UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

ALGORITMO BASATO SU ISTOGRAMMI DI COLORE PER L'INSEGUIMENTO VISIVO DI OGGETTI IN RETI DI TELECAMERE

Relatore:
Prof. Emanuele Menegatti

Laureando:
Diego Bruno Zabotto

Anno Accademico 2010/2011

Indice

Sommario

Il lavoro di questa tesi ha come scopo la realizzazione di un sistema di *visual tracking* per telecamere mobili in grado di inseguire oggetti qualsiasi e che sia robusto ai cambi di posa e alle occlusioni, cercando il target nel caso questo venga occluso totalmente. L'inseguimento automatico di un soggetto è una caratteristica importante per un sistema di videosorveglianza, ambito che al giorno d'oggi ha acquisito una particolare rilevanza sociale. Il progetto è partito dalla costruzione di un data set di video che fossero rappresentativi allo scopo, ovvero, che presentassero scene in cui il soggetto inseguito cambiasse evidentemente posa o fosse occluso parzialmente o completamente. A codesti video sono abbinati dei file contenenti le coordinate del target, così da poter facilmente verificare l'efficacia del sistema. Si sono analizzate alcune delle tecniche di tracking esistenti e si è sviluppato un algoritmo basato su istogrammi di colore corredato da una serie di accorgimenti che hanno permesso di raggiungere lo scopo prefissato. È stato messo a punto un sistema per il rilevamento di situazioni di drift che segnala la maggioranza dei casi in cui il soggetto inseguito viene perso. È stato implementato un meccanismo che permettesse al sistema di ritrovare la traccia nel caso questa venisse persa e studiato un metodo per calibrare i colori di differenti telecamere così da facilitare anche il ritrovamento del target in altri flussi video. Infine, il funzionamento del sistema proposto è stato verificato anche per video in real-time.

Capitolo 1

Introduzione

Con il termine *visual tracking* ci si riferisce al processo di inseguimento di uno o più oggetti che si muovono nel tempo utilizzando una telecamera. Un algoritmo di di questo tipo analizza i frame di un video e restituisce come uscita la posizione dell'oggetto cercato, detto *target*.

Gli ambiti di utilizzo di questa particolare tecnica appartenente al campo della visione computazionale sono molteplici, dall'automazione industriale all'intrattenimento, dalla videosorveglianza all'interazione uomo-macchina. Questa vasta diffusione è stata favorita dalla sempre più crescente disponibilità di reperire sul mercato sistemi in grado di processare flussi video in tempo reale e con costi sempre più ridotti.

In tempi in cui la sicurezza sociale è sempre più importante, il tracking è un aspetto fondamentale per i sistemi di videosorveglianza cosiddetti "intelligenti". Questo lavoro di tesi si inserisce proprio in questo settore, per il quale esistono già numerose applicazioni che prevedono principalmente l'individuazione e la continua determinazione della posizione di persone, oggetti o veicoli. I motivi che innescano questo processo di tracking possono riguardare, ad esempio, la presenza di individui all'interno di un'area vietata la cui costante localizzazione permette di monitorarne gli spostamenti o l'osservazione di un soggetto ai fini della sua sicurezza evitando situazioni potenzialmente pericolose.

Con questo lavoro di tesi si continuano le ricerche che nell'ultimo decennio hanno sviluppato una varietà di algoritmi di tracking, le quali hanno portato ad un miglioramento dell'affidabilità e dell'accuratezza di questo processo.

1.1 Obiettivo

L'obiettivo di questo lavoro è stato sviluppare un algoritmo per il *long term tracking* da installare in un ambiente multicamera, che sia il più robusto possibile alle

occlusioni sia parziali che totali e che sia efficiente da funzionare in un contesto real-time. In altre parole, ideare un algoritmo di tracking che preveda un meccanismo di ricerca del target in seguito ad una occlusione totale o in fase di re-identificazione dello stesso, sfrutti il fatto di avere a disposizione più di una telecamera e riesca ad elaborare il più alto numero di frame per secondo.

1.2 Struttura della tesi

La tesi è strutturata nel seguente modo. Nel capitolo 2 viene illustrato lo stato dell'arte inerente agli algoritmi di tracking e alla calibrazione dei colori tra telecamere; sono descritte le principali tecniche utilizzate ed evidenziati i relativi pregi e difetti. Nel capitolo 3 viene trattato in dettaglio l'algoritmo di tracking sviluppato, con particolare attenzione su quelle che sono le principali caratteristiche. Nel capitolo 4 viene introdotto il problema della re-identificazione di un soggetto catturato da diverse telecamere e proposta una possibile soluzione tramite la calibrazione dei colori. Nel capitolo 5 sono infine presentati i risultati sperimentali.

Capitolo 2

Lo stato dell'arte

In questo capitolo si illustrano alcune delle tecniche di tracking presenti in letteratura; tutte queste prevedono la costruzione di un modello del target, ma ciò che le rende differenti tra di loro è il tipo di modello e la modalità di inseguimento del soggetto.

Per modello del target si intende una sua rappresentazione basata su determinate caratteristiche dello stesso. Questa struttura però può non essere esaustiva, ovvero, può capitare che il modello non sia sufficiente ad identificare il target in modo univoco; inoltre, basandosi solo sulla percezione visiva può risultare arduo determinare una serie di caratteristiche uniche per ogni soggetto. Alla luce di ciò, ogni tecnica presenta un punto debole legato al tipo di modello scelto per il target e questo si traduce, per l'algoritmo di tracking, nella possibilità di stimare erroneamente la posizione o le dimensioni del soggetto inseguito. Il fatto che un software fallisca è del tutto normale, in quanto, a differenza dell'uomo, presenta una flessibilità difficile da riprodurre. Di conseguenza ogni buon algoritmo deve prevedere degli automatismi che dovrebbero gestire il caso appena descritto, al fine di evitare che questo restituisca in output informazioni errate.

Gli algoritmi di tracking più diffusi si basano su tecniche note come *Particle Filter*, *Kalman Filter*, *Template Matching*, *Background Subtraction*, *Object Detection* e istogrammi. Alcuni poggiano su un mix di tecniche così da migliorarne l'affidabilità andando a sopperire ai difetti di un approccio con l'altro.

Il Particle Filter [9] presenta un alto costo computazionale che lo rende poco adatto per applicazioni realtime. Al contrario, il Kalman Filter [16] è molto veloce ma risulta inefficace quando il soggetto da inseguire descrive traiettorie difficilmente predicibili. Il Template Matching [14] è poco funzionale nel caso di frequenti cambi di posa o scala del target. Il Background Subtraction [15] è abbastanza veloce e preciso nell'individuare una possibile sagoma del soggetto, ma non prevede alcun meccanismo di distinzione tra sagome e non può essere usato con telecamera mobile.

L'Object Detection [13], [12] ha un costo computazionale elevato e come il precedente non riesce a distinguere due soggetti, inoltre affinché sia efficace deve essere specializzato su un ben preciso oggetto. Con gli istogrammi si ottiene un algoritmo veloce a patto che questi non siano eccessivamente grandi; possono riguardare l'orientazione degli edge, la texture o i colori; non sono molto indicati quando l'ambiente assume un'aspetto simile al target anche quando questo si presenta con una composizione spaziale differente.

Questa breve panoramica mostra come sia difficile trovare un metodo o una combinazione di più metodi che sia efficace in tutte le situazioni.

Al momento di sviluppare un sistema di tracking, la scelta delle metodologie da adottare dipende ovviamente dallo scopo del tracking e dall'ambiente di lavoro nel quale si inserirà il sistema. Per quanto concerne questo lavoro, l'obiettivo era di sviluppare un sistema per l'inseguimento visivo di oggetti generici la cui forma può cambiare nel tempo e utilizzando come sorgente video telecamere mobili. Alla luce di ciò, si sono dovute scartare tecniche come Background Subtraction, Kalman Filter e Object Detection, in quanto aventi caratteristiche incompatibili con l'ambiente di lavoro. Successivamente si è escluso anche il Particle Filter, perché le informazioni presenti in letteratura non lo trovano adatto per il realtime, e il Template Matching, perché da [11] è emerso come gli istogrammi diano risultati migliori. Di conseguenza la scelta finale è ricaduta sull'utilizzo di quest'ultimi.

Le tecniche che hanno reso interessante il tracking basato su istogrammi sono due: il *MeanShift* e il *CAMShift*. Entrambi stimano la posizione del target trovando il massimo locale di una distribuzione di probabilità data una densità della stessa (l'istogramma nel caso specifico), ma la seconda, in più, determina altre informazioni utili a stimare anche la dimensione e l'orientazione del soggetto. Proprio questa caratteristica ha portato a preferire il CAMShift, il cui nome sta per *Continuously Adaptive Mean Shift*. Con il termine "Continuously Adaptive" ci si riferisce al fatto che, per ogni frame, la distribuzione di probabilità viene calcolata per un'area dell'immagine le cui dimensioni sono in funzione della finestra di ricerca del target. Questa caratteristica di adattamento al target si traduce anche in una maggior robustezza alle occlusioni e ai cambi di posa del soggetto inseguito.

Questo metodo è stato proposto per la prima volta da [3] sviluppando un sistema di tracking per mani e facce che fungesse da interfaccia con l'utente; esso va a coprire la lacuna del MeanShift [7][6][5] di non poter adattare la finestra di ricerca del target. In [8] viene proposto per il tracking di oggetti genereci, ma non essendo nato per un "general purpose tracking", ovvero, per l'inseguimento visivo di un qualsiasi oggetto, l'algoritmo in questione soffre in presenza di scene complesse come può essere un target molto colorato, un background avente simili colori o cambi di luce sostanziali. In [10] viene appunto proposto uno stratagemma che mira a limitare l'effetto di queste situazioni.

In questo lavoro è stato implementato un sistema di tracking partendo dal codice

della funzione CAMShift presente nella libreria software *OpenCV*[1] e aggiungendo alcuni accorgimenti quali il controllo dello spostamento della finestra di ricerca del target tramite coefficiente di Bhattacharyya, il monitoraggio del livello di affidabilità del tracking e la re-identificazione del target in caso di perdita della traccia utilizzando solo l'informazione sul colore e la relativa collocazione spaziale sul target.

L'operazione di re-identificazione del soggetto inseguito presenta delle difficoltà se effettuata da telecamere aventi campi di visione che non presentano sovrapposizioni; questo è dovuto alle differenti condizioni ambientali nelle quali sono installate e alle dissimili caratteristiche intrinseche delle telecamere stesse. Queste diversità si possono tradurre in una percezione dei colori alterata da telecamera a telecamera. Una possibile soluzione a questo tipo di problema è stata proposta in [17] e in [1] nei quali viene introdotto il concetto di calibrazione dei colori e descritte alcune tecniche per effettuarla. Con questo accorgimento è possibile quindi utilizzare il modello del target calcolato da una telecamera per identificare il soggetto in un diverso flusso video.

[1] http://opencv.willowgarage.com/wiki/

Capitolo 3

L'algoritmo

In questo capitolo sono descritte le principali caratteristiche dell'algoritmo di tracking implementato e illustrate anche le eventuali alternative implementative corredate dai motivi che hanno portato alle scelte effettuate.

3.1 Lo spazio dei colori

Uno spazio dei colori è la combinazione di due elementi: un modello di colore e una appropriata funzione di mappatura di questo modello. Il primo è un modello matematico astratto che descrive una modalità di rappresentazione dei colori come combinazioni di numeri.

Esistono diversi modelli di colore e vario è il contesto applicativo in cui vengono utilizzati. Alcuni dei più impiegati sono l'*RGB*, l'*HSB* o *HSV*, il *Lab* e lo *Y'UV*. Ciascuno di essi presenta differenti caratteristiche, le quali sono determinanti per la scelta della loro adozione.

In questa tesi, lo studio dello spazio di colore da utilizzare mira a trovare il modello che renda l'algoritmo in questione meno sensibile ai cambi di luce; l'attenzione si è concentrata su due in particolare, l'RGB e l'HSB, in quanto, nei lavori presenti in letteratura che trattano un simile argomento, questi sono gli spazi maggiormente impiegati.

