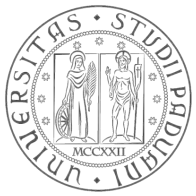


Università degli Studi di Padova

Facoltà di Ingegneria Informatica

Corso di Laurea Magistrale in Ingegneria Informatica

Riconoscimento facciale real time e robusto al cambio di posa da immagini tridimensionali

Laureando

Marco Ciacco

Relatore

Prof. Emanuele Menegatti

Co-relatore

Ing. Marco Carraro

Data di laurea

10/04/2017

Anno Accademico 2016/2017

Vorrei dedicare questa tesi a tutti coloro che mi hanno sostenuto emotivamente ed economicamente in tutto il mio percorso di studi.
La mia famiglia in primis, che non mi ha mai fatto mancare nulla e spronato a dare il meglio di me.
A mio fratello e a Jasmine che si sono prodigati a correggermi la tesi.
Agli amici che hanno sopportato i miei alti e bassi (sopratutto bassi nell'ultimo periodo) e che mi hanno sostenuto con messaggi, uscite e sorrisi.

Indice

Sommario

Lo scopo della tesi è di sfruttare le informazioni tridimensionali in un software di riconoscimento facciale, partendo da fotografie acquisite tramite Microsoft™Kinect di 2^a generazione. Si sono sfruttate le librerie di OpenCV e PCL (Point Cloud Library). La tesi è divisa in due fasi: la prima in cui si prepara l'immagine al riconoscimento, e la seconda dove avviene l'estrazione delle feature e il confronto tra volti. La prima fase comprende l'identificazione del volto e la correzione della posa. L'identificazione del volto è stata sviluppata creando un algoritmo ad hoc che, sfruttando un algoritmo di clustering e un classificatore, identifica l'ubicazione del volto. La correzione della posa si è eseguita applicando una rototraslazione al volto. Quest'ultima viene calcolata tramite un processo che sfrutta i keypoint ottenuti tramite la libreria *OpenFace*. La fase di identificazione prevede la divisione del volto in aree regolari, l'estrazione delle feature e la loro analisi. Le feature utilizzate sono calcolate principalmente sulle posizioni x,y e z dei punti che compongono il volto senza sfruttare le informazioni sul colore. Dai test effettuati, il classificatore 3D ha ottenuto dei buoni risultati con il 93% di identificazioni corrette.

Abstract

The objective of my thesis is to erecognize faces exploiting the 3D data information gathered from a Microsoft™Kinect second generation. We use a combination of two different libraries: OpenCV and PCL (Point Cloud Library). This thesis is split into two parts, in the first part the image is prepared for analysis while the second focuses on feature extraction and comparison of the faces. The first phase is about face detection and pose correction. Face identification was developed by creating a specific algorithm which uses a clustering and classificator method. The pose correction is excuted by applying a roto-translation of the face. This transformation was calculated using the keypoints found by the OpenFace open-source library. *OpenFace*. The identification phase is made up of three parts; splitting the face into small parts, feature extraction and analysis of results. The used feature is mainly calculated by using the position x, y and z of the point that belonged to the face without using colour information. From carrying out relevant tests the 3D classifier has obtained a 93% success rate.

Capitolo 1

Introduzione

La tesi vuol proporre un approccio alternativo al riconoscimento facciale classico. La letteratura attuale si concentra molto su riconoscimento di volti a partire da foto in due dimensioni[1], questo perché attualmente i dispositivi che acquisiscono immagini in due dimensioni sono più diffusi ed economici (cellulari, telecamere ecc..). Dal recente rilascio di dispositivi economici come la Microsoft^{TM}Kinect e ASUS Xtion è possibile acquisire dati tridimensionali a basso costo. Benché la tecnologia non abbia raggiunto una versatilità ed affidabilità pari a quello delle fotocamere, è già possibile avere un livello di dettaglio sufficiente da consentire di sfruttarne i dati[2].

Ne sono un esempio il il People detection and tracking[3][4] e la correzione della posa per il People Detection[5].

In questa tesi utilizzeremo le informazioni 3D per irrobustire i risultati ottenuti dai classici algoritmi di riconoscimento facciale.
Il lavoro si è basato su un dataset di volti acquisiti con il dispositivo Microsoft^{TM}Kinect di 2^a generazione che, oltre a fornire una immagine tradizionale, fornisce anche la distanza di ogni pixel dal sensore. Sfruttando queste informazioni aggiuntive si sono estratte feature (ovvero caratteristiche del volto) che non calcolabili da una immagine bidimensionale. Verranno studiate queste caratteristiche e la loro utilità nel riconoscimento e identificazione degli individui.

Questo tipo di approccio non risente dei classici problemi che affliggono i riconoscimenti bidimensionali quali l'illuminazione, la prospettiva e la posa dato che si basano sulle posizioni reali dei punti nell'immagine e non una loro proiezione.

Le informazioni che si andranno ad estrarre potranno essere affiancate a quelle già note in letteratura per aumentare la confidenza dei classificatori preesistenti.

Capitolo 2

Il Dataset

2.1 Le foto

In questa tesi si analizzeranno foto 3D scattate ad individui posizionati a 1-2 metri dall'obiettivo della Kinect in un'aula dell'Università di Padova. Si è deciso di scegliere solo le foto scattate frontalmente in quanto più ricche di informazioni e più facilmente trattabili.

2.1.1 La Kinect

Il dispositivo di acquisizione delle immagini utilizzato è Microsoft^{TM}Kinect di 2^a generazione (figura 2.2).

Microsoft Kinect, inizialmente conosciuto come Project Natal, è un accessorio originariamente pensato per Xbox 360, sensibile al movimento del corpo umano. Kinect è dotato di telecamera RGB, doppio sensore di profondità a raggi infrarossi composto da due scanner laser a infrarossi e da una telecamera sensibile alla stessa banda. La telecamera RGB ha una risoluzione full HD 1920×1080 pixel, mentre quella a infrarossi usa una matrice di 512×424 pixel. [6]

2.1.2 Imprecisioni nelle foto

Nella figura 2.1 vediamo due rappresentazioni della medesima foto, scattata con la Kinect, una 3D e una 2D. Si può notare che nella rappresentazione 3D sono presenti errori:alcuni pixel non sono rappresentati, mentre altri si trovano posizionati in luoghi errati (vedi anche figura 2.3). Questo è dovuto in parte al rumore dei sensori, e in parte al fatto che, come si vede in figura 2.2, i sensori per la profondità sono distanti dalla fotocamera, quindi acquisiscono l'immagine da un angolazione leggermente diversa. Questo spiega la presenza

Figura 2.1: Esempio di una foto 2D e 3D Scattata con la Kinect

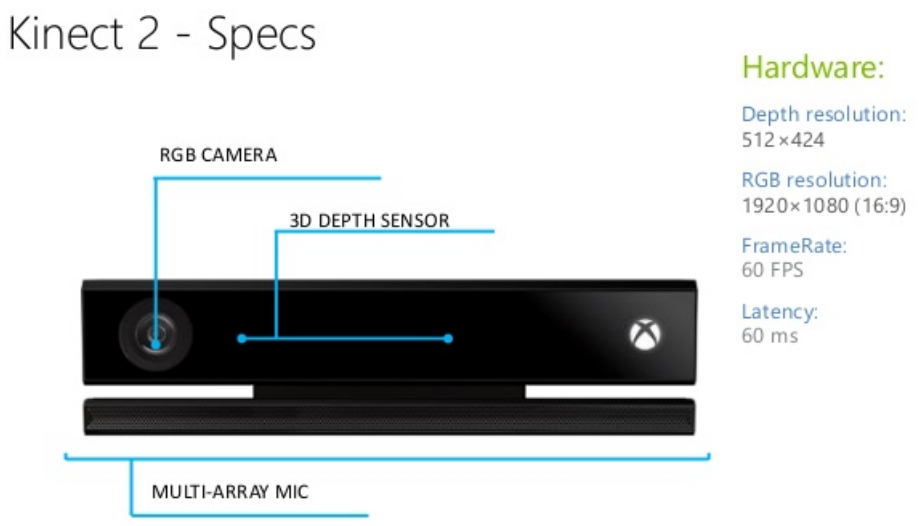


Figura 2.2: Microsoft^{TM}Kinect di 2^{a} generazione

Figura 2.3: Rappresentazione grafica della profondità di un immagine, le tonalità di grigio indicano la vicinanza dei pixel al sensore

di aree in cui è assente l'informazione sulla distanza (ad esempio a destra del volto) e aree in cui l'informazione è doppia.

2.2 Composizione

Il dataset fornito contiene i volti di 27 persone in diverse pose, a 1 e 2 metri di distanza dal sensore, in diverse condizioni luminose e con altri volti e persone presenti sullo sfondo. I file sono nominati nel seguente formato: *nnn_xx_cloud.pcd* e *nnn_xx_image.png* dove nnn è un numero progressivo che identifica la persona visualizzata e xx è un numero che identifica il tipo di posa assunta dal soggetto al momento dello scatto della foto. I numeri xx corrispondo alla condizione di scatto identificati nella tabella seguente.

	Illuminazione buona	Illuminazione Scarsa	Distanza dall'obiettivo maggiore
Volto frontale	00	06	12
Volto rivolto a destra	01	07	-
Volto rivolto a sinistra	02	08	-
Volto rivolto in alto	03	09	-
Volto rivolto in basso	04	10	-
Volto sorridente	05	11	-

2.3 PCL Point Cloud Library

Per ogni foto del dataset sono presenti due file contenenti un'immagine in formato jpeg 2D e una in formato .pcd (vedi 2.1). Per aprire il secondo formato è necessario aver installato **Point Cloud Library PCL**. PCL è uno standalone open project per il processamento di immagini 2D/3D [7]. Questa libreria consente di visualizzare immagini 3D, elaborarle, estrarre feature, ricostruzione delle superfici ecc..

È stata scritta in C++ e rilasciata in licenza BSD [1]. Questa libreria si basa sull'analisi delle Point Cloud.

