Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Metodi e tecnologie per il riconoscimento facciale

Relatore:
Dottor Fabio Vitali

Presentata da:
Matteo Bacci

Sessione 12 Ottobre
Anno Accademico 2015/2016

Alla mia famiglia, a Chiara,
agli amici e ai colleghi che mi hanno sostenuto.

Indice

Introduzione

Lo scopo di questa tesi è illustrare i possibili utilizzi di uno strumento di Riconoscimento Facciale basato su OpenFace in ambiente bancario.

Una tecnologia di Riconoscimento Facciale permette di risalire all'identità di un soggetto o verificarla attraverso una sua fotografia digitale fornita al software.

Le identità che il software è in grado di riconoscere sono quelle fornite dagli utenti del servizio durante un processo detto training, durante il quale inseriscono le proprie immagini nel database.

Le immagini fornite non devono sottostare a molte regole: devono essere illuminate uniformemente e contenere il volto preso frontalmente in tutta la sua interezza, per permettere al servizio di rilevare il volto all'interno dell'immagine.

Il Riconoscimento Facciale è diventato sempre più importante negli ultimi tempi, attirando l'attenzione di diverse software house e aziende interessate alle sue applicazioni in ambito di sicurezza o profiling.

In vista di un progetto di innovazione portato avanti da BPER Banca l'azienda Antreem, con cui collaboro, ha proposto una serie di strumenti innovativi per capire quanto fossero maturi, cosa fosse possibile realizzare al momento e quali fossero le loro potenzialità.

BPER Banca si è mostrata interessata a uno studio degli strumenti di Face Recognition per una possibile applicazione futura, analizzando quindi anche i vari utilizzi interessanti.

Il mio contributo è stato quello di creare un software che mostrasse le

potenzialità del Riconoscimento Facciale in ambito bancario, ho deciso di sviluppare un servizio web che potesse essere utilizzato con gli struemnti previsti da un comune elaboratore moderno.

Il primo capitolo di questa dissertazione conterrà lo studio effettuato sullo stato dell'arte della tecnologia ed una panoramica dei limiti connessi, sia da un punto di vista etico che tecnico.

Le possibili implementazioni del Riconoscimento Facciale sono pressoché infinite, dai sistemi di sicurezza che permettono l'accesso soltanto a pochi soggetti a i software di catalogazione di album fotografici in grado di poter dire quali soggetti sono presenti in una certa immagine.

Vi sono però diverse problematiche legate al Riconoscimento Facciale, gli utenti di un servizio vogliono essere sicuri che i propri dati sensibili restino privati ed il volto è sicuramente uno di questi.

Oltre a questo bisogna considerare i limiti tecnici delle implementazioni, effettuare un riconoscimento si riassume con valutare la distanza euclidea tra due volti nell'insieme di dati, ma definire quando questa distanza sia entro un limite accettabile è estremamente complicato.

Il secondo capitolo proporrà un'implementazione web del Riconoscimento Facciale, realizzata per mostrarne le capacità effettive.

La pagina web realizzata rispetta inoltre gli stili e i principi dei siti moderni, prevedendo la possibilità di essere utilizzata su dispositivi di qualunque dimensione ed utilizzati con qualunque orientamento.

Per mantenere la possibilità di utilizzare il servizio su qualunque dispositivo ho utilizzato stili e metodi previsti dalla maggioranza dei browser sul mercato.

Il server agisce in background e si occupa di tutti gli aspetti legati alla Face Recognition, mentre l'acquisizione delle immagini avviene lato client.

Il client prevede una semplice interfaccia che mostra le immagini acquisite in tempo reale sul monitor arricchite delle informazioni ricevute dal server: area del volto, punti di rilevamento e identità del soggetto rilevato.

È inoltre previsto un pannello per l'inserimento di un'identità tramite

cui effettuare la validazione che restituisce a video il successo o l'insuccesso tramite due tipologie di feedback: il colore delle icone e dei riquadri e il testo più espressivo che indica il risultato.

Oltre all'interfaccia principale, vi è una seconda pagina contenente alcuni dettagli sul software e alcuni grafici interattivi che riportano i risultati di vari test effettuati.

Il terzo capitolo conterrà tutti i dettagli tecnici dell'implementazione entrando nel dettaglio sui metodi utilizzati, le scelte progettuali e gli strumenti in grado di migliorare l'esperienza utente e la manutenibilità del codice.

OpenFace è il software al centro del progetto e implementa tutti i metodi legati alla Face Recognition, nel progetto è stato inserito come una libreria Python accessibile al server.

Per realizzare il client ho utilizzato il task runner Gulp che permette di mantenere il codice diviso in modo estremamente modulare, così da rendere semplice la manutenzione e l'integrazione del servizio in altri ambiti.

Il software è inoltre distribuito tramite Docker, un framework che permette di astrarre la distribuzione creando delle macchine virtuali in cui l'ambiente di sviluppo comprende tutto l'essenziale per eseguire il progetto.

Infine, nel quarto capitolo, verranno esaminati alcuni test sullo strumento e proposti alcuni miglioramenti e sviluppi futuri.

Uno delle principali esigenze emerse da BPER Banca è quella di avere il prodotto disponibile su più piattaforme possibili per cui è stata chiara fin da subito la necessità di utilizzare il client come applicazione nativa dai propri smartphone e questa implementazione è attualmente in sviluppo.

La problematica più grave con cui sono entrato a contatto, sebbene impossibilitato a testarla, è legata a come questo servizio riesca a reggere nel passare del tempo e l'invecchiamento degli utenti, sempre meno simili alle immagini fornite in input.

Una possibile soluzione sarebbe quella di aggiornare frequentemente i dati, permettendo alle immagini nel dataset di rimanere aggiornate con quelle più recenti del soggetto e quindi ottenere risultati precisi anche a distanza di

molto tempo rispetto all'acquisizione delle immagini iniziali.

Capitolo 1

Strumenti di riconoscimento facciale

Un sistema di riconoscimento facciale è un software in grado di identificare o verificare l'identità di un soggetto tramite un'immagine digitale.

Questo capitolo illustrerà lo stato dell'arte degli strumenti di riconoscimento facciale, con particolare attenzione alle cosiddette tecniche tradizionali e agli ambiti in cui applicarle.

1.1 Tecniche tradizionali

In questa prima sezione entrerò nel dettaglio su cosa si intenda per tecnica tradizionale e quali miglioramenti l'hanno portata ad essere il metodo più utilizzato per il riconoscimento facciale.

Sebbene ci siano diversi possibili approcci al problema, che sfruttano diverse tecnologie, come ad esempio l'utilizzo di telecamere 3D o termiche, ho preferito concentrarmi sulle tecniche tradizionali data la grande disponibilità di materiale e documentazione ad esse legata.

