SmartRingbell

AUTORI

Alessandro Lillo

Amedeo D’Amelio

# INDICE

[SmartRingbell 1](#_Toc43586304)

[INDICE 2](#_Toc43586305)

[Introduzione 3](#_Toc43586306)

[Casi d’uso 4](#_Toc43586307)

[Manuale tecnico 5](#_Toc43586308)

[Raccolta dei dati 5](#_Toc43586309)

[Volto 5](#_Toc43586310)

[Voce 6](#_Toc43586311)

[Pre-Processing 7](#_Toc43586312)

[Volto 7](#_Toc43586313)

[Voce 8](#_Toc43586314)

[Training 9](#_Toc43586315)

[Volto 9](#_Toc43586316)

[Voce 11](#_Toc43586317)

[Recognition 14](#_Toc43586318)

[Volto 14](#_Toc43586319)

[Voce 15](#_Toc43586320)

[Sistema di notifiche 16](#_Toc43586321)

# Introduzione

Smart Ringbell nasce dall’intenzione di applicare tecniche biometriche all’ambito della sicurezza domestica. Si presenta come un citofono smart che prevede l’immissione di due tipi diversi di tratto biometrico: la voce ed il volto. Attraverso un riconoscimento di questi ultimi sarà possibile consentire l’accesso in casa solo a chi sarà registrato come utente nel sistema. Il riconoscimento per i tratti avverrà in maniera separata ed indipendente in modo tale che se una delle due metodologie di riconoscimento non dovesse andare a buon fine, la “compromessa” non infici con le performance del sistema. In situazioni di dubbio o rigetto, la possibilità che la porta si apra non sarà esclusa a priori, ma sarà lasciata alla discrezione del padrone di casa attraverso l’utilizzo di una apposita app Android.

All’apertura dell’applicativo sarà visualizzato un menu dal quale si potrà scegliere tra le seguenti azioni:

* Aggiungi un utente
* Modifica un utente
* Suona il campanello (Recognition)
* Elimina un utente
* Visualizza gli utenti registrati
* Esci dal programma

# Casi d’uso

La progettazione ed implementazione è stata effettuata tenendo conto dei seguenti casi d’uso.

1. Utente in fase di adding: l’utente immette campioni di tipo audiovisivo tali da generare un modello da utilizzare durante la fase di test.
2. Utente in fase di modifica dei dati salvati: l’utente può modificare i dati già immessi in un modello sovrascrivendoli. La sovrascrittura permetterà di ricreare un modello completamente nuovo eliminando il precedente.
3. Utente in fase di eliminazione di dati salvati: l’utente può eliminare un modello precedentemente creato immettendo lo username corrispondente.
4. Utente riconosciuto in fase di test: l’utente immette un campione audiovisivo che verrà confrontato con i modelli presenti nel data set e riconosciuto attraverso un confronto sia con i modelli generati utilizzando i campioni sonori sia con i modelli generati utilizzando i campioni video.
5. Utente non riconosciuto in fase di test: l’utente immette un campione audiovisivo che verrà confrontato e rigettato con i modelli presenti nel data set. Precisamente, entrambi i campioni non supereranno una soglia di somiglianza con i modelli presenti del data set tale da poter rivendicare un’identità. È inviata una notifica al dispositivo Android dell’utente proprietario di casa informandolo della presenza di un potenziale sconosciuto alla sua porta. Attraverso l’utilizzo dell’app Android sarà possibile visualizzare la foto del possibile sconosciuto e decidere se aprire o meno la porta a distanza. La foto scattata al possibile sconosciuto sarà cancellata appena presa una decisione. Per rispettare la privacy dell’utenza non sarà neanche possibile fare screenshots all’interno dell’applicazione. Una volta che l’utente proprietario avrà deciso se aprire o meno la porta arriverà un messaggio di conferma sia sullo schermo del proprio dispositivo Android, sia al citofono vero e proprio.
6. Utente riconosciuto parzialmente in fase di test: l’utente immette un campione audiovisivo che verrà confrontato e riconosciuto parzialmente con i modelli presenti nel data set. Ciò implica che l’utente avrà superato solo uno dei due riconoscimenti previsti per l’accesso. Il riconoscimento parziale impedisce un rigetto assoluto dell’utente che potrà accedere solo grazie ad una conferma avvenuta da parte dell’utente proprietario di casa. Attraverso l’utilizzo dell’app Android sarà possibile visualizzare la foto della persona alla porta e decidere se aprire o meno quest’ultima a distanza. La foto scattata sarà cancellata appena presa una decisione. Per rispettare la privacy dell’utenza non sarà neanche possibile fare screenshots all’interno dell’app. Una volta che l’utente proprietario avrà deciso se lasciare o meno aperta la porta arriverà un messaggio di conferma sia sullo schermo del proprio dispositivo Android sia al citofono vero e proprio.
7. Mostra tutti gli utenti registrati: vengono mostrati all’utente gli username salvati.

