Università Politecnica delle Marche

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

e dell'Automazione

Implementazione di un algoritmo di identificazione della persona che utilizza frame di profondità

Implementation of a depth-based human identification algorithm

Relatore:
Prof. Ennio Gambi

Correlatori:
Prof.ssa Susanna Spinsante

Ing. Enea Cippitelli

Tesi di Laurea di:
Ilario Pierbattista

Anno Accademico 2014/2015

Indice

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 **??**, 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 dallo 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

Una volta misurata una caratteristica relativa ad un oggetto, è necessario trovare un meccanismo, da utilizzare per classificare l'oggetto, che si basi esclusivamente sul valore della misura stessa.

2.3.1 Definizione di Albero Decisionale

Un albero decisionale è un modello predittivo, per il quale, definita una variabile obiettivo, grazie ad una serie di osservazioni e valutazioni di alcune proprietà (o caratteristiche), si giunge a predirne il valore.

Valgono le seguenti considerazioni:

1. Ogni nodo dell'albero rappresenta una caratteristica osservabile.
2. Ogni arco dal nodo padre al nodo figlio rappresenta una proprietà, per la caratteristica relativa al nodo padre, che deve essere soddisfatta per percorrere l'arco stesso.
3. Le foglie dell'albero rappresentano i valori ammissibili per la variabile obiettivo.
4. Il percorso dalla radice ad una foglia rappresenta la previsione del valore della variabile obiettivo, a fronte delle osservazioni effettuate sulle caratteristiche relative a ciascun nodo attraversato.

2.3.2 Definizione di Decision Stump

Il più semplice albero decisionale è il *decision stump* (letteralmente *ceppo decisionale*), il quale ha profondità unitaria e solamente due foglie. Si basa sull'osservazione di una singola caratteristica per le previsioni della variabile obiettivo.

Alla radice del ceppo decisionale è presente una *funzione di valutazione*[1] $v : L \to \{V, F\}$, utilizzata per valutare la caratteristica associata alla radice stessa. A seconda del risultato della funzione di valutazione, il percorso terminerà in una o nell'altra foglia.

Pragmaticamente, intercalando questa iniziale definizione formale nell'attuale contesto applicativo, la radice di ogni ceppo decisionale conterrà un test con cui valutare la feature associata. Essendo le feature delle misure a valori reali (quanto meno in questo caso), tutto ciò che bisognerà verificare sarà l'appartenenza della misura ad un preciso range di valori per la selezione del percorso da seguire.

Si impone un valore di soglia che andrà a partizionare l'immagine della funzione di calcolo della feature: definita la soglia $\theta \in \mathbb{R}$ da associare alla feature $f : Dom(f) \to Img(f) \in \mathbb{R}$, si partizionerà come segue:

$$Img(f) = \{f(x) | f(x) < \theta\} \cup \{f(x) | f(x) \geq \theta\}$$

[1] Il dominio L è costituito da tutte le *formule ben formate*, ovvero da proposizione logiche e dalle relative combinazioni ottenute per mezzo di connettivi logici.

L'appartenenza della misura ad una o all'altra partizione, decreterà il percorso da seguire verso la foglia. In un problema di classificazione binaria, assumendo una classe come classe di oggetti positivi e l'altra come classe di oggetti negativi, associando alla prima il valore simbolico 1 ed alla seconda 0, un generico test da porre alla radice del ceppo, può essere della forma 2.1 oppure della forma 2.2.

$$h_1(x) = \begin{cases} 1 & \text{se } f(x) > \theta \\ 0 & \text{altrimenti} \end{cases} \tag{2.1}$$

$$h_2(x) = \begin{cases} 1 & \text{se } f(x) < \theta \\ 0 & \text{altrimenti} \end{cases} \tag{2.2}$$

