Sara Buzzi

Luca Cinquetti

FACE AND EYE TRACKING

Sistemi di proctoring

AGENDA

- Introduzione
- Rilevamento del volto
- Codice: simulazione di procotoring
- Implementazioni
- Algoritmo HOG
- Conclusione

INTRODUZIONE: I SERVIZI DI PROCTORING

SERVIZI DI PROCTORING

L’espansione dell’e-learning ha richiesto soluzioni per garantire la correttezza e l'integrità degli esami online.

Sono stati sviluppati numerosi software di proctoring in tempo reale

TECNOLOGIE CHIAVE

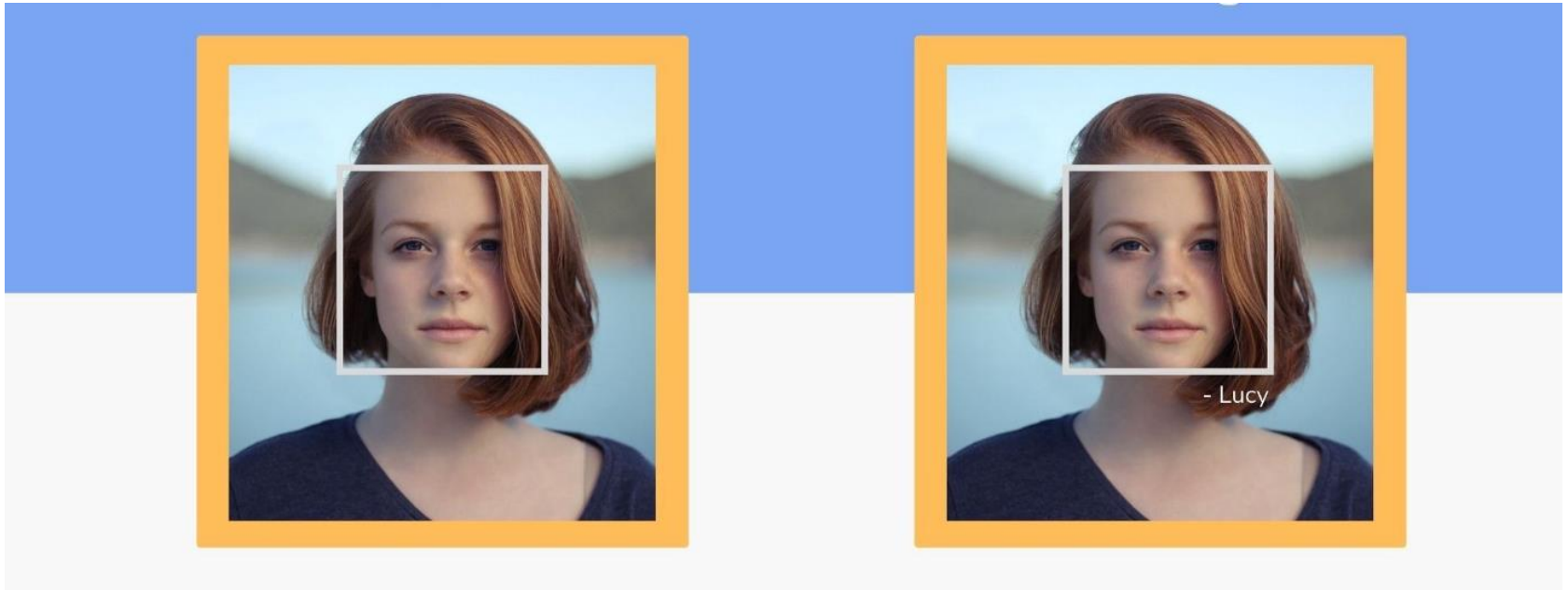
- **Face detection:** identificazione e tracciamento del volto.
- **Eye tracking:** monitoraggio della direzione dello sguardo.
- **Analisi dei movimenti:** rilevamento di comportamenti anomali.

