UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA DELL'INFORMAZIONE

Action Recognition tramite HMM in Video mono e multi azione

Relatori:
Prof. Rita Cucchiara
Prof. Paolo Tiberio

Correlatori:
Ing. Roberto Vezzani

Candidati:
Andrea Corbelli
Andrea Palazzi
Guido Borghi

Anno Accademico 2013/2014

Indice

Capitolo 1

Introduzione

La classificazione di azioni è un task che sta assumendo sempre maggiore importanza, specialmente in applicazioni di video surveillance automatizzata, le cui principali sfide sono il riconoscimento di più tipi di azioni, anche molto simili, e anche in casi in cui sono presenti più soggetti o essi sono parzialmente nascosti.
Gli approcci possibili sono due, il primo sfrutta solamente immagini 2D mentre il secondo dati spaziali in un contesto 3D, anche se in ogni caso nel contesto della video surveillance raramente si dispone di videocamere RGBD. Nel caso di approcci bidimensionale esistono due categorie principali di metodologie, quelle che analizzano dei *feature points* e quelle che invece analizzano l'intera *silhouette* della persona che compie l'azione.
La classificazione delle azioni è ovviamente supportata dal machine learning, per questo la scelta delle feature risulta di importanza fondamentale.

1.0.1 Panoramica del Progetto

Nell'ambito di questo progetto ci si è basati su un approccio bidimensionale ispirato a quello decritto nel lavoro di Vezzani [7], partendo da video ricavati da telecamere fisse, in particolare quelli del Weizmann dataset, descritto in dettaglio in sezione 4.3. Esso contiene quasi un centinaio di video di persone che compiono una decina di azioni diverse; per ogni video compare solamente una persona alla volta.
Si è partiti da questi video per analizzarli in tre step principali, riassunti nello schema in Figura 1.1:

1. **Background Subtraction**: dai frame originali viene ricavato il foreground, ovvero le zone del video in movimento, per isolare la silhouette della persona. La tecnica impiegata è la Mixture Of Gaussians, analizzata in dettaglio nel capitolo 2.

2. **People Detection**: nell'area in cui viene rilevato il movimento si cerca di individuare, attraverso la people detection, effettuata tramite HOG e SVM, se è presente o meno una persona. In caso affermativo si passa allo step successivo.

3. **Classification**: partendo dalla silhoutte ricavata vengono calcolate le feature necessarie per la classificazione, per la quale vengono utilizzati degli Hidden Markov Model, particolarmente adatti al riconoscimento di sequenze temporali, ad esempio largamente utilizzati nella *speech recognition* e nel riconoscimento dell'handwriting.

Maggiori dettagli riguardanti tutti e tre gli step vengono forniti nei capitoli successivi. Nell'ultimo capitolo vengono illustrati i risultati ottenuti.

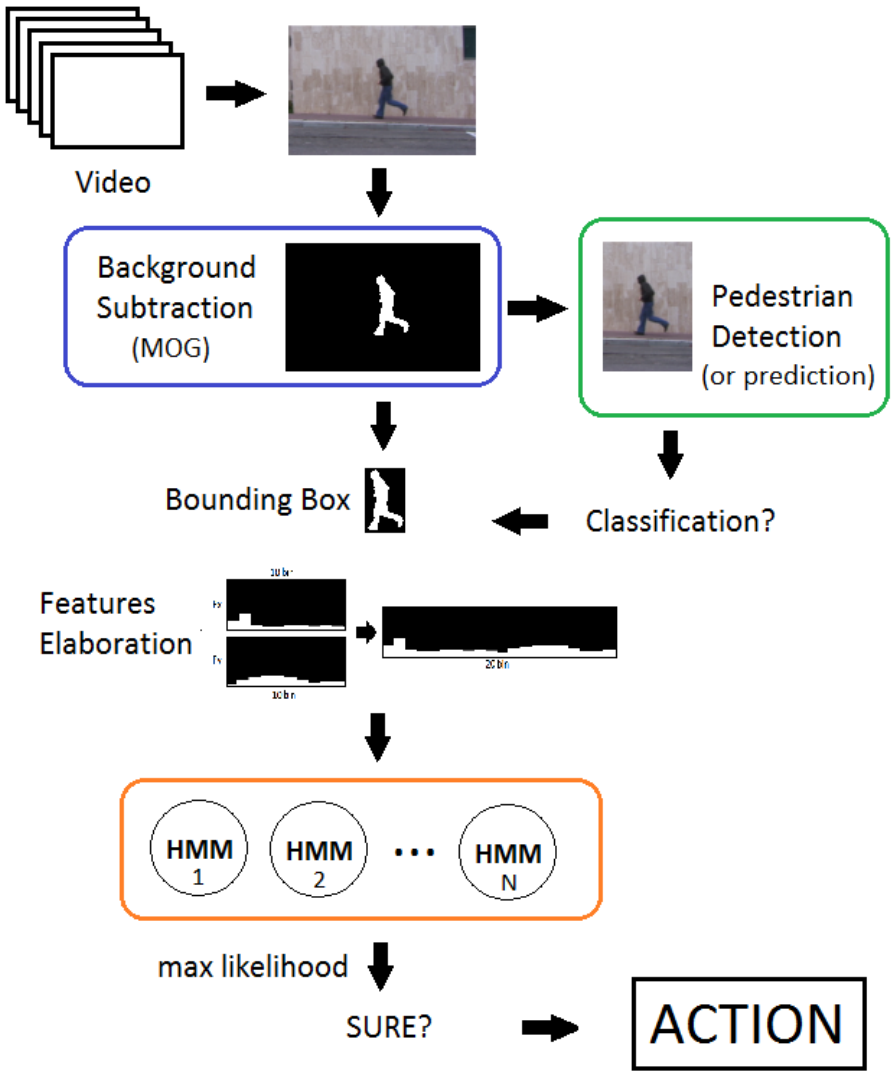


Figura 1.1: Struttura del progetto

Capitolo 2

Background Subtraction

2.1 Introduzione

La background subtraction è una tecnica del campo della visione artificiale attraverso la quale è possibile estrarre da un video un'immagine di foreground, separandola quindi dallo sfondo, per ulteriori analisi. Questo approccio è largamente utilizzato in situazioni in cui i video provengono da telecamere statiche, ossia quando la distinzione tra gli oggetti in movimento e lo sfondo è netta.
In generale, il foreground che si è interessati a trovare consiste di oggetti, spesso in movimento, che devono essere individuati in quanto punti di interesse di un certo processo di analisi (ad esempio si possono cercare persone, animali, auto, ecc.). Raramente la background subtraction è fine a se stessa, solitamente è un punto di partenza per altri procedimenti che, attraverso un reasoning successivo, sono in grado di estrarre risultati di più alto livello.
Un tipico esempio di campo di applicazione di tecniche di background subtraction è la video surveillance, dove può ricoprire un ruolo utile per l'individuazione di zone di interesse in movimento o aiutare altri procedimenti di tracking di oggetti.
Diversi fattori rendono questo task impegnativo. Se si esclude il problema più banale, ossia il movimento accidentale della telecamera, in contesti reali quello che viene percepito da noi esseri umani come background è in realtà tutt'altro che fisso e immutabile. Sia in contesti indoor che outdoor molti elementi intervengono su di esso, nel caso outdoor ad esempio il passare del tempo provoca cambiamenti di luminosità (giorno e notte), alcuni oggetti potrebbero esesre presenti nello sfondo solo in determinati momenti (auto parcheggiate, piante di forma diversa a seconda della stagione, ecc.) e così via, mentre nel caso indoor si possono avere spostamenti di oggetti, riflessi

indesiderati, immagini in movimento su schermi ecc.

Da quanto detto si deduce che non è possibile utilizzare un solo e immutabile background model ma è necessario aggiornarlo in modo adattivo a seconda dei cambiamenti che intervengono sulla scena monitorata. Esistono molte tecniche di background subtraction: le più semplici comportano il calcolo di differenze tra frame successivi (single difference, double difference, ecc.), in queste tecniche il background model è individuato nei frame precedenti del video; altre tecniche più sofisticate individuano un background model attraverso modelli statistici, delle quali l'esempio più famoso è certamente la tecnica chiamata MOG - Mixture Of Gaussians, utilizzata anche nell'ambito di questo progetto.

