UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI ***INGEGNERIA***

Corso di Laurea triennale in ***INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE***

IMPLEMENTAZIONE SU PIATTAFORMA EMBEDDED DI ALGORITMI PER IL RICONOSCIMENTO DA SEGNALI VIDEO DI AZIONI DELLA VITA QUOTIDIANA

EMBEDDED PLATFORM IMPLEMETATION OF ALGORITHMS FOR THE RECOGNITION OF EVERYDAY ACTIONS BY VIDEO SIGNALS

Relatore: Chiar.mo/a

Prof. Ennio Gambi

Correlatore:

Dr. Linda Senigagliesi

Tesi di Laurea di:

Viola Terribili

A.A. **2022** / **2023**

“Ideo propera, Lucili mi, vivere, et singulos dies singulas vitas puta”

Seneca

Indice

Introduzione

Negli ultimi anni l'Internet of Things, che descrive la rete di oggetti fisici, ossia le "things", che hanno sensori, software e altre tecnologie integrate allo scopo di connettere e scambiare dati con altri dispositivi e sistemi su Internet, è diventata una delle tecnologie più importanti. Una notevole progressione nei metodi basati su sensori ha determinato una rapida evoluzione delle applicazioni dell'IoT per lo sviluppo di qualsiasi sistema di monitoraggio in tempo reale. Gli sviluppi nelle tecnologie dell'informazione e della comunicazione hanno portato all'uso più ampio dell'Internet of Things (IoT). Nelle moderne applicazioni sanitarie, l'utilizzo dell'IoT riunisce medici e pazienti per il monitoraggio automatizzato e intelligente delle attività quotidiane degli anziani. L'integrazione dell'IoT nel settore sanitario ha portato all'avvio di applicazioni intelligenti come l'assistenza sanitaria mobile e sistemi intelligenti di monitoraggio sanitario. Il progetto sviluppato presenta un sistema intelligente m-healthcare, Mobile Health, ovvero una branca della Digital Health che comprende l'uso di tecnologie di telecomunicazione mobile, integrate in sistemi di erogazione dell'assistenza sanitaria, basato sulla tecnologia IoT, per fornire il riconoscimento delle attività umane. Viene utilizzato un set di dati che contiene il movimento del corpo di undici volontari di profilo diverso mentre eseguono quattro differenti attività fisiche. Il processo di analisi software è stato effettuato da un sistema di reti neurali del programma MediaPipe, che rileva punti di tracking nel video in input e fornisce in output le loro coordinate. L'obiettivo è quello di fornire un sistema di monitoraggio in tempo reale per il controllo della corretta esecuzione delle attività di vita quotidiana da parte di soggetti fragili.

Per la realizzazione abbiamo utilizzato una Rapsberry Pi 4 model B, una scheda di memoria microSD, un mouse, la Raspberry Pi Camera, un router e dei cavi Ethernet.

Capitolo 1. Stato dell'arte

Di seguito vengono presentati alcuni modelli sviluppati da progettisti nei vari campi dell'Internet of Things, da cui si evince la poliedricità di questa nuova tecnologia. Nella tabella sottostante viene riportato il nome dell'autore, l'anno di pubblicazione dello studio, gli algoritmi utilizzati e i risultati ottenuti.

Autori [1]	**Anno**	**Algoritmo Utilizzato**	**Risultati Ottenuti**
Abdulhamit Subasi, Mariam Radhwan, Rabea Kurdi, Kholoud Khateeb	2018	Artificial Neural Networks (ANN), K-Nearest Neighbors (k-NN), Support Vector Machine (SVM), C4.5 (J48), Decision Tree, CART (classification and regression tree), Random Forests (RF), Rotation Forest (RoF).	I risultati sperimentali sono stati valutati accuratamente dopo aver utilizzato le diverse tecniche di data mining. Random Forest e SVM hanno ottenuto risultati molto simili, con una buona precisione. Queste sono le tecniche che sembrano essere le più performanti in relazione ai 10 soggetti presi in considerazione. L'accuratezza di avarage dei due algoritmi è pari al 99.89%. I risultati dimostrano anche che Random Forest è più veloce di SVM. **Dataset**: MHEALTH **Accuracy**: SVM e Random Forest superano le otto tecniche di data mining utilizzate. L' accuratezza media di Random Forest e SVM è uguale al 99.89%. **Real Time**: NO

Autori [2]	**Anno**	**Algoritmo Utilizzato**	**Risultati Ottenuti**
Prayag Bhatia, Shakti Rajput, Shashank Pathak, Shivam Prasad	2018	HOG, Histogram of Oriented Gradients LBPH, Local Binary Pattern Histogram	Il volto della persona viene rilevato e riconosciuto in tempo reale, una volta che il confronto da esito positivo, la porta è sbloccata. In condizioni normali la precisione è del 90%, mentre in condizioni di scarsa illuminazione la precisione è ridotta all'80%. **Accuracy**: Condizioni Normali 90%, Condizioni Degradate 80%. **Real Time**: SI
Autori [3]	**Anno**	**Algoritmo Utilizzato**	**Risultati Ottenuti**
Navita, Pooja Mitta	2022	SVM, K-NN, Random Forest Tree, Naive Bayes, Decision Tree	Il risultato mostra che NB, Naive Bayes, ha raggiunto una precisione media di 0.781%, 0.78%, 0.75% di richiamo e 0.76% valore di f1_score. Successivamente DT, Decision Tree, ha ottenuto una precisione media di 0.8049%, 0.79% di precisione, 0.80% di richiamo e 0.80% di punteggio f1 che è superiore a NB. Ulteriori K-NN superano sia NB che DT con una precisione media di 0.9572%, 0.94% di precisione, 0.91% di richiamo e 0.93% di punteggio f1. RFT, Random Forest Tree, ha ottenuto un risultato migliore rispetto a NB, DT e K-NN con una precisione media di 0.9742%, 0.97% di precisione, 0.96% di richiamo e 0.98% di punteggio f1_score. Infine SVM ha raggiunto una precisione media dello 0.9803%, 0.97% di precisione, 0.96% di richiamo e 0.95% di

			punteggio f1 che è il più altro tra tutti gli algoritmi di classificazione. **Dataset**: Il set di dati è preso dalla fonte di dati di Kaggle. **Accuracy**: il risultato mostra che tra tutti gli algoritmi, SVM ha raggiunto la massima precisione che è del 98.03%. **Real Time**: NO

In *"IoT based mobile Healthcare System for Human Activity Recogniton"* [1] è presentato un sistema intelligente m-health basato sulla tecnologia IoT per fornire il riconoscimento dell'attività umana basato sull'uso di tecniche di data mining. Viene presentato un modello di riconoscimento delle attività umane preciso e robusto. Il modello in questione fa uso di un set di dati che contiene il movimento e i segni vitali di 10 volontari con profili diversi mentre svolgono 12 diverse attività fisiche. Il dataset utilizzato, MHEALTH, è stato costruito utilizzando sensori posizionati sulla caviglia sinistra del soggetto, sul polso e sul petto.

