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

Correlatore:
Prof.ssa Susanna Spinsante
Ing. Samuele Gasparrini (?)

Tesi di Laurea di:
Ilario Pierbattista

Anno Accademico 2014/2015

Indice

Capitolo 1

Introduzione

Presentazione del problema, Stato dell'arte e descrizione delle caratteristiche peculiari del problema di riconoscimento. Cenni sul funzionamento del sensore Kinect. Head and Shoulder Profile. Analisi del problema di classificazione.

Il monitoraggio delle persone è un problema di visione artificiale di importanza fondamentale. Sistemi di *human sensing* vengono continuamente sviluppati ed utilizzati nei più disparati contesti applicativi. Sistemi di sorveglianza, apparecchiatura di supporto per missioni di salvataggio, dispositivi designati all'utilizzo in ambienti assistivi automatizzati e persino alcuni sistemi automatici per effettuare indagini di mercato utilizzano tecniche di percezione automatica delle persone dall'elaborazione dei dati acquisiti per mezzo di sensori.

Nel 2013 Zhu e Wong descrivono in [10] un sistema allenabile di rilevamento e conteggio delle persone che attraversano una stanza. Il riconoscimento della persona avviene elaborando i dati catturati dal sensore di profondità di un dispositivo Kinect, il quale è montato sul soffitto della stanza ed è puntato verso il pavimento. La crescente popolarità dei dispositivi Kinect, anche al di fuori degli ambienti videoludici, lo rende un interessante oggetto di studio.

1.1 Head and Shoulder Profile

In condizioni ottimali, la figura umana ripresa dall'alto è composta solamente dalla testa e dalle spalle. Finchè si trova quasi in corrispondenza del sensore, nella parte centrale dell'immagine di profondità, la restante parte del corpo rimane quasi del tutto nascosta. Sulla posizione delle braccia non si possono fare delle assunzioni precise.

In figura 1.1, la figura dell'individuo a destra corrisponde a tale descrizione, mentre dell'altro è visibile parte del corpo ed una delle due spalle è nascosta dalla testa. Nelle zone periferiche di un frame di profondità, l'immagine è soggetta alla *distorsione prospettica* così come lo è quella di qualunque telecamere RGB.

Tale distorsione costituisce un disturbo, dal momento che la figura dello stesso soggetto varia a seconda della relativa posizione nell'area di visualizzazione. Si vedrà più avanti come affrontare tale situazione, per il momento si considerano solamente le immagini proveniente dalle zone centrali dei frame (come quella in figura 1.2).

Un grande vantaggio dell'utilizzo del Kinect in questa configurazione è l'*assenza di occlusione*. Infatti, rispetto ai molteplici sistemi di riconoscimento frontali, i soggetti non possono nascondersi l'uno con l'altro al sensore.

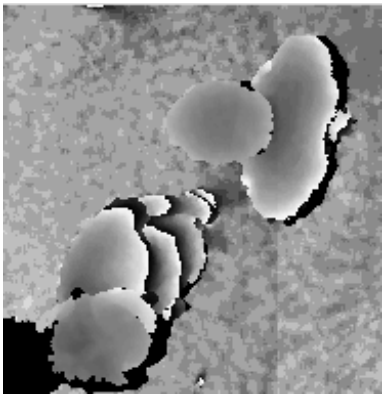

Figura 1.1: Due persone in un ritaglio proveniente da un frame di profondità. Il frame è stato acquisito con il Kinect V1, a differenza di quello in figura 1.2, acquisito con il Kinect V2.

Utilizzare le immagini di profondità significa ragionare con le distanze: piuttosto che cercare di descrivere l'immagine del profilo umano in termini di forma, deve essere descritto in termini di differenze di quota rispetto all'ambiente circostante. Alla luce di ciò, si possono identificare i seguenti criteri descrittivi:

1. L'immagine di una persona è caratterizzata da uno spazio vuoto di fronte ad essa e dietro di essa. Per *spazio vuoto* si intende una regione la cui distanza dal sensore è circa quella del pavimento.

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.