Questi test non fanno altro che discriminare gli oggetti, misurandone una feature, in due classi distinte, a seconda che il valore misurato sia al di sotto o al di sopra del valore di soglia. Volendo trovare una descrizione formale unica, si introduce un ulteriore parametro $p \in \{-1, 1\}$, detto *polarità*, il cui unico compito è quello di invertire il segno della disequazione, permettendo di fondere le formule 2.1 e 2.2 in 2.3.

$$h(x) = \begin{cases} 1 & \text{se } f(x)p > \theta p \\ 0 & \text{altrimenti} \end{cases} \tag{2.3}$$

Capitolo 3

L'Algoritmo di Allenamento: Adaboost

3.1 *Ensamble Supervised Learning*

L'algoritmo di *machine learning* utilizzato dal Zhu e Wong in [11] è chiamato *Adaboost*, appartenente alla famiglia degli algoritmi di allenamento supervisionati, in particolare a quella degli *ensamble learner*.

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.

$$T = \{(x_1, y_1), \cdots, (x_n, y_n)\} \text{ con } y_1, \cdots, y_n \in L = \{l_1, \cdots, l_k, \cdots\} \tag{3.1}$$

Esiste una funzione f che associa ad ogni elemento x l'etichetta y corretta, ovvero in modo tale che:

$$f : Dom(f) \to L | \forall x \in Dom(f), (x, f(x)) \in T \tag{3.2}$$

Se l'insieme L delle etichette è costituito da un numero finito di elementi ($L = \{l_1, \cdots, l_k\}$), allora il problema dell'associazione oggetto-etichetta, prende il nome di problema di *classificatore* e la funzione f sarà il relativo classificatore reale.

L'obiettivo dell'apprendimento supervisionato è quello di trovare la *migliore* funzione h che approssima il classificatore reale, essendo sconosciuto.

L'algoritmo di apprendimento cerca la funzione h, chiamata *ipotesi*, all'interno dello *spazio delle ipotesi* $\mathcal{H}$, scegliendo quella in grado di fornire le prestazioni migliori, sia rispetto agli elementi dell'insieme di allenamento, che rispetto ad altri elementi aggiun-

tivi[1]. Si considera una ipotesi h una buona *generalizzazione* di f, quando riesce ad etichettare correttamente anche nuovi oggetti.

Molto spesso la funzione di classificazione reale f non dipende strettamente dall'oggetto x, ma è costituita da un *processo stocastico*: in tal caso, ciò che l'algoritmo deve apprendere è una *distribuzione di probabilità condizionale* $P(Y|x)$.

$$h^* = \arg\max_{h \in \mathcal{H}} P(h|data) \tag{3.3}$$

La scelta del corretto spazio delle ipotesi è fondamentale. È necessario trovare un compromesso tra l'*espressività delle ipotesi* nello spazio $\mathcal{H}$ e la *complessità computazionale della scelta di una buona ipotesi*. Complessivamente, il tempo di calcolo dell'allenamento, tende ad aumentare molto velocemente all'aumento della *dimensione* dello spazio $\mathcal{H}$ e della *complessità computazionale* nel calcolo da ogni singola ipotesi.

Esempi di Supervised Learning

Esistono molti algoritmi di apprendimento supervisionato. Alcuni dei più famosi sono l'*apprendimento di alberi decisionali*, l'*addestramento su reti neurali* e le *macchine a vettori di supporto*.

Nell'apprendimento di alberi decisionali, definito un insieme di allenamento (3.1) ed una collezione di *attributi* valutabili sugli oggetti di tale insieme, si costruisce l'albero ramo per ramo, associando alla valutazione di ogni attributo il risultato più probabile. Più nel dettaglio, se l'insieme degli esempi di allenamento non è vuoto, se la classificazione su tali esempi non è sempre la stessa e se l'insieme degli attributi valutabili non è vuoto, allora si seleziona l'attributo più *importante*[2]. Si valuta l'attributo su tutti gli esempi di allenamento e, per ogni valore che esso assume, si partiziona l'insieme di allenamento stesso in base a tale valore. Per ogni partizione, si allena un sottoalbero utilizzando la stessa procedura, prendendo in considerazione però, la collezione degli attributi privata dell'attributo appena valutato.