L'obiettivo che viene raggiunto nell'articolo *"IoT based facial recognition system for home security using LBPH algorithm"* [2] è l'ottenimento di un sistema di sicurezza domestica semplice e infallibile. Il sistema fa uso di istogrammi binari locali per il riconoscimento delle persone facendo un confronto con le immagini delle facce nel data-set. L'hardware utilizzato per l'implementazione di questo tipo di sistema sono il Raspberry Pi 3, una webcam esterna, un autoportante e un motore passo-passo.

Navita e Pooja Mittal [3], sviluppano un modello di monitoraggio delle attività umane basato su IoT. Lo scopo è quello di monitorare le attività degli anziani attraverso sensori intelligenti. Nel documento in esame i dati vitali sono raccolti attraverso sensori e vengono analizzati utilizzando diversi algoritmi per il rilevamento di qualsiasi rischio nel comportamento dell'attività umana.

Capitolo 2. Materiali e Metodi

In questo capitolo vengono illustrati gli strumenti e le tecnologie utilizzate per l'implementazione del progetto.

2.1 Raspberry Pi

2.1.1 Descrizione

Il Raspberry Pi [4] è un single board computer, ovvero computer a scheda singola, cioè una scheda elettronica che implementa un intero computer o quasi, caratterizzato da un processore ARM con un fattore di forma estremamente ridotto, pari a quello di una carta di credito, che integra varie funzionalità utili per applicazioni domestiche e industriali.

È stato sviluppato dalla Raspberry Pi Foundation, un'organizzazione di beneficenza britannica, come strumento per promuovere lo studio dell'informatica nelle scuole e nei paesi in via di sviluppo. Non viene offerta la potenza di un computer desktop, ma può essere impiegato in tutte quelle attività che non necessitano delle prestazioni di un PC costoso e ingombrante. Ad un prezzo irrisorio, questo single board computer dalle dimensioni di una carta di credito offre un processore ARM, RAM, funzionalità grafiche e tutte le porte hardware standard che si trovano di solito in un computer.

Il nome Raspberry, in italiano "lampone", è un omaggio alle prime aziende di computer, il cui nome prendeva spunti da nomi di frutti, come Apple, Tangerine, Computer System, Apricot Computers e Acorn. La sigla "Pi" fa riferimento al linguaggio di programmazione Python, adottato come primo linguaggio di programmazione da utilizzare su questo single board computer, che in realtà supporta linguaggi come C/C++, Java e molti altri.

Le ragioni per cui Raspberry Pi ha riscosso un enorme successo in giro per il mondo sono riconducibili essenzialmente al suo costo contenuto, le sue dimensione ridotte e il bassissimo consumo di potenza elettrica.

La guida rapida per Pi consiglia una scheda micro-SD con velocità di scrittura minima di classe 4. Oltre alla micro-SD, è necessario procurarsi tutto ciò che si desidera connettere alla scheda, poiché nulla è incluso nel pacchetto. Qui di seguito elenco l'essenziale per un utilizzo generico:

• Scheda micro-SD;

• Alimentatore;

• Cavo per connessione a schermo, un conduttore da HDMI a HDMI/DVI per connettere il Pi alla propria TV o monitor;

• Tastiera, Mouse e Monitor/TV;

• Cavo Ethernet (solo modello B/B+), se per qualche motivo non si vuole utilizzare il WiFi incorporato nei modelli più recenti.

Non è necessario alcun monitor dedicato. Una volta completate le impostazioni, si collegano mouse e tastiera alle porte USB del Pi e il monitor via HDMI. Ciò provoca un avvio con il sistema operativo selezionato e la corrispondente GUI, interfaccia grafica utente. Alcuni progetti non necessitano che il Raspberry Pi utilizzi una normale GUI, in questi casi si può collegare il Pi alla rete e accedervi da remoto tramite il protocollo SHH (Secure Shell).

2.1.2 Architettura

I componenti che costituiscono una Raspberry Pi, esempio in *Figura 2.1*, sono:

• 1.2GHz 64-bit quad-core ARMv8 CPU.

• 4 porte USB 2.0;

• 1 uscita HDMI;

• 1 uscita combinata di 3.5 mm per audio e video composito;

• 1 slot per Micro SD Card;

• 1 porta micro USB per l'alimentazione;

• 1 porta Ethernet;

• 802.11n Wireless LAN;

• Bluetooth 4.1;

• Bluetooth Low Energy (BLE);

• 40 GPIO pin;

• Camera Interface (CSI);

• Display Interface (DSI);

•VideoCore IV 3D graphics core.

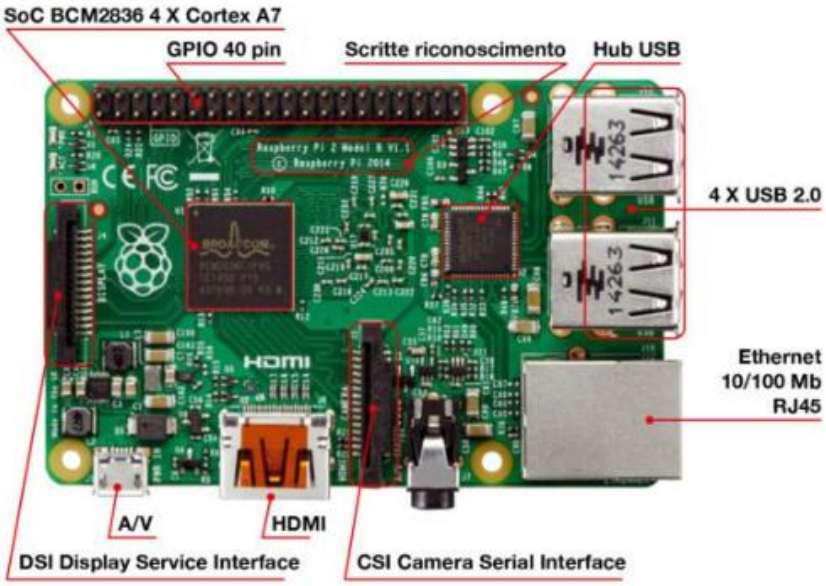


Figura 2.1 *Architettura di un Raspberry Pi 4*

2.1.3 Implementazione Raspberry Pi

Per la realizzazione del progetto abbiamo utilizzato una Rapsberry Pi 4 model B, una scheda di memoria microSD, un mouse, cavi HDMI, la Raspberry Pi Camera, e il cavo Ethernet. Il cavo HDMI varia a seconda del Raspberry Pi in nostro possesso. Il Raspberry Pi 4B ha due uscite micro-HDMI, quindi, richiede cavi da micro HDMI AD HDMI o adattatori.