L'RGB è un modello di tipo additivo che si basa sui colori rosso (Red), verde (Green) e blu (Blue), da cui appunto il nome RGB; questa scelta dei colori è legata alla fisiologia dell'occhio umano. Questo modello è utilizzato dalla maggior parte dei dispositivi di acquisizione di immagini, oltre ad essere scelto anche per la memorizzazione delle stesse. Nonostante questa vasta diffusione, però, l'RGB non è sufficiente per riprodurre tutti i colori e risulta non percettivamente uniforme[1].

[1]La distanza tra i colori calcolata nello spazio RGB non riflette la reale distanza tra i colori

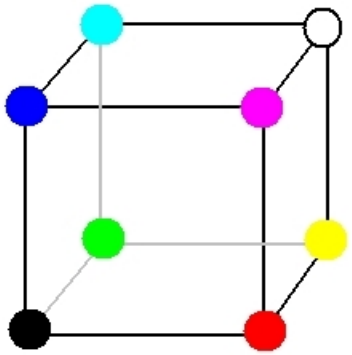


Figura 3.1. Lo spazio RGB rappresentato come un cubo

L'HSB è anch'esso un modello di tipo additivo, ma si basa sui concetti di tonalità (Hue), che definisce il colore stesso; saturazione (Saturation), che indica l'intensità e la purezza; brillantezza (Brightness). Questo modello risulta perciò essere meglio orientato alla prospettiva umana. Viene largamente utilizzato nella computer graphics, ma meno nella computer vision. Uno dei motivi che supporta questa scelta è l'ambiguità nel definire i colori molto scuri e molto chiari. Come si può vedere dalla figura 3.2, il nero può essere ottenuto a partire da una qualsiasi combinazione di tonalità e saturazione unita alla minima brillantezza; il bianco invece lo si può comporre con bassa saturazione, alta brillantezza e una qualsiasi tonalità.

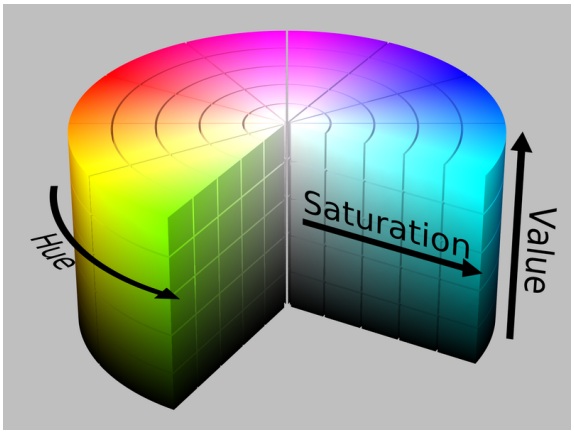


Figura 3.2. Lo spazio HSV rappresentato come un cilindro; la componente V corrisponde alla componente B

percepita dall'occhio umano.

All'interno di questo lavoro sono state eseguite alcune prove con lo spazio HSB tralasciando la componente B, con l'obiettivo di rendere l'algoritmo meno sensibile alla condizione di luminosità. Le immagini in figura 3.3 sono chiari esempi di ambiguità di espressione del colore, che in un contesto come il tracking può essere un fattore altamente penalizzante. Si noti appunto che la giacca nera della signora viene rappresentata come se fosse blu e la parte inferiore dell'auto, chiaramente nera anch'essa, viene considerata come se fosse bianca.

(a) (b)

Figura 3.3. A sinistra l'immagine originale RGB; a destra quella convertita in HSB la cui componente B, per scopi puramente rappresentativi, assume il valore arbitrario 224

L'idea di scegliere comunque lo spazio HSB utilizzando tutte e tre le componenti non porta a vantaggi significativi, inoltre, il costo computazionale dell'algoritmo aumenta in quanto si deve aggiungere il processo di conversione dell'immagine RGB. Per questi motivi, la scelta del modello da utilizzare è ricaduta sull'RGB.

3.2 Il modello del target

In questo algoritmo di tracking il modello del target è rappresentato da un istogramma 3D, una dimensione per ogni canale dello spazio dei colori; esso descrive il target indicando quali colori e in quale percentuale questi lo costituiscono. Per motivi principalmente computazionali, l'istogramma ha una dimensione pari a $16 \times 16 \times 16$ e ciò vuol dire che ogni canale viene quantizzato in 16 valori anziché 256. Contrariamente a quanto si possa pensare, questo notevole calo della profondità di colore non altera esageratamente la qualità dell'immagine (vedi figura 3.4), inoltre funge in parte da filtro: in alcune situazioni, immagini troppo dettagliate possono risultare rumorose e fornire informazioni fuorvianti.

Un modello del target ben costruito è fondamentale per il corretto comportamento dell'algoritmo, dunque, per renderlo affidabile è necessario conoscere con

(a) (b)

Figura 3.4. A sinistra l'immagine originale; a destra quella quantizzata che per scopi rappresentativi è stata calcolata nel seguente modo. Sia $colore_o(x,y) = (r,g,b)$ la tripla che identifica il colore del pixel (x,y) dell'immagine originale; sia invece $colore_q(x,y) = (16 \cdot \lfloor \frac{r}{16} \rfloor, 16 \cdot \lfloor \frac{g}{16} \rfloor, 16 \cdot \lfloor \frac{b}{16} \rfloor)$ il colore del relativo pixel nell'immagine quantizzata

maggior accuratezza possibile quali pixel formano il target e quali invece no. Il primo dei due gruppi in questione costituisce il *foreground*, mentre il secondo definisce il *background*.

In questo lavoro si suppone che l'input per il suddetto processo sia il rettangolo che contiene strettamente l'immagine del target, il cosiddetto *bounding box*. Il bounding box può essere definito manualmente o essere l'output di un certo tipo di *detector*, ma in ogni caso conterrà anche pixel del background.

3.2.1 Segmentazione del target

In questa tesi sono stati testati due metodi di segmentazione del target: il primo si basa su di una tecnica che si riconduce alla risoluzione di istanze del problema di "massimo flusso in un grafo"[2] e che, tramite il teorema "massimo flusso minimo taglio"[3], arriva a definire un probabile contorno del soggetto; il secondo poggia su un procedimento statistico quale la formula della probabilità condizionata di Bayes.

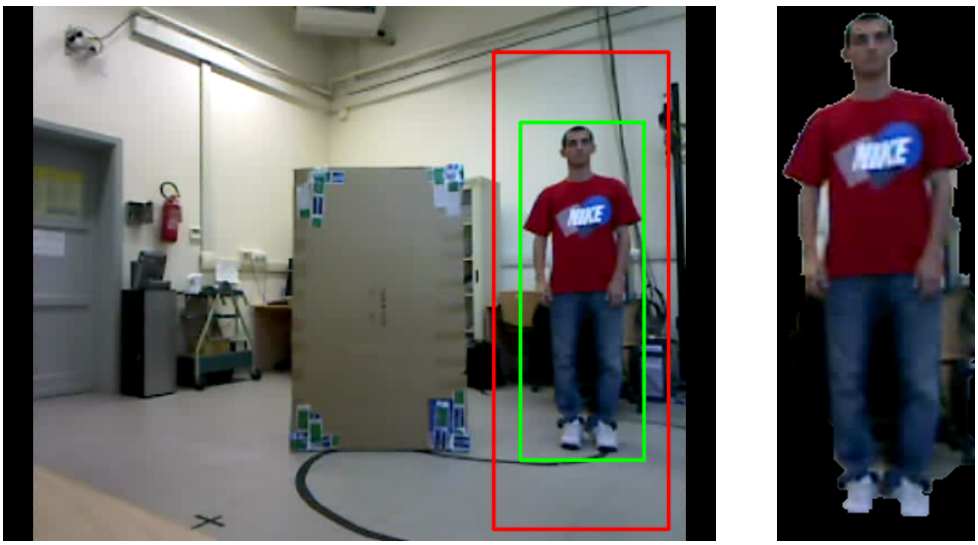
[2]Maximum flow problem: http://en.wikipedia.org/wiki/Maximum_flow_problem

[3]Max-flow min-cut theorem: http://en.wikipedia.org/wiki/Max-flow_min-cut_theorem

L'intervento umano limitato alla sola determinazione del bounding box in fase di inizializzazione del target fa sì che entrambi i metodi ottengano risultati affetti da errori, che in questo caso sono rappresentati da pixel del background considerati come foreground e viceversa. Le prove effettuate hanno comunque mostrato discreti risultati considerato lo scopo per cui sono stati utilizzati.

Il primo dei due approcci è implementato dall'algoritmo *Grab Cut* presente nella libreria software OpenCV che si basa su [4], il quale a sua volta poggia su [18].

Di seguito sono riportati i passi per ottenere la silhouette del target. A partire dal bounding box si determina un ulteriore rettangolo, il *bounding frame* (un rettangolo avente le stesse proporzioni del bounding box, ma scalato in modo che l'area risulti 4 volte maggiore), e si definisce una maschera binaria avente le medesime dimensioni; in corrispondenza dell'area compresa tra la cornice rossa e quella verde (vedi figura 3.5(a)) le viene assegnato un particolare valore che indica che i pixel in questione sono sicuramente background, mentre nella zona del bounding box viene attribuito il valore che esprime la possibile appartenenza al foreground. Viene quindi eseguito l'algoritmo passando come input, tra le altre cose, la porzione di immagine relativa al solo bounding frame, i due rettangoli e la maschera binaria, la quale, opportunamente modificata, costituirà l'output. Al fine di aumentare l'efficacia del metodo, l'immagine di input viene equalizzata così da aumentarne il contrasto. Viene infine calcolato e poi normalizzato l'istogramma di colore della sola zona indicata dalla maschera (vedi figura 3.5(b)).



(a) In verde il bounding box; in rosso il bounding frame (b)

Figura 3.5. Immagini relative alla fase di creazione del modello del target con Grab Cut

Nel secondo metodo si considerano gli istogrammi di colore come distribuzioni di probabilità e sfruttando la regola di Bayes si è in grado di stabilire un peso per ciascun pixel, il quale esprime la probabilità che questo appartenga al target date le componenti del suo colore. La regola di Bayes è:

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

dove:

- $p(f)$ è la probabilità a priori che un pixel appartenga al foreground, definita come rapporto tra l'area del bounding box e del bounding frame;
- $p(x)$ è la distribuzione dei pixel all'interno del bounding box sommata a quella di una cornice circostante;
- $p(x|f)$ è la distribuzione dei pixel all'interno del solo bounding box.

Una volta trovata la matrice dei pesi tramite Bayes (figura 3.6(a)), ne viene poi effettuata una sogliatura (la soglia in fase di inizializzazione è più alta rispetto a quella usata durante il tracking), così da individuare i pixel che serviranno a comporre il modello del target (figura 3.6(b)).

Il motivo per cui sono stati provati due metodi per eseguire lo stesso compito è legato ai rispettivi pregi e difetti.

Il GrabCut presenta il vantaggio di segmentare il target seguendone in qualche modo la forma; in questo modo si evita che si formino dei "buchi" al suo interno che potrebbero incidere negativamente nel calcolo dell'istogramma, inoltre rimane abbastanza fedele a quella che potrebbe essere la segmentazione effettuata manualmente dall'utente. Presenta però anche due svantaggi quali il tempo di computazione e la necessaria precisione nel definire il bounding box come input dell'algoritmo. Per questi motivi si è reso possibile utilizzarlo esclusivamente in fase di inizializzazione, per la quale i vincoli temporali sono meno restrittivi rispetto alla fase di tracking vero e proprio e presumibilmente si dispone di un bounding box più accurato; durante la fase di tracking, infatti, le dimensioni del target posso essere sottostimate, rendendo impossibile una segmentazione accettabile.

La tecnica probabilistica di Bayes, invece, è per certi versi complementare al GrubCut, in quanto la segmentazione risulta frastagliata e presenta buchi all'interno dell'area del target, ma la computazione è più veloce e fornisce dati utili anche quando il target è sottostimato.