# Manuale tecnico

## Raccolta dei dati

All’avvio della fase di raccolta dei dati sarà chiesto all’utente di scegliere un nome univoco con cui saranno salvati i suoi modelli, i quali, saranno usati per effettuare il confronto in fase di riconoscimento.

### Volto

La rilevazione di oggetti che utilizzano i cascade classifiers (classificatori a cascata) si basano su Haar che è un metodo di rilevamento di oggetti molto efficace. È un approccio che si basa sul machine learning. La “cascade function” viene allenata attraverso molte immagini positive (contengono ciò che vogliamo identificare) e negative (immagini che non contengono l’elemento che vogliamo identificare). Fatto questo, la funzione viene utilizzata per rilevare gli oggetti in altre immagini. Dopo aver allenato il classificatore, bisogna estrarre le features. Per fare questo vengono utilizzate le funzioni di Haar. Ogni feature è un singolo valore ottenuto sottraendo la somma dei pixel sotto il rettangolo bianco dalla somma dei pixel sotto il rettangolo nero. Tutte le dimensioni e posizioni possibili, vengono utilizzate per calcolare tutte le features. I calcoli da effettuare sono tantissimi anche se si ha una finestra molto piccola. Ad esempio, una finestra 24\*24 produce oltre 160000 funzioni. Per ridurre il tempo di computazione, viene utilizzata un’immagine integrale. L’immagine integrale è una struttura dati per il calcolo rapido della somma dei valori in un sottoinsieme rettangolare di una griglia.

Per la realizzazione di questo progetto non è stato creato un classificatore ad-hoc, ma è stato utilizzato il classificatore “haarcascade\_frontalface\_default.xml” fornito dalla libreria OpenCV. Questo classificatore è in grado di identificare solo i volti posizionati di fronte alla videocamera. È stato utilizzato questo classificatore, perché è abbastanza “leggero” e non ha bisogno di un hardware molto performante per dare risultati in tempi accettabili.

#### Implementazione

La chiamata alla funzione che consente di identificare la presenza di un volto all’interno dell’immagine è la seguente.

Immagine che contiene telefono

Descrizione generata automaticamente

I parametri sono:

* image: è l’immagine che deve essere processata. Deve essere in scala di grigi.
* scaleFactor: determina il rapporto tra le misure dei volti che devono essere analizzati. In questo caso il valore è 1.05 che non è il più veloce, ma molto accurato.
* minNeighbors: è un parametro che specifica il numero di vicini che ogni rettangolo candidato deve avere per conservarlo. Un numero più alto fornisce meno falsi positivi. Si può considerare come il grado di confidenza del face detector.

### Voce

L’acquisizione dell’audio avviene con una frequenza di campionamento pari a 44100 Hz, una frequenza molto utilizzata nell’ambito dell’analisi audio. Poiché 20000 Hz è il limite per l’udito umano, si realizza che un audio deve essere campionato ad almeno 40000 Hz, questo per la legge di Nyquist-Shannon. La ragione sta nel ridurre l’entropia e così facendo garatire una più accurata riproduzione delle onde originali. La riproduzione del suono, come in altri sistemi chiusi, deve fare i conti con l’entropia. In una situazione di riproduzione del suono, le onde vanno attraverso diverse trasduzioni di tipo elettrico, magnetico o meccanico. Questo processo crea entropia; nel caso del suono ciò comporta rumore. Conseguentemente, per ridurre il rumore, l’audio digitale è tipicamente campionato a 44100 Hz. Teoricamente, più alta è la frequenza di campionamento, più si allontana la possibilità che l’orecchio umano percepisca il rumore. Il campionamento a 44100 Hz risulta un ottimo bilanciamento tra le limitazioni di storage e le richieste in merito a qualità. La registrazione dei file .wav avviene attraverso l’utilizzo della funzione rec della libreria sounddevice.

#### Implementazione

La chiamata alla funzione che consente di avviare una registrazione audio è la seguente.

Immagine che contiene bottiglia, vicino, nero, rosso

Descrizione generata automaticamente

I parametri sono:

* frames: secondi da registrare
* samplerate: frequenza di campionamento
* channels: numero di canali da registrare

## Pre-Processing

### Volto

Durante la fase di pre-processing viene preso il frame e si verifica che il volto sia delle dimensioni minime richieste. Questo serve per scartare i frame che contengono volti troppo piccoli, i quali sono o dei falsi positivi, o dei volti troppi piccoli che non vanno presi in considerazione. Il frame, se contiene un volto che rispetta le dimensioni minime richieste, viene convertito da BGR (Blue, Green, Red) a RGB (Red, Green, Blu). Successivamente, viene effettuato un crop dell’immagine in cui viene eliminato tutto ciò che non è il volto e infine viene effettuato un resize dell’immagine su scala 96\*96 poiché è la grandezza richiesta dal modello utilizzato.

