

MASSIMO PAPA

INTELLIGENZA ARTIFICIALE

ESPERIENZE DI LABORATORIO



Intelligenza Artificiale:
Generative AI - Computer Vision
Machine Learning - Perceptron



Massimo PAPA

ATTENZIONE: Documento in BOZZA

Questo documento è un working progress, le informazioni presenti potrebbero essere non complete, errate e in fase di revisione.

INTELLIGENZA ARTIFICIALE

- ESPERIENZE DI LABORATORIO -

SOMMARIO

ATTENZIONE: Documento in BOZZA	3
4. Computer Vision	11
4.1 La camera	11
4.1.4 Gestione della webcam incorporata o esterna	11
4.3 Rilevamento dei volti con MediaPipe	12
4.3.1 L'ambiente di Google MediaPipe	12
4.3.2 Il rilevamento facciale	15
4.3.3 Installiamo MediaPipe	15
4.3.4 Iniziamo a codificare	16
4.3.4.1 Descrizione funzionalità del programma	16
4.3.4.2 Il codice completo	16
4.3.4.3 Importazione librerie	18
4.3.4.4 La funzione resize()	18
4.3.4.5 Configurazione del rilevatore facciale	18
4.3.4.6 main loop	19
4.3.4.8 Rilascio delle risorse	20
4.3.4.9 Considerazioni finali	20

4. Computer Vision

4.1 La camera

4.1.4 Gestione della webcam incorporata o esterna

Infine vediamo il codice per gestire una vera webcam. Per vera intenderemo le diverse tipologie di hardware e software che possono essere viste come webcam dal s.o. in uso.

Per esempio potremmo avere una classica webcam incorporata nel pc portatile, oppure una webcam esterna collegata mediante porta USB o anche una webcam simulata da un'applicazione su dispositivo mobile (p.e. droidcam¹). Possiamo anche far sì che flussi di streaming vengano visti come webcam.

Ogni webcam viene visto da OpenCV come un dispositivo a se stante individuato da un numero intero, in questo modo si possono campionare diverse immagini da più webcam collegate allo stesso dispositivo hardware contemporaneamente.