Entrambi i metodi, però, risultano particolarmente inefficaci quando, al momento dell'inizializzazione, il background è composto da colori simili a quelli del soggetto da seguire. Di conseguenza i colori del target comuni al background non vengono inclusi nel calcolo del relativo istogramma, creando i presupposti per il fallimento pressoché

(a) In rosso i pixel con alta probabilità; in blu quelli con bassa probabilità

(b) I pixel utilizzati per il calcolo dell'istogramma del modello

Figura 3.6. Immagini relative alla fase di creazione del modello target

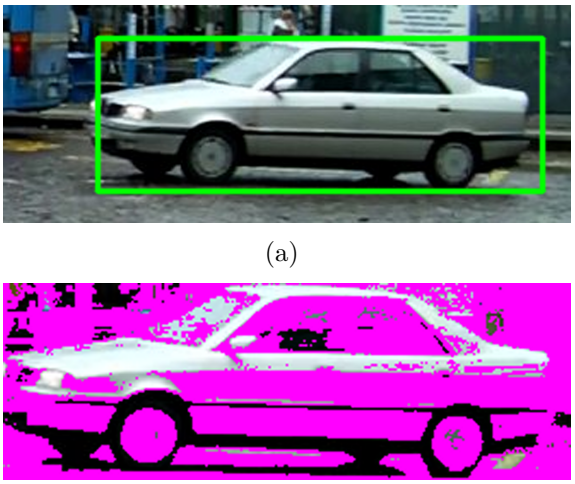
istantaneo del tracking. Nella fase di analisi delle prestazioni (capitolo 5) sono stati quindi utilizzati video in cui il soggetto viene inizializzato in modo sufficientemente corretto. Di seguito si riportano due esempi che documentano la situazione appena descritta.

La decisione in merito a quale tecnica utilizzare è stata presa dopo aver effettuato qualche test su alcuni video; i risultati hanno evidenziato che il metodo probabilistico conduce a migliori risultati, nonostante il Grab Cut fornisca una silhouette più "pulita".

C'è infine da aggiungere che, al fine di prolungare la vita del tracking, è necessario mantenere aggiornato il suddetto modello; questo avviene attraverso una combinazione lineare convessa con l'istogramma del target calcolato al frame corrente: sia $hist_{TG}(t)$ l'istogramma del modello al frame t e sia $hist(t)$ l'istogramma del target calcolato al frame t.

$$hist_{TG}(t) = 0{,}99 \cdot hist_{TG}(t-1) + 0{,}01 \cdot hist(t)$$

Il tipo di sistema di tracking implementato si è rivelato particolarmente sensibile alla percentuale di aggiornamento dell'istogramma target. Si è notato che modificando questo valore si viene a creare una situazione di trade off tra la robustezza ai cambiamenti d'aspetto del target, come i cambi di luminosità, e l'efficacia nel



(a)

(b)

Figura 3.7. Inizializzazione con Bayes

(a) (b)

Figura 3.8. Inizializzazione con Grab Cut

segnalare una situazione di *drift*, ovvero, quando la visione del target viene occlusa o quando c'è somiglianza tra i suoi colori e qualli dell'ambiente che lo circonda. Ovviamente, ipotizzando il caso di tracking in realtime, non si può sempre sapere con precisione e a priori cosa capiterà al target; stabilire, quindi, se il soggetto sta passando da una zona soleggiata ad una in ombra potrebbe servire a modificare in tempo reale la percentuale di aggiornamento così da prolungare maggiormente la vita del tracking.

3.3 Il modello del background

Al fine di aumentare la robustezza del tracking, risulta conveniente modellare il background allo stesso modo del target, ovvero ricorrendo ad un istogramma delle medesime dimensioni. Il procedimento per calcolarlo è semplice e veloce e prevede l'utilizzo dei pixel appartenenti alla sola cornice esterna (l'area compresa tra la cornice rossa e quella verde in figura 3.5(a)) che ha un'area pari a quella del bounding box in fase di inizializzazione del modello e tre volte tanto durante il tracking. Il metodo di aggiornamento, analogo a quello per il modello del target, consiste in una combinazione lineare tra l'istogramma del background al frame precedente e quello della cornice al frame corrente, con un coefficiente di aggiornamento pari a 0,33.

La modellazione del background è utile per determinare il verificarsi di una circostanza che, se non gestita correttamente, potrebbe alterare il comportamento dell'algoritmo. Il responsabile di questo controllo è il *Drift Detector*, il cui funzionamento verrà spiegato successivamente.

3.4 Vertical feature

A volte, la descrizione di un soggetto esclusivamente tramite il suo istogramma dei colori non è sufficiente a garantire l'affidabilità del tracking. Probabili infatti, sono gli scenari in cui i colori del target sono simili a quelli dell'ambiente che lo circonda e questo accade anche per i rispettivi istogrammi. Risulta quindi facile che il bounding box stazioni in un'area scorretta senza che il meccanismo che ne controlla l'affidabilità, cioè il Drift Detector, si accorga dell'errore.

In [1] viene proposto un metodo per analizzare il target che tiene conto non solo dei colori, ma anche della loro posizione. Il descrittore che ne risulta fornisce un'indicazione riguardo il colore più rappresentativo per ogni sezione orizzontale del soggetto trackato, ottenendo quindi una sorta di firma del soggetto stesso. L'analisi viene effettuata lungo l'asse principale del target (testa-piedi), da qui appunto il termine *vertical feature*.

Il procedimento in questione si compone di 4 passaggi:

1. stabilire un valore costante h
2. estrarre dall'immagine del bounding box solo il target
3. scalare l'immagine appena trovata affinché l'altezza sia pari ad h
4. per ciascuna sezione determinare il colore più rappresentativo

Quello che si ottiene è un vettore di h elementi i quali sono le terne RGB che identificano un colore. Le tecniche possibili presenti in [1] per determinare il colore

più rappresentativo sono il calcolo della media e della mediana effettuato per ciascun canale.

Sempre in [1] viene anche proposto un metodo per confrontare due vertical feature: la distanza di Mahalanobis

$$D(VF^1,VF^2) = \sqrt{\sum_{i=1}^{h}\sum_{ch=1}^{3}\left(\frac{VF^1_{(i,ch)} - VF^2_{(i,ch)}}{\sigma_{(i,ch)}}\right)^2}$$

che, rispetto alla distanza Euclidea, tiene conto anche della varianza associata a ciascun elemento da confrontare.

Anche la vertical feature, essendo descrittore di un target i cui colori cambiano nel tempo, necessita di essere aggiornata; la modalità scelta ricade nuovamente su di una combinazione lineare tra la vertical feature attuale e quella calcolata nell'ultimo frame con fattore di aggiornamento pari a 0,05.

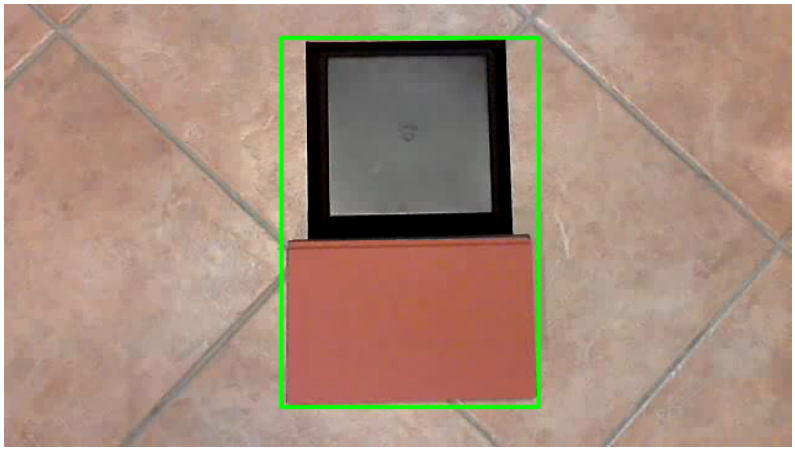
(a) Solo i pixel del target sono usati per il calcolo della vertical feature

(b) La vertical feature risultante

Figura 3.9. Esempio di vertical feature

In questo lavoro, considerata l'alta variabilità del bounding box, il controllo di questa firma non avviene in fase di tracking per monitorare eventuali stazionamenti in aree del frame errate, ma viene usata solo dalla procedura di recovery della traccia (sezione 3.6) per distinguere zone dell'immagine aventi colori abbastanza simili al target ma differente collocazione spaziale.

Le figure 3.10 e 3.11 riportano alcune immagini che mostrano la potenzialità di questo descrittore; si rappresenta infatti una situazione estrema in cui due oggetti hanno gli stessi colori ma collocati in modo differente. Per questo scopo gli oggetti visibili nelle figure 3.10(a) e 3.11(a) devono essere considerati come oggetti distinti. Si noti come le due immagini di probabilità sono sostanzialmente uguali nonostante il target risulti capovolto; risulta quindi evidente che basarsi solo sull'analisi del colore non è sufficiente a distinguere due oggetti aventi colori simili ma in posti differenti. Se non fosse per la vertical feature, la procedura di ricerca della traccia considererebbe come target un oggetto simile ad esso ma fondamentalmente differente.



(a) Oggetto target con relativo bounding box

(b) Pixel effettivi del target

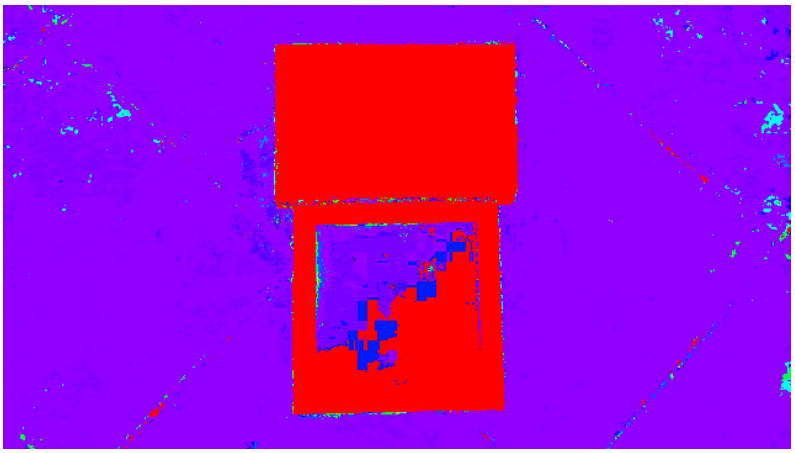
(c) Immagine di probabilità

(d) Vertical feature

Figura 3.10. Immagini relative al tracking di un oggetto nero (parte superiore) e rosso (parte inferiore)

(a) Oggetto simile al target

(b) Pixel effettivi del candidato target



(c) Immagine di probabilità

(d) Vertical feature

Figura 3.11. Immagini relative alla ricerca di un oggetto nero nella parte superiore e rosso in quella inferiore; le immagini 3.10(c) e 3.11(c) sono molti simili in quanto sono evidenziati tutti i pixel aventi i colori del target (nero e rosso) indipendentemente dalla loro posizione nei rispettivi frame; ne consegue che i due oggetti risultano indistinguibili se non fosse per la vertical feature che ne cattura la caratteristica che li differenzia

3.5 Drift detector

Ogni buon sistema di tracking deve dotarsi della capacità di poter rilevare la perdita del target, in quanto si presume che qualunque algoritmo possa fallire nell'inseguimento. Questo però non è l'unico motivo: la perdita del target infatti, può essere causata da occlusioni con altri oggetti o più semplicemente dall'uscita del soggetto dal campo di visione della telecamera.

Con il termine Drift Detector si indica quel meccanismo che permette di dichiarare perso il target o di segnalare una situazione di scarsa affidabilità della stima della sua posizione e in questo lavoro è stato pensato come un insieme di condizioni che sono sufficienti a segnalare il cosiddetto "drift". Una volta che l'algoritmo entra in questo stato, ci rimane fino a quando il target non viene ritrovato (si veda il paragrafo successivo).

Si considerino i seguenti due indici:

Bhat$_{\mathbf{IntExt}}$ indice di Bhattacharyya derivante dal confronto tra l'istogramma del modello background e l'istogramma dell'immagine racchiusa dal bounding box dopo l'esecuzione dell'algoritmo sul frame corrente;

Bhat$_{\mathbf{ModInt}}$ indice di Bhattacharyya derivante dal confronto tra l'istogramma del modello target e l'istogramma dell'immagine racchiusa dal bounding box dopo l'esecuzione dell'algoritmo sul frame corrente.

