

Appunti “Biometric Systems”

Prof. Maria De Marsico. Appunti scritti da [Alessio Lucciola](#) durante l'a.a. 2022/2023. I seguenti appunti sono incompleti e trattano solo i macroargomenti in generale. Dovrebbero comunque essere abbastanza per la prova orale.

Sei libero di:

- Condividere: Copiare e distribuire il materiale in qualsiasi mezzo o formato.
- Adattare: Trasformare o modificare il materiale.

Rispettando i seguenti termini:

- Attribuzione: E' necessario fornire i crediti appropriati, un collegamento alla licenza e indicare se sono state apportate modifiche.
- Fine non commerciale: Non è possibile utilizzare il materiale per scopi commerciali.

Le note possono contenere errori di battitura. Se ne vedi uno, puoi contattarmi utilizzando i link nella [pagina Github](#).

Se questi appunti ti sono utili, potresti prendere in considerazione l'idea di [offrirmi un caffè](#) ☺.

1. Face Localization

Le proprietà della faccia per effettuare la face localization sono le seguenti: Il volto ha un alto valore di **Universalità** (banalmente tutti hanno un volto), **Collezionabilità** (basta scattare una foto) e **Accettabilità** (in generale, tutti sono disposti a farsi scattare una foto). Ha un valore medio per **Permanenza** (più o meno la faccia rimane uguale per un certo numero di anni) e infine faccia ha un valore basso per **Unicità** (questo si riferisce alla possibilità di avere tratti simili con i parenti, ad esempio), **Performance** (in un ambiente non controllato) e **Circonvenzione** (è facile da falsificare). Potremmo anche considerare altre proprietà come il costo (che è basso), il livello di igiene (anch'esso basso) e il FTE rate che sta per "failed to enroll" (ed è bassa per il riconoscimento facciale).

I problemi principali che possiamo avere nel riconoscimento facciale sono le **Variazioni Intra-personali** (Posa, illuminazione ed espressione (PIE) sono tutti fattori che possono aumentare la quantità di variazioni anche tra template della stessa persona. Possiamo anche considerare l'effetto causato dall'invecchiamento. L'età è difficile da tenere in considerazione poiché di solito non abbiamo database verticali in cui ci sono template appartenenti alla stessa persona ma con un'età diversa) e le **Somiglianze interpersonali** (può capitare che i parenti siano molto simili ma può capitare anche tra persone che non hanno niente a che fare tra loro).

Solitamente la struttura di un riconoscimento facciale è la seguente: Il primo passo è l'**acquisizione dell'immagine** del volto che, a seconda della distanza, può contenere solo il volto o un ambiente ampio. Questo è il caso per cui l'immagine viene prima migliorata. Il miglioramento dell'immagine si riferisce ad esempio ad aumentare la nitidezza di un'immagine, cercare di eliminare alcune sfocature quando possibile, aumentare il contrasto e così via.

Successivamente, abbiamo il **rilevamento** e la **localizzazione**. Spesso ci riferiamo al rilevamento e alla localizzazione come se fossero più o meno la stessa operazione, ma c'è una differenza. In generale il rilevamento restituisce solo una risposta binaria: sì → questo oggetto è nell'immagine, no → questo oggetto non è nell'immagine. La localizzazione fornisce anche la posizione esatta dell'elemento all'interno dell'immagine. Una volta localizzato o rilevato il volto, possiamo anche tagliare la regione di interesse (ROI) per ridurre il tempo di elaborazione limitando la successiva fase di elaborazione solo alle regioni pertinenti. Infine, abbiamo l'estrazione delle caratteristiche e la costruzione del template.

Per quanto riguarda la **localizzazione delle facce**, il problema è che data una singola immagine o una sequenza video vogliamo rilevare la presenza di uno o più volti e individuare la loro posizione all'interno della singola immagine e nel fare ciò è necessario essere indipendenti rispetto a posizione, orientamento, scala, espressione (possibilmente diversa per i diversi soggetti dell'immagine), illuminazione e sfondo disordinato (con molto rumore).

Esiste un algoritmo, per effettuare il face detection, chiamato “**Algorithm A by Hsu, Mottaleb and Jain, 2002**”.

Tale algoritmo si basa su diverse fasi:

- **Illumination compensation:** Il tono della pelle dipende dall'illuminazione generale della scena, quindi per normalizzare i colori si usano **riferimenti di bianco** (reference white). Consideriamo i pixel con il 5% superiore della luminanza, valori come il bianco di riferimento se il numero di questi pixel bianchi di riferimento è maggiore di 100. I componenti rosso, verde e blu di un'immagine a colori vengono regolati in modo che questi pixel bianchi di riferimento vengono scalati al livello di grigio di 255. Quindi il bianco di riferimento diventa il valore 255 e tutti gli altri colori vengono scalati di conseguenza.
- **Color space transformation:** La modellazione del colore della pelle richiede la scelta di uno spazio di colore appropriato e l'identificazione di un cluster associato al colore della pelle in questo spazio. Usiamo lo spazio YCbCr poiché è ampiamente utilizzato negli standard di compressione video. Poiché il colore della tonalità della pelle dipende dalla luminanza, trasformiamo in modo non lineare lo spazio colore YCbCr per rendere il luma del cluster di pelle indipendente. Ciò consente anche un rilevamento affidabile dei colori della tonalità della pelle scuri e chiari.

Localization based on skin model: abbiamo due alternative per effettuare questo step, cioè la Variance-Based Segmentation o le connected components:

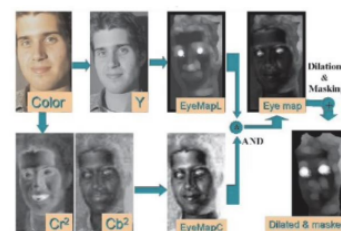
- **Variance-Based Segmentation:** il metodo più semplice di segmentazione dell'immagine è il thresholding method. La chiave è selezionare il miglior threshold, che può essere fatto con il metodo della max entropy, metodo di Otsu (massima varianza) o il k-means clustering. Per il metodo di Otsu, sia i un'immagine in scala di grigi con L livelli di grigio di dimensione $M \times N$, $f(x, y)$ il valore di grigio del pixel nel punto (x, y) , n_i il numero di pixel con il grigio livello i , C_0 e C_1 le classi di livello di grigio prodotte da un threshold t (tipicamente oggetto e sfondo), allora il threshold ottimale è quello che minimizza la varianza intra-class tra le due classi.
- **Connected Components:** I toni dei pixel della pelle sono segmentati iterativamente usando colori locali e la vicinanza spaziale, raggruppando quindi le varie componenti (occhi, bocca, etc.), fino a generare insiemi di candidati per il viso. In generale, una componente connessa è una regione senza buchi o una regione tale che da ogni pixel è possibile raggiungere qualsiasi altro pixel senza uscire dalla regione.

Possiamo quindi localizzare le caratteristiche facciali:

- **Localizzazione degli occhi:** L'algoritmo costruisce due mappe differenti per gli occhi:

- CHROMINANCE MAP: Questa viene costruito prendendo in considerazione il fatto che la regione intorno agli occhi è caratterizzata da alti valori di Cb e bassi valori di Cr
- b) LUMINANCE MAP: Gli occhi generalmente contengono sia zone chiare che zone scure che possono essere evidenziate dai due operatori morfologici: Dilation and Erosion con una struttura semisferica (struttura simile all'occhio).

Successivamente queste due mappe vengono combinate insieme tramite l'operatore logico AND e al risultato ottenuto viene applicato un processo di masking e dilation in grado di esaltare la posizione degli occhi.



- **Localizzazione della bocca:** l'algoritmo costruisce la mappa della bocca basandosi sul fatto che, nella regione della bocca, la componente Cr è maggiore di quella Cb, e la risposta a Cr/Cb è bassa, mentre la risposta a Cr^2 è alta.
- **Contorno della faccia:** l'algoritmo analizza tutti i triangoli composti da due occhi candidati e una bocca candidata. Ogni triangolo è verificato in questo modo:
 - Variazioni di luma e media del gradiente di orientamento delle macchie contenenti occhi e bocca;
 - Geometria e orientamento del triangolo,
 - Presenza di un contorno del viso attorno al triangolo,
Il triangolo con lo score più alto viene selezionato.

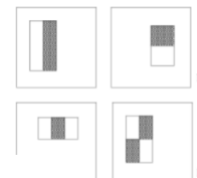
L'**operatore di dilatazione** prende due dati come input: l'immagine e un kernel che determina l'effetto della dilatazione dell'immagine in input. La definizione matematica di dilatazione per immagini binarie è la seguente: supponiamo che X sia l'insieme di coordinate euclidee corrispondente all'immagine binaria in input e che K sia l'insieme di coordinate del kernel. Sia K_x la traslazione di K in modo che la sua origine sia in x , allora la dilatazione di X per K è semplicemente l'insieme di tutti i punti x tali che l'intersezione di K_x con X **non è vuota**.

La definizione matematica di **erosione** è la seguente: supponiamo che X sia l'insieme di coordinate euclidee corrispondente all'immagine in input, e che K sia l'insieme di coordinate del kernel e sia K_x la traslazione di K così che la sua origine è in x , allora l'erosione di X da parte di K è semplicemente l'insieme di tutti i punti x tali che K_x è un sottoinsieme di X .

L'algoritmo **Viola-Jones**, richiede la creazione di un classificatore che viene inizialmente addestrato utilizzando istanze della classe da identificare (esempi positivi) e istanze di immagini che non contengono alcun oggetto della classe e che possono causare degli errori (esempi negativi). Il training è progettato per estrarre diverse caratteristiche degli esempi e seleziona quelle più discriminanti (elementi che possono meglio suggerire la presenza del faccia rispetto ad altri tipi di elementi geometrici). Durante il training alcuni errori possono essere i **miss** (sono le immagini false negative in cui non viene rilevato un oggetto presente) e i **falsi allarmi** (un oggetto viene rilevato ma non è presente) e possono essere diminuiti ripetendo il training e aggiungendo nuovi esempi sia positivi che negativi. In questo tipo di valutazione è importante avere il ground-truth cioè bisogna sapere quali sono le regioni contenenti volti ed è anche importante considerare accuratamente il risultato del sistema nel senso che non è sufficiente contare solo se il numero di regioni di interesse è uguale al numero di facce previste. Questo perché alcune di queste regioni di interesse potrebbero essere falsi allarmi mentre si perde un vero volto.

Viola-Jones utilizza **AdaBoost** che è una tecnica di training che ha lo scopo di apprendere la migliore sequenza di **weak classifier** e i loro corrispondenti pesi. Un weak classifier è un linear classifier è un classificatore che tenta di dividere due o più classi utilizzando solo elementi lineari. Un weak classifier è un classificatore con una precisione probabilmente molto bassa (bassa come il lancio di una moneta). Vogliamo comporre uno **strong classifier** usando weak classifier. Lo strong classifier è rappresentato dalla combinazione o dalla somma dei valori restituiti dai weak classifier ponderati in base alla sua rilevanza nella decisione finale (restituiscono un valore nell'insieme $\{-1, +1\}$ in base alla presenza o meno di una certa caratteristica). In generale -1 rappresenta un risultato negativo (non volto), mentre $+1$ è un rilevamento positivo (volto). Una volta effettuata la somma pesata per limitarci nel range $(-1, +1)$, dobbiamo dividere questa somma per la somma dei pesi. Quindi all'inizio tutti i campioni hanno lo stesso peso (ad es. 1), perché all'inizio non abbiamo informazioni su quale sia il peso di ogni pattern. Dopo l'iterazione m , viene assegnato un peso w_m maggiore ai pattern che sono risultati più difficili da classificare, in modo che nella successiva $m+1$ -esima iterazione tali pattern **ricevono maggiore attenzione**.

Per quanto riguarda le immagini, si utilizzano gli **Haar Classifier**: un algoritmo che identifica gli oggetti in un'immagine e in un video sfruttando tecniche di machine learning. Si inizia con il calcolo delle **Haar Features**: questi sono dei weak classifiers basati su features semplici (rettangolari). Sono elementi rettangolari che possono essere divisi verticalmente in chiaro e scuro oppure divisi orizzontalmente in chiaro e scuro o con pattern misti. Ogni feature si trova in una sottoregione di una sottofinestra dell'immagine (finestra scorrevole sull'immagine). Le feature vengono applicate modificando le dimensioni, la forma e la posizione nella finestra. Il valore restituito dal sistema per queste Haar Features è la **somma dei valori dei pixel che rientrano nell'area bianca meno la somma dei valori dei pixel che rientrano nell'area nera**.



Possiamo stabilire un threshold per questa differenza in modo che possa in qualche modo restituire un risultato sì/no in base alla presenza del tipo di pattern nell'immagine originale. Questo tipo di calcolo può essere impegnativo dal punto di vista computazionale infatti con M filtri, N campioni e T possibili threshold il costo è $O(MNT)$. Quindi usiamo le **Integral Images** per trovare un modo per calcolare rapidamente il valore di quella differenza. Data un'immagine, l'immagine integrale in un punto (x, y) , è la somma dei pixel sopra e a sinistra di (x, y) . Quindi integral images e haar features sono collegate dalla necessità di calcolare la somma di pixel. Possiamo usare **AdaBoost** per selezionare un piccolo sottoinsieme di tutte le possibili features per costruire un buon classificatore. Questa è una tecnica di allenamento che ha lo scopo di apprendere la migliore sequenza di weak classifiers e i loro pesi corrispondenti. L'ideale sarebbe ridurre al minimo il risultato per ogni passaggio scartando il maggior numero possibile di regioni che non contengono una faccia e continuando con il minor numero di possibili regioni con immagine vera da sottoporre al classificatore successivo. Il problema è che, una volta scartata una regione, questa non verrà più considerata e restituiranno direttamente un risultato parziale senza volto per quella regione dell'immagine. I risultati positivi del primo classificatore attivano la valutazione di un secondo classificatore (più complesso) e così via. Un esito negativo in qualsiasi momento porta al rifiuto immediato della sottofinestra quindi una volta rifiutata, una certa sottofinestra in un certo punto verrà etichettata come "non-faccia" senza passare al classificatore successivo. Solo le finestre dell'immagine che hanno eseguito correttamente la catena complessiva verranno classificate come "faccia". Ogni classificatore è addestrato sui falsi positivi delle fasi precedenti. La

localizzazione dei volti avviene analizzando sottofinestre consecutive (sovrapposte) dell'immagine come input e valutando per ognuna di esse se appartiene alla classe dei volti.

Utilizzando questa tecnica fa sì che Viola-Jones sia **lento nel training** e abbastanza **veloce ed efficiente nelle operazioni real time**.

Per valutare un sistema di localizzazione si deve tenere conto dei **False Positive** (la percentuale di finestre classificate come volti che non contengono alcun volto) e dei **Volti Non Localizzati** (la percentuale di volti che non sono stati identificati). Possiamo avere un'immagine con più volti all'interno quindi dobbiamo anche tenere conto di quale fosse il numero corretto di volti all'interno di una singola immagine. Possiamo anche perfezionare ulteriormente la valutazione della localizzazione utilizzando il **C-Error** che non si basa solo sulla possibile errata classificazione della regione dell'immagine, ma anche sulla corretta localizzazione.

2. Face Recognition 2D

Un'immagine può essere rappresentata come un insieme di punti in uno spazio multidimensionale. Una funzione RGB definita in uno spazio Cartesiano può assegnare un valore tra 0 e 255 a tutte le posizioni (x, y) nel piano. Questi valori possono essere salvati in una matrice $w \times h$ nella quale ogni elemento della matrice rappresenta il valore di un pixel. Oppure si possono salvare in un vettore uni-dimensionale $n = w \times h$ ad esempio concatenando ogni riga dell'immagine una dopo l'altra.

Più in generale, un'immagine I può essere rappresentata come un punto in uno **spazio multidimensionale** che può essere:

- **Image Space**: Ogni immagine viene rappresentata in un vettore $w \times h$ tipicamente con dimensioni piuttosto elevate. Per gestire ed eventualmente comparare tali immagini bisogna prima normalizzarle in maniera che abbiamo la stessa dimensione, ma facendo ciò viene fuori un fenomeno noto come **curse of dimensionality**: Immaginiamo che abbiamo numerose dimensioni e ogni dimensione corrisponde ad una posizione nello spazio. A causa della normalizzazione abbiamo spazi molto dispersi (scattered spaces) e qui le distanze tra i punti sono molto difficili da valutare perché maggiore è la dimensionalità (più disperso è lo spazio), più i punti possono essere distanti tra loro anche se sono rappresentano elementi vicini. Questo porta ad alcune conseguenze come: alcune dimensioni (pixel) non portano informazioni significative ai fini del riconoscimento, è necessaria una grande quantità di immagini di training per avere una conoscenza adeguata dello spazio e il confronto è un'operazione molto costosa in termini di complessità computazionale. Quindi, in questo caso, ci sono tecniche per la riduzione della dimensionalità che vengono adottate.
- **Feature Space**: Dato un vettore, possiamo associare una feature estratta dall'immagine ad ogni elemento. Ogni elemento viene poi normalizzato in maniera tale che possano rappresentare così una sorta di coordinata nello spazio.