2.3.1 Point Cloud

Una Point Cloud è un insieme di punti caratterizzati dalla loro posizione in un sistema di coordinate e da eventuali valori di intensità (colore, profondità ecc..) ad essi associati.
Le nuvole di punti sono spesso usate come rappresentazioni di strutture tridimensionali, come oggetti o superfici in rilievo (quali, ad esempio, la superficie terrestre).[8]

Le Point Cloud che si andranno a esaminare sono salvate in file con estensione *.pcd*. Sono state acquisite utilizzando la Kinect e quindi presentano le seguenti caratteristiche:

[1] La licenza BSD è un tipo di licenza che si applica ad un software che impone restrizioni minime

- **Formato dati XYZRGB**: Le Point Cloud possono essere di vari tipi, questo in particolare contiene informazioni sul colore (RGB) e sulla posizione (XYZ) di tutti i pixel
- **Formato dati strutturato**: La foto mantiene una corrispondenza diretta tra l'immagine 2D e quella 3D, ovvero è possibile facilmente passare dalle coordinate del pixel dell'immagine bidimensionale a quella tridimensionale.
- **Punti NAN**: Come spiegato in 2.1.2, l'immagine ha dei pixel presi dalla fotocamera che non hanno informazioni sulla profondità e quindi non possono essere rappresentati nel 3D ma, le informazioni sul colore e sul posizionamento 2D sono comunque contenute nel file *pcd*

2.3.2 I vantaggi delle immagini 3D

Rispetto alle foto bidimensionali, come già detto, avere l'informazione sulla distanza dal sensore è molto utile per diverse ragioni.

Prospettiva: Nei metodi classici di riconoscimento facciale, la distanza dall'obiettivo influisce sul calcolo delle distanze, infatti spesso si utilizzano feature che si basano sul rapporto di distanze tra punti[9]. Nel 3D invece, i punti hanno dimensione reale e la distanza tra due punti è indipendente dalla distanza dell'obiettivo.

Illuminazione: Le immagini 2D con scarsa illuminazione non sono facilmente analizzabili. Le informazioni prese dalla Kinect sfruttano un sensore a che lavora nella frequenza dell'infrarosso e pertanto rende possibile analizzarle anche in assenza di luce.

Clustering: Gli algoritmi di clustering[2] di immagini bidimensionali si basano esclusivamente sui dati RGB dell'immagine (colori e luminosità), in quelle tridimensionali invece è possibile sfruttare algoritmi basati sulla distanza tra gli oggetti, quindi più accurato e talvolta anche più veloce.

[2]Il Clustering è un processo che permette di suddividere un insieme di oggetti in K gruppi sulla base dei loro attributi

Capitolo 3

Preparazione dell'immagine

In questo capitolo verrà descritto come si è proceduto alla preparazione dell'immagine, in modo da renderla pronta per il riconoscimento vero e proprio. La preparazione dell'immagine consiste in:

- Individuazione del volto/ dei volti (face detection)
- Isolamento del volto
- Stima della posa
- Raddrizzamento del volto

3.1 Ambiente di sviluppo

Il software è stato realizzato in ambiente **Linux**, la distribuzione scelta è **Ubuntu** versione 14.04, scritto in **C++** e sfrutta le librerie OpenSource:

- **PCL - Point Cloud Library** v 1.8 (descritta in 2.3)
- **Open Face** v 0.2.1
- **OpenCV** v 3.1.0

Si è scelto questa distribuzione e versione di Linux per semplificare l'installazione delle varie librerie (vedi Appendice A).

Il computer utilizzato per l'esperimento è un laptop della Samsung R509 che monta un hardware con le seguenti caratteristiche:

- **Processore** Intel Pentium Dual-Core T3200, 2.0 GHz, 1MB cache L2, FSB 667 MH
- **Memoria** 4096 MB DDR2
- **Scheda Video** Intel Graphics Media Accelerator (GMA) X4500HD

3.2 Individuazione del volto

Individuare il volto in un'immagine 2D è un problema ampiamente studiato e risolto in letteratura. Per individuare il volto si è scelto di utilizzare la libreria **OpenFace**

3.2.1 OpenFace

OpenFace è un implementazione in Python e Torch di riconoscimento facciale basata su una rete neurale profonda, ispirandosi alla pubblicazione *CVPR 2015 FaceNet* in cui si descrive un metodo che unisce il Face Recognition e Clustering. A cura di Florian Schroff, Dmitry Kalenichenko, e James Philbin a Google [1] [10].

OpenFace per il riconoscimento dei volti sfrutta un algoritmo appartenente alla libreria **Dlib**[1] che a sua volta sfrutta un **HOG Classifier**[11]. Un classificatore **HOG** è un classificatore che si basa su un SVM[2] allenato su un dataset. In questo lavoro, usiamo un classificatore fornito da OpenFace già allenato e pronto all'uso. Testando l'algoritmo si è notato che il tempo che impiega ad analizzare completamente una foto in fullHD (1920×1080) è elevato, usando la macchina a disposizione (vedi 3.1) impiega intorno ai 10 secondi.

[1] **Dlib** è un toolkit C++ toolkit contenente algoritmi di machine learning e strumenti per creare software complesso in c++ per risolvere problemi reali. Dlib è open source ed è quindi possibile utilizzarla in qualunque progetto

[2] SVM ovver(Support Vector Machine) è una tecnica di Machine Learning per la regressione e la classificazione di pattern, sviluppati negli anni '90 da Vladimir Vapnik ed il suo team presso i laboratori Bell della AT&T.

3.2.2 Riconoscimento volti dal 3D

Per velocizzare il riconoscimento dei volti si è pensato di sfruttare le informazioni contenute nell'immagine 3D. Guardando l'immagine 2.3 si possono facilmente individuare i volti contenuti.

Miglioramento della foto: Come già detto in 2.1.2, a causa della distanza tra il sensore di profondità e la fotocamera, la foto presenta degli errori e delle mancanze dovute alla doppia informazione sulla profondità in alcuni punti e la mancanza di esse per altri. Nel caso in cui l'informazione sia doppia è stata trovata una soluzione. Si è notato che questi punti sono spesso isolati e circondati da pixel che hanno la giusta informazione sulla profondità, quindi esaminando i circostanti è possibile individuarli e correggerne la profondità esaminando gli stessi.

Per ogni pixel appartenente alla foto con coordinate bidimensionali x,y si è calcolata la media e la varianza del valore della coordinata z per ogni pixel appartenente al quadrato di lato 2 costruito nel suo intorno (ovvero tutti i punti appartenenti alle coordinate comprese tra x-2 e x+2 e y-2 e y+2) Se il valore di z del punto x,y non rispetta le seguenti caratteristiche

$$\bar{z} - 2\sigma_z < z_{x,y} < \bar{z} + 2\sigma_z \tag{3.1}$$

dove $\bar{z}$ è la media dei valori della coordinata z dei punti appartenenti al quadrato intorno al punto x,y e σ_z è la varianza della coordinata z negli stessi. Se tale condizione è verificata allora il punto è da considerarsi in posizione errata, si correggono quindi le coordinate x,y e z dello stesso facendo la media tra i punti del suo intorno che rispettano la 3.1 Questa operazione ha complessità computazionale $O(n * k^2)$ dove k è la dimensione del quadrato dell'intorno. Si possono ridurre le dimensioni o addirittura saltare questo passaggio per aumentare la velocità di esecuzione.

Riduzione dell'immagine: Una volta migliorata l'immagine, prima di passare al punto successivo, riduciamo il numero dei punti per velocizzare l'operazione di clustering che avrà luogo nel punto successivo. Per farlo creiamo una copia dell'immagine ridotta di un fattore k. Questa operazione è $O(n/(k^2))$.

Clustering dell'immagine: Sfruttando le librerie di PCL si effettua un clustering euclideo sull'immagine 3D. Questa tecnica raggruppa i punti in base alla distanza euclidea che li separa. Si sono impostati i valori di:

- **Tolleranza:** Ovvero la massima distanza che possono avere due punti per essere considerati parte dello stesso cluster, nel caso in esame si è scelto 0.1 metri;

- **Dimensione massima e minima del cluster:** Dimensione massima e minima, in termini di numero di pixel, che i vari cluster possono avere, si è scelto un range tra 25 e i 25000 punti;

- **Metodo di ricerca:** Il metodo e la struttura dati utilizzati per organizzare i punti in modo da fare una ricerca efficiente. Nel nostro caso **K-D Tree**[3][1].

A questo punto si è partizionata l'immagine ridotta. Si deve quindi riportarla alla dimensione originale, aggiungendo i punti dell'intorno di ogni pixel appartenente al cluster e selezionando solo quelli che rispettano questa condizione

$$z_{x,y} - tolleranza < z_{a,b} < z_{x,y} + tolleranza \tag{3.2}$$

dove $z_{a,b}$ è la coordinata z di un pixel dell'intorno del punto alle coordinate x,y. $z_{x,y}$ è invece il valore della coordinata z del punto di riferimento appartenente al cluster di punti ridotto. *tolleranza* invece è lo stesso valore di tolleranza visto prima, ovvero 0.1. Dopo questa fase la PointCloud si presenta come in figura 3.1.

Figura 3.1: Immagine Clusterizzata

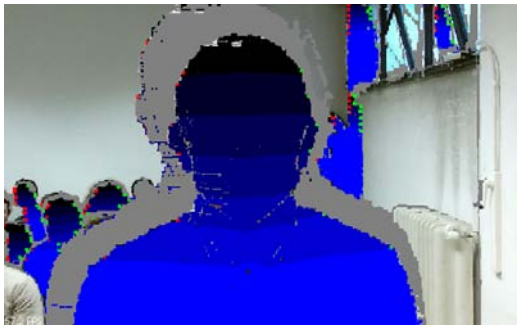
[3]Un K-D tree (abbreviazione di albero a k dimensioni) è una partizione spaziale di una struttura dati per organizzarli in uno spazio k-dimensionale.

Scelta dei Cluster: Tra tutti i cluster ora si provvederà a scegliere soltanto quelli che hanno maggiori probabilità di contenere un volto. L'algoritmo sviluppato sfrutterà il fatto che le dimensioni dei volti sono reali e non dipendenti dalla prospettiva. Innanzitutto si è controllato che ogni cluster di punti rispetti queste caratteristiche:

- profondità massima inferiore ai 0.8 metri
- larghezza compresa tra 0.15 e 1.2 metri.

I cluster che superano anche questa condizione vengono filtrati in base alla forma della parte superiore. La forma della testa umana è ovoidale, la cui altezza è compresa tra i 20 e i 30 cm. Si prendono quindi i 30 cm del cluster partendo dall'alto, li si partiziona in 6 gruppi (sempre basandosi sulla coordinata y) e per ogni partizione si prende la coppia di punti con la coordinata x maggiore e minore. Nella figura si possono notare le varie aree colorate con una diversa gradazione di blu, mentre i punti verdi e rossi rappresentano i minimi e i massimi di coordinata x di ogni partizione (vedi figura 3.2).