#### Implementazione

Immagine che contiene portatile, schermo, tavolo, computer

Descrizione generata automaticamente

Immagine che contiene cibo

Descrizione generata automaticamente

### Voce

Sugli audio sono stati applicati rimozione del silenzio e riduzione del rumore. La rimozione del silenzio avviene attraverso l’utilizzo della funzione rms\_silence\_filter appartenente alla libreria speaker\_verification\_toolkit la quale adatta l’audio sulla base di una soglia di treshold fissata. La funzione di noise removal è stata implementata attraverso l’utilizzo di un butterworh filter applicato attraverso la libreria scipy.signal. Il filtro si applica ottenendo un grado di smoothing del grafico rappresentante l’audio, nel nostro caso è 1 per permettere uno smooth lento, e una frequenza di cutt-off normalizzata ottenuta scegliendo una frequenza di cut-off e dividendola per la frequenza di Nyquist. Prima di applicare queste due tecniche il dato viene letto attraverso la funzione load appartenente alla libreria librosa che si occupa di convertire l’audio in un vettore di floating point. Dalla suddetta funzione otteniamo anche un intero maggiore di zero che indica il tasso di campionamento del nostro file wav.

#### Implementazione

Immagine che contiene screenshot, schermo, nero, tavolo

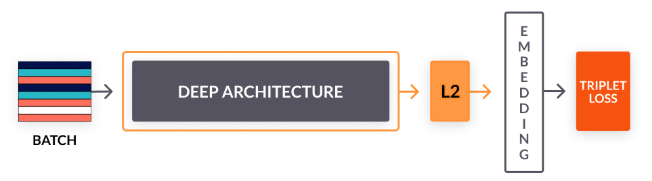
Descrizione generata automaticamente

## Training

### Volto

Quando viene identificato un volto, il frame viene processato. Durante questa fase vengono processate circa inserire il numero preciso immagini.

In questo progetto, viene utilizzato l’algoritmo di Google Facenet, un framework molto usato per l’implementazione di moderne applicazioni di riconoscimento facciale. FaceNet utilizza una tecnica chiamata “apprendimento one-shot”. La sua rete è composta da un livello di input batch e una rete neurale convoluzionale profonda (CNN) seguita da normalizzazione L2.



La rete neurale convoluzionale di FaceNet si basa sui pixel dell’immagine come caratteristiche (features), anziché estrarli manualmente. L’idea principale alla base dell’algoritmo è quella di rappresentare un volto come un vettore a 128 dimensioni, mappando le caratteristiche di input ai vettori.

Poiché questi vettori sono rappresentati nello spazio vettoriale condiviso, è possibile utilizzare la distanza vettoriale per calcolare la somiglianza tra due vettori. Questa è una tecnica per calcolare quanto sono simili due facce.

L’ultimo stadio dell’architettura FaceNet è la “perdita di tripletta” (triplet loss), che minimizza la distanza tra un’ancora e un positivo noto (somiglianza tra i due volti), massimizzando al contempo la distanza tra l’ancora e un negativo noto (dissomiglianza).

Immagine che contiene oggetto, orologio

Descrizione generata automaticamente

* f(a) si riferisce alla codifica di output dell’ancora.
* f(p) si riferisce alla codifica in uscita del positivo.
* f(n) si riferisce alla codifica in uscita del negativo.
* ɑ è una costante utilizzata per assicurarsi che la rete non tenti di ottimizzare verso f(a) - f(p) = f(a) - f(n) = 0.
* [...]+ è uguale a max(0, somma).

###### Rete siamese

FaceNet è una rete siamese. Una rete siamese è un tipo di architettura di rete neurale che impara a distinguere tra due input. Ciò consente loro di apprendere quali immagini sono simili e quali no. Le reti siamesi sono costituite da due reti neurali identiche, ognuna con gli stessi pesi esatti. innanzitutto, ogni rete prende come input una delle due immagini ottenute in ingresso. Quindi, gli output degli ultimi layer di ciascuna rete vengono inviati a una funzione che determina se le immagini contengono la stessa identità. In FaceNet questo viene fatto calcolando la distanza tra le due uscite.

Immagine che contiene orologio

Descrizione generata automaticamente

#### Implementazione

Immagine che contiene screenshot, sedendo, nero, portatile

Descrizione generata automaticamente

Iniziamo inizializzando la rete con una forma di input di (3, 96, 96). Ciò significa che i canali Rosso-Verde-Blu (RGB) sono la prima dimensione del volume dell’immagine messa nella rete. Tutte le immagini che vengono inviate alla rete devono essere immagini di 96x96 pixel.

Successivamente, viene definita la funzione di “Perdita della Tripletta” definita precedentemente.