Inoltre abbiamo effettuato l'installazione e la configurazione della Raspberry Pi Camera. *Figura 2.2*.

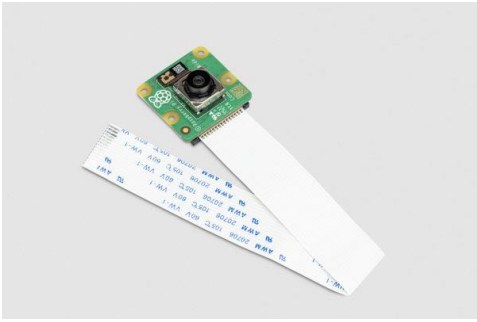


Figura 2.2 Raspberry Pi Cam

Una volta collegata fisicamente la Raspberry Pi Cam possiamo accendere la nostra board Raspberry. Come mostrato nella *Figura 2.3*, si va nel menu principale e si apre lo strumento di configurazione di Raspberry Pi, si seleziona la sezione *Interfaces* e ci si assicura che la telecamera sia abilitata, come si può vedere in *Figura 2.4*. Infine riavviamo il sistema. Fatto questo, si è passati alla configurazione del sistema operativo sulla Raspberry.

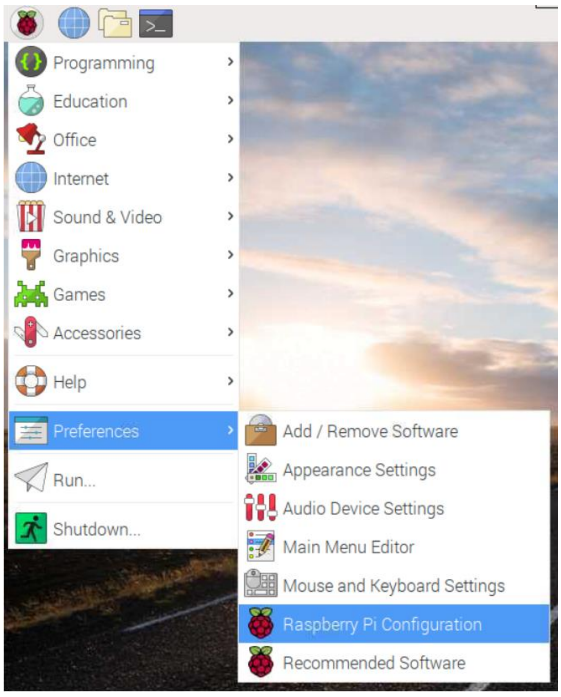


Figura 2.3 Menù principale della Raspberry Pi

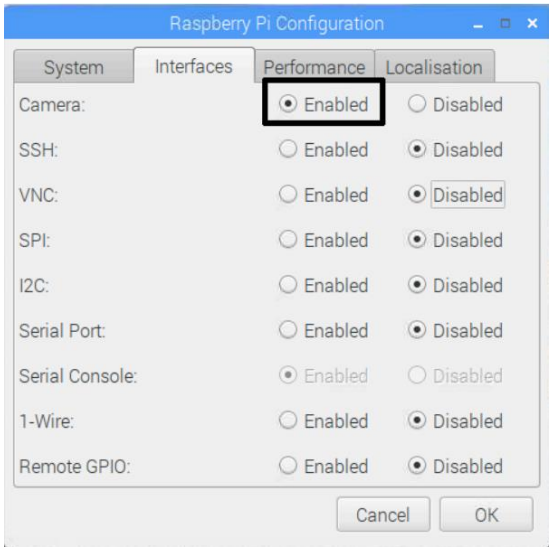


Figura 2.4 Raspberry Pi Configuration

Procediamo con la configurazione dell' OS Raspberry su microSD, assicurandoci che la scheda abbia almeno 8 GB di spazio, senza l'ausilio di periferiche attaccate al Raspberry Pi.

Scarichiamo Raspbian OS. Si può ottenere l'ultima versione di Raspbian su *raspberrypi.org*. Si può scegliere tra la versione lite, che garantisce solo il prompt dei comandi e la versione desktop, con interfaccia grafica completa.

Raspberry Pi Imager è il modo più semplice e veloce per installare Raspberry Pi OS e altri sistemi operativi su una scheda microSD pronta per l'uso con il Raspberry Pi.

Se abbiamo intenzione di accedere da remoto al Raspberry Pi tramite una connessione SSH, non avremo bisogno di monitor, mouse e tastiera da collegare alla board.

Abbiamo scaricato sulla nostra Raspberry Pi 'miniconda'. Miniconda, è un'alternativa più leggera di 'Anaconda', la quale porta in automatico tutti i pacchetti necessari per il lavoro scientifico. Conda [5] è un package manager open source che gira su Windows, macOS e Linux; ti consente di installare, eseguire e aggiornare rapidamente dei pacchetti e le relative dipendenze, di creare, salvare, caricare e passare facilmente da un ambiente ad un altro sul tuo computer. È stato creato per Python, ma può essere usato per qualsiasi linguaggio.

Dopo averlo configurato, apriamo il terminale, aggiorniamo il gestore dei pacchetti e controlliamo se una versione di Python è installata nella board.

I comandi che usiamo per aggiornare sono i seguenti

```
labtlc@raspberrypi: ~ $ sudo apt update
labtlc@raspberrypi: ~ $ sudo apt upgrade
```

Per verificare se è già presente una versione di Python, scriveremo

```
labtlc@raspberrypi: ~ $ python --version
```

Per scaricare Miniconda abbiamo usato il seguente link

```
labtlc@raspberrypi: ~ $ wget http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-aarch64.sh
```

2.2 Tecniche di Intelligenza Artificiale

2.2.1 Face Detection e Face Recognition

Il Face Detection [6], rilevamento dei volti, è un componente dei sistemi di riconoscimento facciale, Facial Recognition, e insieme rappresentano metodi non intrusivi per identificare e riconoscere le persone. Lo sviluppo di approcci biometrici come il riconoscimento facciale, si rivela un'ottima soluzione per identificare gli individui.