Figura 3.2: Cluster di volto, diviso in base alle coordinate dell'asse y

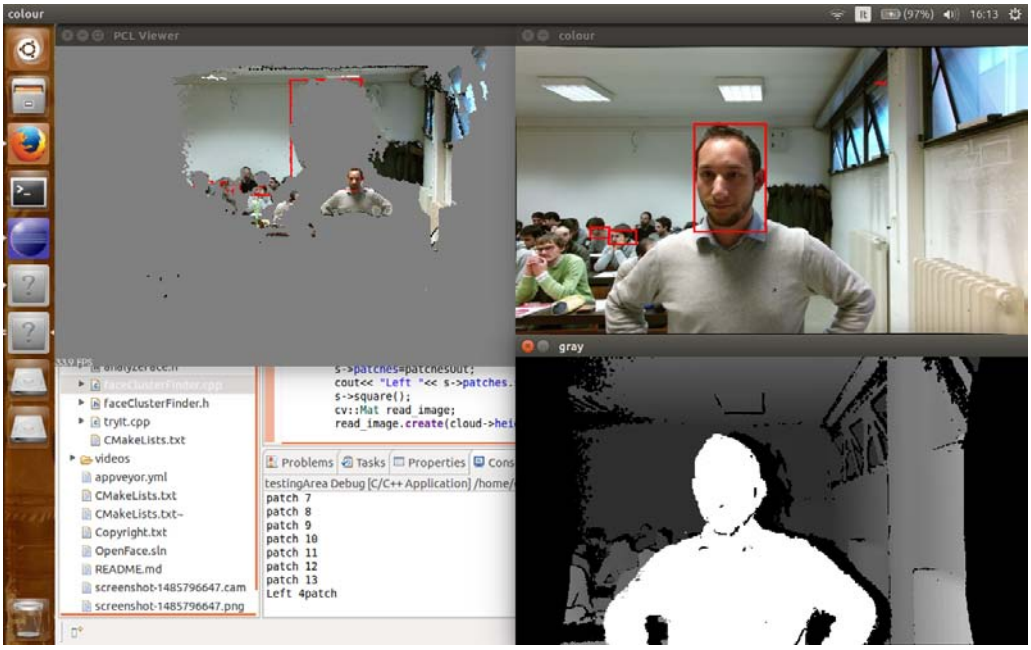


Se i minimi e i massimi della figura hanno un andamento prima crescente e poi decrescente e coerente con le dimensioni di un volto, si hanno buone probabilità di ottenere così solo i volti. In base all'andamento della forma inoltre si deduce anche la grandezza del volto.

3.3 Raddrizzamento del volto

Si hanno ora a disposizione una serie di cluster contenenti i volti trovati dall'algoritmo (come mostrato in figura 3.3). Si utilizza poi l'algoritmo HOG di OpenFace per effettuare un ulteriore filtraggio e migliorare la precisione del

Figura 3.3: Volti trovati con l'algoritmo di analisi delle forme 3D



riquadro. Computazionalmente si è così passati da un algoritmo che impiega circa 10 secondi a uno che ne impiega 1-2.

Una volta trovato il volto, OpenFace è in grado di trovare le posizioni del naso, degli occhi, sopracciglia e contorno del viso (vedi 3.4).

OpenFace funziona molto bene in foto prese frontalmente e anche con distanze di 1 o 2 metri dall'obiettivo, fatica invece con immagini prese di profilo.

3.3.1 Miglioramento della precisione dei punti del naso

Per effettuare un buon allineamento è necessario che i punti del naso abbiano una buona precisione, ecco perché si è speso del tempo per migliorarli. OpenFace spesso trova i punti con buona precisione e quando sbaglia, lo fa di pochi pixel. Si può quindi migliorare la precisione andando ad esaminare i punti nell'intorno di quelli già trovati.

Algoritmo di miglioramento dei singoli punti: I punti che si andranno a migliorare sono 4 e sono situati sul dorso del naso (i quattro punti blu nella figura 3.4).

L'algoritmo sviluppato, uguale per tutti e 4 i punti, sfrutta l'informazione sulla profondità per migliorarne la precisione. Per ognuno si prendono tutti i

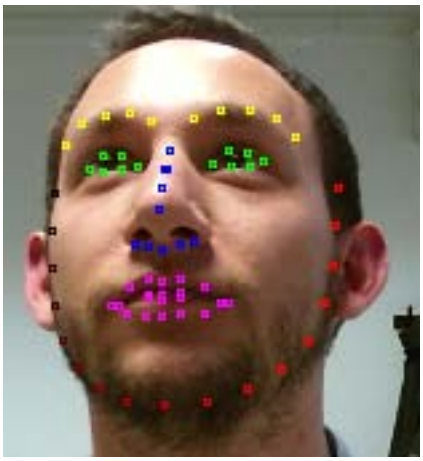


Figura 3.4: Volto analizzato con OpenFace

punti, appartenenti all'immagine, che risiedono sulla stessa coordinata y del punto scelto (vedi Fig 3.5).

Di questi punti, se si prende in considerazione la coordinata z, si ottiene un grafico come quello visualizzato in figura 3.6b. Il nostro obiettivo è trovare il punto di massimo più vicino a quello già trovato da OpenFace. Essendo immagini prese con una Kinect, le informazioni sulla profondità sono rumorose(fig 3.6a e 3.6b), di conseguenza si è proceduto a rimuoverle sostituendo ogni punto con la media punti dei 10 che lo circondano (fig 3.6c). Infine si è provveduto ad effettuare la derivata prima di tali valori per trovare il minimo della funzione (fig 3.6d).

Una volta ottenuto il punto di minimo della funzione, corrispondente al punto in cui la derivata raggiunge il valore zero, si effettua un ulteriore aggiustamento. Il punto cercato infatti è precisamente sul minimo solo e soltanto se il volto è rivolto frontalmente verso l'individuo, nel caso di posa differente si corregge il punto selezionato di una quantità di pixel proporzionale alla rotazione del volto.

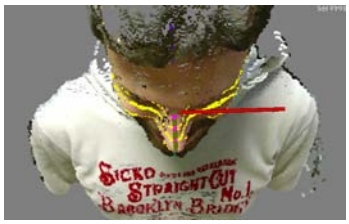
Allineamento dei punti: dai punti trovati, si ricavano i parametri dell'equazione di una retta (3.3) che ne approssima l'andamento tramite una regressione lineare.

$$y = ax + b \tag{3.3}$$

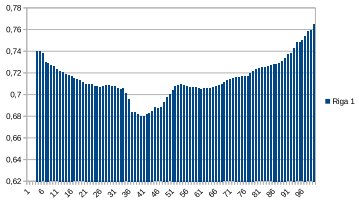
$$a = \frac{\sum_i y_i \sum_i x_i^2 - \sum_i x_i \sum_i x_i y_i}{N \sum_i x_i^2 - (\sum_i x_i)^2} \tag{3.4}$$

$$b = \frac{N \sum_i x_i y_i - \sum_i x_i \sum_i y_i}{N \sum_i x_i^2 - (\sum_i x_i)^2} \tag{3.5}$$

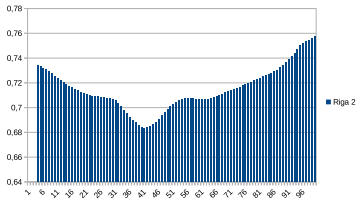
Figura 3.5: Fascia di punti dove avviene il miglioramento dei punti del naso



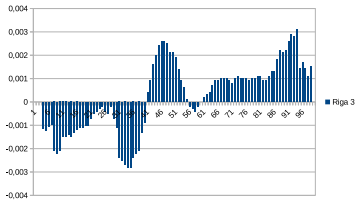
(a) Fascia dove avviene il miglioramento dei punti del naso vista dall'alto



(b) Rappresentazione grafica dei punti evidenziati in giallo nella figura 3.6a



(c) Grafico ottenuto fancendo la media dei valori rappresentati in 3.6b



(d) Derivata dei valori rappresentati in 3.6c

Figura 3.6: Miglioramento dei punti del naso

I parametri a e b vengono calcolati tramite le equazioni (3.4) e (3.5) e i valori x e y corrispondono alle coordinate bidimensionali dei 4 punti del naso [12].

Una volta calcolata la retta, si impone ai 4 punti di appartenere alla retta in modo da allinearli.

3.3.2 Preallineamento

Traslazione: sfruttando la posizione della radice del naso, calcoliamo la funzione di traslazione del volto dalla posizione iniziale all'origine degli assi.

Rotazione: in seguito, sfruttando i parametri della retta (3.3) si calcola una funzione di traslazione che imponga alla stessa di traslare ed essere parallela all'asse y.

3.3.3 Allineamento finale

A questo punto si ha un volto sicuramente rivolto verso la telecamera, ma potrebbe guardare verso il basso o l'alto.

Si è deciso di utilizzare un algoritmo di *Pairwise Registration* [13] il quale permette a diverse PointCloud di allinearsi in modo da avere un'immagine 3D completa e presa da diverse angolazioni. Questo algoritmo sfrutta l'*Iterative Closest Point* per allineare due PointCloud con abbastanza punti in comune. Sfruttando questa parte dell'algoritmo si allinea il volto in esame, con uno di riferimento già allineato.

Perché questo abbia successo è importante che il volto sia già preallinato e vicino a quello di riferimento, cosa fatta nella sezione 3.3.2.

In figura 3.7 si può notare il progressivo allineamento dell'immagine data (in rosso) con quella di riferimento (in verde).

Una volta completato questo passaggio, si ha il volto perfettamente allineato. In ultimo, basandosi sui keypoint del volto, si sono rimossi tutti i punti che non appartengono al volto della persona come il busto e il collo. Si ha ora l'immagine di un viso rivolto verso la telecamera, senza sfondo, pronto per essere riconosciuto.

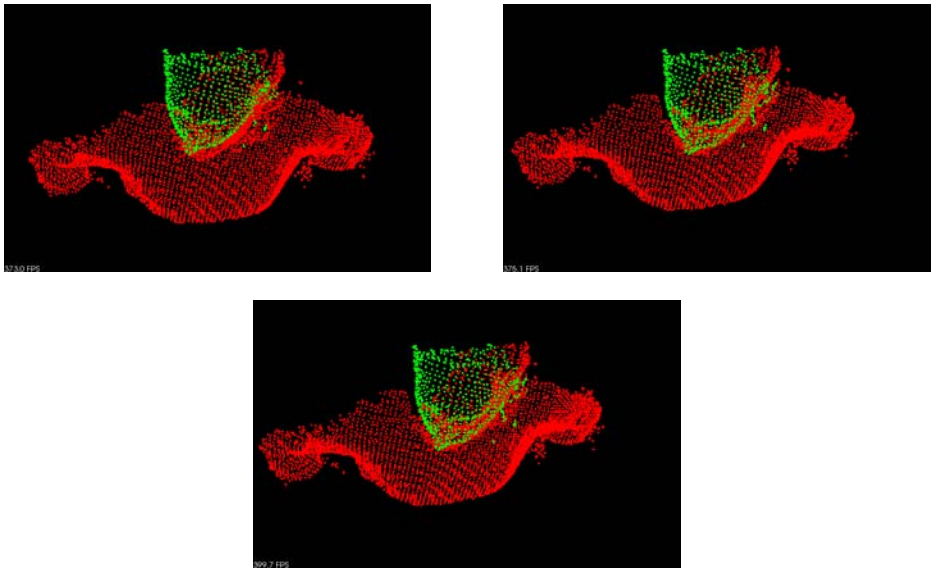


Figura 3.7: Fasi di allineamento del volto tramite Iterative Closest Point

Figura 3.8: Immagine originale e con faccia allineata

Figura 3.9: Immagine di partenza

Figura 3.10: Risultato finale con rimozione dello sfondo e correzione della posa

Capitolo 4

Estrazione delle feature

Per effettuare il riconoscimento facciale è necessario estrarre delle feature del volto, che sono caratteristiche misurabili, come la lunghezza del naso o la distanza dagli occhi.