Vediamo ora un semplice codice che ci permette di catturare le immagini da una webcam qualsiasi e di visualizzarle in una finestra. Il termine dell'acquisizione è dato dalla pressione del tasto 'q' da parte dell'utente (così come per il codice che utilizza una IPCam).

```
import cv2

# 0 -> Droidcam
# 1 -> cam portatile
# 3 -> webcam usb
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if ret: # non è strettamente necessario testare
        # se l'acquisizione è andata a buon fine
        cv2.imshow("frame", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

I commenti per la definizione delle diverse webcam dipendono da dispositivo a dispositivo.

¹ <https://droidcam.app/>

4.3 Rilevamento dei volti con MediaPipe

4.3.1 L'ambiente di Google MediaPipe

Seguiamo la documentazione di MediaPipe che è in continuo aggiornamento.

MediaPipe Solutions fornisce una suite di librerie e strumenti per applicare rapidamente tecniche di intelligenza artificiale (IA) e machine learning (ML) nelle tue applicazioni. Puoi collegare immediatamente queste soluzioni alle tue applicazioni, personalizzarle in base alle tue esigenze e utilizzarle su più piattaforme di sviluppo. MediaPipe Solutions fa parte del [progetto open source](#) MediaPipe, che ti consente di personalizzare ulteriormente il codice delle soluzioni in base alle tue esigenze di applicazione.

[estratto da]

<https://ai.google.dev/edge/mediapipe/solutions/guide.md?hl=it>]

Andiamo alla soluzione **Rilevamento facciale**

(https://ai.google.dev/edge/mediapipe/solutions/vision/face_detector?hl=it)

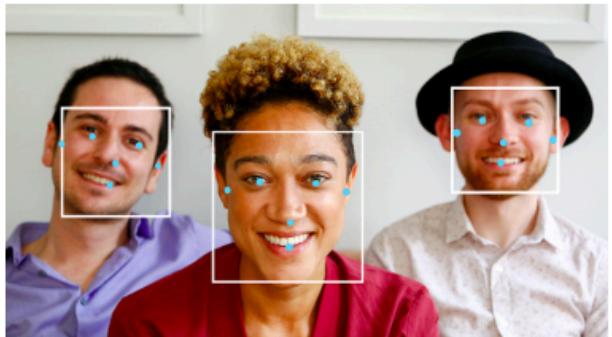
Home page > Google AI Edge > Soluzioni Questa pagina è stata utile?  

Guida al rilevamento dei volti

Invia feedback

L'attività Rilevatore di volti MediaPipe consente di rilevare i volti in un'immagine o in un video. Puoi usare questa attività per individuare i volti e le caratteristiche del viso all'interno di un'inquadratura. Questa attività utilizza un modello di machine learning (ML) che funziona con immagini singole o un flusso continuo di immagini. L'attività restituisce la posizione del volto, oltre ai seguenti punti chiave del viso: occhio sinistro, occhio destro, punta del naso, bocca, tragione dell'occhio sinistro e tragione dell'occhio destro.

[Prova! →](#)



Cliccando sul bottone *fai una prova!* si accede ad un ambiente di prova in cui si puo' vedere il modello di rilevamento dei volti all'opera.



Le immagini sopra riportate sono delle possibili prove che potete provare a fare. Come si puo' notare l'inferenza consiste nel riconoscere il volto riportando il livello di confidenza dell'inferenza effettuata.

L'applicazione permette di regolare alcuni parametri del modello, come si può vedere dal seguente screenshot:

Face Detection

Detect multiple faces and 6 facial landmarks of each detected face.

This solution is based on [BlazeFace](#), which can run ultrafast on mobile devices' GPU. For more information on the model, performance, etc, see the [documentation](#).

If you need a solution which detects more facial landmarks, check out the [Face Landmark](#) Detection solution.

Code examples
[Android](#) | [iOS](#) | [Python](#) | [Raspberry Pi](#) | [Web](#)

The sample parameters below can be changed. See [documentation](#) for more details

Inference delegate:

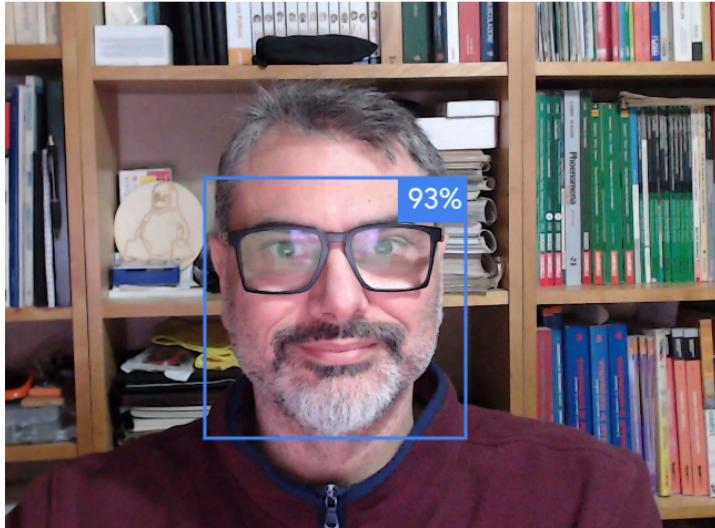
Model selections:

Display language:

Max results:

Score threshold:

Input: Logitech StreamCam (046...)



Inference time (ms): 17.4

Ritornando alla prima videata si può scegliere di visionare e provare direttamente il codice:

Prova! →

Inizia

Inizia a utilizzare questa attività seguendo una di queste guide all'implementazione per la tua piattaforma di destinazione. Queste guide specifiche per piattaforma illustrano un'implementazione di base di questa attività, inclusi un modello consigliato ed un esempio di codice con le opzioni di configurazione consigliate:

- **Android** - [Esempio di codice](#) - [Guida](#)
- **Python** - [Esempio di codice](#) - [Guida](#)
- **Web** - [Esempio di codice](#) - [Guida](#)
- **iOS** - [Esempio di codice](#) - [Guida](#)

Cliccando su *Esempio di codice* si accede a un Google Colab che vi permetterà di utilizzare

direttamente il codice.

Cliccando su *Guida* avrete una descrizione di come dovete utilizzare e personalizzare il codice secondo le vostre esigenze.

4.3.2 Il rilevamento facciale ²

I modelli di rilevamento del volto possono variare a seconda dei casi d'uso previsti, come il rilevamento a corto e lungo raggio. I modelli in genere fanno anche compromessi tra prestazioni, accuratezza, risoluzione e requisiti di risorse e, in alcuni casi, includono funzionalità aggiuntive.

Il modello che utilizzeremo è una variante di BlazeFace, un rilevatore di volti leggero e accurato ottimizzato per l'inferenza GPU mobile. I modelli BlazeFace sono adatti per applicazioni come la stima dei punti chiave facciali 3D, la classificazione delle espressioni e la segmentazione delle regioni facciali. *BlazeFace* utilizza una rete di estrazione di caratteristiche leggera simile a *MobileNetV1/V2*³.

La variante che utilizzeremo è *BlazeFace short-range*, è un modello leggero per rilevare volti singoli o multipli all'interno di immagini simili a selfie da una fotocamera di uno smartphone o da una webcam. Il modello è ottimizzato per immagini della fotocamera frontale del telefono a breve distanza. L'architettura del modello utilizza una tecnica di rete convoluzionale *Single Shot Detector (SSD)*⁴ con un codificatore personalizzato.

4.3.3 Installiamo MediaPipe

Per installare la libreria media

Su GNU/Linux:

```
$ pip install mediapipe
```

Su Microsoft Windows:

```
C:\Users\pythondev>pip install mediapipe
```

E' necessario anche scaricare il modello preaddestrato per il rilevamento facciale. Lo

² https://ai.google.dev/edge/mediapipe/solutions/vision/face_detector#models

³ <https://arxiv.org/abs/1704.04861>

⁴ <https://arxiv.org/abs/1512.02325>

facciamo andando a effettuare il download da questo link:
https://storage.googleapis.com/mediapipe-models/face_detector/blaze_face_short_range/tf16/latest/blaze_face_short_range.tflite

La scheda di questo modello, la cosiddetta *model-card* la si trova a questo indirizzo:
[https://storage.googleapis.com/mediapipe-assets/MediaPipe%20BlazeFace%20Model%20Card%20\(Short%20Range\).pdf](https://storage.googleapis.com/mediapipe-assets/MediaPipe%20BlazeFace%20Model%20Card%20(Short%20Range).pdf)

Queste informazioni si trovano sempre nella prima pagina della soluzione: "Rilevamento facciale" che abbiamo visto prima e più precisamente nel seguente riquadro.

Model name	Input shape	Quantization type	Model Card	Versions
BlazeFace (short-range)	128 x 128	float 16	info	Latest

Le label in azzurro sono i link riportati poco fa.

4.3.4 Iniziamo a codificare

4.3.4.1 Descrizione funzionalità del programma

Basicamente il programma che andremo a implementare consente di riconoscere i volti presenti nell'inquadratura attraverso la rilevazione mediante webcam.

Il programma stampa il numero di volti che riesce a individuare. Se non viene riconosciuto alcun volto allora stamperà a video 0.

4.3.4.2 Il codice completo

Ecco il codice completo dell'applicazione.

```
import cv2
import math
import mediapipe as mp

def resize(image):
    DESIRED_HEIGHT = 480
    DESIRED_WIDTH = 480
    h, w = image.shape[:2]
    if h < w:
        img = cv2.resize(image, (DESIRED_WIDTH, math.floor(h / (w / DESIRED_WIDTH))))
    else:
        img = cv2.resize(image, (math.floor(w / (h / DESIRED_HEIGHT)), DESIRED_HEIGHT))
    # cv2.imshow("frame", img)
```

```

    return img

# Abbreviazioni
BaseOptions = mp.tasks.BaseOptions
FaceDetector = mp.tasks.vision.FaceDetector
FaceDetectorOptions = mp.tasks.vision.FaceDetectorOptions
VisionRunningMode = mp.tasks.vision.RunningMode

# Crea un oggetto FaceDetector
base_options = BaseOptions(
    model_asset_path="blaze_face_short_range.tflite",
    delegate=BaseOptions.Delegate.CPU
)
options = FaceDetectorOptions(
    base_options=base_options, running_mode=VisionRunningMode.VIDEO
)
detector = FaceDetector.create_from_options(options)

# 0 -> Droidcam portatile
# 1 -> cam portatile
# 3 -> webcam usb
cap = cv2.VideoCapture(3)
while True:

    ret, frame = cap.read()
    # frame = resize(frame)
    # Flip the image horizontally for a selfie-view display.
    cv2.imshow("Frame", cv2.flip(frame, 1))
    # cv2.imshow("frame", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

    # Carica il frame
    image = mp.Image(image_format=mp.ImageFormat.SRGB, data=frame)
    # Analizza il frame per riconoscere un viso
    detection_result = detector.detect_for_video(
        image, int(cap.get(cv2.CAP_PROP_POS_MSEC)))
    # Se riconoscimento andato a buon fine
    # print(len(detection_result.detections))
    print("** Rilevati ", len(detection_result.detections), " visi")

detector.close()
cap.release()
cv2.destroyAllWindows()

```

4.3.4.3 Importazione librerie