Il primo serve soprattutto a segnalare il caso in cui il target è troppo simile all'ambiente che lo circonda rendendo così poco affidabile l'algoritmo in quanto può essere distratto da altri oggetti. Il secondo è invece utile per rilevare la maggior parte delle occlusioni totali, in cui il target risulta decisamente differente dall'immagine contenuta nel bounding box.

Come si può facilmente intuire, la disgiunzione di clausole che serve a dichiarare drift riguarda proprio questi due indici. La prima risulta vera quando $Bhat_{IntExt} > 0.91$ per 3 frame consecutivi; la seconda quando $Bhat_{ModInt} < 0.3$ per 3 frame consecutivi. Per quanto riguarda la scelta delle soglie, queste sono il risultato di una serie di test e corrispondono ai valori che fanno mediamente ottenere all'algoritmo i migliori risultati. Le prove sono state effettuate su una serie di video e per questo motivo i valori risultanti non possono essere considerati come assoluti. Invece, i numeri di frame consecutivi minimi sono uguali a quelli presenti in [11]. Il motivo per cui si è ritenuto di dover rideterminare solo le due soglie risiede nel fatto che quest'ultime sono molto più influenzate dal tipo di algoritmo di tracking.

3.6 Ricerca del target

L'obiettivo di questa tesi non è solo quello di sviluppare un algoritmo di tracking, ma quello di implementare un meccanismo che permetta di riprendere la localizzazione del target in seguito ad una interruzione dovuta a drift. Il procedimento in questione conferisce all'algoritmo un automatismo fondamentale per prolungare la vita del tracking, senza che si debba ricorrere all'intervento umano, ovvero, reinizializzare la traccia manualmente.

L'idea di base è quella di cercare all'interno del frame blob di pixel aventi colori simili al target, per poi valutare quanto, queste macchie di colore, si avvicinino all'istogramma del modello.

Una volta acquisito il frame sul quale effettuare la ricerca, ne si calcola l'istogramma dei colori. Successivamente se ne determina un altro che è il risultato del rapporto componente per componente tra l'istogramma del target e quello del frame. Tramite un'operazione di *backprojection* si assegna a ciascun pixel del frame un valore che esprime una sorta di probabilità di appartenenza al target (figura 3.12).

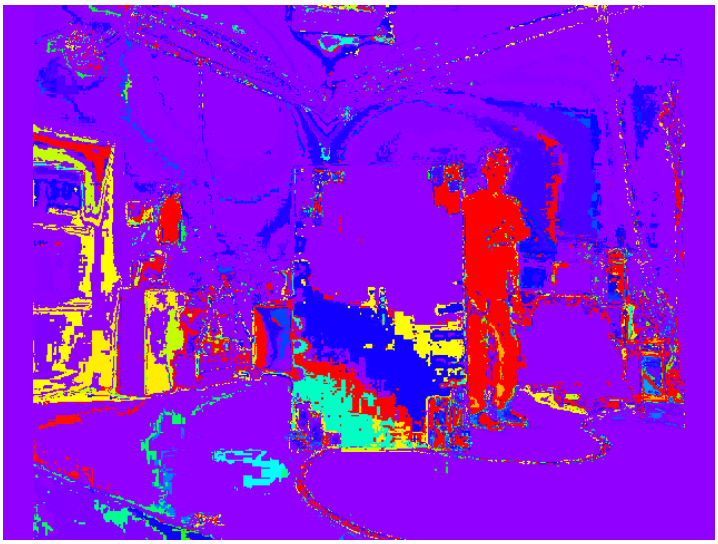


Figura 3.12. Immagine di probabilità; i pixel rossi indicano alta probabilità di appartenere al target, quelli viola bassa

Su quest'ultima immagine si esegue una sogliatura: vengono eliminati tutti i pixel aventi valore minore di 0,6 e a tutti gli altri viene attribuito il valore 1 (vedi figura 3.13); ciò sarà utile, in seguito, nella determinazione di un blob.

Servendosi dell'ultima posizione conosciuta del target si individua un'area dell'immagine concentrica ad essa; casualmente si individua un punto al suo interno avente valore 1 che servirà come partenza per selezionare altri pixel di ugual valore e tra di loro contigui. A quest'ultima immagine, contenente solo il blob di pixel scelti, si applica un'operazione di "apertura", ovvero un'erosione seguita da una dilatazione (vedi figura 3.14).

Ora si calcola l'istogramma di colore del frame per i soli pixel del blob e lo si confronta con quello del modello target. Se l'esito è positivo, si determina il bounding box corrispondente al blob, altrimenti sarà necessario far ripartire il procedimento generando un altro punto di partenza. Per motivi computazionali viene limitato a 5 il numero massimo di ripetizioni dell'algoritmo per ogni frame.

Figura 3.13. Immagine di probabilità dopo la sogliatura

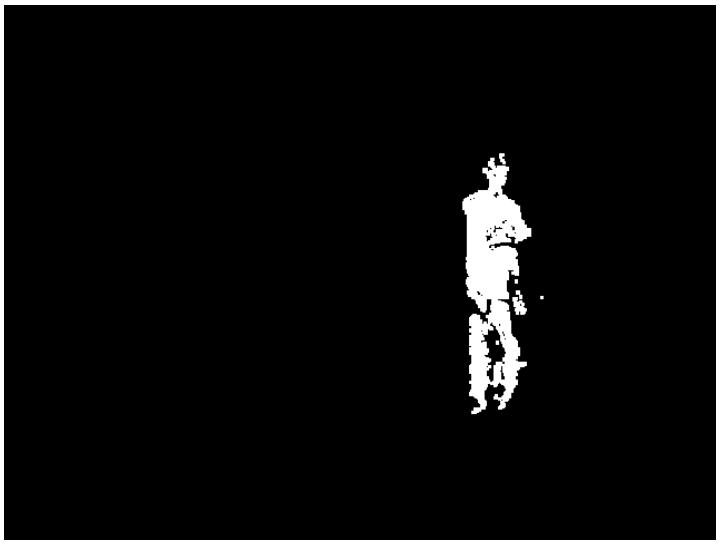


Figura 3.14. Blob di pixel

Per poter però dichiarare che il target è stato presumibilmente trovato, l'area del frame supposta tale deve superare anche il controllo della vertical feature. Se anche questo esito risultasse positivo, allora l'algoritmo terminerebbe restituendo il bounding box, altrimenti si concluderebbe restituendo un rettangolo dall'area nulla. La successiva esecuzione di tale procedura per un nuovo frame vedrebbe come unica modifica quella della grandezza dell'area entro la quale generare il punto di partenza; fintanto che non viene ritrovato il target questa zona aumenta le sue dimensioni così

da ampliare anche il campo di ricerca.

3.7 Il CAMShift modificato

L'algoritmo in questione è il cuore del tracking e, come scritto nel precedente capitolo, è un metodo di stima non parametrica che consente di trovare un massimo locale in una densità di probabilità dato un campione della stessa, ovvero, l'istogramma del target. Si ricorda, anche, che il CAMShift nasce come evoluzione del più semplice MeanShift; ciò che lo distingue è la capacità di modificare le dimensioni della finestra di ricerca in modo autonomo.

Alla luce di ciò, la densità di probabilità usata, assume un ruolo centrale nell'algoritmo; è fondamentale che questa sia definita nel modo più accurato possibile in quanto, oltre ad influenzare lo spostamento della finestra di ricerca, partecipa anche nel determinare le nuove dimensioni. La densità di probabilità viene rappresentata da una matrice, i cui valori esprimono in corrispondenza le probabilità che il relativo pixel appartenga al target dato il suo colore e proprio per questo prende il nome di *Color Probability Density Distribution Image* (a titolo di esempio si veda la figura 3.12).

Le dimensioni di questa matrice, al momento in cui viene calcolata, sono almeno pari a quelle della finestra di ricerca; queste influenzano il comportamento dell'algoritmo, in quanto ne determinano la quantità di informazioni disponibili. Ne consegue che maggiori sono le dimensioni della matrice, maggiore è sia la libertà di movimento che di cambiamento di forma della finestra di ricerca. Questo è il presupposto fondamentale per ottenere una stima abbastanza corretta delle dimensioni del target, ma contemporaneamente aumenta la probabilità di introdurre errori di posizione o scala. Al contrario, se si fissassero dimensioni più aderenti al bounding box, si rischierebbe di focalizzare il tracking su un'area specifica del target, così da sottostimarlo.

Per quanto riguarda il calcolo dei valori da assegnare alla matrice, le modalità provate sono state tre, le quali si differenziano nel modo di determinare l'istogramma usato per la *backprojection*, ovvero, associare al pixel il relativo valore in base al suo colore.

La prima risulta la più semplice e intuitiva e consiste nel normalizzare l'istogramma del target rispetto al massimo; la backprojection che ne risulta è, però, troppo aderente al modello e gli eventuali picchi di probabilità concentrerebbero l'attenzione del CAMShift solo in certe zone del target.

La seconda combina con la regola di Bayes gli istogrammi della finestra di ricerca con quello della porzione di immagine sulla quale calcolare la CPDDI; in questo caso l'immagine di probabilità risalta i colori che sono maggiormente presenti solo nel riquadro di ricerca; ciò è un vantaggio sotto il punto di vista delle interferenze

con colori simili, ma uno svantaggio in quanto si perdono le informazioni legate al modello target.

La terza prevede di costruire un istogramma come rapporto tra quello del target e quello della porzione di immagine sulla quale calcolare la CPDDI limitandone il valore massimo a 1; ne risulta una matrice che risalta i pixel aventi i colori del target, sia che essi vi appartengano o meno; a quest'ultima viene poi moltiplicata una matrice delle stesse dimensioni che serve a ridurre l'impatto dei pixel lontani dal centro della finestra di ricerca.

Da questi risultati, il metodo che si è rivelato il più adatto alla scopo, è quello descritto per ultimo nonostante possa risentire comunque della eventuale presenza di picchi di probabilità nell'istogramma del target e di blob di colori simili al soggetto inseguito.

In questa tesi sono state testate due versioni dell'algoritmo CAMShift prendendo come base di sviluppo l'algoritmo proposto nell'omonima funzione della libreria OpenCV. Entrambe presentano modifiche rispetto alla versione originale che mirano a migliorare l'affidabilità dell'inseguimento: ogni spostamento e cambio di dimensioni della finestra di ricerca viene valutato in termini di coefficiente di Bhattacharya e confrontato con quello associato alla finestra di ricerca di partenza; solo in caso di miglioramento, ovvero il primo indice maggiore del secondo, la posizione e le dimensioni calcolate verrebbero considerate valide.

La prima versione presenta però un'ulteriore modifica, questa volta più sostanziale. A differenza dell'algoritmo originale in cui viene eseguito il MeanShift fino a convergenza, ovvero fino a quando il vettore di spostamento della finestra scende sotto una certa soglia, e poi proposte le nuove dimensioni, in questa variante si modificano posizione e dimensioni fino a convergenza, ricalcolando ad ogni iterazione la distribuzione di probabilità.

Questa proposta di CAMShift era nata con l'intento di velocizzare l'adattamento del bounding box al target così da rendere l'algoritmo particolarmente efficiente nel caso di cambi di posa o occlusioni parziali; ciò è fondamentale nel caso di tracking in cui questo accade molto velocemente. In figura 3.15 si riportano una serie di frame che mostrano la differenza tra le due versioni proprio in un caso simile.

Al vantaggio di un rapido adattamento, si accompagnano due svantaggi legati proprio al ricalcolo della distribuzione di probabilità più volte nello stesso frame. Il primo riguarda l'ovvio aumento del costo computazionale che è particolarmente evidente quando il target assume dimensioni elevate; il secondo concerne la capacità del bounding box di rimanere "incollato" al target, che in questo caso diminuisce a causa di una possibile continua espansione della distribuzione di probabilità che va ad includere oggetti di simile aspetto.

Alla luce di quest'ultima analisi si è deciso di utilizzare la versione più simile a quella originale, avente quindi solo il controllo dell'indice di Bhattacharya tra finestre di ricerca.