2.2 Mixture Of Gaussians

La tecnica Mixture Of Gaussians utilizzata per la background subtraction [5] è una delle più famose e utilizzate e si propone di modellare ogni pixel indipendentemente dagli altri in base a due o più modelli di background presenti contemporaneamente.
Alcuni dei suoi punti di forza sono quelli di essere una tecnica completamente general purpose, non è infatti studiata per ambienti o background specifici, e di non richiedere alcuna conoscenza precedente del background da modellare. È infatti solamente necessario impostare alcuni parametri a seconda dell'ambiente in cui ci si trova a lavorare.
Più formalmente, per un pixel la probabilità di appartenere al background è data dalla somma pesata di K modelli gaussiani (solitamente K è compreso tra 3 e 5).

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

Dove $\omega_{i,t}$ è il peso relativo a quel modello gaussiano e $\eta(X_t, \mu_{i,t}, \Sigma_{i,t})$ è il i-esimo modello gaussiano considerato.

$$\eta(X_t, \mu_{i,t}, \Sigma_{i,t}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(X_t-\mu_t)^T\Sigma^{-1}(X_t-\mu_t)}$$

I vari $\omega_{i,t}$ oltre che essere i pesi delle i-esime gaussiane, stabiliscono anche l'*ordine* delle gaussiane stesse, gaussiane con peso maggiore avranno infatti importanza maggiore e rappresenteranno il background model principale per un determinato istante di tempo t.
La gaussiana con peso ω minore rappresenta il *foreground* mentre le altre

$K-1$ rappresentano i *background models*.
L'utilizzo di questa tecnica prevede a questo punto tre operazioni, da ripetere per ogni nuova osservazione, nel caso di video ciò corrisponde all'analisi del frame successivo.

2.2.1 Comparison

Ad ogni istante di osservazione t, ogni pixel del nuovo frame viene confrontato con tutti i K modelli di background ed è associato ad uno di essi, in particolare quello più vicino.
Si intende che un pixel *appartiene* a un background model se la differenza tra il suo valore e quello del background model è minore di 2,5.
Un pixel può ovviamente anche non appartenere a nessun background model e fare parte del foreground, questo accade quando esso risulta essere associato al modello gaussiano con peso minore oppure a nessuno di essi.

2.2.2 Update

Ad ogni passo è necessario aggiornare i pesi delle gaussiane, in modo da poter aggiornare l'importanza dei modelli di background per poter reagire alle modifiche del background reale della scena.

1. Il peso delle gaussiane che ottengono un match viene incrementato
$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha$$
2. Il peso delle gaussiane che *non* ottengono un match viene decrementato
$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1}$$
3. Per completare l'aggiornamento del modello viene determinato un fattore ρ:
$$\rho = \alpha * \eta(X_t|\mu_k, \sigma_k)$$
in base al quale vengono ricalcolate media e varianza di ogni gaussiana
$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$
$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)$$

Il parametro α è da intendersi come un *learning rate*. Aumentandolo o diminuendolo si può infatti stabilire quanto sarà veloce o lento l'aggiornamento dei modelli di background. Questo è molto importante in quanto permette di adattare questo metodo al contesto in cui viene utilizzato, background molto dinamici richiederanno infatti learning rate più alti e viceversa.

2.2.3 New Models

È possibile anche l'eliminazione di una gaussiana, questo accade quando quella con peso minore non ottiene alcun match con il pixel di un frame. In questo caso essa viene eliminata e e sostituita da una nuova creata utilizzando il pixel di foreground.
Grazie ai passi descritti precedentemente, se il pixel in questione non cambierà colore per molto tempo, il peso della sua gaussiana aumenterà fino a superare quello di una delle altre K-1, entrando così a far parte di uno dei modelli di background.

2.3 Utilizzo nel progetto

Nell'ambito di questo progetto la background subtraction ha avuto un ruolo fondamentale in quanto le features utilizzate per la classificazione sono state ricavate interamente dalla silhouette della persona, intesa come maschera dei pixel appartenenti alla figura della persona rispetto al background.
Per l'implementazione è stato utilizzato il background subtractor contenuto nelle OpenCV, il quale basa la propria implementazione su una versione [3] che introduce alcuni miglioramenti rispetto alla versione descritta nella sezione precedente. Esso infatti introduce un metodo di shadow detection per l'eliminazione delle ombre, molto importante per gli scopi di questo progetto, e migliora la fase di update apportando modifiche alle equazioni che gestiscono l'aggiornamento dei pesi.
Le OpenCV mettono a disposizione anche un'altra variante di questa tecnica basata sul procedimento descritto in [6], ma dopo alcuni test è stata preferita la prima versione in quanto forniva in output risultati più consoni allo scopo da raggiungere.

Nell'implementazione realizzata in questo progetto la background subtraction è stata il primo step effettuato per l'analisi di un frame, ed è stata suddivisa in due fasi:

1. Background subtraction vera e propria.
2. Applicazione di operatori morfologici.

2.3.1 BG subtraction

L'applicazione della tecnica MOG, come già accennato, è stata realizzata tramite le funzioni built-in presenti nelle OpenCV. L'oggetto che realizza

questa funzione viene creato al momento dell'inizializzazione della classe che si occupa dell'analisi del video.
Va notato che prima dell'inizio dell'analisi vera e propria viene passato al background subtractor un frame di background preso da quelli forniti insieme ai video del dataset, in modo da evitare che la persona, già presente sin dal primo frame in ogni video, venga considerata parte del background per qualche tempo. Dopo questa inizializzazione la background subtraction procede normalmente e produce per ogni frame una maschera in cui il nero rappresenta il background e il bianco il foreground.

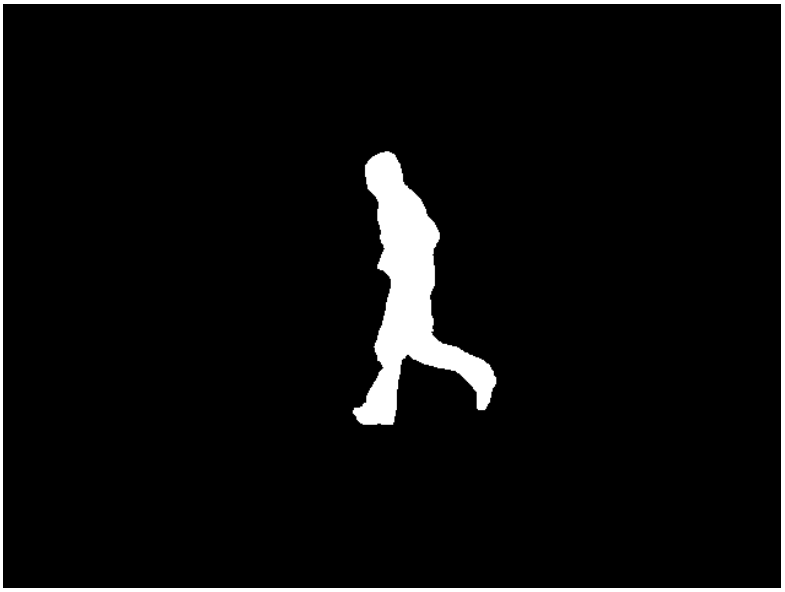
2.3.2 Morfologia

Al fine di migliorare la silhoutte della persona prodotta dalla background subtraction sono stati utilizzati degli operatori morfologici, in particolare è stato effettuato un passaggio di tipo *closing* sulla maschera di foreground.
L'operatore di closing racchiude in realtà altri due operatori morfologici di base, *dilation* ed *erosion*. La prima permette di allargare i contorni delle aree di foreground, chiudendo i gap tra zone collegate e, mentre la seconda permette, una volta applicata la dilation, di diminuire l'area effettiva della maschera, riportandola alle dimensioni iniziali.
In questa implementazione l'operazione di closing è stata realizzata attraverso le funzioni disponibili nelle OpenCV utilizzando come kernel una matrice 7x7 come quella sottostante:

$$E_{ij} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

In Figura 2.1 si può notare il risultato finale di questa fase dell'analisi di un frame.