Image Space

Metodi come il **Principal Component Analysis (PCA)** mirano a ridurre dati con un'alta dimensionalità senza avere una grande perdita di informazioni, identificando le dimensioni più informative e ottenendo **sottospazi rilevanti**. La **Principal Component Analysis (PCA)** è una procedura che utilizza una trasformazione ortogonale per convertire un insieme di variabili eventualmente **correlate** (in cui è coinvolta una certa ridondanza) in un insieme di variabili non correlate (nessuna ridondanza) chiamate **componenti principali**. In questo modo, lo spazio delle caratteristiche n -dimensionali viene proiettato su un sottospazio **k -dimensionale**. Le componenti principali sono ortogonali quando sono **autovettori (eigenvector)** della **matrice di covarianza** (covariance matrix).

Il valore di k ottimale viene calcolato a partire da un insieme di m immagini linearizzate n -dimensionali che costituiscono il training set (cioè prendiamo ogni riga e le incolonniamo via via fino ad ottenere un vettore $1 \times n$). A questo punto calcoliamo il **mean vector**.

Con il mean vector prendiamo gli m samples (i vettori n -dimensionali) sommiamo le componenti per poi andare a dividere questa somma per il numero di sample. In questo modo possiamo andare a catturare le caratteristiche comuni di tutte le immagini. Il risultato è che in questo modo sembra che sia stato

$$TS = \{x_i \in \mathbb{R}^n | i=1, \dots, m\}$$

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

creato un nuovo volto. La parte più definita dell'immagine è quella rappresentante gli occhi. Questo accade perché i soggetti, pur essendo notevolmente diversi tra loro, hanno una certa somiglianza nella parte degli occhi.

Quindi, una volta che abbiamo il training vector e abbiamo calcolato il mean vector, per ogni immagine del dataset vi sottraiamo la relativa mean face. Calcoliamo quindi la **covariance Matrix**: Prendiamo ogni valore x_i e lo sottraiamo al mean vector e lo moltiplichiamo per la trasposta della stessa sottrazione. Si sottrae il mean vector per ogni valore x_i poiché al suo interno (il mean vector) contiene informazioni ridondanti.

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

La dimensione della matrice C di covarianza è $n \times n$. Una volta che abbiamo la matrice di covarianza, dobbiamo selezionare gli autovettori e per farlo andiamo a controllare i rispettivi autovalori. Ordiniamo gli autovalori in maniera decrescente tenendo conto degli autovettori ai quali corrispondono e infine selezioniamo i k autovettori corrispondenti ai k autovalori più alti della matrice. Il nuovo spazio k-dimensionale è definito dalla matrice di proiezione le cui colonne sono i k autovettori di C corrispondenti ai k autovalori più alti. Questi autovettori rappresentano gli elementi con la **maggiore varianza** tra le immagini nel training set.

Sirovich e Kirby furono i primi ad utilizzare PCA per il face recognition.

Loro hanno dimostrato che tutte le facce possono essere rappresentate lungo le **eigen pictures coordinate space**. Per fare ciò prima prendiamo un set di immagini dal training set, quindi calcoliamo le eigen pictures che sono in grado di rappresentare il nuovo spazio (si prendono le immagini più rappresentative) e ogni immagine originale può essere proiettata su questo nuovo spazio dimensionale.

Successivamente ci si è resi conto che le proiezioni lungo le proprie immagini possono essere utilizzate come un modo per riconoscere i volti. E' stato sviluppato un sistema di riconoscimento facciale che utilizza le eigenfaces, corrispondenti agli autovettori associati agli autovalori più grandi della matrice di covarianza. Il sistema riconosce volti confrontando le loro proiezioni lungo le eigenfaces con quelle delle immagini dei volti degli individui conosciuti, quindi l'identificazione del volto viene effettuata in questo spazio ridotto. Possiamo classificare questo approccio nella classe della **Holistic Classification** perché tiene conto del volto complessivo senza considerare alcuna regione che fornisca alcun tipo di informazione speciale.

Le proiezioni vengono eseguite moltiplicando la matrice di proiezione trasposta per il vettore originale a cui è stata prima sottratta la media. L'idea di base è che invece di confrontare le facce pixel per pixel, che forniscono un risultato estremamente inaffidabile,

$$Proj(x) = \varphi_k^T (x - \bar{x})$$

possiamo piuttosto confrontare tali immagini una volta che sono proiettate sul sottospazio dimensionale costruito. In sostanza, per capire se un nuovo soggetto è presente nel database iniziale consideriamo la sua proiezione nel nuovo spazio e calcoliamo la distanza rispetto alle altre facce. Nel caso sia abbastanza vicino ad una di esse avviene il riconoscimento. Ciò significa che non è obbligatorio che ogni singolo soggetto nella galleria partecipi nel training set. Ciò che è importante è che le immagini utilizzate per costruire il nuovo spazio siano rappresentative delle variazioni più rilevanti tra i soggetti.

Quali sono i problemi con PCA?

PCA è utile per la rappresentazione nel senso che è molto facile mappare il nuovo feature vector sull'immagine originale, ma manca di un vero **potere discriminante**. La separazione di eigenfaces (scattering) dipende non solo dalle variazioni inter-class (utili per la classificazione) ma anche dalle variazioni intra-class. Nel senso che la varianza che si ottiene lungo le dimensioni rappresentate dalle eigenfaces può essere dovuta sia a differenze inter-class che intra-class. Ma quando vogliamo realizzare il riconoscimento, siamo molto più interessati alla inter-class separation. Quindi siamo interessati a quella porzione di varianza che è dovuta alla separazione tra le classi piuttosto che alle differenze all'interno delle classi. Quando ci sono variazioni significative di PIE (Pose, Illumination, Expression), esse possono interferire nella determinazione dell'identità degli individui, così le classi possono non essere separate correttamente.

Per risolvere questi problemi, sono state proposte le **Fisherfaces** utilizzate nel contesto della **Linear Discriminant Analysis (LDA)**. Ci sono varie **differenze tra PCA e LDA**: PCA è **unsupervised**. Quando raccogliamo campioni da utilizzare per costruire il nuovo feature space, non sappiamo quale sia il soggetto che è stato raffigurato in ogni immagine. Raccogliamo solo un numero di campioni abbastanza rappresentativi da soggetti diversi e proviamo a creare un nuovo feature space su cui mappiamo ogni immagine della galleria e ogni probe in arrivo prima di confrontarli. Per questo motivo PCA è molto più **incline ai falsi rifiuti**, nel senso che è più difficile per PCA riconoscere lo stesso individuo in una situazione diversa.

L'idea è quella di passare a LDA che è un metodo **supervised** perché tiene conto non solo delle raccolte di campioni ma anche di una partizione di tali campioni nelle diverse classi rappresentate. In **Linear Discriminant Analysis (LDA)** si etichettano i campioni per ogni soggetto che partecipa al training set. Questo tipo di partizionamento aiuta ad apprendere meglio quali sono gli elementi nel nuovo spazio che meglio si riferiscono alle variazioni intra-class rispetto a quelli che in realtà si riferiscono alle variazioni inter-class (che sono quelle che ci interessano).

LDA è una **tecnica di riduzione di dimensionalità lineare** (come PCA) e **supervisionata** il cui obiettivo è **massimizzare la separazione tra le classi e mantenere le informazioni discriminatorie** il più possibile. La trasformazione dello spazio è determinata sulla base di un criterio di ottimizzazione che ha lo scopo di massimizzare la separazione tra le classi nello spazio ridotto quindi si effettuano una sequenza di azioni che all'inizio sono molto simili al PCA ma queste vengono effettuate su classi differenti.

Un approccio è il seguente:

- Partiamo da un insieme di m campioni n -dimensionali che costituiscono il training set $TS = \{x_i \in \mathbb{R}^n : i = 1, \dots, m\}$ dove m è la cardinalità del TS.
- Diversamente da PCA, il training set è partizionato in base alle classi.
- Cerchiamo di ottenere uno scalare y proiettando i campioni x su una retta tale che $y = w^T x$.
- Nel caso più semplice, considero due classi P_1 e P_2 con vettori m_1 e m_2 (nel caso generale avremo s classi).
- Calcoliamo i mean vector di ciascuna classe nei due spazi:

$$\mu_i = \frac{1}{m_i} \sum_{j=1}^{m_i} x_j \quad \tilde{\mu}_i = \frac{1}{m_i} \sum_{j=1}^{m_i} y_j = \frac{1}{m_i} \sum_{j=1}^{m_i} w^T x_j = w^T \mu_i \quad \text{<- Retta nel nuovo spazio}$$

Il mean vector nel nuovo spazio può essere anche rappresentato come il mean vector della proiezione del vettore iniziale sul nuovo spazio (ovvero di x_j che sta sopra di lui).

- Adesso vogliamo massimizzare la distanza tra i mean vector delle due classi:

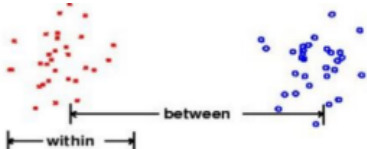
$$J(w) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |w^T(\mu_1 - \mu_2)|$$

μ_1 (barrato) meno μ_2 (barrato) non sono però delle buone misure perché non tengono in considerazione la standard deviation (non ci informa su quanto i dati varino) tra le classi per questo motivo si usano le **scattering matrices**.

Riass: The starting step of the LDA reduction process resembles very closely the step that was applied in PCA, that is: we collect all the samples, but this time belonging to the same class, and then we compute the average vector. Consider that whenever we're able to represent our feature vectors as points in a space, when we compute the mean vector this mean vector can represent the centroid of the cluster of points collecting a certain class. The centroid is relevant in order to determine how spread are the classes among them; but the problem is that the centroid is not sufficient because another element that must be taken into account is the spread of the samples in the same class around the centroid.

Un approccio migliore consiste nel massimizzare il rapporto tra la varianza tra le classi e la varianza all'interno della classe a partire dalle **matrici di scattering** (separazione):

- Si parte da un Training Set (TS);
- Diversamente da PCA, il training set è partizionato in base alle classi;
- Per ogni classe P_i , calcoliamo il mean vector (un "centroide" per ogni classe in modo simile a PCA sull'intero insieme) e la media delle medie ("centroide" di "centroidi").
- Calcoliamo la matrice di covarianza per ogni classe P_i (in modo simile a PCA sull'intero insieme).
- Ora possiamo calcolare S_w ed S_b . **S_w (within-class scatter matrix)** è data dalla somma pesata della matrice di covarianza. **S_b (between-class scatter matrix)** è calcolata in maniera simile: prendiamo il mean vector per ogni classe e vi sottraiamo la media dei mean vector.



$$\mu_i = \frac{1}{m_i} \sum_{j=1}^{m_i} x_j \quad \mu_{TS} = \frac{1}{m} \sum_{i=1}^s m_i \mu_i$$

$$C_i = \frac{1}{m_i} \sum_{j=1}^{m_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

$$S_W = \sum_{i=1}^s m_i C_i$$

$$S_B = \sum_{i=1}^s m_i (\mu_i - \mu_{TS})(\mu_i - \mu_{TS})^T$$

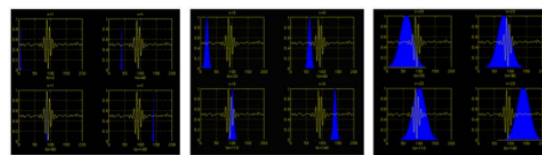
Feature Space

Finora abbiamo considerato lo “image space”, dove ogni pixel nell'immagine originale rappresenta una dimensione in uno spazio molto ampio. D'altra parte, possiamo anche estrarre diversi tipi di features da un'immagine. Le caratteristiche rilevanti di un volto possono essere estratte applicando filtri o trasformazioni ciascuno adatto a rilevare proprietà particolari. Quando la dimensionalità dei feature vector estratti è elevata, possono subire un processo di riduzione della dimensionalità utilizzando una delle tecniche precedentemente descritte.

Si introducono adesso le Wavelet Transform:

Il **Fourier Transform (FT)** fornisce informazioni sulla frequenza del segnale (in questo caso le immagini), il che significa che ci dice quanto di ciascuna frequenza esiste nel segnale, ma non ci dice quando esistono nel tempo. Gli spettri di due segnali completamente diversi possono sembrare molto simili, poiché il Fourier Transform fornisce il contenuto spettrale del segnale, ma non fornisce informazioni su dove nel tempo compaiono tali componenti spettrali. Pertanto, il Fourier Transform può essere utilizzato per segnali non stazionari, cioè se siamo interessati solo a quali componenti spettrali esistono nel segnale, ma non siamo interessati a dove si verificano.

Mentre la **Wavelet Transforms (WS)** è in grado di fornire simultaneamente le informazioni di tempo e frequenza, fornendo quindi una rappresentazione tempo-frequenza del segnale. La linea gialla è il segnale originale, mentre quella blu è la wavelet. Viene fatta passare una **Onda Madre** (quella in blu) nel dominio del tempo (linea gialla) con diverse dimensioni. Maggiore sarà la dimensione dell'onda madre maggiori saranno le feature catturate e discorso analogo vale per la dimensione minore della Onda Madre.



La linea gialla è il segnale originale, mentre quella blu è la wavelet

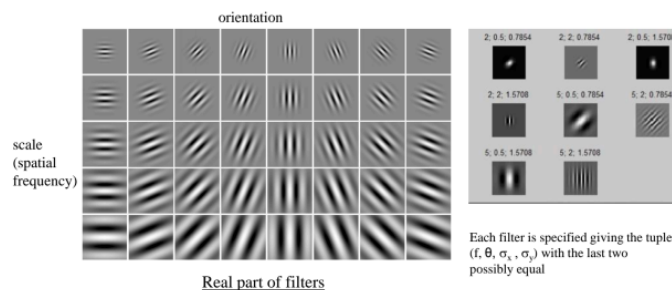
EX. Da un'immagine 2D vogliamo conoscere i livelli di grigio e inoltre dove questi livelli appaiono nell'immagine.

Gli algoritmi Wavelet elaborano i dati a diverse scale o risoluzioni. Possiamo cambiare il periodo della wavelet, cioè la frequenza (quante volte l'onda attraversa l'asse 0 nell'unità di tempo). Se aumentiamo la frequenza abbiamo una **wavelet stretta**, perché l'aumento della frequenza significa che attraversano lo 0 più volte. Al contrario se diminuiamo la frequenza avremo una wavelet larga. In generale, se osserviamo un segnale con una “grande finestra” (grande onda blu), noteremo feature grossolane. Allo stesso modo, se osserviamo un segnale con una “piccola finestra” (piccola onda blu), noteremo feature più piccole.

Una sorta di wavelet sfruttata con le immagini sono i cosiddetti **filtri Gabor**, cioè filtri lineari per effettuare **edge detection** (cioè la rilevazione dei bordi). Si tratta di filtri bidimensionali che vengono applicati all'immagine attraverso l'operazione di convoluzione. Funziona come un filtro passa-banda (**bandpass filter**) (è un filtro che cattura solo le frequenze in un certo intervallo e le taglia tutto ciò che è al di sotto o al di sopra dell'intervallo di frequenza) per la distribuzione della frequenza locale spaziale, ottenendo una risoluzione ottimale sia nel dominio spaziale che in quello della frequenza. Un feature vector è ottenuto dalla convoluzione dell'immagine con un **banco di filtri Gabor** (diversi orientamenti e diverse scale) per catturare diversi tipi di pattern.

Nella prima riga, quello che cambia è l'orientamento del filtro.

Successivamente si può modificare la scala del filtro che rappresenta la frequenza spaziale. Combinando diversi orientamenti e scale, è possibile ottenere un enorme banco di filtri. Quindi effettuiamo la convoluzione di questi filtri con diverse regioni dell'immagine facendoli scorrere sull'immagine stessa e calcoliamo la risposta per rilevare la presenza di pattern di un certo tipo. Le funzioni Gabor 2D migliorano i contorni dei bordi. Ciò corrisponde a migliorare l'occhio, bocca e bordi del naso, e anche talpe, fossette, cicatrici, ecc. Se la convoluzione è eseguita su tutti i pixel dell'immagine e



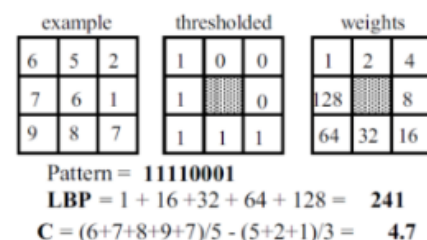
prendiamo il risultato complessivo, la **dimensionalità è piuttosto alta** (dimensione dell'immagine). In alternativa può essere eseguito su una **griglia regolare di punti** i cui vertici sono vicini a regioni di interesse (come gli occhi, o solo sulle **regioni facciali con la maggiore informazione**). Chiaramente siccome la griglia è fissa, potrebbe non andare bene per qualsiasi immagine. Questo tipo di processo è computazionalmente costoso perché dobbiamo eseguire la convoluzione tante volte quanta è la dimensione della cardinalità del banco di filtri. Inoltre, c'è un altro problema: il risultato di ogni convoluzione è esso stesso un'immagine con la stessa dimensione di quella originale, in modo che questa possa essere considerata come una **rappresentazione feature-based dell'immagine originale**.