RILEVAMENTO DEL VOLTO

E differenza con il riconoscimento

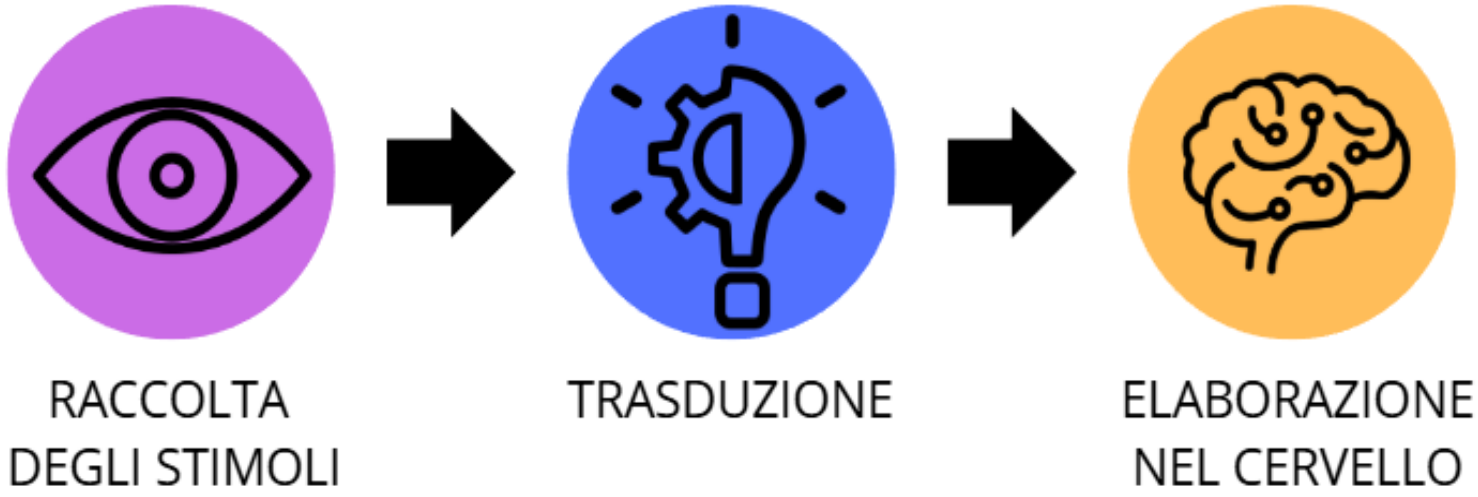
FACE DETECTION E FACE RECOGNITION

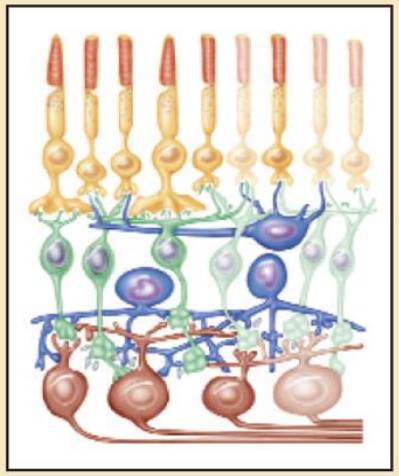
Queste due tecnologie si basano su principi comuni, ma si distinguono per gli obiettivi specifici e i contesti in cui vengono applicate.



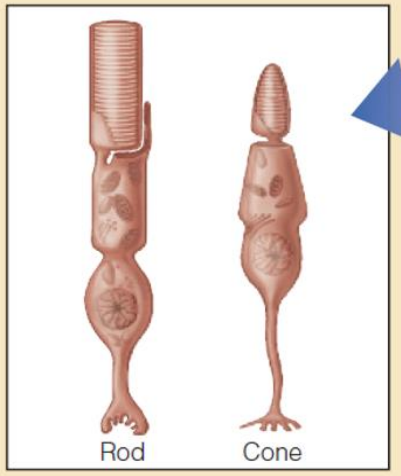
RILEVAMENTO DEL VOLTO: OCCHIO UMANO

Il cervello umano riconosce i volti attraverso una combinazione di **analisi locali** (occhi, naso, bocca) e **globali** (forma generale del volto).