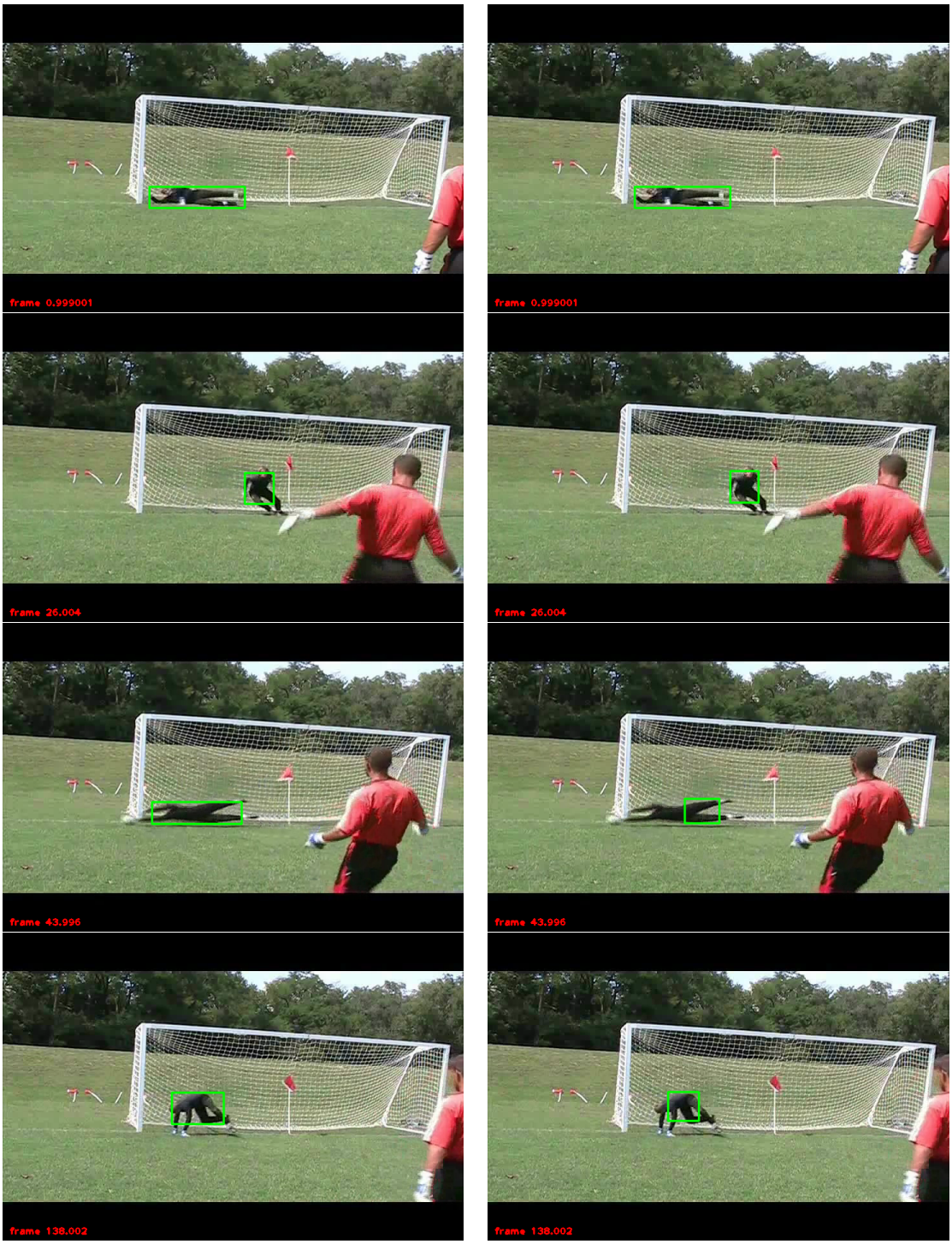


Figura 3.15. A sinistra l'algoritmo con il ricalcolo distribuzione di probabilità; a destra quello più fedele alla versione originale

Per concludere, a seguito di alcuni test è emerso come l'algoritmo soffra in caso di tracking di soggetti "piccoli", aventi cioè un'area del bounding box relativamente scarsa. In queste situazioni la quantità di dati disponibili risulta insufficiente per una stima affidabile e può portare ad un'aggiornamento scorretto del modello target e conseguentemente una sottovalutazione delle dimensioni. Si è quindi imposto un vincolo legato all'area minima del target il cui valore è stato scelto empiricamente osservando il comportamento dell'algoritmo pari a 3000 pixel.

Capitolo 4

Calibrazione dei colori

In molte situazioni, i sistemi di videosorveglianza sono dotati di un certo numero di telecamere che permettono così il monitoraggio di vaste aree. Un algoritmo di tracking distribuito quindi presenta il vantaggio di poter localizzare il soggetto anche quando questo non risulta più inquadrato da una data telecamera; capita spesso infatti che il target entri ed esca dal campo di visione.

Il processo di identificazione di un oggetto precedentemente inquadrato da un'altra telecamera, in un flusso video proveniente da una differente inquadratura, nasconde un problema legato alla possibile diversità del suo aspetto riscontrabile in quest'ultimo. Questa differenza nei colori può essere imputata a fattori quali la diversa angolazione di ripresa, l'impiego di telecamere non uguali, la distanza dal soggetto, la diversa condizione di luminosità o le differenti caratteristiche intrinseche delle telecamere (guadagno, lunghezza focale ecc.).

Per questo motivo, lo stesso algoritmo di ricerca del target precedentemente descritto non può essere impiegato indistintamente utilizzando informazioni provenienti da altre telecamere. In [1], [17] e [2] vengono presentate alcune tecniche di *inter-camera color calibration* che portano un miglioramento significativo al processo di identificazione di un oggetto. L'idea di base è quella di rendere le immagini di due telecamere abbastanza simili da poter utilizzare lo stesso modello del target per entrambe le sorgenti. Una volta avvenuta la localizzazione, il tracking continua in modo indipendente per ogni telecamera e sui rispettivi flussi video originali.

La funzione che mappa lo spazio dei colori di una telecamera nello spazio dei colori di un'altra prende il nome di *Brightness Transfer Function* (BTF). Le tecniche più comuni per definire questa funzione sono due: *Mean BTF* e *Cumulative BTF*; in [1] viene però proposto un terzo metodo che prende il nome di *Modified Cumulative BTF* e che sembra dare risultati interessanti.

Tutte e tre le tecniche prevedono una fase di training durante la quale vengono calcolati gli istogrammi di colore di una serie di oggetti noti per ogni coppia di telecamere. Siano $h_i^1(B_m)$,$h_i^2(B_m)$,$\ldots$ $h_i^k(B_m)$ e $h_j^1(B_m)$,$h_j^2(B_m)$,$\ldots$ $h_j^k(B_m)$ con $m =$

$1 \dots 256$ gli istogrammi dei k oggetti catturati rispettivamente dalla telecamera i e j. Siano poi $H_i^w(B_m)$ e $H_j^w(B_m)$ gli istogrammi cumulativi relativi all'oggetto w calcolati come

$$H^w(B_m) = \sum_{y=1}^{m} h^w(B_y) \tag{4.1}$$

Una volta normalizzati i due istogrammi cumulativi deve risultare che $H_i^w(B_m) = H_j^w(f_{i,j}(B_m))$ e quindi la funzione di trasferimento BTF sempre per l'oggeto w si calcola come

$$f_{i,j}(B_m) = (H_j^w)^{-1}(H_i^w(B_m)) \tag{4.2}$$

Le diverse tecniche accennate poc'anzi permettono di definire una BTF generica che inglobi informazioni di colore relative a più coppie di oggetti, ottenendo così una calibrazione più accurata. Per quanto riguarda la MBTF, si determina la BTF per ogni coppia come in 4.2 per poi ottenere quella finale mediandole. La CBTF prevede invece che venga prima calcolato un singolo istogramma cumulativo per ciascuna telecamera

$$H_i(B_m) = \sum_{k=1}^{m} \sum_{l=1}^{k} h_i^l(B_k) \tag{4.3}$$

poi normalizzato e ricavata la BTF finale con 4.2. Infine per la MCBTF, dopo aver calcolato le k coppie di istogrammi relativi ad ambo le telecamere, per ciascuna di esse se ne determina uno unico ($\bar{h_i}(B_m)$) calcolato come la media dei k istogrammi tenendo conto però dei soli bin che portano un significativo contributo. Sia

$$K_i^m = \{y \in [1 \dots k] : h_i^y(B_m) > \Theta\} \tag{4.4}$$

con $\Theta = 5$ si ha che

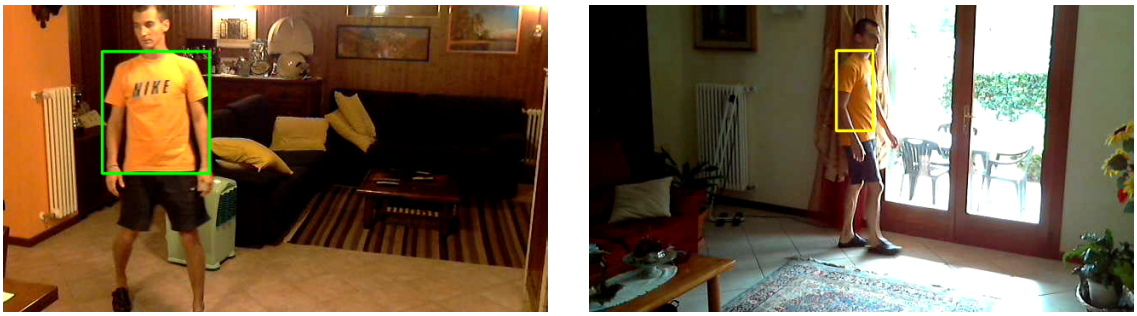
$$\bar{h_i}(B_m) = \frac{\sum_{y \in K_i^m} h_i^y(B_m)}{\#K_i^m} \tag{4.5}$$

Si determinano i due rispettivi istogrammi cumulativi $H_i(B_m)$ e $H_j(B_m)$ come descritto in 4.1 e infine si calcola la BTF come in 4.2.

Considerato che dalla letteratura emergono dati contrastanti riguardo quale sia il metodo con le migliori performance, si è voluto effettuare una serie di test così da scegliere su quale tecnica concentrare l'attenzione. Per queste prove sono stati registrati quattro video, che presentano condizioni e sorgenti di illuminazione differenti, per poi verificare l'efficacia dei metodi sulle dodici coppie che si ottengono incrociando i filmati. C'è da precisare che "passare" dall'*i-esimo* video al *j-esimo* non è affatto come passare da *j-esimo* all'*i-esimo*; per questo motivo si è potuto considerare di avere a disposizione dodici diverse situazioni. Si è verificato inoltre la capacità di riconoscimento del target anche in assenza di una calibrazione dei colori.

I test in questione sono stati eseguiti nel seguente modo: per ogni coppia di video (i,j) si è calcolato l'istogramma del target dall'i-esimo e nel j-esimo, dopo averlo

equalizzato in accordo con la BTF, si è avviata la procedura di ricerca del soggetto. L'individuazione anche parziale del target nel j-esimo video è stata considerata come un successo o *True Positive*; l'individuazione di un altro oggetto, come un *False Positive*; non aver trovato alcuna traccia quando invece il target era visibile, come un *False Negative*; non aver trovato alcuna traccia quando il target non era visibile, come un *True Negative*. Nelle figure 4.1 e 4.2 sono riportati dei frame, tratti dai video di test, che mostrano degli esempi di esito TP ed FP.