Un'applicazione del filtro Gabor è **Elastic Bunch Graph Matching (EBGM)**:

- La rappresentazione dell'oggetto di base è un grafico etichettato, definito come **grafico a grappolo (bunch graph)**, uno per ogni posa, che raccoglie informazioni specifiche sul particolare modello;
- Gli archi sono etichettati con le informazioni sulla distanza ed i nodi sono etichettati con le risposte Wavelet di Gabor raccolte localmente nei **jet**, ovvero una rappresentazione compatta e robusta di una distribuzione locale di livelli di grigio;

La geometria di un volto è codificata dagli archi, mentre la distribuzione dei valori di grigio viene codificata dai nodi. Per descrivere una vista bidimensionale è sufficiente un solo grafo, mentre per rappresentare più viste o per ottenere una vista 3D occorre integrare più grafi. Un **Face Bunch Graph (FBG)** può rappresentare volti in generale ed è progettato per coprire molteplici variazioni nell'aspetto dei volti. Combina informazioni di un certo numero di grafi rappresentanti un volto. I nodi sono etichettati con insiemi di jet detti **bunches**, e gli archi sono etichettati con i vettori di distanza. Durante il confronto di un'immagine, si seleziona il jet più appropriato in ogni bunch. Un FBG è costruito a partire da un insieme rappresentativo di grafi modello (es. 70) aventi la stessa posa (es. frontale, di profilo, ...), dunque la stessa struttura. Ogni nodo è etichettato con tutti i jet presi dai modelli nei medesimi punti. Nuove facce possono essere rappresentate prendendo jet da differenti modelli, ad esempio occhio sinistro da un modello e naso da un altro modello.

LBP (Local Binary Pattern) è un **operatore basato su texture**. Nell'elaborazione delle immagini una texture è qualunque cosa sia in qualche modo composta da un **pattern ricorrente**. In generale gli operatori basati su texture mirano a scoprire quali sono i pattern ricorrenti che compongono una regione e distinguere quindi tra regioni differenti. Abbiamo una finestra scorrevole, ma in questo caso non usiamo un kernel come nelle immagini integrali o nella convoluzione. L'immagine stessa fornisce tutte le informazioni necessarie per poter effettuare l'elaborazione che vogliamo ottenere. L'operatore assegna ai pixel di un'immagine, in un intorno di dimensione 3x3, un valore binario (0 o 1). Sia p un pixel dell'intorno e p_c il pixel centrale, il valore binario è assegnato confrontando il valore del pixel p con quello del pixel p_c : se p ha un valore superiore o uguale a quello di p_c allora a p è assegnato il valore 1, altrimenti il valore 0. Questa operazione viene chiamata **thresholding**. L'immagine risultante sarà un'immagine in scala di grigio in cui i contrasti sono particolarmente accentuati. A partire dall'immagine prodotta da LBP, si può quindi creare un istogramma che rappresenta l'intensità dei pixel dell'immagine e tale istogramma può essere utilizzato per effettuare il confronto.



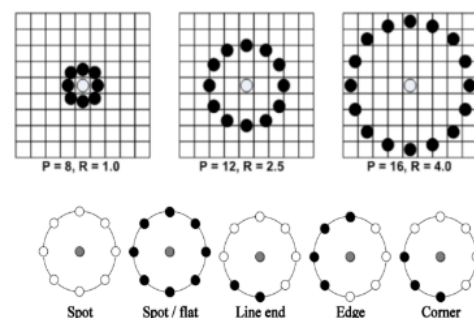
Ci sono alcune varianti di LBP. La prima variante prevede l'utilizzo di una **finestra più ampia**, in modo da gestire intervalli di diverso raggio. In pratica la finestra 3x3 ha raggio 1 (prendiamo pixel a distanza 1 da quello centrale). Il secondo parametro è il numero di vicini che consideriamo, quindi nel caso nel raggio di 1 il numero massimo di vicini che possiamo considerare è 8 (ma possiamo considerare meno di 8, ad esempio solo quelli nelle direzioni principali). Il prossimo passo è aumentare il **raggio**, e bisogna farlo in modo da avere un pixel centrale. Nel secondo esempio abbiamo una finestra 7x7 con il raggio di 2,5. In questo caso possiamo considerare più di 8 vicini. In ogni caso possiamo calcolare l'istogramma e utilizzarlo per il confronto.

I pattern binari più interessanti sono quelli **uniformi**, in quanto rappresentano le strutture locali più rilevanti (es. edge, spot, ecc). Un pattern è detto uniforme quando, considerato in modo circolare, contiene al massimo due transizioni 0-1 o 1-0. Ad esempio, i pattern 10000011, 11110000, 00000000 sono uniformi. Considerare solo i pattern uniformi permette di risparmiare memoria: infatti i pattern totali sono 2^P , mentre i pattern uniformi sono solo $P*(P-1)+2$.

Il feature vector associato a un'immagine è un istogramma calcolato come segue:

- L'immagine è partizionata in $k*k$ sottofinestre.
- Per ogni sottofinestra è costruito un istogramma in cui ciascun bin è associato a uno schema specifico; i bin sono in totale $P*(P-1)+3$: o $P*(P-1)$ bin per i pattern con 2 transizioni, 2 bin per i pattern con 0 transizioni, 1 bin per i pattern non uniformi.

Il feature vector si ottiene concatenando gli istogrammi calcolati per tutte le sottofinestre.



Esistono anche le **Rotation** ovvero in pratica ci sono i pattern, che sono tipo "bordo", "angolo", ecc...

Per riconoscerli senza tenere conto dell'orientazione, si possono fare delle rotazioni sulla stringa binaria e prendere la stringa che ha il maggior numero di 0 nella parte dei bit più significativi (a sinistra). Ottenuta questa stringa si fa il confronto con le stringhe di ogni tipo di pattern, che possono essere codificate in modo più efficiente visto che si considerano solo quelle particolari rotazioni delle stringhe.

3. Face Recognition 3D

Prima di tutto ricordiamo quali sono i principali problemi con il riconoscimento facciale. I problemi più tradizionali sono quelli relativi al **PIE** (Pose, Illumination and Expression). Anche l'invecchiamento è stato preso in considerazione in tempi più recenti (**A-PIE**). Recentemente sono stati presi in considerazione anche alcuni tipi di **travestimenti** legati a situazioni del mondo reale. Potremmo avere un travestimento volontario (una persona che si traveste per ingannare il sistema) o involontario che può essere causato da trucco (può cambiare completamente l'aspetto di una persona), chirurgia plastica (se abbiamo registrato il volto prima dell'intervento, potrebbe essere un po' più difficile riconoscere la stessa persona dopo la chirurgia plastica), occhiali, sciarpe, etc.. (che sono tipi di occlusione).

Quali sono le varianti che influenzano ancora il riconoscimento facciale in 3D? E quali possono essere considerati risolti con questo tipo di approccio?

Per quanto riguarda le **variazioni di posa**, mentre il riconoscimento 2D può essere influenzato da variazioni di rotazione, pitch e yaw, questo non è vero con il 3D perché possiamo testare con diverse pose sul modello 3D: può essere ruotato e la sua posa modificata in base al pitch e allo yaw in modo da poter provare a far assumere al modello 3D la stessa posa rappresentata nell'immagine 2D. Per un motivo simile anche i problemi di **illuminazione** possono essere considerati risolti perché, allo stesso modo, possiamo ruotare il modello e anche sintetizzare illuminazioni diverse. Mentre le **espressioni**

influenzano ancora il riconoscimento facciale in 3D. Se pensiamo soprattutto alle espressioni esagerate che sono quelle più difficili da catturare anche in 2D, creano comunque problemi in 3D perché possono causare una reale modifica rispetto ad un modello 3D neutro. Anche l'**invecchiamento** può influenzare la strategia 3D. Con l'avanzare dell'età, è possibile che alcune parti del viso si rilassino portando ad un modifica del volto che influisce anche sulla strategia 3D. Il **make up** non influisce il modello 3D perché solo considerando le relazioni geometriche, il volume 3D determinato dal viso non cambia. La **chirurgia plastica** può influenzare i modelli 3D a seconda del tipo di peso e dell'estensione dell'intervento. L'**occlusione** influenza il 3D come influenza il 2D.

Variation	2D		3D	
Pose	Affected	●	NOT Affected	●
Illumination	Affected	●	NOT Affected	●
Expression	Affected	●	Affected	●
Ageing	Affected	●	Affected	●
Makeup	Affected	●	NOT Affected	●
Plastic surgery	Affected	●	NOT Affected	●
Glasses, scarves, etc.	Affected	●	Affected	●

Possiamo riassumere i **pro e i contro delle strategie 3D** dicendo che:

- PRO: In 3D abbiamo molte più informazioni, i modelli costruiti sono molto più robusti ad un certo numero di distorsioni, c'è la possibilità di sintetizzare immagini 2D (approssimative) da pose ed espressioni 3D virtuali calcolate da un modello 3D.
- CONTRO: Costo dei dispositivi, costo computazionale delle procedure, possibile rischio che alcuni dispositivi di acquisizione possono presentare (ad esempio il laser scanner è pericoloso per gli occhi).
- Nel mezzo abbiamo tecniche che sono in grado di approssimare un modello 3D da immagini 2D. Quindi abbiamo una specie di adattamento inverso.

L'acquisizione delle immagini avviene in diversi modi:

- Camera Stereoscopica: ha due o più lenti con un frame d'immagine per ogni lente, permette di simulare la visione binoculare umana.
- Structured light scanner: viene proiettato sul soggetto un determinato pattern. L'informazione riguardante il modello 3D viene quindi catturata a seconda di come questo pattern viene deformato sul viso del soggetto.
- Laser scanner: Un unico raggio laser viene proiettato lungo la superficie (faccia). Il raggio viene deformato dalla struttura 3D della faccia e la deformazione dà la misura della profondità per ogni punto. La cattura deve essere ripetuta da diversi punti di vista per ottenere una modello 3D completo. Costo medio-alto, alta precisione, elevata robustezza all'illuminazione, 6-30 secondi per scansione, PERICOLOSO PER GLI OCCHI.

Ci sono problemi durante la scansione 3D?

Il rumore nell'immagine, polvere nell'aria o qualche tipo di inquinamento nell'aria possono causare errori occasionali nell'acquisizione, quindi sia la scansione 3D che le immagini 2.5D possono presentare entrambi il problema del rumore. Il rumore può creare:

- **Picchi:** Possiamo applicare dei filtri oppure possiamo provare alcuni interventi manuali correggendoli, notando che di solito non abbiamo un cambiamento improvviso ma il cambiamento è graduale.
- **Presenza di fori:** Ad esempio ci sono alcuni casi in cui alcuni punti non vengono catturati e quindi abbiamo immagini 2.5D e 3D con fori. In questo caso possiamo usare ad esempio livellamento gaussiano.

Possiamo anche avere il problema della **levigatura**, nel senso che, anche senza avere un picco, possiamo avere qualche imprecisione nella cattura per cui potrebbe essere necessario levigare la superficie.

E poi anche l'**allineamento**, perché ogni volta che abbiamo immagini diverse, prima di costruire il modello 3D i punti di riferimento principali devono essere perfettamente allineati, altrimenti introduciamo delle distorsioni nel modello 3D.

I possibili spazi in cui rappresentare una faccia sono:

- **2D:** La tipica rappresentazione è la **intensity image** dove ogni pixel dell'immagine è dato dal riflesso dell'illuminazione in quel particolare punto.
- **2.5D:** Una rappresentazione tra il 2D e il 3D. In questo caso si utilizzano le **range image** dove ogni pixel è dato dalla distanza tra quel punto e la sorgente di luce ed è solitamente espresso in scala di grigi. Non possiamo estrarre informazioni completamente 3D da una range image, perché possiamo solo dedurre le distanze dai punti al dispositivo in una determinata posizione. Se ad esempio acquisiamo più immagini a distanza a rotazione fissa, è possibile estrarre un modello 3D abbastanza affidabile.
- **3D:** Nella rappresentazione 3D, la rappresentazione più utilizzata è lo **shaded model**: abbiamo una struttura composta da punti e poligoni collegati nello spazio 3D e ogni poligono rappresenta una patch dell'oggetto molto piccola.

La costruzione di un modello 3D avviene mediante l'integrazione di diversi modelli 2.5D i quali sono catturati da diverse angolazioni. Per ogni immagine 2.5D viene generata una nuvola di punti 3D, con coordinate x e y equidistanti e coordinate z (profondità) derivate dal valore nell'immagine 2.5D. Alla fine del processo questi punti vengono **triangolarizzati** in modo tale da ottenere il modello finale.

Cosa si intende per shape from shading?

Questa rappresentazione sfrutta la relazione che c'è tra intensità e forma, in particolare calcola una forma 3D a partire dall'illuminazione del viso, assunta come **Lambertiana** (che riflette la luce in tutte le direzioni). Si può calcolare unendo immagini 2D e applicando tecniche statistiche PCA per derivare una rappresentazione dimensionalmente ridotta per forme nella stessa classe.

Cosa sono i morphable models?

L'espressione influisce ancora sul riconoscimento facciale in 3D. E' possibile sfruttare il cosiddetto **Morphable model**, che può essere in qualche modo distorto per riprodurre qualsiasi tipo di espressione. Il modello è chiamato Morphable perché possiamo **modificare alcune relazioni tra i poligoni** nel modello per **creare aspetti diversi della stessa faccia** o anche per **creare facce diverse**. Questo si basa su un set di dati di facce 3D e richiede che tra le facce ci sia una piena corrispondenza (allineamento). Quindi si parte da una o più immagini guida (tipicamente in posizioni differenti), si cambiano la forma e i colori di un modello generico di volto (morphable model) in accordo al contenuto delle immagini e infine si cerca di distorcere tale modello in maniera tale da arrivare ad una costruzione dell'immagine 3D.

Quali sono le features 3D?

Gli algoritmi di riconoscimento facciale 3D lavorano spesso sulla **curvatura locale e globale** del modello facciale. È possibile estrarre le informazioni riguardanti la forma di una faccia 3D analizzando la curvatura locale della superficie come ad esempio:

- **Linee di cresta:** selezione delle aree con la maggiore curvatura (dove c'è un cambio nell'orientazione).
- **Curvature locali:** rappresenta la curvatura locale con un colore chiaro o scuro in base alla curvatura.
- **Caratteristiche locali:** segmentazione del viso in regioni di interesse (ad esempio gli occhi, il naso, etc..).

Prima del riconoscimento è importante allineare il viso in maniera tale da avere una buona accuratezza. Questo può essere fatto:

- **COARSE:** Viene trovato un insieme di punti di riferimento e il viso viene allineato minimizzando la distanza tra i punti corrispondenti.
- **ICP (Iterative Closest Point):** Si trova un match iniziale tra le due superfici (calcolato in base alle curve locali, punti di riferimento, etc.), si calcola la distanza tra le due superfici, si calcola la trasformazione che minimizza tale distanza, la si applica e infine si reitera la procedura fin quando la distanza non sia sotto un certo threshold.

Due modi per effettuare il riconoscimento facciale 3D sono i seguenti:

- **Mappa normale:** Acquisiamo il modello 3D, lo proiettiamo in un'immagine 2D e da qui generiamo la mappa normale che non è altro che un'immagine RGB dove le 3 componenti corrispondono alle coordinate X, Y, Z della superficie dell'immagine. Utilizzando la mappa normale otteniamo una rappresentazione bidimensionale dell'informazione tridimensionale. La lettura e l'elaborazione di un'immagine 2D sono molto più veloci rispetto alla lettura e all'elaborazione di un modello 3D. Prese due mappe normali, possiamo effettuare la differenza tra le due (difference map) ed ogni pixel di tale mappa rappresenta la distanza angolare tra le due mappe.
- **Iso-Geodetic Stripes:** La **distanza Geodetica** è il percorso più breve tra due punti in uno spazio curvo, tali distanze sono **influenzate dai cambiamenti delle espressioni**. Le informazioni sono codificate in una rappresentazione sotto forma di grafo e il riconoscimento facciale è ridotto alla corrispondenza dei grafi. L'immagine del viso 3D è divisa in strisce facciali iso-geodetiche di uguale larghezza e distanza crescente a partire dalla punta del naso. Ogni stripe può essere rappresentata come un nodo di un grafo. Due nodi sono connessi tra di loro tramite **3D Weighted Walkthroughs (3DWW)**. La similarità tra i due modelli avviene confrontando la distanza tra i nodi appartenenti alle stesse stripes dei due grafi. Questo approccio è abbastanza resistente al cambio di espressione poiché la maggior parte dei punti in una stripe rimane nella stripe stessa anche se l'espressione cambia.

4. Face Spoofing

Uno Spoofing attack cerca di ingannare l'applicazione biometrica, utilizzando una copia o eseguendo un'imitazione del tratto biometrico utilizzato da quel sistema per autenticare legittimamente un utente.

Un sistema biometrico può essere attaccato in diversi punti: nei canali di trasmissione, nel feature extractor e nel matcher. È anche possibile attaccare il Database con template che non appartengono a persone iscritte.

Tutti questi tipi di attacchi lungo i canali, il database e così via, possono essere considerati come **attacchi indiretti**. Difendersi da questo tipo di attacchi è una questione di sicurezza informatica.

Un altro tipo di attacchi sono gli **attacchi diretti**. In questo caso, lo scopo finale del sistema anti-spoofing non è riconoscere la persona, ma piuttosto rilevare l'attacco in modo da poter procedere con una contromisura.

Possiamo fare una classificazione degli attacchi di spoofing: attacco di **spoofing 2D** e attacco di **spoofing 3D**.

