Eye Distance Measurement

Name: Kirlos Ayad Libeb || كيرلس عياد لبيب

Program: Artificial intelligence

Studying Year : 4th year

group : 2

Code: 2230159

Supervisor : Dr. Shaima Al-Sabahi

Supervisor : Eng. Neehal fawzy

Eye Distance Measurement

Using MediaPipe and Camera Calibration

1. Introduction

This project aims to measure the distance between a person's eyes in real time using a webcam.
The system relies on **MediaPipe Face Mesh**, a powerful model developed by Google that provides highly accurate facial landmark detection.

To convert the measured distance from **pixels to centimeters**, the project uses a calibration step with a known physical object—a **VISA/ID card**—because its width is fixed (8.56 cm).
This ensures accurate real-world measurements regardless of camera resolution or distance.

2. Tools & Technologies

- **Python 3.12**
- **OpenCV** – for camera handling and drawing
- **MediaPipe Face Mesh** – for extracting facial landmarks
- **Math Library** – for distance calculations
- **Webcam**
- **VISA/ID card (8.56 cm width)** – for calibration

3. Project Concept

The system works in two main stages:

Stage 1 — Calibration

Since the camera only sees objects in **pixels**, it is necessary to provide a reference object of known size to compute a scale factor (cm per pixel).

A VISA/ID card is used because its width is precisely **8.56 cm**.

The user manually clicks two points on the card edges using the mouse.
From these two clicks:

$$\sqrt{{}^2({}_1y_2 - y) + {}^2({}_1x_2 - x)} = pixel\_distance$$

:And then

$$\frac{8.56}{pixel\_distance} = cm\_per\_pixel$$

Stage 2 — Eye Distance Measurement

MediaPipe Face Mesh detects the face and extracts landmark points:

- **33 → Right eye landmark**
- **263 → Left eye landmark**

The distance in pixels between the two eyes is:

$$\sqrt{{}^2({}_1y_2 - y) + {}^2({}_1x_2 - x)} = eye\_px$$

:Then convert it to centimeters

$$cm\_per\_pixel \times eye\_px = eye\_cm$$

.The results are displayed in real time on the screen

4. Why MediaPipe Face Mesh?

MediaPipe provides:

- 468 highly accurate facial landmarks
- Real-time performance (~30–60 FPS)
- Pretrained model → **no training required**
- Easy integration with OpenCV
- Very stable even with low lighting or movement

This makes it ideal for academic and applied computer vision projects.

5. Implementation Overview

Step 1: User points at two edges of the VISA card

The program records the pixel distance between these two points.

Step 2: Scale factor is computed

This factor converts any pixel measurement to centimeters.