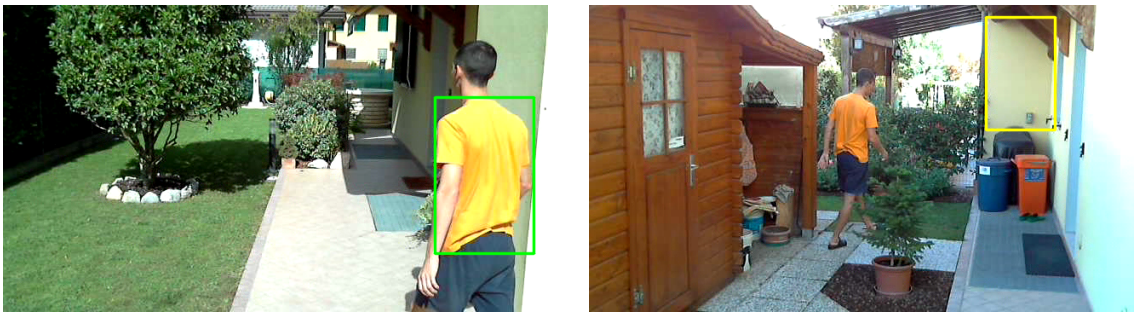


(a)

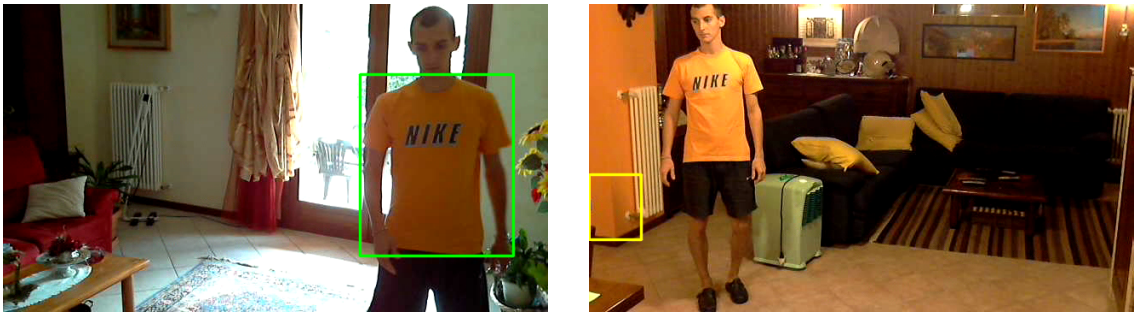
(b)

Figura 4.1. Esempio di individuazione corretta del target; sulla sinistra i frame sui quali è stato calcolato l'istogramma del target e il relativo bounding box, mentre sulla destra i frame dei video sui quali è stato ritrovato il target e il relativo bounding box

Dai risultati di questi test si evince che tra le tre proposte la migliore è risultata essere MCBTF, seguita da CBTF e MBTF. MCBTF ha ottenuto un esito paragonabile ma leggermente migliore di quello della prova senza alcuna calibrazione in termini di numeri di *falsi positivi* e *falsi negativi*; dove però un caso ha fallito l'altro ha avuto successo. In figura 4.3 si riportano alcune immagini che mostrano le differenze qualitative tra i tre metodi di calibrazione e nella tabella 4.1 si presentano i risultati numerici dei test, ovvero, per ciascun metodo la numerosità dei quattro possibili esiti.



(a)



(b)

Figura 4.2. Esempio di individuazione scorretta del target; sulla sinistra i frame sui quali è stato calcolato l'istogramma del target e il relativo bounding box, mentre sulla destra i frame dei video sui quali è stato ritrovato il target e il relativo bounding box

Metodo	**TP**	**TN**	**FP**	**FN**
Senza calibrazione	8	0	2	2
MCBTF	9	0	2	1
CBTF di Posa	7	0	1	4
MBTF	6	0	0	6

Tabella 4.1. TP (True Positive) = target correttamente individuato; TN (True Negative) = target correttamente non individuato; FP (False Positive) = target non correttamente individuato; FN (False Negative) = target non correttamente non individuato

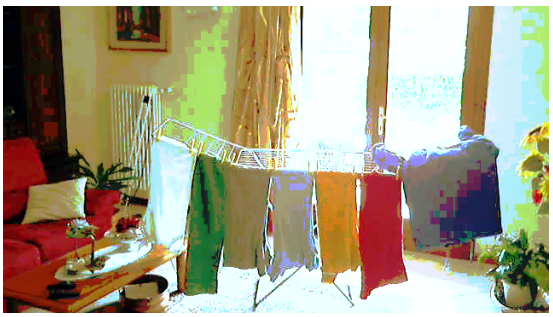
Alla luce di questo si ritiene che la calibrazione dei colori non sia sufficiente a garantire sempre il riconoscimento del target, ma se utilizzata a supporto del flusso video originale si ritiene possa sicuramente aumentare l'efficacia del riconoscimento.

Durante questo studio è emerso come la qualità della BTF sia particolarmente influente sul rate di *hit* e che in letteratura non vi sono dettagli su come condurre

(a) Immagine sulla quale viene calcolato l'istogramma target

(b) Immagine sulla quale deve essere ritrovato il target

(c) MCBTF



(d) CBTF

(e) MBTF

Figura 4.3. Esempio di calibrazione; da queste immagini si può apprezzare la differenza tra i tre metodi di calibrazione: maggiore è la somiglianza tra i colori dell'immagine sottoposta a calibrazione con i colori dell'immagine sulla quale è stato calcolato l'istogramma, migliore è la qualità della calibrazione stessa e più facilmente sarà possibile ritrovare il target a partire dal suo modello

la fase di training, su quanti e quali oggetti utilizzare e sulle condizioni ambientali. Si è notato, però, che la via per ottenere una buona BTF è l'acquisizione di un gran numero di istogrammi relativi a oggetti aventi colori differenti così da coprire una vasta gamma cromatica. La figura 4.4 mostra proprio questa particolarità.

(a) Immagine sulla quale viene calcolato l'istogramma target

(b) Immagine sulla quale deve essere ritrovato il target

(c) Immagine di destinazione calibrata solo con un colore (verde)

(d) Immagine di destinazione calibrata con tutti i colori

Figura 4.4. Esempio di MCBTF; dalle due immagini in basso risulta evidente come la qualità della calibrazione sia maggiore se la BTF viene calcolata a partire da molti colori

In questa tesi si è voluto mettere alla prova anche alcune tecniche, più "clasiche", come il bilanciamento del bianco e l'equalizzazione dell'immagine. Questi accorgimenti non hanno però portato alcun miglioramento significativo in quanto non risolvono il problema della BTF di essere particolarmente legata alla condizione ambientale in cui viene calcolata, inoltre sono due procedimenti che interessano tutti i frame di entrambi i flussi video e che, quindi, vanno a penalizzare il tempo di esecuzione. Analogamente a quanto fatto per i metodi di calibrazione, in figura 4.5 si riportano alcune immagini che mostrano le differenze qualitative tra le due tecniche appena citate e nella tabella 4.2 sono proposti i risultati numerici dei test.

(a) Immagine equalizzata sulla quale viene calcolato l'istogramma target

(b) Immagine equalizzata sulla quale deve essere ritrovato il target

(c) Immagine bilanciata sulla quale viene calcolato l'istogramma target

(d) Immagine bilanciata sulla quale deve essere ritrovato il target

Figura 4.5. Esempio di calibrazione tramite equalizzazione e bilanciamento automatico del bianco

Metodo	**TP**	**TN**	**FP**	**FN**
Equalizzazione	7	0	2	3
AWB	5	0	1	6

Tabella 4.2. TP (True Positive) = target correttamente individuato; TN (True Negative) = target correttamente non individuato; FP (False Positive) = target non correttamente individuato; FN (False Negative) = target non correttamente non individuato

Capitolo 5

Risultati sperimentali

In questo capitolo si presentano le performance del sistema di tracking proposto. L'insieme dei video di test è composto da 16 video aventi durata e problematiche differenti; per poter valutare meglio le qualità e i difetti di questo sistema si è deciso di suddividere il data set in categorie che tenessero conto del tipo di difficoltà da affrontare.

Il primo insieme raggruppa i video il cui soggetto da inseguire non subisce né occlusioni né sostanziali cambi di posa; il secondo gruppo è formato da quelli in cui il target cambia vistosamente posa; nel terzo e ultimo sono presenti video con occlusioni parziali o totali del target, per i quali risulta di fondamentale importanza il meccanismo di recovery della traccia.

Nelle figure 5.1, 5.2, 5.3, 5.4 e 5.5 si riportano alcuni frame significativi che mostrano il comportamento del tracker.

L'indice scelto per valutare le prestazioni è l'*indice di Pascal*, che è quello comunemente usato per analisi di questo tipo. Il suddetto indicatore confronta il bounding box con un rettangolo che rappresenta la corretta inquadratura del target; per ogni video è stata quindi determinata la cosiddetta *Ground Truth*, ovvero, un insieme di posizioni e grandezze della giusta stima del target per ogni frame.

Detto questo, è possibile definire l'indice in questione. Siano rispettivamente $R_{GT}(t)$ e $R(t)$ i bounding box che stimano il target secondo la Ground Truth e secondo l'algoritmo in questione; l'indice di Pascal tra le due rappresentazioni al frame t è definito come:

$$P(t) = \frac{Area(R_{GT}(t) \cap R(t))}{Area(R_{GT}(t) \cup R(t))}$$

Per questo lavoro, un frame si considera valido quando $P(t) \geq \frac{1}{3}$ e in presenza di una segnalazione di drift corretta.

Accanto a questo indicatore ne vengono determinati altri due che valutano l'errore relativo di posizione e di dimensione del bounding box.

Il primo esprime la distanza euclidea tra il centro della stima del target e il centro della ground truth, normalizzata rispetta alla diagonale di quest'ultima; indica, in percentuale, quanto sono distanti i centri dei relativi rettangoli:

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

Il secondo rappresenta la distanza euclidea tra i due vettori (relativi a GT e BB) formati dall'altezza e dalla larghezza del rispettivo riquadro, normalizzata secondo la diagonale del riquadro della ground truth; indica, in percentuale, quanto sono diverse le dimensioni dei relativi rettangoli:

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

Di seguito si riporta la tabella riassuntiva dei risultati mediati per ogni categoria.

Categoria	**Frame totali**	**% Frame validi**	**% E_{pos}**	**% E_{dim}**	**FPS**
Normali	1902	93,3	12	26	53,2
Cambi di Posa	1710	91,7	8	27	47,9
Occlusioni	2963	90,5	26	29	66,9
Totale	**6575**	**91,6**	**17**	**28**	**58,0**

Dalla tabella si evince che la percentuale di frame validi si attesta attorno al 90%; il sistema di tracking proposto riesce quindi ad ottenere risultati pressoché costanti indipendentemente dal tipo di difficoltà affrontata. A differenza di altri lavori, in cui i risultati esposti sono stati migliori ma per ottenerli sono state necessarie varie reinizializzazioni del modello con la ground truth, il tracker proposto in questo lavoro sì raggiunge prestazioni leggermente inferiori ma le ottiene senza alcun "aiuto esterno", aspetto che ritengo alquanto importante soprattutto nell'ottica di tracking realtime. Si può notare inoltre che le prestazioni generali tendono a calare quando il grado di difficoltà invece aumenta. Questo fatto però non riguarda la velocità d'esecuzione in quanto essa dipende soprattutto dall'area del bounding box e dalla capacità elaborativa del calcolatore che per questi test è stato un Intel® Centrino® Duo (2.0 GHz) con 2 GB di RAM in ambiente Linux. Il framerate ottenuto per l'inseguimento di una traccia si attesta attorno a 50 fps, valore abbondantemente al di sopra di quello necessario per un tracking real-time, ossia 30 fps. Considerato il fatto che l'algoritmo non è stato ottimizzato per processori multicore e non viene sfruttata la GPU, si può affermare che esiste comunque un margine di miglioramento che potrebbe assicurare un tracking real-time anche in presenza di più tracce da inseguire.