Negli **Spoof 2D**, l'attacco viene effettuato utilizzando una superficie 2D come una foto o un video che viene riprodotto una volta che abbiamo in qualche modo registrato la persona attaccata. Quest'ultimo tipo di attacco è anche chiamato **"Replay Attack"**.

Negli **Spoof 3D**, l'attacco richiede che non ci sia una superficie piana ma una superficie tridimensionale e in generale vengono utilizzate maschere per questo tipo di attacco (**Mask Attack**).

Per ogni tipo di attacco ci sono diverse tecniche anti-spoofing che si basano principalmente sul **liveness detection**. In generale, quello che cerchiamo di fare, oltre a studiare il pattern di riflessione della superficie e la microtexture, è "sfidare" l'utente al fine di rilevare il tipo di reazione della faccia che abbiamo di fronte.

Abbiamo tecniche anti-spoofing che funzionano a **livello di sensore** (basato su hardware): in pratica sfruttano le proprietà di un corpo vivente, compreso il tipo di riflettanza che sta producendo e segnali involontari (ad esempio micro movimenti degli occhi) che sono completamente assenti in una foto o in una maschera. Un esempio di tecnica anti-spoofing che sfrutta questo tipo di movimenti è **Eye Blink**. In effetti, il battito delle palpebre è un movimento involontario naturale che può essere rilevato per distinguere una persona reale da una foto.

D'altra parte, possiamo utilizzare una strategia che fa uso di reazioni volontarie come la “**Challenge-Response**”. In questo caso possiamo chiedere ad una persona che sta presentando un probe di fare un certo tipo di espressione o movimento. Se ciò che ci aspettavamo non accade sulla faccia mostrata al sistema, possiamo concludere che si tratta di un attacco.

Altri tipi di tecniche anti-spoofing vengono eseguite a livello di **Feature Extractor** (basate su software) e possono essere statiche o dinamiche. Un esempio di studio statico è lo studio della microtexture dell'immagine acquisita. Con la dinamica studiamo il tipo di caratteristiche dinamiche che un certo movimento può produrre quando viene eseguito naturalmente.

Possiamo avere anche una **Score level fusion** dove possiamo fondere anti-spoofing e riconoscimento in un unico modulo oppure effettuare anticipatamente l'anti-spoofing e procedere al riconoscimento solo se l'anti-spoofing restituisce una risposta genuina.

Prima di esaminare nel dettaglio i vari tipi sistemi anti-spoofing, è bene introdurre la differenza tra camouflage e spoofing: Nel caso del **camouflage** l'attacco viene effettuato presentando al sistema un tratto biometrico per ingannarlo fingendo di non essere se stesso. Quindi in quel caso non vogliamo essere riconosciuti. Introduciamo infatti elementi estranei sul viso il processo di rilevamento potrebbe fallire. Nello **spoofing** invece si utilizza una copia o si esegue un'imitazione del tratto biometrico utilizzato dal sistema per autenticare legittimamente un utente.

Uno degli approcci più intuitivi al **2D Print Attack** è **Liveness Detection**. In pratica, la differenza essenziale tra il viso dal vivo e la fotografia è che un viso dal vivo è un oggetto completamente tridimensionale mentre una fotografia potrebbe essere considerata come una struttura bidimensionale.

Possiamo utilizzare il movimento per ricavare informazioni di profondità per distinguere una persona dal vivo da una foto fissa. È difficile stimare le informazioni di profondità quando la testa è ferma (dobbiamo chiedere alla persona di effettuare un movimento) e la stima è molto sensibile al rumore e all'illuminazione (un ambiente buio può aiutare l'attaccante a nascondere qualche cambiamento di caratteristica rispetto a un volto reale).

È possibile calcolare l'**Optical Flow** (tecnica che cerca di estrarre il vettore di movimento confrontando la posizione di ogni pixel in un frame e in quello successivo) sul video in ingresso per ottenere l'informazione del movimento del viso per la liveness detection.

Un possibile **approccio multimodale** fonde volto-voce contro lo spoofing sfruttando il movimento delle labbra mentre si parla.

Un'altro approccio sfrutta l'**analisi del battito di ciglia** essendo movimento fisiologico che non si può assolutamente evitare, ovvero l'Eye Blink.

Un'altra possibilità si basa sul **Micro-texture analysis**. Ciò è particolarmente efficiente quando abbiamo 2D Print attacks o Photo attacks perché qualsiasi tipo di carta fotografica o carta da stampa ha un tipo di microtexture che, pur non essendo visibile agli occhi, può essere rilevata utilizzando le texture features. I volti umani e le stampe riflettono la luce in modi diversi perché un volto umano è un oggetto 3D complesso non rigido (quindi riflette la luce in direzioni diverse) mentre una fotografia è un oggetto rigido planare (diverse sfumature e riflessi speculari). Le proprietà delle facce e stampe, ad es. pigmenti, sono diverse anche perché i pigmenti naturali sono diversi dai pigmenti dell'inchiostro. Questi ultimi contengono alcuni componenti metallici quindi questo influenza il modo in cui la luce viene riflessa. Il lavoro sfrutta **modelli binari locali multiscala (LBP)**. La strategia utilizzata nei modelli binari locali multiscala è più o meno la stessa ma molto più semplice da adottare rispetto al bunch of wavelet con frequenze diverse. I modelli binari locali multiscala vengono calcolati utilizzando finestre di dimensioni diverse. Come ulteriore vantaggio, le stesse caratteristiche di texture utilizzate per il rilevamento dello spoofing possono essere utilizzate anche per il riconoscimento facciale. I vettori nello spazio delle caratteristiche vengono quindi fornite ad un **classificatore SVM** che determina se le micro-texture caratterizzano una persona viva o un'immagine falsa.

Un altro approccio importante è l'approccio **Captured-Recaptured**. Dobbiamo considerare che tutte le distorsioni che sono presenti quando catturiamo un'immagine influenzano l'immagine del viso quando passa attraverso il sistema della fotocamera. Tali distorsioni vengono applicate due volte quando scattiamo la foto di una persona e poi quando la stampiamo. In pratica, quando utilizziamo una foto scattata da una foto, come può succedere quando si utilizza una foto scattata su internet, abbiamo una qualità dell'immagine molto inferiore rispetto alla cattura di un'immagine del viso reale. I volti umani e le stampe riflettono la luce in modi diversi perché un volto umano è un oggetto 3D complesso non rigido (quindi riflette la luce in direzioni diverse) mentre una fotografia è un oggetto rigido piano (con diverse sfumature e riflessi speculari).

Un altro approccio è la **Gaze Stability**. Esiste un algoritmo che si basa sul presupposto che la coordinazione spaziale e temporale dei movimenti dell'occhio, della testa e (possibilmente) della mano coinvolti nel compito di seguire uno stimolo visivo è significativamente diversa quando si fa un tentativo vero e proprio rispetto a certi tipi di tentativi di spoofing. In una strategia **challenge-response**, quando il sistema chiede al soggetto genuino di ruotare la testa, il tipo di coordinazione temporale tra il movimento degli occhi, il movimento della testa (se eventualmente coinvolti) è diverso quando da quando la persona sta guardando lo schermo attraverso i fori di una maschera. Il compito richiede di fissare un semplice bersaglio che appare su uno schermo davanti all'utente. Nel caso di un attacco di spoofing fotografico, sono necessari movimenti della mano guidati visivamente per orientare la foto in modo che punti nella direzione corretta verso la sfida. Quindi la coordinazione temporale tra il movimento degli occhi e la rotazione della testa è in qualche modo influenzata da questo movimento della mano. L'**Optical Flow Correlation** tiene conto del movimento della testa dell'utente che cerca di autenticarsi e del movimento dello sfondo della scena, perché se si muovono insieme significa che c'è un attacco di spoofing, perché viso e sfondo si muovono insieme mentre non dovrebbero muoversi insieme. Optical Flow non funziona in due casi: nel caso delle maschere, e anche se abbiamo un'immagine del viso piegato sul viso dell'aggressore.

Un altro possibile approccio consiste nell'utilizzare l'**Image Distortion Analysis**. Si basa sul tipo di riflessi prodotti da una superficie di carta stampata o da uno schermo LCD (ad esempio lo schermo di uno smartphone) durante l'acquisizione. Esiste una sorta di analisi composta che viene effettuata in base a questi diversi elementi e quindi è possibile concatenare le caratteristiche estratte da ciascuno di questi fattori di distorsione dell'immagine e addestrare il classificatore.

Un altro modo per utilizzare efficacemente lo studio delle **microtexture** è legato ai **Replay Video Attacks**. Quando eseguiamo un Replay Video Attack, mostriamo un video su un altro schermo. In questo caso esiste un tipo speciale di pattern chiamato **moiré pattern aliasing** che è causato dalla sovrapposizione di due pattern di pixel: uno dal video originale e l'altro dallo schermo su cui viene mostrato il video. È possibile rilevare questo modello speciale utilizzando Multi-scale LBP e DSFIT (questo cerca di rilevare le caratteristiche con la più alta energia nell'immagine). In pratica, ogni volta che è presente una sorta di moiré pattern, possiamo concludere che siamo in presenza di un video replay attack.

Nel caso **maschere facciali 3D** non si possono sfruttare i metodi di anti-spoofing che solitamente vengono messi in atto nel caso immagini 2D. Nel caso di 3D Mask Attack utilizziamo le caratteristiche di riflettanza di diversi materiali utilizzati nella creazione delle maschere che sono diverse dalle **caratteristiche di riflettanza** della pelle. Abbiamo quindi una riflettanza che avviene lungo diverse lunghezze d'onda. Possiamo scegliere le lunghezze d'onda specifiche dopo aver ispezionato le curve di albedo della pelle del viso e i materiali della maschera con distanze variabili in base alla distanza dal sensore.

Un altro approccio che si basa sul tipo di comportamento fisiologico del corpo umano: la **fotopletiografia**. In pratica, con il volto reale, è possibile, con avanzate acquisizioni ed elaborazioni video, catturare micro variazioni volumetriche che vengono prodotte dal flusso sanguigno sotto i tessuti sottocutanei e questo può permettere la distinzione tra una maschera e un volto reale.

Per valutare un sistema anti-spoofing, una possibilità è sfruttare lo **Spoof False Acceptance Rate (SFAR)**: il numero degli attacchi di spoof che vengono falsamente accettati. Utilizziamo lo SFAR invece del FAR perché quest'ultimo è dovuto ai cosiddetti **zero effort attacks** o attacchi zero sforzo (una persona rivendica un'identità senza intraprendere alcuna azione particolare per assomigliare all'utente autentico). In uno Spoof Scenario invece dobbiamo considerare anche i tentativi che sono compiuti per cercare di riprodurre l'aspetto della persona attaccata. Si utilizza anche il False Rejection Rate. In questo caso il FRR non si riferisce al mancato riconoscimento di un campione, ma all'errata classificazione del campione autentico come un campione contraffatto. In alcuni casi possiamo **fondere il riconoscimento e l'anti-spoofing** in un unico sistema con un'unica risposta Accetta-Rifiuta. Quindi possiamo combinare un Classificatore Biometrico con un Classificatore Binario, perché un Classificatore Anti Spoofing non è che un algoritmo di sistema in grado di decidere se un probe è genuino o meno.

5. Evaluation of Face Recognition

Anche in questo caso ci sono alcuni problemi aperti con Face Recognition. Facciamo un riassunto:

- **Illuminazione:** le condizioni di illuminazione possono cambiare in modo significativo durante il giorno, in giorni diversi, all'interno o all'esterno. La luce diretta sul viso produce ombre e zone iper illuminate. A causa di tali variazioni il feature vector associato ad un soggetto in condizioni di illuminazione diverse può risultare più vicino a quello di un soggetto diverso con illuminazione simile, che a quello dello stesso soggetto con illuminazione diversa. Possiamo affrontare questo problema scegliendo un acceptance threshold ragionevole. Si noti che nel caso di identificazione open set, qualunque sia il threshold, abbiamo una classifica e quindi il rango dell'immagine appartenente alla persona sbagliata può essere superiore al rango dell'immagine appartenente alla persona giusta se sono presenti ombre. Ci sono tre tipi di algoritmi volti ad affrontare le variazioni di illuminazione in FR:
 - **Shape from Shading:** Partendo dai livelli di grigio in una o più immagini estraggono la forma del viso 3D e la proiettano anche in 2D per ottenere una migliore rappresentazione del volto candidato da abbinare alla gallery.
 - **Metodi basati sulla rappresentazione:** Utilizzano classificatori che per loro natura intrinseca sono robusti rispetto alle variazioni di illuminazione. LBP non può essere considerato tra quei metodi basati sulla rappresentazione perché nella regione dell'ombra, i valori restituiti da LBP saranno completamente inaffidabili.
 - **Metodi generativi:** Partendo da un modello di volto 3D generano un ampio set di immagini con il maggior numero possibile di variazioni di illuminazione, che vengono utilizzate successivamente per il training del soggetto.
- **Posa:** Nella maggior parte delle applicazioni la posa del soggetto durante il test può essere diversa da quella durante il training. I sistemi che affrontano i problemi di posa possono essere suddivisi in due categorie:
 - **Sistema multivista:** le immagini della galleria raffigurano il soggetto in una serie di pose possibili.
 - **Sistemi di correzione della posa:** Partendo dall'immagine di prova il sistema deduce la posa del soggetto, cercando di correggerla.
- **Occlusione:** Gli algoritmi di riconoscimento devono essere resistenti alle occlusioni. Quando abbiamo una grande occlusione, non c'è modo di colmare in modo affidabile il vuoto perché in ogni caso stiamo riproducendo informazioni inesistenti e qualunque sia la strategia che usiamo, potremmo introdurre una fonte di errore. La maggior parte degli approcci si basa sull'elaborazione localizzata: In pratica, in questi casi, invece di cercare di correggere l'occlusione, gli approcci più diffusi si occupano prima di rilevare l'occlusione: una volta rilevata l'occlusione, la si usa come una regione "non importa" e si prova ad effettuare il confronto usando solo il resto dell'immagine del volto.
- **Variazioni di tempo ed età:** le **variazioni di tempo** (anche di una sola settimana) tra due immagini dello stesso soggetto, possono ridurre le prestazioni di un sistema di riconoscimento. Il termogramma è più robusto rispetto a tali variazioni. Le **variazioni di età** sono ancora più critiche. Un sistema resistente alle variazioni di età è utile in applicazioni come l'identificazione di persone ricercate o scomparse. In questo caso cerchiamo di imitare il processo di invecchiamento. La base sono gli studi etnografici e lo studio di come si sviluppano alcune regioni somatiche e regioni anatomiche rilevanti del viso.
- Altre variazioni possono essere il **trucco**, la **chirurgia plastica**, il **cambio di sesso** e il **cambio di peso**.

Il **confronto tra sistemi** non solo è necessario per essere in grado di valutare l'accuratezza delle prestazioni di un determinato sistema, ma è anche importante poter confrontare le prestazioni di sistemi diversi. Può essere fatto in base al tipo di applicazioni che abbiamo, perché possiamo avere ad esempio misure diverse quando è prevista la verifica o quando è prevista l'identificazione. Bisogna tener conto che misure come FAR, FRR, CMS, ... non sono sufficienti per una valutazione completa di un algoritmo in quanto troppo slegate dal reale contesto operativo.

Dobbiamo anche considerare:

- Numero di dataset utilizzati: è necessario per valutare la generalizzabilità dei metodi che abbiamo proposto.
- Dimensione dell'immagine: immagini grandi e di alta qualità forniscono maggiori informazioni; ma maggiore è la risoluzione, maggiore è il rumore che può influenzare il sensore.
- Dimensioni del probe set e del gallery set: dobbiamo lavorare con dimensioni ragionevoli per garantire la generalizzabilità dei nostri metodi.

- Numero e livello delle variazioni tollerate: dipendono dal tipo di applicazione che stiamo progettando. Perché se possiamo presumere ad esempio che la persona presenterà sempre un volto in posa normale con un'espressione neutra e una buona illuminazione, forse possiamo sbarazzarci di tecniche computazionalmente più impegnative per rimanere con qualcosa di più semplice.

Per valutare un sistema sono stati ideati dei protocolli. Esempi di tali protocolli sono le seguenti:

1. **FERET** (Facial Recognition Technology): Il programma consisteva in tre elementi principali: sponsorizzazione della ricerca (per raccogliere campioni), creazione del database e l'esecuzione di valutazioni. L'obiettivo era quello di sviluppare algoritmi di riconoscimento facciale. Prima della creazione di FERET, solo alcuni degli algoritmi di riconoscimento facciale finora creati utilizzavano un database comune. Di conseguenza, non c'era modo di effettuare confronti affidabili tra vari algoritmi. Il database FERET ha reso possibile quindi sviluppare algoritmi su un database comune e di riportare i risultati in letteratura utilizzando questo database.
2. **FRVT** (Face Recognition Vendor Test): Una serie di valutazioni su larga scala per sistemi di riconoscimento facciale. So stati fatti vari test in cui è stato dimostrato che: un compito difficile per un sistema di riconoscimento del viso 2D è quello di riconoscere i volti che non sono rappresentati in posa frontale e le prestazioni degli algoritmi 2D sono drasticamente ridotte quando si utilizzano immagini dello stesso soggetto ma scattate con illuminazioni molto diverse,
3. **FRGC** (Face Recognition Grand Challenge): E' una sfida tra diversi gruppi di ricercatori. La sfida è progettata per ottenere un aumento di prestazioni degli algoritmi di riconoscimento 2D e 3D. L'obiettivo era ottenere un tasso di verifica del 98%. Il dataset è composto da immagini catturate con condizioni interne controllate, condizioni interne/esterne non controllate e modelli 3D.