(a) Il frame originale



(b) La silhoutte ricavata

Figura 2.1: Il procedimento di background subtraction

Capitolo 3

People detection

3.1 Introduzione

Il riconoscimento di persone da parte di un sistema di *computer vision* è un compito tutt'altro che semplice, a causa dell'aspetto mutevole e del gran numero di pose diverse che un essere umano può assumere. Inoltre spesso numerose occlusioni (dovute all'ambiente o ad accessori indossati dalla persona stessa) complicano ulteriormente il processo.

Allo stesso tempo, il problema della *people detection* non può non essere affrontato, in quanto:

1. Il problema di *detection* e del *tracking* di esseri umani è imprescindibile per le applicazioni che hanno l'obiettivo di raggiungere un'interfaccia uomo-macchina il più naturale possibile

2. La cosiddetta *pedestrian detection*ha enormi applicazioni in settori come la videosorveglianza, la robotica, *l'assisted living*, nonchè ovviamente il settore dell'*automotive*.

3. Riconoscere le persone presenti in un immagine è il primo e necessario step di analisi più complesse, come l'analisi del comportamento o delle dinamiche di gruppi di persone.

3.2 Letteratura

Molto lavoro è stato fatto negli ultimi anni nella direzione, non solo della people detection, ma più in generale della cosiddetta *object detection*. Uno dei lavori certamente più famosi è quello di Dalal e Triggs [1]. Esso descrive

un algoritmo per la pedestrian detection che utilizza come descrittori un set di Histogram Of Gradients (HOGs) e un SVM (Support Vector Machine) come classificatore.
Gli HOG vengono calcolati su una finestra di 64x128, la quale viene poi successivamente suddivisa in blocchi 16x16 sovrapposti, sui quali vengono calcolati gli istogrammi dei gradienti lungo nove direzioni. Successivamente tutti gli istogrammi calcolati per una data finestra vengono concatenati, formando così un unico grande istogramma che servirà da feature vector per il classificatore, il quale potrà stabilire se nella finestra 64x128 originale è presente o meno una persona. Il classificatore già allenato per la pedestrian detection, dato un feature vector in input, dovrà solamente calcolarne la posizione nello spazio degli HOG sui quali è stato trainato.

3.2.1 Calcolo degli HOG

Il calcolo degli HOG descritto nel lavoro di Dalal e Triggs segue quattro passi principali:

1. **Calcolo dei gradienti**: sulla già citata finestra 64x128 vengono calcolati i gradienti attraverso il classico procedimento di Sobel, con filtri [-1 0 1] e senza smoothing, ricavando così *magnitude* e *orientation* del gradiente per ogni pixel della finestra.

2. **Calcolo degli istogrammi**: vengono calcolati gli istogrammi su celle della finestra. Ogni istogramma misura la *forza* del gradiente lungo le varie direzioni per una certa cella. Il numero di bin degli istogrammi può variare così come la dimensione della cella, nel caso di Dalal e Triggs i risultati migliori sono stati ottenuti utilizzando istogrammi a 9 bins e celle di 6x6 pixels.

3. **Normalizzazione degli istogrammi**: gli istogrammi calcolati vengono normalizzati utilizzando blocchi sovrapposti delle celle descritte al passo precedente, anche in questo caso la dimensione dei blocchi può variare e i risultati migliori sono stati ottenuti con blocchi di 3x3 celle.

4. **Concatenazione**: tutti gli istogrammi calcolati per la finestra vengono concatenati per formare il feature vector finale da utilizzare per la classificazione.

Il feature vector così calcolato, anche se variabile in base ai parametri utilizzati, assumerà una dimensionalità molto elevata, nell'ordine delle migliaia di dimensioni.

3.2.2 Classificazione

La classificazione ci si affida a un SVM classico, il training viene effettuato su due dataset, il primo è l'INRIA Person Data Set (disponibile a [2]), mentre il secondo è il MIT Pedestrian Database (disponibile a [4]).
Le immagini di training sono state ritagliate in finestre 64x128 attorno alle figure di persone (e non-persone) in modo da allenare correttamente il classificatore, per un totale di 2478 immagini positive e 1218 immagini negative. Ritagliare le immagini è necessario in quanto gli HOG sono descrittori globali e non permettono la localizzazione.
Dato questo particolare la detection deve per forza avvenire scorrendo l'immagine con una finestra di dimension fissa (sempre 64x128) in più scale e posizioni e controllando l'output del classificatore ogni volta. La localizzazione è possibile in quanto si conoscono i dettagli della finestra quando il classificatore trova un match.

3.3 Utilizzo nel progetto

La people detection viene utilizzata nel progetto per *attivare* le funzioni di classificazione.
Per realizzare questa funzione è stato utilizzato il people detector presente nelle OpenCV, il quale non fa altro che implementare il lavoro di Dalal e Triggs, fornendo anche un classificatore SVM di default, già allenato per la people detection.
In un primo momento è stata applicata la people detection a tutto il frame, senza particolari riguardi. Questa fase si è però rivelata particolarmente pesante nell'analisi di un frame causando un significativo rallentamento nella riproduzione del video. Conoscendo bene le caratteristiche del dataset da utilizzare (il Weizmann dataset) si è quindi pensato di *aiutare* il people detector riducendo il suo lavoro e fornendogli in input non tutto il frame, bensì solo una parte di esso. Questo è stato possibile in quanto tutti i video del dataset non contengono mai più di una persona.

3.3.1 Riduzione dell'area di lavoro

Il modo più immediato per ridurre il carico di lavoro del people detector è quello di ridurre l'area che esso deve analizzare. La riduzione della dimensione del frame è resa possibile dal lavoro effettuato dalla background subtraction descritta nel capitolo 2, dovendo cercare persone in movimento è ovviamente inutile cercarle nell'area riconosciuta come background e ci si può limitare all'analisi del foreground.

Dovendo cercare persone in piedi si è deciso di ritagliare una sezione verticale del frame originale di larghezza massima pari a 250 pixel seguendo il procedimento seguente:

1. Attraverso la funzione `findContours()` vengono individuati tutte le aree facenti parte del foreground.

2. Attorno ad ognuna di esse viene creata la bounding box che le contiene usando la funzione `boundingRect()`

3. Di tutte le bounding box create viene individuato il centro.

4. Le coordinate del cosiddetto *centroide* di tutte le aree di foreground facendo una media pesata delle coordinate dei centri delle singole bounding box. il peso di ogni centro è dato dall'area della relativa bounding box, secondo la formula seguente:

$$x_c = \frac{\sum_{i=1}^{N} x_i * A_i}{\sum_{i=1}^{N} A_i} \qquad y_c = \frac{\sum_{i=1}^{N} y_i * A_i}{\sum_{i=1}^{N} A_i}$$

   Dove (x_c, y_c) sono le coordinate del centroide, N è il numero di bounding box trovate e A_i e (x_i, y_i) sono rispettivamente l'area e le coordinate del centro della i-esima bounding box.

5. Attorno alle coordinate del centroide viene ritagliata la sezione verticale del frame che verrà passata in input al people detector. Essendo una sezione verticale essa avrà sempre la stessa altezza (l'altezza del frame, ossia 480 pixel) e larghezza minore o uguale a 250 pixel variabile in funzione delle parti che escono dai bordi dell'immagine.

Riducendo l'area di lavoro del people detector è stato possibile ridurre il carico computazionale di tutto il processo aumentando così il frame rate in output.

Capitolo 4

Classificazione

4.1 Introduzione

Il progetto ha la finalità di classificare le azioni compiute da una persona davanti a un normale dispositivo di acquisizione video.
In particolare si assume che ci sia una singola persona nella scena e che questa sia inquadrata, in maniera completa, dalla testa ai piedi. La classificazione viene fatta su un video che contiene una serie di azioni del soggetto: a fianco quindi della tematica della classificazione delle azioni, si pone il problema dell'individuazione delle singole azioni all'interno di un video. Lo strumento utilizzato per la classificazione è il modello di Markov nascosto (Hidden Markov Model).