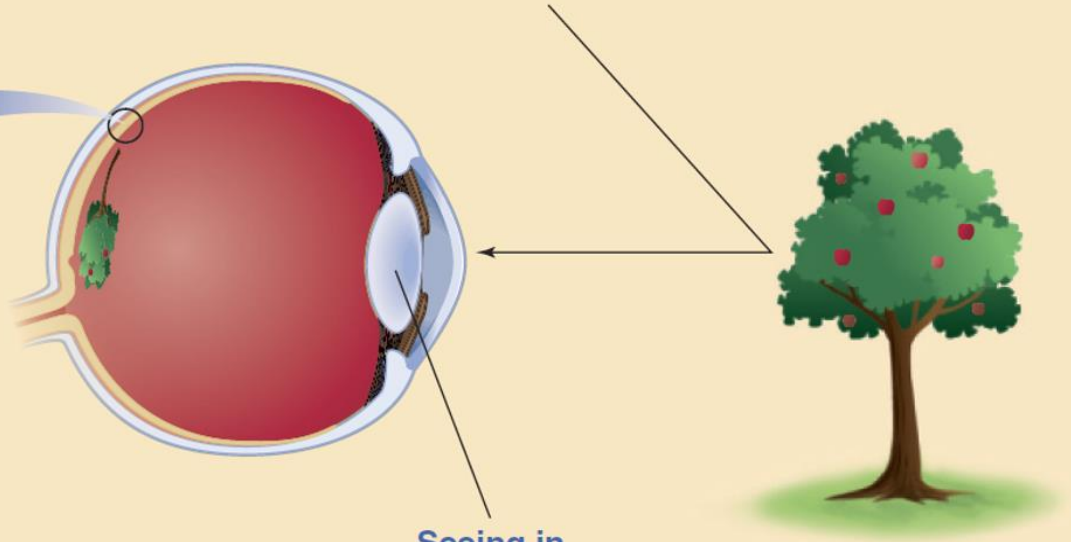




Seeing fine details



Seeing in dim light



Seeing in focus

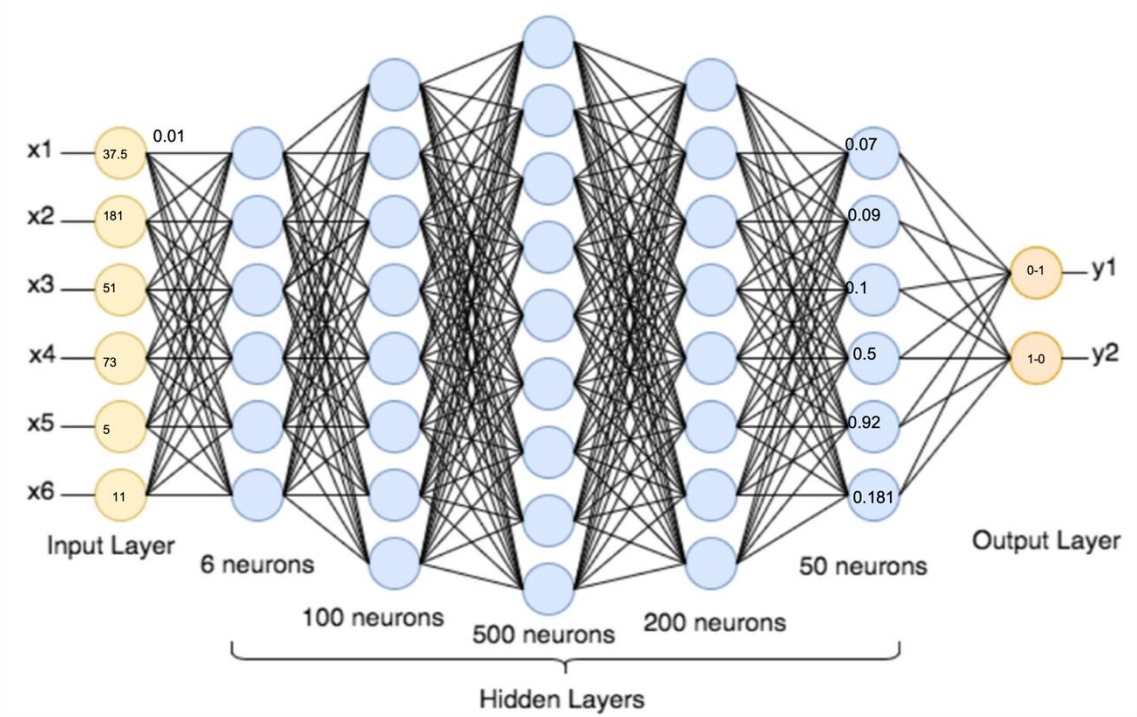
STEP 4
Neural processing: Signals travel in a network of neurons.

STEP 3
Receptor processes: Receptors transform light into electricity.

STEP 2
Light is reflected and focused to create an image of the tree on the retina.