La prima idea a rientrare in questa categoria risale al 1989, da una pubblicazione di Tuevo Kohonen[1] i cui concetti fondamentali sono ancora alla base di tutti i software di riconoscimento facciale moderni.

Kohonen introduce la possibilità di rappresentare un volto in uno spazio bidimensionale, creando di fatto un'impronta unica del volto in un formato comprensibile alla maggior parte degli elaboratori.

È possibile semplifcare ulteriormente la rappresentazione ed utilizzare solo l'insieme di autovettori legato alla matrice in grado di rappresentare le caratteristiche uniche del volto del soggetto. Questi insiemi vengono chiamati eigenfaces quando applicati al riconoscimento facciale.

Il grande vantaggio di utilizzare autovettori sta nel fatto che riescano a rappresentare un volto in uno spazio dimensionale ancor più ridotto, permettendo quindi operazioni che al tempo risultavano complicate su un normale elaboratore[2].

Un altro importante algoritmo che ha permesso alle tecniche tradizionali di raggiungere un livello molto alto in termini di velocità e precisione è PCA.

PCA (Principal Component Analysis) è un algoritmo in grado di isolare e rappresentare le componenti *tipiche* di un insieme di dati, utilizzato principalmente nel pattern recognition.

Questo algoritmo cerca di estrarre i fattori tipici di un dataset per focalizzarsi sulle variazioni rispetto al pattern calcolato.

Ha ottenuto ottimi risultati nel campo del riconoscimento facciale, infatti le variazioni calcolate da PCA sulle eigenfaces corrispondono ai fattori determinanti che distinguono un volto da un altro, aumentando precisione e velocità.

Nel riconoscimento facciale, si utilizza PCA per definire quali sono i tratti del volto in grado di caratterizzarlo, portando a massimizzare le differenze tra volti.

Vi è tuttavia un grosso effetto collaterale a questa tecnica, ovvero la possibilità di massimizzare non solo le caratteristiche del volto, ma anche i difetti dell'immagine, come ad esempio la scarsa luminosità.

Per aggirare questo problema, una ricerca di Peter N. Belhumeur, Joao P. Hespanha e David J. Kriegman nel 1997[3], introduce la possibilità di inserire l'utilizzo di un modello statisticamente accurato del volto; questo fa sì che

l'accuratezza di PCA possa migliorare escludendo le caratteristiche estranee al volto.

Questo algoritmo, che prevede l'utilizzo di un modello statistico compatibile all'insieme di dati su cui si sta lavorando, è detto LDA (Latent Dirichlet Allocation) e quando applicato al mondo del Riconoscimento Facciale prende il nome di Fisherfaces.

Tutti questi studi legati all'argomento hanno portato le tecniche tradizionali ad essere uno dei metodi migliori ad oggi per eseguire il Riconoscimento Facciale.

1.2 Possibili utilizzi e limiti connessi

Nella prossima sezione illustrerò i due compiti fondamentali del Riconoscimento Facciale e le loro implementazioni tipiche.

Il concetto di riconoscimento facciale si espande a due principali temi:

- Identificazione, si occupa di riconoscere un soggetto partendo da un'immagine;
- Validazione, si occupa di controllare l'identità fornita da un soggetto attraverso un'immagine.

Su entrambi si hanno grossi limiti, già approfonditi in diversi studi, la maggior parte pone diversi dubbi sulla sicurezza e sul rispetto della privacy degli utenti sottoposti a questi strumenti.

1.2.1 Identificazione e privacy

Con identificazione si intende la capacità di riconoscere un soggetto fornendo una sua foto all'elaboratore; questo ha fatto emergere una serie di dubbi relativi all'etica di questi servizi.

Dopo l'attenato dell'11 Settembre nel 2001 è aumentata la richiesta di utilizzo di fattori biometrici (come il riconoscimento facciale) per identificare

potenziali soggetti pericolosi e già nel 2004 un articolo di K. W. Bowyer[4] affianca gli strumenti di riconoscimento facciale al diritto alla privacy, esponendo i propri dubbi etici, arrivando a citare Banjamin Franklyn riguardo al baratto della libertà per ottenere più sicurezza.

Qualche anno dopo, nel 2009, un altro studio legato all'argomento [5], inizia a mostrare una possibile ed efficiente implementazione per uno strumento di riconoscimento facciale che rispettasse anche la privacy degli utenti.

Questa implementazione prevede un'applicazione in cui il server contiene un insieme di dati relativi ai soggetti e uno (o più) client che vi si interfacciano.

Il client richiede al server di controllare la presenza (o l'assenza) di uno specifico volto nel server, il server allo stesso modo mantiene la privacy del suo dataset non esponendo altre informazioni se non il risultato dell'algoritmo di riconoscimento.

L'implementazione di un sistema ispirato a questo studio è comunque molto pesante in termini computazionali per essere utilizzata su grandi sistemi, ma offre diversi punti a favore dell'utilizzo di sistemi di riconoscimento facciale.

1.2.2 Identificazione e sicurezza

Il concetto di identificazione è estremamente compatibile con quello di sicurezza, ogni volta che effettuiamo l'autenticazione su un servizio ci stiamo in qualche modo identificando.

I metodi standard di autenticazione si sono sempre basati su vari fattori, riconducibili a quattro categorie principali[6]:

- Qualcosa che sai, le classiche password;
- Qualcosa che hai, una chiave fisica;
- Qualcosa che sai e qualcosa che hai, l'insieme dei precedenti;
- Qualcosa di unico dell'utente, l'impronta digitale, il volto, ecc.

Il nostro caso è ovviamente l'ultimo, ma tra questi il volto è il metodo meno sicuro da utilizzare per l'identificazione, prima di tutto perché sebbene sia estremamente facile per un essere umano dare un nome ad una faccia, per un elaboratore risulta altrettanto difficile.

Non si tratta di semplici dati statici: un volto può variare nel tempo o anche solo con un cambio di illuminazione o espressione, rendendo il riconoscimento sempre meno preciso.

Diverse ricerche[7] ne propongono invece l'utilizzo in ambienti di autenticazione, dove l'utente dichiara la propria identità (ad esempio attraverso l'username) e il sistema si occupa solamente di convalidarla attraverso un'immagine del volto.

La *Face Authentication* è computazionalmente molto semplice, utilizzando un sistema matematico per rappresentare i volti è sufficiente calcolare la distanza euclidea tra la rappresentazione del viso del soggetto e il vettore corrispondente all'interno del database.

In questo modo è possibile anche mantenere i concetti che ne garantiscono il rispetto della privacy dell'utente, utilizzando una rappresentazione matematica difficilmente reversibile e rispondendo all'utente soltanto con l'esito dell'operazione di autenticazione.