4.2 Hidden Markov Models (HMM)

Per la comprensione degli HMM è innanzitutto necessario introdurre il concetto di **modello di Markov**.
Un modello di Markov è un modello stocastico che può essere utilizzato per descrivere l'evoluzione *random* di un sistema dinamico, in cui si assume che uno stato futuro del sistema dipenda solamente da un numero finito di elementi precedenti e non dalla intera sequenza di eventi che l'hanno preceduto. Tale numero è detto *ordine* e determina quindi il grado di approssimazione operato dal modello nel descrivere l'universo delle sequenze. Un modello del primo ordine rispetta la *Proprietà di Markov* o, equivalentemente, ha una condizione di *assenza di memoria*, ovvero:

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, ...] = P[q_t = S_j | q_{t-1} = S_i]$$

in cui q_t è lo stato corrente all'istante di tempo t, mentre $S_1, S_2, ..., S_N$ sono gli stati del sistema. Tale modello è definibile come catena di Markov del primo ordine.

A seconda della tipologia del sistema trattato, infatti, vi sono particolari modelli di Markov che vengono utilizzati: nel momento in cui si parla di un sistema *autonomo*, ovvero di un sistema che non dipende esplicitamente dalla variabile indipendente, come ad esempio il tempo, completamente osservabile nei suoi stati e tempo discreto, si parla di **Catena di Markov**.
Formalmente, mantenendo la notazione sopra adottata, una catena di Markov è una tripla (S, π, A) in cui:

- $S = \{S_1, S_2, ..., S_N\}$ è un insieme finito di N stati
- π rappresenta l'insieme delle probabilità iniziali
- A è l'insieme delle probabilità di transizione; per ogni elemento $a_{ij} \in A$, $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$, con $1 \leq i \leq N, 1 \leq j \leq N$

A è rappresentata da una matrice, detta *matrice di transizione*; valgono le seguenti proprietà:

- $a_{ij} \geq 0$
- $\sum_{j=1}^{N} a_{ij} = 1$

In una catena di Markov vi è corrispondenza biunivoca tra i simboli emessi dall'automa e gli stati corrispondenti: ad ogni stato corrisponde un evento osservabile.
Tuttavia questo modello è troppo restrittivo per poter essere applicato in larga scala a problemi reali: è infatti assai facile avere caso in cui all'osservatore è accessibile solamente una sequenza di simboli in base alla quale egli può inferire soltanto la probabilità degli stati corrispondenti non osservabili.

Si introduce quindi un'altra tipologia di modello di Markov, ovvero il modello di Markov nascosto (**Hidden Markov Model - HMM**), utilizzato con sistemi autonomi e parzialmente o completamente non osservabili.
Un HMM può essere quindi visto come una catena di Markov dotata di un certo numero di stati, in grado di generare un evento secondo una certa distribuzione di probabilità che dipende solo dallo stato stesso, in cui è solamente osservabile l'evento mentre lo stato rimane nascosto.
Per queste sue caratteristiche risulta essere uno strumento particolarmente utilizzato nel momento in cui si ha la necessità di analizzare *dati sequenziali*,

quindi ad esempio nel campo della *speech recognition*, del riconoscimento online della scrittura a mano e in campi quali la bioinformatica (per esempio per la classificazione di particolari sequenze all'interno del genoma umano).

Formalmente, sempre mantenendo la notazione precedente, un HMM è definito da una quintupla (S, V, π, A, B) dove:

- $S = \{s_1, s_2, ..., s_N\}$ è un insieme finito di stati nascosti
- $V = \{v_1, v_2, ..., v_M\}$ è un insieme finito di simboli osservabili (alfabeto)
- $\pi_i = p(q_1 = s_i), 1 \leq i \leq N$ è la probabilità iniziale degli stati (è rappresentato da un vettore)
- A è un insieme di probabilità di transizione: $a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$, con $1 \leq i, j \leq N$. E' rappresentata da una matrice.
- B rappresenta le probabilità di emissione di un simbolo da parte di uno stato tale che: $b_j(k) = p(v_k \text{ at t} | q_t = s_j)$, con $1 \leq j \leq N$, $1 \leq k \leq M$. E' rappresentata da una matrice.

La rappresentazione grafica di un generico HMM viene riportata in 4.3; in particolare si indicano con x_i gli stati (nascosti), con y_i le uscite osservabili, con a_{ij} la probabilità di transizione e b_i la probabilità di uscita.

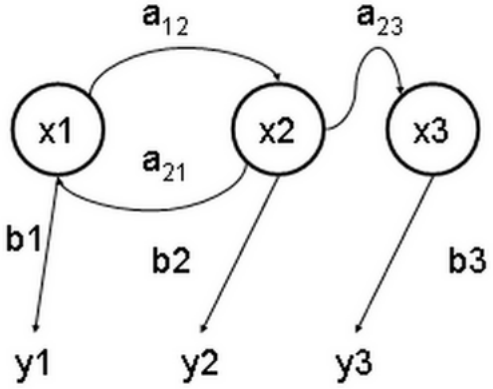


Figura 4.1: Esempio grafico di HMM

L'utilizzo pratico degli HMM ruota attorno alla risoluzione di tre problemi canonici, ovvero:

- Dati i parametri del modello, calcolare la probabilità di una sequenza particolare dell'uscita. E' un problema di **valutazione**.

- Dati i parametri del modello, trovare la sequenza più probabile che potrebbe generare una data sequenza dell'uscita. E' quindi un problema legato al tema della **decodifica**.
- Data una sequenza dell'uscita, o un insieme di tali sequenze, trovare l'insieme più probabile per il quale si possano dichiarare le probabilità dell'uscita e di transizione. E' un problema di **addestramento**, ovvero si cerca di addestrare i parametri dell'HMM dato mediante il gruppo dei dati relativi alle sequenze.

Per ogni problema esistono uno o più algoritmi che ne permettono la soluzione e che quindi offrono uno strumento pratico per addestrare e valutare gli HMM addestrati.

4.2.1 Problema della valutazione

Formalmente: considerata una sequenza di stati nascosti fissata $Q = q_1, q_2, ..., q_T$, data una sequenza di osservazioni $O = o_1 o_2 ... o_T$ e un modello $\lambda = (A, B, \pi)$, si richiede si calcolare $P(O|\lambda)$. In altre parole, si richiede di calcolare la probabilità che una determinata sequenza sia prodotta da un determinato modello.
Si può riscrivere il problema in questo modo:

$$P(O|Q, \lambda) = \prod_{t=1}^{T} p(o_t|q_t, \lambda) = b_{q_1}(o_1) \cdot b_{q_2}(o_2) \cdot ... \cdot b_{q_T}(o_T)$$

La probabilità della sequenza degli stati nascosti è data da:

$$P(Q|\lambda) = \pi_{q_1} \cdot a_{q_1 q_2} \cdot a_{q_2 q_3} \cdot ... \cdot a_{q_{T-1} q_T}$$

Il problema originale può essere visto quindi in questi termini:

$$p(O|\lambda) = \sum_{q_1,...,q_T} p(O|Q, \lambda) p(Q|\lambda) = \pi_{q_1} b_{q_1}(o_1) \cdot a_{q_1 q_2} b_{q_2}(o_2) \cdot a_{q_2 q_3} ... \cdot a_{q_{T-1} q_T} b_{q_T}(o_T)$$

Si tratta però di una soluzione computazionalmente non attuabile, in quanto ha una complessità pari a $2TN^T$ calcoli.
Tale problema viene quindi risolto tramite l'**algoritmo di forward**.
Si introduce innanzitutto una variabile di forward, ovvero la probabilità di una sequenza osservata *parziale* $o_1 o_2 ... o_t$ e dello stato s_i, dato il modello λ

$$\alpha_t(i) = P(o_1 o_2 ... o_t, q_t = s_i|\lambda)$$

L'algoritmo poi risulta essere suddiviso in tre step:

1. **Inizializzazione**

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N$$

poichè nel caso di $t = 0$ non si hanno transizioni da altri stati, non compare il termine della probabilità di transizione, ma compare il valore di probabilità iniziale dello stato.

2. **Induzione**

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N$$

Lo stato s_j può essere raggiunto al tempo $t+1$ dagli N possibili stati al tempo t. $\alpha_t(i)$ rappresenta la probabilità di aver osservato la sequenza $o_1 o_2 ... o_{t-1}$ e di trovarsi a tempo $t-1$ nello stato nascosto s. Quindi il prodotto $\alpha_t(i) a_{ij}$ rappresenta la probabilità di osservare $o_1 o_2 ... o_{t-1}$ e di raggiungere lo stato s_j al tempo t proveniendo dallo stato s_i. Il tutto viene moltiplicato per la relativa probabilità di emissione.
Sostanzialmente la probabilità di trovarsi in un certo stato, avendo oservato una determinata sequenza, moltiplicata per la probabilità di emissione dello stato stesso permettono di ottenere un valore che permette di comprendere quanto una certa sequenza parziale sia aderente al modello preso in considerazione.