```
import cv2
import math
import mediapipe as mp
```

Qui importiamo le librerie necessarie:

- OpenCV (cv2) per la gestione delle immagini e video
- math per calcoli matematici
- mediapipe per il rilevamento dei visi

4.3.4.4 La funzione `resize()`

```
def resize(image):
    DESIRED_HEIGHT = 480
    DESIRED_WIDTH = 480
    h, w = image.shape[:2]
    if h < w:
        img = cv2.resize(image, (DESIRED_WIDTH, math.floor(h / (w /
DESIRED_WIDTH))))
    else:
        img = cv2.resize(image, (math.floor(w / (h /
DESIRED_HEIGHT)), DESIRED_HEIGHT))
    return img
```

Questa funzione mantiene le proporzioni dell'immagine durante il ridimensionamento. Calcola le nuove dimensioni in base al lato più lungo dell'immagine, mantenendo un'altezza o larghezza massima di 480 pixel. È importante per gestire immagini di diverse dimensioni in modo uniforme.

In realtà questa funzione potremo anche non utilizzarla per i nostri scopi didattici.

4.3.4.5 Configurazione del rilevatore facciale

```
BaseOptions = mp.tasks.BaseOptions
FaceDetector = mp.tasks.vision.FaceDetector
FaceDetectorOptions = mp.tasks.vision.FaceDetectorOptions
VisionRunningMode = mp.tasks.vision.RunningMode

base_options = BaseOptions(
```

```

    model_asset_path="blaze_face_short_range.tflite",
delegate=BaseOptions.Delegate.CPU
)
options = FaceDetectorOptions(
    base_options=base_options, running_mode=VisionRunningMode.VIDEO
)
detector = FaceDetector.create_from_options(options)

```

Configura il rilevatore di visi di MediaPipe:

- Definisce degli alias per rendere il codice più leggibile
- Crea le opzioni base specificando il modello da utilizzare (blaze_face_short_range.tflite) e l'uso della CPU
- Configura il rilevatore per l'analisi video in tempo reale
- Crea l'oggetto detector che useremo per il rilevamento

4.3.4.6 main loop

```

cap = cv2.VideoCapture(3)
while True:
    ret, frame = cap.read()
    cv2.imshow("Frame", cv2.flip(frame, 1))
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

    image = mp.Image(image_format=mp.ImageFormat.SRGB, data=frame)
    detection_result = detector.detect_for_video(
        image, int(cap.get(cv2.CAP_PROP_POS_MSEC)))
    print("** Rilevati ", len(detection_result.detections), " visi")

```

Apre il flusso video dalla webcam. Il numero 3 indica quale webcam utilizzare (in questo caso una webcam USB esterna, ma su altri sistemi potrebbe essere un altro dispositivo).

Il ciclo while racchiude le fasi di acquisizione e analisi:

- Acquisisce un frame dalla webcam
- Mostra il frame ribaltato orizzontalmente (per effetto specchio)
- Permette di uscire premendo 'q'
- Converte il frame in un formato compatibile con MediaPipe
- Esegue il rilevamento dei visi sul frame
- Stampa il numero di visi rilevati Se è stato rilevato un viso allora si avrà una lista non vuota che contiene le informazioni sui visi rilevati. Il numero di elementi di tale lista

rappresenta il numero di visi rilevati.

4.3.4.8 Rilascio delle risorse

```
detector.close()  
cap.release()  
cv2.destroyAllWindows()
```

Questa parte è fondamentale per una corretta gestione delle risorse quando si raggiunge la fine del programma: chiude il rilevatore di visi, rilascia la webcam e chiude tutte le finestre create da OpenCV.

4.3.4.9 Considerazioni finali

Questo codice è un esempio introduttivo alla computer vision che esplora alcuni interessanti punti:

- Mostra l'integrazione tra diverse librerie (OpenCV e MediaPipe)
- Implementa un'applicazione real-time di video processing
- Introduce concetti base come l'acquisizione video, il processing delle immagini e il machine learning applicato alla visione artificiale

Si fa notare che è necessario implementare una logica più robusta di gestione degli errori mediante opportuni costrutti (`try-except`). In questa trattazione non è stato volutamente fatto per non distogliere il lettore dal focus della logica del codice.