1.3 Problema: identificazione contro autenticazione

Alla luce degli studi evidenziati precedentemente ho preferito lasciare da parte il concetto di identificazione, che avrebbe messo a rischio la sicurezza degli utenti, o leso l'usabilità dei servizi a causa della sua imprecisione.

L'autenticazione è invece rimasta un punto solido su cui ho voluto investire; per non mettere a repentaglio la sicurezza dell'utente è utilizzata come una verifica in due passaggi, in cui si suppone che l'utente si stia collegando da un dispositivo affidabile e il servizio possa cercare un riscontro direttamente con i dati dell'utente.

Molte software house hanno iniziato ad utilizzare tecnologie di Face Recognition già da diverso tempo (Facebook, Google e tanti altri), sono quindi disponibili diversi software, più o meno avanzati, da utilizzare a questo scopo.

Per arrivare all'implementazione ho preso in esame diversi software open source: OpenCV, OpenBR e OpenFace.

OpenCV[1] contiene tutte le basi fondamentali per il Ricoscimento Facciale ed è una tra le scelte più diffuse contando circa quarantasettemila progetti che lo includono, tra questi gli altri due software citati.

OpenBR[2] fornisce invece l'accesso a moltissime applicazioni di fattori biometrici, tra cui quella del Riconoscimento Facciale, eseguito appunto con l'aiuto di OpenCV.

La scelta è ricaduta infine su OpenFace che si occupa unicamente di Riconoscimento Facciale utilizzando inoltre algoritmi in grado di migliorare nel corso del tempo tramite l'utilizzo di machine learning e pattern recognition, entrerò nei dettagli di OpenFace nella prossima sezione.

A differenza di OpenBR, essendo OpenFace un software specializzato nel Riconoscimento Facciale è anche meglio documentato e contiene solo i metodi necessari all'implementazione desiderata.

1.4 Openface

OpenFace[3] è un software open source scritto in Python che implementa la Face Recognition, e si basa su una ricerca del 2015[8] fine a questo scopo.

OpenFace utilizza OpenCV (altro noto software per la Face Recognition) per allineare le immagini, in modo che occhi e labbra appaiano sulla stessa linea, su tutte le immagini, al di là di questo non fa nessun altra trasformazione sulle immagini.

Viene inoltre sfruttato fortemente il machine learning, una tecnica in grado di analizzare i dati in maniera simile agli esseri umani, nello specifico i

[1] http://opencv.org/

[2] http://openbiometrics.org/

[3] http://cmusatyalab.github.io/openface/

volti vengono rappresentati attraverso una *deep neural network*, implementata tramite Torch[4], un framework scritto in C++ che riporta i principali concetti di machine learning.

Questa è la grossa differenza con gli altri software di riconoscimento facciale ed è anche il motivo che permette ad OpenFace di essere estremamente veloce.

La maggior parte dei software inoltre utilizza soltanto la distanza euclidea tra le rappresentazioni per effettuare il riconoscimento, utilizzando la rete neurale, due facce molto distanti nel modello utilizzato per il riconoscimento, appartengono con grande probabilità a due soggetti distinti aumentando la precisione anche quando la distanza euclidea non è significativa.

Il test LFW (Labeled Faces in the Wild)[9] è un test per la valutazione dell'accuratezza di un algoritmo di Face Recognition, quello implementato da OpenFace ha ottenuto ottimi risultati riportati nella figura 1.1.

Il test mette a confronto vari algoritmi di Face Recognition con i dati statistici legati al riconoscimento umano sulla base dei risultati corretti rispetto a un tetto massimo legato ai falsi negativi.

Oltre alle varie linee grigie e quella nera che rappresentano i vari modelli utilizzati da OpenFace, la linea rossa rappresenta il valore statistico del riconoscimento umano, la linea azzurra l'algoritmo implementato da OpenBR e la linea verde quello di Eigenface.

Visti gli ottimi risultati dichiarati da OpenFace sia dal punto di vista della precisione che dal punto di vista della rapidità, ho deciso di sceglierlo come base su cui realizzare l'implementazione.

[4]http://torch.ch/

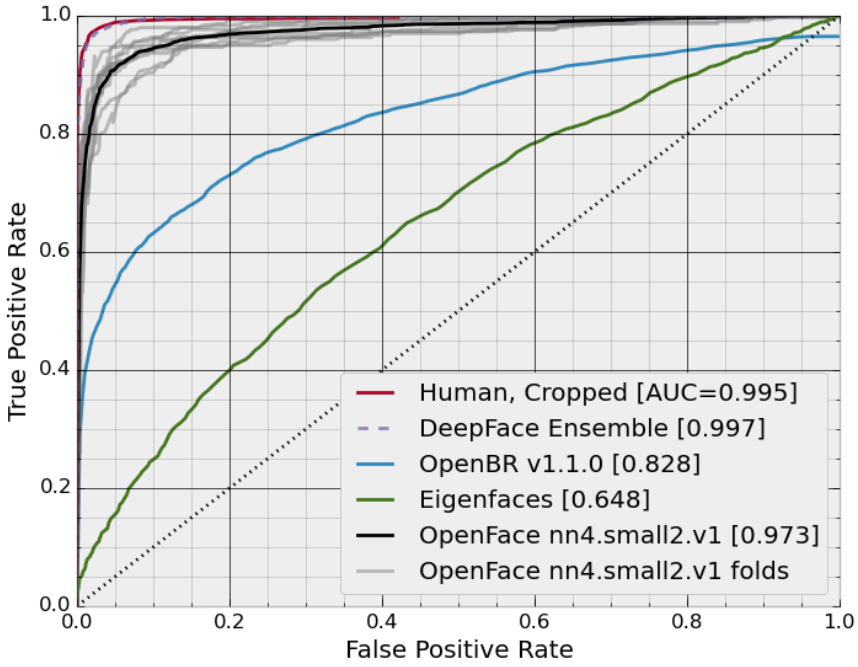

Figura 1.1: Risultati del test LFW con nn4.small2.v1 di Agosto 2015

Capitolo 2

Un'implementazione della Face Recognition

Nel prossimo capitolo illustrerò il flusso tipico di Riconoscimento Facciale e l'implementazione web di Riconoscimento ed Autenticazione che è stata proposta a BPER.

2.1 Il flow di Face Recognition

Il flusso legato alle tecnologie di Riconoscimento Facciale è simile indipendentemente dalla piattaforma scelta, di seguito verranno riportati tutti i passaggi fondamentali utilizzando come esempio l'implementazione tipica suggerita dalla documentazione di OpenFace.

