Biometric Systems

Eduardo Rinaldi

Contents

1 Introduzione

Il termine "biometric" deriva dal greco dalle parole "bios" (*vita*) e "metron" (*misura*), in informatica questo termine viene usato per descrivere il processo di riconoscimento di una persona tramite **caratteristiche fisiche** o **comportamentali**.

Un possibile approccio ai "Sistemi biometrici" è tramite **riconoscimento di pattern** (pattern recognition): due pattern p_1 e p_2 vengono ritenuti simili se e solo se, data una misura di distanza d, abbiamo: $d(p_1, p_2) < t$, dove t è una soglia entro la quale la loro distanza deve rientrare. Come possiamo vedere già qui incontriamo i primi problemi riguardanti la scelta di d e t.

In generale l'autenticazione di un individuo può avvenire in 3 modi:

1. Tramite **qualcosa che abbiamo** (un oggetto) → l'oggetto potrebbe essere rubato
2. Tramite **qualcosa che sappiamo** (una password) → molto sicuro ma è difficile ricordare tante password
3. Tramite **qualcosa che siamo** (un tratto biometrico) → facile da usare per l'utente ma non sicuro al 100%

1.1 Storia

L'apripista ai sistemi biometrici fu **Bertillon** nel 1882, capo della polizia di Parigi che, iniziò a tenere traccia di alcune misure del corpo dei diversi detenuti che gli si presentarono, istituendo pian piano così il primo sistema di riconoscimento biometrico. Il sistema fu poi criticato su un punto di vista statistico da parte di **Galton**, quest'ultimo invece, introducendo il concetto di "minuzie", fu il pioniere del riconoscimento tramite impronte digitali.

1.2 Architettura di un sistema biometrico

Un sistema biometrico può essere suddiviso in principalmente due fasi:

1. **Enrollment**: avviene la cattura del dato biometrico e viene poi processato andando a estrarne le feature, creando così il template; questo viene poi salvato all'interno di una "gallery" (che possiamo vedere come un database).
2. **Riconoscimento**: avviene la cattura del dato biometrico e la conseguente creazione del template, proprio come nella fase di enrollment, poi successivamente questo viene confrontato con altri template nella gallery. Il processo di confronto può essere di due tipi:
    - **Verifica**: l'utente reclama una certa identità e il sistema deve verificarla (1:1)

- **Identificazione**: il sistema deve identificare da solo l'identità dell'utente tra tutti quelli nella "gallery" (1:N). Può a sua volta dividersi in "**open-set**" (i template in input possono non essere presenti nella gallery) e "**closed-set**".

Da qui in avanti per **probe** intenderemo ogni template dato "in pasto" al sistema per la fase di riconoscimento.

Tipi di utente cooperativi/non-cooperativi, abituati/non abituati, pubblico/privato, consapevole/non consapevole.

1.2.1 Tratti biometrici

Ogni tratto biometrico deve avere le seguenti proprietà:

- **Universalità**: chiunque (o quasi) può averlo
- **Unicità**: è diverso per ogni individuo
- **Permanenza**: non cambia negli anni
- **Misurabilità**: deve poter essere misurabile da un sensore
- **Accettabilità**: gli utenti coinvolti non devono essere contrari alla raccolta di dati per quel tratto

Definiamo come "tratti biometrici **strong**" quelli che hanno le proprietà sopra citate, alcuni esempi sono:

- **Tratti fisici**: impronte digitali, occhi (iride e sclera), faccia, orecchie, mani
- **Tratti comportamentali**: camminata, firma
- **Mixed**: voce
- **Biologici**: DNA

Definiamo, invece, come "tratti biometrici **weak**" quelli che al contrario non sono sufficienti ad identificare una persona da un'altra (es.: il colore dei capelli).

I sistemi biometrici portano un grande vantaggio per quanta riguarda la facilità di utilizzo per un utente (basti pensare allo sblocco del telefono tramite face-id rispetto a uno tramite PIN), ma bisogna tenere bene a mente gli aspetti negativi come:

- la non affidabilità al 100%
- la non usabilità da parte di alcuni utenti
- il cambio di alcuni tratti con il passare degli anni
- il sistema biometrico potrebbe non essere utilizzabile in alcuni momenti (es.: face-id con la mascherina per il Covid)

2 Valutazione delle performance

In un sistema biometrico è molto importante effettuare una fase di valutazione delle performance del modello, in quanto da questa possiamo ricavare informazioni utili riguardo a esso. Si possono presentare i seguenti problemi/casi:

- Una variazione intra-classe molto alta → vedendo i sample come punti in uno spazio possiamo dire che quelli della stessa classe sono molto distanti tra loro
- Una variazione inter-classe molto bassa → sample tra classi diverse sono molto vicini tra loro
- Acquisizione distorta o con "rumore" (*noise*)
- Attacchi in diversi punti del sistema biometrico (*spoofing*)

2.1 Errori in fase di verifica

Nella fase di verifica, un soggetto viene accettato se e solo se la distanza tra il probe e i template associati all'identità reclamata è minore o uguale di una certa soglia (*threshold*). Quindi si possono presentare 4 output:

- Accettazione "genuina" (**Genuine acceptance - GA**): l'identità reclamata è vera e il soggetto viene accettato dal sistema.
- Rifiuto "errato" (**False rejection - FR**): l'identità reclamata è vera e il soggetto viene rifiutato dal sistema. (*type I*)
- Accettazione "errata" (**False acceptance - FA**): l'identità reclamata è falsa e il soggetto viene accettato dal sistema. (*type II*)
- Rifiuto "genuino" (**Genuine rejection - GR**): l'identità reclamata è falsa e il soggetto viene rifiutato dal sistema.

Diverse misure di valutazione possono essere utilizzate, le principali sono:

- **FA Rate**: probabilità di un impostore di essere accettato dal sistema. Viene calcolato dalla seguente formula (semplificata)

$$FAR(t) = \frac{|\{\text{probe di soggetti impostori accettati}\}|}{|\{\text{probe di soggetti impostori}\}|}$$

  (Da notare che i soggetti impostori potrebbero non essere nella gallery, ma potrebbero anche essere nella gallery e fanno un claim "da impostore")

- **FR Rate**: probabilità di un utente "genuino" di essere rifiutato dal sistema. Viene calcolato dalla seguente formula (semplificata)

$$FRR(t) = \frac{|\{\text{probe di soggetti con claim genuino ma rifiutati}\}|}{|\{\text{probe di soggeti con claim genuino}\}|}$$

  (Da notare qui che i soggetti presi in considerazione sono solo quelli nella gallery)

Da questi possiamo derivare:

- $GAR = 1 - FRR$
- $GRR = 1 - FAR$
- $EER : FAR = FRR$
- Detection Error Trade-off (**DET**): mette in relazione FAR e FRR
- Receiving Operating Characteristics (**ROC**): mette in relazione 1-FRR (GAR) e FAR

2.2 Errori in fase di identificazione - Open-set

L'obiettivo del sistema è quello di verificare se il soggetto è all'interno del database e in tal caso capire chi è (senza alcun "claim" di identità da parte dell'individuo). Rispetto alla fase di verifica si presentano casi diversi:

1. Ci sono individui nel database che superano la soglia di accettazione & l'identità associata al template a rango 1 è quella corretta → *corretta identificazione*
2. Non ci sono individui nel database che superano la soglia di accettazione → *identificazione errata*
3. Ci sono individui nel database che superano la soglia di accettazione & l'identità associata al template a rango 1 è errata → *identificazione errata*