Nei capitoli precedenti si è visto come ottenere, da un immagine 3D completa, solo la rappresentazione del volto rivolta verso la telecamera con la radice nel naso traslata nell'origine degli assi.

Avendo sempre immagini allineate allo stesso modo, si è pensato di dividere il volto in aree uguali in base alla loro posizione tridimensionale.

4.1 Divisione del volto in piccole aree/rettangoli

Effettuare il riconoscimento su ogni punto del volto risulterebbe troppo complesso dal punto di vista computazionale, è quindi necessario ridurre il numero di punti. Sebbene PCL sia già provvisto di metodi per filtrare e ridurre il numero di punti da analizzare, questi non si adattano bene alla situazione in oggetto. L'algoritmo che si andrà ad utilizzare sfrutta il fatto che ogni punto del viso da esaminare risiede sempre nella stessa posizioni a prescindere dalla posa dell'individuo. Si é diviso il volto in rettangoli di dimensioni il più possibile costanti. Si é convenuto che la forma geometrica semplice che più assomigliasse ad un volto fosse il Cilindro.

- Viene creato un cilindro immaginario con con base di raggio 10cm e centrato nel punto 0,0,-0.1
- Viene proiettato il volto sul cilindro
- Viene diviso il cilindro in aree rettangolari

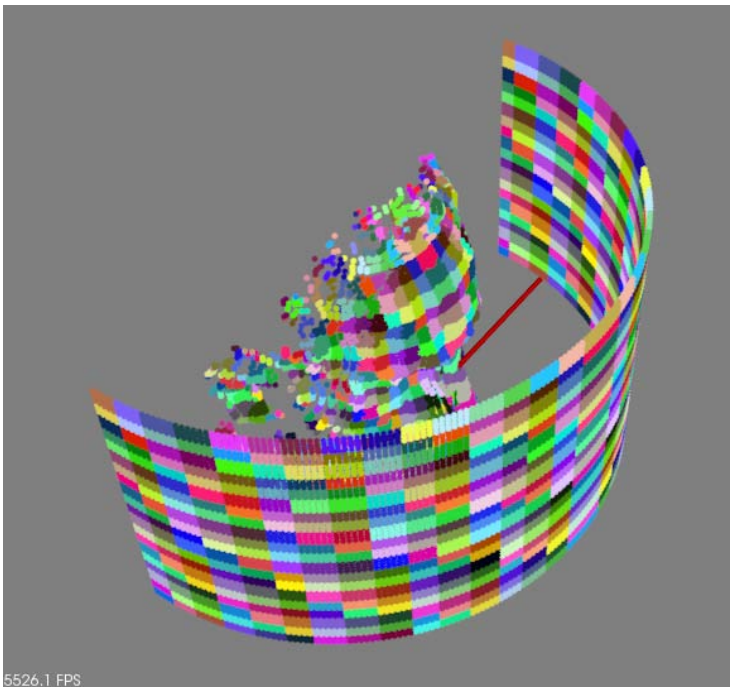


Figura 4.1: Nella foto si può vedere la suddivisione in rettangoli del viso, e la sua proiezione su un cilindro di raggio 20 cm avente centro in 0,0,-0.1 (metri)

- Vengono raggruppati i punti del volto che insistono nella stessa area rettangolare.

Il risultato dell'operazione lo si può vedere nella figura 4.1

4.2 Estrazione delle maschere

Per ogni area rettangolare ricavata, si genera un punto che abbia le seguenti caratteristiche:

- Deve giacere nella semiretta che abbia queste caratteristiche
  - deve essere parallela al piano XZ
  - deve passare per il punto X=0, Z=0
  - deve passare per il centro dell'area rettangolare corrente
- La distanza dall'asse del cilindro deve essere la media di tutte le distanze di tutti i punti appartenenti all'area rettangolare corrente

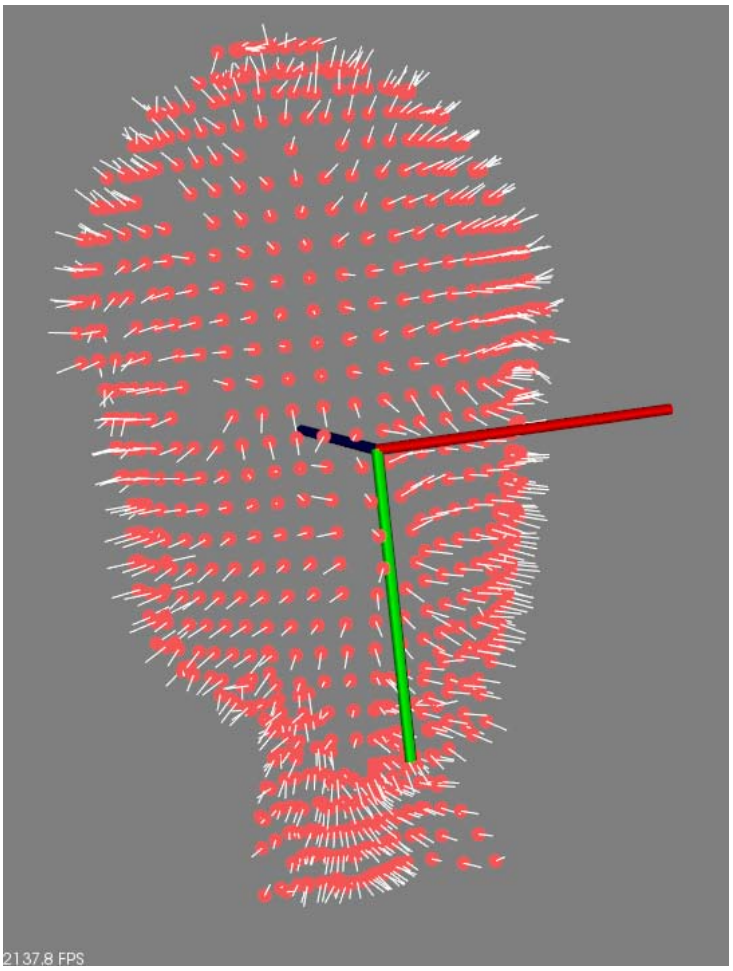


Figura 4.2: Raffigurazione di punti estratti da un volto, allineati in modo regolare all'interno del reticolo.

Per ognuno di questi punti vengono ricavate anche informazioni sulla normale del punto, calcolata semplicemente come la media delle normali dei vari punti contenuti nell'area corrente.

Questa operazione viene svolta dal metodo *posizioniMaschera()* che restituisce un array bidimensionale di punti di tipo *PointXYZRGBNormal*[1]. Definiamo **maschera** l'insieme di punti estratti con questo metodo.

[1]Una struttura di un punto che rappresenta le coordinate euclidee xyz, il colore RGB, insieme alle coordinate della normale.

4.3 Distanza euclidea

Ottenute la maschere di riferimento, una prima feature corrisponde alla posizione vera e propria di ogni punto in termini di coordinate x,y e z.

E' stata potenziata questa feature introducendo l'informazione data dalle normali.
La distanza tra le normali viene calcolata ottenendo il modulo del vettore risultante dalla sottrazione delle normali delle coppie dei 125 punti.

Di seguito è riportata la formula per il calcolo della distanza euclidea tra due punti e la differenza tra le loro normali.

$$DistanzaEuclidea = \sqrt{(a_x - b_x)^2 + (a.y - b.y)^2 + (a.z - b.z)^2}$$

$$DifferenzaNormali = \sqrt{(a.n_x - b.n_x)^2 + (a.n_y - b.n_y)^2 + (a.n_z - b.n_z)^2}$$

Questa feature basa la sua forza sull'idea che ogni volto abbia le sue dimensioni e forma. E analizzandole si può avere un delle informazioni importanti del distinguere un volto dall'altro.

4.4 Distanza tra punti

Oltre ad usare direttamente le posizioni tra i punti e confrontarne la distanza, si è pensato di effettuare un calcolo sulle distanze tra coppie di punti appartenenti alla stessa maschera.
Questa feature risente meno dei difetti di allineamento perché si tratta di una distanza tra due punti in un corpo rigido, di conseguenza in alcuni casi da migliori performance rispetto al confronto della posizione. La distanza tra i punti viene calcolata come spiegato nel capitolo 4.3, quindi tenendo conto delle normali.

4.5 Feature dai Keypoint di OpenFace

Si sono calcolate feature di distanza e posizione basate sui keypoint calcolati da Open Face (vedi Fig. 3.4). Non tutti i punti, però, sono da considerarsi affidabili, infatti OpenFace si basa sull'immagine bidimensionale per trovarli, mentre i calcoli che verranno effettuati sono basati sulle proiezioni di questi nell'immagine 3D.

Parte di queste proiezioni non va a buon fine per i motivi descritti in 2.1.2, bisogna innanzitutto individuare i punti affidabili e scartare gli altri. Per rilevare i punti errati è sufficiente controllarne la coordinata z e controllare

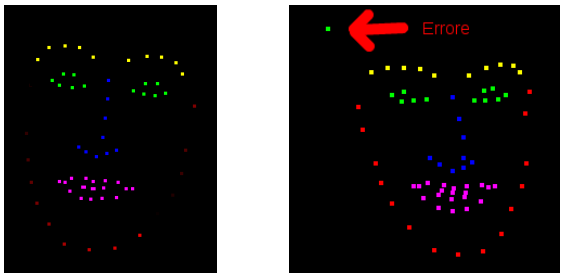


Figura 4.3: Keypoint di un volto 3D con e senza errori

se questa ne rispetta i criteri (simili a quelli visti in 2.1.2). L'algoritmo per l'analisi e il confronto delle feature che vedremo in seguito è studiato in modo da funzionare anche in assenza di alcuni punti di riferimento, quindi scartare punti non compromette il riconoscimento.