- Training del modello

  Prima di procedere con il riconoscimento facciale vero e proprio è necessario effettuare il training del modello. Partendo da un'insieme di volti noti rappresentati bidimensionalmente e assegnati ai soggetti corrispondenti, OpenFace genera una Support Vector Machine (SVM), ovvero un modello in grado non solo di rappresentare l'insieme di volti, ma anche di effettuare operazioni di confronto molto rapide su di esso. L'algorit-

mo utilizzato per il confronto è tipicamente grid search (un algoritmo in grado di effettuare una ricerca su un vettore bidimensionale).

- Acquisizione immagine del soggetto
- Face Detection

  Individuare il volto all'interno dell'immagine aiuta a isolare i punti dell'immagine non interessanti, velocizzando le operazioni successive.

- Permutazioni sull'immagine acquisita

  Nel caso di OpenFace si tratta unicamente dell'isolamento della parte interessante.

- Rappresentazione matematica del volto
- Identificazione

  OpenFace assegna un'identità al soggetto attraverso la rete neurale assegnando inoltre un valore probabilistico ad ogni identità.

Nel caso di Face Authentication l'unico punto a cambiare è l'identificazione, la rete neurale non viene utilizzata, al suo posto viene fatto un confronto con le rappresentazioni delle altre foto del soggetto per stimarne la distanza e, fissato un valore di soglia, decidere se il soggetto corrisponde a quello dichiarato.

2.2 Training

Seguendo il flow specificato ho iniziato realizzando un trainer per generare un SVM in grado di eseguire la Face Recognition.

Il compito del trainer è quello di calcolare gli autovalori delle varie foto in input, si occupa inoltre di popolare un database contenente le principali informazioni relative alle immagini (quali nome del soggetto corrispondente, path dell'immagine sul sistema e ovviamente l'autovettore collegato ad ogni foto).

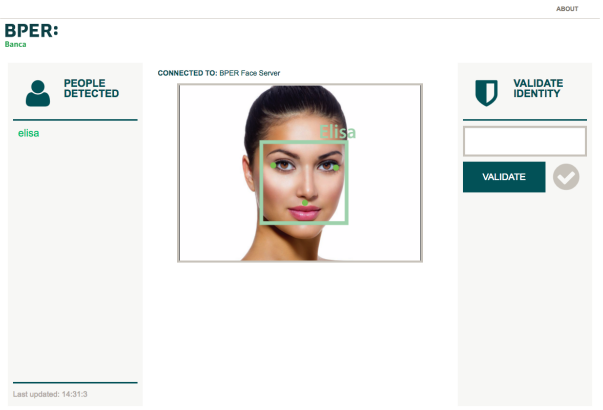

Figura 2.1: Anteprima client

2.3 Server

Volendo realizzare un'implementazione web della Face Recognition ho realizzato un server in grado di gestire la comunicazione con il database ed effettuare il riconoscimento, così facendo ho inoltre reso possibile l'utilizzo di un solo dispositivo dedicato alle operazioni più pesanti, permettendo l'esecuzione del software su client che non richiedono grandi caratteristiche tecniche.

2.4 Client

Il client è una pagina web, realizzata seguendo i più recenti standard e garantendo la responsiveness, la capacità di adattarsi a schermi di ogni dimensione, in modo da poter essere utilizzato su tutti i dispositivi.

Le aree della pagina sono facilmente identificabili: il menu di sinistra riporta l'username dei soggetti rilevati all'interno dell'immagine acquisita,

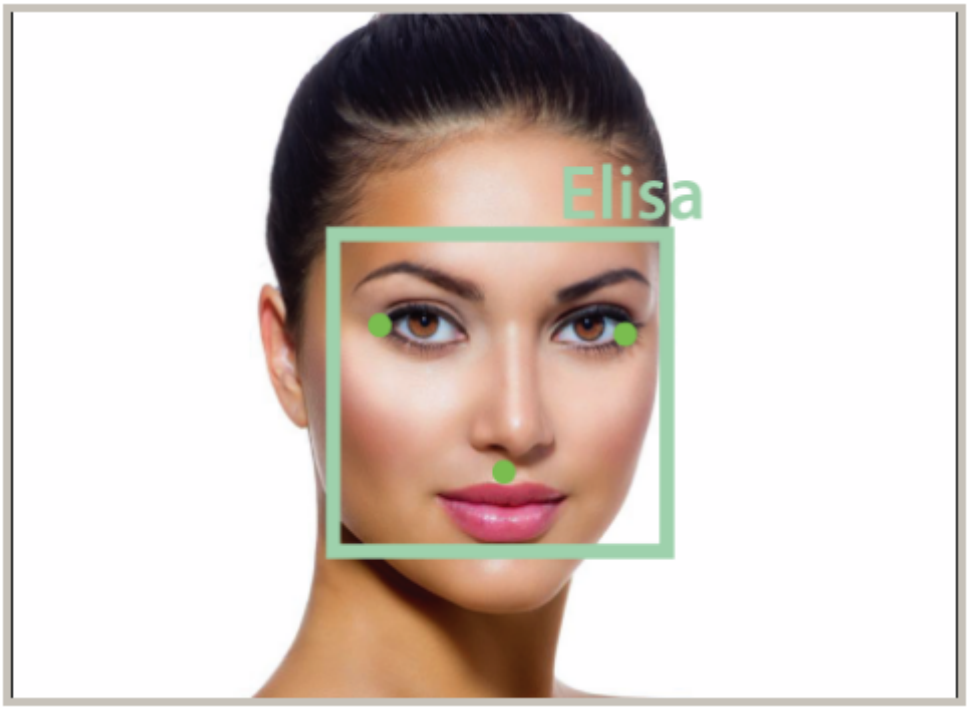

Figura 2.2: Volto rilevato

la parte centrale mostra i risultati della Face Recognition in tempo reale, mentre a destra è disponibile l'interfaccia per effettuare un'autenticazione.

Il campo di testo prevede un completamento automatico e permette all'utente di scegliere uno degli username presenti nel database contro cui effettuare l'autenticazione, l'icona immediatamente alla destra del pulsante ne indica il risultato una volta ottenuto.

La parte centrale della pagina riporta il volto acquisito dalla webcam, figura 2.2.

