**Relazione Computer Vision**


1. Introduzione
2. Obiettivi
3. Requisiti
4. Tecnologie utilizzate
5. Implementazione
    - 5.1 Riconoscimento del volto e degli occhi
    - 5.2 Riconoscimento dei landmarks della mano
        - 5.2.1 Regolazione del volume
        - 5.2.2 Blocco della riproduzione
6. Validazione funzionale
7. Conclusioni e sviluppi futuri

# Relazione Computer Vision

## 1. Introduzione

Negli ultimi anni la necessità di applicazioni accessibili ad un numero sempre maggiore di persone ha portato allo sviluppo di nuove forme di interazioni che si inseriscono in un quadro di sviluppo molto ampio denominato "interazione multimodale". L'intento di questo progetto è quello di sfruttare l'*image processing* e la *computer vision* al fine di realizzare un'applicazione che riconosca i gesti di un utente per controllare un'applicazione multimediale senza l'utilizzo di dispositivi come mouse o tastiera. In concreto, attraverso l'utilizzo di una semplice webcam sul proprio computer si può controllare la riproduzione multimediale di un video. L'applicazione multimediale di riferimento è VLC e le funzionalità, che verranno illustrate in seguito, spaziano dall'interruzione della riproduzione quando il volto dell'utente non è più rilevato all'interno dello spazio di visualizzazione della webcam fino alla modifica del volume attraverso l'utilizzo della mano. Nei prossimi capitoli verranno illustrati a fondo gli obiettivi prefissati, le funzionalità principali che si è deciso di implementare, i requisiti e infine l'implementazione vera e propria con tutte le scelte effettuate.

## 2. Obiettivi

Partendo da quello che volevamo fosse il risultato finale del nostro progetto, ovvero un'applicazione che permettesse ad un utente di controllare un video riprodotto su VLC attraverso delle "gestures" abbiamo definito i seguenti macro obiettivi per raggiungere il nostro scopo, ai quali successivamente si sono aggiunti ulteriori micro obiettivi che vedremo meglio nella fase di implementazione:

- **Riconoscimento del volto:** in primo luogo abbiamo ritenuto necessario il riconoscimento della presenza di un volto all'interno di ogni frame video, in quanto una delle funzionalità che si è deciso di implementare riguarda l'interruzione automatica del video qualora il volto dell'utente non venga rilevato. Questo perché si è immaginato che un'applicazione del genere debba tenere conto del comportamento dell'utente e debba reagire di conseguenza, anche se l'utente non ha impartito esplicitamente alcun comando. Ad esempio, se l'utente è distratto o improvvisamente qualcuno entra nella sua stanza e si gira per comunicare, il video si interrompe e riprenderà una volta che l'utente sarà di nuovo disponibile.
- **Riconoscimento degli occhi:** successivamente abbiamo ritenuto necessario il riconoscimento dell'apertura o chiusura degli occhi in quanto, qualora l'utente fosse stanco e chiudesse gli occhi, il video non avrebbe motivo di proseguire nella sua riproduzione.
- **Riconoscimento della mano:** infine abbiamo ritenuto necessario il riconoscimento della mano, in quanto strumento attivo dell'interazione tra utente e sistema. In particolare, oltre al riconoscimento della forma completa della mano al fine di permettere all'utente di mettere in pausa la riproduzione e riavviarla (tramite una gesture che verrà illustrata in seguito), abbiamo ritenuto necessario individuare le dita per sviluppare un meccanismo di controllo del volume di riproduzione.

## 3. Requisiti

Oltre a disporre di una webcam, che costituisce un requisito fondamentale per il funzionamento dell'applicazione, è necessario che l'utente sia all'interno di una stanza adeguatamente illuminata evitando una fonte di illuminazione alle spalle, ma piuttosto favorendone una frontale che permetta di avere un giusto livello di contrasto tra l'utente e lo sfondo. E' inoltre necessario che l'utente sia ben inserito all'interno del frame, ad una distanza di circa mezzo metro dallo schermo, e che abbia una mano libera per poter effettuare le *gestures* di controllo della riproduzione video e del volume. Inoltre, per il corretto funzionamento dell'applicazione, è necessario aver installato *VLC Media Player*.