Nell'addrestramento su reti neurali

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.

La variabilità *intraclasse* delle caratteristiche degli esempi di allenamento, viene equivocata, in fase di eselezione, per variabilità *extraclasse* e utilizzata, erroneamente, per la classificazione.

[1] Si costruisce anche un insieme di testing per la valutazione delle prestazioni. Per non appesantire eccessivamente la lettura, i dettagli riguardo al testing vengono omessi ad un livello di trattazione così generale. Si faccia riferimento alla sottosezione 4.1.1 per ulteriori chiarimenti.

[2] Minore è l'*entropia* correlata all'attributo, maggiore è la sua *importanza*. Nella teoria dell'informazione, ci si riferisce al concetto di *entropia di Shannon*, ovvero alla misura dell'incertezza su una variabile aleatoria. Per una variabile aleatoria V con $v_1, \cdots, v_k$ possibili valori, ognuno con probabilità $P(v_i)$, l'entropia relativa a V sarà $H(V) = \sum_{i=1}^{k} P(v_i) \log_2 \frac{1}{P(v_i)} = -\sum_{i=1}^{k} P(v_i) \log_2 P(v_i)$

Ciò porta ad un sequenziale miglioramento delle prestazioni del riconoscimento per gli elementi del dataset di allenamento, ma a prestazioni peggiori nella classificazione di nuovi elementi.

Nella sottosezione 4.1.1, verranno esposte alcune tecniche per evitare l'insorgere di problemi di overfitting.

Ensamble Learning

Gli algoritmi di *ensamble learning* sono una particolare categoria di algoritmi di apprendimento supervisionato.

L'approssimazione della funzione obiettivo avviene mediante la selezione di una collezione di funzioni dallo spazio delle ipotesi. La predizione della classificazione 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 fa 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 reale ($y_i \in \{0, 1\}$). 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 presente che, essendo P ed N partizioni di D, valgono le seguenti[3]:

$$D = P \cup N \tag{3.4}$$

$$P \cap N = \emptyset \tag{3.5}$$

$$\#(D) = \#(P \cup N) = \#(P) + \#(N) \tag{3.6}$$

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

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

di appartenenza di quest'ultima. Nel dettaglio:

$$h(x) = \begin{cases} 1 & \text{se } pf(x) < p\theta \\ 0 & \text{altrimenti} \end{cases} \tag{3.7}$$

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:

   $$F(x) = \begin{cases} 1 & \text{se } \sum_{t=1}^{T} \alpha_t h_t(x) > \theta \sum_{t=1}^{T} \alpha_t \\ 0 & \text{altrimenti} \end{cases} \tag{3.8}$$

   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:

   $$h(x) := \begin{cases} 1 & \text{se } pf(x) < p\theta \\ 0 & \text{altrimenti} \end{cases} \tag{3.9}$$

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 $values[i]$[4]. 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 (3.9) 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 (3.9) con $p = 1$, la quantità $S^- + (T^+ - S^+)$ è *l'errore pesato*;
6. Analogamente alle osservazioni 4 e 5, un classificatore (3.9) 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:

$$p = \begin{cases} 1 & \text{se } S^- + (T^+ - S^+) < S^+ + (T^- - S^-) \\ -1 & \text{altrimenti} \end{cases} \tag{3.10}$$

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 $\Theta(kn \log n)$.

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

Capitolo 4

Validazione e Testing dei Classificatori

Al termine della fase di allenamento si ottengono dei classificatori della forma:

$$F(x) = \begin{cases} 1 & \text{se } \sum_{t=1}^{T} \alpha_t h_t(x) > \theta \sum_{t=1}^{T} \alpha_t \\ 0 & \text{altrimenti} \end{cases} \tag{3.8}$$

La valore di soglia θ, a questo punto, è ancora ignoto e dovrà essere fissato in modo tale da massimizzare le prestazioni del sistema. È necessario, inoltre, impostare il corretto numero di *weak learner* per ogni *strong learner*, al fine di evitare problemi di *overfitting*.