Feature classiche di riconoscimento facciale come la distanza tra gli occhi e la larghezza della bocca vengono automaticamente inserite con queste feature.

4.6 Feature di Haar 3D

Le Feature di Haar classiche sono una serie di filtri per immagini formati come sommatoria e sottrazioni di sottoparti puramente rettangolari dell'immagine stessa. Esempi di feature di Haar sono mostrati in figura 4.4. Il valore risultante del filtro è la somma dei toni di grigio dei pixel sottesi alle aree in bianco, sottratto il valore dei pixel sottesi alle aree indicate in nero. Per loro natura tali filtri possono venire efficacemente implementati usando l'immagine integrale[2] [14].

Le feature di Haar sono spesso utilizzate per l'individuazione dei volti nelle foto. In OpenCV vi è un'implementazione di un riconoscimento facciale basata su queste feature. Si basa sul fatto che immagini simili abbiano distribuzioni di chiaro/scuro simili e che quindi rispondano in modo simile ai vari filtri.

Ispirandosi a quelle bidimensionali si sono sviluppate delle feature di Haar 3D. Il principio è lo stesso, ma al posto di effettuare conti sulle gradazioni di grigio, le si effettua sulla profondità. Il pattern utilizzato è il cerchio invece

[2]L'immagine integrale è una tecnica utilizzata durante l'analisi di immagini utilizzata per velocizzare i calcoli delle immagini quando si devono testare pattern uguali lungo tutta la dimensione della foto

Figura 4.4: Esempi di pattern per le feature di Haar

del quadrato perché più facilmente utilizzabile nel 3D. I pattern scelti sono 8 e sono raffigurati in figura 4.5.

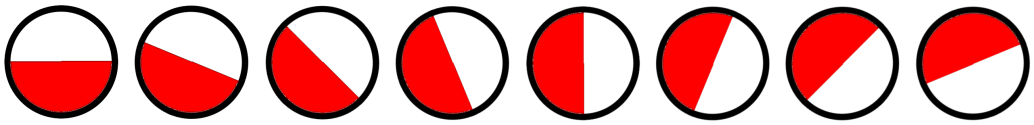


Figura 4.5: Feature di Haar 3D usate nel progetto

Il raggio del cerchio è di 5cm selezionato dopo numerose prove. Nelle feature di Haar 2D è necessario calcolare la feature per diverse grandezze, perché queste ultime risentono della prospettiva, in quelle 3D invece non è necessario.
Il calcolo della feature avviene tramite questa formula

$$\frac{\sum_{p \in parteRossa} p_z - P_z}{|p \in parteRossa|} - \frac{\sum_{p \in parteBianca} p_z - P_z}{|p \in parteBianca|} \tag{4.1}$$

dove p_z è il valore della coordinata z in un punto appartenente alla parte rossa o bianca del cerchio costruito attorno al punto P (vedi figura 4.5). Al denominatore troviamo invece la cardinalità dei punti appartenenti alla parte rossa e bianca

La figura 4.6 mostra una maschera su cui è evidenziata l'area in cui viene calcolata la feature di Haar 3D. Il punto verde è quello preso in esame, i punti colorati in rosso sono quelli la cui somma delle coordinate z verrà sottratta alla somma dei punti colorati in blu.

I risultati ottenuti danno un contributo importante per determinare l'identità dell'individuo solo perché questo è stato pre allineato. Ogni fisionomia del volto ha le sue caratteristiche e con questo metodo siamo in grado di valorizzarne le peculiarità.

Si è pensato, oltre ad utilizzare soltanto l'asse Z, anche di sfruttare le normali, calcolando la feature in questo modo:

$$\frac{\sum_{p \in parteRossa} p_{Nx} - P_{Nx}}{|p \in parteRossa|} - \frac{\sum_{p \in parteBianca} p_{Nx} - P_{Nx}}{|p \in parteBianca|} \tag{4.2}$$

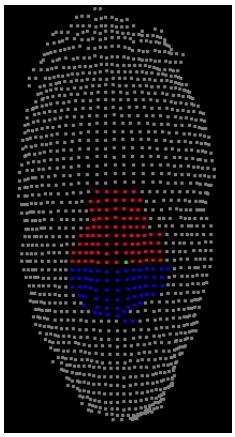


Figura 4.6: Maschera in cui sono evidenziati i punti in cui si è calcolata la feature di Haar 3D

dove p_{Nx} e P_{Nx} sono i valori x dei versore rappresentanti le normali nei punti p e P.

4.7 Feature PFH

Le PFH (Point Feature Histogram) sono delle feature il cui scopo è di codificare le proprietà geometriche dei k punti vicini. Queste sono invarianti rispetto alla posa. Sono molto resistenti al rumore e utilizzate generalmente nel riconoscimento di oggetti nel 3D.

Queste sono basate sulla relazione tra i k punti vicini a quello interessato e le loro normali.[15]

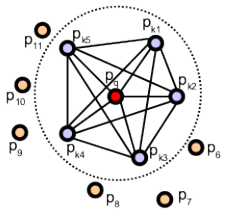


Figura 4.7: Diagramma di un calcolo della PFH del punto (p_q), colorato in rosso e posizionato alcentro del cerchio (sfera in 3D) con raggio r, e tutti i suoi k vicini (punti la cui distanza euclidea è inferiore a r)

La feature PFH è calcolata come un istogramma di relazioni tra tutte le coppie di punti nel vicinato, e la sua complessità computazionale è di $O(k^2)$.

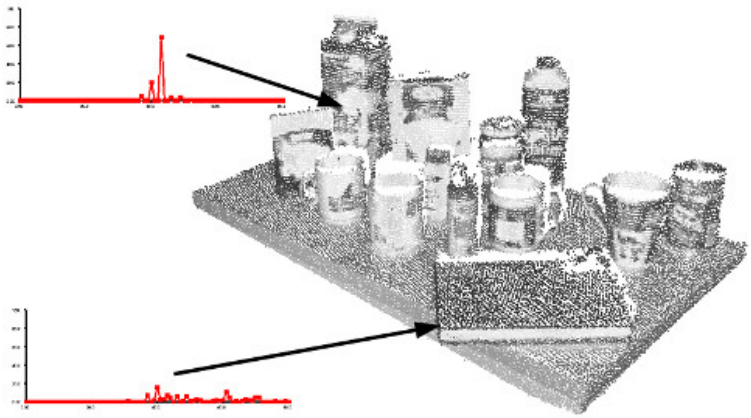


Figura 4.8: Esempio di calcolo di PFH su un immagine tridimensionale

Capitolo 5

Analisi e selezione delle feature

In questa fase si analizzeranno i dati provenienti dall'estrapolazione delle feature dalle varie foto 3D.

Per ogni maschera a disposizione, si possono estrapolare un numero elevato di feature, il confronto di queste risulterebbe molto dispendioso in termini computazionali.

Lo scopo di questa fase è di trovare le feature più utili e determinanti sia in modo da velocizzare il processo di elaborazione, sia aumentarne l'affidabilità dando maggiore peso alle feature più promettenti.

5.1 Analisi

Si é fatta una analisi statistica delle feature estrapolate nei punti, si é notato che molte sono uguali per tutti i volti, mentre alcune non rimangono costanti nemmeno in maschere diverse raffiguranti lo stesso individuo. Si è pensato quindi di concentrare i calcoli su feature significative.

5.2 Feature significative

Una feature verrà considerata **significativa** se ci aiuta a distinguere un individuo da un altro.
Perché questa condizione si verifichi è necessario che, i valori di questa, siano simili se estrapolati su foto raffiguranti il medesimo individuo e differenti, se presi da individui diversi.

5.3 Metodo di selezione

Il metodo di selezione è stato sviluppato in maniera separata e indipendente rispetto all'estrazione delle feature. Questa scelta ci consente di realizzare e ideare feature nuove e di confrontarle con quelle vecchie valutandone l'affidabilità.

5.3.1 Media e varianza

Il metodo adottato si basa su due valori probabilistici che si estraggono testando le feature nel dataset. Per ogni feature si sono fatte analisi statistiche tra i valori calcolati con immagini appartenenti allo stesso volto e con quelle relative ad immagini diverse.

L'analisi statistica consiste nel calcolare la media e la varianza di ogni feature. Si definisce $\overline{f_{interna_x}}$ la media dei valori della feature calcolate nelle varie foto raffigurante l'individuo x.

$$\overline{f_{interna_x}} = \frac{\sum_{i \in individuo_x} f_i}{|foto_{\in individuo_x}|} \tag{5.1}$$

Dove al numeratore abbiamo la somma dei valori delle feature calcolate nella foto raffigurante l'individuo x, e al denominatore il numero delle foto a disposizione per l'individuo. Si definisce anche $\overline{\sigma_{f_{interna_x}}}$ come la varianza dei valori della feature per uno stesso individuo e si calcola nel seguente modo:

$$\sigma_{f_{interna_x}} = \sqrt{\overline{f^2_{interna_x}} - (\overline{f_{interna_x}})^2} \tag{5.2}$$

Dove $\overline{f^2_{interna_x}}$ è la media dei quadrati dei valori calcolati mediante la 5.1, mentre $(\overline{f_{interna_x}})^2$ è il suo valore al quadrato.

Per avere un'analisi qualitativa della bontà della feature si è proceduto calcolando la varianza di $\overline{f_{interna_x}}$ e la media di $\sigma_{f_{interna_x}}$ tra immagini raffiguranti individui diversi.
Si è proceduto calcolando prima $\overline{f}$, ovvero la media tra i $\overline{f_{interna_x}}$, mediante:

$$\overline{f} = \frac{\sum_{x \in dataset} \overline{f_{interna_x}}}{|individui \in dataset|} \tag{5.3}$$

dove al numeratore si ha la somma di tutti i valori di $\overline{f_{interna_x}}$ calcolati in tutti gli individui $x \in dataset$ mentre al denominatore si ha il numero di individui che lo popolano (nel caso in esame 27)

Con questo valore si può calcolare la varianza di $\overline{f_{interna_x}}$ mediante la formula

$$\sigma_{\overline{f}} = \sqrt{\overline{f^2} - (\overline{f})^2} \tag{5.4}$$

dove $\overline{f^2}$ è la media del quadrato dei valori calcolati nella 5.3 mentre $(\overline{f})^2$ è il valore al quadrato.

