Pedestrian Decection in Bargello National Museum

Matteo Simoncini, Alessandro Tibo, Stefano Vannacci

7 agosto 2014

Indice

Capitolo 1

Introduzione

L'obiettivo di questo elaborato è realizzare un sistema in grado di valutare l'accuratezza di un detector di persone, considerando come scenario i dati relativi ad una stanza del Museo Nazionale del Bargello.

Sono presenti in letteratura molti algoritmi di pedestrian detection. L'algorimto preso in esame utilizza i descrittori HOG (Histogram Oriented Gradient) per estrarre le informazioni dalle immagini. Tali descrittori vengono poi passati in ingresso ad una SVM precedentemente addestrata che effettua il rilevamento utilizzando un approccio multiscala. Inoltre, una volta ottenute le *detected windows*, è previsto l'utilizzo di un algoritmo di NMS (Non-Maximum Suppression) per rimovere quelle spazialmente troppo vicine tra loro e relativi alla medesima persona. Questo elaborato utilizza un'implementazione efficiente di questo algoritmo presente in *OpenCV*.

Per poter valutare l'efficacia del detector è stato necessario creare una *ground truth*. Per fare questo, è stato realizzato un *web-tool* che permette un'etichettatura manuale e rapida di una scena composta da persone e di esportare i dati ottenuti in un file di testo.

Infine, sono state applicate determinate operazioni di *preprocessing* sulle immagini e *postprocessing* sui risultati al fine di migliorare la detection. Nello specifico sono stati confrontati i risultati delle immagini reali, quelli ottenuti applicando un filtro omomorfico [6] [4] e quelli ottenuti applicando un'equalizzazione. Successivamente i risultati sono stati filtrati sfruttando l'algoritmo di *Background Subtraction* [5]. Tale algoritmo, a partire dalle immagini precedenti, riesce a distingure il background dal foregound dell'immagine corrente: tale informazione risulta essere un'ottima tecnica per scremare i risultati ottenuti e ridurre il numero di falsi positivi.

Capitolo 2

Tool per creazione di Ground Truth

Il tool premette di etichettare le persone presenti in un video. Oltre alla *detect window* della specifica persona nel frame, è possibile specificare ulteriori informazioni sulla stessa, come l'appartenenza o meno ad un gruppo o l'orientazione del volto. In particolare è possibile specificare:

- *Bounding box (bb)*: cioè il rettangolo all'interno del quale è contenuta la persona;
- *Bounding box visibile (bbV)*: cioè il rettangolo che contiene la parte visibile (e.g., solo la testa) della persona;
- *Face*: la direzione nella quale è rivolta la faccia della persona taggata;
- *Body*: la direzione nella quale è rivolto il corpo della persona taggata;
- *Group*: l'appartenenza o meno di una persona ad un gruppo di persone;
- *Gestione dei gruppi*: sono presenti una serie di strumenti per la gestione dei gruppi. In particolare è possibile specificare il nome del gruppo al momento della creazione, dopodichè l'utente può visualizzare il numero di persone all'interno del gruppo nel frame corrente e rimuovere un gruppo. Questa ultima opzione è disponibile solo se nessun utente è mai stato associato a quel gruppo in un qualsiasi frame (altrimenti, rimuovendolo, avremmo una perdita di informazione);
- *Opera*: specificare se una persona sta osservando una specifica opera d'arte o meno;
- *Rimozione di una persona*: è possibile rimuovere tutte le informazioni relative ad un particolare frame di una persona (i.e., *bb*, *bbV*, *face*, *body*, *Group*, *Opera*);

- *Aggiunta di una persona*: l'aggiunta di una nuova persona è prevista in due scenari distinti. Il primo, se si tratta di una persona mai etichettata prima, in tal caso viene associato un nuovo ID. Il secondo, in cui si tratta di una persona già presente nei frame precedenti: in questo si vorrebbe poter associare lo stesso ID. È previsto un tool per la gestione di questo scenario che permette di far selezionare all'utente la nuova persona da una serie di immagini. Per maggiori informazioni si consulti la Sezione 2.1;

- *Zoom*: per l'etichettatura di parti lontane e poco visibili e per le dimensioni limitate del monitor, è possibile zoommare il contenuto del frame in un particolare punto;

- *Cambio frame*: è possibile con semplicità passare al frame successivo, al precedente o ad un frame qualsiasi specificandone il numero;

- *Suggerimenti sul frame successivo*: E' molto probabile che una persona sia rimasta ferma da un frame ad un altro, o che si sia mossa di poco. Per questo quando l'utente accede ad un nuovo frame vengono suggeriti i dati delle persone etichettate nel frame precedente.

Quasi tutto può essere specificato tramite determinate combinazioni di tasti, in modo da minimizzare il più possibile il tempo richiesto alla creazione della GT.

2.1 Implementazione

Il tool è stato implementato come pagina web, utilizzando *jQuery* per la gestione delle chiamate asincrone lato client e PHP lato server.

Nel sistema sono presenti un certo numero di utenti, ciascuno dei quali si può autenticare utilizzando solamente il suo username (si tratta di un sistema privato e pertanto non è stato ritenuto necessario l'utilizzo di una password) per etichettare i frame. Le etichette così create sono indipendenti da quelle create da altri utenti, questo per evitare conflitti che potrebbero portare a problemi di concorrenza. Inoltre, con questo approccio, è possibile la creazione della GT da parte di più utenti contemporaneamente, come descritto in Sezione 2.2.1, per poi confrontare o unire i risultati.

Le persone presenti all'interno del frame vengono identificate da un ID univoco ed un'immagine (avatar) al fine di aiutare l'utente nell'etichettare il frame. L'associazione di tale immagine ad una persona avviene tenendo conto della dimensione bbV e della direzione di *face*. Ogni volta che una nuova *bb* viene creata su una persona con lo stesso ID, viene confrontata la

"qualità" dell'immagine salvata con quella appena acquisita e se quest'ultima risulta migliore viene sostituita.
Viene utilizzata la seguente formula:

$$\alpha \cdot \frac{|face_y - 180|}{360} + \beta \cdot \frac{|face_z|}{360} + 2\gamma \cdot \left(\frac{1}{\frac{dim[0] \cdot dim[1]}{width \cdot height} + 1} - \frac{1}{2} \right) \tag{2.1}$$

dove

- $face_y$ e $face_z$ sono le direzioni di *face* in gradi;
- $dim\,[0]$ e $dim\,[1]$ sono le dimensioni della bbV;
- *width* e *height* sono le dimensioni dell'immagine, nel nostro caso 1280× 800;
- α, β e γ sono tre pesi a somma 1. Essendo una combinazione convessa, i gradi di libertà sono soltato due, quindi nella pratica $\beta = 1 - \alpha - \gamma$.[1]

Nel nostro caso $\alpha = 0.09$ e $\gamma = 0.9$. Tali valori sono stati scelti considerando che la grandezza della bbV al fine di riconoscere una persona ià vista è un parametro più rilevante dell'orientazione della faccia.
A supporto di tale tesi, vengono riportati cinque esempi di bbV rappresentanti uno stesso soggetto al variare della lontnanza dalla camera e dell'orientazione della faccia. Sono stati effettuati dei test al variare dei valori di α e

Figura 2.1: Le cinque bounding box di test

γ, considerando i casi limite ed ordinando in base al valore della formula 2.1.

[1] I tre fattori che moltipilcano α, β e γ sono i valori di *face* e bbV normalizzati tra 0 e 1.

α	γ	**Best**	**Second**	**Third**	**Fourth**	**Fifth**
0.0	0.0	img-2, 0.0	img-1, 0.02	img-3, 0.02	img-5, 0.02	img-4, 0.93
0.0	0.09	img-2, 0.07	img-3, 0.10	img-1, 0.11	img-5, 0.11	img-4, 0.93
0.0	0.5	img-2, 0.43	img-3, 0.44	img-1, 0.50	img-5, 0.51	img-4, 0.95
0.0	0.9	img-3, 0.77	img-2, 0.77	img-1, 0.87	img-5, 0.89	img-4, 0.98
0.0	1.0	img-3, 0.85	img-2, 0.86	img-1, 0.97	img-4, 0.98	img-5, 0.99
0.09	0.0	img-2, 0.00	img-1, 0.03	img-3, 0.05	img-5, 0.06	img-4, 0.88
0.09	0.09	img-2, 0.07	img-1, 0.11	img-3, 0.13	img-5, 0.14	img-4, 0.89
0.09	0.5	img-2, 0.43	img-3, 0.47	img-1, 0.50	img-5, 0.54	img-4, 0.91
0.09	0.9	img-2, 0.77	img-3, 0.80	img-1, 0.88	img-5, 0.93	img-4, 0.93
0.5	0.0	img-2, 0.00	img-1, 0.04	img-3, 0.20	img-5, 0.22	img-4, 0.68
0.5	0.09	img-2, 0.08	img-1, 0.13	img-3, 0.27	img-5, 0.30	img-4, 0.69
0.5	0.5	img-2, 0.43	img-1, 0.52	img-3, 0.61	img-5, 0.70	img-4, 0.71
0.9	0.0	img-2, 0.01	img-1, 0.06	img-3, 0.34	img-5, 0.37	img-4, 0.49
0.9	0.09	img-2, 0.08	img-1, 0.15	img-3, 0.41	img-5, 0.46	img-4, 0.49
1.0	0.0	img-2, 0.01	img-1, 0.06	img-3, 0.37	img-5, 0.41	img-4, 0.44

Tabella 2.1

Come si nota dalla tabella 2.1, i valori scelti di α e γ selezionano come valore più basso la seconda immagine (in cui la faccia risulta ben visibile) a discapito della terza immagine, anche se più grande, in quanto il viso del soggetto è nascosto. La quarta e la quinta immagine hanno un valore molto vicino, per via dell'inclinazione della testa verso il basso della quarta immagine che la penalizza, anche se più grande.

2.1.1 Database

Il database contenente la *Ground Truth* relativa alla posizione di ogni persona per ogni immagine è stato realizzato secondo il modello relazionale di `MySQL`.

Nota: in figura 2.2 è stato commesso un abuso di notazione per migliorare la leggibilità: nella relazione `People` è stato inserito l'attributo `People's attributes`. Questo attributo in realtà fa riferimento a 13 attributi: `bb_x`, `bb_y`, `bb_width`, `bb_height`, `bbV_x`, `bbV_y`, `bbV_width`, `bbV_height`, `gazeAngle_face`, `gazeAngle_face_z`, `gazeAngle_body`, `gazeAngle_body_z`, `color`.