Decostruendo l'immagine si nota un riquadro attorno al volto e dei punti sul viso. Questi artefatti vengono aggiunti dal processo di Face Detection che prima di tutto cerca di trovare i tratti tipici di un volto (angoli degli occhi, naso e bocca), indicandoli con un punto, poi di capire dove finisce il volto, tipicamente prendendo l'area subito sopra le sopracciglia e terminando

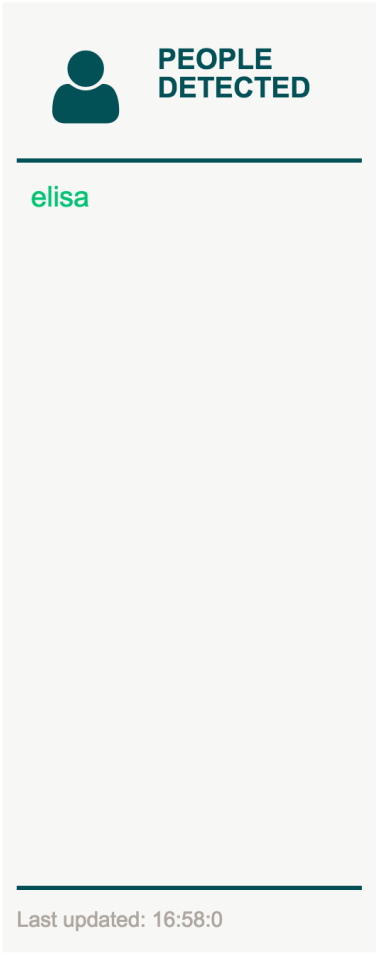

Figura 2.3: Persone rilevate

alla fine del labbro inferiore. Tutte queste informazioni sono riportate anche nell'immagine per chiarezza.

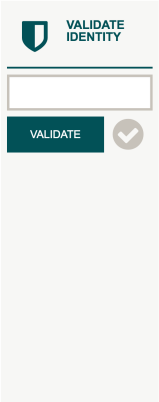

Figura 2.4: "Schermata di partenza"

Figura 2.5: "Inserimento identità"

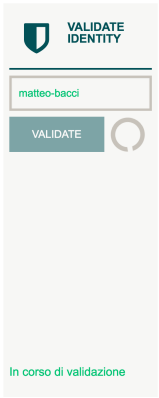

Figura 2.6: "Avvio autenticazione"

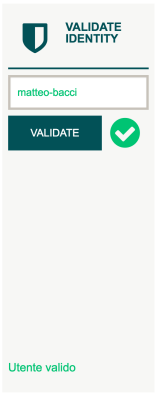

Figura 2.7: "Autenticazione riuscita"

La figura 2.3 riporta un esempio della sidebar e l'identificazione di un soggetto con il timestamp dell'ultimo rilevamento, per accertarsi che i dati visualizzati siano conformi a quelli acquisiti.

Le immagini dalla 2.4 alla 2.7 rappresentano il flusso di autenticazione, prima di tutto va scelta un'identità tra quelle presenti nel dataset, il client fornisce suggerimenti tramite autocomplete. Una volta premuto il pulsante *Validate* comincia l'autenticazione vera e propria che disabilita il pulsante fino alla ricezione del risultato.

Oltre al feedback grafico, sul fondo è disponibile uno stato testuale che

indica se l'autenticazione è andata a buon fine o meno.

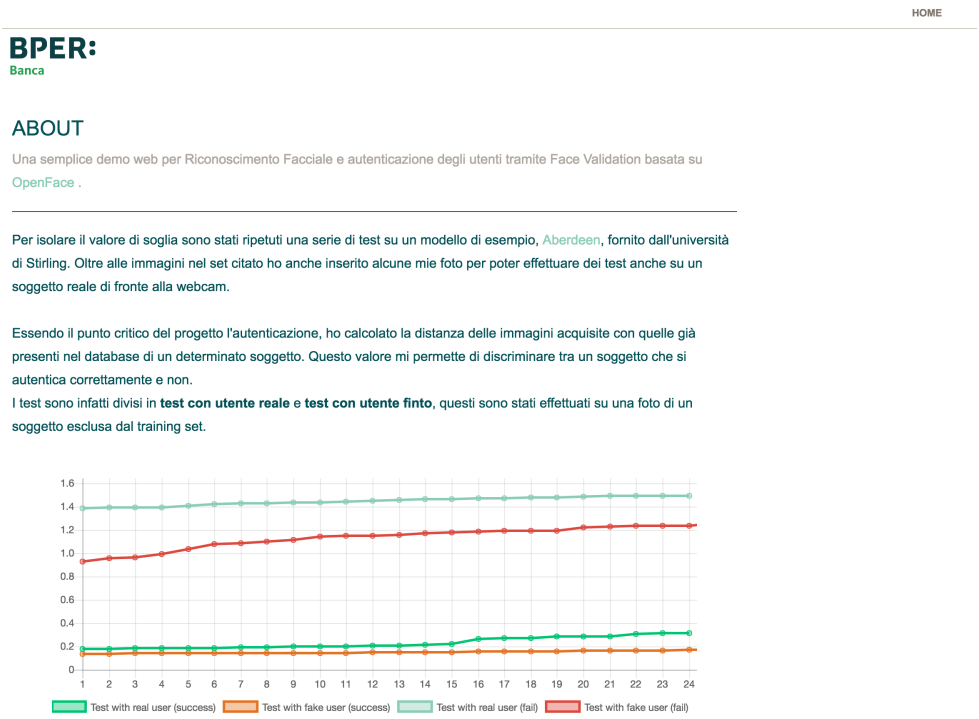

Figura 2.8: Pagina about

La pagina *About* (immagine 2.8) infine riporta una breve descrizione del progetto e i grafici interattivi riguardo alcuni test effettuati riportati anche più avanti nel documento.

Capitolo 3

Dettagli implementativi

In questo capitolo entrerò nei dettagli più tecnici del progetto, analizzando nuovamente il lato implementativo dei vari punti esposti in precedenza e la struttura generale del progetto.

Riporto l'albero delle directory di competenza del progetto, così da avere una visione d'insieme della struttura.

```
/
├── data
│   ├── output
│   └── train_img
├── faceapi
├── trainer
└── web
    ├── client
    └── server
```

La cartella data contiene due sottocartelle, train_img contiene le immagini su cui effettuare il training (più dettagli saranno indicati nella sezione dedicata al trainer), output contiene invece i file necessari per la generazione dell'SVM, l'SVM e il database.

La cartella faceapi è una libreria che ristruttura OpenFace per renderlo più simile ad una libreria Python.

Le cartelle trainer e web corrispondono alle due sezioni spiegate in precedenza e verranno spiegate dettagliatamente nelle sezioni seguenti.

3.1 Trainer

Il trainer è realizzato in Python e ha come unico compito quello di generare un modello SVM su una serie di foto date in input da una directory prestabilita.

OpenFace fornisce modelli su cui effettuare il training, quello che ho scelto di utilizzare per il training è *nn4.small2.v1.t7*, che ha ottenuto i risultati migliori nel test LFW come riportato nella figura 1.1, sebbene sia possibile usare tutti gli altri modelli di OpenFace con un'opzione da riga di comando.

Seguendo la documentazione fornita da OpenFace, il trainer sottostà ad alcune regole: ogni persona deve avere una subdirectory dedicata, all'interno della quale ci devono essere almeno 5 immagini distinte del soggetto e le immagini devono avere estensione png o jpg.