Di qui in avanti, con l'acronimo *HASP* (*Head And Shoulder Profile*), ci si riferirà proprio al profilo della persona ripreso dall'alto che soddisfa i criteri appena elencati.

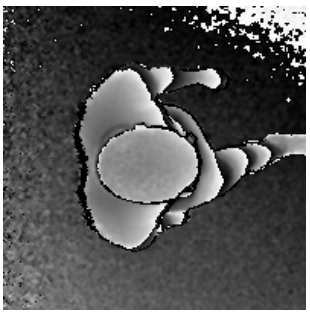

Figura 1.2: Una persona mentre cammina

Capitolo 2

Panoramica del sistema

Riproducendo la configurazione proposta in [10] al soffitto del laboratorio è stato fissato un dispositivo Kinect V2 rivolto verso il pavimento: all'ingresso di una persona nella visuale del Kinect, ne saranno ben visibili la testa e le spalle.

2.1 Un Problema di Classificazione

Quello che il sistema di rilevamento deve fare è riconoscere se e dove è presente l'immagine HASP di una persona all'interno del frame di profondità. Ciò avviene analizzando sequenzialmente delle porzioni del frame originale in modo da coprire tutta l'area. Si tornerà più avanti su questo aspetto, per il momento si consideri che il problema di rilevamento è naturalmente riconducibile ad un problema di classificazione.

Gli oggetti da classificare sono immagini di profondità e le possibili classi di appartenenza di tali oggetti sono due: la classe delle immagini che ritraggono il profilo di una persona è la classe delle immagini che invece ne sono prive. Il fatto che vi siano solamente due classi lo rende un *problema di classificazione binario*.

L'operazione di classificazione avviene mediante la misurazione di alcune *caratteristiche* (*feature*) interessanti dell'immagine di profondità. I criteri di riconoscimento presentati nella sezione 1.1 sono dei validi esempi di caratteristiche. Sono espresse in linguaggio naturale e sono comprensibili agli esseri umani, ma mancano di rigore e perciò non sono direttamente implementabili come criteri di classificazione delle immagini HASP.

Il *classificatore* è il componente che esegue effettivamente la classificazione. La definizione di un classificatore può avvenire *manualmente*, mediante l'applicazione di un modello atto a descrivere al meglio gli oggetti di una determinata classe, oppure può essere *allenato*. Gli algoritmi per la definizione di classificatori allenati si distinguono a loro volta tra algoritmi di *allenamento supervisionato* e di *allenamento non supervisionato*, a seconda che facciano uso o meno di un insieme di allenamento.

Un insieme di allenamento non è altro che un insieme di oggetti adeguati al problema di classificazione per cui viene fornita la classificazione *reale*. In questo caso saranno delle immagini di profondità opportunamente marcate come immagini che contengono o meno il profilo di una persona.

Il sistema in esame prevede l'utilizzo di classificatori allenati con *Adaboost*. Sono molti i sistemi di riconoscimento di immagini che utilizzano Adaboost per allenare i

rispettivi classificatori, primo tra tutti il quello di riconoscimento facciale proposto da Viola e Jones [9], ad oggi considerato il più robusto ed efficiente della sua categoria.

La forte somiglianza con il lavoro di Viola e Jones farà sì che saranno molteplici i riferimenti ad esso nel testo e i dettagli implementativi in comune.

2.2 Preprocessing

Ogni pixel di un'immagine di profondità denota la distanza in millimetri della superficie dell'oggetto dal sensore. Tuttavia, prima di poter essere utilizzate da qualsiasi componente del sistema di rilevamento, occorre convertire il valore di ogni pixel in quota della superficie dal pavimento.

Il pavimento è la superficie la cui distanza dal sensore è massima, quindi per individuarne il valore sarebbe sufficiente ricercare il valore massimo di ciascuna immagine di profondità. Nella pratica ciò non è vero, in quanto, a causa della distorsione prospettica, nelle aree periferiche dell'immagine la distanza percepita del pavimento è maggiore rispetto a quella misurata al centro dell'immagine.