Queste operazioni sono volte alla *validazione* dei vari classificatori e devono essere affiancate da opportune operazioni di *testing* per la valutazione complessiva del sistema.

4.1 Criteri di Valutazione

Applicando i vari classificator ad un set di elementi si può visualizzare la distribuzione delle istanze degli oggetti rispetto alla classificazione predetta dai classificatori e rispetto a quella reale.

Si utilizza la *matrice di confusione* per descrivere in modo compatto tale distribuzione.

Definizione 1 (Matrice di Confusione)**.** In un generico problema di classificazione ad n classi per il quale è definito un insieme di *label* $L = \{l_1, ..., l_n\}$ ed è stato allenato un classificatore $F(x)$, sia $T = \{(x_1, y_1), ..., (x_m, y_m)\}$ un insieme di elementi per cui è nota la reale classificazione.

Si chiama *matrice di confusione* del classificatore F e relativa all'insieme di elementi T, la matrice $C \in \mathbb{N}^{n \times n}$, per cui ogni elemento c_{ij} è pari al numero di oggetti dell'insieme T la cui reale etichettatura corrisponde a l_i e per i quali il classificatore $F(x)$ prevede l'etichettatura l_j.

$$c_{ij} = \#(\{x | (x, l_i) \in T \wedge F(x) = l_j\}) \tag{4.1}$$

Gli elementi sulla diagonale principale di tale matrice denotano il numero di elementi per cui la predizione fornita dal classificatore è corretta. Infatti, nel caso in cui $i = j$, la

relazione 4.1 diventa:

$$c_{ii} = \#(\{x|(x, l_i) \in T \wedge F(x) = l_i\}) \Longrightarrow c_{ii} = \#(\{x|(x, F(x)) \in T\}) \tag{4.2}$$

Nel caso di classificazione dicotomica, la struttura della matrice di confusione è molto semplice:

$$C = \begin{pmatrix} \text{True Positives} & \text{False Negatives} \\ \text{False Positives} & \text{True Negatives} \end{pmatrix} \tag{4.3}$$

4.1.1 Modalità di Test

La scelta del set di elementi con i quali testare i classificatori, in fase di valutazione, è particolarmente importante. Esistono alcune alternative:

Training Dataset Si utilizza lo stesso dataset impiegato per allenare il costruttore. È la soluzione più semplice, ma presenta notevoli svantaggi, in quanto diventa impossibile rilevare ed evitare problemi di *overfitting*.

Validation Dataset Si crea un dataset da utilizzare esclusivamente per il testing o per la validazione che non contenga elementi dell'insieme di allenamento.

Cross Validation Questa tecnica prevede l'utilizzo di un unico dataset, utilizzato sia per l'allenamento che per la valutazione:

1. Si suddivide l'insieme in k sottoinsiemi, detti *k-fold*[1]
2. Il classificatore viene allenato utilizzando $k - 1$ sottoinsiemi e viene testato utilizzando il sottoinsieme rimanente. Questa operazione viene eseguita k volte.
3. La media delle singole prestazioni ottenute nei k esperimenti costituisce le prestazioni complessive del sistema.

Split Si suddivide l'insieme di dati disponibili in due set distinti, uno da utilizzare per l'allenamento, l'altro - solitamente più piccolo del primo - per la validazione e per il testing.

Per la validazione e per il testing è stato creato un dataset apposito, estratto da registrazioni completamente diverse da quelle utilizzate per i dataset di allenamento. Le porzioni di immagine di profondità che ritraggono la figura della persona, in questo dataset, provengono dalle registrazioni in cui i vari soggetti percorrono delle traiettorie a loro piacimento all'interno dell'area d'interesse.

Il dataset di validazione è costituito da 1500 elementi positivi (immagini che ritraggono persone) e da 1600 elementi negativi.

[1] Si è notato che, sperimentalmente, sono sufficienti una decina di *fold*.

