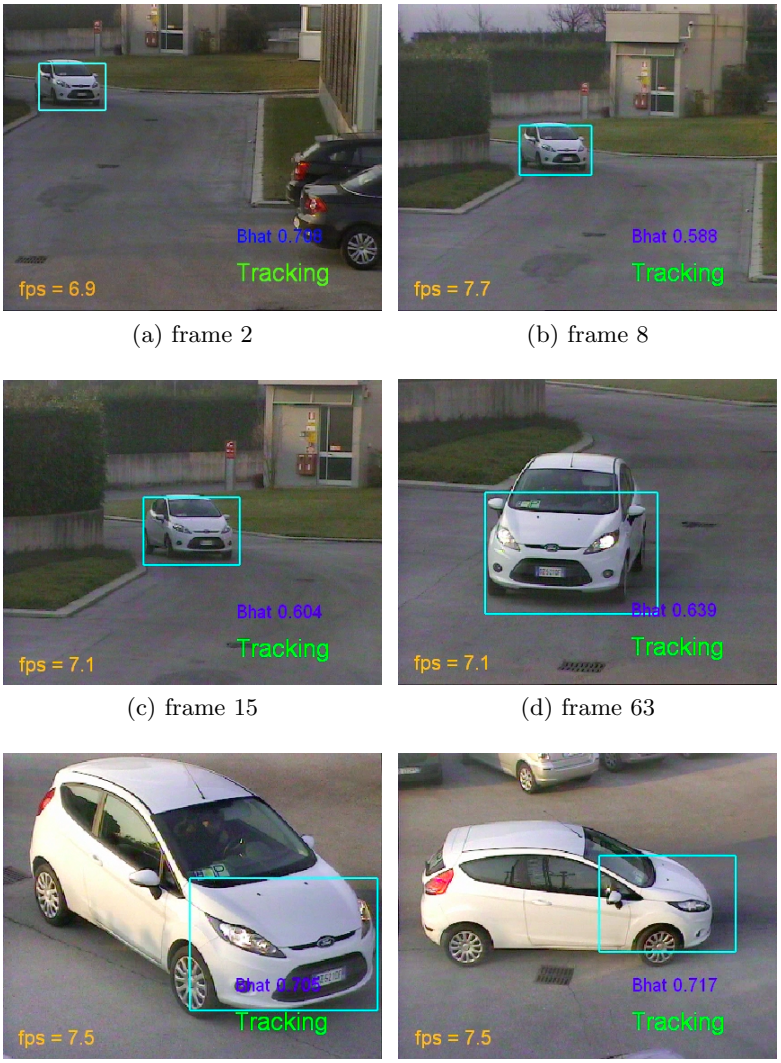
(a) frame 2 (b) frame 4 (c) frame 8

(d) frame 18 (e) frame 35 (f) frame 81

(g) frame 131 (h) frame 253 (i) frame 404

(j) frame 443 (k) frame 460 (l) frame 483

Figura 8.9: Algoritmo MS in real time. Persona.
Tracking in real time con Ulisse Compact, durata reale 31 secondi. La persona è selezionata manualmente al primo frame. Dal frame 2 al frame 8 si vede come la telecamera PTZ venga spostata in modo da centrare il target. Al frame 18 è evidente anche il movimento dello zoom, che è regolato in base alle dimensioni del soggetto. Al frame 81 il soggetto (che è in movimento) viene ripreso con un'inquadratura più larga. Al frame 404 il target è molto più lontano dalla telecamera rispetto all'inizio del tracking, ma rimane con le giuste proporzioni, segno che la scala è cambiata in maniera corretta. Dal frame 443 è evidente il rumore del segnale video. Combinato con i colori dell'edificio simili a parte del target, l'algoritmo è ingannato in primo luogo sulle dimensioni. Nel giro di 20 frame il drift viene segnalato e il tracking si interrompe (frame 483, in basso a destra). La durata reale del test è di 61 secondi.