STEP 1
Distal stimulus: The tree

RILEVAMENTO DEL VOLTO: MACCHINA

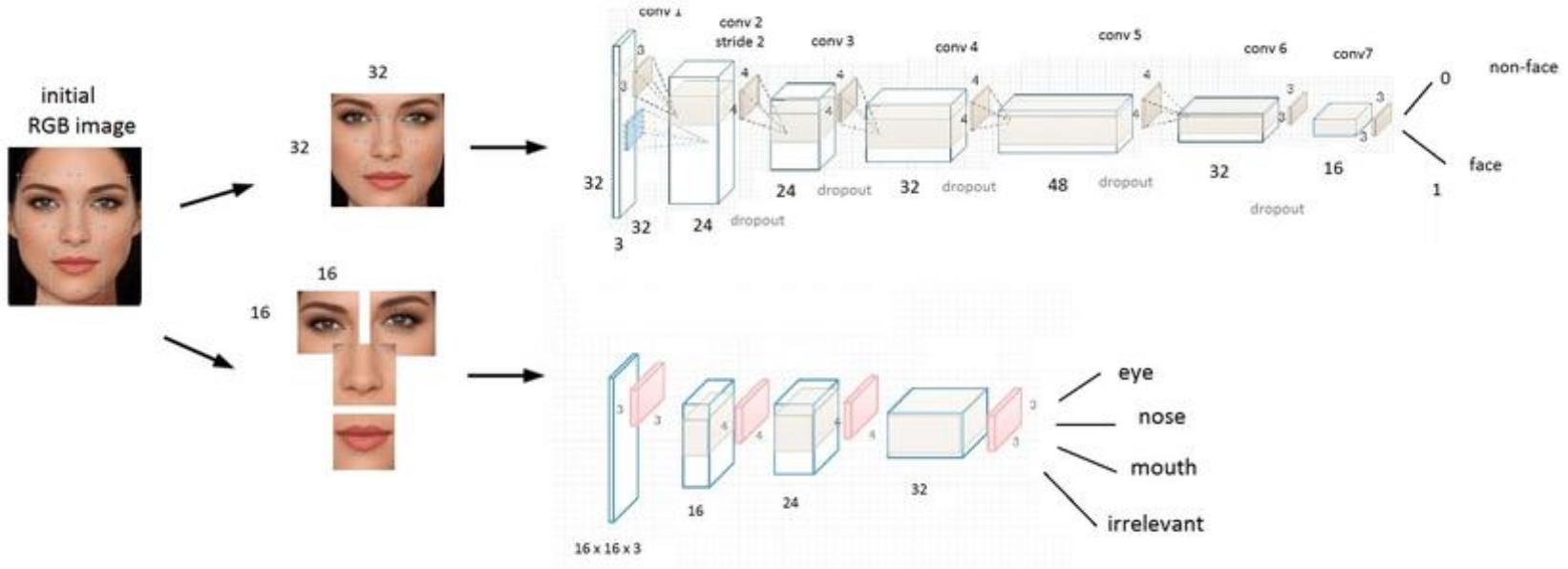


I computer imitano il processo umano usando le reti neurali convoluzionali (CNN)

Nei sistemi di proctoring, le CNN possono essere utilizzate per analizzare immagini e video in tempo reale

Il processo di face detection avviene in tempo reale e include diverse fasi:

- **Rilevamento del volto**
- **Analisi delle caratteristiche principali**
- **Tracciamento e monitoraggio**



SIMULAZIONE SERVIZIO DI PROCTORING

Codice con una basilare funzionalità di proctoring

IL CODICE

```python
import cv2
import face_recognition
import dlib
import numpy as np

# Caricamento del rilevatore di landmark per l'eye tracking
predictor_path = "shape_predictor_68_face_landmarks.dat"
face_landmark_predictor = dlib.shape_predictor(predictor_path)

# Apre la webcam
video_capture = cv2.VideoCapture(0)

# Funzione per rilevare gli occhi e il loro stato
def detect_eye_direction(landmarks):
    # Coordinate degli occhi
    left_eye = landmarks[36:42]
    right_eye = landmarks[42:48]

    # Calcolo del centroide per ciascun occhio
    def get_eye_center(eye_points):
        return np.mean(eye_points, axis=0)

    left_eye_center = get_eye_center(left_eye)
    right_eye_center = get_eye_center(right_eye)

    # Verifica se gli occhi sono rivolti verso il centro dello schermo
    # Ottieni la risoluzione effettiva della webcam
    frame_width = int(video_capture.get(cv2.CAP_PROP_FRAME_WIDTH))
    threshold = 0.18  # Soglia di tolleranza
    screen_center_x = int(frame_width)/2 # basato sulla risoluzione della webcam
    eyes_center_x = (left_eye_center[0] + right_eye_center[0]) / 2

    # Restituisce True se gli occhi sono centrati, False altrimenti
    return abs(eyes_center_x - screen_center_x) < threshold * screen_center_x
```

Qui abbiamo caricato un modello pre-allenato, il cui scopo è quello di seguire gli occhi tramite il calcolo di alcuni landmark

Questa sezione si occupa di utilizzare l'importazione precedente per avere le informazioni che ci interessano