Il Riconoscimento Facciale. Il sistema di riconoscimento facciale, Facial Recognition, è una tecnologia in grado di indentificare e confrontare l'identità di una persona da un'immagine digitale o da un database di volti. Il sistema utilizza la biometria per mappare le caratteristiche facciali che sono uniche per un individuo. Si tratta perciò di un sistema che misura una varietà di caratteristiche facciali chiamate punti di riferimento o punti nodali sul viso. Questi possono includere la distanza tra gli occhi, la larghezza del naso, la profondità delle orbite e altro ancora.

Il Rilevamento dei Volti. Il rilevamento dei volti, Face Detection, è una tecnologia informatica basata sull'intelligenza artificiale utilizzata per identificare e localizzare i volti umani in immagini e video digitali. Identifica semplicemente se è presente un volto umano all'interno di un'immagine o di un video, ma non può identificare la persona. Il rilevamento del viso è un componente dei sistemi di riconoscimento facciale utilizzati per localizzare ed estrarre la regione del viso dallo sfondo. La *Figura 2.5* mostra un esempio di sistema di rilevamento facciale.

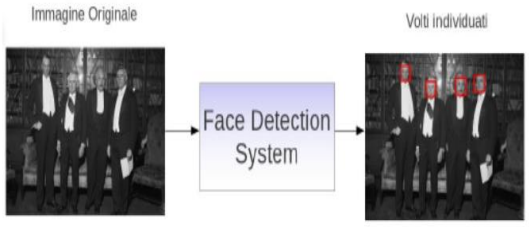


Figura 2.5 Esempio di sistema di localizzazione di volti

2.3 Tecniche di Estrazione delle Feature e Tecniche di Riduzione di Dimensionalità

Nel caso del riconoscimento del volto possono essere estrapolate le informazioni degli aspetti di maggior rilevanza del viso, occhi, naso, bocca, fronte, lineamenti, sulle quali viene costruito un modello matematico di rappresentazione. Tale insieme ridotto di informazioni può essere ottenuto applicando, singolarmente o in combinazione, tecniche di estrazione di feature e tecniche di riduzione di dimensionalità.

2.3.1 Riduzione di Dimensionalità

Gli obiettivi dei metodi per la riduzione di dimensionalità è quello di eseguire un mapping dello spazio iniziale ad uno di ambiente inferiore. Le principali tecniche sono, PCA Principal Component Analysis, LDA, Linear Discriminant Analysis, ICA, Indipendent Component Analysis.

2.3.2 Riconoscimento delle Feature

L'ultima fase del processo di Face Recognition è il riconoscimento delle feature. Questa è una fase che sia avvale di algoritmi di riconoscimento complessi e di una conoscenza acquisita durante l'addestramento del classificatore, sulla base di un insieme di dati detto training set. Il training set è un insieme di dati usato per addestrare un sistema supervisionato, in modo tale che il sistema ad ogni test dell'addestramento apprenda la capacità di determinare le differenti categorie di appartenenza dei dati in base alle loro caratteristiche (feature).

Nel nostro progetto abbiamo utilizzato uno dei migliori algoritmi di machine learning più utilizzati e con le migliori prestazioni nell'ambito del riconoscimento facciale, l'algoritmo SVM, Support Vector Machine.

Si tratta di un'efficace tecnica di apprendimento automatico in grado di gestire set di dati molto complessi. Grazie alla sua capacità di estrarre proprietà intrinseche di dataset dimensionali superiori, è ampiamente utilizzato in molte applicazioni, come la computer vision, la biometria e il riconoscimento vocale. Nel campo del riconoscimento facciale, SVM ha dimostrato di essere superiore ad altre tecniche in termini di accuratezza e

robustezza. SVM funziona bene con piccoli set di dati, ed è anche in grado di gestire dati rumorosi. SVM è estremamente efficiente, il che significa che può elaborare grandi quantità di dati rapidamente. Questo lo rende adatto per applicazioni in tempo reale.

2.3.3 scikit-learn

scikit-learn [7], conosciuto anche come sklearn, è un libreria di machine learning per il linguaggio di programmazione Python. Presenta molti algoritmi di classificazione, regressione e clustering, tra cui SVM, Support Vector Machine, Random Forest e altri ancora, è anche progettato per interagire con le librerie numeriche e scientifiche Python NumPY e SciPy

2.4 MediaPipe

MediaPipe [8] è un framework open-source per la costruzione di pipeline per eseguire l'inferenza della visione computerizzata su dati sensoriali arbitrari come video o audio.

2.4.1 MediaPipe Face Landmarker

MediaPipe è una piattaforma OpenSource di Google che permette l'applicazione della teoria del deep learning per l'acquisizione e l'assegnazione dei punti di tracking.

L'attività MediaPipe Face Landmarker consente di rilevare punti di riferimento e espressioni facciali in immagini e video. È possibile utilizzare questa attività per identificare le espressioni facciali umane, applicare filtri ed effetti facciali e creare avatar virtuali.

Rileva il volto più prominente da un'immagine di input, quindi stima 478 punti di riferimento facciali 3D e 52 punteggi di modellazione facciale in tempo reale. Questa soluzione può essere utilizzata per creare un'esperienza di prova virtuale o un avatar virtuale che imita le espressioni facciali di una persona.

2.5 OpenCV

OpenCV [9], Open Source Computer Vision Library, è una libreria software multipiattaforma nell'ambito della visione artificiale in tempo reale. È ricco di librerie per Computer Vision e Machine Learning.

Usare Haar Cascade in OpenCV e Python risulta un modo semplice di rilevare i volti.

Come prima cosa è bene assicurarsi che sia installato OpenCV, in caso di risposta negativa è possibile farlo attraverso il comando, da digitare nel terminale:

```
labtlc@raspberrypi: ~ $ pip install opencv-python
```

Il rilevamento facciale con Haar Cascade è un approccio basato sull'apprendimento automatico in cui una funzione a cascata viene addestrata con una serie di dati in input. OpenCV contiene molti classificatori pre-formati per viso, occhi, sorrisi, ecc.