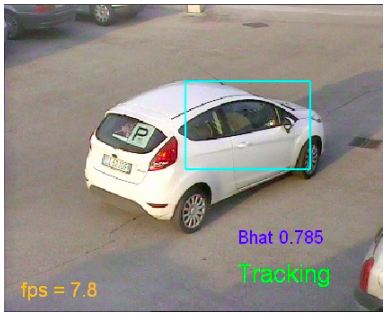


(a) frame 2 (b) frame 8

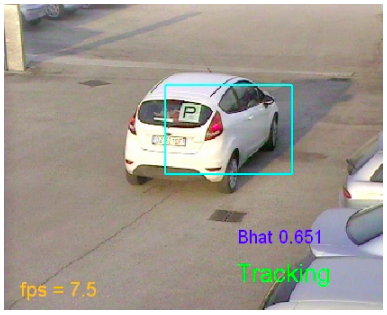
(c) frame 15 (d) frame 63

(e) frame 80 (f) frame 117

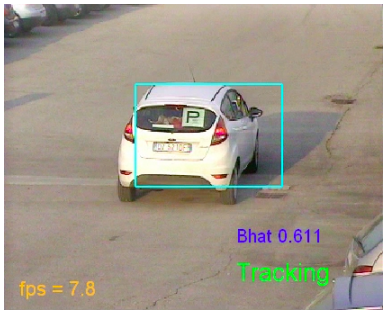
Figura 8.10: Algoritmo MS in real time. Veicolo Parte 1.
Tracking in real time con Ulisse Compact. Il veicolo, inizializzato a mano, si trova in una zona d'ombra. Dal frame 63 al frame 117 passa nelle vicinanze della telecamera, nei quali il bounding box rimane fissato alla parte anteriore dell'auto a causa del sovrapporsi di 3 criticità. Innanzitutto il cambio di illuminazione sulla fiancata, con una forte variazione del colore dell'auto che impedisce al bounding box di espandersi. In secondo luogo il cambio di posa, con conseguente variazione delle proporzioni del target. Infine la vicinanza del target alla telecamera (meno di 4 metri) che comporta un veloce aumento delle dimensioni sul piano immagine seguito da una altrettanta veloce riduzione. Il controllo implementato consente tuttavia di tollerare le imprecisioni del tracker, infatti il veicolo è sempre contenuto nel frame consentendo la prosecuzione del tracking.
Segue parte 2.



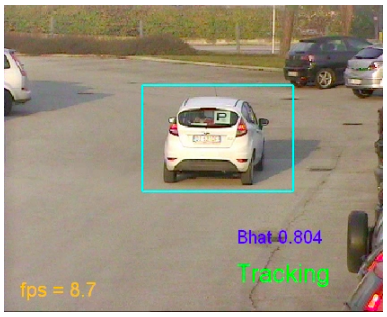
(a) frame 131



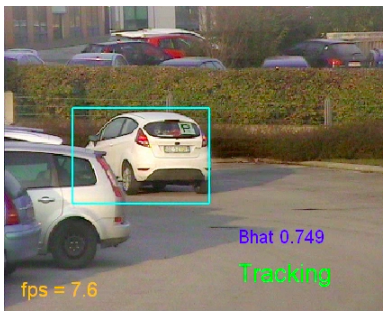
(b) frame 140



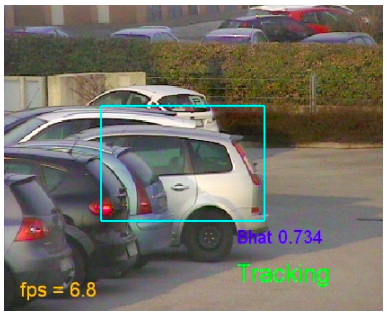
(c) frame 161



(d) frame 184



(e) frame 235



(f) frame 246

Figura 8.11: Segue dalla pagina precedente. Veicolo Parte 2.
Quando il target si allontana dalla telecamera (frame 140) il tracking continua correttamente nonostante il cambio di illuminazione, questo grazie all'aggiornamento del modello (in tal caso abbastanza veloce perché consentito da un $Bhat_{IntExt}$ *basso).*
Dal frame 235 il target è occluso dalle auto parcheggiate e dal frame 246 il bounding box si sposta nel punto dell'immagine più simile al modello, ovvero l'auto grigia e lì rimane. Questo è uno dei casi di drift che il drift detector non è in grado di segnalare, in quanto la distanza tra l'istogramma del background e quello del foreground rimane elevata e allo stesso tempo vi è un buon match con il modello. In più basti confrontare l'aspetto del target nel primo frame con quello del veicolo che ha causato il drift. L'aggiornamento del target non può essere infatti così veloce da ridurre in maniera significativa i bin relativi ai frame passati.