Questa sezione calcola il posizionamento degli occhi rispetto al frame

IL CODICE

```python
while True:
    # Cattura un frame dalla webcam
    ret, frame = video_capture.read()
    if not ret:
        break

    # Converti il frame in RGB
    rgb_frame = frame[:, :, ::-1]

    # Trova i volti nel frame
    face_locations = face_recognition.face_locations(rgb_frame)

    # Alert per più di un volto rilevato
    if len(face_locations) > 1:
        cv2.putText(frame, "Diversi volti rilevati!", (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    # Se non vengono rilevati volti
    if len(face_locations) == 0:
        cv2.putText(frame, "Nessun volto rilevato!", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
```

Il feed acquisito dalla camera viene convertito in valori RGB, così da poterlo analizzare

Alert: segnala la mancata rilevazione di un volto

Alert: segnala una pluralità di volti

IL CODICE

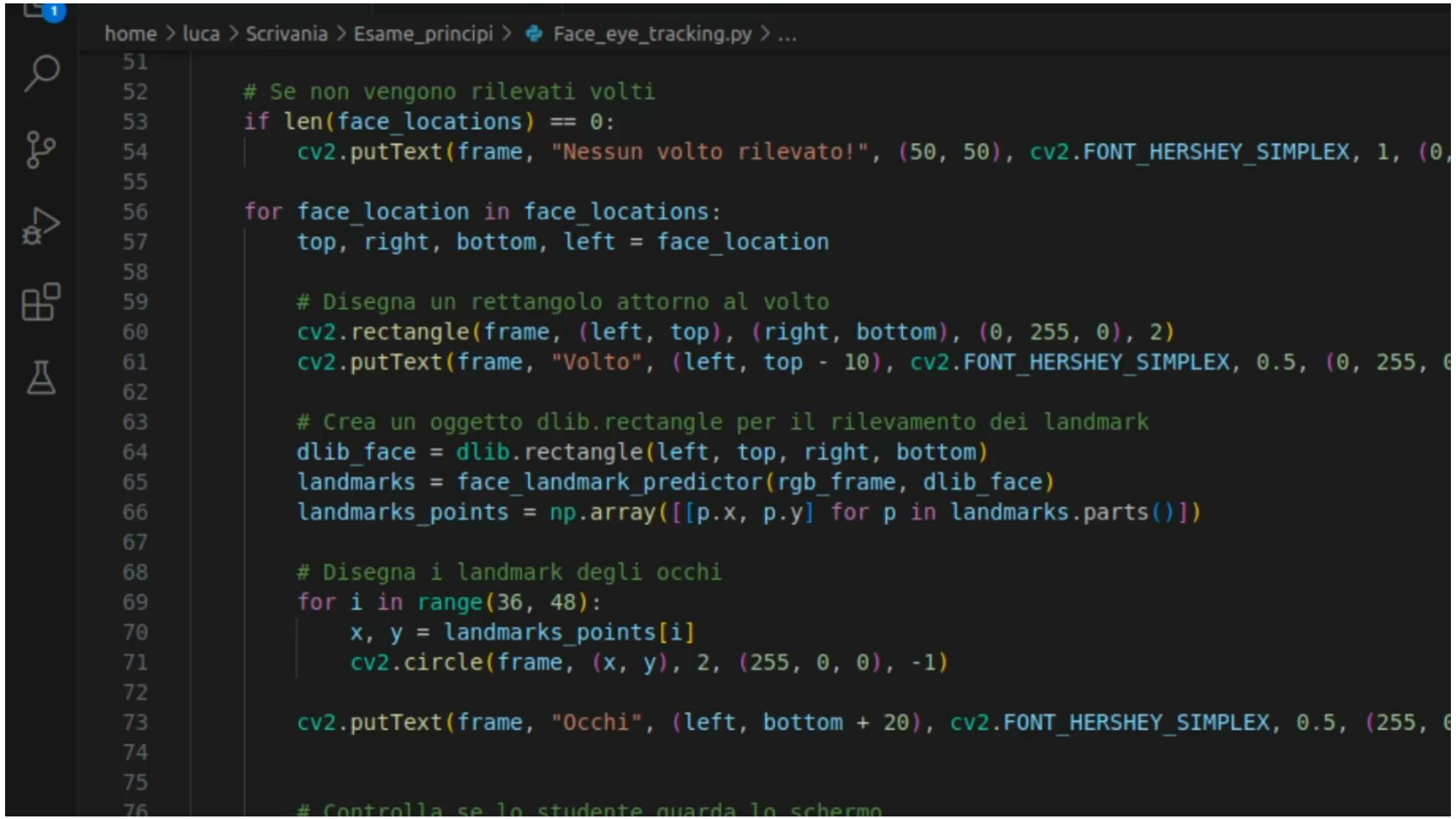
```
    # Se non vengono rilevati volti
    if len(face_locations) == 0:
        cv2.putText(frame, "Nessun volto rilevato!", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    for face_location in face_locations:
        top, right, bottom, left = face_location

        # Disegna un rettangolo attorno al volto
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.putText(frame, "Volto", (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

        # Crea un oggetto dlib.rectangle per il rilevamento dei landmark
        dlib_face = dlib.rectangle(left, top, right, bottom)
        landmarks = face_landmark_predictor(rgb_frame, dlib_face)
        landmarks_points = np.array([[p.x, p.y] for p in landmarks.parts()])

        # Disegna i landmark degli occhi
        for i in range(36, 48):
            x, y = landmarks_points[i]
            cv2.circle(frame, (x, y), 2, (255, 0, 0), -1)

        cv2.putText(frame, "Occhi", (left, bottom + 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)


        # Controlla se lo studente guarda lo schermo
        if not detect_eye_direction(landmarks_points):
            cv2.putText(frame, "ATTENZIONE: Guardare lo schermo!", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    # Mostra il frame
    cv2.imshow("Face Detection with Eye Tracking", frame)

    # Premi 'q' per uscire
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Rilascia la webcam e chiudi le finestre
video_capture.release()
cv2.destroyAllWindows()
```