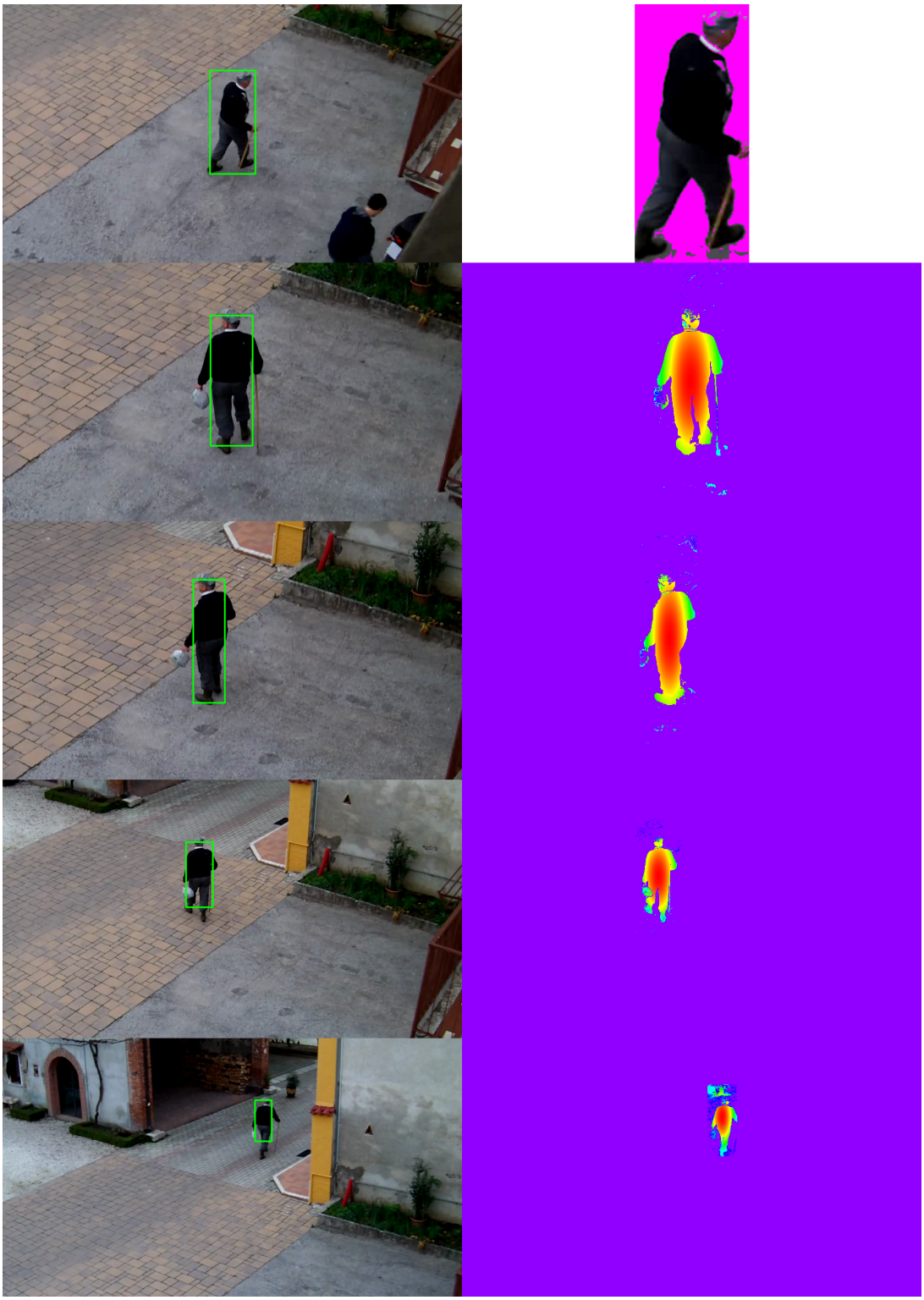


Figura 5.1. Esempio di inseguimento facile; il soggetto non cambia posa e non viene occluso

