Università Politecnica delle Marche

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

e dell'Automazione

Implementazione di un algoritmo di identificazione della persona utilizzando frame di profondità

Implementation of a depth-based human identification algorithm

Relatore:
Prof. Ennio Gambi

Correlatori:
Prof.ssa Susanna Spinsante

Ing. Samuele Gasparrini

Tesi di Laurea di:
Ilario Pierbattista

Anno Accademico 2014/2015

Indice

1 Introduzione **3**
- 1.1 Human Sensing . . . 3
  - 1.1.1 Human Sensing . . . 3
  - 1.1.2 Stato dell'arte . . . 4
- 1.2 Panoramica Generale . . . 4
  - 1.2.1 Introduzione al lavoro di Zhu e Wong . . . 4
  - 1.2.2 Configurazione dell'Hardware . . . 4
  - 1.2.3 *Head and Shoulders Profile* . . . 5
  - 1.2.4 Flusso di Lavoro . . . 6

2 Haar-Like Features **8**
- 2.1 Definizione . . . 8
  - 2.1.1 Richiamo: cosa è una feature (caratteristica) . . . 8
  - 2.1.2 Wavelet di Haar . . . 8
  - 2.1.3 Formula di calcolo standard . . . 9
  - 2.1.4 Cosa mette in evidenza la feature di Haar . . . 9
  - 2.1.5 Formula di calcolo invariante ai resize . . . 10
  - 2.1.6 Vantaggi . . . 10
- 2.2 Immagine Integrale . . . 11
  - 2.2.1 Definizione rigorosa dell'immagine integrale . . . 11
  - 2.2.2 Complessità computazionale generale . . . 11
- 2.3 Decision Stump . . . 12
  - 2.3.1 Problema: utilizzare le feature . . . 12
  - 2.3.2 Definizione di albero decisionale . . . 12
  - 2.3.3 Definizione di decision stump . . . 12

3 L'Algoritmo di Allenamento: Adaboost **13**
- 3.1 Apprendimento Supervisionato *Ensamble* . . . 13
  - 3.1.1 Apprendimento Supervisionato . . . 13
  - 3.1.2 *Adaptive Boosting* . . . 14
- 3.2 Dataset di Allenamento . . . 15
  - 3.2.1 Categorie di Classificatori . . . 15
  - 3.2.2 Preparazione dei Dataset . . . 16
  - 3.2.3 Preprocessing . . . 17
- 3.3 *Strong Learner* [Da revisionare] . . . 17
  - 3.3.1 Procedura di estrazione del classificatore forte . . . 17

3.4 *Weak Learner* [Da revisionare] . . . 19
3.4.1 Procedura di estrazione del classificatore debole . . . 19
3.4.2 Valutazione della complessità computazionale . . . 21

4 Validazione e Regolazione dei Classificatori 22
4.1 Criteri di Valutazione . . . 22
4.2 Dataset di Validazione . . . 22
4.2.1 Criteri di creazione delle registrazioni . . . 22
4.2.2 Altre caratteristiche . . . 22
4.3 Massimizzazione all'*Accuracy* . . . 22
4.3.1 Parametri liberi del classificatore . . . 22
4.3.2 Algoritmo di ricerca della soglia e del NWL ottimi . . . 23
4.4 Analisi dei Risultati . . . 23

5 Rilevamento 25
5.1 Tecnica di Rilevamento . . . 25
5.1.1 Detection Window . . . 25
5.1.2 Rilevazione su frame . . . 25
5.2 Selezione della Finestra Migliore . . . 25
5.2.1 Introduzione al problema . . . 25
5.2.2 Algoritmo di selezione . . . 25
5.3 Confronto con l'Algoritmo G-C . . . 25

6 Conclusioni 26

Appendici 27

A Software Sviluppato 28
A.1 Componenti . . . 29
A.1.1 Creatore dei Dataset . . . 29
A.1.2 Allenamento . . . 29
A.1.3 Tuning, Testing, Rilevamento . . . 29
A.2 Tecnologie utilizzate . . . 29
A.2.1 C++ e Matlab . . . 29
A.2.2 Git e Github [Opzionale] . . . 29
A.3 Proposte di miglioramento . . . 29

B Cenni del funzionamento del sensore Kinect 30

Capitolo 1

Introduzione

1.1 Human Sensing

1.1.1 Human Sensing

Definizione

L'insieme di tecniche di riconoscimento della presenza di una persona nello spazio prendono il nome di tecniche di *human sensing*.

Sensori di vario tipo vengono utilizzati nelle tecniche di riconoscimento.

Una volta acquisite le informazioni dai sensori, vanno elaborate da un apposito algoritmo per rilevare la presenza e la posizione della persona nell'ambiente.

Contesti Applicativi

La possibilità di rilevare la presenza di persone, rende le applicazioni di *human sensing* perfette per le applicazioni di sorveglianza.

Sistemi di *people counting* sono utili per la conduzione di indagini di mercato.

Dispositivi in grado di rilevare la presenza di corpi umani in contesti di crisi sono utilizzati nelle attività di *search & rescue*.

Le applicazioni di *human tracking* sono utili anche negli ambienti assistivi automatizzati al fine di monitorare le attività dell'assistito.

HS e Computer Vision

Le applicazioni di human sensing che utilizzano sensori di acquisizione *visiva* risolvono problemi di computer vision.

Lo scopo della *computer vision* è quello di riprodurre la vista umana. L'obbiettivo di tale riproduzione non si limita alla semplice acquisizione di una rappresetazione bidimensionale di una regione di spazio, ma mira all'interpretazione del relativo contenuto.

1.1.2 Stato dell'arte

Pedestrian Detection and Counting

Papageorgiou et Al [6] hanno sviluppato un sistema di riconoscimento e conteggio di pedoni a partire da immagini RGB.

Face Recognition

Viola e Jones [9] hanno proposto un framework per il riconoscimento dei volti all'interno di immagini RGB. Al momento è il sistema più solido nel suo contesto.

Kinect: a serious game

Gli ambiti d'utilizzo del dispositivo Kinect, nota periferica legata a sistemi videoludici, si stanno espandendo constantemente. La quantità e la qualità dei sensori di cui è equipaggiato, il costo relativamente contenuto e l'evoluzione di framework e toolkit di sviluppo, lo rendono un dispositivo particolarmente versatile e adatto allo studio di problemi di computer vision.

1.2 Panoramica Generale

1.2.1 Introduzione al lavoro di Zhu e Wong

Elenco delle tecnologie coinvolte

La sorgente di informazione è il sensore di profondità del Kinect.

Il sistema descritto prevede l'utilizzo di un algoritmo di allenamento per sviluppare i criteri di riconoscimento della persona.

Ciò che viene presentato da Zhu e Wong è un sistema di rilevamento che prende notevolmente in considerazione le soluzioni proposte da Viola e Jones, eccezion fatta, naturalmente, per il dispositivo di acquisizione.