Di seguito il modello relazionale:
`people(peopleid, frameid, cameraid, operaid, id_utente, groupid, bb_x, bb_y, bb_width, bb_height, bbV_x, bbV_y, bbV_width, bbV_height, gazeAngle_face, gazeAngle_face_z, gazeAngle_body, gazeAngle_body_z, color)`

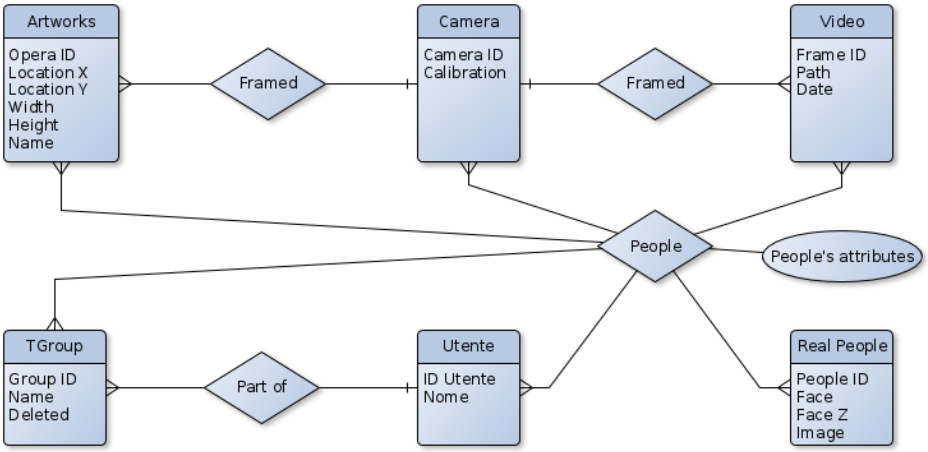

Figura 2.2: Modello ER del database

artworks(operaid, cameraid, location_x, location_y, width, heigth, name)

camera(cameraid, calibration)

video(frameid, cameraid, path, date)

real_people(peopleid, face, face_z, image)

utente(id_utente, nome)

tGroup(groupid, peopleid, name, deleted)

2.1.2 Lato Client

Sono stati utilizzati i seguenti plugin e framework:

- *JQuery*: `http://jquery.com/`
- *Bootstrap*: framework per la gestione della grafica `http://getbootstrap.com`;
- *bootstrap-colorpicker*: colorpicker per Bootstrap per cambiare il colore associato alle persone `http://mjolnic.github.io/bootstrap-colorpicker/`;
- *bootstrap-editable*: insieme a *select2* e *typeahead* per la creazione di menù a tendina avanzati con strumenti di ricerca integrati `http://vitalets.github.io/x-editable/`;
- *WebGL*: per la creazione di figure 3D utilizzate per specificare *face* e *body* `http://get.webgl.org/`;
- *image-picker*: per la selezione delle persone già presenti nel DB tramite il loro avatar `http://rvera.github.io/image-picker/`;
- *DataTable*: plugin per la creazione di tabelle con opzioni avanzate `http://www.datatables.net/`;
- *hot-keys*: per la gestione dei comandi da tastiera `https://github.com/jeresig/jquery.hotkeys`;
- *mouse-wheel*: per il controllo della wheel del mouse `http://https://github.com/brandonaaron/jquery-mouseweel`;
- *jQueryUI*: per la creazione delle bounding box `http://www.jqueryui.com/`;
- *Panzoom*: per la gestione automatica dello zoom `https://github.com/timmywil/jquery.panzoom`.

2.1.3 Lato server

Questa parte ha il compito di interfacciarsi con il Database, dal quale preleva i dati che vengono richiesti dal Client.
Per fare questo è stato implementato un Server Rest in PHP.

2.2 Caso d'uso

Alla selezione dell'opzione *GT Making* l'untente immette il suo nome utente e seleziona la camera (scegliendo tra quelle disponibili) ed il frame dal quale iniziare ad etichettare. Sono presenti tre opzioni:

- Il primo frame non taggato (dall'utente);
- Il primo frame in assoluto;
- Selezionare uno specifico frame.

Tali scelte vengono salvate in sessione e non devono essere immesse ad ogni nuovo accesso. Per poter cambiare utente e/o camera è necessario fare un logout tramite l'apposito pulsante. E' presente la tabella degli utenti taggati

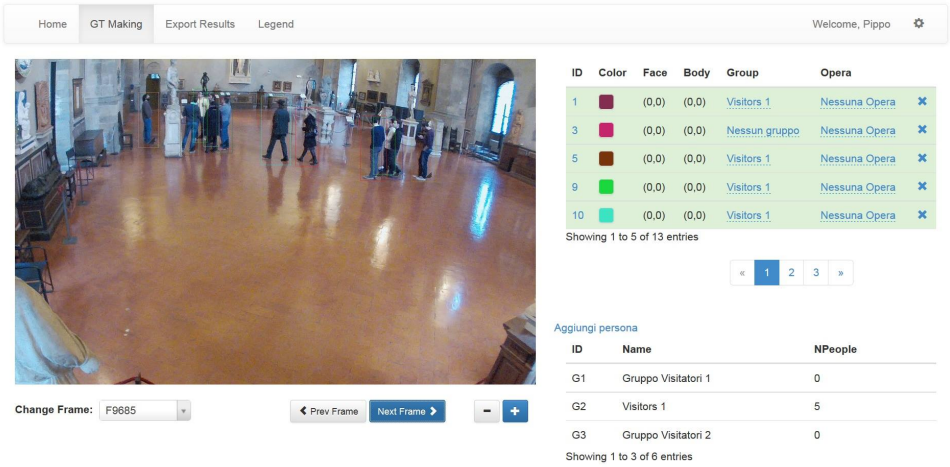

Figura 2.3: Esempio di schermata di GT Making

per quel frame (in verde) e degli utenti taggati nel frame precedente (in bianco). Per ognuno di questi, la tabella contiente:

- *id*: l'identificativo della persona taggata, passandoci sopra col mouse si può visualizzare l'avatar della persona;
- *color*: il colore associato alla *bb* della persona, modificabile tramite l'apertura di un *colorpicker* con un click;
- *face*: valori dell'orientazione della faccia in gradi;
- *body*: valori dell'orientazione del corpo in gradi;
- *group*: il gruppo associato alla persona, modificabile tramite l'apertura di una *select2* con un click;
- *opera*: l'opera associata alla persona, modificabile tramite l'apertura di una *select2* con un click;
- *remove*: tasto per la rimozione della persona dal frame corrente.

Nell'immagine, invece, si possono visualizzare le *bb* delle persone, con un bordo tratteggiato per quelle suggerite e con bordo continuo per quelle presenti nel frame corrente.
Selezionando una persona (cliccando sulla *bb* o sulla riga della tabella) diventa visibile anche la bbV e lo sfondo della riga della tabella si colora di blu. Adesso è possibile, tramite specifici comandi da tastiera, modificare la posizione e la dimensione della *bb* e della bbV. E' inoltre possibile visualizzare un cono che aiuta nell'immissione dell'angolazione di *face* e *body*.
Una volta che la persona perde il focus (perchè viene cliccata una seconda persona o cliccando in un punto dello sfondo) viene fatto l'aggiornamento dei valori inseriti tramite chimata ajax.
E' disponibile un ulteriore tabella per la gestione dei gruppi, dove è possibile visualizzare i gruppi esistenti, le persone contenute in quel gruppo per il frame corrente, creare un nuovo gruppo tramite il pulsante apposito e rimuoverne uno, se disponibile.
Inoltre, è possbile creare una nuova persona, scegliendo tra due opzioni: una persona mai vista e non presente nel DB o una persona già presente (magari in altri frame o che era uscita per alcuni secondi dalla visuale della telecamera) tramite il suo avatar.
Infine, è possibile passare al frame successivo, tornare al precedente o cambiare frame tramite pulsanti e comandi intuitivi.

Come già detto in precedenza, la gran parte di questi comandi è impartibile da tastiera. Tra questi troviamo:

- **Arrow**: muove la *bb* selezionata;
- **WASD**: modifica le dimensioni della *bb* selezionata;
- **Alt**+**Arrow**: muove la bbV selezionata;
- **Alt**+**WASD**: modifica le dimensioni della bbV selezionata;
- **Ctrl**+**Z**: aumenta il fattore di Zoom;
- **Ctrl**+**A**: diminuisce il fattore di Zoom;
- **M**: mostra il cono della *face* della persona selezionata; se premuto due volte mostra il cono del *body* della persona e se premuto ancora torna a visualizzare *bb* e bbV;
- **F**: ruota il cono attorno all'asse y;
- **G**: ruota il cono attorno all'asse z;
- +: apre il modal per aggiungere una persona;

- **U**: seleziona la persona successiva (da notare che la precedente perde il focus e quindi i suoi dati vengono aggiornati). Se non ci sono persone selezionate, seleziona la prima;
- **Return**: va al frame successivo.

2.2.1 Utilizzo multiutente

La fase di login iniziale gestisce la presenza di più utenti all'interno del sistema che collaborano alla creazione della GT.
Le etichette ed i gruppi creati da ogni utente sono indipendenti e salvate nelle tabelle `people` e `tgroup` con un utente diverso. Pertanto, ogni utente che accede al sistema può creare la sua GT che, una volta estratta, può essere unita alle altre.
L'unico elemento in comune che hanno i vari utenti sono le persone della tabella `real_people`. L'immagine associata ad una persona può essere modificata da un qualsiasi utente.

In alternativa, più utenti fisici possono effettuare il login con lo stesso account e generare parti diverse della GT. I suggerimenti al frame successivo o i possibili problemi di concorrenza relativi all'etichettatura dello stesso frame non sono stati gestiti.

2.3 Export

Il tool consente l'esportazione, in formato *json*, dei dati della ground truth. Il file contiene un array di dizionari, dove le chiavi sono tutti gli elementi che è possibile etichettare. Di seguito, è riportato un esempio di file di output.


```
[{"id":"1","frame":"1\/24-03-14\/16h21m46s_3820.jpg",
  "angle_face":"0","angle_face_z":"0","angle_body":"0",
  "angle_body_z":"0","group":"G0","artwork":"O0",
  "frameid":"F3820","cameraid":"C1","userid":"U1",
  "bb":[1229,200,48,124],"bbV":[1229,200,47,122]},
 {"id":"1","frame":"1\/24-03-14\/16h21m46s_3821.jpg",
  "angle_face":"0","angle_face_z":"0","angle_body":"0",
  "angle_body_z":"0","group":"G0","artwork":"O0",
  "frameid":"F3821","cameraid":"C1","userid":"U1",
  "bb":[1227,200,48,124],"bbV":[1227,200,47,122]},
 {"id":"1","frame":"1\/24-03-14\/16h21m46s_3822.jpg",
  "angle_face":"0","angle_face_z":"0","angle_body":"0",
  "angle_body_z":"0","group":"G0","artwork":"O0",
  "frameid":"F3822","cameraid":"C1","userid":"U1",
  "bb":[1223,200,48,124],"bbV":[1223,200,47,122]}]
```

Questo file viene poi passato in input ad un programma Python che elabora i risultati.

2.4 Creazione della Ground truth

Abbiamo utilizzato lo strumento da noi creato per etichettare una grande quantità di frame, che vanno a coprire quattro scenari di intreresse:

- *Individuals*: le persone si muovono singolarmente;
- *Groups*: le persone si muovono in gruppi;
- *Mixed Groups*: le persone si muovono in gruppi e cambiano gruppo di appartenenza durante i frames;
- *Big Group*: le persone fanno parte di un solo gruppo di grandi dimensioni.

Sono stati etichettati manualmente 1000 frames per ogni scenario di interesse, per un totale di 4000 frames per la prima telecamera. Di seguito, per valutare l'andamento del detector nelle varie telecamere, sono stati etichettati 500 frames per le altre tre, di cui 250 per lo scenario *individuals* e 250 per lo scenario *groups*.
Tale *Ground truth* è necessaria per ottenere una valutazione dell'accuratezza del detector.