Evidenzia la zona interessata, nella quale si concentra il programma

Gestisce la visualizzazione dei landmark

Verifica il posizionamento e notifica anomalie

DIMOSTRAZIONE



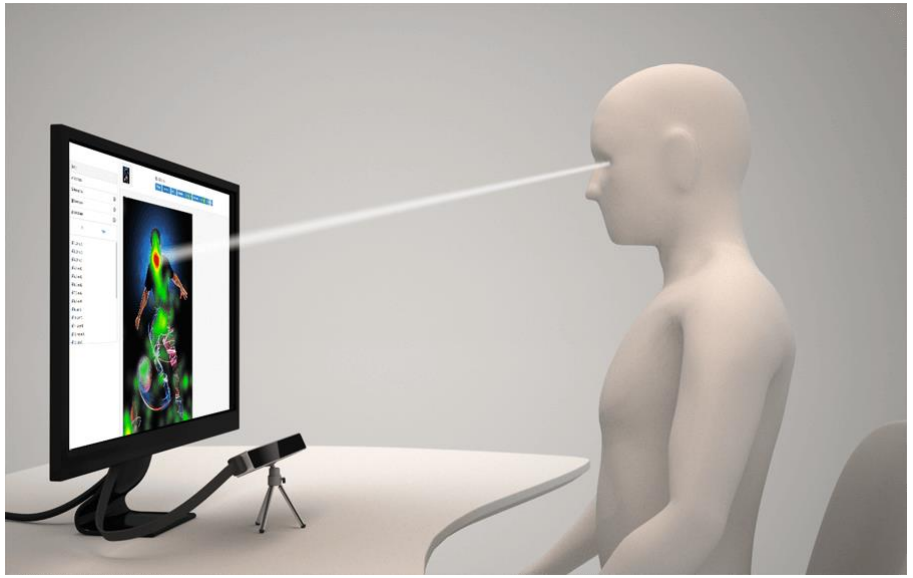
EYE TRACKING E OBJECT RECOGNITION

Integrazioni per un software più ricco

EYE TRACKING

Per rendere un servizio di proctoring più efficace, una delle integrazioni possibili è l'eye tracking.