1.2.2 Configurazione dell'Hardware

Sensore utilizzato

Il sensore di profondità del Kinect V2 fornisce una rappresentazione bidimensionale dello spazio sotto forma di immagini. In tali immagini ogni pixel corrisponde il valore in millimetri della distanza dal sensore della superficie dell'oggetto interessato.

Ci si riferirà a tali immagini chiamandole *immagini di profondità*. Il sensore del Kinect, di cui è disponibile una piccola descrizione più dettagliata all'appendice B, fornisce uno stream di tali immagini ad una frequenza di 30 frame al secondo: è possibile quindi registrare dei *video di profondità*.

Configurazione Top-Down

Il dispositivo Kinect viene montato al soffitto di una stanza e l'ambiente viene ripreso da tale prospettiva. Solitamente l'altezza a cui viene montato è di poco inferiore alla distanza del soffitto dal pavimento (poco meno di 2m). La linea focale del sensore

dovrebbe essere quanto più possibile ortogonale al pavimento della stanza, in modo da ridurre ai minimi termini la presenza di asimmetrie nelle riprese. Tali distanze sono perfettamente compatibili con le specifiche tecniche del dispositivo stesso. Nel caso in cui vi sia la necessità di montare il Kinect a soffitti più alti di 4m, si possono utilizzare delle lenti correttive per aumentare il range di affidabilità del sensore.

Molto sistemi di riconoscimento utilizzano il Kinect in posizione frontale ai soggetti da riconoscere. Di fatti, il dispositivo, concepito per applicazioni videoludiche, è progettato per operare in tale posizione. Tuttavia, la configurazione descritta precedentemente, ha l'enorme vantaggio di eliminare l'occlusione del soggetto: normalmente una persona non può nascondere dietro di sè un'altra persona alla vista del sensore (se non in scomode posizioni), cosa frequentessima invece con i sistemi di rilevamento frontali.

1.2.3 *Head and Shoulders Profile*

L'attività di riconoscimento è un'attività di classificazione

Ciò che bisogna riconoscere è, all'interno di un'immagine di profondità, la figura della persona.

Considerando un altro punto di vista, l'attività di riconoscimento consiste nel discriminare gli oggetti che sono figure di persone, da oggetti che non lo sono.

Si definiscono quindi due classi. Il concetto di classe è molto simile alle *classi di equivalenza* dell'algebra astratta e costituiscono degli insiemi di oggetti che condividono determinate proprietà. Distinguere gli oggetti che rappresentano persone da quelli che non le rappresentano, significa classificare tali oggetti in due classi: le persone e le non persone. Il processo di rilevamento, quindi, si basa sulla determinazione della classe di appartenenza dei vari oggetti: *classificazione*.

La classificazione si basa sulla misurazione di alcune caratteristiche

Gli oggetti di una stessa classe hanno alcune proprietà in comune, ma differiscono per altre. Individuare le caratteristiche - ovvero proprietà osservabili e misurabili di un oggetto - in base alle quali discriminarli nelle due classi non è un problema banale. Si può intuire quanto sia vasto l'insieme delle caratteristiche valutabili nella rappresentazione di un oggetto. Ovviamente la natura della rappresentazione influisce nella scelta delle caratteristiche più rilevanti. Nei capitoli successivi verrà presentato un algoritmo che automatizzerà la selezione delle caratteristiche più rilevanti dell'immagine.

Caratteristiche del profilo HASP in linguaggio naturale

Un'immagine di profondità, per sua natura, rappresenta la realtà attraverso il valore della distanza misurata in ogni punto dello spazio osservato. È naturale, quindi, considerare tali distanze come caratteristiche misurabili dell'oggetto rappresentato.

È utile quindi fornire una descrizione, se non altro in linguaggio naturale, della forma del profilo umano ripreso dall'alto, obiettivo del riconoscimento. Tale descrizione è informale.

1. L'immagine di una persona è caratterizzata da uno *spazio vuoto*[1] di fronte ad essa e dietro di essa.

2. A sinistra della spalla sinistra ed a destra della spalla destra del profilo dall'alto di una persona, sono presenti degli spazi vuoti.

3. Tra la testa e le spalle vi è una differenza di altezza.

1.2.4 Flusso di Lavoro

Definizione dei moduli funzionali

Un modulo software sarà dedicato all'allenamento.

Un modulo software sarà dedicato al rilevamento.

Allenamento

Per allenare il sistema è necessario creare un insieme di allenamento, ovvero un insieme i cui elementi sono delle immagini che ritraggono persone e non. In fase di creazione, ogni elemento viene dotato di un'etichetta che identifica la classe di appartenenza reale dell'oggetto.

La componente software che si occupa dell'allenamento del sistema implementa l'algoritmo Adaboost. Quest'ultimo riceve in input l'insieme di allenamento, i cui elementi, dotati della rispettiva classificazione reale, sono alla base della scelta delle caratteristiche migliori per la descrizione delle classi di oggetti.

Alla fine della sua esecuzione, Adaboost restituisce come output un classificatore. Nei capitoli successvi si darà una definizione più formale di quello che è un classificatore. Per il momento è sufficiente una definizione intuitiva: un classificatore *classifica* i vari oggetti, ovvero fornisce una *previsione* della relativa classe di appartenenza. La classificazione effettuata da questa componente approssima solamente la classificazione reale. La bontà di tale approssimazione sarà il parametro di valutazione della bontà generale del sistema. Nel caso di Adaboost il classificatore risultante sarà simile ad una collezione di test: il risultato di tali test, eseguiti su di un qualsiasi oggetto, fornirà la previsione della classificazione dell'oggetto stesso.

Rilevamento

In questa fase il sistema analizza i frame di profondità delle acquisizione in ordine sequenziale, alla ricerca di persone al suo interno.

Il classificatore ottenuto al termine dell'esecuzione di Adaboost, sarà in grado di classificare porzioni di immagini di profondità, ma non è in grado di predire direttamente, a partire da un intero frame, la presenza e la posizione di un persona al suo interno. Le porzioni analizzabili dal classificatore hanno dei vincoli dimensionali da rispettare. In prima approssimazione si può pensare a tali porzioni come a dei quadrati di dimensione costante. L'attività di rilevamento della persona all'interno del frame, quindi, consterà

[1] Per *spazio vuoto* si intende una regione di spazio il cui valore della distanza, percepita dal sensore, è molto vicino al quello della distanza del pavimento della stanza.

della sequenziale analisi di tutte le porzioni di frame che rispettano tali vincoli, al fine di coprire l'intera area.

Si vedrà in seguito che nei pressi di una persona nell'immagine di profondità, saranno molteplici le porzioni di frame per le quali il rilevamento darà esito positivo. Si pone quindi l'ulteriore problema di selezionare, delle tante porzioni che hanno dato esito positivo, quella che meglio approssima la reale posizione della persona.