Per calcolare media di $\sigma_{f_{interna_x}}$ invece si può procedere direttamente applicando la formula della media:

$$\overline{sigma_f} = \frac{\sum_{x \in dataset}(\sigma_{f_{interna_x}})}{|individui \in dataset|} \tag{5.5}$$

Infine si calcola la **percentuale di varianza** e la **percentuale di varianza interna** , che è il rapporto espresso in percentuale tra la varianza e la media.

$$percentualeVarianza = \frac{\overline{sigma_f}}{\overline{f}} * 100 \tag{5.6}$$

$$percentualeVarianzaInterna_x = \frac{\overline{f_{interna_x}}}{\overline{f_{interna_x}}} * 100 \tag{5.7}$$

5.3.2 Selezione

Le due percentuali di varianza esprimono, da sole, la bontà di una determinata feature.
Più è alta la *percentuale di varianza*, più il valore di una feature varia da un volto all'altro. Di conseguenza più è alto questo valore, più la feature è discriminante nell'individuare volti diversi.

La *percentuale di varianza interna*, invece, più è bassa più è probabile che la feature sia affidabile, ovvero che vari poco se calcolata su immagini diverse rappresentati lo stesso individuo. Se invece è alta indica che la feature è molto dipendente dalla posa con cui è stata scattata la foto, quindi probabilmente non è una buona feature.

La combinazione di queste due ci fornisce un indicazione chiara sull'utilità della feature. Affinché la feature sia **significativa**, è necessario che la *percentuale di varianza interna* sia inferiore alla *percentuale di varianza*. Le feature che rispettano questa condizione, statisticamente, sono utili al riconoscimento facciale.
Maggiore sarà la differenza tra questi due valori, maggiore sarà l'efficacia della feature.

Si è quindi proceduto al calcolo di questi valori per ogni feature, si sono ordinati in base alla differenza tra questi in modo che siano dal più rilevante al meno rilevante. Così in base alle esigenze (velocità di analisi o accuratezza) è possibile scegliere i primi n valori.

Il tipo di feature viene salvato in un file di testo per poi essere ricaricato in fase di confronto.

5.4 Metodo Alternativo

Durante lo sviluppo sono stati svolti anche dei test alternativi. Fin ora si è provato a confrontare lo stesso tipo di feature indipendentemente dal volto in questione. Si è supposto che ogni volto abbia caratteristiche peculiari, e che queste che vengano messe in risalto solo da determinate feature. Mentre per altri individui le stesse feature potrebbero risultare inefficaci. Ad esempio se una persona ha il naso grande, le feature che descrivono il naso saranno molto determinanti ai fini dell'identificazione. Tali feature, però, non danno le stesse performance in individui con le dimensioni del naso nella norma.

Si è provato quindi a non generalizzare la lista dei punti per ogni individuo, bensì a crearne una per ognuno, analizzando e ordinando come nel paragrafo precedente ogni feature, ma limitando il calcolo alle immagini provenienti dallo stesso individuo. Stilando così diverse liste per tutti gli individui appartenenti al dataset.
I risultati sono stati soddisfacenti, la percentuale di riconoscimento è alta (intorno al 90%) e consente di riconoscere volti in qualunque posa.
Svantaggio di tale approccio è la necessità di avere più foto dell'individuo per un training personalizzato. Inoltre, il fatto che l'analisi del volto risulti più complessa rispetto al metodo precedente, lo rende difficilmente applicabile su un database di grandi dimensioni.

Capitolo 6

Riconoscimento facciale

Una volta ottenuto il file con le feature migliori, si è proceduti all'ultima fase, ovvero il riconoscimento vero e proprio. Il risultato del confronto tra immagini restituirà un valore che indicherà quanto diverse siano le due immagini.

6.1 Confronto tra due volti

La distanza tra i valori delle feature calcolate sui due volti sarà utilizzata per comprendere se si tratta della stessa persona oppure no.

6.1.1 Il file con le feature

Il calcolo verrà effettuato sulle feature scelte dall'analisi descritta in 5. Questa analisi restituisce un file in cui sono contenute le informazioni su quali feature, in determinati punti, danno una performance migliore. Ecco un estratto del file.

```
11 43 0 53|0.000900471
7 18 1 33|0.00249221
4 17 1 28|0.00216222
6 60 1 41|0.000858405
6 21 0 0|0.0022296
6 18 0 28|0.00222824
2 62 1 44|0.00188214
8 21 0 4|0.00194265
4 21 0 28|0.00203956
```

I primi tre valori sono relativi alla posizione del punto all'interno della maschera; il quarto è riferito all'ID della feature, mentre nell'ultimo è riportata la $\overline{\sigma_f}$ (risultato del calcolo della 5.5) di quella feature.

6.1.2 Calcolo della compatibilità

La distanza tra i volti si calcola solo con le feature contenute nel file descritto in 6.1.1.

```
float val=(abs((float)(a[i]-b[i])))<(mediaVarianzeInterne*3)?1:0;
```

dove a e b sono array di dati contenenti le feature estratte dalle due immagini, *mediaVarianzeInterne* è la $\overline{\sigma_f}$ relativa a quella feature. Questa funzione restituisce 1 se la differenza è inferiore al triplo della $\overline{\sigma_f}$, altrimenti zero.
Per ogni feature, di tutti i punti presente in entrambi i volti, vengono sommati i valori risultanti dai confronti (1 e 0) e infine divisi per il numero di feature confrontate.
Si ottiene così un valore espresso in percentuale di compatibilità tra i due volti.

6.2 Considerazioni

Questo metodo consente di confrontare volti a due a due calcolandone la compatibilità. Dopo i test svolti sul set a disposizione, si è visto che se la compatibilità è superiore al 80% si può assumere che si tratti della stessa persona. Il metodo confronta immagini non sempre complete. Se le feature estratte non sono almeno un centinaio, il valore ottenuto non è affidabile.
Questo accade se le due foto sono state prese da angolazioni differenti (vedi figura 6.1), quindi alcune feature, per mancanza di informazioni, non possono essere confrontate.

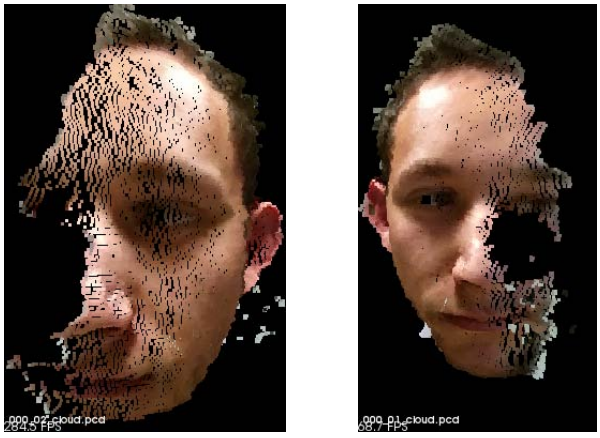


Figura 6.1: Esempi di pattern per le feature di Haar

Capitolo 7

Risultati e considerazioni

In questo capitolo si analizzeranno le performance degli algoritmi precedentemente descritti. Il programma è stato testato su una macchina le cui specifiche sono riportate in 3.1. Si tenga conto che, su un computer di ultima generazione, la velocità di esecuzione risulta ridotta di almeno di un ordine di grandezza.

7.1 Performance riconoscimento volto ed estrazione dello sfondo

L'algoritmo proposto in 3.2.2, testato su tutte le foto del dataset, ha trovato nel 100% dei casi, l'ubicazione esatta del volto. Il tempo di esecuzione dell'algoritmo è approssimativamente tra 1 e 2 secondi, comprensivo di fixing della point cloud.

7.2 Performance dell'allineamento del volto

Con l'algoritmo proposto in 3.3 si sono allineate le immagini. Tale algoritmo, sul dataset, ha una percentuale di riuscita del 77,08%.

Non si raggiunge il 100% per la difficoltà di OpenFace nell'individuare i volti presi di profilo. Talvolta viene selezionata un area dove non è presente il volto (ad esempio vengono individuati gli occhi sui capelli).

Su immagini frontali, invece, la percentuale è del 100% a prescindere dalla luminosità e dalla distanza dal sensore.

Il tempo di allineamento totale, comprensivo del secondo tramite ICP, è di $5 \sim 6$ secondi.

7.3 Risultati dell'analisi delle feature

7.3.1 Tempo di esecuzione

Una volta allineati e salvati tutti i volti si sono testate le feature sui volti ottenuti. Questo passaggio, a seconda delle complessità di calcolo delle feature scelte, può richiedere un tempo compreso tra i 3 e i 7 minuti. Questa operazione si divide in due fasi: la prima di estrazione delle feature e la seconda di elaborazione e ordinamento delle stesse. Il 95% del tempo di calcolo è impiegato nella fase di estrazione, quella di elaborazione non richiede generalmente più di 10 secondi.

7.3.2 Posizione feature più performanti

L'analisi sulle feature ha dato risultati diversi rispetto agli studi trovati in letteratura. In figura 7.1 sono rappresentate due maschere ove vengono evidenziati i punti delle 200 feature più rilevanti.

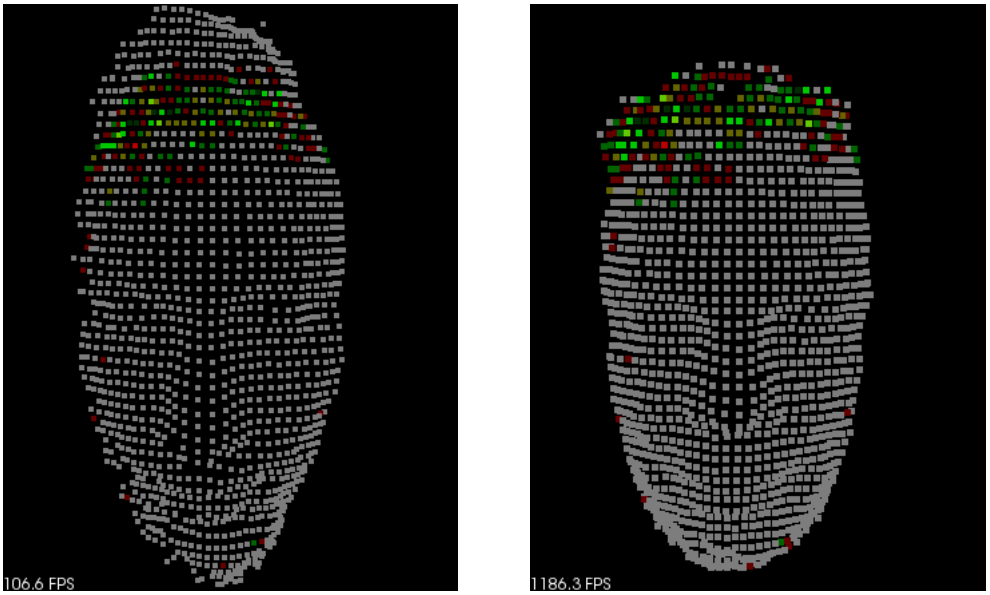


Figura 7.1: Le 400 feature più rilevanti

I colori indicano la quantità di feature rilevanti per ogni punto, variando dal rosso (1 o 2) al verde (5 o 6). Come si può notare la maggior parte dei punti trovati è situata sulla fronte dell'individuo e non sul naso o gli occhi come ci si aspetterebbe.
Questo può essere dovuto al fatto che, l'allineamento del volto (descritto in

3.3.3), si concentra sul naso, rendendo i punti nell'area centrale molto vicini tra le varie maschere anche di volti diversi. Lasciando quindi le peculiarità di ogni volto alle estremità.