Le misure di valutazione che possiamo utilizzare sono le seguenti:

- Detect e Identification rate (**DIR**) a rango k: probabilità di avere una corretta identificazione entro il rango k. Possiamo calcolarlo dalla seguente formula:

$$DIR(t,k) = \frac{|\{p_j : rango(p_j) \leq k \wedge d_{i,j} \leq t \wedge id(p_j) = id(g_i)\}|}{|P_G|}$$

- **FRR**: 1 - DIR(t, 1)
- **FAR**: è definito dal rapporto tra il numero di volte un probe non presente nella gallery supera la soglia di accettazione e il numero di probe non presenti nella gallery che sono stati testati

$$FAR(t) = \frac{|\{p_j : max_i\ d_{i,j} \leq t\}|}{|P_N|},$$

$\forall p_j \in P_N$ e $\forall g_i \in G$

Da questi è poi possibile ottenere altre metriche come:

- EER

- ROC, questa volta vengono messi in relazione $FAR(t)$ e $DIR(t, 1)$

In base all'applicazione finale ovviamente vanno scelte le metriche a cui dare maggior rilevanza:

- Se volessimo un'applicazione con pochi falsi allarmi (FAR basso), dovremmo scendere a compromessi sulla misura DIR
- Se volessimo un'applicazione in cui è più importante un'identificazione corretta invece, dovremmo scendere a compromessi sulla misura FAR

2.3 Errori in fase di identificazione - Closed-set

L'identificazione closed-set è un caso speciale in cui si assume che il soggetto associato a ogni probe ha sicuramente una corrispondenza nella gallery. Questo caso, seppur applicabile in pochi contesti, va comunque tenuto in considerazione. Possiamo utilizzare le seguenti metriche:

- Cumulative Match Score a rango k (**CMS(k)**): probabilità di identificare un soggetto entro il rango k
- Cumulative Match Characteristic (**CMC**): curva che mostra per ogni rango k il valore $CMS(k)$.
- Recognition Rate (**RR**): $CMS(1)$

2.4 Organizzazione del dataset

Una cosa da tenere bene a mente è che in fase di sviluppo e test del modello noi abbiamo sempre a disposizione il "*ground truth*" di ogni template, nel momento in cui invece un modello viene utilizzato poi in situazioni reali, questa informazione è ovviamente non disponibile.

Per una buona valutazione di un sistema è molto importante avere una buona suddivisione del dataset. Questa avviene su diversi livelli.

Suddivisione TR vs TS Anche se non è richiesto alcun procedimento di "*training*" in senso lato ML, bisogna comunque suddividere il dataset in "*train*" e "*test*" set in quanto potrebbe essere necessario effettuare il tuning di qualche parametro (es.: una normalizzazione). L'obiettivo qui è quello di consentire una fase di training che possa generalizzare, e quindi vedere, esempi molto differenziati tra loro. E' possibile inserire in **TS** template di soggetti che non appaiono completamente in **TR**, così che si possa testare una maggiore generalizzazione del modello. $TR \cap TS = \emptyset$ è fondamentale.

Suddivisione probe vs gallery Questa suddivisione viene effettuata per sample, una condizione fondamentale è $P \cap G = \emptyset$. E' buona norma inserire nella gallery i template catturati in condizioni controllate, in quanto nella realtà è la condizione più frequente.

Suddivisione probe set Questa suddivisione è fatta per soggetto e può essere di due tipi:

- $P = P_G \cup P_N$
- $P = P_G$

Il primo caso potrebbe condizionare i risultati di valutazione nel caso di identificazione open-set.

Un modo per diminuire il bias in fase di valutazione è quello di ripetere la valutazione con differenti combinazioni di training e test set per poi effettuarne la media. Un possibile approccio è dato dal **K-Fold cross validation**.

2.5 All probe vs All gallery

E' possibile calcolare da subito una matrice di distanze tra coppie di template probe-gallery e può essere utilizzata per la valutazione di diversi tipi di applicazioni. Ogni riga corrisponde a un'operazione di riconoscimento su un probe in input.

Il problema di questo approccio è che i risultati dipendono molto dalla suddivisione "*probe vs gallery*", in quanto forniscono ognuna una matrice di distanze diversa e quindi bisogna effettuare un numero sufficiente di test.

2.6 All vs All

Un metodo alternativo al precedente consiste nel calcolare una matrice di distanze tra tutti i possibili template, ognuno dei quali giocherà entrambi i ruoli di "probe" e "gallery template". Ogni probe (quindi ogni riga) potrà a turno essere considerato impostore o genuino.

Analizziamo pro e contro:

- Pro: facile da programmare e mette sotto stress il sistema in quanto ci saranno molti più tentativi da impostore rispetto a quelli da "genuino".
- Contro: il calcolo della matrice può essere dispendioso e non è possibile creare delle distribuzioni particolari tra probe e gallery, le quali potrebbero fornire dei risultati interessanti.

Di seguito vengono riportati gli algoritmi per i possibili tipi di operazione; hanno in comune i seguenti elementi:

- M = matrice delle distanze
- N = numero di soggetti
- G = numero totale di sample = numero di righe/colonne di M
- S = numero di template per soggetto ($|G| = SxN$)

- i = indice righe, j = indice colonne
- $label(x)$ = verà identità di x

Verifica, template singolo ogni riga è un set di $|G|-1$ operazioni, con $S-1$ test genuini e $(N-1) \times (S)$ test da impostore.

```
for each threshold t:
    for each cell M_{i,j} with i ≠ j:
        if M_{i,j} ≤ t:
            if label(i) = label(j): GA++
            else: FA++
        else if label(i) = label(j): FR++
        else: GR++
    GAR(t) = GA/TG
    FAR(t) = FA/TI
    FRR(t) = FR/TG
    GRR(t) = GR/TI
```

Figure 1

Dove TG è il numero totale di tentativi genuini e TI è il numero totale di impostori.

Verifica, template multipli ogni riga è un set di N operazioni, con 1 test genuino e $N-1$ test da impostore.

```
for each threshold t:
    for each row i:
        for each group M_label of cells M_{i,j} with the same label(j) excluding M_{i,j}:
            diff ← min(M_label)
            if diff ≤ t:
                if label(i) = label(M_label): GA++
                else: FA++
            else if label(i) = label(M_label): FR++
                else: GR++
    GAR(t) = GA/TG
    FAR(t) = FA/TI
    FRR(t) = FR/TG
    GRR(t) = GR/TI
```

Figure 2

Identificazione, open-set template multipli ogni riga è un set di 2 operazioni, uno da impostore e uno da genuino.

```
for each threshold t:
    for each row i:
        {L_{i,m} : m = 1...|G|-1} = {M_{i,j} : j = 1...|G|} - M_{i,i} ordered by increasing value
            // the identical element is excluded
        if L_{i,j} ≤ t: // potential accept
            if label(i) = label(L_{i,j}): DI(t,1) ++ // genuine case
            find the first L_{i,k} such that label(i) ≠ label(L_{i,k}) ∧ L_{i,k} ≤ t
            if shuch k exists: FA++  // impostor case: jump the label
            else: find the first L_{i,k} such that label(i) = label(L_{i,k}) ∧ L_{i,k} ≤ t
                // if genuine yet not the first, look for higher ranks
            if shuch k exists: DI(t,k)++  // end of genuine
            FA++   // impostor in parallel, distance below t but different label,
                   // no need to jump since the first label is not the impostor
        else: GR++   // impostor case counted directly, FR computed through DIR
DIR(t,1) = DI/TG
FRR(t) = 1-DIR(t,1)
FAR(t) = FA/TI
GRR(t) = GR/TI

k=2   // higher ranks
while DI(t,k) ≠ 0:
    DIR(t,k) = DI(t,k)/TG + DIR(t,k-1)   // we have to compute rates
```