Capitolo 3

Filtro omomorfico ed equalizzazione

Per cercare di migliorare l'efficienza del detector è stato deciso di effettuare due tipi di preprocessing sui frame. Il filtro omomorifoco cerca contemporaneamente di migliorare l'immagine attraverso la compressione dei livelli di intensità luminosa ed il miglioramento del contrasto. Un'immagine $f(x, y)$ può essere vista come il prodotto delle componenti di illuminazione $I(x, y)$ e di riflettanza $R(x, y)$.

$$f(x, y) = I(x, y) \cdot R(x, y). \tag{3.1}$$

Secondo le assunzione idel modello fatte in [4], è ragionevole che $I(x, y)$ contenga le basse frequenze, mentre $R(x, y)$ le alte frequenze. Vorremmo operare separatamente sulle due componenti in modo da cercare di smorzare le basse frequenze e contemporaneamente migliorare le alte. Questo non è possibile immediatamente in quanto

$$\mathcal{F}\left[f(x, y)\right] \neq \mathcal{F}\left[I(x, y)\right] \cdot \mathcal{F}\left[R(x, y)\right] \tag{3.2}$$

con $\mathcal{F}$ e $\mathcal{F}^{-1}$ rappresentano rispettivamente la trasformata di Fourier e la sua inversa. All'immagine $f(x, y)$ viene applicato il logaritmo[1] al fine di separare le due componenti.

$$z(x, y) = \log\left(f(x, y)\right) = \log\left(I(x, y)\right) + \log\left(R(x, y)\right) \tag{3.3}$$

Di conseguenza,

$$\mathcal{F}\left[z(x, y)\right] = \mathcal{F}\left[\log\left(f(x, y)\right)\right] = \mathcal{F}\left[\log\left(I(x, y)\right)\right] + \mathcal{F}\left[\log\left(R(x, y)\right)\right] \tag{3.4}$$

[1] Se un'immagine con intensità nel range $[0, L-1]$, dove L indica il numero di possibili livelli, ha qualche valore pari a 0, è necessario aggiungere 1 a ciascun elemento dell'immagine. Alla fine del processo di filtraggio sarà necessario sottrarre 1.

Per semplicità di notazione identifichiamo con $Z(u,v) = \mathcal{F}[z(x,y)]$, con $F_i(u,v) = \mathcal{F}[\log(I(x,y))]$ e con $F_r(u,v) = \mathcal{F}[\log(R(x,y))]$. A questo punto applichiamo il filtro ad entrambe le componenti. Il filtraggio prevede l'utilizzo di un semplice filtro passa alto, ad esempio un filtro di *Butterworth* che indicheremo con $H(u,v)$.

$$S(u,v) = H(u,v) \cdot Z(u,v) = H(u,v) \cdot F_i(u,v) + H(u,v) \cdot F_r(u,v) \quad (3.5)$$

dove $S(u,v)$ è l'output dell'operazione di filtraggio. A questo punto tornando nel dominio spaziale otteniamo,

$$s(x,y) = \mathcal{F}^{-1}[S(u,v)] = \mathcal{F}^{-1}[H(u,v) \cdot F_i(u,v)] + \mathcal{F}^{-1}[H(u,v) \cdot F_r(u,v)] \quad (3.6)$$

Definiamo adesso

$$i_{new}(x,y) = \mathcal{F}^{-1}[H(u,v) \cdot F_i(u,v)] \quad (3.7)$$

e

$$r_{new}(x,y) = \mathcal{F}^{-1}[H(u,v) \cdot F_r(u,v)] \quad (3.8)$$

Quindi

$$s(x,y) = i_{new}(x,y) + r_{new}(x,y) \quad (3.9)$$

Infine, dato che $z(x,y)$ è stata ottenuta applicando il logaritmo a $f(x,y)$, possiamo invertire il processo applicando l'esponenziale al risultato filtrato ottenendo l'immagine filtrata $f_{filtered}(x,y)$.

$$f_{filtered}(x,y) = e^{z(x,y)}. \quad (3.10)$$

È possibile, al fine di migliorare l'efficienza di computazione come spiegato nell'articolo [6], implementare una versione spaziale del filtro. Dalle equazioni scritte in precedenza otteniamo che l'intero processo di filtraggio può essere riassunto come segue

$$f_{filtered} = \exp(\log(f(x,y)) * h(x,y)) \quad (3.11)$$

dove il simbolo $*$ identifica l'operazione di convoluzione e $h(x,y) = \mathcal{F}^{-1}[H(u,v)]$. Sebbene tale calcolo risulti computazionalmente oneroso, in quanto dovrei applicare una trasformata di Fourier ed una convoluzione, approssimando $h(x,y)$ con un kernel di questo tipo

$$h(x,y) \simeq \begin{bmatrix} -1 & -1 & -1 \\ -1 & 128 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (3.12)$$

è possibile evitare il calcolo della trasformata inversa. Prima di applicare il filtro, però, è necessario convertire l'immagine dallo spazio colore RGB allo spazio colore HSI (Hue Saturation Illuminace). Il filtro verrà applicato alla sola componente I.

Figura 3.1: A sinistra l'immagine originale, a destra l'immagine filtrata.

L'altra operazione di preprocessing proposta consiste in un processo di equalizzazione. L'immagine viene convertita nello spazio colore YCbCr e viene equalizzato solamente il livello Y, attraverso l'algoritmo CLAHE (*Contrast Limited Adaptive Histogram Equalization)* di equalizzazione dell'istogramma.
Questo algroitmo prevede di suddividere l'immagine in blocchi di dimensione 8×8, ognuno dei quali viene equalizzato tramite il cassico algorimo di equalizzazione:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p(r_j) \tag{3.13}$$

dove r_i è il valore del livello in ingresso, che assume valori in $[0, (L-1)]$, $p(r_i)$ è la probabilità del livello r_i di appartenere all'immagine (stimata come il numero di pixel che assumono quel livello su il totale dei pixel dell'immagine), s_i il valore equalizzato.
In presenza di rumore, però, questo verrà amplificato all'interno del blocco. Per evitare questo viene usato la *limitazione del contrasto*. Prima dell'appli-

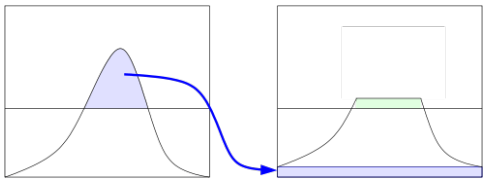

Figura 3.2: Limitazione del contrasto

cazione dell'equalizzazione, tutti i livelli dell'istogramma che superano una

certa soglia vengono fatti saturare al valore di soglia ed i pixel rimossi vengono ridistribuiti uniformemente su tutti gli L livelli.
Infine, per rimuovere gli artefatti tra i bordi, viene applicato un'intrpolazione bilineare.

Figura 3.3: A sinistra l'immagine originale, a destra l'immagine equalizzata.

Di seguito, l'applicazione dei due filtraggi ad un immagine di esempio.

Figura 3.4: Immagine originale

Figura 3.5: Applicazione dell'equalizzazione

Figura 3.6: Applicazione del filtro omomorfico

Capitolo 4

HOG Descriptor

Gli HOG (Histograms of Oriented Gradients) sono un tipo di feature che vengono estratte da un'immagine. Lo scopo di queste feature è quello di generalizzare gli oggetti all'interno di un'immagine in modo tale che gli stessi oggetti (nel nostro caso le persone) producano più o meno le stesse feature quando vengono osservati da punti di vista e condizioni diverse.
I descrittori HOG per il riconoscimento di persone vengono calcolati usando una finestra, *detection window*[1], che si muove all'interno all'immagine.
Da osservare che per rilevare persone a scale diverse l'immagine verrà sottocampionata a dimensioni diverse. In ciascuna di queste immagini sottocampionate verrà effettuata la ricerca. Queste features estratte da ogni immagine verranno passate in input ad un classificare SVM già addestrato che rileverà o meno la presenza di una persona rispetto alla detection window considerata.

4.1 Estrazione dei descrittori HOG

La figura 4.1 mostra l'intero schema algoritmico per l'estrazione dei descrittori HOG da un'immagine.

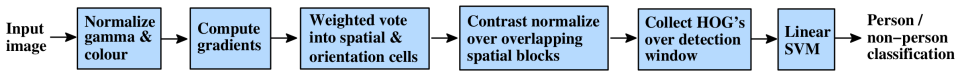

Figura 4.1

4.1.1 Gamma Correction and Compute Gradients

La prima operazione che viene fatta è l'applicazione di una *gamma correction* all'intera immagine. Questa correzione come spiegato in [2] porta ad un

[1] Nel nostro caso la detection window è di dimensioni 64×128 che risulta essere il valore ottimale come riportato in [2]

leggero miglioramento del detector. Successivamente per ogni spostamento della detection window all'interno dell'immagine calcoleremo i descrittori HOG all'interno di regioni di dimensioni 8×8 che chiameremo *cells*. In figura 4.2 è rappresentata la regione di un'immagine corrispondete ad una posizione della detection window dove sono evidenziate le *cells* in verde. All'interno

Figura 4.2

di ogni cella viene calcolato il gradiente mediante l'utilizzo di due semplici kernel del tipo $[-1\ 0\ 1]$ per quanto riguarda il calcolo del gradiente lungo la direzione x e $[-1\ 0\ 1]^T$ per quanto riguarda il calcolo del gradiente lungo la direzione y. Ricordiamo che l'applicazione di un kernel ad un'immagine consiste nell'effettuare un operazione di convoluzione tra l'immagine e il kernel. Questa operazione produce una nuova immagine. Nel nostro caso l'immagine prodotta con l'applicazione del primo kernel è un'immagine in cui ogni pixel (x, y) è il risultato della differenza tra il pixel a destra e il pixel a sinistra del pixel (x, y) dell'immagine originale. L'immagine invece prodotta con l'applicazione del secondo kernel è un'immagine in cui ogni pixel (x, y) è il risultato della differenza tra il pixel in alto e il pixel in basso del pixel (x, y) dell'immagine originale. Per ogni cella avremo quindi 64 gradienti lungo la direzione x, ∇F_x e lungo la direzione y, ∇F_y.[2] Da questo punto in poi del gradiente andremo a considerare modulo e orientazione.

[2]Prima del calcolo del gradiente è possibile applicare un filtro guassiano alle detection windows. Come riportato in [2] questa scelta porta ad un peggioramento del detector in quanto un filtro di smooth peggiora il calcolo delle derivate

4.1.2 Weighted vote into spatial and orientation cells

Una volta che sono stati calcolati i 64 gradienti per ogni cella viene generato un istogramma nel seguente modo: vengono divisi 180 gradi[3] in 9 classi, 20 gradi per classe. Ogni gradiente all'interno della cella darà il contributo all'istogramma in termini di modulo. Il modulo verrà diviso tra le due classi più vicino secondo una relazione di proporzionalità ($\frac{20-|\alpha-b_1|}{20}$ e $\frac{20-|\alpha-b_2|}{20}$, dove α rappresenta l'orientazione del gradiente e b_1 e b_2 sono i valori dei rappresentanti più vicini a α). Consideriamo ad esempio il gradiente relativo ad un pixel di un cella. Supponiamo che abbia un'orientazione pari a 45 gradi. $\frac{3}{4}$ del modulo contribuiranno ad aumentare la classe rappresentata dal numero 50 e $\frac{1}{4}$ del modulo contribuirà ad aumentare la classe rappresentata dal numero 30. Infine, come ultima operazione, tali istogrammi vengono normalizzati.
In figura 4.3 viene mostrato un esempio di istogramma per una cella.

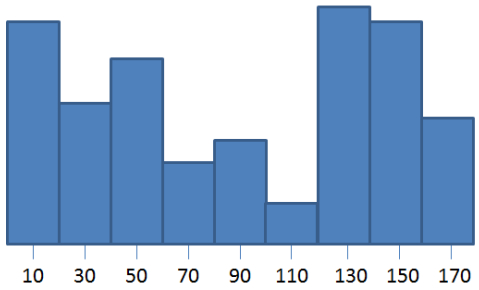

Figura 4.3

4.1.3 Contrast normalize over overlapping spatial blocks

A partire dalle celle del passo precedente si considera un blocco 2 celle × 2 celle. All'intrno di ogni blocco, vengono concatenati gli istogrammi delle 4 celle e nuovamente normalizzati secondo una normalizzazone chiamata *L2-hys*, come descritto in [2]. Tale normalizzazione prevede una normalizzazione classica seguita da un *clipping* (vengono limitati i valori dell'istogramma che superano 0.2 a 0.2).
Quindi, per ogni blocco, avremo un feature vector di 36 elementi. I blocchi risultano tra loro sovrapposti; l'effetto è che ciascuna cella compare più volte nel decrittore finale ma normalizzata con un diverso insieme di celle vicine. In particolare, le celle negli angoli dell'immagine contribuiscono per un solo blocco (4 celle), le celle ai bordi per due blocchi, mentre tutte le altre contribuiranno quattro volte.