8.7 Problematiche rilevate

Si sono portati alla luce anche altri limiti sia dell'algoritmo che della telecamera PTZ. Le maggiori difficoltà rilevate sono le seguenti:

- *Video rumoroso*: Il video acquisito dal frame grabber presenta molto più rumore rispetto a quello dei video del data set. In particolare si tratta di rumore colorato che va ad influire sugli istogrammi, diminuendone le componenti nulle e rendendo di fatto più simili tra loro gli istogrammi del background e del foreground. Il fatto è evidenziato dall'indice $Bhat_{IntExt}$ che assume valori molto più elevati rispetto a quelli del data set in situazioni paragonabili. Ciò comporta un decadimento delle prestazioni del tracker e una minore efficacia del drift detector.

  Il rumore è causato dalla trasmissione su lunga distanza (centinaia di metri) del segnale analogico. Il segnale si degenera anche a causa di interferenze dovute alla rete elettrica. Sono presenti inoltre disturbi che talvolta deformano l'immagine.

Figura 8.12: *Ritaglio di un frame che riassume i problemi principali avuti durante le prove in real time con la telecamera PTZ. In alto si notano sia il disturbo che il rumore del video. In basso, sul finestrino dell'auto parcheggiata, si vede il riflesso dovuto alla luce incidente del sole.*

  L'assenza di rumore nei video del data set è dovuto, oltre al fatto che il video non viene trasmesso in analogico, alla compressione, la quale elimina in gran parte l'effetto dei disturbi.

- *Video interlacciato*: il video trasmesso dalla telecamera è interlacciato. Per deinterlacciare si dimezza il numero di righe (e per non modificare le proporzioni dell'immagine si dimezza anche il numero di colonne). Ciò riduce di un quarto il numero di pixel utili all'algoritmo. Il problema maggiore è nel momento in cui si va a costruire il modello del target. Un modello basato su pochi pixel (meno di 2000) è poco significativo. Lo

stesso discorso vale nei primi istanti di tracking, quando il livello di zoom è ancora basso e il target occupa un'area ridotta del fotogramma: una finestra di ricerca piccola rende l'algoritmo poco preciso fino a creare problemi di stabilità.

- *Frequenza massima di scrittura sulla seriale*: la massima frequenza di scrittura sulla seriale è 12.5 Hz (1 scrittura ogni 80 ms). Questo implica che il limite superiore al framerate ottenibile con il sistema dato è di 12.5 fps, non sufficienti se si vogliono inseguire target veloci. A causa di questo problema non è nemmeno possibile interrogare la telecamera per conoscerne la posizione. Sapere le coordinate di pan e tilt con precisione è fondamentale se si inserisce l'algoritmo di tracking in un sistema di più telecamere calibrate.

- *Frequenza massima per lo zoom*: nel brandeggio Ulisse Compact possono essere dati al massimo 3 comandi al secondo per lo zoom. Se vengono dati con frequenza maggiore lo zoom rimane fermo. Questo problema, che non c'è invece per le velocità dei motori, comporta un andamento a scatti dell'area inquadrata.

- *Basso framerate*: il tempo di elaborazione di un fotogramma è dato dalla somma del tempo di acquisizione e del tempo di esecuzione dell'algoritmo e vale mediamente 120 ms (8 fps). L'algoritmo è però tarato su un framerate di 25 fps. Un basso framerate implica l'impossibilità di inseguire i target più veloci. Per risolvere i problemi legati al basso framerate bisogna accontentarsi di inquadrare la scena con un angolo più grande (e quindi ridurre il livello di zoom), rinunciando ad avere il target in primo piano.

- *Centratura del target*: Ci si è imposti come vincolo quello di avere un controllo senza sovraelongazioni. Per far ciò il centro di un target in movimento non può coincidere con il centro del frame. Una soluzione a questo problema potrà esser fornita da un metodo per stimare le coordinate del target e la sua velocità nello spazio ambiente.

Capitolo 9

Conclusioni

Il lavoro di tesi è la parte principale di un progetto più ampio, che vede il tracking di persone e veicoli come un aiuto per gli addetti alla videosorveglianza. Esso è composto da un primo blocco che ha il compito di rilevare gli eventi anomali. Un secondo blocco ha il compito di scegliere il target da inseguire e di segmentarlo, così da costruire un modello per il tracking. Il terzo blocco è costituito dall'algoritmo di tracking, come quello basato sul mean shift proposto in questa tesi. Un quarto blocco consiste nel drift detector, che ha il compito di segnalare la perdita del target al fine di reinizializzare il tracking. Infine l'ultimo blocco consiste nel controllo della telecamera PTZ che ha il compito di inquadrare in maniera fluida l'area richiesta, ma anche quello di agevolare il tracking vero e proprio.