I sistemi di riconoscimento classici si basano su caratteristiche biometriche come: naso, bocca e occhi. Il fatto che i punti siano la fronte e le guance è interessante perché fornisce altri strumenti per il riconoscimento, che potrebbero essere affiancati a quelli già esistenti per aumentare l'efficacia dei classificatori.

Considerando tutte le feature rilevanti, anziché le prime 400, la distribuzione appare come in figura 7.2.

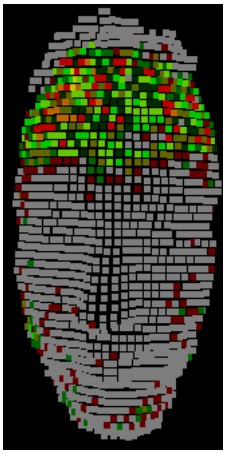


Figura 7.2: Rappresentazione di tutte le feature più rilevanti

7.3.3 Tipologia feature più performanti

Analizziamo ora le feature per tipologia invece che per posizione. Il grafico 7.3 mostra quali feature danno risultati migliori tra tutte quelle elencante in 4.

Dall'analisi del grafico 7.3 e della tabella 7.1, emerge che la scelta tra le feature migliori ricade solo su alcune di queste. Come si può notare, la più performante è la n° **28** che è l'indice della feature di Haar 3D sulla coordinata z calcolata con la settima feature di Haar mostrata in figura 4.5, ovvero quella con l'angolo di 157,5°.

Le altre feature con il punteggio più alto sono: la **0**, **4** e **8**; tutte feature di Haar sulla cordinata z con angoli rispettivamente di 0° 22,5° 45° (figure 1 2 e 3 di 4.5).

La feature **33**, invece, contiene la coordinata z del punto della maschera in questione.

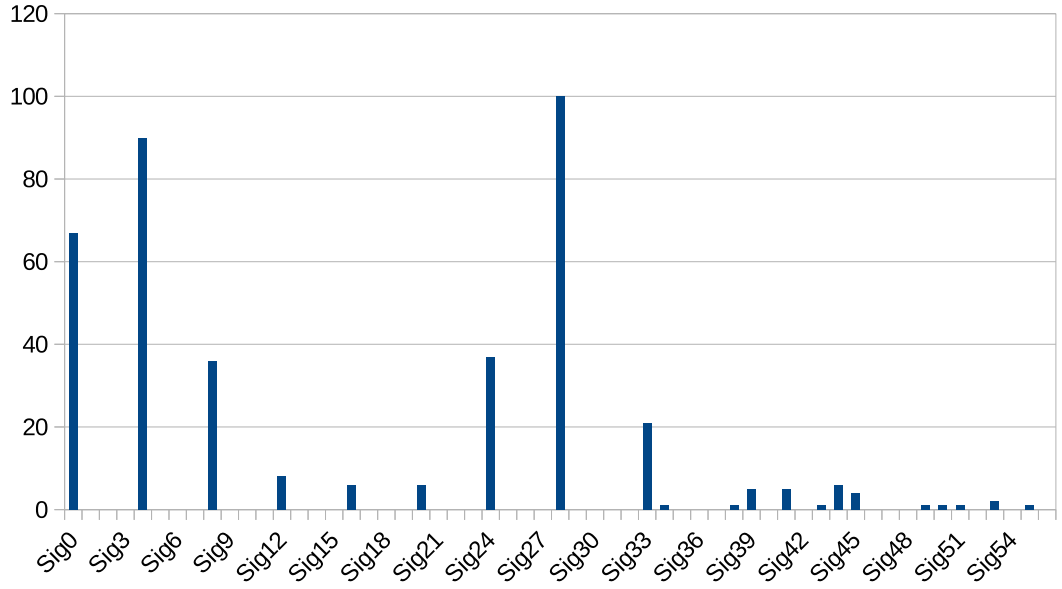


Figura 7.3: Grafico delle 400 più rilevanti trovate

In figura 7.4 sono rappresentate tutte le feature significative[1].
Da questo grafico si può notare la totale inutilità delle feature di Haar calcolate sulla differenza delle normali (ovvero tutte le feature dalla 0 alla 32 non divisibili per 4). L'unica che spicca, tra le feature non di Haar, è la feature **33** che contiene la coordinata z del punto della maschera.

Le feature dalla **36** in poi calcolano la distanza euclidea tra i punti circostanti (come spiegato in 4.4). Queste, benché siano rilevanti, danno un contributo minimo al riconoscimento.

7.3.4 Considerazioni sulle PFH

Durante l'analisi si è deciso di rimuovere tutte le feature basate sul metodo PFH, per via delle loro scarse performance e dell'elevata onerosità in termini di calcoli. Questa feature funziona molto bene nell'individuare punti come: spigoli, piani e superfici di vario genere. Non si è, invece, dimostrata adatta nel discriminare volti per le caratteristiche simili degli stessi.

[1] Vedi 5.2

ID	N°	ID	N°	ID	N°
0	67	28	100	44	6
4	90	33	21	45	4
8	36	34	1	49	1
12	8	38	1	50	1
16	6	39	5	51	1
20	6	41	5	53	2
24	37	43	1	55	1

Tabella 7.1: Tabella delle 400 più rilevanti trovate

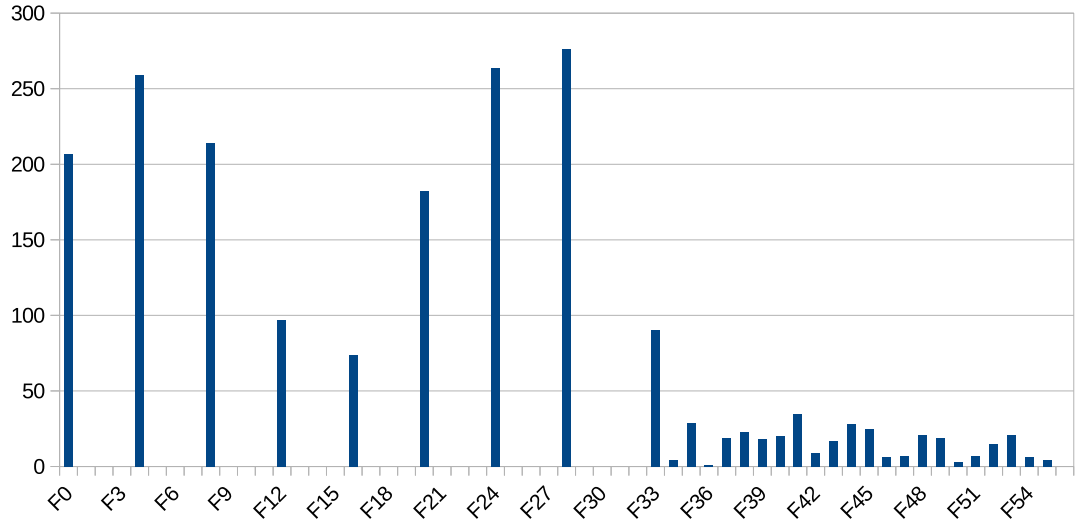


Figura 7.4: Grafico di tutte le feature più rilevanti trovate

7.3.5 Considerazioni sui Keypoint di OpenFace

I Keypoint di OpenFace, nonostante il loro contributo non sia consistente, sono stati considerati rilevanti ai fini del riconoscimento di individui. Le performance sono comunque inferiori alle feature di Haar 3D e al confronto delle posizioni z di alcuni punti.

7.4 Testing dell'algoritmo

L'algoritmo è stato testato su **185** immagini e si sono confrontate tutte le coppie di queste calcolando la loro percentuale di compatibilità. In base a quest'ultima si è dedotta la percentuale di affidabilità dell'algoritmo.

Si è notato, inoltre, che per poter calcolare una percentuale di compatibilità affidabile è necessario confrontare un numero di feature almeno superiore ai $\frac{2}{5}$ del totale delle feature. Questa restrizione, però, non rende possibile alcuni confronti (come spiegato in 6.2).

Nei file a disposizione si è potuto confrontare con successo solo il **57.3%** delle coppie di immagini possibili.

Il test per verificare l'efficacia dell'algoritmo si è svolto nel seguente modo:

```
Per ogni coppia a,b di immagini a disposizione;
  Si estraggono le feature per entrambe le foto;
  Se è possibile confrontare tra loro
  più di 2/5 delle feature totali
  ALLORA:
    Si calcola la percentuale di affinità
    di una foto con un'altra;
    Se questa è maggiore della soglia scelta
    ALLORA:
      Se la coppia di foto raffigura la stessa persona
      ALLORA:
        truePositive++;
      ALTRIMENTI:
        falsePositive++;
    ALTRIMENTI:
      Se la coppia di foto raffigura persone diverse
      ALLORA:
        trueNegative++;
      ALTRIMENTI:
        falseNegative++;
```

Il valore di soglia della percentuale di affinità utilizzato è del **82%**.

I valori che si ottengono al termine del testing sono:

truePositive: numero di confronti tra coppie di foto raffiguranti il medesimo individuo che sono state correttamente riconosciute;

trueNegative: numero di confronti tra coppie di foto raffiguranti individui diversi che non sono state correttamente rilevate come diverse;

falsePositive: numero di confronti tra coppie di foto raffiguranti individui diversi rilevati come se fossero la stessa persona;

falseNegative: numero di confronti tra coppie di foto raffiguranti il medesimo individuo che sono state rilevate come diverse;

Nell'esperimento condotto, tra tutte le coppie in cui è stato possibile effettuare un confronto, abbiamo ottenuto:

TruePositive	285
TrueNegative	7656
FalsePositive	484
FalseNegative	88

da cui possiamo dedurre il:

True Positive Rate=$\frac{TruePositive}{FalseNegative}$=76,4%

True Negative Rate=$\frac{TrueNegative}{FalsePositive}$=94%

Percentuale affidabilità=$\frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$=93,2%

Il primo indica la probabilità che due foto vengano identificate come uguali se raffiguranti due volti della stessa persona mentre il secondo indica la probabilità che due foto vengano identificate come diverse se raffiguranti due volti diversi.

L'ultimo valore indica la percentuale di volte in cui il classificatore ha correttamente identificato la somiglianza tra due individui.