Capitolo 2

Haar-Like Features

2.1 Definizione

2.1.1 Richiamo: cosa è una feature (caratteristica)

Le caratteristiche di un oggetto sono quelle proprietà elementari osservabili e misurabili.

È stato già detto che la scelta delle caratteristiche è fondamentale e dipende da cosa si vuole mettere in evidenza dell'oggetto in questione.

Ovviamente la scelta delle caratteristiche è sempre subordinata a ciò che si ha disposizione.

2.1.2 Wavelet di Haar

Le feature di Haar derivano dalle wavelet di Haar

Le feature di Haar si adattano molto bene alle proprietà che si vogliono misurare degli oggetti appartenenti all'applicazione d'interesse. Sono un costrutto derivante dalle *wavelet di Haar*.

Definizione informale delle wavelet di Haar

Alfréd Haar sviluppò il primo tipo di wavelet.

Furono sviluppate come un esempio di funzioni ortonormali di base per uno spazio funzionale.

In quanto base di uno spazio ortornomale, con le wavelet di Haar è possibile esprimere un qualsiasi segnale limitato. Esse costituiscono, sotto particolari ipotesi, un sistema di rappresentazione dei segnali duale all'analisi spettrale di Fourier. Hanno anche il vantaggio, rispetto a quest'ultimo, di mantenere l'informazione del tempo (approfondire).

Wavelet di Haar e DWT

Le wavelet di Haar sono state utilizzate nelle *Discrete Wavelet Transform*, in breve DWT.

Un'importante applicazione delle DWT è quella definita dal nuovo standard di compressione delle immagini JPEG2000.

Nelle applicazioni di machine learning e pattern recognition, le trasformazioni DWT furono utilizzate nei primi lavori di riconoscimento a partire da immagini RGB (riconoscimento dei pedoni). È da quest'ultima applicazione che hanno origine le feature di Haar come verranno trattate.

2.1.3 Formula di calcolo standard

Rappresentazione visuale

Le feature di Haar sono rappresentabili come due aree adiacenti, una chiara ed una scura. La somma delle intensità (il valore numerico) di tutti i pixel dell'area scura, viene sottratta alla somma delle intensità di tutti i pixel nell'area chiara. Ciò permette di evidenziare le differenze di intensità medie tra i valori dei pixel nelle due aree. [Formula a due aree] È da notare il fatto che, con le feature di Haar, non si vanno a valutare i singoli pixel al fine di individuarne un pattern, ma si ragiona procedendo per aree adiacenti.

Formula generale

La definizione delle feature di Haar può essere estesa all'utilizzo di più di due aree adiacenti. Il principio resta lo stesso: si hanno due gruppi di aree, uno chiaro ed uno scuro. La formula di calcolo si generalizza con estrema semplicità. [Formula generale con più aree]

Altri tipi di feature (OpenCv)

È possibile formulare feature di moltissime formule. La libreria di computer vision OpenCV mette a disposizione una grande varietà di feature, introducendo anche quelle la cui forma è inclinata di 45°. (Citare articolo in cui vi è la definizione delle feature a 45°).

Tipi di feature utilizzate

In questa applicazione vengono utilizzati solo due 4 tipi di feature, contro i 5 utilizzati nel sistema di riconoscimento di Viola-Jones.

2.1.4 Cosa mette in evidenza la feature di Haar

Immagini normali (Viola Jones)

Nella framework di riconoscimento dei volti di Viola-Jones, le immagini RGB sono la fonte di informazioni del sistema. Vengono applicate le feature di Haar a tali immagini (ovviamente dopo che queste ultime sono state trattate da opportune operazioni di preprocessing) e ciò che viene evidenziato sono le differenze di intensità tra le regioni della foto. A far variare l'intensità di un pixel in una foto concorrono l'illuminazione, il colore e molti altri fattori. Tuttavia, le feature di Haar divengono il mezzo con il quale si può riconoscere un volto umano. Ad esempio, è stato osservato che nella foto di un volto, l'area che racchiude entrambi gli occhi e l'inizio del naso è caratterizzato da una

particolare variazione di luminosità, misurabile e utilizzabile per descriminare le i volti da i non volti.

Immagini di profondità (Zhu Wong)

Nelle immagini di profondità, dove il valore di ogni pixel corrisponde alla distanza in millimetri della superficie dal sensore, applicare le feature di Haar ad un'area dell'immagine equivale a misurare le differenze di quota tra due aree adiacenti tra di loro. Il sistema di allenamento deciderà quali sono le feature migliori per il riconoscimento della persona, ma tutto si basa sul concetto che le differenze di quota osservabili dal profilo ripreso dall'alto di una persona sono caratteristiche della persona stessa e costituiscono il parametro di riconoscimento rispetto ad un qualsiasi altro oggetto. Una sedia, un tavolo o qualsiasi altro elemento presenterà delle differenze di quota differenti dal profilo della persona.

2.1.5 Formula di calcolo invariante ai resize

Anticipazione del problema del ridimensionamento

In seguito sarà necessario ridimensionare una feature in modo da coprire un'area più grande, in quanto, dovendo misurare le caratteristiche degli oggetti di interesse, questi ultimi sono variabili in dimensione. Il valore calcolato con la feature in questione, tuttavia, non dovrebbe essere troppo sensibile ai ridimensionamenti (che saranno frequenti).

Formula: Normalizzazione sull'area

Al fine di ottenere la massima invarianza ai ridimensionamenti dell'area della feature, il valore di essa viene normalizzato con l'estensione dell'area totale valutata. [Formula normalizzata]

2.1.6 Vantaggi

Differenze di intensità vs Valutazione dei singoli pixel

Il primo indiscutibile vantaggio delle feature di Haar sta nel fatto che la caratterizzazione dell'oggetto viene effettuata sulla base di osservazioni d'insieme su intere aree dell'immagine e non su ossevazioni globali effettuate su singoli pixel. Oltre alla grandissima complessità che una valutazione su singoli pixel introdurrebbe, bisogna prendere atto che, con dati soggetti a disturbi e alla presenza di rumore, delle caratteristiche misurate sui singoli pixel non sarebbero molto significative.

Differenze di intensità vs Estrazione dei contorni

L'estrazione dei contorni potrebbe essere più significativo della valutazione sui singoli pixel, ma continuano ad essere caratteristiche abbastanza complesse da calcolare ed ottenere. L'estrazione dei contorni, inoltre, è un concetto fortemente legato alle immagini RGB, usarlo con le immagini di profondità è un forzatura.

Estrema efficienza computazionale

Il principale vantaggio delle feature di Haar rispetto ad altri tipi più elaborati di feature resta la loro efficienza computazionale. Si vedrà che, con una particolare struttura dati di supporto, calcolare il valore di un feature di Haar per un'immagine è un'operazione eseguibile in un tempo costante.