L'algoritmo di tracking è il blocco che richiede la maggiore affidabilità a causa della difficoltà intrinseca del compito che deve svolgere. Per questo è stata la parte più studiata e quella che ha richiesto un lungo lavoro di messa a punto. Si è partiti da una tecnica ben collaudata e la si è adattata allo scopo sia con idee già presenti in letteratura, sia con soluzioni originali. L'algoritmo implementato si è dimostrato molto efficace nella maggior parte dei video e nelle prove con la telecamera PTZ. Al tempo stesso esso ha una ridotta complessità computazionale, caratteristica necessaria per un sistema in real-time.

In fase di studio è fondamentale avere un data set di video rappresentante le situazioni principali e le problematiche più diffuse in relazione all'applicazione. Si è costruito un data set composto da più di 23000 frame, tutti abbinati alla ground truth annotata attraverso un apposito toolbox. Il data set è stato organizzato in 8 categorie, in modo da agevolare il confronto tra algoritmi diversi e avere così una valutazione globale, ovvero non influenzata dalle criticità del singolo video. I video saranno una buona base anche per i futuri studi sul tema.

Il drift detector si è dimostrato efficace nella maggior parte delle situazioni, ed è caratterizzato da un numero esiguo di falsi allarmi. Allo stesso tempo non va ad appesantire l'algoritmo perché sfrutta dati e informazioni in gran parte già disponibili per il funzionamento del tracker. Si è poi trovato un indice che è in grado di segnalare l'affidabilità dell'algoritmo di tracking ed è stato sfruttato per adattarne il comportamento in base alle caratteristiche del video. Lo stesso indice sarà molto utile in lavori futuri che cercheranno di combinare diverse tecniche di tracking.

Di tutti i lavori presenti in letteratura riguardanti algoritmi di tracking, solo un numero esiguo considera l'utilizzo dello stesso con una telecamera PTZ. Nel lavoro di tesi è presentato un metodo per il controllo del brandeggio e si è riusciti nell'intento di avere un movimento fluido della telecamera e allo stesso tempo veloce per soddisfare le esigenze del tracker. Sono state proposte alcune soluzioni che si avvantaggiano della conoscenza dei parametri intrinseci della telecamera, integrando in questo modo tra loro l'algoritmo di tracking con l'hardware della telecamera. Il tutto è stato verificato tramite test in real-time con il brandeggio Ulisse Compact in situazioni realistiche, ottenendo buoni risultati sia sul tracking di veicoli che sul tracking di persone.

9.1 Sviluppi futuri

Al fine di sviluppare un prodotto adatto ad essere commercializzato è necessario studiare ancora molti aspetti del sistema. Analizzare e implementare altri algoritmi di tracking sarà sempre la parte principale del lavoro. Ci sono molti lavori recenti in letteratura che potrebbero contribuire sensibilmente al miglioramento della robustezza del tracking. Un algoritmo di tracking sarà più utile se si è in grado di associare ad esso un indice che ne misuri il grado di attendibilità della stima. Ciò renderà possibile la combinazione efficace di diversi algoritmi. Il modo di combinarli dovrà essere studiato attentamente e dovrà essere valutata l'efficacia in rapporto all'aumento delle risorse computazionali necessarie.

Sono state definite delle linee guida per la costruzione di un data set, in questo modo sarà più semplice integrarlo con nuovi video che rappresentino situazioni attualmente non considerate.

Il sistema di comunicazione tra PC e brandeggio andrà rivisto, in modo da consentire almeno 25 comandi di controllo al secondo. Sarebbe interessante studiare una tecnica per rilevare i cambi di luce e regolare di conseguenza l'esposizione della telecamera, in modo da favorire il tracking e allo stesso tempo mostrare all'operatore un'immagine con la giusta luminosità.

La precisione del tracking ed il suo successo dipendono anche dall'affidabilità del modello iniziale che si costruisce. Se la posizione del target viene passata da un algoritmo di event detection, è da valutare l'impatto che questo avrà sulle prestazioni del tracker.

Dal punto di vista del materiale presente in letteratura si sente la necessità di avere dei metodi standard per poter confrontare soluzioni diverse ad un dato problema. Per quanto riguarda il tracking è evidente la disomogeneità dei video sui quali viene dimostrata l'efficacia di un algoritmo. Si sente l'esigenza di avere dei video annotati, in modo da avere la possibilità di provare la propria soluzione. Contemporaneamente mancano delle metriche globalmente riconosciute che permettano di misurare la qualità di un algoritmo.