Per testare massivamente il sistema, è possibile calcolare **All_{Probe} Against All_{Gallery} Matrix** dove tutti contro tutti in questo caso significa tutti i template nel probe set contro tutti i template nella galleria. L'idea è quella di effettuare vari test in cui ogni probe rivendica un'identità diversa. Per ogni coppia probe-gallery, è possibile calcolare preventivamente una matrice di distanza ALL PROBE VS ALL GALLERY, memorizzando tutte le distanze tra le coppie di modelli (uno del probe rispetto a uno della galleria). Ogni riga della matrice corrisponde ad un'operazione di riconoscimento su un probe in ingresso ed ogni partizione probe-gallery produce una matrice di distanza diversa. La matrice delle distanze può essere utilizzata per la valutazione delle prestazioni di tutti i tipi di applicazioni come la Verifica, l'Open-Set Identification e la Closed-Set Identification:

- **Verifica:**

Solo i modelli in gallery appartenenti all'identità dichiarata vengono confrontati con il probe. Non è importante chi c'è nella gallery, ma l'identità rivendicata. Ogni riga è etichettata con l'identità della ground truth probe (ad esempio ID A) e, per la verifica, con l'identità dichiarata (ad esempio Claim A). La valutazione delle prestazioni può essere ottenuta separando chiaramente probe e gallery (diversi sottoinsiemi di modelli) e veri e propri impostori (i modelli che svolgono il ruolo di impostori sono associati alla rivendicazione di un'identità diversa).

EX.

Example: identities A, B, C, D, E, F → in the dataset
 identities A, B, C, D → in the gallery with a single instance
 identities E, F → play the role of impostors in all cases
 $d()$ = distance from the probe t = acceptance threshold

	Probes	A1	B1	C1	D1
ID A – Claim A	P1	1	4	2	3
ID D – Claim C	P2	4	1	3	2
ID E – Claim D	P3	4	2	1	3
ID C – Claim C	P4
ID F – Claim B	P5

Example of distance matrix (only order of values is shown)
 Each row is a single verification operation

La probe P1 con identità A claim A (reclama l'identità A). Questa è una GENUINE CLAIM (e lo sappiamo grazie al ground truth) e deve essere accettata dal sistema. Se ho una distanza $d(A1) \leq t$ ($d(A1) = 1$ è la

distanza) allora avrò una Genuine Acceptance, se invece ho $d(A1) > t$ allora in QUESTO caso avrò una False Rejection (poiché l'identità è corretta ma ha sbagliato il sistema). In questo caso per ottenere una GA devo settare un valore di threshold $t=1$. P2 appartiene all'identità D ma l'identity claim è C. D è in galleria ma reclama un'altra identità. Ovviamente se $d(C1) > t$ abbiamo una Genuine Rejection ed il sistema ha captato efficacemente l'impostore, altrimenti abbiamo una False Acceptance.

La distanza con la claimed identity (C1) è 3. Se si pone $t=2$ allora ho una GR, in questo modo $C1 > t$.

P1 GENUINE: $d(A1) \leq t \rightarrow GA++$, $d(A1) > t \rightarrow FR++$.

P2 IMPOSTORE (il soggetto è in galleria ma rivendica un'altra identità): $d(C1) \leq t \rightarrow FA++$, $d(C1) > t \rightarrow GR++$.

P3 IMPOSTORE (soggetto non in galleria): $d(D1) \leq t \rightarrow FA++$, $d(D1) > t \rightarrow GR++$.

Incrementando il numero di samples per soggetto si va a decrementare il FRR (perché abbiamo più possibilità di riconoscere le genuine identity) ma aumenta il FAR (poiché aumenta la possibilità che un False probe sia simile ad una Genuine Identity).

È necessario effettuare un numero sufficiente di valutazioni per calcolare un risultato medio attendibile: ogni volta probe set e gallery set sono scelti in modo diverso, e c'è una possibile diversa distribuzione di probe autentici e impostori.

- **Identificazione Open-set:**

In questo caso il probe da identificare potrebbe non essere nel sistema. Ogni riga è etichettata solo con l'identità ground truth (ad es. A) del probe (nessuna rivendicazione, infatti non c'è Claim A). I probe autentici sono quelli appartenenti alle identità nella galleria, mentre i probe impostori sono quelli associati alle identità non nella galleria.

EX.

Example: identities A, B, C, D, E, F
 identities A, B, C, D
 identities E, F
 $d()$ = distance from the probe

in the **dataset**
 in the **gallery** with a single instance
 play the role of **impostors**
 t = acceptance threshold

Example of distance matrix (only order of values is shown)

Each row is a single identification operation

	Probes	A1	B1	C1	D1
<u>A</u>	P1	1	4	2	3
<u>D</u>	P2	4	1	3	2
<u>E - Impostor</u>	P3	4	2	1	3
<u>C</u>	P4
<u>F - Impostor</u>	P5

Il valore di threshold t è la distanza dalla probe che possiamo tollerare per accettare eventuali probe. Per ogni P_n viene ordinata la lista delle distanze, come ad esempio per $P1 = d(A1), d(C1), d(D1), d(B1)$. La probe P1 con identità A claim A (reclama l'identità A). Questa è una GENUINE CLAIM e deve essere accettata dal sistema. Se $d(A1) < t$ allora incremento il $DI(1,t)$, ovvero incremento il Detect and Identification outcomes at rank 1 per il threshold t , mentre se $d(A1) > t$ avrò un False Rejection.

Anche in questo caso è importante effettuare un numero sufficiente di valutazioni, cambiando gallery set e probe set e distribuzione di probe autentici e impostori.

- **Identificazione Closed-set:**

La probe da identificare appartiene sempre ad un soggetto inserito nella galleria (infatti non c'è E - impostor). Tutti i template della galleria vengono confrontati con il probe. Ogni riga è etichettata con la vera identità del probe. Nessun impostore compare negli esperimenti (tutte le identità sono nella gallery). La valutazione delle prestazioni può essere ottenuta separando chiaramente probe e gallery. Si noti che non c'è un threshold t di accettazione.

Pros: easy to program, in practice computes a kind of «average» over all possible specific distributions of genuine/impostor attempts.
Pros: The number of impostors is always much higher than genuine, therefore it is possible to overstress the system to assess situations with many impostor attempts.

Cons: the time requested to compute the full matrix may become very high.

Cons: does not to analyze specific distributions of genuine/impostor attempts, that may achieve peculiar results.

EX.

Example: identities A, B, C, D, E, F in the **dataset**
 identities A, B, C, D, E, F in the **gallery** with a single instance
 $d() = \text{distance form the probe}$

Tutte le identities sono nella gallery

Probes	A1	B1	C1	D1	E1	F1	
A	P1	1	4	2	3	6	5
D	P2	6	1	4	2	3	5
E	P3	5	2	1	3	4	6
C	P4
F	P5

● Example of distance matrix (only order of values is shown)

Each row is a single identification operation

Per ogni P_n viene ordinata la lista delle distanze, come ad esempio per $P1 = d(A1), d(C1), d(D1), d(B1), d(F1), d(E1)$. Per $P1$ il template corretto per A è A1 e questo risultato contribuisce al CMS (Cumulative Match Score) al rank 1 ed è anche detto Recognition Rate (RR). Per $P2$ il template corretto per D è D1 e questo risultato contribuisce al CMS (Cumulative Match Score) al rank 2.

È possibile calcolare statistiche più comprensive utilizzando una matrice di distanza completa nel senso che non solo calcoliamo solo l'intera matrice ma consideriamo anche ogni riga come un diverso set di esperimenti. In pratica ogni template viene comparato con ogni template e la matrice delle distanze viene calcolata una volta per tutte per poi andarla ad applicare sull'algoritmo. Ad ogni turno, un template gioca sia il ruolo del template nel probe set che nel gallery set. Chiaramente non si deve considerare la diagonale cioè i casi in cui si compara un template con se stesso. Una possibile strategia di utilizzo della matrice di distanza **ALL vs. ALL** è la seguente:

- Ogni probe (riga) è considerata a sua volta genuina o impostore;
- I risultati vengono accumulati per ottenere le statistiche finali;
- Ogni riga contribuisce eventualmente più volte in base al numero di esperimenti che può eventualmente rappresentare (gli esempi sono i modelli passati al sistema per addestrarlo);
- Per semplicità si può ipotizzare lo stesso numero di campioni per soggetto (nel nostro caso 3 per ogni set).

Alcune definizioni:

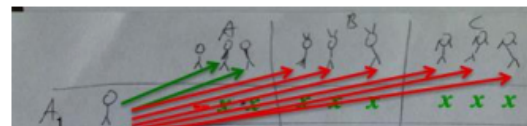
- M = distance matrix.
- N = numero di subject (insieme di uomini come ad esempio A con i tre personaggi).
- S = numero di templates per subject (il numero di omini per subject).
- $|G|$ = numero totale di samples ($|G| = S \times N$).
- i = indice di riga (probe templates) e $\text{label}(i)$ è l'identità associata.
- j = indice di colonna (gallery template) e $\text{label}(j)$ è l'identità associata.
- $\text{label}()$ = ground truth.

Vediamo ora come funziona per ogni tipologia di riconoscimento:

● **Verifica Single Template:**

Ogni riga è un insieme di $|G| - 1$ operazioni. Ogni riga contiene $S-1$ tentativi genuine e $(N-1) \times S$ tentativi impostor. Il numero totale di tentativi genuine è $TG = |G| \times (S-1)$ mentre il numero totale di tentativi impostor è $TI = |G| \times (N-1) \times S$. L'algoritmo funziona in questa maniera:

Per ogni threshold t e per ogni cella (tranne quelle nella diagonale principale), se la distanza è inferiore al threshold allora se l'identità di i è uguale all'identità di j (ricordiamo che i rivendica j) allora abbiamo una GA, altrimenti abbiamo una FA. Se la distanza è maggiore del threshold allora



```

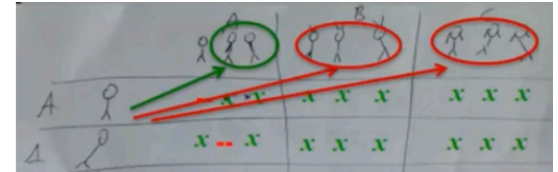
for each threshold t
  for each cell  $M_{ij}$  with  $i \neq j$ 
    if  $M_{ij} \leq t$  then
      if  $\text{label}(i) = \text{label}(j)$  then GA++
      else FA++
    else if  $\text{label}(i) = \text{label}(j)$  then FR++
    else GR++
GAR(t) = GA / TG; FAR(t) = FA / TI;
FRR(t) = FR / TG; GRR(t) = GR / TI
  
```

se l'identità di i è uguale all'identità di j allora abbiamo un FR, altrimenti abbiamo un GR. Ora che abbiamo calcolato tutti i valori (quindi abbiamo la matrice di confusione) possiamo calcolare i tassi.

- **Verifica Multiple Template:**

Questa volta non confronteremo il probe con ogni template ma con un gruppo di template (una strategia comune, ad esempio, è quella di prendere il miglior risultato data una certa identità dichiarata). Quindi in questo caso ogni riga è un insieme di N operazioni dove uno è un tentativo genuino e gli altri impostori. Quindi il numero totale di tentativi genuini è dato da $|G|$, mentre il numero totale di tentativi impostori è dato da $|G| \cdot (N-1)$. L'algoritmo funziona in questa maniera:

Per ogni soglia t , ogni riga i (ogni probe) e ogni gruppo di template appartenenti allo stesso soggetto (escludendo eventualmente il caso in cui la sonda e il template che stiamo confrontando sono lo stesso oggetto), si seleziona il template nel gruppo con la distanza inferiore con il probe e si procede come prima.



```

for each threshold t
  for each row i
    for each group  $M_{label}$  of cells  $M_{i,j}$  with same label(j)
      excluding  $M_{i,i}$ 
      select  $diff = \min(M_{label})$ 
      if  $diff \leq t$  then
        if  $label(i) = label(M_{label})$  then  $GA++$ 
        else  $FA++$ 
      else if  $label(i) = label(M_{label})$  then  $FR++$ 
      else  $GR++$ 
     $GAR(t) = GA/TG$ ;  $FAR(t) = FA/TI$ ;
     $FRR(t) = FR/TG$ ;  $GRR(t) = GR/TI$ 

```

- **Identificazione Open Set Multiple Template:**

In questo caso, considerando i singoli template presenti nella gallery e considerando che l'operazione restituisce un elenco ordinato, dovremmo considerare tutte le possibili combinazioni per i template. Ogni riga è un insieme di 2 operazioni di identificazione (ogni riga (probe) può essere considerata come un'identità registrata o meno). Quindi ogni riga contiene 1 tentativo genuino (soggetto nella galleria) e 1 tentativo impostore (soggetto non nella galleria). In questo caso abbiamo solo due esperimenti perché non abbiamo una rivendicazione da parte dell'utente, anche se l'utente è nella galleria o meno. Quindi il numero di Total Genuine Attempts è dato dal numero totale di campioni: $TG = |G|$. Anche il numero di Total Impostor Attempts è dato dal numero totale di campioni, quindi: $TI = |G|$. L'algoritmo funziona in questo modo:

Per ogni threshold e per ogni riga (probe) consideriamo tutti i template ordinandoli per distanza crescente escludendo l'elemento identico nell'insieme. Prendiamo la prima posizione nell'insieme $L_{i,1}$ (che il sistema assume essere l'identità del probe): se la distanza di questo elemento è inferiore alla threshold (quindi se può essere una potenziale accettazione) possiamo considerare in parallelo casi diversi: se l'etichetta del probe e l'etichetta del primo elemento della lista sono le stesse, possiamo aumentare il DI al rango 1 e, contemporaneamente, possiamo anche aumentare il numero di FA perché c'è il caso in cui il probe non appartenga a una persona nella galleria quindi troviamo il primo template $L_{i,k}$ (se k esiste) dove il probe è diverso dal template $L_{i,k}$ (e la distanza tra loro è minore del threshold). Se k non esiste allora la prima posizione in cui l'etichetta del probe è diversa da quella del template, avrebbe una distanza più alta del threshold quindi non è accettata (quindi aumento il GR). Consideriamo ora il caso in cui l'identità del probe non è l'identità del primo template restituito nella lista e la loro distanza è comunque inferiore al threshold: se k esiste allora aumentiamo il DI al rango k e in parallelo aumentiamo anche il numero di FA (avremo in prima posizione un template la cui etichetta è diversa dall'etichetta del probe). L'ultimo caso è quando $L_{i,1}$ è maggiore del threshold quindi il probe non è nella galleria: in questo caso aumentiamo semplicemente il GR.

Alla fine, possiamo calcolare il DIR semplicemente dividendo il DI per il numero

```

for each threshold t
  for each row i
     $\{L_{i,m} \mid m=1 \dots |G|-1\} =$ 
     $\{M_{i,j} \mid j=1, \dots, |G|\} \setminus M_{i,i}$  ordered by increasing value (the identical element is excluded)
    if  $L_{i,1} \leq t$  then (potential accept)
      if  $label(i) = label(L_{i,1})$  then  $DI(t, 1)++$  (genuine case detected+identified)
      (parallel impostor case: jump the templates belonging to label(i) since i not in G)
      find the first  $L_{i,k}$  such that  $label(L_{i,k}) \neq label(i)$  AND  $L_{i,k} \leq t$ 
      if this  $k$  exists, then  $FA++$  (the first template != label(i) has a distance  $\leq t$ )
      else  $GR++$  (impostor is correctly not detected)
    else find the first  $L_{i,k}$  such that (if genuine yet not the first, look for higher ranks)
       $label(i) = label(L_{i,k})$  AND  $L_{i,k} \leq t$ 
      if this  $k$  exists, then  $DI(t, k)++$  (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  $DI(t, k) \neq 0$ 
     $DIR(t, k) = DI(t, k)/TG + DIR(t, k-1)$  (we have to compute rates)

```


di tentativi genuini: $DIR(t, 1) = DI/TG$. False Rejection Rate è solo 1-DIR al rango 1: $FRR(t) = 1 - DIR(t, 1)$, False Acceptance Rate: $FAR(t) = FA/TI$ e Genuine Rejection Rate: $GRR(t) = GR/TI$. Possiamo anche calcolare il DIR a un rango superiore k , dividendo il numero di rilevamento e identificazione per il numero di tentativi genuini al rango k con la somma dei tentativi genuini e DIR calcolato fino a k (escluso).