Uniformando le dimensioni delle immagini è possibile migliorare le prestazioni del software: di default il trainer effettuerà un crop delle immagini mantenendo l'area interessante in un quadrato di 96 per 96 pixel.

Per la demo ho utilizzato un database di volti disponibile online, nello specifico il set aberdeen dell'università di Stirling, da cui ho tolto tutte le immagini fuori dagli standard (ad esempio soggetti di profilo, foto con scarsissima luminosità che previene la detection, ecc.) ed aggiunto alcune mie foto al fine di effettuare test su soggetti reali.

3.2 Web

La cartella web contiene a sua volta due sottocartelle: client e server, che corrispondono alle rispettive funzioni.

3.2.1 Client

Il client è un'interfaccia web per il riconoscimento facciale, la comunicazione col server avviene tramite JavaScript.

L'acquisizione dell'immagine avviene tramite webcam utilizzando il metodo getUserMedia disponibile in tutti browser moderni.

All'apertura viene richiesto all'utente il permesso di utilizzare i dispositivi necessari dopodiché viene acquisito uno stream video.

Partendo dal flusso vidoe vengono estratti i vari frame e rappresentati su un oggetto canvas, un elemento HTML5 che permette rappresentazioni grafiche a vanzate.

La connessione con il server avviene tramite protocollo websocket, basato su TCP, ma che condivide con il protocollo HTTP soltanto il concetto di handshake, garantendo una trasmissione rapida di dati tra client e server (o viceversa).

Alla connessione col server, il client riceve l'elenco di username disponibili nel database, in modo da poter popolare l'autocomplete per l'authentication e procede quindi a inviare un frame dallo stream video.

Il frame viene inviato al server come *data URI*, ovvero in un formato in grado di trasmettere dati come se fossero risorse esterne.

Inviata un'immagine, il client attende la risposta dal server che conterrà l'immagine arricchita delle informazioni sul volto riconosciuto.

Il client può modificare lo stato del server solo richiedendo un'autenticazione.

I tipi di messaggi previsti sono:

- NULL, il server non ha effettuato il riconoscimento, il client invia una nuova immagine;
- PROCESSED, il server ha ricevuto l'immagine ed è pronto a riceverne una nuova;
- IDENTITIES, il server ha ricevuto l'immagine e restituisce l'elenco di soggetti all'interno;
- ANNOTATED, il server ha restituito l'immagine arricchita delle informazioni richieste;

- DB_LIST, l'elenco dei dati sul server interessanti per il client;

3.2.2 Server

In questa demo il server si occupa interamente dell'aspetto legato alla Face Recognition, dalla detection all'identificazione. Ho separato la gestione di autenticazione ed identificazione attraverso un comando in grado di cambiare lo stato del server da predefinito (identificazione) a autenticazione.

L'identificazione avviene per ogni frame ricevuto assegnando ad ogni immagine un'identità e la probabilità relativa. Non ho fissato alcun valore di soglia per cui OpenFace tenta sempre di assegnare un'identità ai volti trovati.

Il server passa in stato di autenticazione quando riceve un messaggio di tipo VALIDATING, acquisisce quindi i 5 frame successivi, ne calcola la distanza con le immagini del soggetto dichiarato e ne restituisce la media.

Nel caso della identificazione non ho utilizzato alcun valore di soglia, restituendo di fatto vari falsi positivi, OpenFace restituisce sempre l'identità più vicina a quella del soggetto ricevuto in input, ma non ha alcun modo per definire il grado di certezza.

Nel caso dell'autenticazione invece ho fissato un valore di soglia costante, da applicare alla distanza euclidea tra il volto in input e quelli corrispondenti al soggetto dichiarato, deciso in base alle valutazioni fatte nel capitolo seguente, nel quale introdurrò anche una possibile soluzione per aggirare i falsi positivi dell'identificazione.

Capitolo 4

Valutazione

Nel seguente capitolo esporrò i test fatti per decidere il valore di soglia legato all'autenticazione, i limiti e i problemi legati all'implementazione corrente e i possibili miglioramenti delle implementazioni future.

4.1 Test

La difficoltà più grossa legata all'implementazione della demo è legata alla definizione del valore di soglia da utilizzare per decidere quando un utente corrisponda effettivamente all'identità dichiarata.

A tale scopo ho effettuato alcuni test che mi hanno aiutato a definire tale valore.

Per iniziare ho diviso il database in due sezioni, training set e testing set, ovvero ho estratto dal database alcune immagini che ho utilizzato come unicamente per eseguire test e inserito alcune mie foto per ripetere il test su un soggetto reale di fronte alla webcam del dispositivo.

I test effettuati consistono nel calcolo della distanza euclidea tra la rappresentazione dell'immagine in input e quelle del soggetto dichiarato. I dati prodotti da queste simulazioni sono stati una base fondamentale per decidere il valore di soglia che sta al centro dell'autenticazione.

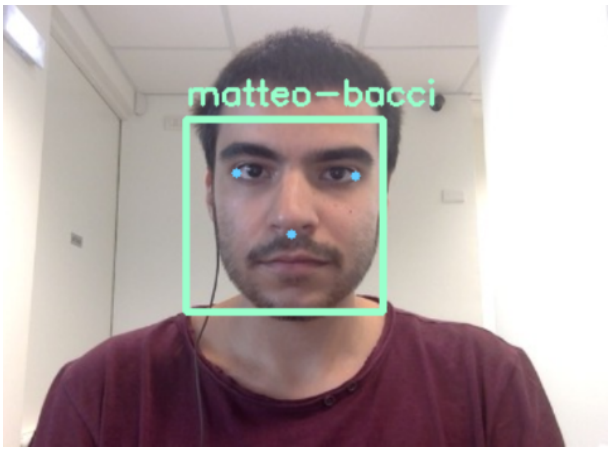

Figura 4.1: Soggetto reale

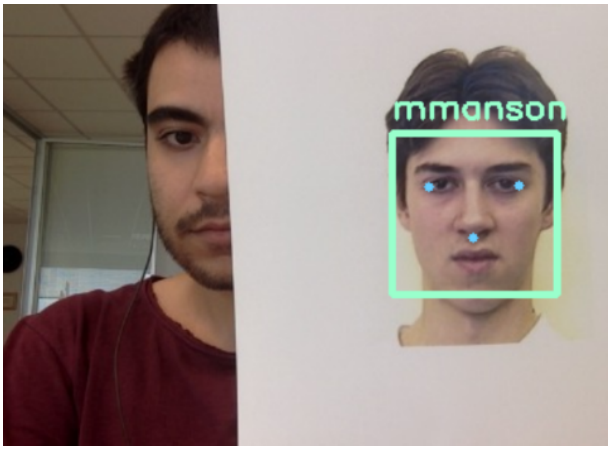

Figura 4.2: Soggetto fittizio

L'immagine 4.1 riporta un esempio di utente reale, fisicamente presente di fronte alla webcam, mentre l'immagine 4.2 riporta un utente fittizio, l'immagine stampata di un soggetto del testing set riportata di fronte alla webcam, in questo secondo caso avendo il volto per metà coperto rileva solo il soggetto fittizio.