Quindi, per determinare la reale distanza del pavimento è necessario disporre di un'immagine di riferimento che ritragga solamente il pavimento stesso. A questo punto la distanza di riferimento è quella misurata esattamente al centro del frame.

La trasformazione delle distanze in quote avviene, come è intuibile, sostituendo il valore di ogni pixel d con il valore d'.

$$d' = d - d_{pavimento} \tag{2.1}$$

L'operazione di conversione ha complessità computazionale $\Theta(n \cdot m)$, dove n ed m sono le dimensioni del frame.

Capitolo 3

Haar-like Features

Definizione delle feature. Derivazione delle feature dalle wavelet di Haar. Immagini integrali. Decision Stump.

La descrizione di un qualsiasi oggetto avviene attraverso la descrizione dei suoi attributi. La forma geometrica, le dimensioni, il peso, il colore sono solo una manciata dei possibili attributi con i quali descrivere un oggetto. L'essere umano è dotato di 5 sensi attraverso i quali è in grado di *percepire* la realtà circostante e di un'infinità di modi con cui *descriverla*.

Il riconoscimento di un oggetto avviene solo in un secondo momento, attraverso l'analisi della sua descrizione, valutando gli attributi che, in quel contesto, sono più signicativi. Nell'ambito della visione artificiale, il riconoscimento degli oggetti segue esattamente lo stesso flusso. Gli attributi di un oggetto vengono descritti mediante l'utilizzo di *feature*, ovvero dei meccanismi per misurare le varie proprietà dell'oggetto stesso.

Le immagini di profondità sono il frutto della percezione della realtà, le feature di Haar costituiscono gli attributi con i quali è possibile descrivere tali immagini.

3.1 Definizione

Le feature di Haar riescono a misurare la quantità e il verso delle variazioni di intensità tra due regioni adiacenti di una immagine. La comune rappresentazione grafica di tali feature mette in evidenza le due regioni colorandone una bianco e l'altra di nero. La somma delle intensità dei pixel della regione bianca a cui viene sottratta la somma delle intensità della regione nera fornisce una misura della variazione media di intensità tra le due regioni (equazione 3.1).

Di feature ne esistono di diverse forme: in figura 3.1 sono presenti quelle utilizzate in questo ambito, ma disponibili molte forme di feature utilizzate da altri sistemi di object detection. Non tutte sono formate solamente da *due regioni adiacenti*, ma l'importante è che tutte abbiano una regione *nera* ed una regione *bianca*.

L'equazione 3.1 fornisce una descrizione informale del funzionamento delle feature di Haar. Con $E(Area)$ si intende la somma delle intensità di tutti i pixel che appartengono all'area specificata.

$$f(Img) = E(Area_{white}) - E(Area_{black}) \tag{3.1}$$

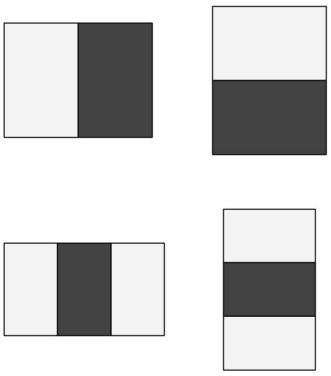

Figura 3.1: Tipi di feature di Haar utilizzati in [10].

Il segno del valore di una feature identifica il verso della variazione. Se si prende in esame la forma di feature in alto a sinistra della figura 3.1, un valore positivo denota un'intensità mediamente maggiore nella regione bianca rispetto alla media della regione nera e viceversa.

Evidenziare le variazioni di intensità delle immagini integrali attraverso l'applicazione delle feature di Haar equivale ad evidenziare le differenze di quota medie delle regioni individuate dalla feature e questo le rende particolarmente adatte a questa applicazione.