Adesso, è possibile compilare il modello di riconoscimento facciale usando Keras.

Immagine che contiene screenshot, nero, monitor, schermo

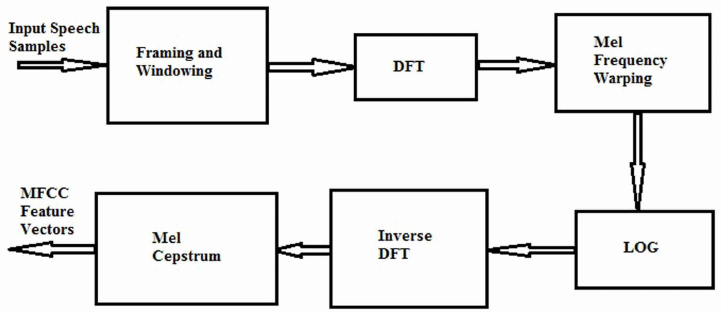
Descrizione generata automaticamente

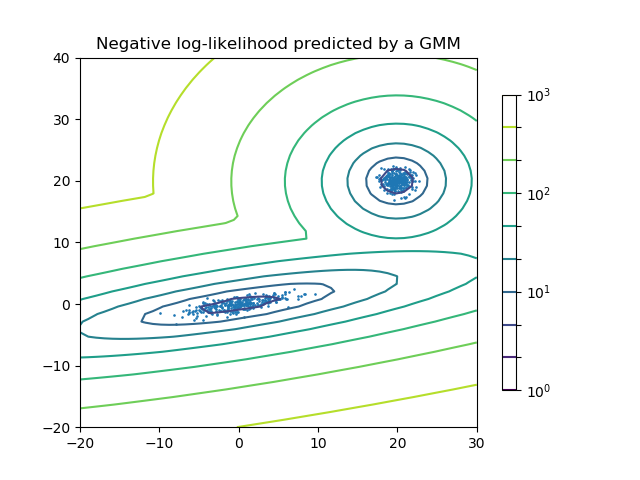
Per ogni immagine acquisito dalla webcam, i dati dell’immagine vengono convertiti in una codifica di 128 numeri float. Questo è fatto tramite la funzione img\_to\_encoding.

La funzione prende come parametro di ingresso l’immagine che viene convertita in RGB; questa, viene trasmessa alla rete di riconoscimento del volto. La funzione ritorna l’output dato dalla rete, che risulta essere la codifica dell’immagine. Il risultato viene poi inserito nel database.

### Voce

In questa fase ci concentriamo su due caratteristiche fondamentali per il nostro algoritmo: Mel frequency Cepstral coefficients (MFCC) e la sua derivata, chiamata Delta-MFCC. I MFCC sono coefficienti che collettivamente costruiscono e rendono tale un mel-frequency cepstrum, il quale si presenta come una rappresentazione dello spettro di potenza su scala short-term di un suono basata sulla trasformata lineare coseno di uno spettro di potenza su scala logaritmica. I MFCC si concentrano su una serie di calcoli che usano il Cepstrum. Per ottenere melcepstrum, il segnale audio è diviso in finestre applicando una funzione di windowing a intervalli fissi, per poi utilizzare la trasformata di Fourier. L’algoritmo continua mappando le potenze dello spettro ottenuto sulla scala mel usando finestre triangolari. Infine, si attua il logaritmo delle potenze per ognuna delle frequenze mel e facciamo la trasformata discreta coseno: gli mfcc saranno l’amplitude dello spettro risultante.



Successivamente un Gaussian Mixture Model è generato a partire dagli MFCC memorizzati per ogni singolo utente. Un GMM è un modello probabilistico che assume che tutti i data points sono generati da una combinazione di un numero finito di distribuzioni gaussiane con parametri sconosciuti. Ogni distribuzione gaussiana è spiegabile attraverso i seguenti parametri:

* Una media μ che definisce il centro.
* Una covarianza Σ che definisce la larghezza.
* Una probabilità mista π che definisce quanto grande o piccola sarà la funzione gaussiana.

#### Implementazione

Il calcolo degli MFCC avviene grazie alla funzione extract\_mfcc appartenente alla libreria speaker\_verification\_toolkit; il calcolo della derivata avviene per mezzo della funzione delta appartenente alla libreria librosa. Si sottolinea l’utilizzo della funzione scale appartenente alla libreria sklearn, la quale attua una standardizzazione del dataset lungo ogni asse, requisito comune per molti stimatori in ambito machine-learning. Se le feature non fossero standardizzate gli stimatori potrebbero dare risultati inaspettati. I coefficienti vengono infine combinati e salvati in un’apposita cartella.