- **Identification Closed Set Multiple Template:**

Nel caso di Identification Closed Set il calcolo è più semplice perché non abbiamo il threshold, assumiamo che ogni probe appartenga a un soggetto nella galleria e prendiamo in considerazione solo il rango in cui viene restituita l'identità corretta per valutare il comportamento complessivo del sistema. Anche se il sistema non è in grado di recuperare l'identità corretta nel primo caso, forse è utile sapere se il sistema è in grado di recuperare l'identità corretta entro il k -esimo posto. Quindi ogni riga è un'operazione, ogni riga contiene 1 tentativo genuino e i tentativi totali sono $TA = |G|$. L'algoritmo funziona in questo modo:

Per ogni riga si considerano tutti i template della riga ordinandoli per distanza crescente escludendo l'elemento identico nell'insieme. Dobbiamo trovare il primo $L_{i,k}$ tale che l'identità del probe sia uguale all'identità $L_{i,k}$ e aumentiamo $CMS(k)$ di uno. Possiamo calcolare $CMS(k)$ semplicemente dividendo se stesso per la somma dei tentativi totali e il CMS al rango precedente.

```
for each row i
  {Li,m | m=1 ... |G|-1} =
  {Mi,j | j=1, ... |G|} \ Mi,i ordered by increasing value
  find the first Li,k such that
    label(i)=label(Li,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)
```

All Against All non è adatto a situazioni specifiche in cui abbiamo partizioni precise del set di dati in due sessioni diverse. Quando c'è una suddivisione precisa dei template in sessioni diverse non sovrapposte in momenti diversi, è più corretto creare probe set e gallery set utilizzando template di sessioni diverse. Questo perché accade che i campioni catturati nella stessa sessione siano solitamente più simili di quelli di sessioni diverse, a causa delle variazioni che si verificano nel tempo (ad es. variazioni del volto), condizioni ambientali (ad es. illuminazione), prestazioni (ad es. sporco depositato su un sensore di impronte digitali). Pertanto, le prestazioni complessive potrebbero apparire migliori.

6. Recognition Reliability

Misure come FAR, FRR, CMS, ... **non sono sufficienti per dare una valutazione approfondita degli algoritmi**. Per un **confronto affidabile** dei sistemi dobbiamo considerare: il numero e le caratteristiche dei database utilizzati, dimensione delle immagini (immagini più grandi possono rappresentare una risoluzione maggiore), dimensione del Probe e della Gallery (in particolare la dimensione relativa), quantità e qualità delle variazioni indirizzate e tollerate e possibile **interoperabilità** (ad es. generalizzazione tra set di dati).

What's Doddington Zoo?

La maggior parte degli errori nel sistema può essere attribuita a una specifica classe di utenti. Il fatto che una persona appartenga a una delle cattive classi del **Doddington zoo** può pregiudicare l'affidabilità della singola operazione di riconoscimento. Doddington ha definito pecore, capre, agnelli e lupi nel contesto dei sistemi di speaker recognition.

Una **Pecora** è una persona che produce un campione biometrico che combacia bene con gli altri della stessa persona e male con quelli di altre persone. Quindi genera meno false accepts e rejects rispetto alla media. Raggiunge per lo più Genuine Acceptance raramente raggiunge una False Acceptance quando rivendica un'identità diversa.

Una **Capra** è una persona che produce un campione biometrico che corrisponde male con quelli prodotti da lui stesso. Questi punteggi di corrispondenza bassi implicano un false reject rate superiore alla media perché questa persona non è ben riconosciuta.

Un **Agnello** è una persona che può essere facilmente impersonata. Quando il campione biometrico di tale persona viene abbinato a un campione biometrico di un'altra persona, il punteggio di corrispondenza risultante sarà superiore alla media. Di conseguenza, i false matches sono più probabili.

Un **Lupo** è una persona brava a imitare. Quando una tale persona presenta un campione biometrico per il confronto, ha un'alta possibilità di generare un punteggio di corrispondenza superiore alla media rispetto a un campione

memorizzato di una persona diversa. Quindi abbiamo molte False Acceptances.

Capre, agnelli e lupi sono definiti in termini di punteggi medi genuini o impostori di un utente. Le **nuove aggiunte sono definite in termini sia di punteggi autentici di un utente che di quelli dell'impostore.**

I **Camaleonti** sembrano sempre simili agli altri. Questo è qualcosa di leggermente diverso dai Lupi: in pratica, ha un punteggio alto sia quando impersona se stesso, sia quando interpreta il ruolo di impostore. Quindi ricevono un punteggio di corrispondenza elevato per tutte le verifiche. Per questo motivo, i camaleonti raramente causano false rejects, ma è probabile che causino false acceptance. Un esempio di utente che potrebbe essere un camaleonte è qualcuno che ha caratteristiche molto generiche.

I **Fantasm**i portano a punteggi bassi indipendentemente con chi vengono confrontati. Quindi i Fantasm hanno un punteggio basso sia come utenti genuini che come impostori. I fantasm possono essere la causa dei false rejects, ma è improbabile che siano coinvolti nelle false acceptance. Un esempio è dato in questo caso da quelle persone che hanno problemi a registrarsi al sistema. Ciò potrebbe portare a un'estrazione delle features più difficile e di conseguenza a punteggi di corrispondenza bassi per tutte le verifiche. Ad esempio una persona con una pelle scadente sulle impronte digitali può avere problemi con l'iscrizione automatica.

Le **Colombe** hanno un punteggio medio genuino alto e punteggi medi di impostori più bassi. Sono riconoscibili, combaciano bene con se stessi e male con gli altri. Le colombe sono raramente coinvolte in qualsiasi tipo di errore di verifica.

I **Vermi** sono gli utenti peggiori perché come genuini hanno un punteggio medio basso ma come impostori un punteggio medio alto. Hanno poche caratteristiche distintive, quindi pochi sono correttamente riconosciuti come se stessi, ma possono impersonare con successo altre persone.

Come possiamo quantificare l'affidabilità di un sistema di riconoscimento?

A causa della diversa qualità dell'input e della possibile accuratezza nelle procedure di riconoscimento, può accadere che non tutte le risposte siano ugualmente affidabili. Ad esempio, può capitare di avere due immagini del volto della stessa persona, una con gli occhiali e l'altra senza e in questo secondo caso ci manca un elemento importante (la regione periculare) per riconoscere una persona e per questo motivo possiamo dire che il riconoscimento dato dall'immagine con gli occhiali può essere meno affidabile.

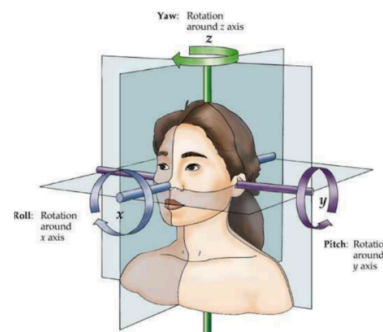
La **definizione di una misura di affidabilità** per ogni singola risposta di un sistema fornisce ulteriori informazioni utili per impostare una politica operativa (ad esempio, se l'identificazione non è abbastanza affidabile e se possibile ripetere l'acquisizione), ma anche per unire i risultati di diversi sistemi (architetture multi biometriche). Quindi il problema dell'affidabilità non è legato all'accuratezza complessiva del sistema ma a come possiamo fidarci della singola decisione del sistema in base ai campioni che ha.

Una possibilità per verificare in anticipo l'eventuale affidabilità di un'operazione di riconoscimento potrebbe essere quella di **analizzare la qualità di un'immagine in arrivo**. Un'immagine catturata può essere: controllata (un'immagine nitida con un buon contrasto quindi va bene), degradata (l'immagine è sfocata ma rimangono molti dettagli) e sfavorevole (contrasto elevato e problemi di illuminazione che rendono più difficile il riconoscimento). Possiamo misurare la correlazione con una cosiddetta "**immagine media**", nel senso che possiamo creare un modello medio da tutte le facce (per quanto riguarda illuminazioni, nitidezza, qualunque cosa), e poi prendiamo la qualità di questo come riferimento.

Un altro modo per misurare la qualità dell'immagine del viso è considerare le **deviazioni della posa**. La deviazione varia in base al **roll** (la rotazione attorno all'asse x), al **pitch** (ovvero la rotazione attorno all'asse y) e alla **yaw** (la rotazione attorno all'asse z). Un modo per misurare o stimare in qualche modo la qualità di una possibile risposta, potrebbe essere analizzare in anticipo la posa del viso in ordine e, se possibile, scattare alcuni fotogrammi nel video in cui il viso ha un yaw o pitch troppo alti. Per misurare la qualità di un'immagine del volto, possiamo ad esempio escogitare una misura complessiva delle distorsioni relative sia all'illuminazione che alla posa del volto. Per quanto riguarda la **misurazione delle distorsioni**:

Come possiamo valutare il Roll? Per misurare il Roll ci affidiamo ai centri degli occhi, tracciamo la linea che collega i centri degli occhi e misuriamo quell'angolo. Maggiore è l'angolo, maggiore è la quantità di roll.

Come possiamo valutare Pitch? Si considerano alcuni punti di riferimento rilevanti: la distanza tra il centro degli occhi e la punta del naso è più o meno uguale alla distanza tra il



mento e la punta del naso o la regione immediatamente sottostante. Misurando il rapporto tra queste due misure possiamo supporre quant'è il valore del Pitch.

Come possiamo valutare Yaw? Facciamo affidamento sulla presunta simmetria del volto quindi misuriamo la distanza tra il centro dell'occhio sinistro e la punta del naso e la distanza tra il centro dell'occhio destro e la punta del naso. Più il viso è frontale, minore è la differenza tra questi due elementi.

La **rotazione che distorce in modo irreparabile** un'operazione di riconoscimento riguarda il pitch ed è quando si **abbassa la testa**.

Possiamo anche provare a misurare la **“qualità” di una misura di qualità** per capire quali sono le misure migliori per anticipare l'attendibilità dei risultati ottenuti. Quindi possiamo fare dei confronti e possiamo identificare come queste misure possono influenzare il threshold da adottare per avere un buon riconoscimento:

- Il primo test per una misura di qualità consiste nel verificare come i valori di un dataset sono distribuiti rispetto ai valori restituiti. Questo permette di capire qual è il livello medio di qualità di un dataset facciale rispetto ad una determinata misura;
- Una misura della qualità dei campioni in ingresso consente di scartare a priori (prima del riconoscimento) quelli affetti da una distorsione troppo elevata che porterebbe ad una risposta errata o non affidabile da parte del sistema;
- Una misura di buona qualità deve fornire una buona riduzione dell'errore scartando il minor numero possibile di campioni.

Un altro approccio possibile utilizza i **margini basati sulla stima dell'errore** in cui il margine è definito come la differenza tra FAR e FRR dato un certo threshold.

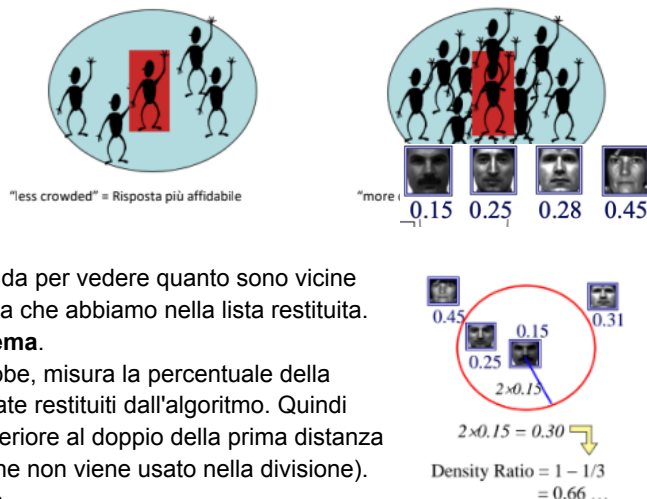
Un altro approccio è il **System Response Reliability (SRR)** che misura l'abilità del sistema nel dividere i soggetti genuini con gli impostori per ogni probe. L'SRR si basa su diverse versioni della funzione ϕ (phi). Abbiamo due diverse funzioni che sono la distanza relativa e il **density ratio**. Entrambe le funzioni misurano la quantità di **"confusione"** tra i possibili candidati cioè quanto è **affollato il soggetto**, più è affollato più è probabile che sia un impostore e si avrà una risposta meno attendibile di conseguenza.

La **Distanza Relativa** misura la differenza tra la prima distanza restituita (nella lista ritorna dall'algoritmo di riconoscimento) e la seconda per vedere quanto sono vicine l'una all'altra. Quindi prendiamo il rapporto rispetto alla distanza più alta che abbiamo nella lista restituita.

Quindi **più basso è questo valore, peggiore è l'affidabilità del sistema**.

Un'alternativa alla distanza relativa è il **Density Ratio** che, dato un probe, misura la percentuale della galleria che è molto vicina al probe secondo l'elenco ordinato di template restituiti dall'algoritmo. Quindi contiamo quanti template nell'elenco hanno una distanza dal probe inferiore al doppio della prima distanza e la dividiamo per il numero di template nella galleria (meno il probe che non viene usato nella divisione).

Quindi **più alto è questo valore, migliore è l'affidabilità del sistema**.



Occorre individuare un valore che favorisca una corretta separazione tra errati respingimenti di soggetti iscritti e errati riconoscimenti di non iscritti, entrambi supportati dal valore di attendibilità. Il ϕ_k critico è un valore in grado di minimizzare le stime errate della funzione $\phi_{(p)}$, cioè impostori erroneamente riconosciuti con $\phi_{(p)}$ maggiore di ϕ_k , oppure soggetti genuini erroneamente scartati perché riconosciuti con $\phi_{(p)}$ minore di ϕ_k . **La distanza tra $\phi_{(p)}$ e ϕ_k è significativa per l'affidabilità**. Definiamo:

$$S(\phi_{(p)}, \bar{\phi}) = \begin{cases} 1 - \bar{\phi} & \text{if } \phi_{(p)} > \bar{\phi} \\ \bar{\phi} & \text{otherwise} \end{cases}$$

Se $\phi_{(p)}$ è maggiore del valore critico ϕ_k allora prendiamo come possibile intervallo entro il quale ϕ può assumere valori rispetto al valore critico come $1 - \phi_k$, altrimenti andiamo sotto e questo significa che prendiamo come fattore di normalizzazione esattamente ϕ_k . Alla fine, il valore finale per il **System Response Reliability (SRR)** è la differenza tra il valore ottenuto di $\phi_{(p)}$ e ϕ_k e questa funzione fornisce il massimo che potrebbe essere raggiunto:

$$SRR = (\phi_{(p)} - \bar{\phi}) / S(\bar{\phi})$$

Dopo aver calcolato una formulazione ragionevole per il SRR possiamo anche impostare un **threshold per l'affidabilità** per valutare quando sia effettivamente conveniente rifiutare anche un'identificazione apparentemente corretta per il fatto che non la consideriamo sufficientemente affidabile. Quindi ora abbiamo due threshold: una che determina le false accettazioni rispetto alla verifica e all'identificazione e l'altra per l'affidabilità.

Una possibile soluzione per aumentare l'affidabilità del sistema è quella di **aggiornare i template nella galleria**. Una volta che abbiamo un certo numero di template memorizzati, quando siamo abbastanza sicuri che l'operazione di riconoscimento funzioni, potremmo decidere di includere un nuovo probe all'interno della galleria. Eventualmente eliminando alcuni template meno informativi. Un'altra soluzione è affrontare il problema dei **template di scarsa qualità**. Ad esempio, se disponiamo di un'impronta digitale che è stata registrata per qualche motivo con un sensore a bassa risoluzione, una volta acquistato un nuovo sensore, man mano che arrivano nuovi probe, includiamo i probe acquisiti con il nuovo sensore nella galleria aggiornata. In generale abbiamo diversi modi per eseguire l'aggiornamento dei template:

- Supervisionato: C'è una persona (un supervisore) che, una volta riconosciuta una persona, determina se tale riconoscimento era corretto e decide di includere il template nella galleria.
- Semi-supervisionato: Nessuna persona supervisiona il processo. Quindi potremmo provare ad applicare qualche tipo di statistica per confrontare il nuovo probe con la galleria esistente.

Possiamo anche selezionare i template più rappresentativi (dopo aver ottenuto troppi template, l'identificazione diventa troppo grande). La selezione dei template più rappresentativi da utilizzare per l'aggiornamento può essere effettuata:

- Online: La selezione viene effettuata non appena i nuovi dati di input vengono acquisiti dal sistema di riconoscimento;
- Offline: la selezione viene eseguita dopo che una certa quantità di dati è stata acquisita durante uno specifico lasso di tempo.

7. Ear Recognition

Esistono molti modi per descrivere un volto, ma pochi modi per descrivere un **orecchio**. L'orecchio ha una struttura non-randomica, ma ben definita costituita da varie parti che convergono nel **punto zero** cioè il centro dell'orecchio da cui tipicamente partono le misurazioni. Alcuni **aspetti positivi** sono:

- L'orecchio è un tratto biometrico passivo e statico (e quindi **permanente**). L'orecchio cresce in maniera proporzionale fino ai 4 mesi di età poi tende ad allungarsi fino agli 8 anni e poi rimane costante fino a circa 70 anni in cui si allunga ulteriormente a causa del rilassamento dei tessuti.
- Poiché è associato a uno dei sensi umani, di solito viene lasciato scoperto per un migliore udito.
- Se poco visibile è possibile interagire con l'utente.
- Se confrontato con altri dati biometrici, in particolare il viso, alcuni vantaggi sono che ci sono meno dettagli e quindi richiede una risoluzione inferiore per il riconoscimento e c'è una distribuzione del colore più uniforme.

Uno dei **problemi principali** è l'occlusione che è dovuta alla presenza dei capelli. Un altro problema è dato dal fatto che le orecchie sono intrinsecamente strutture 3D, quindi quando cambiamo l'orientamento, hanno bisogno di una sorta di procedura di normalizzazione per correggere in qualche modo quel cambiamento.

L'orecchio ha una dimensione inferiore così come una complessità inferiore rispetto al viso, ma allo stesso tempo ha lo stesso colore della pelle circostante. Questo rende la **localizzazione** e l'estrazione più difficile.