In seguito sarà necessario dover calcolare gruppi di feature in diverse scale. Ridimensionare un gruppo di feature è una cosa semplice e verrà trattata in seguito, ma affinchè il valore della feature sia il meno possibile sensibile ai ridimensionamenti è necessario rapportare tale valore all'estensione totale dell'area interessata (equazione 3.2).

$$f'(Img) = \frac{E(Area_{white}) - E(Area_{black})}{size(Area_{white}) + size(Area_{black})} \tag{3.2}$$

3.2 Dalle Wavelet di Haar alle Feature

Le feature di Haar derivano dall'estensione bidimensionale delle wavelet di Haar, che sono il primo esempio di wavelet e vennero sviluppate nel 1909 da Alfrèd Haar [4]. La wavelet madre (equazione 3.3) è una funzione oscillante di lunghezza finita, caratteristica comune a tutti i tipi di wavelet.

$$\psi(x) = \begin{cases} 1 & 0 \leq t < 1/2 \\ -1 & 1/2 \leq t < 1 \\ 0 & \text{altrimenti} \end{cases} \tag{3.3}$$

Ogni segnale può essere rappresentato per mezzo delle wavelet di Haar. Costituiscono un sistema di rappresentazione duale alla serie di Fourier.

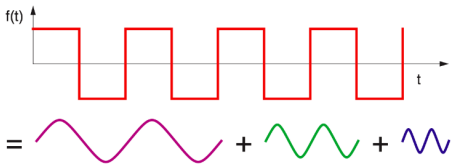

Figura 3.2: Rappresentazione di un segnale con una serie di funzioni armoniche.

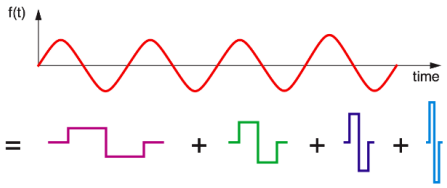

Figura 3.3: Rappresentazione di un segnale con una serie di wavelet di Haar.

Alfrèd Haar propose anche la prima DWT (*Discrete Wavelet Transform*) con la quale, le wavelet che compongono un segnale, vengono campionate discretamente. Le applica-

zioni sono notevoli e ad ampio spettro, prime tra tutte quelle nell'ambito della codica dei segnali e nella compressione dei dati. Per fare un esempio, lo standard di compressione JPEG2000 sfrutta la trasformazione DWT [7] per ottenere risultati qualitativamente migliori rispetto allo standard precedente.

Una variante della trasformazione DWT con le wavelet di Haar bidimesionali (figura 3.2) è stata utilizzata da Oren et al. [5] in un sistema in grado di riconoscere la presenza di pedoni in delle immagini.

Figura 3.4: Wavelet di Haar bidimensionali utilizzate in [5].

Applicando alle immagini la trasformata DWT in diverse scale, si passa da una rappresentazione dell'immagine in scala dei grigi, ad una rappresentazione in termini di coefficienti delle wavelet di Haar. Tali coefficienti denotano le differenze di intensità tra le aree adiacenti dell'immagine e vengono utilizzati per evidenziare analogie strutturali delle immagini che contengono la figura di un pedone.

Nella descrizione di un framework generale per l'object detection, Papageorgiou et al. [6] utilizzano la trasformata DWT a wavelet di Haar bidimensionali non per l'estrazione di un template dell'oggetto espresso in variazioni di intensità, bensì per la selezione delle wavelet più significative al fine del riconoscimento dell'oggetto.

Una wavelet di Haar bidimensionale di una data dimensione, utilizzata per campionare un'immagine in una posizione specifica mette in evidenza le differenze di intensità dell'immagine nella regione descritta dall'area della wavelet. Tale differenza di intensità costituisce una proprietà osservabile e misurabile dell'immagine che ritrae l'oggetto. Queste sono le *Haar-like feature* (o semplicemente feature di Haar).

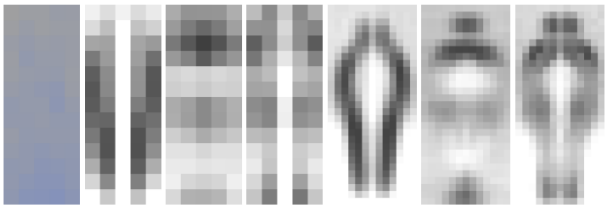

Figura 3.5: Coefficienti delle wavelet di trasformate DWT di diverse scale applicate alla stessa immagine. I coefficienti vengono codificati utilizzando la scala dei grigi [5, Figura 3].