[3]Come descritto in [2] vengono considerato, per le orientazioni del gradiente solamente l'intervallo da 0 a 180.

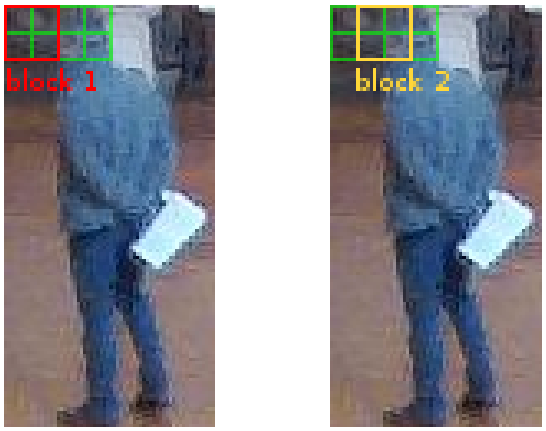

Figura 4.4: Alcuni esempi di blocchi

4.1.4 Collect HOG's over detection window

Sull'immagine viene fatta scorrere la *detection window* che si sposta di un numero prefissato di pixel ad ogni passo in entrambe le direzioni. Ricordando che la dimensione di tale finestra è 64×128 e che la dimensione dei blocchi è 16×16 con sovrapposiione di 8 pixel, all'interno di ogni detection window sono presenti

$$\frac{64-8}{8} = 7 \text{ blocchi orizzontali} \tag{4.1}$$

$$\frac{128-8}{8} = 15 \text{ blocchi verticali} \tag{4.2}$$

per un totale di 105 blocchi. Ricordiamo inoltre che ad ogni blocco corrispondono 36 features, per un totale di 3780 features associate alla singola detection window.
Tale features vengono passate in ingresso al decisore SVM precedentemente addestrato.

Capitolo 5

Implementazione del detector utilizzando OpenCV

Come detto in precedenza, è stata utilizzata l'implementazione del detector HOG di OpenCV. Per far ciò, è necessario creare un'istanza della classe che rappresenta i descrittori HOG.

```
hog = cv2.HOGDescriptor()
```

Tale metodo presenta dei parametri che lo caratterizzano (sono stati scelti i valori di default):

- `win_size=(64, 128)`: Dimensione della detection window;
- `block_size=(16, 16)`: Dimensione del blocco in pixel, supportato solo (16,16);
- `block_stride=(8, 8)`: Deve essere un multiplo di `cell_size` in quanto rappresenta lo spostamento in pixel del blocco ad ogni passo;
- `cell_size=(8, 8)`: Dimensione della cella, supportato solo (8, 8);
- `nbins=9`: numero di *bin*, cioè il numero di livelli di quantizzazione dell'orientazione del gradiente (supportati solo 9);
- `win_sigma=DEFAULT_WIN_SIGMA`: -1, parametro di *smoothing* gaussiano della finestra, -1 in quanto non utilizzato perchè provoca un deterioramento dei risultati come spiegato in [2];
- `threshold_L2hys=0.2`: Soglia massima che può raggiungere un *bin*, come spiegato in Sottosezione 4.1.3;
- `gamma_correction=true`: specifica se effettuare il processo di *gamma_correction* alla detection window in fase di preprocessing;

- `nlevels=DEFAULT_NLEVELS`: l'immagine viene analizzata a diverse scale in modo da rilevare persone più o meno vicine; in fase di detection l'immagine viene scalata iterativamente di un fattore costante e il parametro *nlevels* specifica il numero massimo di operazioni di scaling, a meno che non venga raggiunta prima la dimensione della detection window (vale 64).