Figure 3

Per ottenere informazioni riguardo il CMS, possiamo utilizzare il seguente algoritmo dove, ogni riga è un'operazione da genuino (zero impostori, in quanto non usiamo alcuna threshold)

```
for each row i:
    {L_{i,m} : m = 1...|G|-1} = {M_{i,j} : j = 1...|G|} - M_{i,i} ordered by increasing value
    find the first L_{i,k} such that label(i) = label(L_{i,k})
    CMS(k)++
CMS(1) = CMS(1)/TA
RR = CMS(1)
k=2
while k < |G| - 1:
    CMS(k) = CMS(k)/TA + CMS(k-1)
```

Figure 4

Questi stessi algoritmi possono essere riutilizzati per una valutazione All probe vs All gallery, in quanto sulle righe abbiamo i probe e sulle colonne i template della gallery quindi è facile simularne la distribuzione.

3 Definizione di metrica

E' una funzione

$$d : X \times X \to \mathbb{R}$$

dalle seguenti proprietà:

- $d(x, y) \geq 0$

- $d(x, y) = 0 \iff x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$ (una semimetrica non ha questa proprietà)

Se non è simmetrica possiamo usare

$$d^*(x, y) = \frac{d(x, y) + d(y, x)}{2}$$

4 Affidabilità del riconoscimento (Recognition Reliability)

4.1 Doddington Zoo

Doddington ha definito alcune similitudini con le classi animali per quanto riguarda lo speaker recognition:

- **Pecora**: produce tratti biometrici che matchano bene con altri template di se stesso e male con quelli di altri soggetti (classe media)
- **Capra**: produce tratti biometrici che matchano poco con altri template di se stesso, aumentando così i casi di FR
- **Agnello**: produce tratti biometrici che sono facilmente riproducibili da altri, aumentando così i casi di FA nei suoi confronti
- **Lupo**: produce un matching score più alto rispetto alla media quando viene confrontato con template diversi dal suo, aumentando così i FA

Successivamente Yager ha individuato altre 4 classi, prendendo però in considerazione sia score genuini (G_k) che score impostori (I_k), al contrario di Doddington che invece prendeva in considerazione solo uno dei due. Ha individuato quindi:

- **Camaleonte**: ha sia G_k che I_k alti, quindi riesce facilmente ad apparire simile ad altri. Può produrre tanti FA ma pochi FR
- **Fantasma**: ha sia G_k che I_k bassi, quindi causa tanti FR
- **Colomba**: è un'estensione della pecora di Doddington, ha G_k alto e I_k basso, quindi è l'utente ideale.
- **Verme**: avendo G_k basso e I_k alto è il peggiore utente che si possa avere.

4.2 Misurare la qualità di un sample

I sample presi in input possono avere una qualità diversa (per diversi motivi, es.: qualità sensore, tempo, ombre, etc..), possiamo quindi definire delle misure di affidabilità del riconoscimento che ci aprono ulteriori opzioni come ad esempio il richiedere all'utente di ripetere la cattura del sample.

4.2.1 Qualità dell'immagine

Una possibile misura di qualità è tramite la stima della qualità dell'immagine. Questa può essere effettuata in diversi modi:

- Prendendo tutte le immagini in TR e calcolando un **template medio** da utilizzare come **riferimento**.
- Stimando la **nitidezza dell'immagine** andando a visualizzare l'istogramma delle frequenze; una mancanza di alte frequenze significa una bassa nitidezza (immagine sfocata)
- Nel caso di un face recognizer:
  - SP per misurare la distorsione rispetto a una posa frontale, misuriamo quindi dei disallineamenti sui vari assi di rotazione.

$$SP = \alpha(1 - roll) + \beta(1 - yaw) + \gamma(1 - pitch)$$

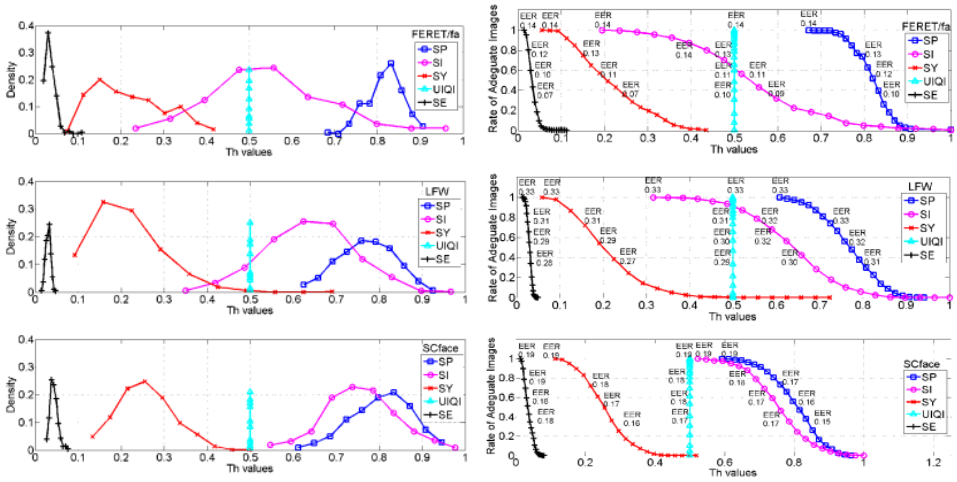
  - SI per misurare l'omogeneità dei livelli di grigio in alcune zone predeterminate della faccia, utile per vedere se ci sono ombre nette in alcuni punti.

$$SI = 1 - F(std(mc))$$

  - SY per misurare la simmetria della faccia

$$SY = \sum_{(i,j) \in X} sym(P_i, P_j)$$

Visualizzando la **distribuzione di un dataset rispetto alle misure proposte/scelte** è possibile decidere **quali sample scartare** (possibilmente solo quelli che presentano gravi distorsioni), sapendo a priori quanto questo inciderà sulla grandezza finale del dataset. Per valutare le performance di una misura di qualità possiamo verificare l'aumento o la diminuzione delle performance del modello finale (EER): una buona misura di qualità dovrebbe aumentare l'efficacia del modello sui sample con score alti.



(a) Distribuzione di vari dataset rispetto alle misure proposte

(b) Performance di un sistema in seguito all'uso di misure di qualità

Figure 5

Un altro approccio è tramite il concetto di "**margine**", che viene calcolato nel seguente modo:

$$M(t) = |FAR(t) - FRR(t)|,$$

da notare che $M(t) = 0$ se siamo sull'EER.

4.3 SRR - System Response Reliability

C'è una netta differenza tra misurare la qualità di un'immagine in input rispetto al misurare l'affidabilità di una risposta da parte del sistema.