4.1.2 Misure per Valutare le Prestazioni

Scegliere i criteri di valutazione della bontà delle previsioni fornite dai classificatori è un punto cruciale e non banale: l'utilizzo di una misura piuttosto che un'altra dipende dall'obiettivo che si vuole raggiungere.

Per i problemi di classificazione binaria sono disponibili le seguenti misure per valutarne le prestazioni.

Precision Porzione di predizioni positive corrette.

$$PR = \frac{TP}{TP + FP} \tag{4.4}$$

Sensitiviy o Recall Porzione delle istanze realmente positive classificate correttamente. Per P si intende il numero totale di istanze realmente positive.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{4.5}$$

Specificity Porzione di istanze realmente negative classificate correttamente. Per N si intende il numero totale di istanze negative.

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{4.6}$$

False Positive Rate Porzione di istanze realmente negative classificate erroneamente come positive.

$$FPR = \frac{FP}{N} = \frac{FP}{TN + FP} = 1 - TNR \tag{4.7}$$

Accuracy Porzione di istanze, sia positive che negative, classificate correttamente.

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.8}$$

F1 measure Media armonica tra *precision* e *recall*.

$$F1 = \frac{2PR \cdot TPR}{PR + TPR} = \frac{2TP}{2TP + FP + FN} \tag{4.9}$$

F1 measure e *accuracy* sono le misure più indicate per la valutazione complessiva dei classificatori, mentre le altre forniscono informazioni parziali.

Solitamente si utializzano *precision*, *recall* e *F1 measure*, nei problemi in cui, tra le due classi, una è molto meno interessante dell'altra, oppure nei casi in cui si vuole studiare l'andamento del classificatore nelle predizioni di una classe di oggetti in particolare.

D'altra parte, l'*accuracy* viene utilizzata quando tutte le classi ricoprono una posizione di egual interesse. In particolare, nei problemi binari, tale misura è una adeguata fonte di valutazione quando le due classi sono bilanciate.

In questa applicazione, poichè le classi e la distribuzione degli elementi di allenamento rispettano i criteri di quest'ultima misura, verrà utilizzata l'*accuracy* come misura di valutazione delle prestazioni.

4.2 Massimizzazione all'*Accuracy*

4.2.1 Parametri liberi del classificatore

I classificatori allenati con Adaboost non sono ancora pronti per poter essere utilizzati, vi sono ancora dei parametri liberi da vincolare. Dal momento che l'obiettivo è ottenere un sistema di rilevamento che sia più accurato possibile, questi parametri devono essere impostati in modo tale che l'*accuracy* sia massima.

Numero di weak learner

Il numero di weak learner[2] 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 ricavare 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.

4.2.2 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. Ciò implica la massimizzazione del numero di oggetti che vengono classificati correttamente 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}) [3]. 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. Di seguito viene presentato un metodo che permette di classificare molto velocemente il dataset di validazione utilizzando diversi valori per la coppia (θ, N_{wl}).

1. Si definisce il classificatore forte $F(x)$ (equazione 3.8) costituito dai primi T weak learner[4], selezionati da Adaboost.

[2] Abbreviando: NWL

[3] Per θ si intende il valore di soglia del classificatore e N_{wl} il numero di weak learner che lo compongono

[4] In questa applicazione vengono estratti i primi 200 classificatori deboli. Questi ultimi sono il limite superiore al numero di weak learner totali che potranno essere utilizzati.