Dopodiché viene richiamato il metodo:

```
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

che restituisce un'instanza di un detector SVM addestrato al rionoscimento di persone. Il detector HOG che è stato utilizzato (tramite il metodo `hog.detectMultiScale()`), è caratterizzato invece dai seguenti parametri:

- `hitThreshold 0.0`: ricordiamo che la funzione di detection di SVM, come mostrato in [1], è

$$f(x) = sign\left(\omega^T x + \omega_0\right) \tag{5.1}$$

  dove ω e ω_0 sono i parametri dell'iperpiano; dato un nuovo vettore di feature x (estrato con la tecnica descritta nel capitolo precedente) viene classificato come positivo se il valore di $f(x) = +1$ e come negativo se $f(x) = -1$. Come ulteriore vincolo è possibile specificare la distanza minima che deve avere l'esempio dal piano SVM per essere classificato correttamente come positivo, cioè

$$f(x) = sign\left(\omega^T x + \omega_0 - \text{hitTreshold}\right); \tag{5.2}$$

  un valore alto di questo parametro diminuisce il livelo di tolleranza del detector (aumentando il *miss rate*), mentre con un valore basso si ottiene l'effetto contrario (aumentando il numero di falsi positivi);

- `winStride (8,8)`: Indica il passo di spostamento delle detection windows in entrambe le direzioni. Un valore basso di questo parametro comporta una magior precisione a discapito della velocità di esecuzione, come riportato in [2]. È stato scelto il valore $(8, 8)$ come compromesso.

- `padding (32,64)`: Numero di pixel di padding da aggiungere ai bordi dell'immagine al fine di rilevare persone vicine ai bordi; tale padding è di tipo simmetrico, cioè viene replicata l'immagine in maniera speculare; questo valore è stato scleto in quano si presuppone che almeno metà persona compaia nella scena ed è pertanto la metà della detection window;

- `scale 1.02`: Coefficiente di scaling dell'immagine in fase di detection multiscale; l'immagine viene più volte rimpicciolita di questo fattore di scala al fine di rilevare persone a diverse distanze; la scelta di questo prametro è stata fatta considerando i dati di 2000 frames (500 per ogni telecamera, di cui 250 dello scenario individuals e 250 dello scenario groups) e calcolando il valore del *miss rate* al variare della soglia tra 1.01 e 1.15 con passo 0.01. Come si nota dalla Figura 5.1, il valore che

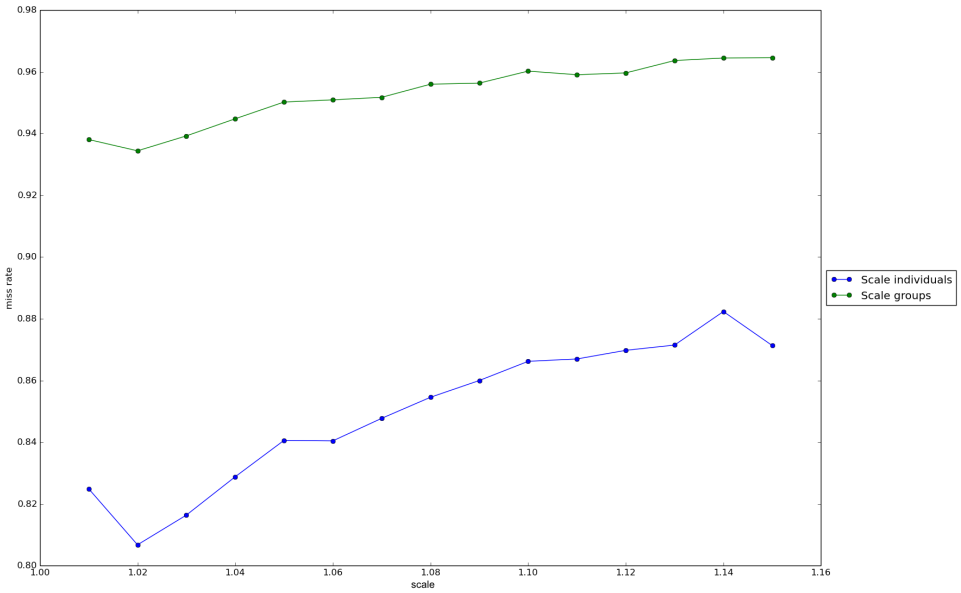

Figura 5.1

rende più basso il *miss rate* in entrambi gli scenari è 1.02.

- `finalThreshold`: Soglia usata per la soppressione dei non massimi, ha due utilizzi: se `useMeanshiftGrouping` vale *true*, viene utilizzata come soglia per l'algoritmo di `MeanshiftGrouping`, altrimenti viene convertito ad intero ed utilizzato come parametro per un differente algoritmo di NMS;

- `useMeanshiftGrouping true`: determina quale algoritmo di NMS utilizzare.

OpenCV mette a disposizione due algoritmi di NMS:

1. `MeanshiftGrouping`: effettua tale operazione utilizzando l'algoritmo *MeanShift*;

2. `groupRectangles()`: riunisce in *cluster* le detection windows di dimensione e posizione simile e permette che ci siano nello stesso cluster al massimo un numero di *bb* pari al valore di soglia.

Capitolo 6

Background subtraction

Il semplice utilizzo dell'*HOG Detector* comporta una certa quantità di falsi positivi, alcuni dei quali dovuti alla presenza di statue o finestre erroneamente identificate come persone. Per cercare di filtrare questi risultati, è stato pensato a posteriori l'utilizzo dell'algoritmo di *Background Subtraction*. Questo algoritmo prende in ingresso una sequenza di immagini e, studiando i pixel che variano tra un'immagine e l'altra, crea una maschera che distingue il background dal foreground di una nuova immagine. Questo può essere effettuato sottraendo da un immagine la precedente e raffinando i risultati con un *thresholding*. Tuttavia questo metodo comporta varie problematiche: richiede la presenza di immagini di training con solo elementi di background; non funziona correttamente in caso di variazioni di illuminazione. Per ovviare a tali problematiche è necessario creare un modello per il background ed aggiornare tale modello ad ogni nuova immagine.
Una possibilità è quella di utilizzare un modello adattivo non parametrico basato sulla mistura di Gaussiane per stimare la probabilità che un determinato pixel assuma un certo valore al fine di determinare se il pixel corrente è di background o foregound.

Nello specifico, ogni pixel della scena è modellato da una mistura di K distribuzioni Gaussiane. La probabilità che un certo pixel abbia un valore x_N al tempo N è

$$p(x_n) = \sum_{j=1}^{K} w_j \eta(x_N; \theta_j) \tag{6.1}$$