Infine lo studio di un sistema di tracking multi-camera potrebbe fornire maggiori garanzie sull'affidabilità del tracking, e risultare per questo un vasto campo di studi per il futuro.

Bibliografia

[1] C. Bibby, I. Reid. *Robust Real-Time Visual Tracking using Pixel-Wise Posteriors*. Proceedings of European Conference on Computer Vision, 2008.

[2] P. Remagnino, G.A. Jones, N.Paragios, C.S. Regazzoni. *Video-Based Surveillance Systems: Computer Vision and Distibuited Processing*. Kluwer, November 2001.

[3] R.T. Collins, A.J. Lipton, T. Kanade, H. Fujiyoshi, D.Duggins, Y.H. Tsin, D.A. Tolliver, N. Enomoto, O. Hasegawa. *A system for video surveillance and monitoring*. CMU-RI-TR, 2000.

[4] G. Gennari, P. Donaggio. *Case Study For Distributed Smart Sensor Network For Surveillance Applications*. "FeedNetBack" EU Project – Grant Agreement num. 223866, 2009.

[5] E. Campana. *Algoritmo di rilevazione eventi per sistemi di videosorveglianza*. Tesi di Laurea Specialistica in Ingegneria Informatica, Università di Padova, 2007

[6] S. Gasparella. *Inseguimento tramite camera mobile con controllo dello zoom per videosorveglianza*. Tesi di Laurea Specialistica in Ingegneria dell'Automazione, Università di Padova, 2009.

[7] P. Salvagnini. *Sviluppo di un sistema di Tracking PTZ per videosorveglianza*. Tesi di Laurea Specialistica in Ingegneria dell'Automazione, Università di Padova, 2009.

[8] A. Soto. *Probabilistic Adaptive Agent Based System for Dynamic State Estimation Using Multiple Visual Cues*. In Proceedings of the International Symposium of Robotics Research, pp. 9-12, 2001.

[9] B. Stenger, T. Woodley, R. Cipolla. *Learning to Track with MUltiple Observers*. Computer Vision and Pattern Recognition, pp. 2647-2654, 2009.

[10] S. Baker, R. Gross, I Matthews. *Lucas-Kanade 20-years on: A unifying framework*. International Journal of Compter Vision, pp. 1-21, 30-39, 2004.

[11] P. Dollàr. http://vision.ucsd.edu/ pdollar/toolbox/doc/index.html

[12] D. Comaniciu, V. Ramesh, P. Meer. *Kernel-based object tracking.* IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 25, No. 5, pp. 564-575, 2003.

[13] D. Comaniciu, V. Ramesh, P. Meer. *Real-time Tracking of Non-Rigid Objects using Mean Shift.* IEEE, 2000.

[14] K. Fukunaga, L. Hostetler. *The estimation of the gradient of a density function, with application in pattern recognition.* IEEE Transactions on Information Theory, vol. 21, pp. 32-36, 2 1975.

[15] D. Comaniciu. *Non Parametric Robust Methods For Computer Vision.* PhD Thesis, The State University of New Jersey, 1 2000. pp 5-17.

[16] Q. Zhao, Z. Yang, H. Tao. *Differential Earth Mover's Distance with Its Applications to Visual Tracking.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, pp. 274-287, 2010.

[17] P. Perez, C. Hue, J. Vermaak, M. Gangnet. *Color Based Probabilistic Tracking.* Springer-Verlag Berlin Heidelberg, pp. 665-670, 2002.

[18] R. T. Collins. *Mean-shift blob tracking through scale space.* Tech. rep., Carnegie Mellon University. pp. 3-6.

[19] Vladimir Nedovic, *Tracking moving video objects using mean-shift algorithm.* Project report.

[20] S. Nejhum, J. Ho, M.H. Yang. *Visual tracking with histograms and articulating blocks.* CVPR08, pp. 1-8, 2008.

[21] R.Valenti, F. Haegeloh. *Implementation and evaluation of the mean shift tracker.* Project report.

[22] W. Hu. *Implementation and discussion about ensemble tracking algorithm.* cs.sunysb.edu, 2008.

[23] B. Gorry, Z. Chen, K. Hammond, A. Wallace, G. Michaelson. *Using Mean-Shift Tracking Algorithms for Real-Time Tracking of Moving Images on an Autonomous Vehicle.* World Academy of Science, Engineering and Technology, pp. 209-214. 2008.