Quest'ultimo approccio viene chiamato indice SRR, ovvero un valore nel range $[0, 1]$ che fornisce una misura di quanto un sistema, in fase di identificazione, riesce a separare bene soggetti genuini da soggetti impostori sulla base di un singolo probe. Questo sistema utilizza una funzione φ che misura la quantità di "confusione" tra i possibili candidati: presa la lista data in output da una fase di identificazione, si guarda nell'intorno del risultato a rango 1. Se i soggetti a ranghi più bassi sono molto vicini, avremo una risposta poco affidabile, altrimenti se c'è una buona distanza avremo una risposta affidabile. Possibili esempi di φ sono:

- $$\varphi(p) = \frac{F(d(p, g_1)) - F(d(p, g_2))}{F(d(p, g_{|G|}))}$$

- $$\varphi(p) = 1 - \frac{|N_b|}{|G|}$$

  dove $N_b = \{g_i \in G | F(d(p, g_i)) < 2F(d(p, g_1))\}$

Definiamo poi φ_k, come quel valore che minimizza gli errori di $\varphi(p)$, ovvero i casi in cui, probe impostori hanno $\varphi(p_I) > \varphi_k$ e probe genuini hanno $\varphi(p_G) \leq \varphi_k$. Valori $\varphi(p)$ molti distanti da φ_k avranno un SRR maggiore, quindi:

$$S(\varphi(p), \varphi_k) = \begin{cases} 1 - \varphi_k & \text{se } \varphi(p) > \varphi_k, \\ \varphi_k & \text{altrimenti} \end{cases}$$

e

$$SRR = \frac{\varphi(p) - \varphi_k}{S(\varphi(p), \varphi_k)}$$

4.4 Aggiornamento dei template

Un altro modo per aumentare la qualità e l'affidabilità di un sistema è quella tramite l'aggiornamento dei template (evitando problemi come l'invecchiamento). Questa operazione per una maggiore sicurezza deve essere fatta in soli due possibili modi:

- Supervisionata (supervised)
- Semi-supervisionata (semi-supervised)

5 Riconoscimento facciale

I fattori più importanti di un sistema biometrico sono l'**accettabilità**, l'**affidabilità** e l'**accuratezza**. Il **DNA**, ad esempio, fornisce un'alta accuratezza e affidabilità ma una bassa accettabilità, in quanto il metodo di prelievo è sicuramente intrusivo. Le **impronte digitali**, invece, forniscono anch'esse buone prestazioni, ma possono spesso presentarsi in modo "parziale" e inoltre sono spesso associate ai "criminali". Il **riconoscimento facciale** invece è altamente accettato, in quanto siamo abituati a farci foto e a pubblicarle, ma l'accuratezza può diminuire drasticamente in casi non controllati. Possibili problemi relativi a essa sono:

- A-PIE: invecchiamento, posa, illuminazione ed espressione
- Facilmente camuffabile: makeup, chirurgia plastica, occhiali, etc..
- Variazione intra-personale e similarità inter-personale

La struttura generale di un riconoscitore facciale prevede:

cattura immagine → localizzazione, crop, normalizzazione → estrazione feature

5.1 Localizzazione della faccia

L'obiettivo qui è quello di contare quante facce ci sono all'interno di un'immagine e successivamente individuarne la posizione, indipendentemente da A-PIE e sfondo.

Possiamo avere approcci di diversa natura:

- **Basati su feature**: si individuano le feature principali di una faccia (ad esempio posizione occhi, posizione naso, etc..) e poi si possono verificare diverse proprietà di queste (es.: colore della pelle corretto o distanza tra naso e occhi entro una certa soglia)
- **Basati su immagine**: solitamente vengono utilizzati dei modelli di machine learning che imparano da immagini esemplificative.

5.1.1 Algoritmo A

Vediamo un primo esempio di algoritmo per localizzare una faccia ideato da Hsu. E' composto dai seguenti step:

1. Ricerca di possibili facce
   (a) Aggiustamento dell'illuminazione, in quanto il colore della pelle varia in base all'illuminazione dell'ambiente
   (b) Passaggio a uno spazio di colori diverso da RGB, ovvero Y, C_b, C_r
   (c) Localizzazione in base al modello della pelle, che può essere creato ad esempio tramite K-Means, e successiva connessione delle componenti individuate da questo modello.
2. Verifica delle possibili facce individuate, tramite l'individuazione delle feature facciali (occhi e naso).
3. Assegnamento di uno score basato sulle possibili combinazione di occhi e naso e scelta del triangolo con score maggiore (e superiore a una certa soglia).
4. Individuazione dei contorni della faccia tramite Hough.

5.1.2 Algoritmo B

Viola Jones rappresenta una vera e propria innovazione per quanta riguarda la localizzazione di una faccia all'interno di un'immagine. Essendo l'algoritmo basato su machine learning, il training di questo è avvenuto utilizzando un dataset personalizzato nel quale vengono etichettate immagini come positive, nel caso in cui ci sia una faccia e come negative nel caso in cui non ve ne sia alcuna.

Le idee utilizzate dietro questo algoritmo sono sostanzialmente:

- Utilizzo di **Haar feature** con valutazione veloce tramite integrazione di immagini: vengono calcolate su una sotto-regione di una sotto-finestra di un'immagine, andando a calcolarne il valore tramite la seguente formula

  $$Value = \sum \text{valore dei pixel nell'area bianca} - \sum \text{valore dei pixel nell'area nera}$$

- **Ada-Boosting** per la selezione di feature: vengono creati diversi weak-classifier, uno per feature, e tramite adaptive boosting riusciamo a creare uno strong classifier composto da un subset di weak-classifier.
- **Classificatori a cascata** per velocizzare l'algoritmo: vengono creati diversi strong classifier, ognuno dei quali viene messo a cascata.

6 Riconoscimento facciale 2D

Nel riconoscimento facciale 2D abbiamo principalmente due problematiche da risolvere:

- Costruzione di feature discriminative e rappresentative
- Costruzione di un classificatore che possa generalizzare anche su oggetti mai visti nel training

In questo caso quindi ci troviamo a rappresentare le facce all'interno di immagini digitali, che vengono rappresentate da matrici a due dimensioni ($w \times h$) oppure vettori unidimensionali ($d = w \times h$). L'alta dimensionalità dei dati (es.: immagine 100×100 ha dimensione 10000) ci porta subito a un primo problema, ovvero il "**Curse of dimensionality**". Quando la dimensionalità aumenta, il volume dello spazio aumenta così velocemente che i dati diventano sparsi/radi. La sparsità dei dati, nel momento in cui vengono utilizzati strumenti statistici come ad esempio i modelli di machine learning, diminuisce drasticamente la capacità predittiva di questi in quanto si hanno bisogno di tanti più esempi per generalizzarne le regole predittive.

6.1 PCA - Principal Component Analysis

Una possibile soluzione al curse of dimensionality è data da PCA, ovvero un metodo statistico che ci consente di ridurre l'alta dimensionalità di uno spazio (es. N) mappando i dati in un altro spazio con una dimensionalità decisamente più piccola (es. $K << N$) minimizzando la perdita di informazioni.

Ciò che viene fatto sostanzialmente è individuare dei nuovi assi, ortogonali tra loro, su cui proiettare i dati che però ne massimizzano la varianza. L'ortogonalità di questi ci permette di escludere componenti correlate tra loro che risultano quindi ridondanti.

Nella pratica vengono calcolati due elementi:

1. Elemento medio del training set $\bar{x}$

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

2. Matrice di covarianza C $(n \times n)$

$$C = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{x})(x_i - \bar{x})^T$$

La nuova dimensione k è data dalla matrice di proiezione dove le colonne sono i k autovettori di C, corrispondenti ai k autovalori di C. Plottando gli autovalori possiamo ottenere la varianza lungo gli autovettori.