# 4. Tecnologie utilizzate

Per la realizzazione del progetto è stato utilizzato ***Python 3.10*** sfruttando le seguenti librerie:

- ***Numpy*** (*Numerical Python*): libreria creata da Travis Oliphant nel 2005 e utilizzata per lavorare con gli array. Ha anche funzioni per lavorare nel dominio dell'algebra lineare, della trasformata di Fourier e delle matrici.
- ***Matplotlib***: è una libreria per la creazione di grafici per il linguaggio di programmazione Python e la libreria matematica NumPy. Fornisce API orientate agli oggetti che permettono di inserire grafici all'interno di applicativi usando toolkit GUI generici. Nel nostro caso la libreria è utilizzata per mostrare a schermo il video catturato dalla webcam dell'utente.
- ***OpenCV***: è una libreria software multipiattaforma nell'ambito della visione artificiale in tempo reale. Per il nostro progetto abbiamo utilizzato la versione *Python* per l'elaborazione dei singoli frame.
  - **Classificatore *Haar***: tecnica *tree-based* ed è usata da *OpenCV* per effettuare la *face detection*. Va bene per oggetti rigidi che non si deformano. Usa *wavelet* come feature, collegate a bordi, linee e centri. Nel nostro progetto, utilizziamo i dati già calcolati forniti col materiale del corso per la *face detection* e la *eyes detection*.
- ***MediaPipe*:** è una libreria multipiattaforma sviluppata da Google che fornisce soluzioni di machine learning pronte all'utilizzo nell'ambito della computer vision. Nel nostro progetto è stato utilizzato per il riconoscimento della mano e delle dita grazie ad operazioni di machine learning che permettono di rilevare per ogni singolo frame 21 punti di riferimento 3D all'interno della mano.
- ***Python-vlc***: *binding* per *Python* per l'uso di ***VLC Media Player***, il software di *media player* scelto per il progetto perché open source e cross-platform.

  **Nota**: per poter utilizzare il modulo *vlc* in *Python*, è necessario che l'utente abbia installato *VLC Media Player* sulla propria macchina

# 5. Implementazione

## *5.1 Riconoscimento del volto e degli occhi*

Nella prima parte dello sviluppo dell'applicazione ci siamo occupati del riconoscimento del volto e degli occhi e per fare ciò abbiamo utilizzato due classificatori di tipo *Haar Cascade*. A fronte di un'equalizzazione di ogni singolo frame per fornire al classificatore delle immagini classificabili con maggiore semplicità, il meccanismo di controllo del player video è molto semplice. Per fare ciò è stata realizzata la classe ***Detector***, che contiene le funzioni necessarie al riconoscimento della faccia e degli occhi.

```
1  def detect_face(self):
2          self.faces = self.face_cascade.detectMultiScale(self.videocapture.frame_equalized,
   scaleFactor=1.3, minSize=(144, 80))
3
4  def detect_eyes(self):
5      for (x, y, w, h) in self.faces:
6          current_face = self.videocapture.frame_equalized[y:y + h, x:x + w]
7          self.eyes = self.eye_cascade.detectMultiScale(current_face)
```

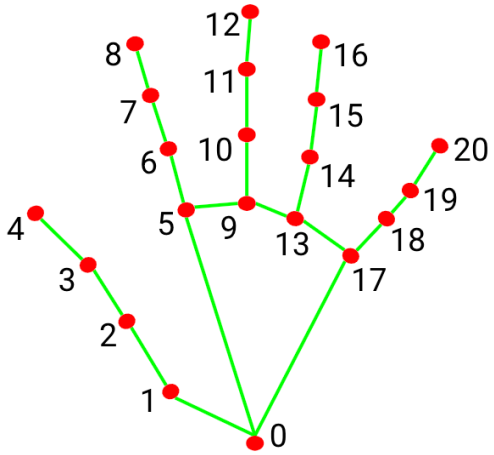
Se il volto viene rilevato e gli occhi sono aperti allora la riproduzione prosegue normalmente, altrimenti, se gli occhi (aperti) o l'intero viso non vengono rilevati per un numero prefissato di secondi, viene chiamata la funzione `player.pause()` e il player viene appunto messo in pausa. Il video viene automaticamente fatto ripartire quando vengono rilevati il volto e gli occhi aperti con la funzione `player.play()`.

Il numero di secondi di attesa prima che il video venga messo in pausa, qualora non vengano rilevati gli elementi descritti, è stato stabilito facendo alcune considerazioni ed effettuando delle prove. Da una parte abbiamo considerato alcuni comportamenti comuni degli utenti e alcuni fattori di distrazione, come ad esempio guardare le notifiche del telefono, guardare l'orologio o semplicemente dare un'occhiata all'ambiente circostante. Per questo motivo abbiamo ritenuto che

un'applicazione che rispondesse immediatamente alla non rilevazione degli elementi discussi precedentemente fosse sgradevole per l'utente e piuttosto abbiamo preferito un'applicazione flessibile. D'altra parte abbiamo considerato i limiti dei classificatori utilizzati e dell'ambiente in cui l'utente è inserito e, di conseguenza, la possibilità che per una breve sequenza di frame gli elementi necessari alla riproduzione del video non vengano individuati correttamente nonostante siano presenti. In definitiva, abbiamo optato per un periodo di 2 secondi.