2.2 Immagine Integrale

2.2.1 Definizione rigorosa dell'immagine integrale

Problema: efficienza nel calcolo di somme di pixel

Il calcolo della somma delle intensità di ciascun pixel appartenente ad un'area, necessario al fine di calcolare il valore delle feature di Haar, è un'operazione il cui costo varia all'aumentare della dimensione complessiva dell'area. Calcolare tali somme infatti ha complessità computazionale $\Theta(m \cdot n)$ con m ed n dimensione dell'area.

La soluzione a tale problema consiste nell'utilizzo dell'immagine integrale, una struttura dati che mette a permette di calcolare la somma dei pixel di qualsiasi area all'interno di essa in un tempo costante.

Definizione immagine integrale

Un'immagine integrale è un matrice delle stesse dimensioni dell'immagine di partenza. Ogni elemento di tale matrice contiene il valore della somma dei pixel che si trovano al di sopra e a destra (estremi inclusi) del pixel relativo alla posizione dell'elemento. [Formula]

Formula di calcolo della somma dei pixel in un'area

Attraverso l'immagine integrale è possibile calcolare l'area necessaria per una feature sommando i valori dei due vertici dell'area sulla diagonale principale e sottrandovi quelli dei vertici sulla diagonale secondaria. [Formula]

[Dimostrazione formula]

Qualsiasi calcolo di questo tipo richiederà un tempo costante, non più legato alle dimensioni dell'input. La complessità computazionale è quindi $\Theta(1)$.

2.2.2 Complessità computazionale generale

Complessità del calcolo dell'immagine integrale

La complessità computazione totale del calcolo dell'immagine integrale continua ad essere legata alla dimensione dell'input. Se l'immagine è larga w pixel ed alta h pixel, la complessità computazionale per la generazione dell'immagine integrale è pari a $\Theta(w \cdot h)$.

Convenienza del calcolo dell'immagine integrale

L'utilizzo delle immagini integrale è molto vantaggiosa nel momento in cui è necessario calcolare molte feature sulla stessa immagine. Volendo essere più espliciti, se la

complessità computazionale totale del calcolo di n feature senza l'utilizzo dell'immagine integrale è maggiore di quella per la generazione dell'immagine integrale stessa, allora è vantaggioso utilizzare tale struttura dati.

2.3 Decision Stump

2.3.1 Problema: utilizzare le feature

Una volta misurata una caratteristica di un oggetto, bisogna essere in grado di utilizzarla ai fini di classificarlo.

È necessario, quindi, un meccanismo per effettuare una previsione della classe di appartenenza dell'oggetto che si basa esclusivamente sul valore della feature misurata.

2.3.2 Definizione di albero decisionale

Un albero decisionale è un modello predittivo. Ogni nodo dell'albero rappresenta una variabile o una proprietà osservabile. Ogni arco dal nodo padre verso un nodo figlio rappresenta un possibile valore per la proprietà del nodo padre. Le foglie dell'albero rappresentano i valori che si predice possa assumere una variabile obiettivo. Il percorso dalla radice ad una foglia rappresenta la previsione del valore di tale variabile obiettivo a fronte delle osservazioni effettuate sui valori delle variabili rappresentate da ciascun nodo attraversato.

2.3.3 Definizione di decision stump

Radice: Test, funzione booleana

Il più semplice albero decisionale è il *decision stump*. Ha profondità unitaria, cioè si basa sull'osservazione di una sola variabile, in base al valore della quale può effettuare due diverse previsioni della variabile obiettivo (ha due foglie).

Foglie: risultati possibili

Formule di calcolo binaria

Formula di calcolo unica: polarità

Capitolo 3

L'Algoritmo di Allenamento: Adaboost

3.1 Apprendimento Supervisionato *Ensamble*

3.1.1 Apprendimento Supervisionato

Definizione

Gli algoritmi di apprendimento supervisionato funzionano attraverso l'utilizzo di un insieme di allenamento, ovvero un insieme i cui elementi sono formati da coppie di oggetti e le relative etichette. La funzione che associa ad ogni oggetto la corretta etichetta è sconosciuta. L'obiettivo dell'algoritmo è quello di trovare una funzione che approssima tale funzione, cioè che permetta di prevedere l'etichetta dell'oggetto.

Le possibili funzioni candidate ad approssimare il comportamento della funzione di classificazione reale, fanno parte dello spazio delle ipotesi.

Esempi di Supervised learning

Addestramento su reti neurali con apprendimento supervisionato. Un famoso algoritmo di supervised learning è il *Support Vector Machine* (SVM).

SVM → Pedestrian counting.

Overfitting

Le tecniche di addestramento supervisionato potrebbero incorrere in problemi di *overfitting*. Dato un insieme di allenamento, infatti, la ricerca delle ipotesi che approssimano al meglio la funzione obiettivo potrebbe essere influenzata da caratteristiche che sono comuni tra gli elementi dell'insieme stesso, ma che non sono discriminanti nella descrizione più generale della classe di oggetti. Ciò porta ad un sequenziale miglioramento delle prestazioni del riconoscimento per gli elementi del dataset di allenamento, ma a prestazioni peggiori in scala globale (ad esempio su elementi di un insieme di validazione o di testing).

Ensamble Learning

Gli algoritmi di ensamble learning sono una categoria di algoritmi di apprendimento supervisionato. L'approssimazione della funzione obiettivo avviene mediante la selezione di un insieme di funzioni dallo spazio delle ipotesi. La predizione della classe dell'oggetto avviene, poi, combinando tra loro il risultato delle predizioni formulate dalle singole ipotesi. Indicativamente, un classificatore allenato con un algoritmo di ensamble learning avrà prestazioni migliori.

3.1.2 *Adaptive Boosting*

Strong learner e Weak learner

Si fanno riferimento a due tipologie di classificatori (o ipotesi). I *weak learner* sono delle singole ipotesi, la cui attendibilità si considera di poco maggiore rispetto ad una previsione casuale. Gli *strong learner* invece sono delle combinazioni di weak learners. Le previsioni effettuate con un classificatore forte sono nettamente migliori delle precedenti.

Algoritmi di Boosting

Gli algoritmi di boosting sono la famiglia più popolare di algoritmi che implementano l'ensamble learning. Inizialmente, gli algoritmi di questo tipo, associano ad un insieme di allenamento una distribuzione: ad ogni elemento dell'insieme viene associato un peso. Iterativamente viene selezionato il migliore weak learner, cioè quello che riesce a classificare gli elementi del dataset compiendo meno errori possibili. Sostanzialmente l'idea che è alla base del successo degli algoritmi di boosting sta nel fatto che, tali pesi, vengono aggiornati per ogni elemento ad ogni iterazione in base al risultato della classificazione con il weak learner corrente. Viene diminuito il peso degli elementi classificati correttamente e aumentato quello degli altri, in modo tale da favorire la selezione, all'iterazione successiva, di un weak learner che copra gli errori del precedente. Quest'ultimo viene aggiunto allo strong learner finale, che alla fine del processo sarà costituito da una combinazione dei vari weak learner.