3. **Terminazione**

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_T(j)$$

Nello step finale si accumulano i valori di tutte le variabili di forward per tutti i possibili stati. Si ottiene così il valore di likelihood, ovvero di verosomiglianza al modello, per l'intera sequenza osservata (e non solo per quella parziale).

Il problema può essere risolto tramite un altro algoritmo, speculare a quello appena visto, chiamato **algoritmo di backward**. Come detto è analogo al caso forward, l'unica differenza è che, come si intuisce dal nome stesso, in questo caso si procede a ritroso e che naturalmente bisogna introdurre una nuova *variabile di backward*:

$$\beta_t(i) = p(o_{t+1} o_{t+2} ... o_T | q_t = s_i \lambda)$$

In maniera del tutto analoga si possono costruire i tre step di inizializzazione, induzione e terminazione.
Tale algoritmo viene qui introdotto poichè necessario alla risoluzione del problema di addestramento.

4.2.2 Problema della decodifica

Formalmente: data una sequenza di osservazioni $O = o_1 o_2 ... o_T$ e un modello λ, bisogna determinare la sequenza di stati $Q = q_1 q_2 ... q_T$ più probabile che potrebbe generare la sequenza data. L'algoritmo che risolve questo problema è l'**algoritmo di Viterbi**.
L'idea di base è presa dall'algoritmo di forward: l'idea è quella di determinare il cammino ottimale andando a calcolare la probabilità di transizione massima invece che sommarla alle altre trovate, e mantenendo un indice degli stati nascosti migliori ad ogni passo, ovvero:

$$\delta_{t+1}(i) = \left[\max_{1 \leq i \leq N} \delta_t(i) a_{ij} \right] b_j(o_{t+1})$$

Tale algoritmo non viene analizzato ulteriormente poichè non è utilizzato all'interno del progetto.

4.2.3 Problema dell'addestramento

Il problema è aggiustare i parametri del modello $\lambda = (A, B, \pi)$ per massimizzare $P(O|\lambda)$, ovvero la probabilità di avere una determinata osservazione. E' sicuramente il problema più complesso.
La procedura iterativa di **Baum-Welch** (o equivalentemente il motodo della *Expectation Maximization*, EM) permette di massimizzare localmente la probabilità dell'osservazione.

Definiamo la probabilità di essere nello stato s_i al tempo t e nello stato s_j al tempo $t + 1$:

$$\xi_t(i, j) = p(q_t = s_i, q_{t+1} = s_j | O, \lambda)$$

Questa probabilità può essere riscritta andando ad esplicitare le tre componenti al suo interno, ovvero la storia *passata* del sistema (la probabilità di essere in s_i all'istante di tempo t, date le osservazioni $o_1 o_2 ... o_t$), riassunta dalla variabile di forward $\alpha_t(i)$, la probabilità di passare da s_i a s_j, data dal prodotto della probabilità di transizione a_{ij} con la probabilità di emissione $b_j(o_{t+1})$ e la storia *futura* (ovvero la probabilità di avere le restanti osservazioni $o_{t+1} ... o_T$ partendo dallo stato s_j), riassunta dalla variabile di backward. Formalmente:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}$$

Inoltre esprimiamo la probabilità di trovarsi nello stato s_i al tempo t data l'osservazione O; tale espressione può esprimersi in termini di variabili forward-backward:

$$\gamma_t(i) = p(q_t = s_i|O, \lambda)$$

Possiamo quindi correlare $\gamma_t(i)$ con $\xi_t(i, j)$ con una semplice sommatoria:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

Possiamo quindi esprimere le formule di Baum-Welch per ottenere iterativamente una stima migliore dei parametri del modello $\lambda' = (A', B', \pi')$ partendo da una precedente stima $\lambda = (A, B, \pi)$:

1. $\pi'_i = \gamma_t(i), 1 \leq i \leq N$, ovvero la probabilità di essere in s_i all'istante $t = 1$
2. $a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$ ovvero il rapporto fra il valore atteso di transizioni dallo stato i allo stato j e il valore atteso di volte dello stato j.
3. $b'_{jk} = \frac{\sum_{t=1, Y_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(i)}$ ovvero il rapporto tra il valore atteso di volte nello stato j e simbolo osservato k e il valore atteso di volte dello stato j.

Se si utilizza iterativamente λ' al posto di λ e si esegue una nuova stima, si può migliorare la probabilità delle osservazioni a partire dal modello fino al raggiungimento di un certo punto limite.

4.3 Weizmann Dataset

Il dataset utilizzato per il progetto è il **Weizmann dataset**. Questo contiene 90 video di 10 tipi principali di azioni appartenenti a 9 differenti soggetti. Le tipologie di azioni possono essere considerate 16 in quanto alcune azioni vengono compiute in maniera differente dalle altre.
I video originali sono a bassa risoluzione (180x144), nel progetto, per esigenze di programmazione, si lavora alla risuluzione standard di 640x480.
Le classi di azioni presenti sono: bend, jack, jump, pjump, run, side, skip, walk, wave1 (con un solo braccio) e wave2 (con due braccia).

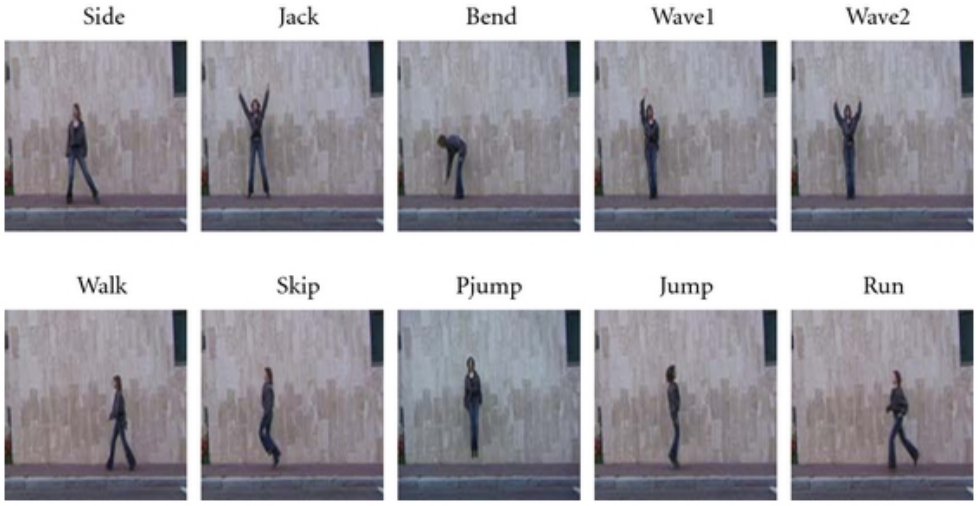


Figura 4.2: Esempio di frame appartenenti ad azioni differenti presenti nel dataset

4.4 Features set

La scelta delle features risulta essere molto importante ai fini di una corretta classificazione. Si cerca nello specifico di individuare delle features in grado di descrivere le peculiarità di un'azione, senza perdere efficacia nella generalizzazione.
Si è scelto di ricavare direttamente dalla maschera binaria di foreground (ottenuta con azioni di background subtration e operazioni morfologiche) una bounding box che contiene la sagoma del soggetto ripreso; in seguito si sono costruiti due istogrammi delle proiezioni lungo l'asse x e lungo l'asse y, ognuno di 10 bins; successivamente questi vengono concatenati. Si ottiene quindi, per ogni frame, un features set composto da 20 valori, normalizzati tutti tra 0 e 1.
Costruendo in tale modo le features, appare evidente che nè la posizione, nè il moto globale del soggetto vengono presi in considerazione.
Formalmente gli istogrammi sono stati costruiti in questo modo: data la maschera binaria di foreground $F(x, y)$ (0: pixel di background, 1: pixel di foreground), l'istogramma dell'asse x è definito come:

$$\sum_{y=0}^{F_y} \phi(F(x,y)), \phi = \begin{cases} 1, & F(x,y) = 1 \\ 0, & F(x,y) = 0 \end{cases}$$

mentre l'istogramma dell'asse y è definito come:

$$\sum_{x=0}^{F_x} \phi(F(x,y)), \phi = \begin{cases} 1, & F(x,y) = 1 \\ 0, & F(x,y) = 0 \end{cases}$$

dove F_x ed F_y sono, rispettivamente, la larghezza e l'altezza dell'immagine.