Per entrambi i soggetti sono stati eseguiti due test separati:

- L'utente esegue l'autenticazione contro la propria identità;
- L'utente esegue l'autenticazione contro un'identità errata;

Sono state prese in considerazione 24 iterazioni, ovvero 24 tentativi di autenticazione verso un singolo soggetto. I risultati rappresentati nel grafico corrispondono alla distanza nell'asse delle ordinate, mentre l'asse delle ascisse rappresenta semplicemente l'elenco delle iterazioni. I risultati sono ordinati per distanza, in modo da vedere chiaramente minimo e massimo per ogni test.

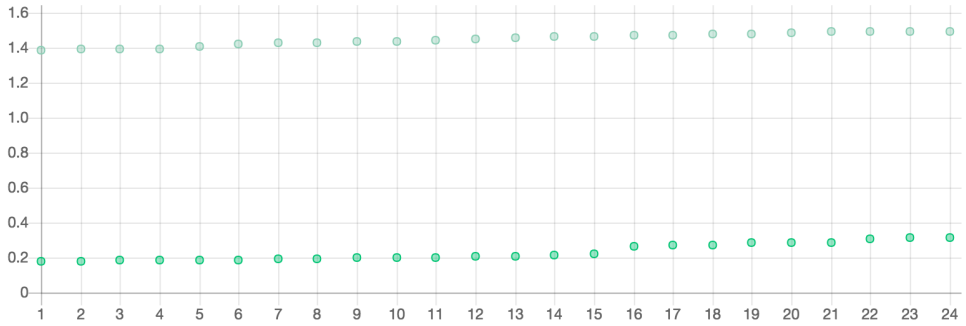

Figura 4.3: Test utente reale

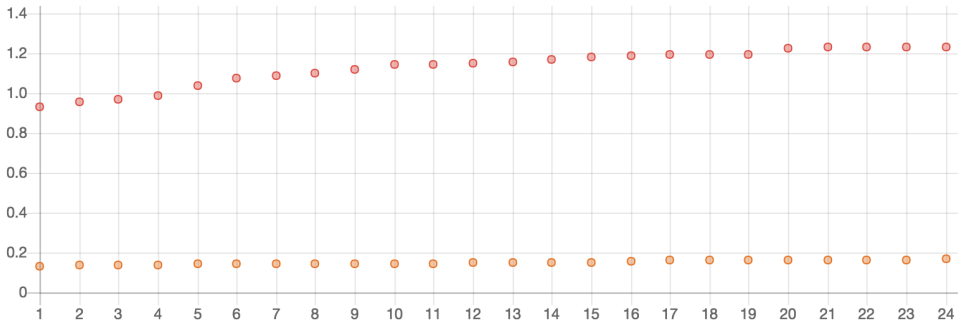

Figura 4.4: Test utente fittizio

La figura 4.3 riporta un confronto tra i risultati dell'utente reale per i due test.

Già da questo primo risultato si nota chiaramente essere presente un grande divario (circa di 1.20 punti) tra un utente riconosciuto e uno non riconosciuto, si ottengono inoltre solo valori superiori ad 1.0 nel caso di autenticazione fallita, dando quindi un largo margine di sicurezza.

Questi risultati sono rafforzati anche dal confronto tra i risultati dell'utente fittizio dove la distanza tra i test riusciti e non scende a 0.75, ma l'autenticazione fallita resta attorno a 1.0.

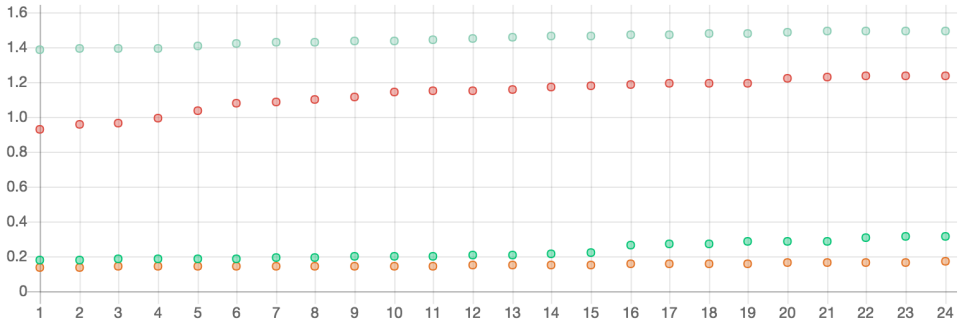

Figura 4.5: Confronto test

La figura 4.5 infine mette a confronto tutti i risultati ottenuti.

I risultati della simulazione non si sono discostati eccessivamente dalle previsioni, fatte in base alle stime dichiarate da OpenFace, ma ho deciso di utilizzare come valore di soglia 0.45, per avere un buon margine di sicurezza.

4.1.1 Punti critici

Un grosso problema di questa implementazione è dovuto al tempo: utilizzando solo la distanza come fattore determinante per l'autenticazione, l'utente invecchiando potrebbe non essere più riconosciuto.

4.2 Miglioramenti ed implementazioni future

Avvenuta la realizzazione della demo web, i test e rilevati quali fossero i punti deboli del progetto, prendo in considerazione possibili miglioramenti ed implementazioni future.

4.2.1 Autenticazione ed età

Per aggirare il problema legato all'invecchiamento dell'utente, una possibile soluzione potrebbe essere arricchire il dataset con i dati relativi alle au-

tenticazioni riuscite, mantenendo l'insieme aggiornato man mano che l'utente lo utilizza.

Con il passare del tempo i dati resterebbero al passo con i cambiamenti dell'utente assicurando un'autenticazione più veritiera.

4.2.2 Riconoscimento

L'implementazione del riconoscimento è stata lasciata da parte, trovando la tecnologia ancora poco matura per essere distribuita in larga scala.

Il problema legato al riconoscimento è di due tipi, prima di tutto per ora è difficile identificare un soggetto come sconosciuto, ma una possibile soluzione potrebbe essere mettere a confronto l'identità presunta con l'immagine acquisita, forzando un'autenticazione.

Questa soluzione porterebbe sì ad identificazioni più sicure, ma rallenterebbe fortemente l'algoritmo.

In secondo luogo non sono da trascurare i problemi legati alla privacy, per poter effettuare il riconoscimento è necessario effettuare il training del modello, che richiede molte più informazioni rispetto a quelle necessarie per l'autenticazione.