Immagine che contiene tavolo, portatile

Descrizione generata automaticamente

Il Gaussian Mixture Model è generato grazie alla funzione GaussianMixture appartenente alla libreria sklearn, la quale opera su una matrice contenente tutti i coefficienti appartenenti ad un singolo utente, e della funzione fit che immette i dati nella matrice. Infine, il file viene salvato per mezzo della funzione dump appartenente alla libreria pickle. La funzione fit stima i parametri del modello grazie all’algoritmo EM (expectation maximization) il quale stima la massima somiglianza statistica attraverso un procedimento iterativo al fine di ottimizzare problemi. Si basa su due passaggi fondamentali: la fase di expectation prevede di stimare le variabili mancanti, la fase di maximization tenta di ottimizzare i parametri del modello per spiegare al meglio i dati.

Sklearn.mixture è un package che permette di stimare Gaussian Mixture Models a partire da dati quali il numero di componenti e il tipo di covarianza.

Immagine che contiene screenshot, portatile, tavolo, schermo

Descrizione generata automaticamente

## Recognition

### Volto

#### Implementazione

La funzione acquisisce un frame dalla videocamera, il quale viene elaborato dalla rete neurale.

Per trovare l’individuo viene esaminato il database e viene calcolata la distanza tra l’immagine acquisita e quella di ciascun individuo presente nel database. L’individuo con la distanza minima dalla nuova immagine viene quindi scelto come candidato più importante. Infine, dobbiamo determinare se l’immagine candidata e la nuova immagine contengano o meno la stessa persona. Dal momento che alla fine del nostro ciclo abbiamo determinato solo l’individuo più probabile.

Se la distanza è superiore a 0.52, l’utente non è presente nel database, altrimenti, abbiamo trovato una corrispondenza.

Il valore 0.52 è stato raggiunto attraverso una serie di tentativi. Con questo valore abbiamo pochi falsi positivi e qualche false rejection.

Immagine che contiene screenshot

Descrizione generata automaticamente

### Voce

#### Implementazione

Immagine che contiene tavolo, schermo, stanza

Descrizione generata automaticamenteDurante la fase di test i GMM sono letti e, per ognuno di essi, è ottenuta una somma di score mostrata su scala logaritmica e basata sul confronto tra i modelli presenti ed un audio immesso in input da un utente talvolta registrato talvolta sconosciuto al sistema. Lo score è ottenuto grazie alla funzione omonima appartenente al Modello. Lo score maggiore appartiene a chi, secondo il sistema, appartiene con più probabilità la voce immessa.

Se lo score maggiore dovesse superate il valore di 62 avremo una corrispondenza. Il valore 62 è stato raggiunto attraverso una serie di tentativi su un data set personale formato da modelli creati appositamente.

Per mostrare all’utente dei dati sensati sia è stato attuato uno scale attraverso la scala minmax che converte il valore più grande a 1, il più piccolo a 0 e gli altri valori in proporzione, sia la scala standard che standardizza features rimuovendo la media e scalando all’unità di varianza. Lo score standard di un campione x è calcolato con la formula dove:

* u è la media dei campioni di training
* s è la deviazione standard dei campioni di training.

Le normalizzazioni sono ottenute attraverso le funzioni scale e minmax\_scale appartenenti alla libreria sklearn.preprocessing.

## Sistema di notifiche

Per andare incontro a situazioni di dubbio da parte dell’applicazione e per dare possibilità all’utente di avere controllo sull’applicativo, è stato implementato un sistema di notifiche legato ad un app Android. Quest’ultimo trova il suo utilizzo in casi in cui il sistema non confermi l’identità dichiarata automaticamente dai campioni prelevati in input con sufficiente sicurezza. Infatti, in casi del genere, l’utente non solo riceverà una notifica, ma attraverso l’utilizzo dell’app potrà visualizzare la foto scattata all’eventuale sconosciuto alla porta. Si sottolinea come il messaggio inviato nella notifica vari in base al grado di certezza da parte del sistema che lo sconosciuto sia un utente registrato: se dovesse superare la soglia di accettazione legata solamente ad un tratto biometrico allora il messaggio sarebbe “qualcuno è alla porta”; se non dovesse superare nessuna soglia di accettazione il messaggio sarebbe “sconosciuto alla porta”. Inoltre, dall’applicazione l’utente potrà decidere se aprire o meno la porta, avendo pieno controllo della situazione. Infine, si sottolinea come non sia possibile effettuare screenshots dello screen del possibile sconosciuto per rispettarne la privacy: l’immagine sarà cancellata automaticamente dopo che l’utente avrà preso una decisione. La comunicazione tra il citofono e l’app android è stata testata attraverso l’utilizzo di un server locale.

#### Implementazione

L’implementazione delle notifiche passa per i seguenti due metodi.