Adaboost

Adaboost è l'algoritmo proposto da Freund e Schapire che è valso loro il premio Gödel. Il nome deriva dalla fusione delle parole *Adaptive* e *Boosting* ed implementa le operazioni descritte precedentemente. Ciò che rende Adaboost un algoritmo di boosting *adaptive* è la politica di aggiornamento dei pesi relativi agli elementi dell'insieme di allenamento: viene decrementato il peso degli elementi classificati correttamente, lasciando invariato il peso degli altri. Il valore di tale decremento non è costante, ma dipende dall'errore pesato di classificazione complessivo per la relativa iterazione. Maggiore è l'errore, maggiore è il decremento dei pesi. In tale senso, Adaboost è un algoritmo di boosting che ad ogni iterazione cambia il proprio comportamento adattandosi al contesto.

3.2 Dataset di Allenamento

3.2.1 Categorie di Classificatori

Variabilità della forma HASP

La forma dell'immagine HASP della persona nel frame di profondità varia per molti motivi.

La prima causa di variazione della forma HASP è l'orientazione delle persona. L'immagine di una persona che cammina seguendo una direzione parallela al lato lungo del frame sarà indubbiamente diversa dall'immagine di una persona che cammina seguendo una direzione perpendicolare, così come sarà diversa dall'immagine di una persona che compie una traiettoria obliqua all'interno della stanza.

Un altro fattore importante che concorre nella variazione della forma è la distorsione prospettiva. L'immagine di una persona ripresa al centro del frame e quella della stessa persona ripresa in una zona periferica è molto diversa, anche se orientata sempre nella stessa direzione. L'entità della distorsione prospettica della forma è tanto maggiore quanto più il sensore è vicino al pavimento. D'altro canto, le caratteristiche tecniche di quest'ultimo impongono un limite massimo alla distanza. Considerata la realtà applicativa del sistema, la distanza massima del sensore dal pavimento sarà di poco inferiore all'altezza del soffitto in una comune abitazione. L'altezza a cui viene montato il sensore, quindi, non verrà considerato un parametro progettuale, bensì una condizione ambientale entro la quale il sistema deve operare. La distorsione prospettica, quindi, non può essere eliminata.

Un'ulteriore causa della variazione della forma dell'immagine HASP consiste nelle differenze di corporatura e di statura delle persone. Tali differenze vengono chiamate *differenze interclasse*, essendo delle caratteristiche variabili di oggetti che appartengono alla stessa classe. Ovviemente un buon sistema di rilevamento dovrebbe essere in grado di riconoscere soggetti di differente statura e corporatura. Per ovviare a questo problema verranno prese delle opportune misure di ridimensionamento delle immagini.

Definizione delle categorie di classificatori

Una forte variabilità intraclasse potrebbe portare alla sintesi di classificatori troppo laschi. Un buon approccio a questo problema consiste nel dividere gli oggetti, appartenenti alla stessa classe, in diverse categorie. Tali categorie devono essere scelte in modo tale che le differenze tra gli oggetti appartenti alla stessa categoria siano lievi, mentre quelle relative agli oggetti di differenti categorie siano più accentuati. Definite tali categorie, per ognuna di esse verrà sintetizzato un classificatore, allendandolo solo su elementi appartenenti ad una stessa categoria. Così facendo le differenze intraclasse di entità superiore vengono arginate.

L'orientazione della persona è il parametro su cui si possono formulare le diverse categorie. In questo sistema vengono allenati due classificatori forti che vanno ad operare in parallelo in fase di riconoscimento: il primo è allenato esclusivamente con immagini di persone che si muovono in direzione parallela al lato lungo del frame (informalmente, direzione *orizzontale*), il secondo con immagini di persone che si muovono in direzione perpendicolare alla prima (direzione *verticale*).

Sarebbe possibile anche definire ulteriori categorie basate sulla direzione delle persona, introducendo due classificatori dedicati alle due direzioni oblique rispetto alle prime due. Inoltre si può pensare di porre rimedio alla distorsione prospettica del sensore suddividendo il frame in aree ed allenando, per ognuna di esse, una coppia di classificatori orizzontale-verticale. Queste ultime due soluzioni introducono una certa complessità aggiuntiva ed in questa fase iniziale di sperimentazione non porteranno grandi vantaggi alla precisione complessiva del sistema.

3.2.2 Preparazione dei Dataset

Acquisizioni

Per la creazione dei dataset di allenamento è stato chiesto a 10 soggetti, di statura e corporatura differente, di percorrere una traiettoria differente per ogni registrazione. Delimitata l'area del pavimento corrispondente all'area di visualizzazione del frame di profondità, sono state individuate 3 traiettorie:

Orizzontale È stato chiesto ai soggetti di coprire tutta l'area muovendosi lungo la direzione parallela al lato maggiore;

Verticale È stato chiesto ai soggetti di comprire tutta l'area muovendosi lungo la direzione perpendicolare al lato maggiore;

Random È stato chiesto ai soggetti di seguire una traiettoria casuale, cercando di coprire tutta l'area del frame.

Le prime due traiettorie corrispondo alle categorie individuate precedentemente: da tali registrazioni vengono estratti gli oggetti per l'allenamento dei classificatori orizzontale e verticale. Le registrazioni dell'ultima traiettoria, invece, vengono utilizzate per la creazione del dataset di validazione.

Con un software apposito vengono sequenzialmente catturati e salvati i frame di profondità provenienti dal dispositivo Kinect, ad una frequenza massima di $30 fps$. Il software è in grado di catturare fino a 1000 frame, che vengono salvati in file binari grezzi, senza ricorrere a compressione. La durata delle registrazioni delle traiettorie orizzontale e verticale è subordinata al tempo necessario al soggetto per coprire tutta l'area del frame, mentre le registrazioni riferite alle traiettorie casuali sono tutte costituite da 1000 frame.

Ritaglio dei samples

È stato sviluppato *ad hoc* un applicativo dedicato alla creazione e alla gestione dei dataset di allenamento. Attraverso un'efficace interfaccia utente, permette di aprire e visualizzare una registrazione e scorrere ciascun frame. Una volta individuato il frame interessato, è possibile estrarre un ritaglio selezionando l'area. La porzione selezionata di frame viene immediatamente salvata in un file binario grezzo nella cartella contenente il dataset correntemente aperto. [Appendici per spiegazioni migliori]

3.2.3 Preprocessing

Resize

In fase di creazione dei dataset è necessario effettuare una prima operazione di preprocessing. In virtù del fatto che, a soggetti di corporatura differente, corrispondo immagini HASP di dimensioni differenti, è necessario normalizzare le dimensioni dei ritagli componenti i dataset di allenamento. Tutte le immagini vengono quindi ridimensionate a 24×24 pixel (valore noto in letteretura).