Al fine del rilevamento facciale con Haar Cascade e OpenCV possono essere utilizzati diversi codici, tra cui quello elencato in esempio, *Figura 2.6*:

```python
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Read the input image
img = cv2.imread('test.jpg')
# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Detect faces
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

Figura 2.6 Codice con OpenCV

Il rilevamento funziona solo su immagini in scala di grigi. Quindi è importante convertire l'immagine a colori in scala di grigi. *(Linea 8)*

detectMultiScale (Linea 10) viene utilizzato per rilevare i volti. Vengono richiesti tre argomenti, l'immagine di ingresso, scaleFactor e minNeighbours. *scaleFactor* specifica quanto la dimensione dell'immagine viene ridotta con ogni scala.

faces contiene un elenco di coordinate per le regioni rettangolari dove sono state trovate le facce. Queste coordinate vengono usate per disegnare i rettangoli nella nostra immagine.

Capitolo 3. Sviluppo del progetto

Gli elementi illustrati nei capitoli precedenti ci offrono una solida base per lo sviluppo del nostro progetto. L'obiettivo dell'elaborato è quello di implementare un sistema in grado di riconoscere, in real-time, le azioni di vita quotidiana che vengono svolte dal soggetto ripreso dalla Picamera.

3.1 Programmazione

Per la programmazione abbiamo utilizzato Python e le librerie precedentemente installate. Abbiamo creato un programma principale Python, *SVMraspberry.py*, per riconoscere le azioni in tempo reale con l'input video.

3.1.1 Struttura della Directory

La directory deve essere bene sistemata per assicurarci di avere un programma ordinato. Si procede perciò con la creazione di una directory di progetto nella cartella home di Raspberry Pi. Tutti i progetti che verranno creati saranno salvati nella cartella del progetto.

Per la creazione della directory si procede inserendo da terminale i seguenti comandi:

```
labtlc@raspberrypi: ~ $ cd

labtlc@raspberrypi: ~ $ mkdir project

labtlc@raspberrypi: ~ $ cd project

labtlc@raspberrypi: ~ $ mkdir face-detection

labtlc@raspberrypi: ~ $ cd face-detection
```

Successivamente abbiamo lavorato nella directory *~/project/face-detection.*

3.1.2 Codice Sorgente

Generiamo un file, con il nome SVMraspberry.py. Per generare un file da terminale è necessario inserire la parola chiave *nano*, mentre per far partire il file scriveremo, sempre da terminale la parola *python*.

```
labtlc@raspberrypi: ~ $ nano SVMraspberry.py
```

```
labtlc@raspberrypi: ~ $ python SVMraspberry.py
```

3.1.3 Creazione del Data-Set

La realizzazione di algoritmi di intelligenza artificiale è possibile solo se si dispone di un set di dati, data-set, che rende possibile l'addestramento del software. Lo sviluppo di un algoritmo di intelligenza artificiale utilizza un data-set di addestramento. Più è complicata l'attività da svolgere, più informazioni sono necessarie. La qualità dei dati disponibili determina le prestazioni dei sistemi di apprendimento automatico. Il nostro data-set è stato creato grazie all'aiuto di undici volontari, che svolgendo le 4 azioni di nostro interesse, *'no_action'*, *'washing_your_face'*, *'brushing_your_teeth'*, *'combing_your_hair'*, ci hanno permesso di allenare il set di dati, così da avere dei risultati migliori e più accurati. Di seguito sono riportate le immagini di cosa appare nel momento in cui si alimenta il set di dati, *Figura 3.1, Figura 3.2, Figura 3.3 e Figura 3.4.*

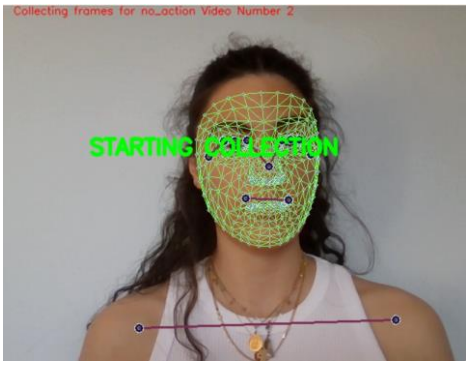


Figura 3.1 Collecting frames "no_actions"

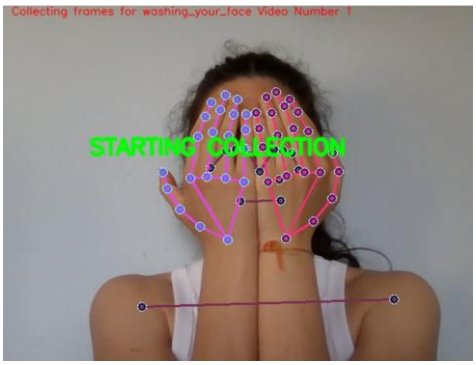


Figura 3.2 Collecting frames "washing_your_face"

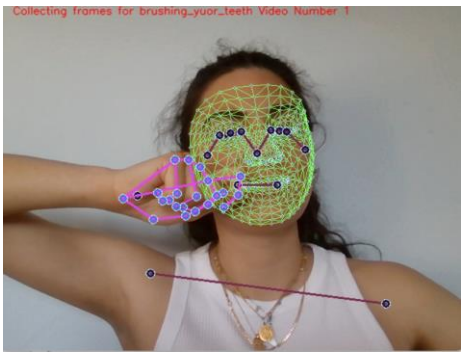


Figura 3.3 Collecting frames "brushing_your_teeth"

Figura 3.4 Collecting frames "combing_your_hair"

3.2 Sistema di Coordinate

Per l'implementazione del codice abbiamo fatto largo uso della libreria MediaPipe che offre strumenti efficaci per poter mappare il volto, il corpo e le mani.

Le informazioni sono elaborate numericamente sugli assi x, y e z. In MediaPipe ogni landmark viene memorizzato come un contenitore che a seconda del tipo di cattura assume un nome diverso:

- FACE_LANDMARK, per il viso.
- HAND_LANDMARK, per la mano.

Il landmark è caratterizzato dalla presenza di 4 elementi:

- x: indica lo spostamento del landmark in orizzontale.
- y: rappresenta la posizione in verticale del punto di riferimento.
- z: denota la profondità del landmark, per il viso, il centro geometrico approssimativo della testa e per la mano, il centro geometrico approssimativo della mano.
- visibility: valore che indica se il punto è visibile o nascosto, ovvero occluso da un'altra parte del corpo, su un frame. Il numero di punti chiave estratto dalla stima della posa si calcola nel seguente modo: *punti chiave in posa x (tre dimensioni + visibilità)*.

3.2.1 FACE MESH PIPELINE

Per la ricostruzione della mesh facciale si usano due sistemi di rete neurale che lavorano in real-time, il Face Detection model, un rilevatore che lavora sull'immagine andando a calcolare le posizioni dei volti, e Face Landmark model, un modello di riferimento 3D che opera sulle posizioni rilevate fornendo una stima della geometria facciale. *Figura 3.5 e Figura 3.6* mostrano esempi di Face Landmarks.

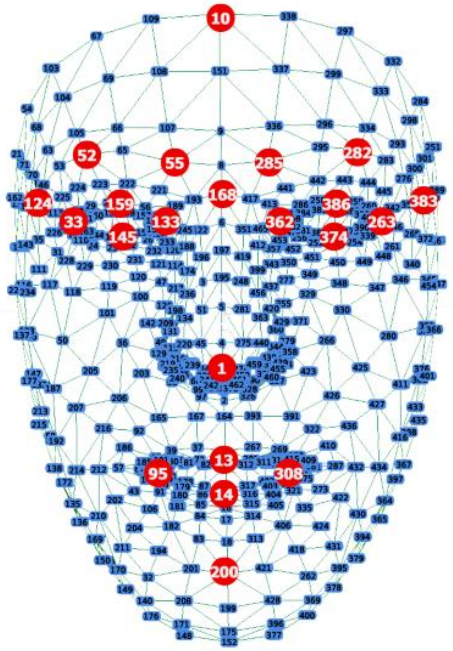


Figura 3.5 Face Landmarks

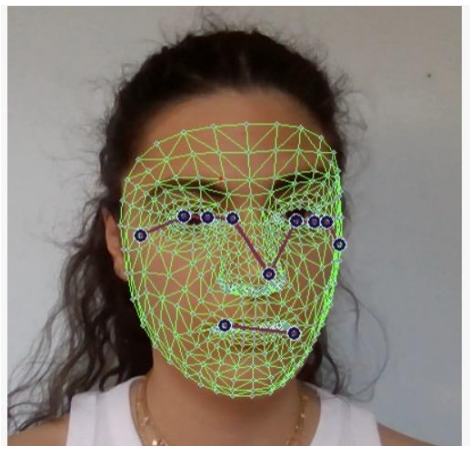
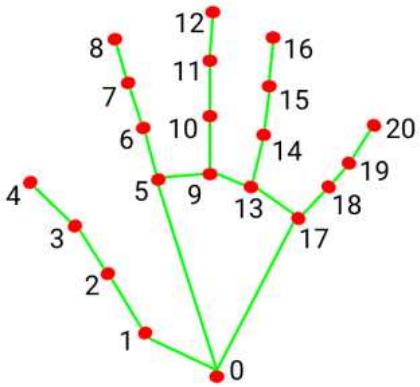


Figura 3.6 Face Landmarks

3.2.2 HAND CAPTURE

L'Hand Capture è reso possibile dalla soluzione "hands" presente in MediaPipe. I landmark di riferimento di una singola mano sono 21 e permettono un tracciamento ad alta definizione. La pipeline per il riconoscimento delle articolazioni di una mano prevede due modelli che lavorano assieme, il Palm Detection Model, che rileva il palmo di una mano, e l'Hand Landmark Model. *Figura 3.7, Figura 3.8 e Figura 3.9* esempi di Hand Landmaks [10].



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

Figura 3.7 Hand Landmarks

Figura 3.8 Hand Landmarks

Figura 3.9 Hand Landmarks

3.3 Frame

Per conoscere i frames abbiamo usato un codice che fa uso della libreria MediaPipe, *Figura 3.10.* "Un frame, nelle reti di computer e nelle telecomunicazioni, è un'unità di trasmissione di dati digitali. Un frame include caratteristiche di sincronizzazione del frame costituite da una sequenza di bit o simboli che indicano al ricevitore l'inizio e la fine dei dati del payload all'interno del flusso di simboli o bit che riceve." [11].