La localizzazione può essere fatta in differenti modi:

- Un possibile approccio utilizza le **reti neurali**, che devono essere addestrate su una grande quantità di immagini. Su ogni immagine vengono selezionati i punti di interesse (in giallo) che ci permetteranno quindi di effettuare la localizzazione. Poiché l'addestramento per le reti neurali richiede un tempo proporzionale alla dimensione dell'immagine in ingresso, e poiché in questa fase non siamo molto interessati ai dettagli fini, le immagini vengono ridimensionate a una risoluzione inferiore. Il contrasto in ogni immagine viene normalizzato per risultati migliori.
- Localizzazione dell'oggetto all'interno di un'immagine utilizzando **AdaBoost**. Si ottengono campioni positivi (cioè immagini che contengono l'orecchio), più campioni negativi (cioè immagini che non lo contengono). L'unione dei due insiemi costituisce il training set. Si effettua quindi il training estraendo le caratteristiche rilevanti cioè quelle con la maggiore capacità di classificare gli oggetti interessanti. È

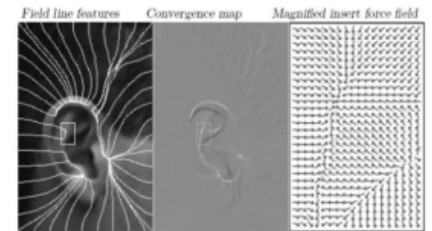


possibile aggiungere nuovi campioni positivi o negativi e ripetere l'addestramento se rispettivamente il tasso di oggetti mancati o il tasso di falsi allarmi cresce troppo.

- Le **tecniche 3D** si basano su una valutazione della profondità e della curvatura delle regioni dell'orecchio. Il training è offline e viene effettuato da un modello 3D del profilo del viso quindi vengono identificati i punti di massima curvatura, viene creata un'immagine binaria e la regione corrispondente all'orecchio viene estratta manualmente. Le regioni estratte vengono fuse insieme per formare un modello di riferimento. Il testing viene eseguito online quindi dato un nuovo modello viene calcolata l'immagine binaria, vengono identificati i punti di curvatura massima e minima e vengono ricercate le regioni corrispondenti all'orecchio.

Gli **approcci di riconoscimento** possono essere classificati in diversi modi: Approcci geometrici 2D (globali) basati su curve/punti di riferimento, Modelli 3D e Termogrammi. Abbiamo vari **approcci geometrici 2D**:

- **Sistema di Iannarelli**: si basa su un insieme di misurazioni fatte sull'orecchio. Richiede un allineamento molto accurato e la normalizzazione delle due immagini da confrontare. Prima di tutto si individua il **crus of helix** e lo si fissa come centro dell'orecchio. A partire da questo punto si effettuano 12 diverse misurazioni (valori interi) e a partire da questo si crea il feature vector insieme alle informazioni circa il sesso e l'etnia della persona. Il problema principale con questo metodo è la precisione richiesta nell'individuare il punto centrale infatti se l'identificazione non è corretta, tutte le misurazioni sono sbagliate.
- **Voronoi diagrams**: siccome l'illuminazione rende inefficace il metodo di Iannarelli, allora è stata proposta una nuova soluzione, basata sui diagrammi di Voronoi e sulla distanza tra grafi. Sebbene il metodo non sia stato sperimentato, la segmentazione dell'orecchio è troppo sensibile a variazioni di illuminazione e posa, rendendo inefficace la rappresentazione di Voronoi.
- **Campi di forza**: Ciascun pixel dell'orecchio è visto come una particella carica (0 neutro, 255 carica massima) che esercita un campo di forza (che si assume a simmetria sferica). La forza totale che agisce su un pixel è pari al contributo di tutte le forze dovute agli altri pixel nell'immagine. Per ciascun pixel si calcola la forza che tutti gli altri pixel esercitano su di esso. Si fissano una serie di punti su un'ellisse intorno all'orecchio. A partire da ciascun punto si segue l'attrazione del campo di forza. Le linee di campo convergono in punti detti pozzi. L'orecchio è rappresentato tramite la mappa di convergenza.
- **Jets**: Un certo numero di funzioni del kernel che determinano i filtri Gabor con diversi gradi di orientamento vengono messe in convoluzione con l'immagine, questo produce feature vectors definiti come Gabor Jets. Questi Jet sono memorizzati come grafici dell'orecchio nella galleria e PCA viene utilizzato per ottenere il grafico dell'eigen-ear. Un orecchio viene rilevato utilizzando la somiglianza tra i Jet nella galleria e quelli ricostruiti a partire dal probe;
- **Angle vectors**: see slides..



Nel riconoscimento dell'orecchio tramite **modelli 3D**, i metodi valutano la profondità e la curvatura delle regioni dell'orecchio rilevanti. In alcuni casi il confronto viene effettuato fra regioni corrispondenti (dette patch) di due modelli 3D di orecchio appartenenti alla stessa persona.

Nel riconoscimento dell'orecchio tramite **termogramma**, l'immagine dell'orecchio viene catturata da una termocamera. Alcuni vantaggi sono che l'orecchio è facilmente localizzabile, è resistente all'occlusione dei capelli e i diversi colori facilitano la segmentazione. Gli svantaggi sono: è sensibile al movimento, la risoluzione è bassa e i costi sono alti.

Riconoscimento dell'orecchio nei sistemi multimodali: numerosi sistemi combinano informazioni 2D e 3D per migliorare le prestazioni di riconoscimento. È possibile combinare viso e orecchio in diversi modi:

- Faccia → Orecchio: sfoiisci usando la faccia, poi verifica usando l'orecchio;
- Orecchio → Viso: sfoiisci usando l'orecchio, quindi verifica usando il viso;
- Viso ⊕ Orecchio: combina le informazioni sul viso e sull'orecchio prima dell'identificazione.

I test effettuati su un numero significativo di soggetti hanno dimostrato che l'opzione migliore è la combinazione parallela di viso e orecchio.

8. Fingerprint Recognition

Un'**impronta digitale** di solito appare come una serie di linee scure che rappresentano la parte alta e appuntita della pelle della cresta, mentre le valli tra queste creste appaiono come uno spazio bianco e sono la parte bassa e poco profonda della cresta.

Esistono due **modalità di base per l'acquisizione** delle impronte digitali:

- L'acquisizione **offline** avviene in due fasi: Si passano prima i polpastrelli su un tampone di inchiostro e poi si trasferisce l'immagine dell'impronta digitale tramite pressione su carta o una successiva digitalizzazione dell'impronta tramite scansione ottica o fotocamera ad alta risoluzione.
- Nell'acquisizione **live-scan**, l'immagine digitale dell'impronta digitale viene acquisita direttamente tramite il contatto del polpastrello con un apposito sensore. In questo caso potremmo avere del rumore dovuto allo sporco sui sensori o ad altri possibili fattori che potrebbero portare ad un'errata acquisizione dell'impronta.

Le **Impronte Latenti** (Latent Fingerprints) sono impronte digitali che lasciamo su qualsiasi superficie quando la tocchiamo. In questo caso dobbiamo confrontare un'impronta digitale che di solito è completa (l'impronta digitale registrata nella galleria) con un'impronta digitale latente che di solito è una frazione (frammento dell'impronta digitale completa). Quindi in una corrispondenza tra un'impronta latente con un'impronta registrata potrebbe esserci il problema dell'**Allineamento**. L'acquisizione delle impronte latenti avviene grazie a speciali polveri e a particolari tecniche che vengono utilizzate per trasferire l'impronta da una superficie su una carta speciale in maniera molto simile a come avviene l'acquisizione offline.

Abbiamo diversi livelli di impronte digitali (**feature levels**): Il primo è quello formato dalle **Macro-Singularities**. Sono anche chiamate "first level features" e comportano una considerazione globale del modello di cresta. Si basano sul numero di "deltas": spirali (2 delta), loop (1 delta) e archi (nessun delta). Il secondo livello, detto "**second level feature**", è formato dalle "**Minutiae**". Queste sono chiamate anche Galton features e sono delle "**micro-singularities**" determinate dagli terminazioni o dalle biforcazioni delle linee di cresta. La prima tipologia, sebbene utile per tagliare la ricerca, non è decisiva per un riconoscimento finale. Il numero di minutiae corrispondenti che si trovano in una coppia di impronte digitali da confrontare viene utilizzato come misura della distanza. Con un sensore ad altissima risoluzione è possibile studiare anche i **pori lungo le creste** che sono tratti estremamente distintivi. Sono chiamati "**third level features**".

Alcuni **problemi con le impronte digitali** sono:

- **Sovrapposizione scarsa**, ad esempio è possibile che un dito non sia ben centrato sul sensore e quindi potremmo avere un'inquadratura completamente diversa.
- **Diverse condizioni della pelle**, ad esempio una pelle secca, possono ostacolare una corretta lettura dell'impronta digitale.
- **Troppo movimento e/o distorsione non lineare della pelle** perché abbiamo una struttura tridimensionale che può essere modificata dall'elasticità della pelle in base alla pressione, in modo che se acquisiamo la stessa impronta in due sessioni diverse questa potrebbe risultare differente.
- **Errori nell'estrazione delle caratteristiche** perché gli algoritmi di estrazione delle caratteristiche sono imperfetti e spesso introducono errori di misura, in particolare con impronte di bassa qualità.

Quali sono i tipi di scanner digitali?

- Optical Scanner: poco costoso, robusto alla variazione di clima e con buona risoluzione, ma è grande e va pulito molto bene dopo ogni utilizzo;
- Capacitive Scanner: migliore risoluzione dell'impronta e dimensioni ridotte ma l'acquisizione è molto dura;
- Thermal Scanner: non può essere ingannato da impronte artificiali perché riconosce pulsazione, temperatura, pori e cambiamento di colore della pelle tramite pressione, anche se l'immagine scompare rapidamente, è infatti possibile riprodurre Fake Fingerprints tramite gelatina, silicone e lattice.

Quali sono le tecniche di confronto?

- **Matching basato sulla correlazione**: le due immagini vengono sovrapposte e viene iterato il calcolo della correlazione tra pixel corrispondenti per diversi allineamenti, che si ottengono tramite roto-traslazioni fino a

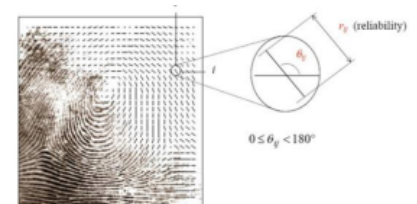
determinare il grado di similarità tra i campioni; sensibile alle trasformazioni rigide e non lineari; elevata complessità computazionale.

- **Matching basato sulle caratteristiche delle creste:** l'estrazione di minuzie nelle immagini delle impronte digitali di bassa qualità è problematica, quindi questi metodi utilizzano altre caratteristiche come l'orientamento delle creste e la frequenza locale, la forma delle creste e la trama, che sono più affidabili e più facili da estrarre, ma anche meno distintivo; basso potere discriminante.
- **Matching basato su minuzie:** le minuzie vengono prima estratte dalle due impronte e memorizzate come due insiemi di punti in uno spazio bidimensionale (eventualmente annotato con l'angolo tra la tangente e il piano orizzontale), quindi i metodi ricercano l'allineamento tra i due set che massimizza il numero di coppie corrispondenti di minuzie, e in base a questa misura la somiglianza tra le impronte digitali (point pattern matching).

Quali sono i diversi livelli di riconoscimento delle impronte digitali?

Prima di tutto dobbiamo segmentare. La **segmentazione** in questo caso indica la separazione tra l'impronta digitale in primo piano (pattern striato e orientato - anisotropo) dal fondo che è isotropo. Una delle caratteristiche delle impronte digitali è l'**anisotropia**, nelle impronte abbiamo varietà di contrasto e intensità che dipendono dall'orientamento. Lo sfondo invece è **isotropo** cioè rimane identico in tutte le direzioni, infatti lo sfondo è tipicamente bianco.

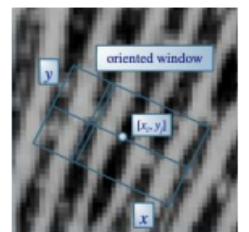
Quindi possiamo estrarre le **macro-caratteristiche** dalle **creste**. Il flusso della linea di cresta può essere descritto da una struttura chiamata **Mappa Direzionale** (o directional map) che è una matrice discreta i cui elementi denotano l'orientamento della tangente alle linee della cresta. È una matrice in cui ogni elemento corrispondente al nodo $[i, j]$, che si sovrappone all'immagine dell'impronta digitale, denota l'orientamento medio della cresta tangente o l'orientamento in un piccolo vicinato.



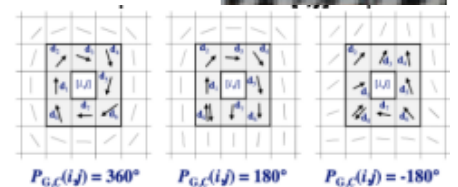
Analogamente, la densità della linea di cresta può essere sintetizzata utilizzando una **Mappa di Densità** (density map), che misura la densità delle linee di cresta in ciascuna regione della punta del dito. L'orientamento locale della linea di cresta nella posizione $[i, j]$ è definito come l'angolo $\theta(i, j)$ che la linea di cresta forma con l'asse orizzontale.

Possiamo considerare un altro approccio che si basa sul gradiente (**gradient**): questo è un modo per estrarre l'orientamento in base al gradiente dell'immagine, ma la stima dell'orientamento rappresenta un'analisi di basso livello che è troppo **sensibile al rumore**. D'altra parte non è invece possibile fare una semplice media di più gradient e questo è dovuto alla circolarità degli angoli. Il concetto di orientamento medio non è sempre ben definito specialmente quando la griglia non è densa (si ha quindi una griglia rada).

D'altra parte, la **Frequency Map** studia per ogni punto dell'impronta digitale il numero di creste per unità di lunghezza lungo un ipotetico segmento centrato in $[x, y]$ e ortogonale all'orientamento della cresta locale. Un possibile approccio consiste nel contare il numero medio di pixel tra picchi consecutivi di livelli di grigio lungo la direzione ortogonale all'orientamento locale della linea di cresta.



Quindi possiamo trovare le **singolarità** usando l'indice di **Poincaré** che viene quindi utilizzata per trovare le macro-singularities. Una directional map è un campo vettoriale (una regione piena di vettori con orientamenti diversi). La nostra impronta digitale è una curva C immersa in questo campo vettoriale G . L'indice di Poincaré $P_{G,C}$ è definito come la rotazione totale dei vettori del campo G lungo la curva C . Tale indice $P_{G,C(i,j)}$ si calcola sommando algebricamente le differenze di orientazione tra elementi adiacenti di C . Per le curve chiuse, l'indice di Poincaré assume solo uno dei valori discreti $0^\circ, \pm 180^\circ, \pm 360^\circ$. In particolare, per quanto riguarda le singolarità delle impronte digitali: 0° significa che non ci sono singolarità; 360° denota la presenza di un vortice; 180° indica un loop; -180° denota un delta.



Il passo successivo è **trovare le minuzie** (micro-singularities). Molti approcci estraggono minuzie e eseguono il match basandosi su di esse o in combinazioni di esse. In generale, l'estrazione delle minuzie comporta:

- Una procedura di **binarizzazione** per convertire un'immagine in scala di grigi in un'immagine binaria (bianco e nero);

- Una procedura di **assottigliamento** che trasforma qualsiasi cresta più spessa di 1 pixel a uno spessore di 1 pixel;

Quindi una volta che abbiamo immagini in bianco e nero con creste assottigliate possiamo procedere alla scansione di tali creste per individuare i pixel che corrispondono a minuzie. La localizzazione delle minuzie si basa sull'analisi del numero di cambiamenti di colore (**crossing number**) che avvengono nelle vicinanze del pixel di cui si tiene conto in quel momento. Calcoliamo per ogni pixel questo crossing number per determinare se rappresenta una minuzia o meno. Se il crossing number è uguale a 2 ad esempio, il pixel rappresenterà un punto interno di una linea di una cresta. Se il crossing number corrisponde a 1, quel pixel è un punto di terminazione perché abbiamo un solo cambio di direzione. Se invece è uguale a 3, indica una biforcazione. Se è maggiore di 3, allora appartiene a una minuzia più complessa.

Tra le caratteristiche usate dagli esperti per il riconoscimento delle impronte digitali c'è anche il numero di ridge (**ridge count**) che è una misura astratta della distanza tra due punti qualsiasi di un'impronta. Presi due punti a e b di un'impronta, il ridge count corrisponde al numero di ridge line intersecate dal segmento ab. Sebbene il ridge count sia definito per due punti qualsiasi, solitamente a e b sono scelti in corrispondenza di punti fissi dell'impronta digitale (es. core o delta). Questo non può essere fatto per ogni coppia di minuzie perché gli endpoint non sono abbastanza affidabili (possono essere causati da un'interruzione di una cresta a causa di una soglia errata).