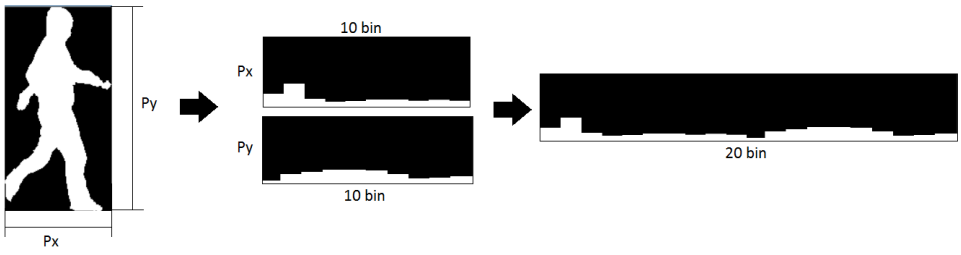


Figura 4.3: Grafico che illustra il calcolo delle fatures

4.5 Utilizzo nel progetto

Nel progetto per l'implementazione degli HMM ci si è serviti di codice in C++ già esistente, messo a disposizione dal prof. Vezzani, per cui gli sforzi si sono concentrati nel creare una struttura per il corretto *training* e *testing* degli stessi, nelle modaità esposte nei prossimi paragrafi.
Il codice degli HMM fornito modella la probabilità di emissione di uno stato $b_j(o)$ tramite un Gaussian Mixture Model; in questo caso quindi il termine $b_j(o_t)$, visto nel paragrafo riguardante il problema di valutazione (paragrafo 4.2.1), può essere approssimato tramite una sommatoria di gaussiane:

$$b_j(o_t) = \sum_{l=1} c_{jl} \mathcal{N}(o_t | \mu_{jl}, \Sigma_{jl})$$

Gli altri parametri degli HMM sono stati settati empiricamente, cercando di massimizzare il *recognition rate* durante la fase di testing: il numero degli stati nascosti utilizzati all'interno dell'HMM è 8, 1 gaussiana per la probabilità di emissione dei simboli.

4.5.1 Training

La tecnica di training utilizzata è stata quella del **Leave-one-out** (LOO) che consiste nell'addestrare un HMM per *ogni* azione (video) presente nel dataset. Quindi per ogni frame di ciascuna azione sono state calcolate le rispettive features; per ogni insieme di frame, e quindi un insieme di features,

appartenenti allo stesso video è stato addestrato un HMM.
Durante la fase di testing verrà poi testata l'intera batteria di HMM, mantenendo memorizzato solamente il valore di likelihood maggiore (approcci alternativi potrebbero essere quelli del calcolo della *moda* sui k risultati migliori), escludendo il classificatore addestrato sull'azione stessa che si sta tentando di classificare.

4.5.2 Testing su una singola azione

La classificazione di una singola azione contenuta in un video rappresenta il caso più semplice, in quanto vengono caricati tutti i frame appartenenti al video, vengono calcolate le features e viene restituita la classificazione in output; come descritto nel paragrafo precedente la classificazione si ottiene con:

$$\arg\max_i L = P(O|\lambda_i)$$

Non sono necessari particolari meccanismo di *action detection* quindi, poichè appunto si parte dall'ipotesi che il video contenga solamente un'azione.

4.5.3 Testing su sequenze di azioni

La classificazione su sequenze di azioni differenti contenute in un unico video è sicuramente più complessa; questo è dovuto principalmente al fatto che risulta ora essere necessario individuare l'azione all'interno del video, per poter fornire al classificatore un insieme di features correlate dal punto di vista logico.
Per fare questo è stato implementato un meccanismo a finestre temporali statiche: ad ogni intervallo di frame viene creata una nuova finestra, di dimensione fissa; in maniera ciclica viene controllato il numero delle finestre e in caso di sforamento del numero massimo viene eliminata la meno recente (una sorta di meccanismo FIFO).
I parametri relativi alle finestre (dimensione, intervallo di creazione e numero massimo) sono stati settati in maniera empirica, analizzando le azioni presenti nel dataset, in modo tale da massimizzare le performance del sistema.
In tal modo si cerca di spezzare il video in molteplici finestre di dimensione identica, con la consapevolezza che almeno in alcune di queste sia contenuta l'intera, o almeno una parte, di un'azione.

Il valore restituito in uscita dagli HMM risulta essere direttamete confrontabile, grazie proprio alla grandezza fissa di finestre (e quindi un uguale

numero di features date in input ai classificatori). La batteria di HMM viene testata nel momento in cui viene una finestra raggiunge la dimensione massima. Come detto, è possibile che una finestra contenga frame appartenenti ad azioni differenti: per questo motivo la classificazione potrebbe risultare influenzata in maniera negativa e dare in output risultati errati.
Per tentare di ovviare a questo problema si è imposta una soglia di sicurezza che la classificazione finale deve superare per essere considerata valida; il livello di sicurezza è calcolato tramite il rapporto tra il valore massimo di likelihood ottenuto, e il valore massimo di likelihood ottenuto di un classificatore addestrato su una tipologia di azione diversa da quella del primo.

Il settaggio del valore di soglia è fortemente legato al campo di applicazione del progetto, in quanto regola il *numero* di classificazioni e l'*accuratezza* della recognition in generale: valori bassi di soglia determinano un aumento dei tentativi di classificazione e un aumento del numero di errori, valori alti di soglia viceversa.

4.5.4 Calcolo delle prestazioni

Nel momento in cui, al termine di una finestra, si ha la classificazione, sorge la necessità di controllare che questa sia corretta.
E' stato quindi implementato un meccanismo di questo tipo: viene definito inizialmente un *ground throuth*, ovvero un vettore contenente le etichette delle azioni per ogni frame di ogni azione; viene confrontata l'etichetta dell'azione classificata con il valore della label del pixel centrale della finestra corrente, basandosi sull'ipotesi che una finestra al più contiene 2 tipi di azioni e che quindi il frame centrale sia altamente significativo per stabilire quale azione è individuata dalla finestra.

Capitolo 5

Conclusioni

5.1 Interfaccia grafica

L'interfaccia grafica per questo progetto è stata realizzata attraverso il framework *Qt*, esso fornisce tutti i componenti grafici di cui si è avuta necessità per rappresentare in output le varie fasi di questo progetto.

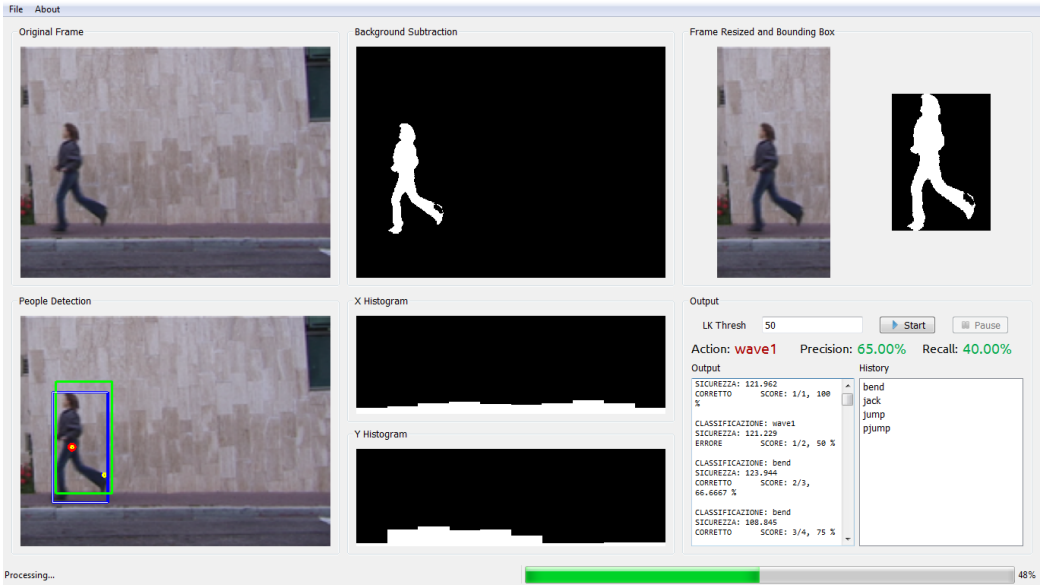


Figura 5.1: Interfaccia grafica del progetto

L'interfaccia, rappresentata in Figura 5.1, contiene il frame originale, la maschera di foreground ricavata dalla background subtraction, la bounding box ricavata attorno all'area in movimento, il risultato della people detection

e gli istogrammi delle feature ricavate.
Essa contiene anche alcuni strumenti per rappresentare in output i risultati della classificazione e una console contenente l'output della classificazione.