```
1   # CHECK IF THE CAMERA IS FUNCTIONING AND FRAMES PER SECONDS WITH MEDIAPIPE ACTIVE
2   frames = 0
3   cap = cv2.VideoCapture(0) # lo 0 è il numero della periferica
4
5   # Set mediapipe model
6   with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
7       start = time.time()
8
9       while cap.isOpened():
10
11          # Read feed
12          ret, frame = cap.read()
13          frames = frames + 1;
14          # Make detections
15          image, results = mediapipe_detection(frame, holistic)
16
17          # Draw landmarks
18          draw_styled_landmarks(image, results)
19          res = extract_keypoints(results)
20          # Show to screen
21          cv2.imshow('OpenCV Feed', image)
22
23          # Break gracefully
24          if cv2.waitKey(10) & 0xFF == ord('q'):
25              end = time.time()
26              seconds = end - start
27              print ("Time taken : {0} seconds\n".format(seconds))
28              print ("Frames taken : {0} \n".format(frames))
29
30              # Calculate frames per second
31              fps  = frames / seconds
32              print("Estimated frames per second : {0}".format(fps))
33              break
34      cap.release()
35      cv2.destroyAllWindows()
```

Figura 3.10 Codice per il calcolo FPS

Il codice di riferimento, *Figura 3.10*, verifica se la camera è attiva *(Linea 3)*, ed effettua il calcolo dei frames per second, FPS, attraverso la libreria MediaPipe, per questo motivo abbiamo bisogno di impostare il modello MediaPipe *(Linea 6)*, *(Linea 31)* mostra come vengono calcolati i frame.

Di seguito, *Figura 3.11-3.12*, riportiamo i risultati ottenuti, risultati che come possiamo notare, variano a seconda del tempo impiegato, *time taken*:

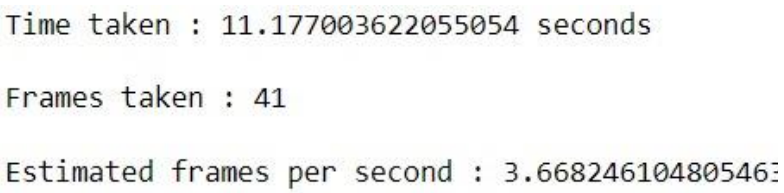


Figura 3.11 Risultati calcolo FPS

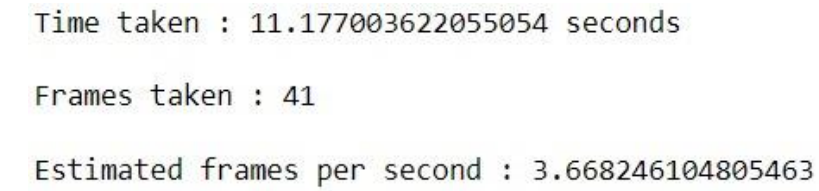


Figura 3.12 Risultati calcolo FPS

3.4 Implementazione del codice

Come prima cosa siamo partiti importando le librerie necessarie al funzionamento dell'algoritmo, *Figura 3.13,* oltre a quelle sopra menzionate, abbiamo impiegato l'uso di altre, tra cui la libreria ***time***, che contiene funzioni di manipolazione date e valori temporali, la libreria ***numpy***, che fornisce gli strumenti necessari per poter lavorare con grandi array di dati numerici, che generalmente vengono utilizzati nel machine learning. Infine menzioniamo ***joblib***, "un set di strumenti per fornire un pipeling leggero in Python. Joblib è ottimizzato per essere veloce e robusto su grandi dati e ha ottimizzazioni specifiche per array numpy" [12].