Un possibile **approccio ibrido** per confrontare le impronte digitali combina la rappresentazione delle impronte digitali basata su minuzie con una rappresentazione basata sul **filtro di Gabor** che utilizza informazioni sulla trama locale. La **fase di allineamento** inizia con l'estrazione delle minuzie sia dall'input che dal template nella galleria. Le due serie di minuzie vengono confrontate attraverso un algoritmo di point matching che seleziona preliminarmente una coppia di minuzie di riferimento (una da ciascuna immagine), e quindi determina il numero di coppie di minuzie utilizzando l'insieme di punti rimanente. La coppia di riferimento che produce il numero massimo di coppie corrispondenti determina il miglior allineamento. Le aree dello sfondo dell'input vengono mascherate come non necessarie. Per eseguire l'estrazione delle caratteristiche dalle celle risultanti dalla tassellatura viene utilizzato un gruppo di **8 filtri Gabor**, tutti con la stessa frequenza, ma con orientamento variabile. Tale filtraggio produce 8 immagini ordinate per ogni cella.

La **feature extraction** avviene in questa maniera: I valori caratteristici di tutte le celle vengono quindi concatenati in un vettore caratteristico. Il **confronto dell'immagine in ingresso con il template** memorizzato viene effettuato calcolando la somma dei quadrati delle differenze tra i corrispondenti vettori caratteristici. Se il punteggio di somiglianza è maggiore ad un certo threshold, è possibile affermare che l'immagine in input ha un modello corrispondente in memoria e il riconoscimento ha esito positivo.

Come si fa a creare una impronta falsa?

I materiali più popolari sono gelatina, silicone, lattice. Ogni materiale produce impronte di diversa qualità e con caratteristiche diverse.

Si può effettuare liveness detection?

Una misura di sicurezza è quella di determinare se la sorgente del segnale in ingresso (il dito) sia un vero tratto biometrico vivente mediante **liveness detection**. Se il dito è vivo, la sua impronta è in realtà della persona a cui appartiene. Uno degli approcci più comuni per effettuare tale test consiste nell'utilizzare uno o più parametri vitali come ad esempio polso e temperatura. Gli scanner ottici di tipo live-scan utilizzano un meccanismo di acquisizione differenziale per le creste e i solchi delle impronte digitali, risultando in questo modo più resistenti agli attacchi in cui si simula artificialmente un'impronta.

La scansione ad alta risoluzione dell'impronta digitale rivela dettagli caratteristici della struttura dei pori che sono molto difficili da imitare in un dito artificiale. Il colore caratteristico della pelle del dito cambia a causa della pressione quando viene premuto sulla superficie di scansione e questo effetto può essere rilevato per identificare l'autenticità del dito. Anche il flusso sanguigno e la sua pulsazione possono essere rilevati con un'attenta misurazione della luce riflessa o trasmessa attraverso il dito. La differenza di potenziale tra due punti specifici della muscolatura del dito può essere utilizzata per distinguerlo da un dito morto. La misurazione dell'impedenza complessa del dito può essere utile per verificare la vitalità del dito. Infine, un dito suda e questo può determinare la vitalità del dito.

9. Iris Recognition

Il **colore dell'iride**, la **trama "regolare"** (per lo più solcata) e i **motivi "irregolari"** (ad esempio, lentiggini e cripte) forniscono un **livello di discriminazione molto elevato**, paragonabile alle impronte digitali.

In ogni caso l'iride ha **svariati problemi** quali:

- **Riflessione**: che dipende da dove proviene la fonte di luce. Questa può anche essere modificata dalla presenza di lenti od occhiali.
- **Piccole dimensioni** dell'iride che richiedono un'apparecchiatura ad alta risoluzione;
- La **limitata profondità di campo** per cui dobbiamo prestare attenzione ai problemi di messa a fuoco e fuori fuoco che possono portare ad una cattiva risoluzione;
- Dobbiamo **allinearci con l'asse ottico** a meno che non usiamo alcuni accorgimenti come il cosiddetto "problema fuori asse" (se la persona guarda in un'altra direzione, l'iride non sarà esattamente al centro della sclera, ma si sposterà verso l'estremità destra o sinistra della sclera);

Alcuni aspetti positivi invece sono:

- L'iride è **visibile ma ben protetta**, quindi all'aumentare della risoluzione dell'attrezzatura di cattura aumenta anche la capacità di catturare l'iride ad una distanza minore;
- È **time invariant** (cioè dopo due anni il tratto è ben definito e non cambia).
- E' un tratto **estremamente distintivo** poiché ha componenti altamente **randomici** cioè non affetti da eredità familiare (destro diverso da sinistro e anche i gemelli hanno iridi diverse).
- La sua immagine può essere acquisita **senza contatto diretto**, quindi è ben accettabile non come il fondo retinico che richiede di toccare la superficie oculare con il dispositivo di acquisizione.
- Può essere acquisito sia nelle **lunghezze d'onda (wavelet) vicine all'infrarosso** che in quelle **visibili**.

Le due principali **modalità di acquisizione** sono la Visible Light e la Infrared light.

Con la **Visible Light** abbiamo la melanina che assorbe la luce visibile, quindi possiamo vedere chiaramente i diversi colori che possono apparire nell'iride. Gli strati che compongono l'iride sono ben visibili ma l'immagine contiene informazioni rumorose sulla trama. In questo caso, possiamo avere il problema della **Riflessione** che dipende dalla fonte di luce e dovrebbe essere rilevata in anticipo. Inoltre presenza di lenti a contatto o occhiali può influenzare il tipo di riflesso che può interessare l'iride. È anche importante considerare quando abbiamo a che fare con un'**iride molto scura**, perché in quel caso la trama dell'iride può diventare completamente invisibile.

Con la **Infrared Light** la melanina riflette la maggior parte della luce infrarossa, quindi non abbiamo informazioni sul colore. La trama è più visibile ma richiede attrezzature speciali. Se adottiamo Infrared sensors risolviamo i due problemi precedenti perché è possibile evitare la maggior parte dei riflessi che si fondano nella luce visibile ed è possibile evidenziare la trama dell'iride scura. Tuttavia, nelle immagini nel vicino infrarosso **mancano le componenti cromatiche** quindi tutte le differenze di colore vengono eliminate. Inoltre c'è un altro problema: i sensori vicini all'infrarosso non sono universalmente disponibili come quelli a luce visibile.

Quali sono le fasi di lavorazione dell'iride?

Prima di tutto c'è la **Segmentazione**, dove scartiamo i riflessi e manteniamo solo la parte che si trova all'interno del segmento di cerchio che è stato rilevato e che riguarda solo la parte visibile dell'iride.

Quindi abbiamo un processo di **Normalizzazione**. Usiamo le coordinate polari perché qualunque cosa sia circolare nell'immagine sorgente, diventa rettangolare nell'immagine proiettata, in modo che i cerchi concentrici strutturali che costituiscono le informazioni più importanti nell'immagine dell'iride, diventino linee in questa mappatura polare. Le linee sono molto più facili da elaborare rispetto ai cerchi da un punto di vista matematico.

Quindi abbiamo la fase di **codifica**. Questa fase tiene spesso conto dei risultati della segmentazione precedente, per cui in genere si cerca di non codificare i cosiddetti "elementi di sfondo" (che non ci interessano) quindi teniamo conto solo dell'iride.

Alla fine abbiamo il **Matching** tra due iridi che fornisce una misura di distanza e, in base a tale distanza e al tipo di applicazione (verifica o identificazione) che stiamo effettuando, decide l'esito finale dell'operazione.

Il **Rubber Sheet Model** tiene conto del problema dell'asse e in particolare del problema di avere una pupilla non perfettamente centrata all'interno dell'iride. Le **coordinate polari** semplificano l'elaborazione dell'iride (le bande circolari diventano strisce orizzontali, quindi l'intero anello dell'iride diventa un rettangolo).

L'**iris unwrapping** è il processo di trasformazione dall'iride circolare in coordinate cartesiane in un rettangolo in coordinate pseudo polari. Determinare il punto centrale per le coordinate polari è di fondamentale importanza ma la **pupilla e l'iride lo sono non perfettamente concentriche** e la **dimensione della pupilla può cambiare a causa dell'illuminazione o condizioni patologiche** (ubriachezza o droghe) e il **Rubber Sheet Model** è un processo di normalizzazione che tiene conto di questi problemi.

Prendendo un numero fisso di punti su ogni raggio che è contenuto tra i bordi della pupilla e dell'iride, è possibile **normalizzare** la distanza deformata: quando abbiamo una regione più ampia il campionamento sarà meno denso, al contrario avremo un campionamento più denso in una regione più ristretta. Alla fine, saremo in grado di ottenere a rappresentazione che rappresenta l'iride ideale con la **pupilla perfettamente centrata**.

Il modello mappa ogni punto dell'iride in coordinate polari dove il centro delle coordinate polari è il centro della pupilla. La trasformazione non è una semplice trasformazione polare perché le nuove coordinate per ogni punto sono date da una combinazione lineare tra le coordinate del contorno pupillare e quelle del contorno esterno dell'iride.

Il modello cerca quindi di compensare la **dilatazione della pupilla** e le **variazioni dimensionali** producendo una **rappresentazione invariante**. Il modello non compensa le rotazioni. Tuttavia, durante il matching, in coordinate polari, questo viene fatto traslando l'iride ottenuta modello fino all'allineamento.

Una volta estratta la curva contenente l'iride e una volta che questa viene normalizzata, Daugman applica i **Gabor Filters** nelle coordinate polari per ottenere il cosiddetto **codice dell'iride**.

Possiamo confrontare due codici dell'iride usando la **distanza di Hamming** dove N è il numero totale di valori nel rettangolo. Per ogni punto corrispondente, consideriamo se hanno o meno lo stesso valore. Se hanno lo stesso valore (qualunque esso sia), abbiamo uno 0 nella distanza di Hamming, ogni volta che i valori sono diversi, aggiungiamo un 1 nella distanza di Hamming. Una volta effettuata la somma, la dividiamo per il numero di valori.

10. Multibiometric Systems

I **Sistemi Multibiometrici** o anche **Insieme di Classificatori** sono una delle soluzioni proposte che possono permetterci di migliorare le prestazioni di un sistema biometrico. Quando usiamo più tratti biometrici è molto più difficile falsificarli, perché l'attaccante dovrebbe usare un campione falso per ciascuno dei tratti coinvolti.

Nei sistemi multibiometrici abbiamo più dispositivi di acquisizione: uno per ogni tratto biometrico. Ogni tratto biometrico può essere elaborato individualmente e quindi, a un certo punto del processo, possiamo unire i risultati ottenuti. Ci sono vari modi per combinare diverse risposte biometriche e il più ovvio è quello di **unire più tratti**: questo è il puro **approccio multimodale** o **Multimodal Approach**.

D'altra parte, possiamo anche avere approcci in cui possiamo sfruttare istanze multiple dello stesso tratto, ovvero il **Multiple Instances**.

Un altro tipo di sistema multi biometrico è un sistema che considera istanze ripetute, quindi il **Repeated Instances**. In questo caso oltre ad avere esattamente lo stesso tratto biometrico, ne catturiamo due campioni. La differenza tra questi ultimi due approcci è che con il Repeated Instances abbiamo ad esempio due impronte dello stesso dito, mentre nel Multiple Instances abbiamo due impronte di due dita diverse. L'idea delle Repeated Instances è che pressioni diverse o alcuni problemi occasionali possono creare differenze nei campioni ottenuti.

Un altro approccio popolare è il **Multiple Algorithms** cioè diversi algoritmi i cui risultati possono portare alla decisione finale. Ad esempio, applichiamo LBP e i Wavelets per elaborare un'immagine del viso, gli algoritmi di matching sono adatti ai diversi metodi di estrazione delle caratteristiche che applichiamo e quindi possiamo fondere i risultati di due diversi algoritmi.

Infine, abbiamo il caso dello stesso tratto o di tratti diversi (ma in generale lo stesso tratto) catturati da sensori multipli, quindi il **Multiple Sensors**. Ad esempio possiamo utilizzare due sensori diversi per acquisire la stessa impronta digitale (entrambi ottici o entrambi capacitivi o uno ottico e uno capacitivo o ecc...) e poi fondere i risultati.

Come si fondono i risultati negli approcci biometrici?

Il primo approccio è **Sensor Level Fusion**. Un esempio di questo approccio lo si ha quando fondiamo informazioni da varie immagini 2D in un unico modello 3D. In questo caso la fusione viene eseguita prima dell'estrazione delle features. Le features utilizzate per il matching e per la decisione vengono estratte direttamente dal modello fuso.

La Sensor Level Fusion è difficile da realizzare perché dobbiamo avere prima di tutto **lo stesso tratto** e il **tipo di segnale che viene estratto deve essere lo stesso**.

L'approccio successivo è quello del **Feature Level Fusion**. In questo caso acquisiamo diversi campioni (uno per ogni tratto) e poi fondiamo i feature vectors che sono stati estratti. Un esempio può essere l'estrazione delle features dall'occhio destro e dall'occhio sinistro; estraiamo separatamente i due codici dell'iride e poi li fondiamo in un unico vettore che viene poi abbinato. Questo approccio può generare feature vectors di dimensioni molto grandi, quindi si è soggetti alla "**curse of dimensionality**", quindi di conseguenza devono essere necessariamente eseguiti delle riduzioni di dimensionalità dello spazio oppure una feature selection. Un altro problema è l'**inflexibilità** di questo sistema perché il match viene addestrato utilizzando la nozione di "vettore fuso" e questo può essere un problema se vogliamo sostituire un probe o aggiungerne uno nuovo. Inoltre abbiamo problemi di **compatibilità**: Se una delle strategie di estrazione delle caratteristiche restituisce un istogramma e un'altra strategia di estrazione delle caratteristiche restituisce un insieme di coefficienti wavelet, la feature level fusion non è fattibile.

L'approccio più popolare e più flessibile è lo **Score Level Fusion**. In questo approccio ogni tratto viene elaborato da un sottosistema diverso. Quindi abbiamo le feature extraction ed i matching che avvengono in maniera separata. La fusione verrà fatta solo dopo i matching. In questo caso il problema è trovare una buona strategia di fusione. In generale abbiamo due possibili strategie:

- **Transformation based:** I punteggi di diversi matcher vengono prima normalizzati in un dominio comune e quindi combinati utilizzando regole di fusione.
- **Classifier based:** Invece di provare a fondere i punteggi, raccogliamo i punteggi in un nuovo feature vector, e lo utilizziamo per addestrare un classificatore.

Le regole di fusione che possono essere utilizzate per lo Score Level Fusion sono Abstract, Rank e Measurement. Nell'**Abstract** il classificatore restituisce un class label per il modello di input. Questo class label rappresenta la risposta del classificatore a una determinata operazione di matching. Ad esempio, nel caso della verifica il sistema può restituire o un'etichetta sconosciuta (se la persona non è riconosciuta) o l'etichetta con l'identità dichiarata. Nel caso dell'identificazione ogni classificatore può riconoscere il probe come un'identità diversa. Uno dei modi più utilizzati per fondere i risultati è il voto di maggioranza o **majority voting**: ogni classificatore vota per una classe e il modello viene assegnato alla classe più votata.

Nel **Rank** ogni classificatore restituisce una classifica (un ranking) delle classi in base alla probabilità che il pattern appartenente a ciascuno di esse. E' possibile utilizzare il **Borda Count**, dove la classifica viene poi convertita in punteggi che vengono sommati; la classe con il punteggio finale più alto è quella scelta dal multi classificatore.

Nel **Measurement** ogni classificatore emette il proprio punteggio (score) di classificazione per il modello rispetto a ciascuna classe. L'unico problema che dobbiamo risolvere è avere una misurazione dei valori che rientri nello stesso intervallo, quindi dobbiamo applicare la normalizzazione. Un metodo banale è solo quello di sommare i punteggi e passare il risultato alla regola della fusione. C'è solo un problema che in qualche modo impedisce la piena flessibilità di questo approccio, ovvero la possibilità di modificare il tipo di soglia che sfruttiamo per ottenere il risultato finale. Sono possibili diversi metodi, tra cui somma, somma ponderata, media, prodotto, prodotto ponderato, max, min, ecc.

L'ultimo livello di fusione possibile è rappresentato dal **Decision Level Fusion**. In questo caso arriviamo alla decisione finale. Ad esempio, in caso di verifica potremmo avere una risposta Sì o No da ciascuno dei sistemi coinvolti e quindi questi vengono fusi utilizzando una politica di voto a maggioranza (**majority voting**), oppure possiamo utilizzare l'AND o l'OR dei risultati (e.g. nel caso dell'and ritorniamo sì se tutti ritornano sì). Il punto debole di questo approccio è che ogni volta che qualcosa va storto, perdiamo tutte le informazioni che potrebbero essere utili per valutare il comportamento del sistema.

Negli approcci multibiometrici, quando vogliamo fondere i risultati, quali problemi possiamo riscontrare?

I **punteggi di diversi matcher sono tipicamente non omogenei**: Somiglianza/distanza; Range diversi (es. [0,1] o [0,100]) diverse distribuzioni.

Per avere una fusione che sia consistente si possono applicare trasformazioni del punteggio mediante la **normalizzazione**. Gli aspetti da considerare quando si sceglie un metodo di normalizzazione sono:

- **Robustezza**: nel senso che non deve essere influenzata da valori anomali (outliers);
- **Efficacia**: nel senso che quella distribuzione di punteggi normalizzati deve presentare la stessa forma e lo stesso andamento della distribuzione dei punteggi originari;

Un altro problema è dato dall'**affidabilità dei diversi sistemi** che può essere utilizzata come una sorta di fattore di ponderazione (una possibile soluzione alla stima dell'affidabilità è rappresentata dai margini di confidenza).