dove w_j è il peso della j-esima componente Gaussiana, $\eta(x; \theta_j)$ è la distribuzione Normale della j-esima componente, cioè

$$\eta(x; \theta_j) = \eta(x; \mu_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)} \tag{6.2}$$

dove μ_j e il valor medio e Σ_j è la covarianza della j-esima componente.

Le K distribuzioni vengono ordinate in base al rapporto w_j/σ_j e solo le prime B distribuzioni sono usate per modellare il background della scena, dove

$$B = \arg\min_b \left(\sum_{j=1}^{b} w_j > T \right) \tag{6.3}$$

con T determinata in maniera adattiva. L'algoritmo indentifica un pixel come di foregound se la deviazione standard del pixel è maggiore di 2.5 volte la varianza per ognuna delle B distribuzioni. Se viceversa esiste una distribuzione per cui la deviazione standard è minore di 2.5 volte la varianza della distribuzione, allora il pixel è ritenuto modellato da tale distribuzione ed etichettato come di background.

Al fine di aggiornare i valori dei pesi w_j e i parametri delle Gaussiane μ_j e Σ_j si considerano solo le L immagini più recenti. Nello specifico

$$\begin{aligned} w_j^{N+1} &= w_j^N + \frac{1}{L}\left(\hat{p}(\omega_j|x_{N+1}) - w_j^N\right) \\ \mu_j^{N+1} &= \mu_j^N + \frac{1}{L}\left(\frac{\hat{p}(\omega_j|x_{N+1})x_{N+1}}{w_j^{N+1}} - \mu_j^N\right) \\ \Sigma_j^{N+1} &= \Sigma_j^N + \frac{1}{L}\left(\frac{\hat{p}(\omega_j|x_{N+1})(x_{N+1}-\mu_j^N)(x_{N+1}-\mu_j^N)^T}{w_j^{N+1}} - \Sigma_j^N\right) \end{aligned} \tag{6.4}$$

dove $\hat{p}(\omega_j|x_{N+1})$ è una variabile indicatore definita come

$$\hat{p}(\omega_j|x_{N+1}) = \begin{cases} 1 \text{ se } \omega_j \text{ è la prima Gaussiana che soddisfa il modello} \\ 0 \text{ altrimenti} \end{cases}$$

Per ulteriori informazioni sul funzionamento dell'algoritmo si rimanda a [5]. Ancora una volta, è stata utilizzata l'implementazione dell'algoritmo fornita da *OpenCV*. In particolare è stata utilizzata la funzione *BackgroundMOG*.

Una volta estratta la maschera per ogni immagine, le *bb* vengono filtrate in base alla percentuale di pixel di foreground che contengono: se tale percentuale è inferiore ad una certa soglia, la *bb* viene scartata. Tale soglia è stata scelta *ad hoc* per il caso in esame, come spiegato in Sezione 7.1.

Capitolo 7

Risultati ottenuti

Per l'analisi dei risultati, è stato fatto riferimento a [3]. Gli esperimenti sono stati fatti prendendo in esame i seguenti algoritmi:

- *HOG*: il detector di HOG utlizzando *MeanShift* come NMS e con i parametri riportati in Sezione 5;
- *HOG_BG_EQUA*: come *HOG*, filtrando i risultati utilizzando l'algoritmo di *Background Subtraction* e preprocessando le immagini con un'equalizzazione, come riportato in Sezione 3;
- *HOG_BG_HOMO*: come *HOG*, filtrando i risultati utilizzando l'algoritmo di *Background Subtraction* e preprocessando le immagini con l'applicazione del filtro omomorfico, come riportato in Sezione 3.

Come in [3], le prestazioni degli algoritmi sono state comparate utilizzando le curve *ROC* (Receiver operating characteristic). In particolare, sono state disegnate le curve al variare dei valori di *miss rate* e *FPPI*:

- *miss rate*: la percentuale di persone non rilevate, calcolato come:

$$miss\_rate = \frac{FN}{TP + FN} \tag{7.1}$$

- *FPPI*: *False positive per image*, il numero di falsi positivi totali diviso il numero delle immagini in esame.

Le curve *ROC* sono state disegnate al variare del parametro `hitThreshold` che, come descritto nel Capitolo 5, è la distanza minima che deve avere un *feature vector* dal piano SVM per essere classificato correttamente. Sono stati presi in considerazione 16 valori equispaziati nell'intervallo $[0, 2)$.

A partire dalle *bb* contenute nella Ground Truth, c'è la necessità di capire quando si può assumere che il detector abbia rilevato correttamente una

bb e quando invece il valore fornito si discosta troppo dalla soluzione. Si possono verificare vari scenari ben descritti in [3]. Per poter distinguere un falso positivo da un falso negativo, allora, vengono considerati in sequenza tutti gli elementi della GT, vengono scorse quindi tutte le *bb* rilevate dal detector e si verifica se l'area in comune tra le due è superiore al 50% dell'area della *bb*: se questo accade, le due *bb* vengono associate e la *bb* del detector non può essere più associata ad un'altra *bb* della GT; se questo non accade si ha un falso negativo. Le *bb* del detector che risultano non associate dopo aver scorso tutte quelle della GT, invece, costituiscono i falsi positivi.

7.1 Scelta della soglia per il postprocessing

L'algoritmo di *Background Subtraction* viene utilizzato per discriminare il background dal foreground e, in fase di postprocessing, per scartare le *detected windows* che non contengono una determinata percentuale di foreground, come descritto nel Capitolo 6. Tale percentuale è il valore di soglia da determinare.