## *5.2 Riconoscimento dei landmarks della mano*

Nella seconda parte dello sviluppo ci siamo occupati dell'individuazione della mano al fine di controllare, attraverso delle gestures, ulteriori funzionalità della riproduzione video come il volume, oltre al play/pausa visti precedentemente. Per fare questo abbiamo inizialmente testato vari classificatori per poi renderci conto che avevamo bisogno di uno strumento più stabile. Abbiamo infine ritenuto che la soluzione più adatta al nostro bisogno fosse la libreria *Mediapipe* per il riconoscimento della mano. E' uno strumento molto veloce ed affidabile che fornisce 21 coordinate dell'articolazione della mano e sostanzialmente costituisce una buona base su cui poter lavorare per lo sviluppo delle features.




0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Una volta superato lo scoglio della ricerca di uno strumento affidabile abbiamo pensato di creare una finestra di controllo (ROI), di dimensioni ridotte all'interno del frame, che fosse una zona in cui l'utente potesse operare con la propria mano. Al di fuori di questa zona tutti i comandi impartiti tramite *gestures* non vengono elaborati dal sistema. Questo per evitare input accidentali nel caso in cui l'utente compia movimenti come grattarsi la testa, ecc...

```
ROI_size = (250, 300)
...
ROI = frame[0:ROI_size[1], 0:ROI_size[0]]
```

Per l'individuazione della mano e di *landmarks* corrispondenti alle dita è stata realizzata la classe ***HandDetector***, basata su *MediaPipe*.

### 5.2.1 Regolazione del volume

Grazie all'individuazione di coordinate dell'articolazione della mano abbiamo poi realizzato un meccanismo di controllo del volume basato sulla distanza tra due dita. In particolare abbiamo considerato la distanza tra i punti 8 (*INDEX_FINGER_TIP*) e 4 (*THUMB_TIP*) in figura, ovvero tra la punta del pollice e la punta dell'indice. L'utente può quindi aumentare o diminuire il volume della riproduzione facendo variare la distanza tra le punte di queste due dita e successivamente lasciando la finestra di controllo. Per effettuare questa operazione è però importante che la mano sia interamente contenuta all'interno della finestra di controllo. Nel pratico, verifichiamo che le coordinate di tutti i 21 punti dell'articolazione della mano siano all'interno della finestra. Qualora un punto sia fuori dalla finestra l'operazione di modifica del volume non viene fatta partire oppure viene interrotta e l'ultimo valore fornito al sistema viene salvato. Questo semplice meccanismo permette all'utente di salvare il valore desiderato semplicemente uscendo dalla finestra di controllo.

La distanza discussa in precedenza è stata poi normalizzata basandosi su un'altra distanza sperimentalmente più stabile, ovvero quella tra il punto 0 (*WRIST*) corrispondente al polso, e il punto 17 (*PINKY_MCP*), metacarpo del mignolo. In questo modo il calcolo del volume è meno suscettibile alla distanza dalla videocamera, quindi mantenendo le dita alla stessa distanza (tra loro) e allontanandoci dalla webcam il valore percentuale di volume rimane quasi costante. Per stabilire la percentuale di volume da impostare abbiamo definito un certo intervallo, corrispondente al 15% e al 150% della distanza tra i due punti di cui abbiamo parlato, oltre il quale il volume viene impostato a zero o al massimo.

```
# Calcolo la distanza tra pollice e indice
   thumb_index_distance = handDetector.findDistance(4, 8, frame)
# Calcolo la distanza tra il polso e il metacarpo del mignolo
   dist = handDetector.findDistance(0, 17, frame, False)
   minDist = dist / 100 * 15  # distanza minima per avere volume zero
   maxDist = dist / 100 * 150  # distanza minima per avere volume massimo
# Imposto il volume
   volume = np.interp(thumb_index_distance, [minDist, maxDist], [0, 100])
   player.set_volume(volume)
```

Infine abbiamo introdotto una barra all'interno dell'interfaccia che mostra in tempo reale la percentuale di volume corrispondente alla gesture, favorendone l'usabilità.

### 5.2.2 Blocco della riproduzione

# 6. Validazione funzionale

# 7. Conclusioni e sviluppi futuri

Pur essendo consapevoli di alcuni limiti dell'applicazione ci riteniamo complessivamente soddisfatti del lavoro svolto in quando l'utente è in grado, seguendo dei requisiti minimi di utilizzo, di eseguire le operazioni di controllo con estrema semplicità. Un possibile sviluppo futuro sarebbe quello di sviluppare un'interfaccia utente che si attivi ogni volta che il sistema riconosce che l'utente sta impartendo un comando al sistema e mostri graficamente l'effetto di questi comandi con elementi grafici, così da restituire un'azione di feedback all'utente. Al dì fuori di questi momenti di controllo l'interfaccia rimane ridotta ad icona e non interferisce con la visione del video. Inoltre sarebbe interessante aumentare il numero di gestures che l'utente può effettuare andando a mappare tutte le azioni possibili di controllo del riproduttore video, come il controllo della velocità di riproduzione oppure l'aggiunta di sottotitoli.