Ad avere una precisione maggiore nel comprendere il comportamento dell'utente

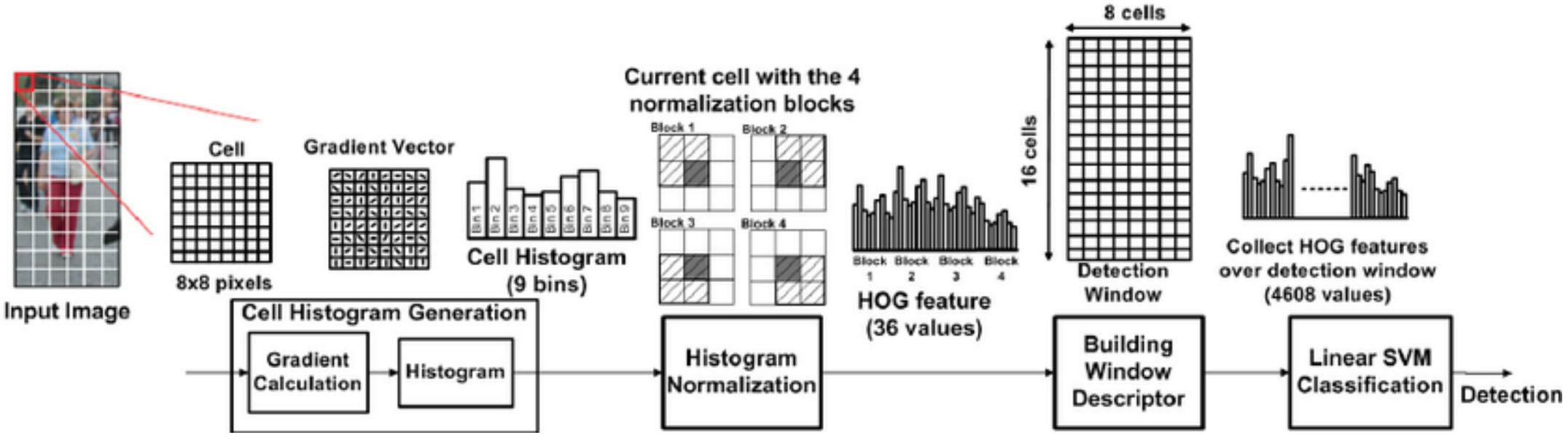


OBJECT RECOGNITION

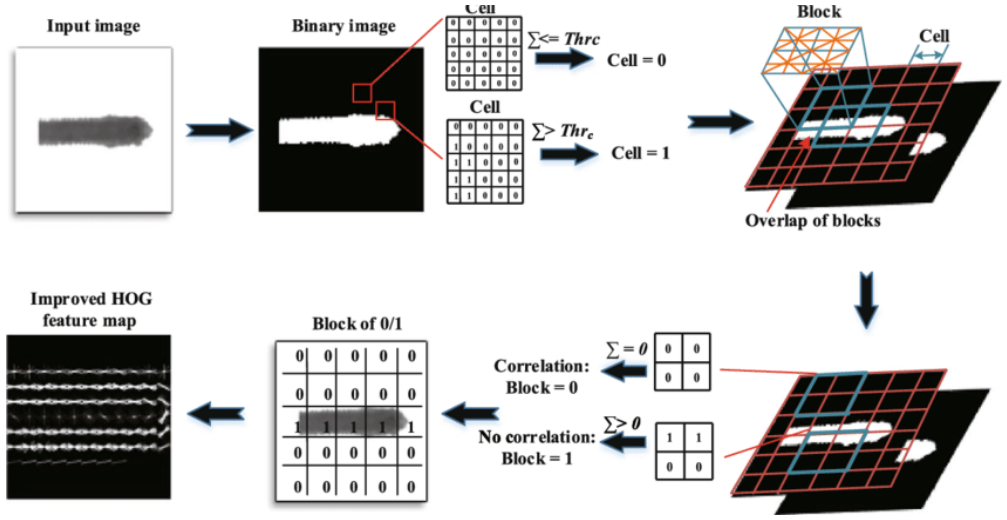
Nel nostro scenario, offre la possibilità di riconoscere con cosa interagisce l'utente