I possibili algoritmi per il ridimensionamento delle immagini sono *nearest neighbour*, *bilinear interpolation* ed *bicubic interpolation*. Il ridimensionamento degli elementi del dataset si può effettuare direttamente dal software di gestione, che implementa i tre algoritmi precedenti.

L'algoritmo nearest neighbour è quello maggiormente utilizzato in quanto è il meno invasivo dei tre.

Conversione delle distanze

Un'ulteriore operazione di preprocessing da effettuare sugli elementi del dataset è la conversione delle distanze. Convertire le distanze dal sensore in quote dal pavimento.

3.3 *Strong Learner* [Da revisionare]

3.3.1 Procedura di estrazione del classificatore forte

Sia $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ un insieme di n coppie costituite da un'immagine (x_i) e la relativa classificazione ($y_i \in \{0, 1\}$). Se $y_i = 1$, allora x_i appartiene alla classe *umano* (la coppia (x_i, y_i) prende il nome di *esempio positivo*), altrimenti appartiene alla classe *non umano* (la coppia (x_i, y_i) prende il nome di *esempio negativo*).

L'insieme D può essere partizionato come segue:

$$P = \{(x, y) \in D | y = 1\} \text{ e } N = \{(x, y) \in D | y = 0\}$$

Si tenga a mente che, essendo P ed N partizioni di D, valgono le seguenti[1]:

$$D = P \cup N \tag{3.1}$$

$$P \cap N = \emptyset \tag{3.2}$$

$$\#(D) = \#(P \cup N) = \#(P) + \#(N) \tag{3.3}$$

Si introduce anche il concetto di *classificatore debole*. Si tratta di una funzione che per una data immagine x in ingresso, assume il valore che simboleggia la presunta classe di appartenenza di quest'ultima. Nel dettaglio:

$$h(x) = \begin{cases} 1 & \text{se } pf(x) < p\theta \\ 0 & \text{altrimenti} \end{cases} \tag{3.4}$$

[1] La scrittura $\#(D)$ denota il numero di elementi dell'insieme D.

dove $f(x)$ è il valore di una feature di Haar applicata all'immagine x, $p \in \{-1, 1\}$ è detta *polarità* e θ è la *soglia* (*threshold*). Tutti i classificatori deboli sono costruiti usando un'unica feature.

L'obbiettivo di Adaboost è quello di formare un *classificatore forte* come combinazione lineare dei migliori classificatori deboli estraibili dal set di allenamento, dove il fattore moltiplicativo di ogni classificatore nella combinazione è inversamente proporzionale agli errori di classificazione compiuti da quest'ultimo in fase di allenamento.

Il seguente algoritmo descrive la procedura di estrazione e combinazone di T classificatori deboli.

1. Si associa ad ogni elemento $(x_i, y_i) \in D$ un peso w_i tale che $w_i = \frac{1}{2l}$ se $(x_i, y_i) \in P$ oppure $w_i = \frac{1}{2m}$ se $(x_i, y_i) \in N$, dove $l = \#(P)$ ed $m = \#(N)$ (numero degli esempi positivi e numero degli esempi negativi).

   Sia inoltre $n := \#(D) = \#(P) + \#(N) = l + m$.

2. *For* $t = [1 : T]$

   (a) Si normalizzano i pesi, in modo che la loro somma sia pari ad 1:

   $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

   (b) Si estrae il miglior classificatore debole. La procedura viene esposta nel dettaglio nella sezione 3.4, ma si tenga presente che il miglior classificatore è quello il cui *errore pesato* è minimo per la corrente iterazione.

   $$\epsilon_t = \min_{f,p,\theta} \{ \sum_{i=1}^{n} w_{t,i} \cdot |h(x_i, f, p, \theta) - y_i| \}$$

   Siano inoltre f_t, p_t, θ_t i parametri del classificatore debole che ne minimizzano l'errore pesato:

   $$h_t(x) := h(x, f_t, p_t, \theta_t)$$

   (c) $\beta_t \leftarrow \frac{\epsilon_t}{1-\epsilon_t}$

   (d) Si aggiornano i pesi

   $$w_{t+1,i} \leftarrow w_{t,i} \cdot \beta_t^{e_i}$$

   dove $e_i = 1$ se $(x_i, h_t(x_i)) \in D$ (ovvero se x_i è classificata correttamente), $e_i = 0$ altrimenti.

   (e) $\alpha_t \leftarrow \log(\frac{1}{\beta_t})$

3. Il classificatore forte è dato da:

$$F(x) = \begin{cases} 1 & \text{se } \sum_{t=1}^{T} \alpha_t h_t(x) > \theta \sum_{t=1}^{T} \alpha_t \\ 0 & \text{altrimenti} \end{cases} \tag{3.5}$$

   dove $\theta \in [0, 1]$ è la soglia.

Si noti che, nell'operazione 2b, l'errore pesato non è altro che la somma dei pesi degli esempi non classificati correttamente. Infatti:

$$h(x_i, f, p, \theta) = y_i \Rightarrow |h(x_i, f, p, \theta) - y_i| = 0$$
$$h(x_i, f, p, \theta) \neq y_i \Rightarrow |h(x_i, f, p, \theta) - y_i| = 1$$

Al punto 2c, il valore di β_t non è altro che il rapporto tra l'errore pesato del classificatore debole e la somma dei pesi delle immagini classificate correttamente. Tale valore è chiaramente $0 < \beta_t < 1$.

In fase di aggiornamento dei pesi (punto 2d), i pesi relativi ad esempi classificati correttamente vengono moltiplicati per β_t ($\beta_t^1 = \beta_t < 1$) e quindi decrementati, mentre gli altri vengono lasciati inalterati ($\beta_t^0 = 1$). Fare in modo che gli esempi non classificati correttamente abbiano un peso maggiore di quelli classificati correttamente è il modo per influenzare la scelta del classificatore debole successivo che andrà a colmare le lacune del suo predecessore.

La scelta della soglia per il classificatore forte (punto 3) deve minimizzare il numero di esempi classificati in modo errato.

3.4 *Weak Learner* [Da revisionare]

3.4.1 Procedura di estrazione del classificatore debole

Si è detto che un classificatore debole è costruito a partire da una feature di Haar. La scelta del migliore, quindi, mira ad identificare la feature, la polarità e la soglia che minimizzano l'errore pesato di classificazione.

Si ricordi che le feature di Haar sono degli indicatori di quanto le intensità dei pixel variano da una regione della feature ad un'altra. Il classificatore debole, quindi, classificherà l'immagine a seconda che tale indice sia maggiore o minore di una certa soglia. Il compito della polarità è quello di stabilire il verso della diseguaglianza.