PCA però ha dei lati negativi:

- Mancanza di potere discriminativo
- In presenza di variazioni PIE, il modello potrebbe utilizzare quelle come componenti principali (feature inutile)

6.2 LDA - Linear Discriminant Analysis

Alcuni dei problemi di PCA possono essere dati dal fatto che lavoriamo su dati in una maniera unsupervised (non supervisionata), ovvero senza tenere in considerazione le classi dei vari sample. Una soluzione supervised è proposta da LDA, ovvero un metodo simile a PCA ma che minimizza la distanza intra-classe e cerca di massimizzare invece la distanza tra classi.

6.3 Estrattori di feature

Le feature rilevanti di un'immagine possono estratte tramite l'utilizzo di **filtri**, **trasformazioni** o diversi **operatori** ognuno designato e utilizzato per uno specifico obiettivo. Di seguito verranno riportati a titolo esemplificativo solo alcuni di questi in quanto ne esistono centinaia.

Ovviamente in caso di alta dimensionalità delle feature potremmo (o forse dovremmo) ricorrere ai metodi di riduzione della dimensionalità descritti nella sezione precedente.

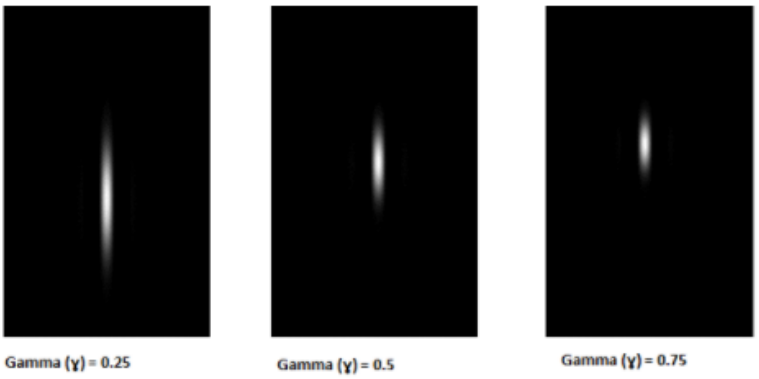
Gabor filter E' un filtro lineare utilizzato in applicazioni di edge detection, analisi di texture e estrazione di feature, e si pensa che sia simile al sistema percettivo visivo di noi umani. Un Gabor filter 2D non è altro che un kernel Gaussiano modulato da un'onda piana sinusoidale.

E' possibile creare diversi Gabor filter andando a cambiare i seguenti parametri:

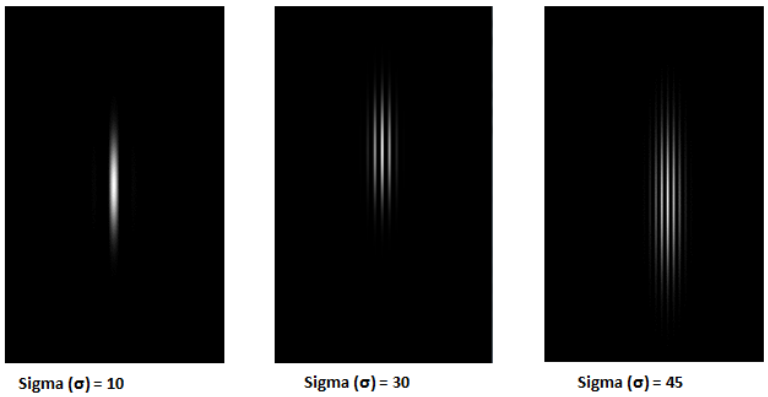
- λ: frequenza, controlla lo spessore della striscia; più è grande più sarà spessa.
- θ: orientamento, controlla la rotazione della striscia ed è espresso come un angolo.
- γ: aspect ratio, controlla l'altezza della striscia; più è grande più l'altezza diminuirà.
- σ: larghezza di banda, controlla la scala generale determinando anche il numero di strisce; aumentando questo valore avremo più strisce con meno distanza tra loro.

(a) Lambda

(b) Theta



(c) Gamma



(d) Sigma

Figure 6

Ovviamente in immagini o task complessi con un solo gabor filter non è possibile individuare le feature discriminanti, ciò che possiamo fare è utilizzare un'insieme di filtri (bank of filters) che ci aiutino a creare un vettore di feature. Ovvero, applicando un filtro su un'immagine otterremo in output una nuova immagine che può essere "appiattita" e utilizzata come vettore di feature.

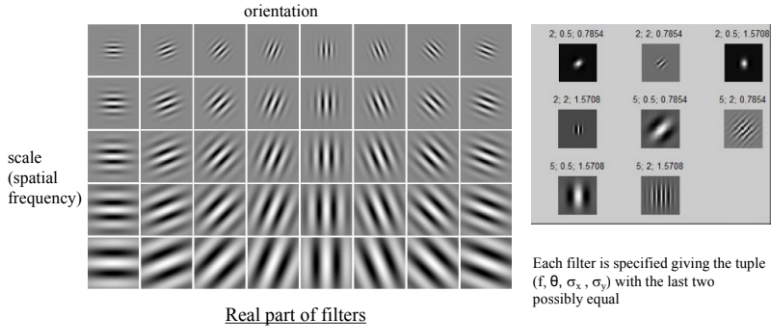


Figure 7: Esempio di "banca di filtri"

Questa procedura, su immagini molto grandi, porta ad un'alta dimensionalità; possibili soluzioni sono:

- Applicare il filtro su un subset di pixel, come ad esempio una griglia $\implies$ dimensionalità ridotta, ma possibile perdità di informazioni salienti (dovuta a possibili rotazioni ad esempio)
- Applico il filtro su tutti i pixel, ma poi seleziono solo i punti con picchi di valori $\implies$ problemi di allineamento

EBGM - Elastic Bunch Graph Matching Questo metodo fa uso di grafi, nello spacifico abbiamo per ogni soggetto una collezione di grafi (per questo il nome "**bunch**"), uno per ogni posa, dove:

- I vertici sono pesati dalla distanza tra i nodi
- I nodi contengono un insieme di risultati di diversi gabor filter (in genere 5 differenti frequenze e 8 diversi orientamenti), conservati in questa struttura chiamata "**Jet**", e sono posizionati in punti importanti come ad esempio naso, occhi e bocca.

LBP - Local Binary Pattern E' un operatore che lavora pixel per pixel che ci aiuta ad estrarre informazioni riguardo le texture all'interno di un'immagine (proprio per questo può essere utilizzato anche per antispoofing). Nella sua prima versione va a considerare una griglia di vicini 3×3 dove a ognuno dei quali viene assegnato un valore binario: se il suo valore era maggiore del valore del pixel centrale veniva assegnato 1 altrimenti 0. Effettuando questa operazione

con un ordine di processo dei vicini fisso (es.: partendo dal vicino in alto a sinistra) si ottiene una stringa binaria, il quale valore, convertito in decimale, viene assegnato al pixel centrale.

Questa versione è stata poi generalizzata andando a parametrizzare il numero di punti vicini (P) e il loro raggio di distanza (R), producendo così la seguente formula:

$$LBP_{R,P} = \sum_{i=0}^{P-1} sign(p_i - p_c)2^P$$

Un pattern viene definito **uniforme** se nella stringa binaria contiene al massimo due transizioni $0 \rightarrow 1$, $1 \rightarrow 0$. Questi rappresentando le informazioni essenziali (la maggior parte dei valori assunti dalla stringa binaria sono dati da stringhe NON uniformi), ci consentono di ridurre notevolmente il numero di punti da considerare riducendo la dimensionalità.