Sono stati presi in considerazione i tre algoritmi (*HOG*, *HOG_BG_HOMO* e *HOG_BG_EQUA*) descritti in precedenza e, per ognuno, è stata cercata la soglia ottima. Sono state calcolate 15 curve ROC, nel caso d'interesse *individuals*, al variare del valore di soglia con passo costante tra 0.01 e 0.15. Le curve sono state generate facendo variare il parametro *hitThreshold* su 20 punti equispaziati nell'intervallo $[0, 2)$.
Per comparare le curve, è stato utilizzato l'approccio descritto in [3], dove si considera un valore di *FPPI* fisso (in questo caso 10^{-1}) ed è stato valutato il valore di *miss rate*. E' stato scelto come valore di soglia ottimo il minor *miss rate* per ciascun caso.

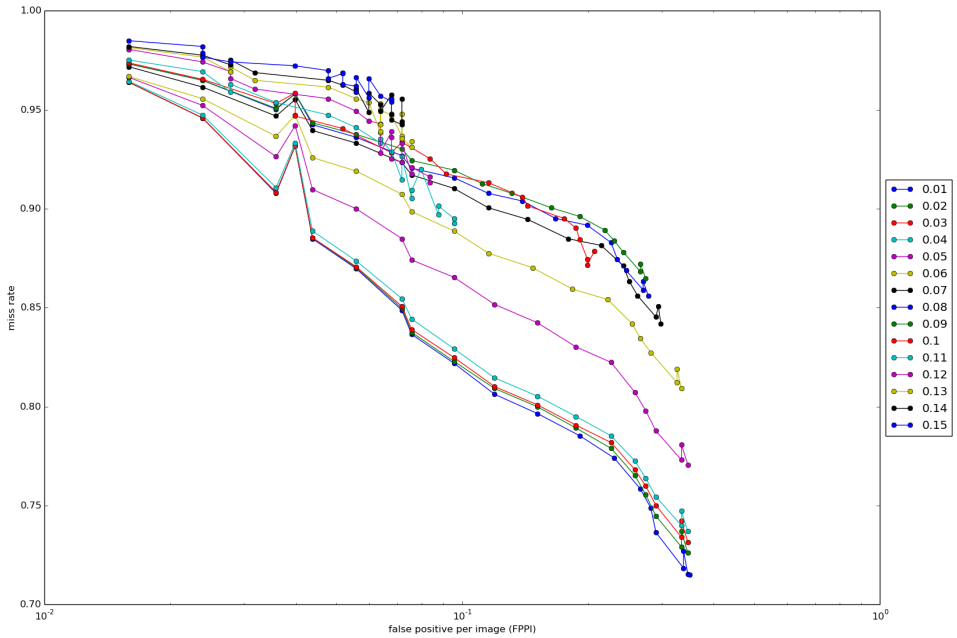

Figura 7.1: Curve ROC per i vari valori di soglia di *Background Subtraction* considerando l'algoritmo *HOG*. Le curve sono state disegnate al variare del parametro *hitThreshold*, considerando *miss rate* sull'asse delle ordinate e *FPPI* sull'asse delle ascisse.

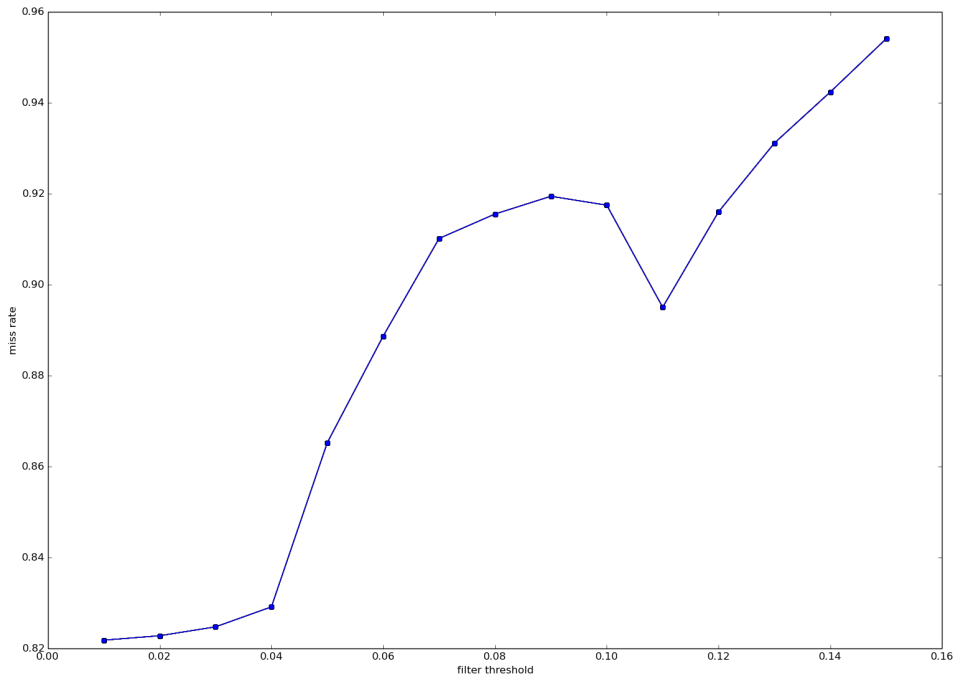

Figura 7.2: Valori di *miss rate* (sull'asse delle ordinate) per *FPPI* al valore 10^{-1} delle curve ROC di figura 7.1 per i vari valori di soglia di *Background Subtraction* (sull'asse dele ascisse). Il valore ottimo di soglia per l'algoritmo *HOG* è 0.01 (1%)

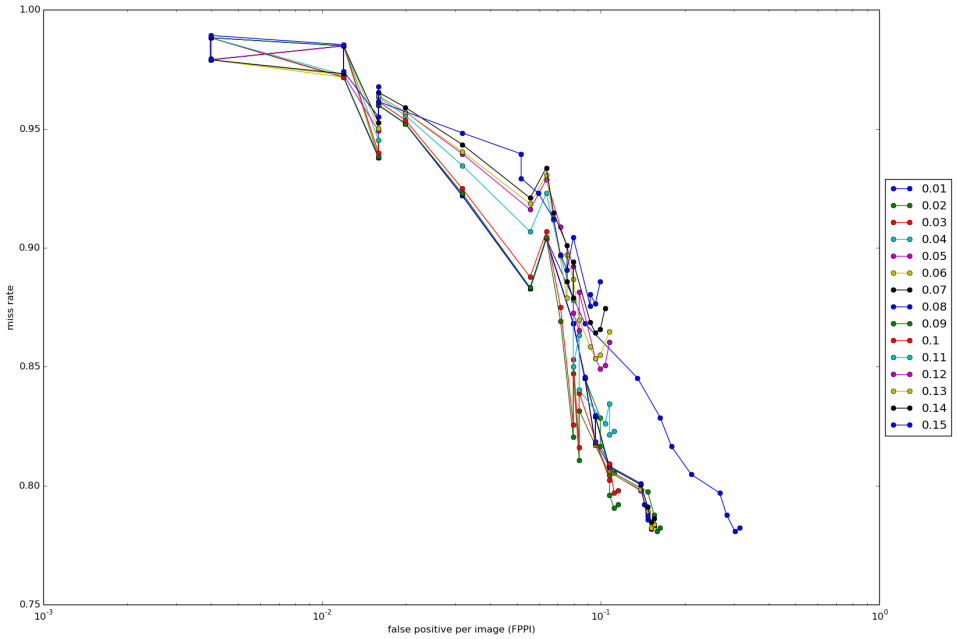

Figura 7.3: Curve ROC per i vari valori di soglia di *Background Subtraction* considerando l'algoritmo *HOG_BG_EQUA*. Le curve sono state disegnate al variare del parametro *hitThreshold*, considerando *miss rate* sull'asse delle ordinate e *FPPI* sull'asse delle ascisse.

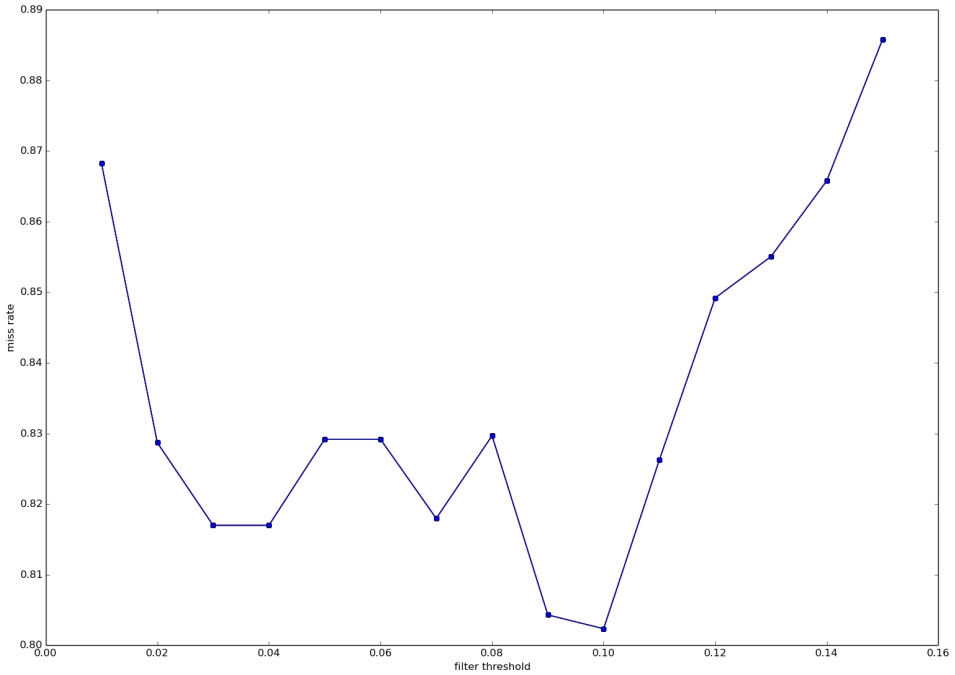

Figura 7.4: Valori di *miss rate* (sull'asse delle ordinate) per *FPPI* al valore 10^{-1} delle curve ROC di figura 7.3 per i vari valori di soglia di *Background Subtraction* (sull'asse dele ascisse). Il valore ottimo di soglia per l'algoritmo *HOG_BG_EQUA* è 0.1 (10%)

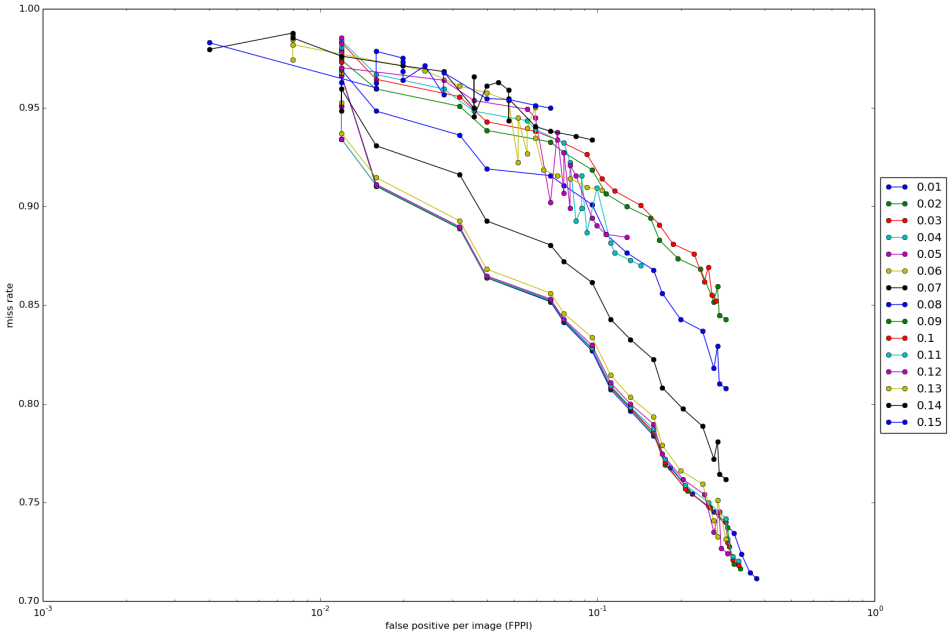

Figura 7.5: Curve ROC per i vari valori di soglia di *Background Subtraction* considerando l'algoritmo *HOG_BG_HOMO*. Le curve sono state disegnate al variare del parametro *hitThreshold*, considerando *miss rate* sull'asse delle ordinate e *FPPI* sull'asse delle ascisse.

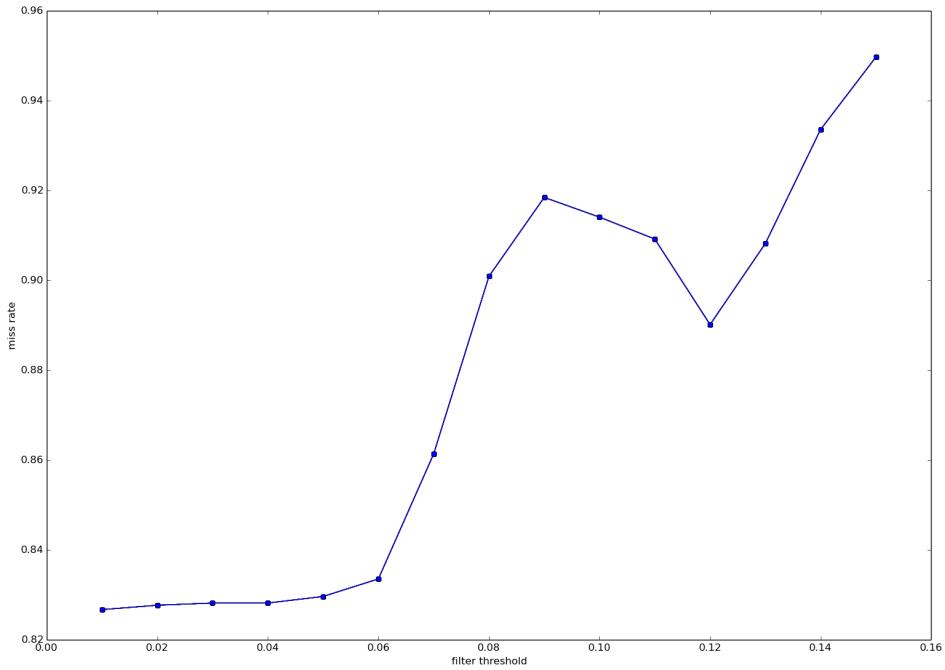

Figura 7.6: Valori di *miss rate* (sull'asse delle ordinate) per *FPPI* al valore 10^{-1} delle curve ROC di figura 7.5 per i vari valori di soglia di *Background Subtraction* (sull'asse dele ascisse). Il valore ottimo di soglia per l'algoritmo *HOG_BG_HOMO* è 0.01 (1%)

7.2 Comparazione tra gli algoritmi

Come descritto in precendeza, è stata generata una serie di curve ROC per ogni scenario di interesse. In particolare sono stati presi in esame i casi *individuals* e *groups* e comparate le prestazioni degli algoritmi *HOG*, *HOG_BG_EQUA* e *HOG_BG_HOMO* nei suddetti scenari. In ognuno dei sei casi, sono state considerate separatamente le curve ROC generate da le quattro telecamere e successivamente calcolata la curva media tramite interpolazione.
Per ogni telecamera sono stati considerati 250 frames al fine di generare la curva, facendo ancora una volta variare il parametro *hitThreshold* nell'intervallo $[0, 2)$ scegliendo 15 valori uniformemente.

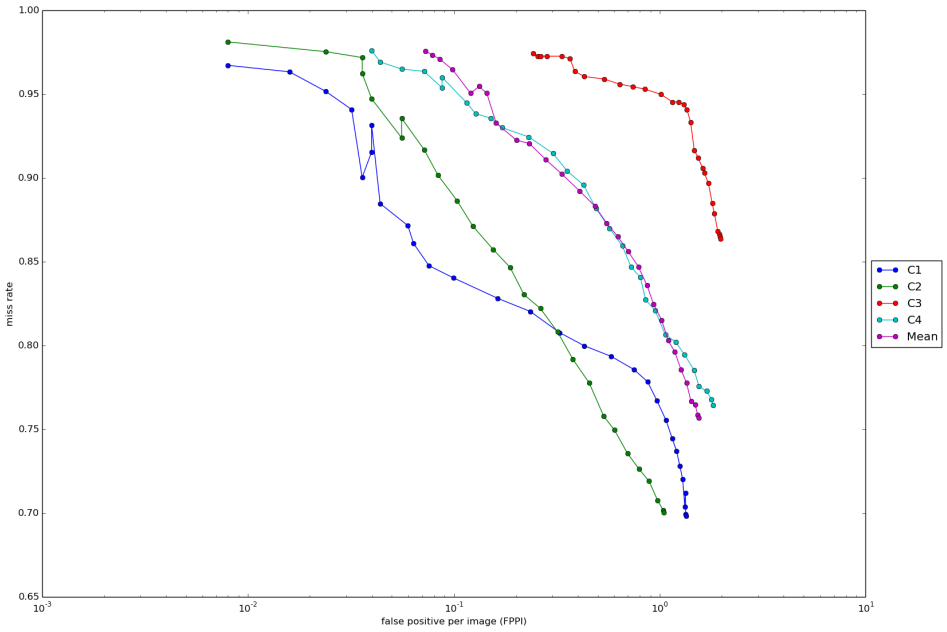

Figura 7.7: Curva ROC per il caso d'interesse *individuals* relativa all'algoritmo *HOG* per le quattro telecamere

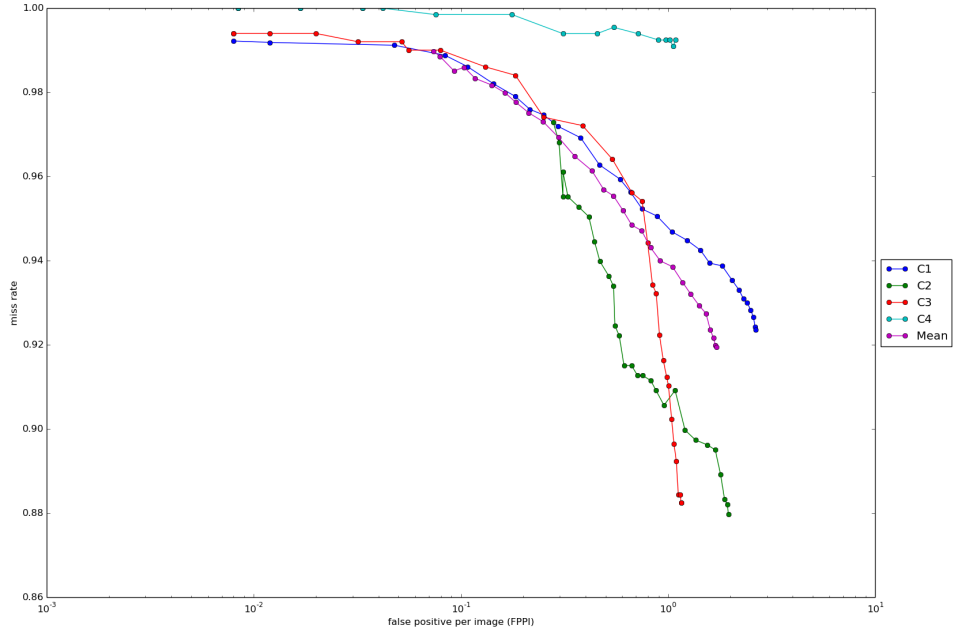

Figura 7.8: Curva ROC per il caso d'interesse *groups* relativa all'algoritmo *HOG* per le quattro telecamere

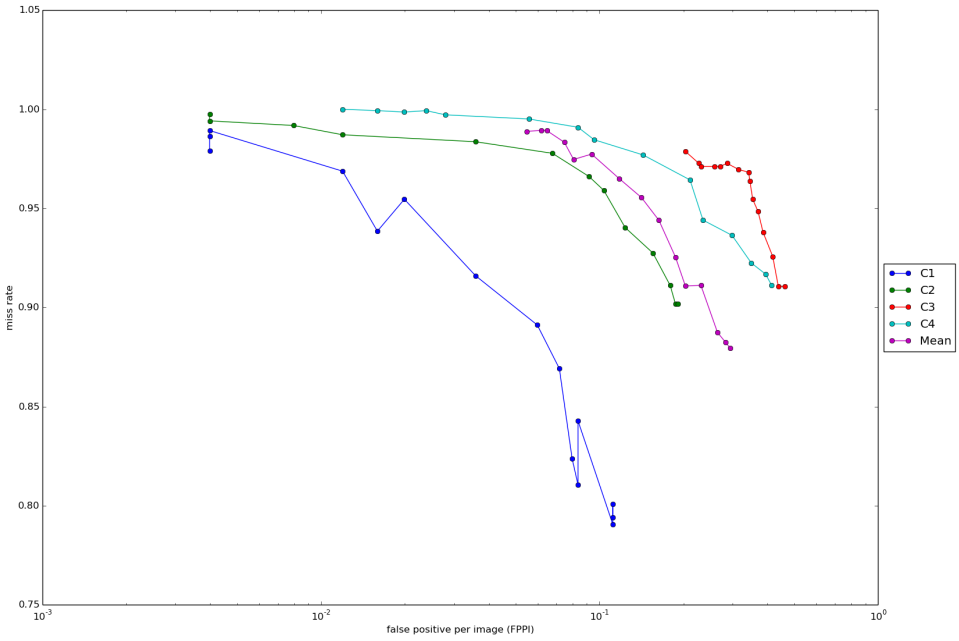

Figura 7.9: Curva ROC per il caso d'interesse *individuals* relativa all'algoritmo *HOG_EQUA_BG* per le quattro telecamere

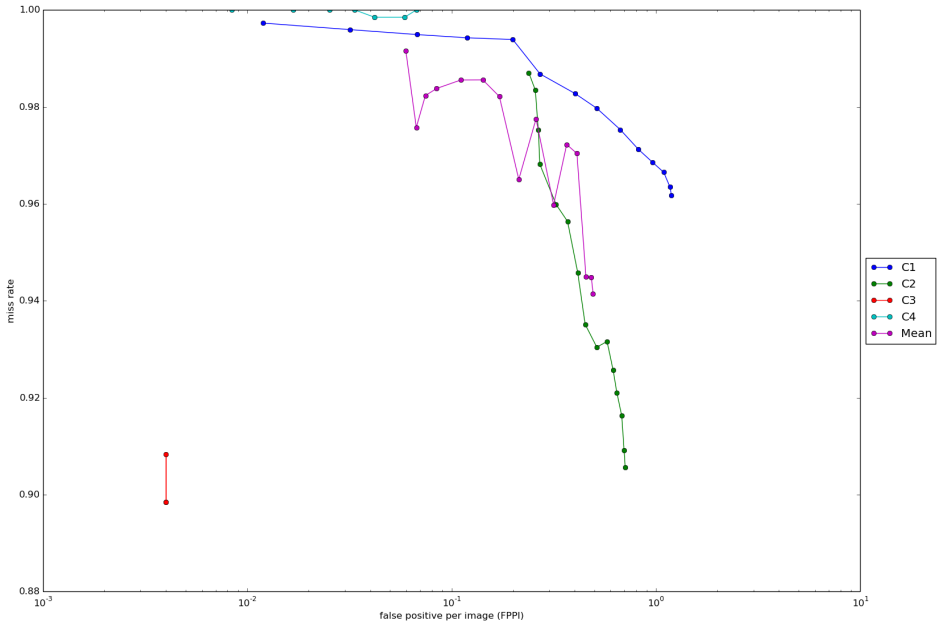

Figura 7.10: Curva ROC per il caso d'interesse *groups* relativa all'algoritmo *HOG_EQUA_BG* per le quattro telecamere

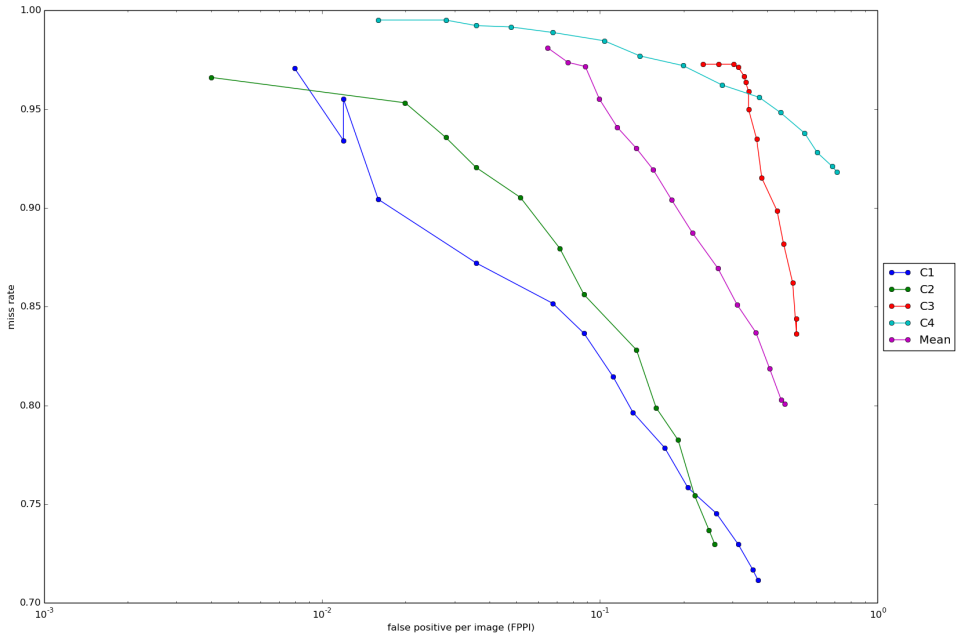

Figura 7.11: Curva ROC per il caso d'interesse *individuals* relativa all'algoritmo *HOG_HOMO_BG* per le quattro telecamere

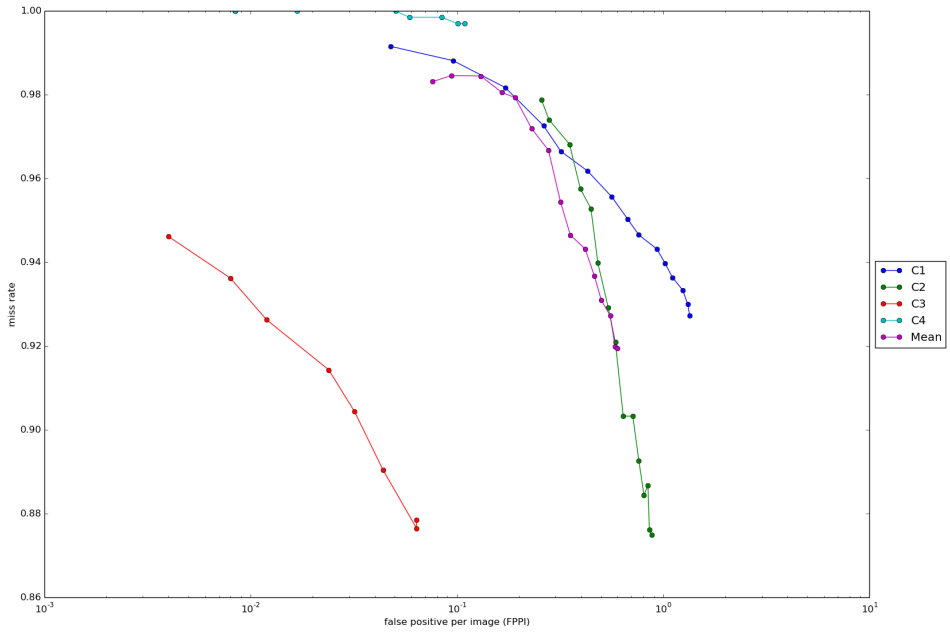

Figura 7.12: Curva ROC per il caso d'interesse *groups* relativa all'algoritmo *HOG_HOMO_BG* per le quattro telecamere

Si nota che le prestazioni del detector sono molto condizionate dal dataset: in alcune camere (ad esempio C3 in *groups* per *HOG_EQUA_BG* o C4 in *groups* per *HOG_HOMO_BG*) la curva ROC ottenuta discosta molto dai valori delle altre camere. Questo è dovuto alla parziale occlusione delle persone nella scena, dalla notevole distanza degli individui dalla camera.