Il pool di feature da testare è costituito - teoricamente - da tutte quelle individuabili nell'immagini di allenamento. Nell'opera di Viola e Jones vengono utilizzate immagini di allenamento di 24×24 pixel e 5 tipologie di feature ([9], sezione 2.2): il numero di possibili features in tale area è maggiore di 160000. In questa applicazione si utilizzano immagini di 160×100 pixel e 4 tipologie di feature: il pool è costituito da un numero di elementi maggiore di almeno 4 ordini di grandezza.

È da tener presente che moltissime di queste feature sono poco significative in questa situazione: non ha senso calcolare la variazione di intensità di due aree molto piccole con delle immagini che hanno una risoluzione tanto alta. Effettuando una prima scrematura, si cercherà di avere un pool di feature selezionabili la cui dimensione non superi quella del pool di Viola-Jones per più di un ordine di grandezza.

Sia $\{f_1, ..., f_k\}$ l'insieme di tutte le feature selezionabili, $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ l'insieme degli esempi di allenamento e $\{w_1, ..., w_n\}$ l'insieme dei relativi pesi. La scelta del classificatore debole avviene come descritto dal seguente algoritmo in pseudocodice:

1. Si calcolano T^+ e T^-, rispettivamente, somma dei pesi degli esempi negativi e di quelli negativi:
$$T^+ \leftarrow \sum_{i=1}^{n}(w_i y_i) \ , \ T^- \leftarrow \sum_{i=1}^{n}[w_i(1 - y_i)]$$

2. *For* $f = [f_1, ..., f_k]$

   (a) Si inizializza una lista di n elementi per memorizzare i valori della feature i-esima applicata ad ogni immagine di allenamento:

   $$values[i] \leftarrow f(x_i)\ \forall x_i \in D$$

   (b) Si ordinano gli elementi della lista in ordine crescente. Si tenga in conto che, dopo tale operazione, all'i-esima posizione della lista non corrisponderà più il valore della feature applicata all'i-esima immagine di allenamento.

   (c) Si inizializzano S^+ ed S^-, con le quali, scorrendo gli elementi della lista con un cursore, indicheremo rispettivamente la somma dei pesi degli esempi positivi e di quelli negativi: $S^+ \leftarrow 0, S^- \leftarrow 0$

   (d) *For* $i = [1 : n]$

      i. A causa del rilocamento degli indici, alla posizione i-esima della lista corrisponderà il valore della feature dell'elemento x_j con classificazione y_j e peso w_j tale che $(x_j, y_j) \in D$:

      $$x_j, y_j, w_j \Leftarrow values[i]$$

      ii. *If* $y_i = 1$ *Then*

      A. $S^+ \leftarrow S^+ + w_j$

      iii. *Else*

      A. $S^+ \leftarrow S^- + w_j$

      iv. Si calcola calcola l'errore pesato di classificazione:

      $$e_i = \min\{S^+ + (T^- - S^-), S^- + (T^+ - S^+)\}$$

   (e) Si determinano la polarità (p_f) e la soglia (θ_f) per cui l'errore pesato (ϵ_f) di classificazione per un classificatore che utilizza la feature f è minimo:

   $$p_f, \theta_f | \epsilon_f = \min\{e_1, ..., e_n\}$$

3. Si scelgono polarità (p) e soglia (θ) finali, ovvero quelle del classificatore debole con errore pesato ϵ minore tra tutti i classificatori possibili:

   $$p, \theta | \epsilon = \min\{\epsilon_1, ..., \epsilon_k\}$$

   Il miglior classificatore debole è quindi:

   $$h(x) := \begin{cases} 1 & \text{se } pf(x) < p\theta \\ 0 & \text{altrimenti} \end{cases} \tag{3.6}$$

Il metodo di identificazione della polarità e della soglia non viene riportato nell'algoritmo, essendo un passaggio che merita una trattazione a parte. Selezionare un valore di soglia vuol dire trovare il *punto che partiziona al meglio la lista dei valori della feature*

calcolata sulle immagini di allenamento, in modo tale da minimizzare gli errori di classificazione. La miglior soglia di una buona feature fa in modo che la maggior parte delle immagini appartenenti alla stessa classe assumano un valore minore (o maggiore). Si può notare, come diretta conseguenza, che con una feature pessima per la classificazione non sarà possibile trovare un valore di soglia che soddisfi tale criterio.

Al punto 2(d)iv viene calcolato l'errore pesato di classificazione per una feature f con soglia $values[i]$[2]. Per essere più espliciti, bisogna effettuare una serie di osservazioni:

1. T^+ (T^-) corrisponde alla somma dei pesi degli esempi positivi (negativi);
2. S^+ (S^-) corrisponde alla somma dei pesi degli esempi positivi (negativi) dalla prima posizione fino all'i-esima della lista (quella su cui è posizionato il cursore);
3. $T^+ - S^+$ ($T^- - S^-$) corrisponde alla somma dei pesi degli esempi positivi (negativi) dalla posizione $i+1$ della lista fino alla fine;
4. Per un classificatore della forma (3.6) con $p = 1$, S^+ corrisponde alla *somma dei pesi degli esempi classificati correttamente*, mentre $S^- + (T^+ - S^+)$ corrisponde alla somma dei pesi degli esempi classificati in modo scorretto;
5. Per l'osservazione 4 un classificatore (3.6) con $p = 1$, la quantità $S^- + (T^+ - S^+)$ è *l'errore pesato*;
6. Analogamente alle osservazioni 4 e 5, un classificatore (3.6) con $p = -1$ commetterà un errore pesato pari alla quantità $S^+ + (T^- - S^-)$.

Grazie alle osservazioni 5 e 6, dal semplice calcolo dell'errore di classificazione pesato, si ottiene anche il valore della polarità:

$$p = \begin{cases} 1 & \text{se } S^- + (T^+ - S^+) < S^+ + (T^- - S^-) \\ -1 & \text{altrimenti} \end{cases} \tag{3.7}$$

3.4.2 Valutazione della complessità computazionale

In definitiva, con uno scorrimento della lista ordinata si ottengo i parametri per la costruzione del classificatore debole. La complessità di tale operazione è fortemente legata all'algoritmo di ordinamento della lista, la quale ha $\Theta(n \log n)$ come limite teorico inferiore [1, p. 167]. Nell'implementazione è stato scelto proprio un algoritmo che avesse complessità $\Omega(n \log n)$ nel caso peggiore. Ripetendo queste operazioni per ognuna delle feature selezionabili, si ottiene che l'algoritmo di selezione del miglior classificatore debole ha complessità $\Theta(kn \log n)$.