5.2 Risultati ottenuti

5.2.1 Video con singole azioni

Inizialmente il progetto è stato testato sui video di singole azioni, quelli già presenti nel dataset. Per questi video non è stato applicato il metodo a finestra scorrevole utilizzato per i video costituiti da più azioni, in quanto non si pone il problema della mancanza di riferimenti relativi all'inizio di un'azione. La detection viene infatti eseguita sulle feature calcolate su tutti i frame del video.

Azione	**% Riconoscimento**
Bend	89 %
Jack	100 %
Jump	89 %
Pjump	78 %
Run	40 %
Side	75 %
Skip	10 %
Walk	80 %
Wave1	89 %
Wave2	89 %

Tabella 5.1: Risultati ottenuti su singole azioni

In tabella 5.1 vengono riportate le percentuali di riconoscimenti corretti per ogni tipologia di azione, e in Figura 5.2 viene riportata la relativa matrice di confusione.

Si può notare che generalmente si ottengono risultati discreti; per alcune tipologie di azione, ad esempio *skip* e *run*, la percentuale di riconoscimento risulta essere particolarmente bassa: le due azioni vengono spesso confuse tra loro a causa di feature poco discriminanti poichè calcolate su silhouette che risultano essere praticamente identiche, come si può notare nelle figure 5.3, 5.4 e 5.5.

	Bend	Jack	Jump	Pjump	Run	Side	Skip	Walk	Wave1	Wave2	TOT
Bend	**8**	0	0	0	0	0	0	0	1	0	**9**
Jack	0	**9**	0	0	0	0	0	0	0	0	**9**
Jump	0	0	**8**	0	0	1	0	0	0	0	**9**
Pjump	0	0	0	**7**	0	0	0	0	2	0	**9**
Run	0	0	0	0	**4**	0	4	2	0	0	**10**
Side	0	0	1	0	0	**7**	1	0	0	0	**9**
Skip	0	0	1	0	4	0	**1**	4	0	0	**10**
Walk	0	0	0	0	2	0	0	**8**	0	0	**10**
Wave1	0	0	0	1	0	0	0	0	**8**	0	**9**
Wave2	0	1	0	0	0	0	0	0	0	**8**	**9**
TOT	**8**	**10**	**10**	**8**	**10**	**8**	**6**	**14**	**11**	**8**	**93**

Figura 5.2: Matrice di confusione ricavata dall'analisi di singole azioni

Figura 5.3: Mask di *run*

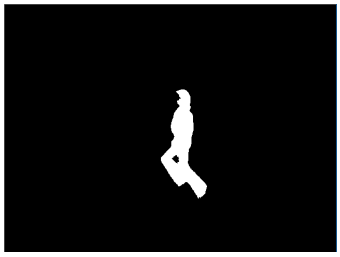


Figura 5.4: Mask di *skip*

Figura 5.5: Mask di *walk*

5.2.2 Video con sequenza di azioni

Si è poi svolta una serie di test su video contenenti più azioni ottenuti unendo alcuni dei video singoli in video più grandi che accostano più azioni. In questo modo è stato possibile realizzare l'approccio a finestra descritto precedentemente.
Un primo indicatore importante della bontà dei risultati è rappresentato dalle matrici di confusione. Esse rappresentano la robustezza della classificazione, permettono di capire come avviene la classificazione e soprattutto permettono di capire meglio in che casi si verificano degli errori e se le feature che si stanno utilizzando sono robuste o meno.
Di seguito vengono riportate due matrici di confusione, la prima in Figura 5.6, è stata realizzata mantenendo la soglia di sicurezza pari a 0, mentre la seconda, in Figura 5.7, è stata realizzata con soglia pari a 160. I valori di soglia sono stati determinati in modo empirico in base ai risultati ottenuti e sono strettamente correlati ai video presi in esame.

Dalle due matrici, si può notare che utilizzando una soglia pari a 0, vengano classificate in maniera robusta solamente le azioni le cui feature sono altamente discriminative, come ad esempio *jack*, *wave1*, *skip* e *bend*. Altre azioni invece vengono spesso confuse con azioni simili per le quali la maschera ricavata risulta avere praticamente la stessa forma. Ad esempio, l'azione

Matrice di confusione calcolata su: all_daria.avi, utilizzando una soglia nulla											
	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2	TOT
bend	**7**	0	0	1	0	0	0	0	0	0	8
jack	0	**11**	0	0	0	0	0	0	1	0	12
jump	0	1	**4**	0	0	0	1	0	3	0	9
pjump	0	0	0	**6**	0	0	0	0	2	0	8
run	0	0	0	0	**1**	0	0	2	3	0	6
side	0	0	0	0	0	**2**	1	4	0	0	7
skip	0	0	0	0	0	0	**6**	1	0	0	7
walk	0	0	2	0	0	0	0	**9**	0	0	11
wave1	0	0	0	1	0	0	0	0	**9**	0	10
wave2	0	0	0	0	0	0	0	0	1	**8**	9
TOT	7	12	6	8	1	2	8	16	19	8	**87**

Figura 5.6: Matrice di confusione ricavata con soglia 0

Matrice di confusione calcolata su: all_daria.avi, utilizzando una soglia 160											
	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2	TOT
bend	**0**	0	0	0	0	0	0	0	0	0	0
jack	0	**10**	0	0	0	0	0	0	0	0	10
jump	0	0	**0**	0	0	0	0	0	0	0	0
pjump	0	0	0	**2**	0	0	0	0	0	0	2
run	0	0	0	0	**0**	0	0	0	2	0	2
side	0	0	0	0	0	**0**	0	0	0	0	0
skip	0	0	0	0	0	0	**0**	0	0	0	0
walk	0	0	0	0	0	0	0	**0**	0	0	0
wave1	0	0	0	0	0	0	0	0	**1**	0	1
wave2	0	0	0	0	0	0	0	0	0	**0**	0
TOT	0	10	0	2	0	0	0	0	3	0	**15**

Figura 5.7: Matrice di confusione ricavata con soglia 160

side viene spesso confusa con *walk*, dato che risultano estremamente simili quando si tratta di analizzare solamente la silhoutte della persona. Una rappresentazione alternativa della matrice in Figura 5.6 è riportata in figura 5.8. Per ogni tipologia di azione si può osservare la percentuale di detection, la relativa percentuale di detection errata e l'azione con la quale viene confusa.

Nella seconda matrice la soglia è stata alzata considerevolmente, fino a 160. Come previsto il numero totale di classificazioni effettuate è molto minore rispetto al caso precedente: naturalmente nel momento in cui viene alzato il valore della soglia si hanno meno classificazioni ma più sicure e, viceversa, abbassandola si hanno più classificazioni ma meno sicure.
Questo viene misurato in base ai valori di precision e recall al variare della soglia di sicurezza del classificatore intesi in questo modo:

- *Precision*: è il rapporto tra il numero di classificazioni corrette e il numero totale di classificazioni effettuate.
- *Recall*: è il rapporto tra il numero di azioni riconosciute e l'effettivo numero di azioni contenute all'interno del video.