Capitolo 8

Conclusioni e sviluppi futuri

8.1 Conclusioni

Al termine del processo si è potuto constatare come le informazioni 3D possano dare un'enorme contributo nel riconoscimento facciale. Si è vista la loro efficacia nell'individuazione ed estrazione dei volti all'interno delle foto (contente altri volti e oggetti di vario genere).
Il metodo sviluppato è risultato efficace nel 100% dei casi e le sue performance sono state migliori sia in termini di tempo computazionale, sia in termini di affidabilità, dei più comuni Face Detector bidimensionali come *HOG* (di Open Face) e *Haar Cascade Classifier* (di OpenCV).

OpenFace è risultato essere un ottimo strumento di elaborazione di volti bidimensionale, veloce e affidabile ma solo per quanto riguarda i volti presi frontalmente, per quelli di profilo invece, si sono presentati non pochi problemi, in parte risolti sfruttando le informazioni 3D.

Grazie al tipo di studio condotto si è assodato che le feature di Haar 3D, adottate in questa tesi siano un ottimo strumento per l'identificazione dei volti. Si è inoltre visto che la maggior parte delle informazioni utili al face reconition vengono estratte dai punti appartenenti alla fronte dell'individuo. I risultati del riconoscimento mostrati in 7.4 mostrano come le feature scelte possano dare dei risultati notevolmente accurati.

Il metodo di analisi della validità delle feature sviluppato è valido, semplice e fornisce dei risultati facilmente interpretabili. È inoltre possibile utilizzarlo per testare l'efficacia di nuove feature in relazione con le vecchie.

La debolezza più grande di questo algoritmo è l'impossibilità di confrontare immagini troppo lontane dall'obiettivo per i motivi spiegati in 6.2. La totalità delle foto a distanza di 2 metri dall'obiettivo, infatti, sono state ignorate durante i confronti per insufficienza di feature confrontabili.

8.2 Sviluppi Futuri

Il processo di allineamento non funzioni in alcune situazioni, potrebbe essere migliorato in vari modi, come allenando il classificatore di OpenFace su pose non frontali o testando altri metodi di allineamento più efficaci.

Si può inoltre inserire nel classificatore le più comuni feature 2D presenti in letteratura, ottenendo una buona classificazione anche in assenza di informazioni dettagliate sulla profondità. In questo modo si avranno feature confrontabili anche in immagini a 2 o più metri di distanza dal sensore. Il sistema, in questo modo, sceglierà automaticamente quali feature usare in base alla situazione.

È anche possibile aumentare il livello di confidenza al fine di ottenere un **True Positive Rate** prossimo al 100% ed inserire il classificatore, così modificato, in un algoritmo AdaBoost di classificatori bidimensionali al fine di aumentarne l'affidabilità.

Il metodo di confronto delle feature sviluppato si basa su semplici confronti di valori, quindi facilmente implementabile su data base. È possibile così confrontare un volto con migliaia di essi e ottenere una risposta in tempi ragionevoli.

Appendice A

Preparazione ambiente di sviluppo

Sulla macchina si è installato Ubuntu 14.04.5 LTS [1].

A.1 Installazione OpenCV

Una volta installato si è seguita la guida ufficiale sul sito di OpenCV[16].

A.1.1 Installazione prerequisiti

Per funzionare ha bisogno che sulla macchina siano installati diversi pacchetti, che vengono installati automaticamente eseguendo questi comandi da terminale

```
sudo apt-get install build-essential
sudo apt-get install cmake git
  libgtk2.0-dev pkg-config
  libavcodec-dev libavformat-dev
  libswscale-dev
sudo apt-get install python-dev
  python-numpy libtbb2 libtbb-dev
  libjpeg-dev libpng-dev libtiff-dev
  libjasper-dev libdc1394-22-dev
```

[1] Reperibile all'indirizzo `http://releases.ubuntu.com/14.04/`

A.1.2 Download

Terminata questa fase la macchina è pronta per l'installazione vera e propria di **OpenCV**. Per reperire i sorgenti si è utilizzato il seguente comando da terminale:

```
cd ~
mkdir opencv
cd ~/opencv
git clone https://github.com/opencv/opencv.git
```

A.1.3 Compilazione

Questo crea la cartella *opencv* e vi scarica al suo interno i sorgenti. A questo punto li si compila eseguendo questi comandi:

```
cd ~/opencv
mkdir release
cd release
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

Il comando dato crea la cartella *release* all'interno di quella di *opencv* dove andranno salvati tutti i file compilati. Inoltre si indica il percorso di destinazione dove, in fase di installazione, si andranno a posizionare i file.

A.1.4 Installazione

Come ultimo passaggio si è proceduto all'installazione vera e propria utilizzando i comandi:

```
make
sudo make install
```

A.2 Installazione di PCL - Point Cloud Library

Installare PCL è molto simile all'installazione di OpenCV, la guida seguita è stata presa dal sito di github [17]. Per semplicità riporteremo soltanto i comandi.

A.2.1 Download e salvataggio

```
sudo apt-get update
sudo apt-get install git

cd ~
mkdir PCL
cd ~/PCL
git clone https://github.com/PointCloudLibrary/pcl.git pcl-trunk
ln -s pcl-trunk pcl
```

A.2.2 Installazione prerequisiti

```
sudo apt-get install g++
sudo apt-get install cmake cmake-gui
sudo apt-get install doxygen
sudo apt-get install mpi-default-dev openmpi-bin openmpi-common
sudo apt-get install libflann1.8 libflann-dev
sudo apt-get install libeigen3-dev
sudo apt-get install libboost-all-dev
sudo apt-get install libvtk5.8-qt4 libvtk5.8 libvtk5-dev
sudo apt-get install libqhull*
sudo apt-get install libusb-dev
sudo apt-get install libgtest-dev
sudo apt-get install git-core freeglut3-dev pkg-config
sudo apt-get install build-essential libxmu-dev libxi-dev
sudo apt-get install libusb-1-0-dev graphviz mono-complete
sudo apt-get install qt-sdk openjdk-7-jdk openjdk-7-jre
sudo apt-get install phonon-backend-gstreamer
sudo apt-get install phonon-backend-vlc
```

A.2.3 Installazione supporto Kinect

```
mkdir ~/kinect
cd ~/kinect
git clone https://github.com/OpenNI/OpenNI.git

cd OpenNI
git checkout unstable

cd Platform/Linux/CreateRedist/
chmod +x RedistMaker
./RedistMaker
cd ../Redist/OpenNI-Bin-Dev-Linux-x64-v1.5.8.5
sudo ./install.sh

cd ~/kinect/
```

```
git clone https://github.com/ph4m/SensorKinect.git

cd SensorKinect
git checkout unstable

cd Platform/Linux/CreateRedist/
chmod +x RedistMaker
./RedistMaker
cd ../Redist/Sensor-Bin-Linux-x64-v5.1.2.1/
chmod +x install.sh
sudo ./install.sh
```

A.2.4 Installazione

```
cd ~/pcl
mkdir release
cd release
cmake -DCMAKE_BUILD_TYPE=None -DBUILD_GPU=ON
  -DBUILD_apps=ON -DBUILD_examples=ON ..
make
```

Bibliografia

[1] B. A. B. L. M. Satyanarayanan, *Openface: A general-purpose face recognition library with mobile applications.* CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[2] S. M. P. Z. A. B. S. G. S. Zennaro, M. Munaro and E. Menegatti., *Performance evaluation of the 1st and 2nd generation Kinect for multimedia applications.* In Proceedings of the IEEE International Conference on Multimedia and Expo, pp. 1-6, doi: 10.1109/ICME.2015.7177380, 2015.

[3] M. Carraro, M. Munaro and E. Menegatti., *A cost-efficient RGB-D smart camera for people detection and tracking.* Journal on Electronic Imaging, SPIE, vol. 25, no. 4, doi:10.1117/1.JEI.25.4.041007, 2016.d.

[4] F. B. M. Munaro and E. Menegatti., *OpenPTrack: Open Source Multi-Camera Calibration and People Tracking for RGB-D Camera Networks.* Journal on Robotics and Autonomous Systems, vol. 75, part B, pp. 525-538, ISSN: 0921-8890, doi: 10.1016.robot.2015.10.004, January 2016.

[5] G. Pitteri, M. Munaro, and E. Menegatti., *Depth-based frontal view generation for pose invariant face recognition with consumer RGB-D sensors.* In Proceedings of the 14th International Conference on Intelligent Autonomous Systems (IAS-14), Shanghai, July 2016.

[6] "Microsoft Kinect." `https://it.wikipedia.org/wiki/Microsoft_Kinect`, 2016. [Online; ultimo accesso 28-Febbraio-2017].

[7] "PCL - Point Cloud Library." `http://pointclouds.org`, 2016. [Online; ultimo accesso 1-Marzo-2017].

[8] "Point Cloud Library." `http://it.wikipedia.org/wiki/Nuvola_di_punti`, 2016. [Online; ultimo accesso 1-Marzo-2017].

[9] R. P. R. . A. S. . M. Sevak, "Face Recognition using Eye Distance and PCA Approaches." https://pdfs.semanticscholar.org/1823/3c55982050292ba7f6a5462c0e7576c3398d.pdf, 2014.

[10] "Open Face." http://cmusatyalab.github.io/openface/, 2016. [Online; ultimo accesso 1-Marzo-2017].

[11] "DLib." http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html, 2016. [Online; ultimo accesso 1-Marzo-2017].

[12] "Regressione Lineare." https://it.wikipedia.org/wiki/Regressione_lineare, 2016. [Online; ultimo accesso 1-Marzo-2017].

[13] "Pairwise Registration." http://pointclouds.org/documentation/tutorials/pairwise_incremental_registration.php, 2016. [Online; ultimo accesso 1-Marzo-2017].

[14] "Feature di Haar." http://www.ce.unipr.it/people/medici/geometry/node122.html, 2016. [Online; ultimo accesso 16-Marzo-2017].

[15] "Point Feature Histograms (PFH) descriptors." http://pointclouds.org/documentation/tutorials/pfh_estimation.php, 2016. [Online; ultimo accesso 16-Marzo-2017].

[16] "Installation in Linux - OpenCV 2.4.13.2 documentation." http://docs.opencv.org/2.4/doc/tutorials/introduction/linux_install/linux_install.html, 2016. [Online; ultimo accesso 16-Marzo-2017].

[17] "PCL 1.8: Ubuntu 14.04 Installation Guide." https://github.com/hsean/Capstone-44-Object-Segmentation/wiki/PCL-1.8:-Ubuntu-14.04-Installation-Guide, 2016. [Online; ultimo accesso 16-Marzo-2017].