Vediamo ora come possiamo ottenere un vettore di features:

- L'immagine viene suddivisa in sotto-finestre grandi $k \times k$
- Per ogni sotto-finestra calcoliamo un istogramma (ad ogni bin corrisponde un pattern)
- Il vettore finale viene costruito concatenando i k^2 istogrammi.

Nota: le rotazioni variano i risultati di *LBP*. Una possibile soluzione è quella di effettuare diversi shift logici verso destra della stringa binaria e prendere sempre il valore (decimale) minimo.

6.4 Classificazioni di sistemi di riconoscimento facciale

- **Metodi basati sull'apparenza della faccia**: PCA, LDA, alcune reti neurali; utilizzano l'immagine per intero e quindi non perdono informazioni da subito concentrandosi su regioni specifiche. Gli svantaggi sono dati dal fatto che danno la stessa importanza a ogni pixel, hanno bisogno di un'alta correlazione tra dati di training e test, su grandi variazioni PIE non performano così bene.
- **Metodi basati su feature locali**: EBGM, LBP; sono robusti a varianze di posizione in quanto vengono prima individuati i punti da cui estrarre le feature, e inoltre sono computazionalmente veloci. Come principale svantaggio hanno la scelta a priori dei punti da cui estrarre le feature (se i punti da cui estrarre le feature non sono molto discriminativi avremo sicuramente pessime performance)
- **Sistemi basati su grafi**: a ogni faccia è associato un grafo, dove ogni nodo corrisponde a dei punti discriminativi della faccia. Sono ottimi dal punto di vista di variazioni di posizione e illuminazione, ma purtroppo il training e il testing sono lunghissimi.

- **Sistemi basati su immagini termografiche**: sono ottimi per quanto riguarda variazioni di illuminazione, però purtroppo richiedono attrezzatura adeguata e la temperatura misurata può variare in base allo stato del soggetto. Sono inoltre sensibili ai movimenti.

7 Riconoscimento facciale 3D

Nel riconoscimento facciale 2D i problemi che occorrono sono dovuti soprattutto a variazioni A-PIE e a possibili travestimenti da parte di impostori. Con il riconoscimento 3D molti di questi problemi non sono più tali (posizione e espressione possono essere corretti ad esempio), introducendo però altre complicanze come una maggiore complessità di acquisizione e una maggiore complessità nello gestire i dati acquisiti. Possibili rappresentazioni sono:

- **2.5D**: solitamente sono immagini 2D che, oltre a conservare l'informazione del colore nei 3 canali rgb, mantengono una quarta informazione che è la distanza d'acquisizione.
- **3D**: una serie di punti che connessi tra loro (solitamente tramite triangolarizzazione) formano un cosiddetto mesh 3D. I metodi di acquisizione sono diversi ognuno con i suoi pro e i suoi contro (camera stereoscopica, scanner di light pattern, laser scanner)

Il preprocessing che viene effettuato sui dati acquisiti può andare a risolvere i seguenti problemi:

- Presenza di spike o rumore
- Presenza di "buchi" nel mesh: smoothing gaussiano o interpolazione lineare aiutano a risolvere il problema
- Decentramento

Immagine 2D → 3D Sfruttando le proprietà di shading, e quindi come la luce riflette su una superficie, o in particolare qui sulla nostra faccia, è possibile stimare informazioni come ad esempio la profondità e ricavare quindi così un mesh 3D di una faccia da un'immagine 2D (in realtà molte volte un'immagina sola non basta, è quindi necessario avere più immagini con più orientamenti della faccia).

Un'altra tecnica è il "**morphing**", dove semplicemente viene creata una faccia 3D di riferimento e viene poi modellata per ottenere un risultato congruo a un'immagine di una persona. Questo ci permette di poter cambiare anche alcune caratteristiche del modello 3D così da poter stimare un modello sotto variazioni di espressione o età ad esempio.

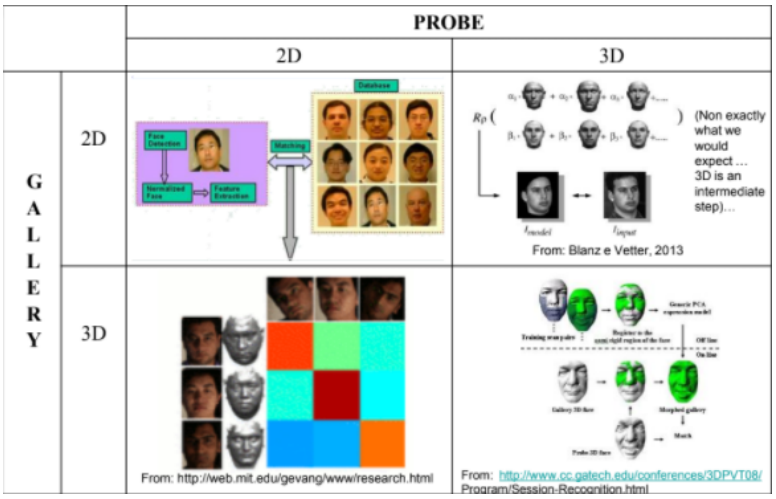


Figure 8: Possibili configurazioni

Da un modello 3D ciò che possiamo fare è:

- Aggiustare la posa
- Calcolare la normal map da confrontare con altre normal map

8 Valutazione di un riconoscitore facciale

Quando sviluppiamo un riconoscitore facciale (ma in generale un pattern recognizer) le classiche misure come FAR, FRR, CMS, etc.. forniscono una valutazione marginale del sistema; bisogna prendere in considerazione le variazioni (non solo le A-PIE) che ci possono essere e soprattutto il dataset scelto può condizionare i risultati.

Proprio per questo motivo **FERET** è stato il primo dataset messo a disposizione di tutti e utilizzato come strumento di benchmark dei propri face recognizer.

9 Spoofing e Camuffamento

Lo **spoofing** consiste nell'ingannare un sistema biometrico assumendo un'identità che non ci appartiene. La differenza sostanziale tra spoofing e **camuffamento** (camouflage) è data dal fatto che nel primo l'obiettivo è essere riconosciuti con un'altra identità e quindi ottenere un genuine accept; nel secondo invece l'obiettivo è quello di non far riconoscere al sistema la propria identità (es.: un criminale che non vuole farsi riconoscere dalle telecamere di sicurezza).

Questo tipo di attacco può avvenire in diversi punti del sistema e possiamo classificarli in due modi:

1. **Diretto**: avviene direttamente al momento dell'acquisizione
2. **Indiretto**: avviene nelle fasi successive della pipeline

9.1 Spoofing del volto

Per quanta riguarda un face recognizer possono avvenire i seguenti attacchi:

- **Print attack**: è un attacco diretto dove viene presentata un'immagine (in forma stampata o su display) del soggetto da identificare. Questo tipo di attacco può essere identificato abbastanza facilmente andando a sfruttare le differenze che ci sono tra un'immagine di una faccia catturata da due dispositivi (foto stampata e ricatturata dal sensore del sistema biometrico) e una catturata una singola volta. Tra queste differenze nominiamo la differenza dei possibili colori, differenza di come la luce riflette, differenza tra le texture. In tutti questi casi LBP viene in nostro aiuto. Un altro metodo consiste nell'individuare gesti involontari che ci danno informazioni riguardo la liveness di un soggetto, come ad esempio il battito delle ciglia.
- **Video**: simile al precedente ma viene presentato un video del soggetto da identificare. Anche qui LBP viene utilizzato.
- **Maschera facciale**: viene prodotta una maschera 3D, queste hanno un notevole costo e quindi sono meno frequenti. Una possibile soluzione è quella di verificare come la luce riflette sulla pelle (su una maschera in plastica la luce ad esempio rifletterà, mentre sulla pelle viene assorbito)

Quando produciamo un sistema biometrico, se prendiamo in considerazione anche gli attacchi di spoofing dobbiamo utilizzare una metrica diversa dal classico FAR, ovvero **Spoofing-FAR**, che prende in considerazione anche i possibili impostori che vengono accettati grazie allo spoofing.