```
import joblib
from sklearn import svm
import time
import cv2
import numpy as np
import mediapipe as mp
```

Figura 3.13 Librerie importate

Si procede definendo le funzioni provenienti da MediaPipe, *Figura 3.14,*

```
mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic
```

Figura 3.14 Funzioni provenienti da MediaPipe

MediaPipe Face Detection è una soluzione di rilevamento facciale che viene fornita con 6 punti di riferimento e supporto multi-face. Si basa su BlazeFace, un leggero e ben performante face detector su misura per inferenza GPU mobile [13].

MediaPipe Holistic integra i modelli di postura, viso e mani. Viene effettuata una stima della posa umana e dei suoi punti di riferimento , che ci aiutano ad ottenere la posizione

delle mani e del volto. Note la posizione di ciascuna di queste, dette regioni di interesse, vengono ritagliate dall'immagine in ingresso per ottenere i punti di riferimento delle mani e del viso.

Per quanto riguarda le definizioni delle classi, abbiamo usato quelle fatte, in precedenza, da altri tesisti. Abbiamo poi avuto il bisogno di definirne una che permettesse l'estrazione dei punti chiave, aggiungendo una funzione in grado di andare ad estrarre i parametri, andando a rimuovere i punti di posa con scarsa visibilità, *Figura 3.15.*

```
def remove_pose_points_with_low_visibility(pose):
    new_pose = []
    face = []
    rh = []
    lh = []
    shoulders = []
    relbow = []
    lelbow = []

    e_to_e_distance = np.sqrt((pose[5,0:2]-pose[2,0:2])**2)

    for i in range(len(pose)):
        if pose[i,3]>0.4 and pose[i,0]<1 and pose[i,1]<1: #if visibility is less than 0.5 do not consider the pose
            new_pose.append(pose[i,0:2].tolist())
            if i<11:
                face.append(pose[i,0:2].tolist())
            elif i<13:
                shoulders.append(pose[i,0:2].tolist())
            elif i==13:
                lelbow.append(pose[i,0:2].tolist())
            elif i==14:
                relbow.append(pose[i,0:2].tolist())
            elif i in [15,17,19,21]:
                lh.append(pose[i,0:2].tolist())
            elif i in [16,18,20,22]:
                rh.append(pose[i,0:2].tolist())