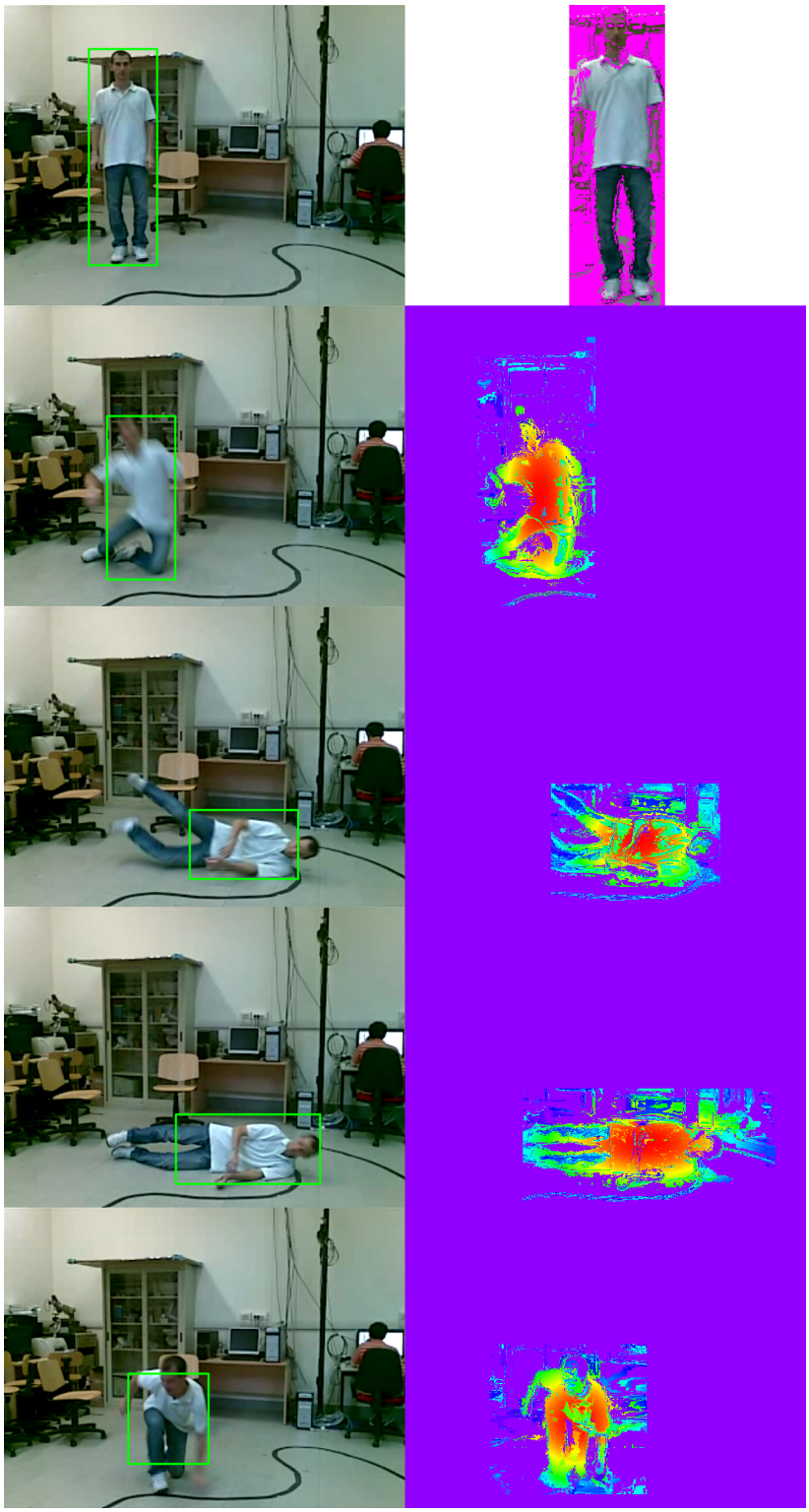


Figura 5.2. Esempio di inseguimento con cambio di posa

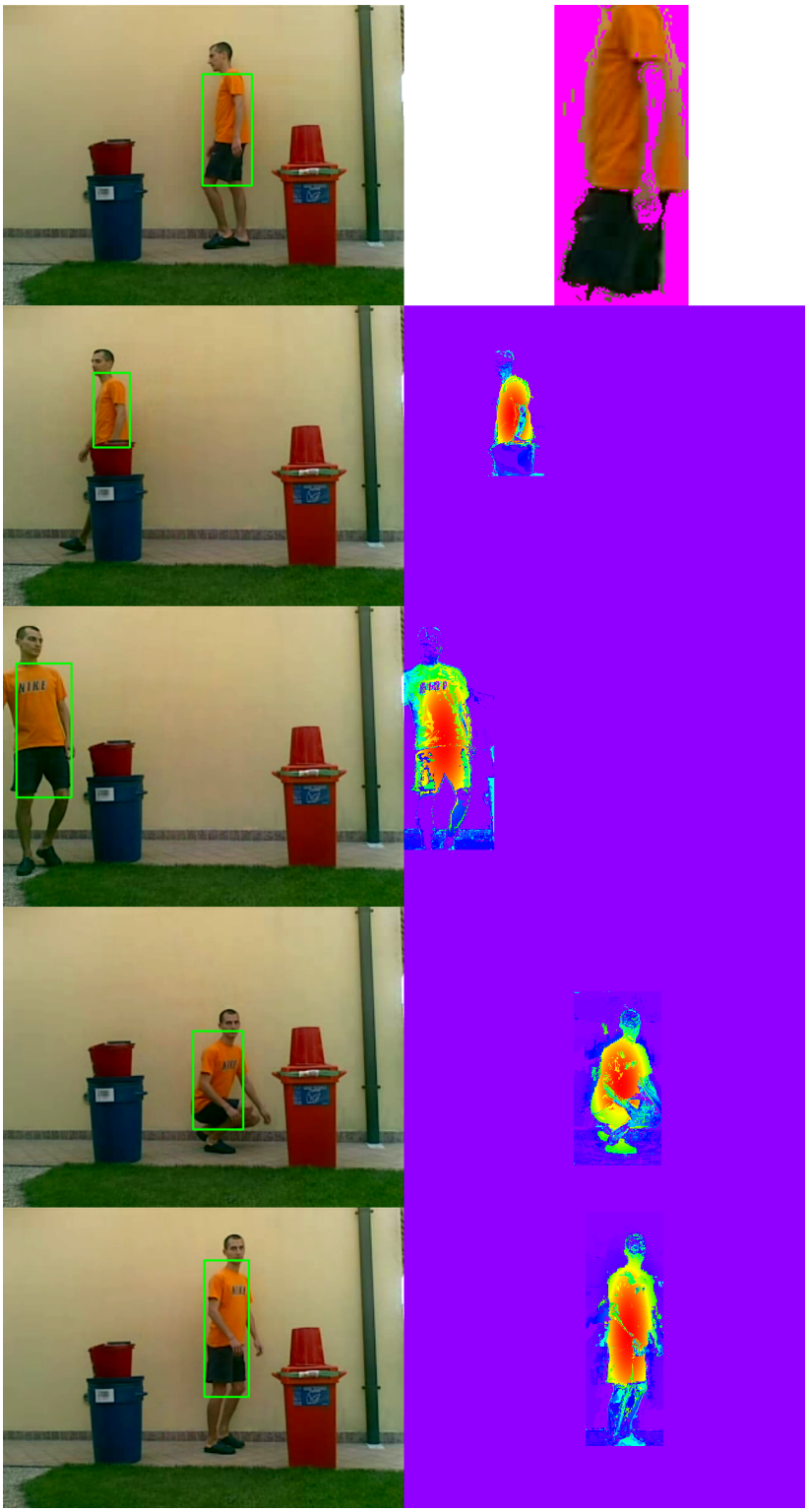


Figura 5.3. Esempio di inseguimento con occlusione

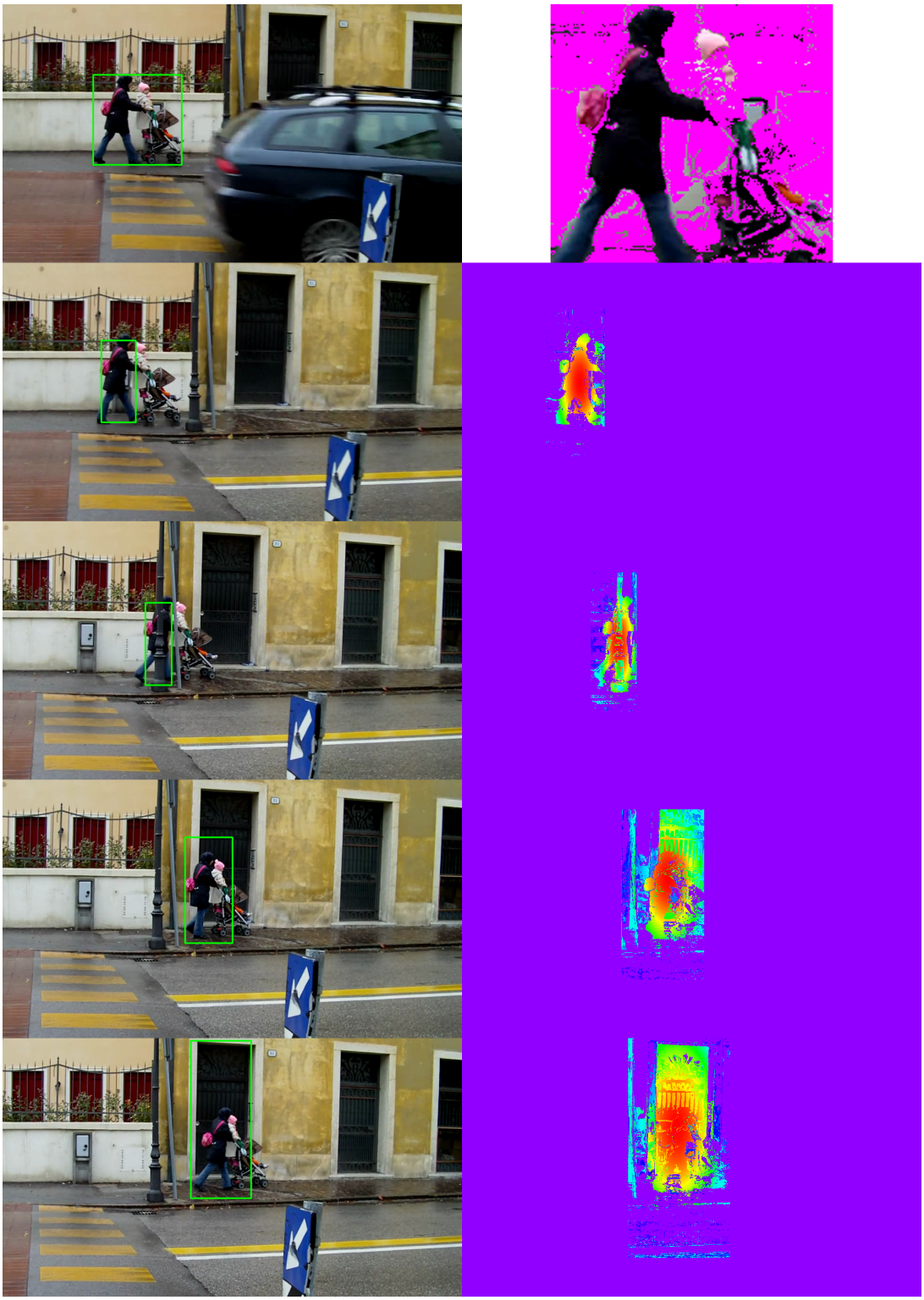


Figura 5.4. Esempio di inseguimento errato causa inizializzazione scorretta e background troppo simile al target

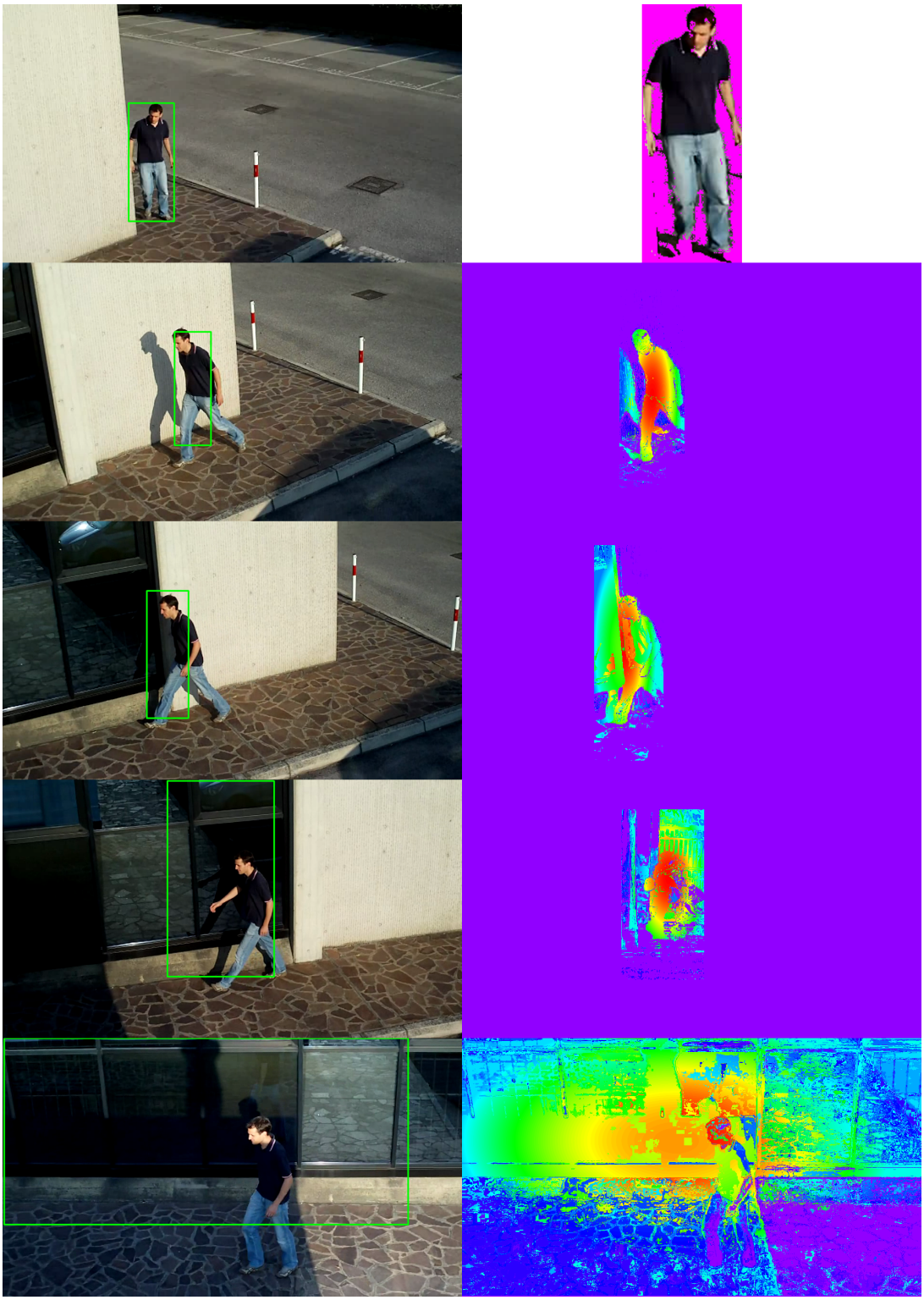


Figura 5.5. Esempio di inseguimento errato causa eccessivo cambio di luminosità e background troppo simile al target

Capitolo 6

Conclusioni

Questo lavoro di tesi è una parte importante di un progetto più ampio, la videosorveglianza di ambienti, che vede il tracking di persone e oggetti in genere come un aiuto per gli addetti alla videosorveglianza stessa. Esso è composto da un primo blocco che il compito di segmentare il target da inseguire così da creare un modello per il tracking; il secondo blocco è costituito dal vero e proprio algoritmo di tracking; un terzo blocco consiste nel drift detector che ha il compito di segnalare la perdita del target dovuta ad occlusioni o più semplicemente dall'uscita dello stesso dal campo di visione della telecamera; il quarto blocco è responsabile della reinizializzazione della traccia in caso questa venisse persa; infine l'ultimo blocco facilita il ritrovamento del target da parte di telecamere la cui condizione ambientale altera la percezione dei colori del target stesso così da permettere un tracking su vasta scala.

L'algoritmo di tracking proposto si è rivelato particolarmente efficace nell'inseguire soggetti che venivano occlusi parzialmente o la cui posa cambiava in modo rilevante, aspetto che pochi algoritmi riescono a gestire. Di fondamentale importanza è stato lo sviluppo del blocco responsabile della reinizializzazione dalla traccia che ha permesso di prolungare la vita del tracker anche in presenza di occlusioni totali soprattutto nell'ottica di tracking real-time e in ambiente aperto in cui le occlusioni sono alquanto probabili. Anche l'ultimo blocco, nonostante si discosti leggermente dal quello che è il tracking vero e proprio, ricopre un ruolo strategico per il sistema in quanto risolve, almeno in parte, il problema della disuguaglianza percettiva di cui possono soffrire telecamere aventi campi di visione non sovrapposti e posizionate in luoghi dove la condizione luminosa è differente. Se si considera la videosorveglianza congiunta di ambienti interni ed esterni, la possibilità di inseguire un soggetto indipendentemente da quale telecamere lo stia effettivamente inquadrando genera un beneficio notevole.

Alla luce dei test effettuati si può quindi concludere affermando che il sistema di tracking proposto in questa tesi ha raggiunto lo scopo per cui è stato pensato; si ricorda infatti che la percentuale dei frame trackati correttamente si attesta attorno

al 90%, corredata da un framerate sui 50 fps che assicura un tracking real-time.

Possibili sviluppi futuri possono sicuramente riguardare un'ottimizzazione del codice per sistemi multicore e che sfrutti la GPU aumentando notevolmente la velocità d'esecuzione così da poter utilizzare, in aggiunta, dei descrittori del target, anche complessi, che portino un incremento dell'affidabilità del tracker.

Bibliografia

[1] S. Miguet A. Ilyas, M. Scuturici. *Inter-Camera Color Calibration for Object Re-identification and Tracking.* In 2010 International Conference of Soft Computing and Pattern Recognition, 2010.

[2] T. Xiang B. Prosser, S. Gong. *Multi-camera Matching using Bi-Directional Cumulative Brightness Transfer Functions.* In British Machine Conference, 2008.

[3] G. Bradski. *Computer Vision Face Tracking as a Component in a Perceptual User Interface.* In IEEE Workshop Applications of Computer Vision, 1998.

[4] A. Blake C. Rother, V. Kolmogorov. Grabcut - interactive foreground extraction using iterated graph cuts. Technical report, Microsoft Research Cambridge, UK.

[5] P. Meer D. Comaniciu. *Mean Shift: A robust approach toward feature space analysis.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002.

[6] P. Meer D. Comaniciu, V. Ramesh. *Real-Time Tracking of Non-Rigid Objects using Mean Shift.* IEEE Conf. on Computer Vision and Pattern Recognition, 2000.

[7] P. Meer D. Comaniciu, V. Ramesh. *Kernel-based object tracking.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 2003.

[8] R. Yang C.G. Jhun K. Chen, B. Hu. *A Bhattacharyya-factor Based Camshift Application for Video Fast Mobile Target Tracking.* In Fuzzy Systems and Knowledge Discovery, 2010.

[9] L.V. Gool K. Nummiaro, E. Koller-Meier. *An adaptive color-based particle filter.* In Image and Vision Computing, 2003.

[10] T. Ikenaga L. Sun, B. Wang. *Real-time Non-rigid Object Tracking Using CAM-Shift with Weighted Back Projection.* In 2010 International Conference on Computational Science and Its Applications, 2010.

[11] B. Lain. *Tracking PTZ in tempo reale per videosorveglianza.* Tesi di Laurea specialistica in Ingegneria dell'Automazione, Università di Padova, 2010.

[12] B. Schiele M. Andriluka, S. Roth. *People-tracking-by-detection and people-detection-by-tracking.* In Computer Vision and Pattern Recognition, 2008.

[13] B. Leibe E. Koller-Meier L. Van Gool M. Breitenstein, F. Reichlin. *Online Multi-Person Tracking-by-Detection from a Single, Uncalibrated Camera.* In Pattern Analysis and Machine Intelligence, 2010.

[14] I. Matthews S. Baker, R. Gross. *Lucas-Kanade 20-years on: A unifying framework.* In Computer Vision, 2004.

[15] C.G. Saneem Ahmed S. Saravanakumar, A. Vadivel. *Multiple human object tracking using background subtraction and shadow removal techniques.* In Signal and Image Processing, 2010.

[16] S.K. Tu S.K. Weng, C.M. Kuo. *Video object tracking using adaptive Kalman Filter.* In Visual Communication and Image Representation, 2006.

[17] P. Spagnolo T. D'Orazio, P.L. Mazzeo. *Color Brightness Transfer Function Evaluation for Non overlapping Multi Camera Tracking.* In ICDS '09, 2009.

[18] M.-P. Jolly Y.Y. Boykov. *Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images.* In International Conference on Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE, 2001.