Immagine che contiene screenshot

Descrizione generata automaticamente

## Testing

### Utenti

L’applicativo è stato testato da 16 utenti, riguardo i quali sono stati raccolti dati riguardo sesso e fascia d’età, in condizioni di utilizzo disparate. Si sottolinea come l’utilizzo in vari ambienti sia stato importantissimo per lo sviluppo dell’applicazione che nasce per un fine prettamente “esterno” e che quindi deve sapersi adattare a condizioni di rumore, luce e, generalmente, di ambiente molto vari tra loro.

Le fasce d’età considerate sono:

1. 11 – 22 anni
2. 23 – 35 anni
3. 36 – 50 anni
4. 51 – 60 anni
5. Più di 60 anni

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Utente 1 | Utente 2 | Utente 3 | Utente 4 | Utente 5 | Utente 6 | Utente 7 | Utente 8 |
| Sesso | M | F | F | F | M | M | M | F |
| Età | 4 | 4 | 3 | 5 | 5 | 4 | 2 | 3 |
|  | Utente 9 | Utente 10 | Utente 11 | Utente 12 | Utente 13 | Utente 14 | Utente 15 | Utente 16 |
| Sesso | M | F | M | M | F | M | M | F |
| Età | 4 | 4 | 1 | 3 | 1 | 2 | 3 | 1 |

### Volto

Il testing per la componente di face recognition appartenente a Smart Ringbell è stata testata in varie condizioni di luce, per comprendere quanto il sistema si adatti all’ambiente circostante. Soprattutto, si è tentato di effettuare training e testing in ambienti diversi, per capire se il sistema comprendesse la correlazione tra un utente e sé stesso. Il testing è stato effettuato anche in condizioni climatiche disparate ed in fasi della giornata diverse, tenendo traccia di quelli che sono stati i risultati al fine di ottenere una soglia di treshold ottimale. Si è sfruttata inoltre la funzionalità del “delete user” per effettuare test con utenti non registrati nel sistema, per comprendere i risultati dovuti ad un estraneo che si approccia in qualità di “sconosciuto”.

### Voce

Il testing per la componente di speaker recognition appartenente a Smart Ringbell è stata testata in varie condizioni di rumore. Specificatamente, ci si è soffermati sull’utilizzo in ambienti chiusi o aperti, con il rumore della strada in sottofondo, con il suono dovuto a condizioni climatiche quali vento o pioggia. Si è testata anche una casistica legata al silenzio dell’utente. Si è sfruttata inoltre la funzionalità del “delete user” per effettuare test con utenti non registrati nel sistema, per comprendere i risultati dovuti ad un estraneo che si approccia in qualità di “sconosciuto”. Infine, ci si è soffermati molto su quelle che sarebbero potute essere le risposte tipiche che è possibile dare ad un comune e prettamente portato all’utilizzo quotidiano citofono, tra le quali si annoverano: “io”, “apri”, il proprio nome, “ho dimenticato il portafogli”, “sono io, scendi”.

### Test

Il sistema è stato testato da tutti e 16 gli utenti sia in condizione di registrazione avvenuta con successo, sia in condizione di recognition con una mancata generazione del modello

Come valore di soglia iniziale si è optato per il valore 0.25 per il volto il quale si presenta come un valore base molto basso ed ottenuto con il semplice modelling dei due sviluppatori. Uno degli obiettivi del testing è, infatti, modificare la soglia per renderla ottimale e corretta. Il valore di treshold iniziale per il riconoscimento del volto è pari a 0.25 (che indica la distanza minima tra il modello e la foto in input dove il valore 0 è indice di riconoscimento perfetto e 100 l’assoluta distanza), soglia alta e molto selettiva.

Per quanto concerne l’audio si è optato per una soglia pari a 65 (che indica lo score su scala logaritmica dell’input con il Gaussian Mixture Model preso in esame), soglia alta e selettiva allo stesso modo della soglia attuata per il riconoscimento del volto.

Gli obiettivi da raggiungere per quanto concerne False Acceptance Rate e False Rejection Rate sono rispettivamente 15% e 34%. Il False Rejection Rate si presenta come più alto in percentuale poiché non si vuole, al fine di diminuire il False Rejection Rate, colpire l’integrità del sistema.

Ogni utente ha attuato un recognition per un totale di 5 tentativi ciascuno, a partire dal quale è stato ottenuto un False Acceptance Rate. Il tasso mostrato in tabella è la media dei valori ottenuti dai 5 tentativi. Risulta chiaro come, in questo specifico test, avremo risultati di F.A.R. molto bassi. L’obiettivo, dunque, sarà quello di ottenere valori simili ma aumentando la soglia di treshold, rendendola meno selettiva.