[2] I possibili valori delle soglie corrispondono esattamente ai valori della feature calcolata sugli esempi di allenamento

Capitolo 4

Validazione e Regolazione dei Classificatori

4.1 Criteri di Valutazione

Criteri di valutazione dell'algoritmo di classificazione binaria. True positive rate, true negative rate, accuracy, (mcc).

4.2 Dataset di Validazione

4.2.1 Criteri di creazione delle registrazioni

Il dataset di validazione è stato creato dalle registrazioni a traiettoria casuale.

4.2.2 Altre caratteristiche

Il dataset è caratterizzato da 1500 elementi positivi e 1600 elementi negativi.

4.3 Massimizzazione all'*Accuracy*

4.3.1 Parametri liberi del classificatore

Numero di weak learner

Il numero di classificatori deboli per ogni classificatore forte è un parametro libero all'interno della procedura di allenamento. Vista la procedura di Adaboost nel capito precedente, potrebbe sembrare che, aggiungendo altri weak learner si possa migliorare progressivamente la precisione del sistema. Ciò generalmente non è vero: troppi classificatori deboli potrebbero generare problemi di *overfitting*. È necessario quindi conoscere il numero ottimale di classificatori deboli a comporre ciascun classificatore forte.

Soglia del classificatore

Il valore di soglia relativo a ciascun classificatore forte allenato non è mai stato precisato. È necessario quindi impostare tale valore al fine di migliorare complessivamente l'affidabilità del sistema.

4.3.2 Algoritmo di ricerca della soglia e del NWL ottimi

I parametri da definire sono quindi il numero di classificatori deboli ed il valore di soglia per ciascuno dei classificatori forti allenati. Entrambi vengono scelti, per ogni classificatore forte, al fine di massimizzarne l'accuratezza in relazione al dataset di validazione. Massimizzare l'accuratezza implica la massimizzazione dei positivi e dei negativi reali e la conseguente riduzione dei falsi positivi e negativi.

L'algoritmo per determinare tali valori è fondamentalmente semplice: ogni strong learner viene utilizzato per classificare gli elementi del dataset di validazione. L'attività di classificazione, in virtù dei parametri liberi appena identificati, sarà dipendente dalla coppia di valori (θ, N_{wl}) (per θ si intende il valore di soglia del classificatore e N_{wl} il numero di weak learner che lo compongono). Per ogni possibile coppia di parametri, viene misurata l'accuratezza del classificatore nel classificatare correttamente l'intero dataset, quindi viene scelta la coppia per cui l'accuratezza è massima.

Una procedura di questo tipo, se implementata senza adoperare dei piccoli accorgimenti, potrebbe richiedere molto tempo per essere portata a termine. [Algoritmo figo di selezione della coppia minima]

4.4 Analisi dei Risultati

Alla fine della procedura di selezione della coppia ottima, si ottengono i valori di accuratezza migliori del sistema. In letteratura il valore di soglia di un classificatore forte si assume essere pari a 0.5 ([3] e [9] aggiungere riferimenti più specifici). Nel lavoro presentato in [10], tuttavia, il valore di soglia non viene esplicitato, considerandolo come parametro libero. I risultati ottenuti con l'algoritmo di massimizzazione dell'accuracy, in quanto a valore di soglia, sono lievemente differenti da quelli proposti in letteratura, ma non troppo distanti.

Il numero dei classificatori deboli che compongono ciascun classificatore forte, è molto differente quello ottenuto in altre applicazioni, prima tra tutte il sistema di riconoscimento dei volti di Viola Jones. I classificatori forte di questo sistema sono formati da una decina di classificatori deboli, un numero straordinariamente piccolo se confrontato con i migliaia di weak learners necessari a costruire un classificatore di volti. Bisogna considerare però la diversa natura dei due problemi e i diversi canali di acquisizione dei dati per il riconoscimento. L'immagine di profondità di una persona ripresa dall'alto non che una grezza rappresentazione dell'individuo se paragonata all'immagine di un volto umano. La ricchezza di particolari di un volto, l'incredibile variabilità intraclasse e la sensibilità alle variazioni ambientali (come la luce), rende molto più complesso il problema di riconoscimento e di conseguenza è necessario avere dei classificatori molto più complessi. Per raggiungere elevate prestazioni con il framework di Viola-Jones, infatti, è necessario dividere un classificatore forte in diversi stadi, creando dei classificatori a cascata in grado di scartare nel minor tempo possibile, il maggior numero di oggetti che

non corrispondono ai volti umani. In questo sistema di rilevamento, tutto questo non è necessario. Un numero così basso di classificatori deboli non è dovuto ad un errore, ma è la conseguenza dell'estrema essenzialità delle immagini di profondità HASP. Si pensi alle caratteristiche espresse in linguaggio naturale dell'immagine del profilo della persona: sono solamente tre. Quante ne sarebbero necessarie per esprimere le principali caratteristiche di un volto umano?

In base a quanto presente in letteratura, i valori calcolati di *specificity* e *sensitivity* rivelano la bontà del sistema.

Capitolo 5

Rilevamento

5.1 Tecnica di Rilevamento

5.1.1 Detection Window

5.1.2 Rilevazione su frame

Resize detection window

Slide detection window

5.2 Selezione della Finestra Migliore

5.2.1 Introduzione al problema

5.2.2 Algoritmo di selezione

5.3 Confronto con l'Algoritmo G-C

Capitolo 6

Conclusioni

Appendici

Appendice A

Software Sviluppato

A.1 Componenti

A.1.1 Creatore dei Dataset

Interfaccia Grafica

Strutture Dati per la persistenza

Tecnologie utilizzate

A.1.2 Allenamento

Punti critici dal punto di vista computazionale

Mex files

Architettura delle librerie

Struttura dati per la persistenza

A.1.3 Tuning, Testing, Rilevamento

Organizzazione degli script e delle funzioni

Struttura dati per la persistenza

A.2 Tecnologie utilizzate

A.2.1 C++ e Matlab

Matlab: prototipazione e componenti non critiche

C++: ottimizzazione delle componenti critiche

A.2.2 Git e Github [Opzionale]

Sistemi VCS

A.3 Proposte di miglioramento

Componenti da ottimizzare

Nuovi linguaggi da utilizzare

Appendice B

Cenni del funzionamento del sensore Kinect

Bibliografia

[1] Thomas H Cormen. Introduction to algorithms. 2009.

[2] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[3] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[4] Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

[5] Michael Oren, Constantine Papageorgiou, Pawan Sinha, Edgar Osuna, and Tomaso Poggio. Pedestrian detection using wavelet templates. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 193–199. IEEE, 1997.

[6] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. IEEE, 1998.

[7] ITUT Rec. T. 800— iso/iec 15444-1,“. *Information technology—JPEG*, 2000.

[8] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 1995.

[9] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[10] Lei Zhu and Kin-Hong Wong. Human tracking and counting using the kinect range sensor based on adaboost and kalman filter. *Advances in Visual Computing*, pages 582–591, 2013.