4.2.3 App

Dopo aver creato questa demo web è nata l'esigenza di renderla accessibile al più alto numero di utenti.

È chiaro che negli ultimi tempi il mezzo più utilizzato per accedere a servizi online sia il proprio smartphone, la prima idea che verrà portata avanti è quella di realizzare un client mobile che si interfacci con il server.

Il server sarebbe quindi incaricato di ricevere l'immagine dal dispositivo, calcolarne la rappresentazione matematica, confrontarla con le rappresentazioni delle immagini nel dataset e restituire uno stato di successo o fallimento, scartando l'immagine, questa implementazione è al momento in sviluppo.

4.3 Scalabilità

La demo realizzata offre molte possibilità di miglioramento e di integrazione, utilizzando principalmente tecnologie in grado di essere eseguite e distribuite facilmente su qualsiasi piattaforma.

Il progetto ricalca il metodo di distribuzione di OpenFace che utilizza Docker[1].

Docker è uno strumento innovativo, rapido e incredibilmente potente che permette di rendere più semplice la distribuzione e l'esecuzione di software utilizzando dei container, ovvero dei pacchetti in cui, oltre al codice sorgente o gli eseguibili, l'utente finale troverà tutte le risorse e librerie già configurate e pronte per lanciarlo.

Il vantaggio di uno sviluppatore che decide di usare Docker è quello di poter lavorare in un ambiente indipendente da quello della macchina su cui si trova, similmente alle Virtual Machine, ma interfacciandosi con risorse native del sistema operativo ed ottenendo quindi performance molto più simili a quelle del proprio sistema.

Inoltre il client è realizzato utilizzando strumenti di sviluppo moderni come Gulp[2] e Browserify[3] che mi hanno permesso di mantenere il codice estremamente modulare e soddisfare tutte le esigenze di un sito web moderno per rapidità e usabilità.

Nello specifico Gulp permette di gestire dei processi automatici per build, struttura e pulizia del codice, mentre Browserify è a sua volta richiamato da Gulp per poter gestire al meglio gli script JavaScript necessari e le loro dipendenze.

Lo stile della pagina, per quanto semplice, è realizzato utilizzando less[4], una sorta di css avanzato, anch'esso compilato grazie a Gulp.

Tutte queste scelte di sviluppo rendono il codice più semplice da mante-

[1] http://docker.com
[2] http://gulpjs.com/br
[3] http://browserify.org/
[4] http://lesscss.org/

nere e da fruire per un altro sviluppatore.

Capitolo 5

Conclusioni

Con questa dissertazione ho voluto analizzare gli strumenti di Riconoscimento Facciale e mostrare un'implementazione per l'utilizzo della tecnologia in ambito bancario.

L'obiettivo prefissato era quello di mostrare le capacità di queste tecnologie applicate a un utilizzo sia per gli utenti della banca che per i propri dipendenti, che devono autorizzarsi per accedere ai propri terminali.

Ho sviluppato un servizio web che permette la comunicazione di un qualunque dispositivo client che disponga di una fonte di immagini con un server che ne gestisce il riconoscimento o l'autenticazione.

La netta divisione tra le informazioni del client e quelle del server è un punto importante che permette di risolvere buona parte dei problemi legati alla privacy degli utenti finali.

Il prototipo da me sviluppato svolge i principali compiti del Riconoscimento Facciale: rilevamento, identificazione e validazione; restando allineato alle valutazioni di precisione dichiarate dai software già presenti.

Dai test svolti sono emersi risultati incoraggianti che mi hanno permesso di definire una soglia affidabile in grado di tener conto sia dell'aspetto legato alla sicurezza che della fruibilità del servizio.

Una futura versione aggiungerà inoltre la possibilità di interfacciarsi al server con il proprio smartphone ed è attualmente in sviluppo.

Dai primi utilizzi di questa implementazione ho notato un netto miglioramento nell'autenticazione dovuto alla qualità dell'immagine acquisita dalla fotocamera dello smartphone, tecnicamente superiore alle webcam.

Oltre a questo miglioramento è prevista la possibilità di aggiornare il dataset con nuove foto, acquisite ad ogni autenticazione avvenuta con successo, questo permetterà al servizio di restare allineato all'aspetto dell'utente nel corso del tempo.

Le tecnologie di Face Recognition, sebbene non del tutto mature, sono in rapida crescita e questa implementazione ne dimostra le grandi potenzialità.

Bibliografia

[1] T. Kohonen *Self-organization and Associative Memory.* Springer-Verlag, 1989.

[2] DOI: 10.1109/CVPR.1991.139758 Mattew A. Turk and Alex P Pentland *Face recognition using eigenfaces.* Computer Vision and Pattern Recognition '91, Pittsburgh, 3 - 6 Giugno 1991, IEEE Comput. Soc. Press, pagine 586-591, 1991.

[3] Peter N. Belhumeur, Joao P. Hespanha e David J. Kriegman *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection* IEEE Transactions on pattern analysis and machine intelligence, Volume 19, Issue 7, IEEE, pagine 711-720, 1997. DOI: 10.1109/34.598228

[4] Bowyer, Kevin W. *Face recognition technology: security versus privacy* In IEEE Technology and society magazine, Volume 23, Issue 1, IEEE, pagine 9-19, 2004. DOI: 10.1109/MTAS.2004.1273467

[5] Sadeghi, Ahmad-Reza, Thomas Schneider, and Immo Wehrenberg. *Efficient privacy-preserving face recognition.* International Conference on Information Security and Cryptology, Seoul, Korea, 2 - 4 Dicembre, 2009, Berlin Heidelberg, pagine 229-244, 2009. DOI: 10.1007/978-3-642-14423-3_16

[6] Ratha, Nalini K., Jonathan H. Connell, and Ruud M. Bolle. *Enhancing security and privacy in biometrics based authentication systems.* IBM

systems Journal, Volume 40, Issue 3, IBM, pagine 614-634, 2001. DOI: 10.1147/sj.403.0614

[7] Sanderson, Conrad, and Kuldip K. Paliwal. *Polynomial features for robust face authentication.* International Conference on Image Processing, Rochester, New York, USA, 22 - 25 Settembre, 2002, IEEE, pagine 997-1000, 2002. DOI: 10.1109/ICIP.2002.1039143

[8] B. Amos, B. Ludwiczuk, M. Satyanarayanan, *Openface: A general-purpose face recognition library with mobile applications*, CMU School of Computer Science, Tech. Rep., 2016.

[9] Huang, Gary B. *Labeled faces in the wild: A database forstudying face recognition in unconstrained environments.* University of Massachusetts, Amherst, Technical Report 07-49, Volume 1, Issue 2, 2008.