3.3 Immagine Integrale

Le feature di Haar vengono utilizzate anche nel riconoscimento dei volti umani di Viola-Jones. Sono molto apprezzate, sebbene siano molto primitive rispetto ad altri tipi più evoluti di feature, per la loro efficienza computazionale.

Il calcolo della somma delle intensità dei pixel di un'area è un'operazione il cui costo aumenta linearmente con la quantità di pixel. Introducendo il concetto di *immagine integrale*, ottenuta rielaborando l'immagine di partenza, tale somma viene eseguita una volta per ogni immagine, permettendo in seguito di calcolare il valore di ogni feature in un tempo costante.

Definizione 1. Sia I un'immagine larga w pixel ed alta h pixel. Con la scrittura $I(x, y)$ si identifica il pixel dell'immagine I che si trova alla colonna x e alla riga y. Si definisce *immagine integrale* (altrimenti detta *Summed Area Table*) una seconda immagine SAT

delle stesse dimensioni per cui vale $\forall x \in [1, w], y \in [1, h]$:

$$SAT(x, y) = \sum_{i=1}^{x} \sum_{j=1}^{y} I(i, j) \tag{3.4}$$

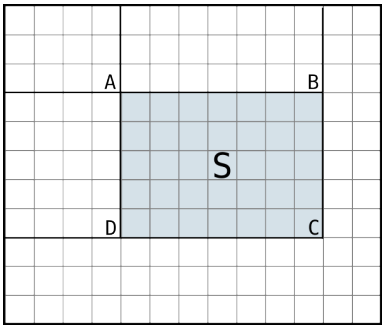

Figura 3.6

Calcolare la somma del valore dei pixel contenuti in una regione rettangolare dell'immagine originale, con l'ausilio dell'immagine integrale, è un'operazione velocissima.

Si consideri la figura 3.6: si vuole calcolare la somma del valore dei pixel nella regione S evidenziata. I vertici del rettangolo saranno i punti $A : (x_1, y_1)$, $B : (x_2, y_1)$, $C : (x_2, y_2)$, $D : (x_1, y_2)$, notando che $1 \leq x_1 \leq x_2 \leq w$ e che $1 \leq y_1 \leq y2 \leq h$ [1]. Quindi:

$$\sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} I(i, j) = \sum_{i=1}^{x_2} \sum_{j=y_1}^{y_2} I(i, j) - \sum_{i=1}^{x_1} \sum_{j=y_1}^{y_2} I(i, j) =$$

$$= \left(\sum_{i=1}^{x_2} \sum_{j=1}^{y_2} I(i, j) - \sum_{i=1}^{x_2} \sum_{j=1}^{y_1} I(i, j) \right) - \left(\sum_{i=1}^{x_1} \sum_{j=1}^{y_2} I(i, j) - \sum_{i=1}^{x_1} \sum_{j=1}^{y_1} I(i, j) \right) =$$

$$= SAT(C) - SAT(B) - SAT(D) + SAT(A)$$

Elaborare l'immagine integrale è un'operazione con complessità computazionale $\Theta(w \cdot h)$, cioè il cui costo varia linearmente con la dimensione dell'immagine. Una volta ottenuta però, permette di calcolare qualsiasi somma di pixel in regioni rettangolari con operazioni di complessità $\Theta(1)$.

Grazie all'immagine integrale è quindi possibile valutare in tempi brevi, gruppi enormi di feature sulla stessa immagine.

3.4 Decision Stump

Si tratta di *alberi decisionali* binari di profondità unitaria. Vi è un unico test alla radice, in base al quale si decide l'appartenenza di un oggetto ad due classi. Combinando tra loro diversi decision stump, è possibile creare alberi decisionali più complessi da poter utilizzare in alberi di classificazione non binari.