Per la valutazione del sistema di spoofing invece possiamo usare:

- **False Living Rate** (FLR): numero di casi di soggetti che sono stati additati come "reali" dal sistema di spoofing ma in realtà erano fake.
- **False Fake Rate** (FFR): numero di casi di soggetti che sono stati additati come "fake" ma erano reali.

10 Riconoscimento dell'orecchio

L'orecchio è un tratto biometrico composto da una sua struttura ben definita:

- **Helix**: il bordo esterno
- **Anti-Helix**: i "bordi" interni
- **Lobo**

- **Tacca intratragica**: è il punto tra il lobo e il foro dell'orecchio

Vediamo i lati positivi e negativi:

- Pro: è un tratto passivo, non varia con gli anni, solitamente non è coperto, non sono richieste alte risoluzioni, non è affetto da variazioni dell'espressione
- Contro: sensibile a variazioni di illuminazione e posa

Una prima prova dell'universalità e unicità di questo tratto è molto recente: 1989 da Iannarelli, purtroppo però l'indagine è stata effettuata campionando pochissimi esempi. Iannarelli ha poi classificato le orecchie in 4 possibili forme esterne: triangolari, circolari, rettangolari e ovali.

10.1 Localizzazione dell'orecchio

Principalmente abbiamo 3 modi:

- **Individuazione di punti di interesse**: possono estratti in diversi modi, un esempio è tramite reti neurali, da notare che qui siamo interessati solo all'estrazione dei punti, quindi possiamo lavorare anche su immagini a bassa risoluzione. Una volta individuati i punti di interesse, questi vengono usati come riferimento per andare a creare il rettangolo che conterrà l'orecchio (che possiamo normalizzare)
- **Object detection**: possiamo utilizzare un object detector come ad esempio viola-jones
- **Cattura della distanza**: può essere utilizzata un'immagine che comprende anche la misura della distanza. L'orecchio conterrà le componenti con distanza minore.

10.2 Estrazione feature e riconoscimento

10.2.1 2D

Possiamo avere i seguenti approcci:

- Iannarelli: dove viene identificato il punto di origine dell'helix e viene usato come punto di partenza di 4 segmenti su cui effettuare 12 misurazioni. Il vettore di feature contiene info come il gender, l'etnicità e le 12 misurazioni. Il problema di questo approccio è che se il punto di origine è errato, tutta la misurazione è errata.
- Diagramma Voronoi
- Campi di forza: basati su proprietà fisiche dove ogni pixel viene considerato come un "attrattore"
- Jet
- SIFT

10.2.2 3D

In questo caso vengono valutati fattori come profondità e curvatura delle varie componenti di un orecchio.

10.2.3 Camera termica

- Pro: facile individuare l'orecchio, robusto a occlusioni
- Contro: sensibile a movimenti, bassa risoluzione, costo alto

11 Riconoscimento dell'iride

L'iride è una membrana muscolare bucata dalla pupilla che, data la sua alta randotipicità la rende un ottimo tratto biometrico, c'è comunque una piccola traccia di tratti ereditati ma si parla di eredità a 4 generazioni di distanza e quindi può creare problemi solo in casi eccezionali.

Tra i **vantaggi** abbiamo: alto potere discriminativo, cattura non invasiva e non varianza nel tempo.

Tra gli **svantaggi** invece: catturare sample di qualità richiede una buona attrezzatura e inoltre può risultare scomoda.

La cattura può avvenire:

- **Luce visibile**: la luce viene riflessa dall'iride generando "rumore" e rovinando la qualità del sample. Una basse illuminazione invece non permette di catturare bene i tratti dell'iride e quindi è richiesta una fase di preprocessing.
- **NIR**: la maggior della luce viene assorbita, questo permette di catturare molti più dettagli soprattutto su iridi molto scure. Come contro abbiamo che è richiesta un'apparecchiatura specifica e perdiamo informazione sul colore dell'iride.

11.1 Sistema di Daugmann

Daugmann è stato uno dei pionieri dei metodi di riconoscimento tramite iride e il suo sistema prevedeva le seguenti fasi:

- Localizzazione
- Normalizzazione
- Estrazione di feature (coding)
- Matching

Localizzazione Per localizzare i due cerchi del limbo e della pupilla, Daugmann fa uso dell'operatore integro-differenziale che funge da "edge detector circolare": l'operatore cerca percorsi circolari nei quali la varianza dei pixel viene massimizzata, variando la posizione e il raggio del cerchio che stiamo cercando.

Per la localizzazione delle palpebre avviene un procedimento analogo dove viene utilizzato un operatore simile ma che cerca archi anzichè cerchi.

Normalizzazione Normalizzare le iridi segmentate è importante in quanto ci permette di lavorare su un piano lineare dove ogni iride assume la stessa dimensione, indipendentemente dalla grandezza dei cerchi individuati. Ciò che viene usato da Daugmann è il **Rubber Sheet Model**, ovvero un modello di mapping pseudo-polare. La differenza con un mapping polare classico sta nel fatto che qui si cerca di correggere un possibile decentramento tra pupilla e limbo (dovuta ad esempio a una variazione dello sguardo) andando a utilizzare come raggio il segmento tra il cerchio della pupilla e il cerchio del limbo (per ogni angolo potremmo avere raggi più o meno lunghi), prendendo poi un numero finito di punti lungo questo.

Coding e matching Applica una serie di **gabor filter** per ottenere poi un'immagine binaria. Per il matching invece utilizza la **hamming distance**.

11.2 NICE

Il sistema di Daugmann funziona molto bene in casi di acquisizione controllata e con acqusizione NIR. E' stata quindi iniziata una challenge chiamata NICE - Noisy Iris Challenge Evalutaion, articolata in due fasi

1. **NICE I**: valutato su segmentazione dell'iride e individuazione e correzione di noise
2. **NICE II**: valutato sulla parte di encoding e matching

In NICE II, il gruppo italiano *BIPLAB* è arrivato sesto proponendo un approccio all'encoding basato su LBP (per l'estrazione delle regolarità della texture dell'iride) e BLOB (per l'estrazione delle singolarità).

11.3 IS_{IS}

IS_{IS} è un algoritmo di segmentazione sviluppato dal gruppo di ricerca della professoressa M. De Marsico che si basa sui seguenti step:

1. Segmentazione: viene applicato un filtro di posterizzazione per risaltare i contrasti nell'immagine e successivamente viene utilizzato canny edge detector per individuare dei possibili bordi. Successivamente viene utilizzato l'algoritmo di circle fitting di Taubin per individuare dei possibili cerchi candidati. I cerchi vengono poi selezionati in base ai loro score di **omogeneità** (solo per la pupilla) e **separabilità**.
2. **Estrazione feature**: LBP + BLOB

12 Riconoscimento di impronte digitali

Le impronte digitali sono state uno dei primi tratti biometrici ad essere utilizzate per identificare un soggetto, in particolare il primo approccio scientifico è dato da **Galton**, il quale ha dato una prima classificazione delle impronte digitali che definiamo come **macro singolarità** (archi, loop e spirali) e successivamente ha introdotto il concetto di minuzie che definiamo come **micro singolarità**; quest'ultime possono essere: *punti di fine*, *biforcazioni* e *delta*.