Le ROC medie ottentue tramite interpolazione sono state raggruppate in uno stesso grafico per poter comparare le prestazioni degli algoritmi negli scenari *individuals* e *groups*. Si nota, come ci si potrebbe aspettare, che lo scenario *individuals* presenta prestazioni migliori.

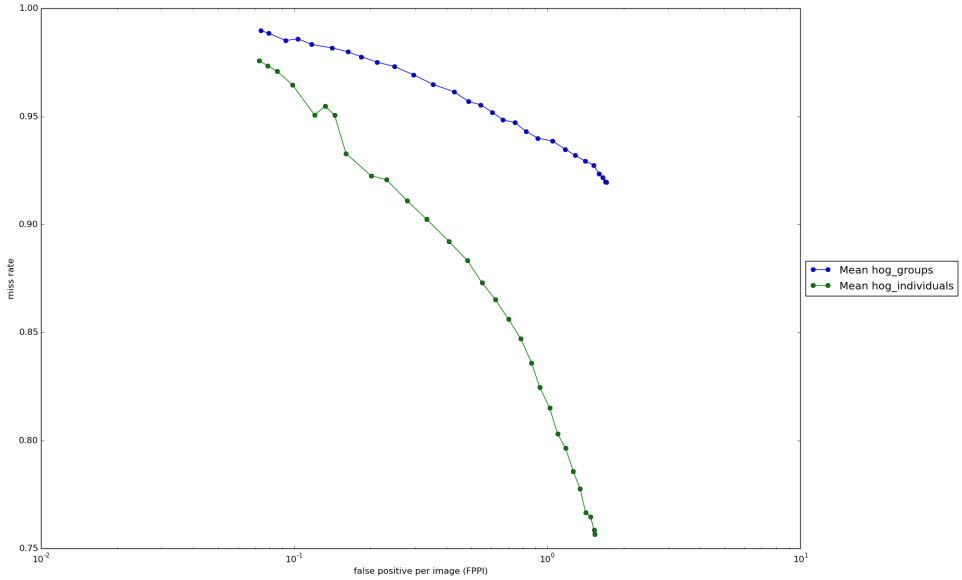

Figura 7.13: Curve ROC medie sulle quattro telecamere relative all'algoritmo *HOG*

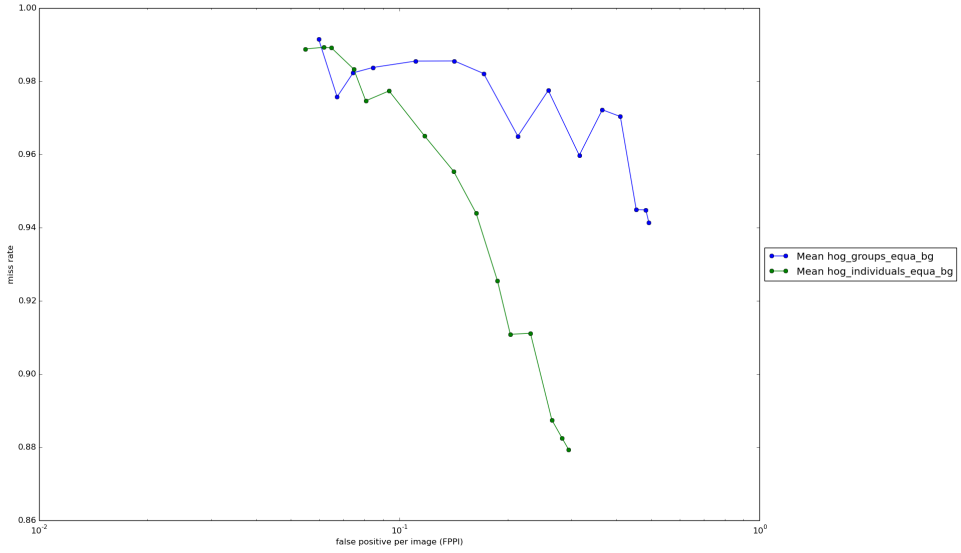

Figura 7.14: Curve ROC medie sulle quattro telecamere relative all'algoritmo *HOG_EQUA_BG*

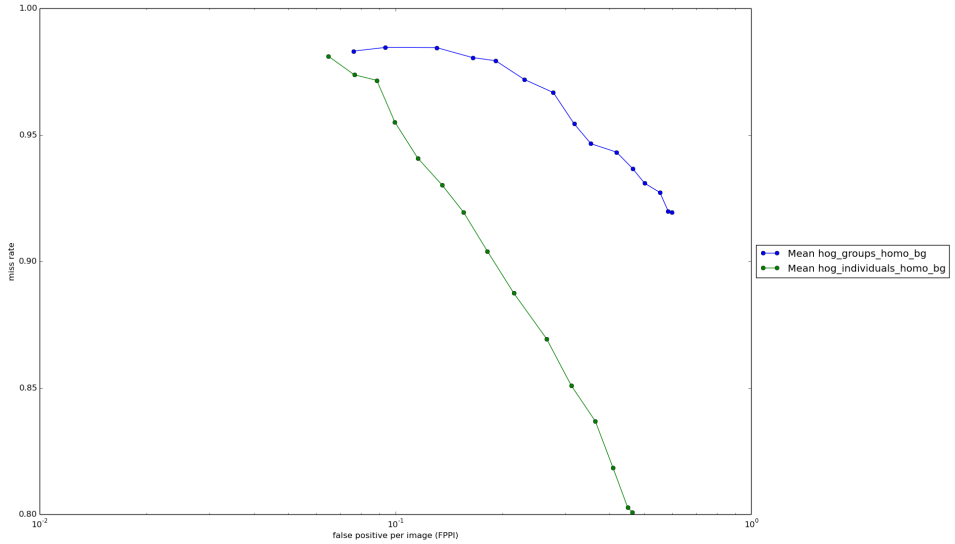

Figura 7.15: Curve ROC medie sulle quattro telecamere relative all'algoritmo *HOG_HOMO_BG*

Infine, per effettuare un'analisi quantitativa della qualità degli algoritmi presi in esame nei due casi di interesse, sono state considerate le ROC medie per le quattro telecamere e, come proposto in [3], è stato considerato il valore di *miss rate* corrispondente ad un valore di *FPPI* di 10^{-1}. Si considera quindi come migliore l'algoritmo con tale *miss rate* più basso.

	Individuals	**Groups**
HOG	0.97	0.985
HOG_BG_EQUA	0.975	0.985
HOG_BG_HOMO	0.96	0.985

Tabella 7.1

7.3 Stima della dimensione del campione

Per valutare il numero di frames che devono comporre un campione affinché questo risulti singificativo ai fini di una valutazione quantitativa sulla stima dell'errore, sono state generate varie curve ROC al variare del numero di campioni. Tali ROC sono state calcolate nello scenario *individuals* sulla camera 1 ed utilizzando l'agoritmo *HOG*.

Sono stati considerati 1000 frames ed effettuate 10 prove campionando uniformemente i suddetti frames. Le curve ROC sono state generate considerando 100, 200, 300, 400, 500, 600, 700, 800, 900 e 1000 frames.

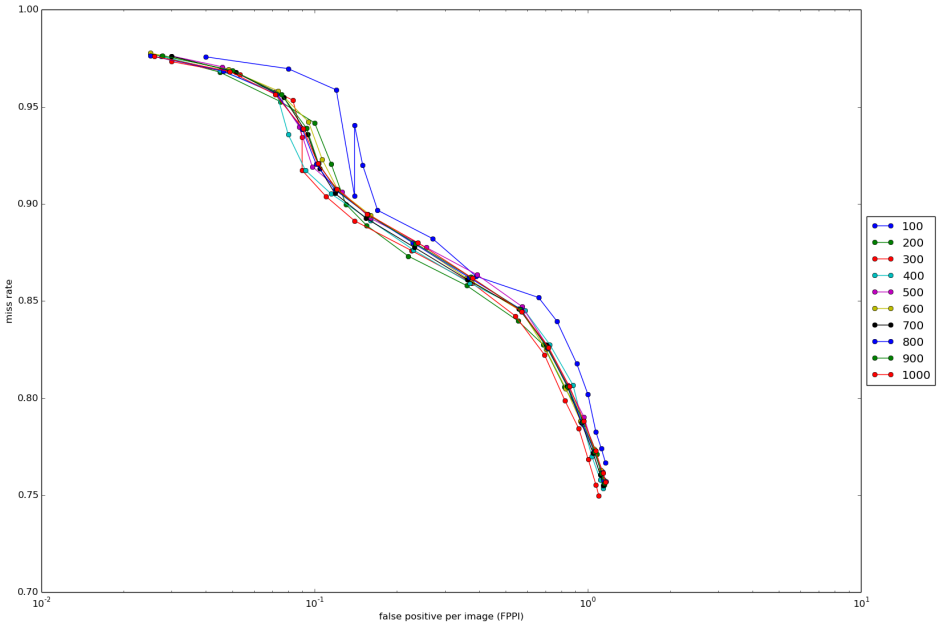

Figura 7.16: Curve ROC relative all'algoritmo *HOG* al variare del numero di frames

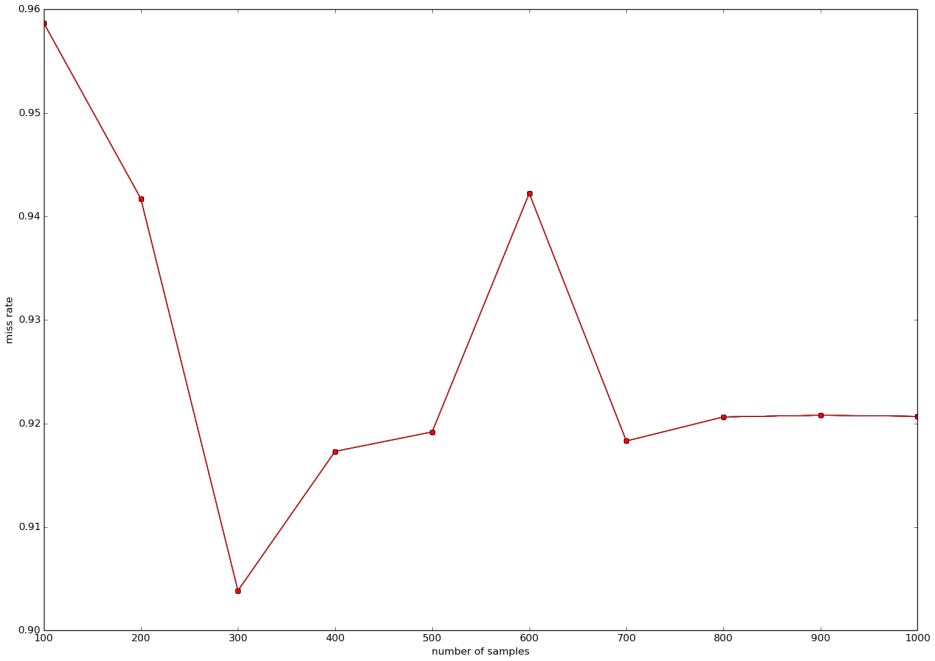

Figura 7.17: Andamento del *miss rate* corrispondente al valore di *FPPI* di 10^{-1} in funzione del numero di frames scelti per generare le ROC

Si osserva che l'andamento del *miss rate* (come riportato in figura 7.17) si stabilizza dopo 700 frames. Pertanto per avere una più corretta analisi quantitativa occorre etichettare (in questo scenario) almeno 700 frames.

Capitolo 8

Conclusioni

Per i casi da noi studiati risulta che nello scenario *individuals*, in cui sono presenti poche persone e ben distinguibili, il *miss rate* è più basso rispetto allo scenario *groups*, in cui si ha occlusione parziale o totale delle persone.

Dalla Tabella 7.1 si nota che l'introduzione della fase di preprocessing iniziale e postprocessing tramite il *Background Subtraction* porta dei miglioramenti nello scenario *individuals* ma non ne porta di significativi nello scenario *groups* in cui i problemi sopracitati prevalgono.

Inoltre, visivamente, si nota che le prestazioni del detector sono buone per persone a distanza medio-corta, mentre degradano se la distanza aumenta o in caso di introduzione di distorsione prospettica.
L'analisi effettuata è quella del caso medio sulle quattro telecamere, che, come si può notare dalle figure 7.7, 7.8, 7.9, 7.10, 7.11 e 7.12, è un'analisi poco informativa in quanto la varianza dei risultati è molto elevata e questi dipendono strettamente dal dataset scelto.

Una possibile strategia per migliorare i risultati è quella di combinare le informazioni delle quattro telecamere tramite algoritmi di visione.

Infine, dall'analisi svolta in Sezione 7.3, i test effettuati sono stati fatti su un numero ridotto di frames, rispetto a quelli stimati per un'analisi quantitativa accurata, in quanto, avendo a disposizione risorse computazionali limitate, si è preferito coprire tutti gli scenari riducendo la precisione dei risultati.

Bibliografia

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine Learning, 20(3):273–297, 1995.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, International Conference on Computer Vision & Pattern Recognition, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[3] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. IEEE Trans. Pattern Anal. Mach. Intell., 34(4):743–761, April 2012.

[4] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[5] P. KaewTraKulPong and R. Rowden. An improved adaptive background mixture model for real-time tracking with shadow detection. Proceedings of the Second European Workshop on Advanced Video Based Surveillance Systems, pages 149–158, 2001.

[6] U. Nnolim and P. Lee. Homomorphic filtering of colour images using a spatial filter kernel in the hsi colour space. In Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE, pages 1738–1743, May 2008.