I decision stump costituiscono il meccanismo più semplice di classificazione che si può avere a disposizione. Il test da collocare alla radice dell'albero viene eseguito sul valore di una feature di Haar.

$$test_1(x) = \begin{cases} true & \text{se } f_1(x) < \theta_1 \\ false & \text{altrimenti} \end{cases} \tag{3.5}$$

[1] Nelle immagini raster, si iniziano a contare le colonne e le righe dal punto in alto a sinistra.

$$test_2(x) = \begin{cases} true & \text{se } f_2(x) > \theta_2 \\ false & \text{altrimenti} \end{cases} \quad (3.6)$$

Le equazioni 3.5 e 3.6 descrivono due possibili test decisionali che possono essere collocati alla radice di un decision stump. Per standardizzare la forma dei test si utilizzerà l'equazione 3.7.

$$test(x) = \begin{cases} true & \text{se } f(x)p > p\theta \\ false & \text{altrimenti} \end{cases} \quad (3.7)$$

In riferimento all'equazione 3.7, x è un'immagine di profondità, f una feature di Haar ed $f(x)$ il valore della feature misurato sull'immagine x. Il parametro $p \in \{1, -1\}$, che assumerà il nome di *polarità*, serve solamente per invertire il segno della disequazione all'occorrenza. Il parametro θ è il valore di *soglia* (*threshold*).

Utilizzare i decision stump con le immagini di profondità equavale a verificare la presenza di variazioni di quota tra regioni contigue dell'immagine al di sopra o al di sotto di un certo valore di soglia. La scelta del valore di soglia non è un problema banale e sarà trattato approfonditamente nel prossimo capitolo.

Il successo o l'insuccesso del test deve essere interpretato correttamente ai fini della classificazione. Di qui in avanti qualsiasi test decisionale applicato ad immagini di profondità che avrà successo, segnalerà la presenza di una persona, mentre, viceversa, quelli che non avranno successo ne segnaleranno l'assenza.

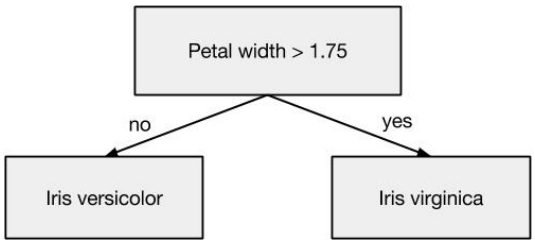

Figura 3.7: Esempio di decision stump usato per dividere gli oggetti in due classi delle tre descritte nel dataset delle caratteristiche dei fiori di Iris [2]

Capitolo 4

Adaboost

Classificazione nell'ambito degli algoritmi di apprendimento automatico. Costruzione e preparazione dei dataset di allenamento: definizione delle categorie di classificatori da utilizzare; operazioni preliminari di preprocessing. Procedura di estrazione del classificatore forte. Procedura di estrazione del miglior classificatore debole.

Adaboost è un algoritmo di apprendimento automatico della famiglia degli *ensamble learner*. Un algoritmo di ensamble learning mira alla costruzione di un classificatore composto da un insieme di

Adaboost è un *meta algoritmo* di *machine learning* che viene utilizzato per aumentare le prestazioni di altri algoritmi di apprendimento. Ciò avviene estraendo un *classificatore forte* a partire da un insieme di *classificatori deboli*, dove per classificatore si intende una funzione $f(x)$ che identifichi la *classe* di appartenenza dell'oggetto x in input.

La realtà d'interesse del problema prevede l'esistenza di due classi di oggetti: *umano* e *non umano*.

I dati in input per la fase di allenamento sono costituiti dal *dataset di allenamento*, ovvero una raccolta di oggetti preclassificati attraverso i quali Adaboost riuscirà a selezionare una combinazione di classificatori deboli che meglio approssimano la classificazione degli elementi del dataset.

4.0.1 Il dataset dall'allenamento

La finestra di visualizzazione, come accennato in precedenza è di 512×424 pixel. In una registrazione che riprende dall'alto una persona che attraversa la stanza camminando, la figura della persona occupa una porzione della finestra di circa 160×100 pixel.

Un dataset di allenamento non è altro che un insieme di porzioni di frame di una certa dimensione (la stessa per tutti) che vegono classificati manualmente dal creatore[1] del dataset.