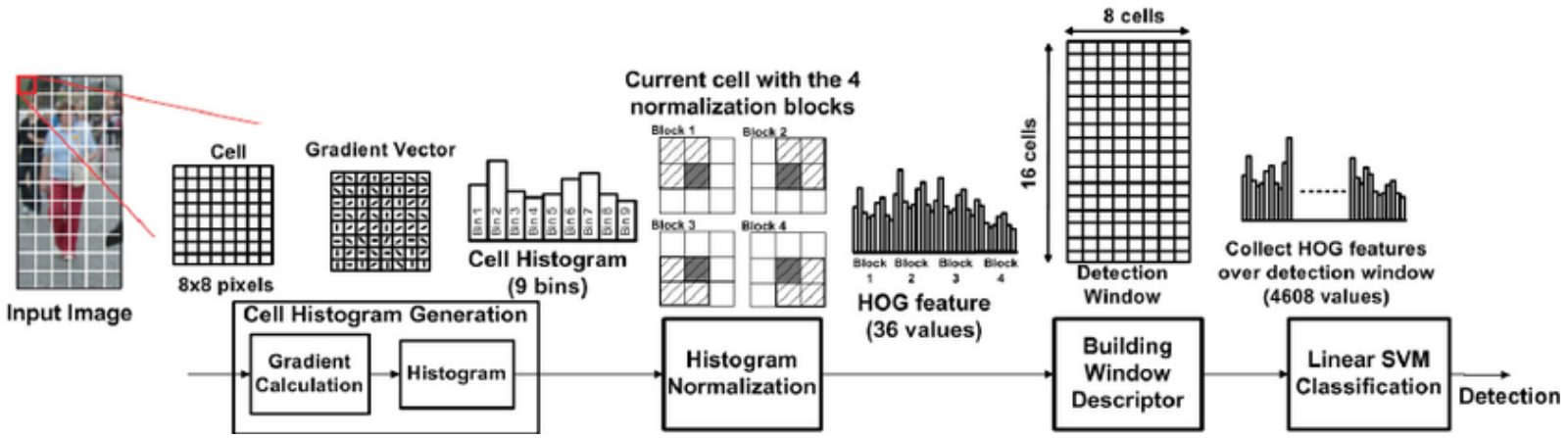
HISTOGRAM OF ORIENTED GRADIENTS

ALGORITMO HOG



ALGORITMO HOG

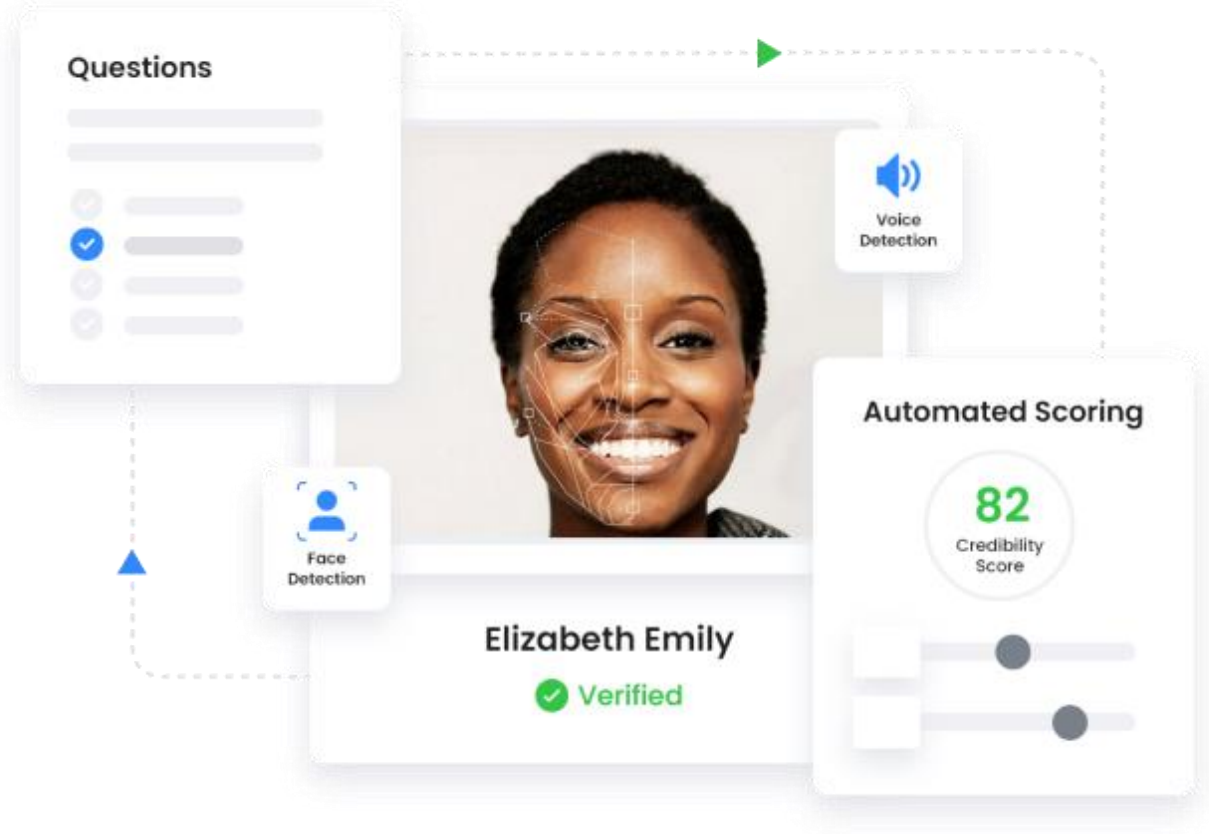




1. Conversione in scala di grigi
2. Suddivisione in celle
3. Calcolo dei gradienti
4. Costruzione dell'istogramma dei gradienti
5. Normalizzazione dei blocchi
6. Creazione del vettore descrittore
7. Classificazione con SVM

CONCLUSIONE

PER RIASSUMERE



APPLICAZIONI SIMILARI

GRAZIE

Sara Buzzi - 12214A
Luca Cinquetti - 20557A