2. Si definisce l'insieme $V = (x_1, y_1), ..., (x_m, y_m)$ di validazione.

3. Si costruisce la matrice di valutazione dei weak learner W. L'elemento alla posizione (i, j) di tale matrice sarà la valutazione del *j-esimo weak lerner* sul *i-esimo elemento del dataset*.

$$W = \begin{bmatrix} h_1(x_1) & h_2(x_1) & \cdots & h_j(x_1) & \cdots & h_T(x_1) \\ h_1(x_2) & h_2(x_2) & \cdots & h_j(x_2) & \cdots & h_T(x_2) \\ \vdots & \vdots & & \vdots & & \vdots \\ h_1(x_i) & h_2(x_i) & \cdots & h_j(x_i) & \cdots & h_T(x_i) \\ \vdots & \vdots & & \vdots & & \vdots \\ h_1(x_m) & h_2(x_m) & \cdots & h_j(x_m) & \cdots & h_T(x_m) \end{bmatrix} \tag{4.10}$$

Poichè $h_j(x_i) : V \to \{0, 1\}$, allora tale matrice sarà composta solamente da 0 ed 1 (con un piccolo abuso di notazione: $W \in \{0, 1\}^{m \times T}$).

4. Si costruisce una matrice triangolare superiore con i moltiplicatore dei weak learner, disposti come di seguito:

$$A = \begin{bmatrix} \alpha_1 & \alpha_1 & \cdots & \alpha_1 & \cdots & \alpha_1 \\ 0 & \alpha_2 & \cdots & \alpha_2 & \cdots & \alpha_2 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \cdots & \alpha_i & \cdots & \alpha_i \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & \alpha_T \end{bmatrix} \in \mathbb{R}^{T \times T} \tag{4.11}$$

5. Si calcola il prodotto della matrice W e della matrice A. L'elemento alla posizione (i, j) della matrice risultante, sarà il *valore della combinazione lineare dei primi j weak learner, valutati sul i-esimo elemento del dataset di validazione.*

$$\forall a_{ij} \in (W \cdot A), a_{ij} = \sum_{t=1}^{j} a_t h_t(x_i) \tag{4.12}$$

6. Si definiscono i possibili valori di sogli assumibili dal classificatore forte. Qui sono stati utilizzati tutti i valori da 0 ad 1, con un passo pari a 0.01, estremi inclusi[5].

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_l \end{bmatrix} \in \mathbb{R}^l \tag{4.13}$$

[5]Quindi i possibili valori di soglia sono 101: $\{0, 0.01, 0.02, ..., 0.98, 0.99, 1\}$

7. Si definisce un array con la somma cumulativa dei fattori moltiplicativi dei weak learner:

$$\widetilde{A} = \begin{bmatrix} \alpha_1 \\ \alpha_1 + \alpha_2 \\ \vdots \\ \sum_{t=1}^{T} \alpha_t \end{bmatrix} \in \mathbb{R}^T \tag{4.14}$$

8. *For* $i = [1...l]$ (ovvero, per ogni possibile valore di soglia):
   (a) *For* $t = [1...T]$:
      i. Si inizializzano $TP = TN = FP = FN = 0$.
      ii. *For* $k = [1...m]$ (cioè, per ogni elemento del dataset di validazione):
         A. Si classifica velocemente l'elemento utilizzando le strutture dati preparate in precedenza:

$$F(x_k) = \begin{cases} 1 & \text{se } a_{kt} > \theta_i \widetilde{a_t} \\ 0 & \text{altrimenti} \end{cases} \tag{4.15}$$

         B. Si confronta tale classificazione con la classificazione reale e si aggiornano i contatori:
            - $F(x_k) = y_k = 1 \Longrightarrow TP \leftarrow TP + 1$
            - $F(x_k) = y_k = 0 \Longrightarrow TN \leftarrow TN + 1$
            - $F(x_k) = 1 \wedge y_k = 0 \Longrightarrow FP \leftarrow FP + 1$
            - $F(x_k) = 0 \wedge y_k = 1 \Longrightarrow FN \leftarrow FN + 1$
         C. Si salvano i valori TP, TN, FP, FN relativi alla coppia (θ_i, N_{wl}) (con $N_{wl} = t$), organizzandoli in una struttura dati a piacere, purchè in un secondo momento si sia in grado di risalire esattamente al valore di soglia e al numero di weak learner utilizzati.
            In questo caso, ad esempio, sono state utilizzate 4 matrici distinte, formate da tante colonne quanti sono i weak learner totali e da tante righe quanti sono i possibili valori di soglia.