    return np.array(new_pose),np.array(face),np.array(rh),np.array(lh),np.array(shoulders),np.array(relbow),np.array(lelbow)
```

Figura 3.15 Remove pose points with low visibility

3.5 Risultati SVM

Nelle tabelle seguenti, *Figura 3.16-3.17-3.18-3.19*, sono presenti i risultati medi di SVM fatti con una Leave One Subject Out Cross Validation, che utilizza un soggetto alla volta come test. SVM prende in input x, y del centro della testa e della mano e fa una predizione per ogni singolo frame. Due tabelle, *Figura 3.16-3.17*, si riferiscono a *precision, recall, f1_score medi e standard deviation* e due tabelle, *Figura 3.18-3.19*, con scritto CM fanno riferimento alla *Confusion Matrix media* e *standard deviation*.

AVG_METRICS	**no_actions**	**washing_face**	**brushing_teeth**	**combing_hair**
precision	0,936984745473473	0,89042961678164	0,895724026545814	0,963081749284701
recall	0,993103448275862	0,863949843260188	0,927586206896552	0,879937304075235
f1score	0,962866543650053	0,874926812909933	0,9086143616043	0,912192995705254
n°score	290	290	290	290

Figura 3.16 AVG_METRICS

STD_METRICS	**no_actions**	**washing_face**	**brushing_teeth**	**combing_hair**
precision	0,077836796092099	0,174062141115949	0,115431857063609	0,0361209311670053
recall	0,0218088114494371	0,160385194880708	0,136987367522205	0,14441513280958
f1score	0,0487513513956894	0,162984167468622	0,118700838850508	0,0922072391469072
n°score	0	0	0	0

Figura 3.17 STD_METRICS

STD_CM	**PRED_no_actions**	**PRED_washing_face**	**PRED_brushing_teeth**	**PRED_combing_hair**
TRUE_no_actions	6,32455532033676	3,73723723474445	0	2,58731808559231
TRUE_washing_face	22,0843611694897	46,5117065154054	22,2071390501656	7,94817927068887
TRUE_brushing_teeth	3,01785321321779	38,432510342673	39,7263365814395	0,987525499200019
TRUE_combing_hair	10,4391190534245	18,9007848890327	24,1335814181426	41,8803885147781

Figura 3.18 STD_CM

AVG_CM	**PRED_no_actions**	**PRED_washing_face**	**PRED_brushing_teeth**	**PRED_combing_hair**
TRUE_no_actions	288	1,18181818181818	0	0,818181818181818
TRUE_washing_face	13,0909090909091	250,545454545455	17,4545454545454	8,90909090909091
TRUE_brushing_teeth	1,72727272727273	18,8181818181818	269	0,454545454545454
TRUE_combing_hair	6,54545454545455	13,8181818181818	14,4545454545455	255,181818181818

Figura 3.19 AVG_CM

Prima di passare alla verifica e valutazione del modello implementato è bene spiegare la logica del codice. Il riconoscimento avviene sfruttando la conoscenza del punto medio del viso e della mano e la distanza che si crea nel momento in cui vengono eseguite le varie azioni; ovvero nel momento in cui si sta svolgendo l'azione del lavaggio dei denti, si viene a creare una distanza tra mano e viso inferiore rispetto al caso in cui si sta compiendo l'azione del pettinarsi i capelli. Il "*no_action"s* viene riconosciuto proprio perché non c'è interazione tra mano e viso.

3.6 Test in Real-Time

Dopo aver definito e importato tutto il necessario affinché l'algoritmo funzionasse, possiamo passare al "Test in Real-Time". Mettiamo alla prova il modello di rete neurale per valutare e verificare se è in grado di riconoscere le azioni che gli vengono fornite in ingresso. *Figura 3.20-3.21-3.22-3.23*, mostrano ciò che appare nello schermo nel momento in cui un'azione viene riconosciuta, in particolare in *Figura 3.20*, il "*no_actions"*, ovvero nessuna interazione tra mani e viso, in *Figura 3.21*, il "*washing_your_face"*, in *Figura 3.22*, il "*brushing_your_teeth"* e in *Figura 3.23* il "*combing_your_hair"*. Si evidenzia che il modello è in grado di riconoscere le azioni anche in condizioni non del tutto ottimali, in cui l'input risulta artefatto. Non è da escludere un miglioramento della precisione, andando ad aumentare il numero di epoche di addestramento.

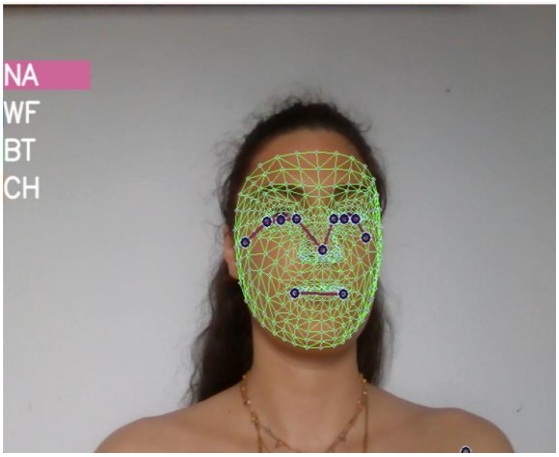


Figura 3.20 "no_actions"

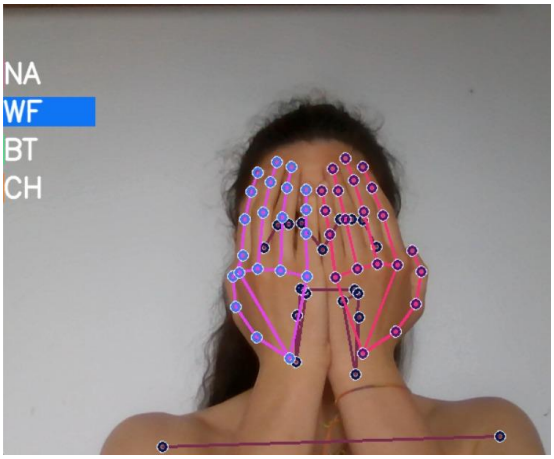


Figura 3.21 "washing_your_face"

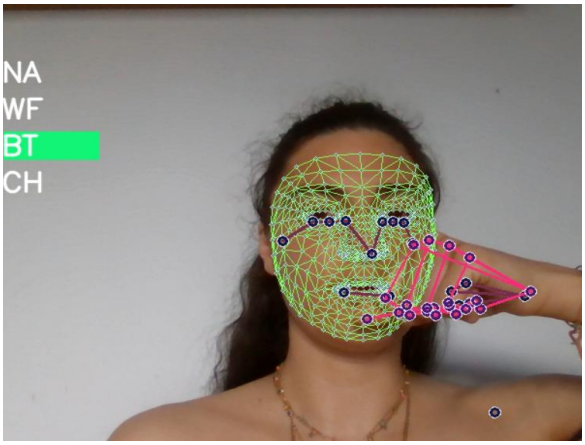


Figura 3.22 "brushing_your_teeth"

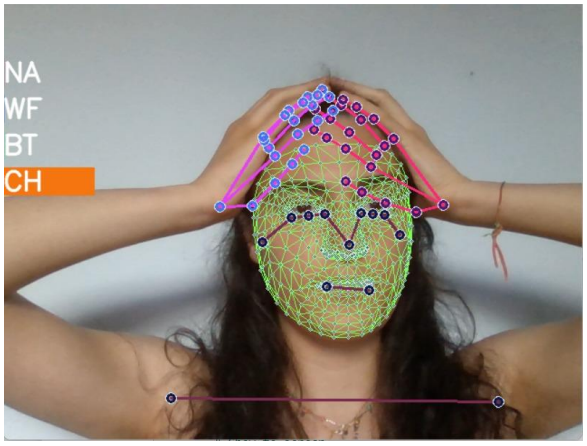


Figura 3.23 "combing_your_hair"

Capitolo 4. Conclusioni

Questo studio contribuirà allo sviluppo di un algoritmo di intelligenza artificiale nell'ambito sanitario, e nello specifico sarà di supporto agli operatori che devono controllare il corretto e regolare svolgimento delle azioni di vita quotidiana di soggetti fragili. Al giorno d'oggi il numero di persone anziane che vive in strutture di ricovero è sempre più alto e questi sistemi possono garantire una maggiore sicurezza a questi individui garantendogli una migliore qualità di vita, evitando che questi possano essere vittime di fenomeni di trascuratezza che nel lungo periodo possono sfociare in gravi conseguenze sia fisiche che mentali. Risulta chiaro ed evidente come lo sviluppo dell'intelligenza artificiale abbia cambiato il nostro stile di vita e il modo in cui lavoriamo; sebbene l'IA abbia introdotto molti benefici, bisogna essere consapevoli dei suoi limiti. L'uso responsabile di questa nuova tecnologia può aiutarci a vivere con maggior efficienza e controllo.

Bibliografia

[1] Subasi, A., Radhwan, M., Kurdi, R., & Khateeb, K. (2018, February). IoT based mobile healthcare system for human activity recognition. In *2018 15th learning and technology conference (L&T)* (pp. 29-34). IEEE.

[2] Bhatia, P., Rajput, S., Pathak, S., & Prasad, S. (2018, November). IOT based facial recognition system for home security using LBPH algorithm. In *2018 3rd International Conference on Inventive Computation Technologies (ICICT)* (pp. 191-193). IEEE.

[3] Mittal, Pooja. "Machine learning (ml) based human activity recognition model using smart sensors in iot environment." *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2022.

[4] https://vitolavecchia.altervista.org/caratteristiche-usi-e-funzionamento-del-raspberry-pi/

[5] https://docs.conda.io/en/latest/

[6] https://vitolavecchia.altervista.org/caratteristiche-e-differenza-tra-face-detection-e-face-recognition-in-informatica/

[7] https://scikit-learn.org/stable/

[8] https://pypi.org/project/mediapipe/

[9] https://www.geeksforgeeks.org/opencv-python-program-face-detection/

[10] https://www.researchgate.net/figure/The-order-and-labels-for-keypoints-that-exist-in-the-hands-of-MediaPipe-30-For-Pose_fig2_364279614

[11] https://it.wikipedia.org/wiki/Frame_(telecomunicazioni)

[12] https://joblib.readthedocs.io/en/stable/

[13] https://github.com/google/mediapipe/blob/master/docs/solutions/face_detection.md