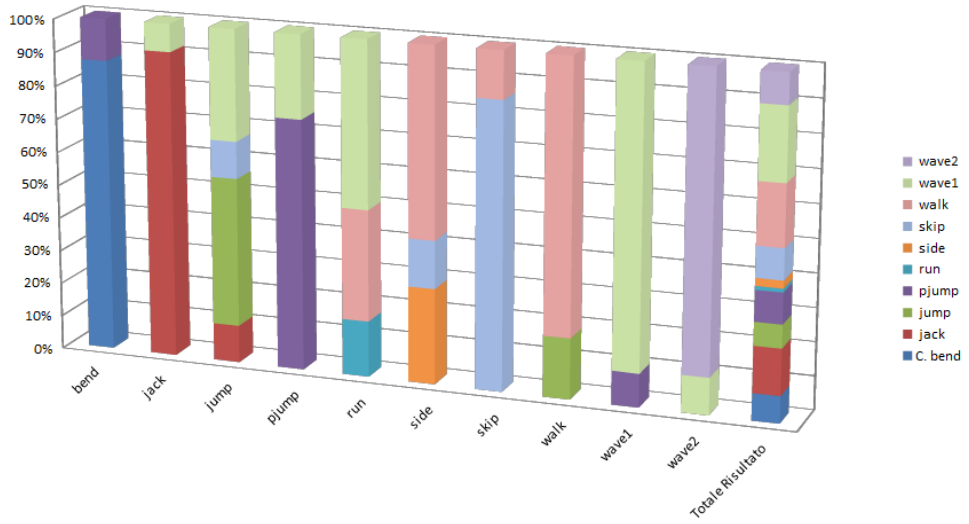


Figura 5.8: Matrice di confusione ricavata con soglia 0 rappresentata come istogramma

La correttezza della classificazione viene determinata in base al meccanismo visto in sezione 4.5.4.
Nel grafico in Figura 5.9 si può notare qual è la correlazione tra precision e recall al variare del valore della soglia.

Sono emerse tre considerazioni principali:

- Parte degli errori sono dovuti alla conformazione delle finestre sottoposte agli HMM, nel caso in cui sono composte da frame appartenenti ad azioni diverse, in numero rilevante.
  Si può ritenere quindi che giustamente i classificatori siano indotti in errore poichè tentano di classificare basandosi su delle features ibride, poco significative.

- Altri errori sono dovuti all'impiego di features poco efficaci derivanti dall'aver inserito nella finestra frame di una parte di azione poco significativa (ovvero in cui le features calcolate sono poco discriminative).
  Si pensi ad esempio al caso del salto su un piede (jack) che risulta essere molto simile, in determinati istanti di tempo, al salto su due piedi, o ancora in cui all'interno della bounding box della silhouette compaiono solo pochi particolari del soggetto poichè questo risulta essere in larga parte fermo (e quindi ritenuto di background dalla background Subtraction, cfr. Capitolo 1), come si può vedere nell'immagine 5.10.

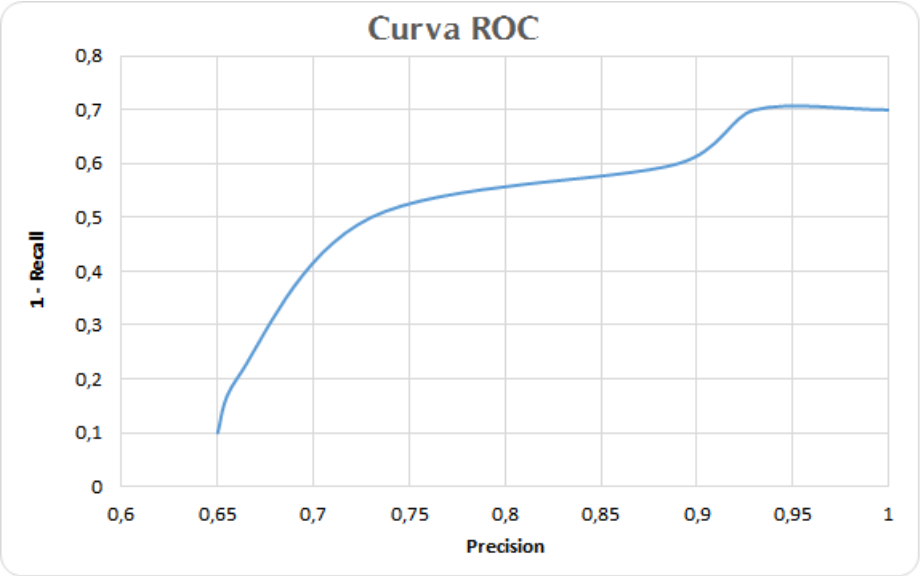


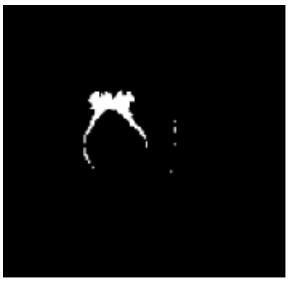
Figura 5.9: Curva ROC rilevata sui dati usati nei test

- Un altro problema è stato rilevato con l'azione *run* in quanto la sua durata, per natura dell'azione stessa, è assai breve e coniugandosi male con l'approccio a finestre temporali non permette ai classificatori di riconoscerla attraverso queste feature.

È stato creato anche un grafico per riassumere il lavoro di classificazione a finestre temporali statiche (Figura 5.11). In rosso viene riportata la reale suddivisione in azioni del video, ossia la ground truth, mentre in blu viene riportata la suddivisione ottenuta dalla classificazione; si può notare come nel settore centrale gli errori di classificazione portino all'individuazione di finestre non corrette.
A dimostrazione della bontà del progetto si può però notare come il trend di classificazione, rappresentato con una linea nera tratteggiata, segua quello della ground truth (le azioni sono state volontariamente inserite nel video con id crescente).

(a) Frame dell'azione *bend*

(b) Frame dell'azione *wave1*



(c) Frame dell'azione *wave2*

Figura 5.10: Esempio di features poco significative

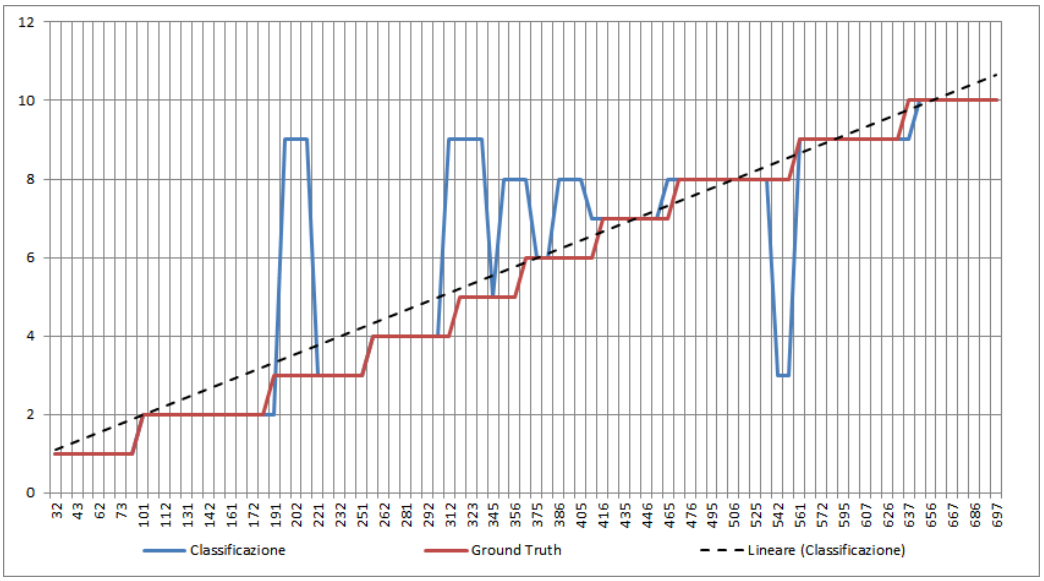


Figura 5.11: Curva ROC rilevata sui dati usati nei test

Bibliografia

[1] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[2] INRIA. Inria person data set, March 2015. http://lear.inrialpes.fr/data.

[3] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection, 2001.

[4] MIT. Mit pedestrian database, March 2015. http://cbcl.mit.edu/software-datasets/PedestrianData.html.

[5] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, August 2000.

[6] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, August 2000.

[7] Roberto Vezzani, Davide Baltieri, and Rita Cucchiara. Hmm based action recognition with projection histogram features. In *Proceedings of the ICPR 2010 Contests*, pages 286–293, Istanbul, Turkey, August 2010.