9. Per ogni coppia (θ, N_{wl}) calcolare l'*accuracy* (formula 4.8). Si seleziona la coppia di valori per i quali l'accuratezza del classificatore è massima.

4.3 Analisi dei Risultati

Alla termine della procedura di selezione della coppia ottima, si ottiene un classificatore la cui accuratezza è la migliore rispetto a tutte le altre possibili coppie di parametri.

4.3.1 Soglia del classificatore

In letteratura il valore di soglia di un classificatore forte si assume essere pari a 0.5, come descritto da Freund e Schapire in [3], Viola e Jones in [9].

Nel lavoro presentato da Zhu e Wong in [11], tuttavia, il valore di soglia non viene esplicitato, ma viene considerato un parametro libero, pur senza fornire alcun metodo esplicativo per la sua derivazione.

Il valore ottenuto tramite l'algoritmo di massimizzazione dell'accuratezza dei classificatori (sezione 4.2) si aggira intorno al valore 0.4 per entrambi.

Una prima interpretazione potrebbe essere la seguente: la discordanza, seppur contenuta, tra il valore di soglia presente in letteratura ed il valore calcolato può essere dovuto a problemi di overfitting al dataset di validazione. Senza dover creare un ulteriore dataset esclusivamente per il testing, potrebbe essere utile utilizza il valore 0.5 come soglia effettiva dei classificatori finali, eliminando l'overfitting, e considerare le soglie calcolate come delle controprova della correttezza dell'algoritmo di allenamento.

La seconda interpretazione, invece, è diametralmente opposta: l'aver ottenuto dei valori di soglia ottimi per accuratezza, potrebbe essere un valore aggiunto rispetto alla formulazione generale classica dei classificatori forti.

Per il momento si assumeranno validi i valori ottimi di soglia appena calcolati, tuttavia sarà solo in fase di rilevamento su frame reali che ci si potrà rendere conto della bontà di tale soluzione.

4.3.2 Numero di Weak Learner

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 learner necessari a costruire un classificatore di volti.

Tuttavia considerare 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 è una rappresentazione grezza, ma essenziale, dell'individuo ed è incredibilmente povera di informazione se paragonata all'immagine di un volto umano. La ricchezza di particolari di un volto, l'incredibile variabilità intraclasse dei volti, per non parlare della fortissima sensibilità delle immagini alle variazioni ambientali (come la luce), rende molto più complesso il problema di riconoscimento. Ecco perchè i classificatori di volti sono cento, mille volte 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 dei profili umani.

Come ulteriore argomentazione, si può pensare alle caratteristiche espresse in linguaggio naturale dell'immagine del profilo della persona: sono solamente tre. Quante ne sarebbero necessarie per esprimere le caratteristiche di un volto umano?

4.3.3 Sensitivity, Specificity, Accuracy

In base a quanto presente in letteratura, un buon sistema di rilevamento presenta dei valori di *sensitivity* intorno al 90% e di *specificity* intorno al 70%.

Ovviamente tali criteri di giudizio dipendono molto dall'applicazione finale del sistema. Esisteranno situazioni più critiche che richiederanno parametri più stringenti, ma in fase sperimentale ci si attiene a questi valori empirici.

I valori di *sensitivity*, *specificity* e *accuracy* complessivi[6], calcolati una volta che i parametri dei classificatori sono stati fissati in condizioni di massima accuratezza, sono i seguenti:

$$TPR = 96.87\% \tag{4.16}$$

$$TNR = 91.25\% \tag{4.17}$$

$$ACC = 93.87\% \tag{4.18}$$

Alla luce dei risultati ottenuti, il classificatore finale, testato sul dataset di validazione, risulta sufficientemente accurato rispetto alle valori empirici di riferimento.

[6] In riferimento al classificatore finale. I singoli classificatori forti, allenati con Adaboost, vengono utilizzati in parallelo in modo da formare uno unico.