4.0.2 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*

[1] Il sistema di apprendimento che si sta trattando ricade nella categoria dei sistemi di *apprendimento supervisionato*.

(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[2]:

$$D = P \cup N \tag{4.1}$$

$$P \cap N = \emptyset \tag{4.2}$$

$$\#(D) = \#(P \cup N) = \#(P) + \#(N) \tag{4.3}$$

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:

$$h(x) = \begin{cases} 1 & \text{se } pf(x) < p\theta \\ 0 & \text{altrimenti} \end{cases} \tag{4.4}$$

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 4.0.3, ma si tenga presente che il miglior classificatore è quello il cui *errore pesato* è minimo per la corrente iterazione.

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

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

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:

$$F(x) = \begin{cases} 1 & \text{se } \sum_{t=1}^{T} \alpha_t h_t(x) > \theta \sum_{t=1}^{T} \alpha_t \\ 0 & \text{altrimenti} \end{cases} \tag{4.5}$$

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.

4.0.3 Il miglior 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:

$$h(x) := \begin{cases} 1 & \text{se } pf(x) < p\theta \\ 0 & \text{altrimenti} \end{cases} \tag{4.6}$$

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 $values[i]$[3]. 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 (4.6) 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 (4.6) con $p = 1$, la quantità $S^- + (T^+ - S^+)$ è *l'errore pesato*;
6. Analogamente alle osservazioni 4 e 5, un classificatore (4.6) 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:

$$p = \begin{cases} 1 & \text{se } S^- + (T^+ - S^+) < S^+ + (T^- - S^-) \\ -1 & \text{altrimenti} \end{cases} \tag{4.7}$$

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

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 $\Theta(kn \log n)$.

Capitolo 5

Tuning

Definizione dei criteri di valutazione delle prestazioni. Algoritmo di massimizzazione dell'accuratezza. Valutazione del risultato dell'algoritmo di apprendimento. Valutazione degli elementi di disturbo.

Capitolo 6

Rilevamento

Presentazione della tecnica di rilevamento su registrazioni reali. Algoritmo evoluto di selezione della finestra ottima. Tecniche di ottimizzazione dell'operazione di rilevamento a regime. Presentazione e confronto dei risultati ottenuti con quelli in letteratura.

Capitolo 7

Conclusioni

Appendici

Appendice A

Struttura Software

Appendice B

Kinect

Ad onor del vero, ciò che viene comunemente chiamato *sensore di profondità* (o sensore di distanza) del Kinect, è in realtà uno *scanner 3D a luce strutturata*[1]. Un sorgente di raggi infrarossi proietta una serie di pattern codificati. La deformazione indotta dalle superfici degli oggetti interessati viene acquisita da una o più telecamere ed utilizzata per il calcolo delle coordinate tridimensionali.

Il risultato di un sensore di questo tipo è un insieme di triplette (x, y, z), organizzate in una *immagine di profondità*, una struttura dati che è molto simile ad una semplice immagine in scala dei grigi, dove il valore di ogni pixel rappresenta la misura in millimetri della distanza della superficie dal sensore. La forte somiglianza con le immagini in scala dei grigi è supportata dal fatto che ogni pixel è codificato utilizzando 16bit.

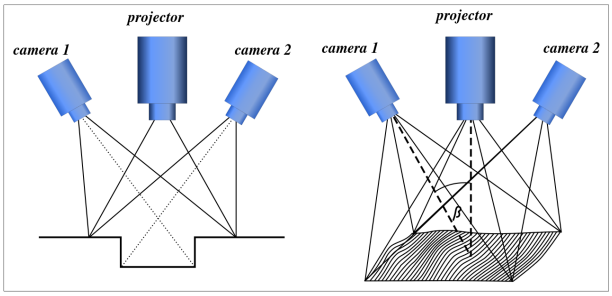

Figura B.1: Schematizzazione di uno scanner 3D a luce strutturata.

La massima affidabilità del sensore del Kinect 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. Il Kinect V2 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] Nonostante ciò, lo si continuerà a chiamare *sensore di profondità*, leggermente inesatto, ma decisamente più breve ed intuitivo

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.