Nello specifico, i 5 test corrispondono a:

* Ambiente chiuso con luce naturale e poco rumore
* Ambiente chiuso con scarsa luce artificiale e medio rumore
* Ambiente chiuso con molta luce artificiale e poco rumore
* Ambiente aperto con molta luce naturale e molto rumore
* Ambiente aperto con scarsa luce naturale e molto rumore

Con l’utente non registrato:

I valori posti nella seguente tabella corrispondono al valore di sicurezza da parte del sistema nel riconoscimento dell’utente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 |
| Utente 1 | 18% | 17% | 21% | 22% | 20% |
| Utente 2 | 20% | 16% | 22% | 23% | 20% |
| Utente 3 | 20% | 14% | 21% | 22% | 19% |
| Utente 4 | 19% | 18% | 20% | 22% | 15% |
| Utente 5 | 20% | 11% | 21% | 22% | 18% |
| Utente 6 | 17% | 13% | 20% | 20% | 12% |
| Utente 7 | 19% | 15% | 18% | 15% | 20% |
| Utente 8 | 18% | 12% | 21% | 21% | 18% |
| Utente 9 | 20% | 12% | 19% | 13% | 21% |
| Utente 10 | 12% | 16% | 22% | 20% | 20% |
| Utente 11 | 15% | 14% | 18% | 19% | 17% |
| Utente 12 | 17% | 18% | 14% | 21% | 16% |
| Utente 13 | 20% | 17% | 15% | 20% | 20% |
| Utente 14 | 18% | 10% | 17% | 21% | 22% |
| Utente 15 | 20% | 16% | 20% | 18% | 17% |
| Utente 16 | 14% | 17% | 20% | 14% | 19% |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Utente 1 | Utente 2 | Utente 3 | Utente 4 | Utente 5 | Utente 6 | Utente 7 | Utente 8 |
| F.A.R. | 19.6% | 20.2% | 19.2% | 19.6% | 18.4% | 16.4% | 17.4% | 18% |
|  | Utente 9 | Utente 10 | Utente 11 | Utente 12 | Utente 13 | Utente 14 | Utente 15 | Utente 16 |
| F.A.R. | 17% | 18% | 16.6% | 17.2% | 18.4% | 17.6% | 18.2% | 16.8% |

Con l’utente registrato:

I valori posti nella seguente tabella corrispondono al valore di sicurezza da parte del sistema nel riconoscimento dell’utente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 |
| Utente 1 | 65% | 57% | 77% | 67% | 55% |
| Utente 2 | 60% | 53% | 78% | 64% | 60% |
| Utente 3 | 61% | 54% | 73% | 65% | 57% |
| Utente 4 | 69% | 52% | 75% | 70% | 59% |
| Utente 5 | 60% | 49% | 72% | 69% | 54% |
| Utente 6 | 67% | 50% | 74% | 69% | 56% |
| Utente 7 | 64% | 52% | 76% | 67% | 58% |
| Utente 8 | 63% | 52% | 76% | 63% | 45% |
| Utente 9 | 60% | 51% | 76% | 59% | 60% |
| Utente 10 | 62% | 51% | 74% | 66% | 60% |
| Utente 11 | 66% | 45% | 68% | 68% | 57% |
| Utente 12 | 64% | 53% | 73% | 69% | 56% |
| Utente 13 | 63% | 57% | 75% | 66% | 60% |
| Utente 14 | 62% | 50% | 70% | 68% | 52% |
| Utente 15 | 60% | 56% | 69% | 66% | 59% |
| Utente 16 | 64% | 57% | 68% | 65% | 51% |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Utente 1 | Utente 2 | Utente 3 | Utente 4 | Utente 5 | Utente 6 | Utente 7 | Utente 8 |
| F.R.R. | 35.8 | 37 | 38 | 35 | 39.2 | 36.8 | 36.6 | 40.2 |
|  | Utente 9 | Utente 10 | Utente 11 | Utente 12 | Utente 13 | Utente 14 | Utente 15 | Utente 16 |
| F.R.R. | 38.8 | 37.4 | 39.2 | 37 | 35.8 | 39.6 | 38 | 37 |

Da questi test è emersa la volontà di diminuire sia il False Acceptance Rate che il False Rejection Rate. Questo è reso possibile modificando i valori di treshold di una quantità minima per evitare un incremento massiccio nel lavore di False Acceptance. I valori scelti, e poi testati, sono stati 0.52 per il volto e 60 per l’audio

I test successivi hanno condotto ai seguenti risultati:

Con l’utente non registrato:

I valori posti nella seguente tabella corrispondono al valore di sicurezza da parte del sistema nel riconoscimento dell’utente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 |
| Utente 1 | 8% | 17% | 11% | 17% | 18% |
| Utente 2 | 12% | 16% | 12% | 17% | 19% |
| Utente 3 | 12% | 14% | 11% | 16% | 19% |
| Utente 4 | 9% | 16% | 10% | 17% | 16% |
| Utente 5 | 10% | 11% | 13% | 16% | 17% |
| Utente 6 | 12% | 12% | 10% | 20% | 15% |
| Utente 7 | 14% | 16% | 8% | 19% | 21% |
| Utente 8 | 13% | 11% | 11% | 17% | 18% |
| Utente 9 | 10% | 11% | 9% | 18% | 19% |
| Utente 10 | 9% | 15% | 12% | 15% | 18% |
| Utente 11 | 13% | 13% | 13% | 18% | 14% |
| Utente 12 | 12% | 17% | 14% | 20% | 15% |
| Utente 13 | 15% | 15% | 15% | 18% | 18% |
| Utente 14 | 18% | 10% | 12% | 16% | 19% |
| Utente 15 | 10% | 14% | 12% | 18% | 18% |
| Utente 16 | 14% | 14% | 13% | 17% | 19% |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Utente 1 | Utente 2 | Utente 3 | Utente 4 | Utente 5 | Utente 6 | Utente 7 | Utente 8 |
| F.A.R. | 14.2% | 15.2% | 14.4% | 13.6% | 13.4% | 13.8% | 15.6% | 14% |
|  | Utente 9 | Utente 10 | Utente 11 | Utente 12 | Utente 13 | Utente 14 | Utente 15 | Utente 16 |
| F.A.R. | 13.4% | 13.8% | 14..2% | 15.6% | 16.2% | 15% | 14.4% | 15.4% |

Che comporta un valore medio pari a 14.5%, che supera di pochissimo la soglia da noi imposta

Con l’utente registrato:

I valori posti nella seguente tabella corrispondono al valore di sicurezza da parte del sistema nel riconoscimento dell’utente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 |
| Utente 1 | 75% | 68% | 77% | 72% | 63% |
| Utente 2 | 68% | 66% | 78% | 69% | 66% |
| Utente 3 | 69% | 64% | 72% | 70% | 62% |
| Utente 4 | 71% | 66% | 75% | 70% | 65% |
| Utente 5 | 70% | 67% | 73% | 69% | 62% |
| Utente 6 | 70% | 65% | 74% | 68% | 61% |
| Utente 7 | 68% | 66% | 75% | 67% | 58% |
| Utente 8 | 68% | 65% | 77% | 68% | 56% |
| Utente 9 | 64% | 64% | 76% | 69% | 61% |
| Utente 10 | 72% | 64% | 76% | 76% | 60% |
| Utente 11 | 69% | 62% | 75% | 70% | 57% |
| Utente 12 | 70% | 63% | 73% | 70% | 56% |
| Utente 13 | 68% | 67% | 75% | 76% | 61% |
| Utente 14 | 69% | 63% | 70% | 66% | 58% |
| Utente 15 | 70% | 66% | 71% | 69% | 59% |
| Utente 16 | 68% | 66% | 70% | 75% | 56% |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Utente 1 | Utente 2 | Utente 3 | Utente 4 | Utente 5 | Utente 6 | Utente 7 | Utente 8 |
| F.R.R. | 29 | 30.6 | 32.6 | 30.6 | 31.8 | 32.4 | 33.2 | 33.2 |
|  | Utente 9 | Utente 10 | Utente 11 | Utente 12 | Utente 13 | Utente 14 | Utente 15 | Utente 16 |
| F.R.R. | 33.2 | 30.4 | 33.4 | 33.6 | 30.6 | 34.8 | 33 | 33 |

Che comporta un valore medio pari a 32.2%, che supera di poco la soglia da noi imposta

## Sviluppi Futuri

### Liveness detection

Allo status quo, nessuno impedisce ad un possibile intruso di utilizzare una fotografia grande abbastanza raffigurante un utente registrato per superare il controllo di face recognition. Per risolvere ciò si potrebbe implementare un sistema di Liveness Detection che impedisca attacchi all’integrità del sistema di questo tipo, andando a migliorare la stabilità del sistema

### Tape Attack

Allo status quo, nessuno impedisce ad un possibile intruso di utilizzare una registrazione appartenente ad utente registrato per superare il controllo di speaker recognition. Per risolvere ciò si potrebbe implementare un sistema che permetta di fronteggiare il tape attack. Si potrebbe, ad esempio, scrivere un file di frasi da far leggere all’utente in fase di recognition; si otterrebbe l’input e si confronterebbe con il testo. Se il testo dovesse corrispondere a quel punto si potrebbe iniziare ad effettuare il controllo di speaker recognition.

Una possibile implementazione è possibile grazie all’utilizzo della libreria speech\_recognition di Python che permette di utilizzare il sistema di conversione del parlato in scrittura utilizzando le API di Google e settando vari tipi di lingue, tra le quali l’italiano.

### IOS System Notification

L’attuale notification sender è collegato ad un app Android. Un possibile sviluppo dell’applicativo potrebbe riguardare lo sviluppo di app nativa IOS per andare incontro alle esigenze di ogni tipologia di utente.