Capitolo 5

Rilevamento

5.1 Tecnica di Rilevamento

5.1.1 Detection Window

Vengono create delle *detection window* per gestire il rilevamento su frames. La detection window viene fatta scorrere sequenzialmente su tutta l'area del frame, in modo da coprirne tutta l'area.

5.1.2 Resize della Detection Window

A causa della variabilità delle dimensioni della figura HASP della persona, si rende necessario poter effettuare dei ridimensionamenti della finestra di rilevamento. Per ridimensionarla bisogna ridimensionare tutte le feature di haar associate ai classificatori della finestra.

5.1.3 Sliding della Detection Window

Si fa scorrere la finestra in modo da coprire tutta l'area del frame. Ciò permette di effettuare delle rilevazioni in contesti statici (su singoli frame o su frame non correlati tra di loro). Quando si ha a che fare con frame sequenziali, è possibile evitare di analizzare tutte le parti del frame in condizioni di regime, rendendo il processo più veloce.

Transitorio

All'inizio di una rilevazione, il sistema è *cieco*: deve analizzare completamente tutta l'area del frame.

A Regime

Successivamente ad una prima scansione iniziale del frame, è possibile ridurre l'area da scandire per i rilevamenti successivi. Se è stato rilevato un umano, si procede ad analizzare solamente la porzione di aree nelle sue immediate vicinanze. In ogni caso, si continua ad analizzare il bordo del frame, in attesa di nuovi ingressi.

5.2 Selezione della Finestra Migliore

In corrispondenza di una persona, ci saranno più finestre di rilevamento a dare esito positivo, per cui è necessario trovare un meccanismo per la selezione dell'area esatta all'interno della quale si trova la persona.

In prima approssimazione è stato sviluppato un algoritmo basato sul calcolo del baricentro delle finestre. Ogni finestra aveva lo stesso peso.

In un secondo momento le finestre sono state pesate con valori decrescenti man mano che queste si allontanano dalla posizione della persona allo stato precedente. Finestre molto distanti, difficilmente portano a risultati veritieri, quindi vengono premiate le finestre di rilevamento nelle immediate vicinanze della persona.

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

Accenni sul Funzionamento e le Caratteristiche del Sensore del Kinect

Ciò che viene comunemente chiamato *sensore di profondità* o *sensore di distanza*, in riferimento al dispositivo Kinect, è in realtà uno *scanner 3D a luce strutturata*.

Un sorgente di raggi infrarossi proietta una serie di pattern codificati nello spazio. Le superfici colpite inducono una deformazione nella struttura di tali pattern. Il pattern deformato viene quindi catturato da una o più telecamere, che confrontando la deformazione con il pattern originario, riescono a ricostruire, dalla rappresentazione bidimensionale di ogni punto nello spazio, le sue coordinate tridimensionali.

Il risultato di un sensore di questo tipo è un insieme di triplette (x, y, z), organizzate in una *immagine di profontità*, una struttura dati che è molto simile ad una semplice immagine in scala dei grigi[1], dove il valore di ogni pixel rappresenta la misura in millimetri della distanza della superficie dal sensore.

La massima affidabilità del sensore del Kinect V2 si ha per distanza comprese tra $50cm$ e $4,5m$. Il dispositivo è montato al soffitto a $2,8m$ da terra e ha un campo visivo di $70° \times 60°$, il quale, all'altezza del pavimento, determina un'area di cattura di circa $4m \times 5m$.

La dimensione di ogni immagine di profondità è di 512×424 pixel. Nativamente non vengono codificate in alcun modo particolare, sono delle semplici matrici di interi. È in grado di catturarne fino a 30 al secondo. Utilizzando un apposito software di registrazione è stato possibile mettere insieme dei video di profondità a 30 fps.

[1] La forte somiglianza con le immagini in scala dei grigi è supportata dal fatto che ogni pixel è codificato utilizzando 16bit.

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] Yi-Qing Wang. An analysis of the viola-jones face detection algorithm. *Image Processing On Line*, 4:128–148, 2014.

[11] 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.