In seguito un secondo approccio è quello di **Henry**, il quale ha proposto un'altra macro classificazione con la frequenza per ogni classe:

- Spirali: 28%
- Archi: 6.6%
- Left Loop: 33.8%
- Right Loop: 31.7%

Le impronte digitali sono un tratto **randotipico**, ovvero vengono generate al momento della nascita e sono condizionate dall'ambiente e dalle superfici con le quali entra in contatto il feto. Questo garantisce un'alta univocità, anche se è stato osservato che tra gemelli, seppur le impronte siano diverse, sono comunque presenti delle similarità.

12.1 AFIS - Automatic Fingerprint Identification System

Il primo ente a collezionare impronte digitali è stata l'FBI, collezionando 10 impronte digitali, una per ogni dito, di diversi soggetti. Questo database crescendo esponenzialmente ha reso necessario l'intervento del calcolatore per verificare in modo automatico la similarità tra due impronte.

12.2 Acquisizione

L'acquisizione può avvenire principalmente secondo due modalità:

1. **Offline**: consiste in due step, la prima è la creazione dell'impronta su carta tramite inchiostro ad esempio, con la successiva digitalizzazione dell'impronta "stampata". In questa categoria rientrano le **impronte latenti**, ovvero quelle che lasciamo involontariamente sulle diverse superfici.
2. **Online**: un solo step di acquisizione, in quanto viene utilizzato un sensore di cattura che digitalizza direttamente.

Possibili problemi associati all'acquisizione sono:

- Allineamento
- Condizioni della pelle non adatte

- Pressione del dito variabile
- Troppo movimento
- Parzialità dell'impronta latente

12.3 Matching

Gli esperti di analisi di impronte digitali consigliano di prendere in considerazione 4 aspetti:

- Valutare prima un pattern globale (per scartare impronte con pattern diversi)
- Valutare la qualità, ovvero valutare le minuzie
- Valutare la quantità, ovvero ci deve essere un numero minimo di minuzie che matchano tra due impronte (consigliano almeno 12)
- Corrispondenza di dettagli che devono essere interconnessi

Di seguito vengono riportati dei possibili tipi di matching:

1. Basato su correlazione: le due immagini vengono sovrapposte e viene fatto un calcolo della correlazione dei vari pixel (iterando sui possibili allineamenti). Ha un costo computazionale molto alto ed è sensibile a trasformazioni non lineari.
2. Basato sulle ridge feature: utilizzato per immagini di bassa qualità quando quindi non è possibile estrarre le minuzie. Utilizza feature facili da estrarre come l'orientamento, la frequenza e la forma dei vari ridge. E' poco discriminativo.
3. Basato su minuzie: vengono estratte le minuzie e vengono salvate come punti in uno spazio a due dimensioni. Il metodo poi cerca un possibile allineamento che massimizzi il numero di match di minuzie.

12.3.1 Segmentazione

Si cerca di separare il background (isotropico) da ciò che è antisotropico, in questo caso le impronte. Isotropico significa che è indipendente dalla direzione (es.: ruotando uno sfondo bianco abbiamo sempre la stessa immagine).

12.3.2 Macro features

Una volta segmentata l'impronta possiamo iniziare a estrarre delle macro features come:

- **Flusso dei ridge**: descritto da una directional map; si lavora su una griglia e non sull'immagine intera (quindi consideriamo i vicini di un certo pixel su questa griglia)

- **Densità dei ridge**: descritto da una density map.

Per estrarre le singolarità un metodo elegante è quello di utilizzare il **Poincarè index**, ovvero un valore che può assumere solo i seguenti valori:

- 0 $\implies$ nessuna singolarità
- 180 $\implies$ loop
- -180 $\implies$ delta
- 360 $\implies$ spirale

E indica la rotazione totale dei vettori su una certa curva.

12.3.3 Micro features

Per estrarre le minuzie sono necessari i seguenti step:

1. Binarizzazione dell'immagine
2. Appiattimento dei ridge: i ridge vengono appiattiti a grandezza un 1 pixel
3. Localizzazione delle minuzie: vengono individuati i pixel corrispondenti alle varie minuzie

Per l'ultimo punto possiamo effettuare un'analisi del crossing number di un certo pixel $cn(p)$

$$cn(p) = \frac{1}{2}\sum_{i=1}^{8} |val(p_{i \ mod \ 8}) - val(p_{i-1})|$$

Se

- $cn(p) = 0$: isola
- $cn(p) = 1$: terminazione
- $cn(p) = 2$: punto interno di un ridge
- $cn(p) = 3$: biforcazione
- $cn(p) > 3$: minuzia complessa

Un'altra feature che possiamo prendere in considerazione è la distanza tra minuzie espressa dal **ridge count**, ovvero il numero di ridge tra una minuzia e un'altra (solitamente i due punti scelti sono punti rilevanti)

12.4 Approccio ibrido

E' possibile utilizzare un approccio ibrido dove l'estrazione di minuzie aiuta a trovare il miglior allineamento possibile tra probe e template e poi utilizziamo una banca di gabor filter per l'estrazione delle feature.

12.5 Impronte finte

E' facile creare delle possibili impronte artificiali con dei materiali come il silicone o gelatina, ma ogni materiale ha risposte diverse al sensore che stiamo utilizzando (e quindi potremmo capire se si tratta di un possibile attacco). Per verificare la "liveness" di un dito possiamo sfruttare ad esempio sensori che riescono a catturare lo scorrere del sangue.

13 Approcci multibiometrici

Gli approcci multibiometrici ci aiutano a contrastare attacchi di spoofing, in quanto l'attaccante dovrà bypassare più livelli. Per sistema multibiometrico possiamo intendere l'utilizzo di:

1. più sensori di acquisizione
2. più tratti biometrici (generalmente intendiamo questo)
3. probe multipli
4. algoritmi diversi
5. istanze multiple

La fusione di più tratti può avvenire in diversi punti:

- A livello del **sensore**: in questo caso si utilizzano più acquisizioni da fondere in un unico modello
- A livello di **feature**: vengono fusi i vettori di feature. Potremmo avere problemi di dimensionalità (magari con dati ridondanti) e di incompatibilità.
- A livello di **score**: vengono fusi i punteggi, i quali a loro volta possono essere di 3 tipi:
  - Astratto: viene restituita la classe del soggetto. In questo caso la fusione avviene tramite votazione, scegliendo la maggioranza.
  - Rank: viene restituita una lista dei rank per ogni classe. La fusione è possibile effettuarla tramite "borda count". Supponiamo di avere n classificatori e k rank (quindi k classi). Ogni classificatore produrrà la sua lista contenente i k rank. Per ogni classe prendiamo i rank ottenuti dagli n classificatori e li sommiamo. La classe con il valore più alto sarà quella restituita come risultato finale.
  - Misura: ogni classificatore da in output uno score basato su una misura. L'unica difficoltà qui è normalizzare i vari score in quanto probabilmente ognuno utilizzerà una misura diversa. Una volta fatto questo possiamo restituire lo score medio o, sapendo il livello di "confidence" dei singoli classificatori, possiamo usare quel valore come peso per una somma pesata.

- A livello di **decisione**: viene fusa la decisione booleana "si/no". In questo possiamo o prendere l'AND di tutte le decisioni o l'OR. Un altro approccio è tramite voto a maggioranza.