Step 3: Eye landmarks are extracted automatically using MediaPipe

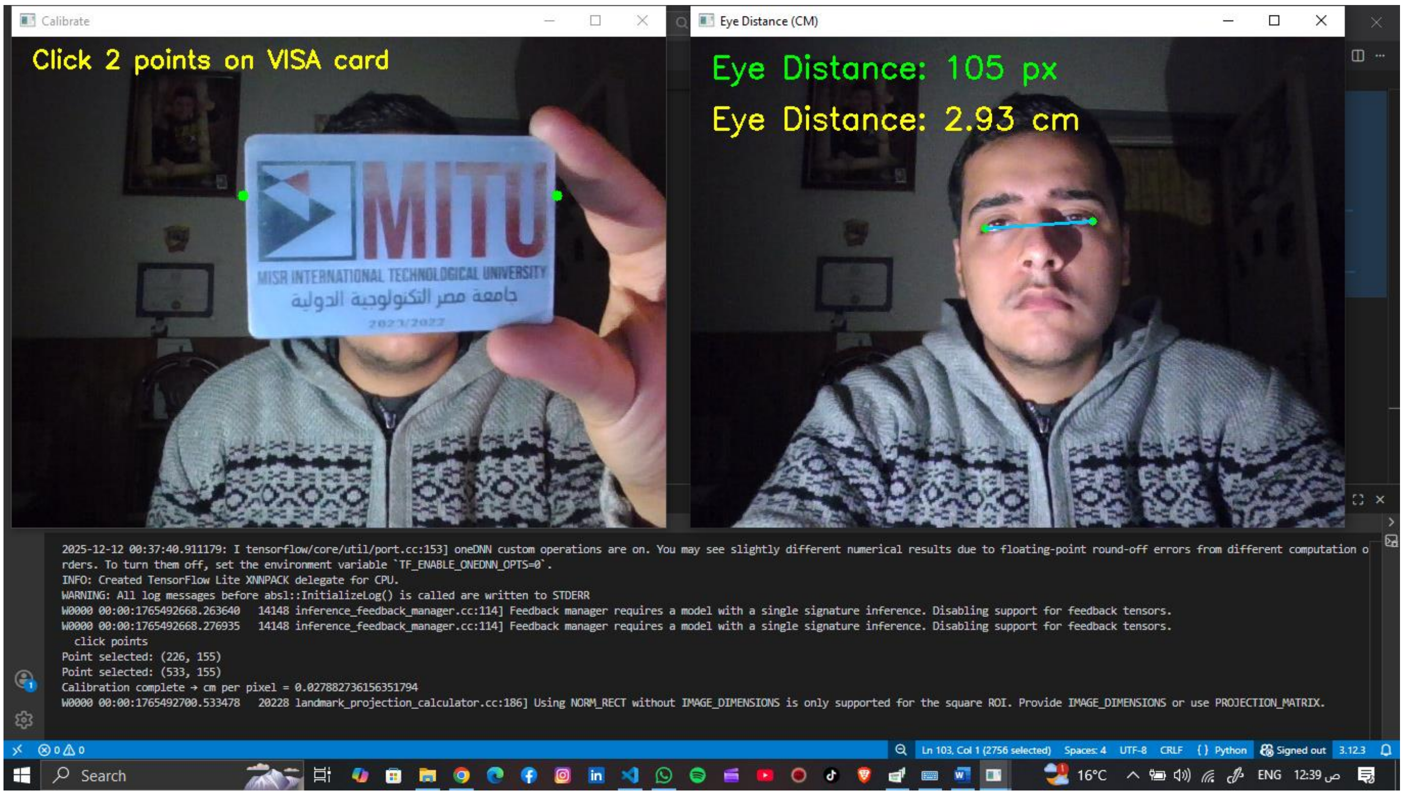
Step 4: Eye distance is displayed live in:

- Pixels
- Centimeters

❖ Code

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

mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(max_num_faces=1)

# الطول الحقيقي للبطاقة (سم)
CARD_WIDTH_CM = 8.56

cap = cv2.VideoCapture(0)

clicked_points = []

def mouse_callback(event, x, y, flags, param):
    global clicked_points
    if event == cv2.EVENT_LBUTTONDOWN:
        clicked_points.append((x, y))
        print("Point selected:", (x, y))

cv2.namedWindow("Calibrate")
cv2.setMouseCallback("Calibrate", mouse_callback)

calibrated = False
cm_per_pixel = None

print("  click points   ")

while True:
    ret, frame = cap.read()
    if not ret:
        break

    display = frame.copy()

    # لو جمعنا نقطتين → نحسب المعايرة
    if len(clicked_points) == 2 and not calibrated:
        p1, p2 = clicked_points
        pixel_distance = math.dist(p1, p2)
        cm_per_pixel = CARD_WIDTH_CM / pixel_distance
        calibrated = True
        print("Calibration complete → cm per pixel =", cm_per_pixel)

    # نعرض النقاط للمستخدم
    for p in clicked_points:
        cv2.circle(display, p, 5, (0, 255, 0), -1)

    cv2.putText(display, "Click 2 points on VISA card", (20, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2)

    cv2.imshow("Calibrate", display)
    if calibrated:
        break

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

# ---------------------- بعد المعايرة نبدأ قياس العينين ----------------------
while True:
    ret, frame = cap.read()
    if not ret:
        break

    h, w, _ = frame.shape
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = face_mesh.process(rgb)

    eye_distance_px = None

    if results.multi_face_landmarks:
        face = results.multi_face_landmarks[0]

        r_eye = face.landmark[33]
        l_eye = face.landmark[263]

        x1, y1 = int(r_eye.x * w), int(r_eye.y * h)
        x2, y2 = int(l_eye.x * w), int(l_eye.y * h)

        cv2.circle(frame, (x1, y1), 4, (0, 255, 0), -1)
        cv2.circle(frame, (x2, y2), 4, (0, 255, 0), -1)
        cv2.line(frame, (x1, y1), (x2, y2), (255, 200, 0), 2)

        eye_distance_px = math.dist((x1, y1), (x2, y2))

    if eye_distance_px and calibrated:
        eye_cm = eye_distance_px * cm_per_pixel

        cv2.putText(frame, f"Eye Distance: {int(eye_distance_px)} px", (20, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        cv2.putText(frame, f"Eye Distance: {eye_cm:.2f} cm", (20, 90),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)

    cv2.imshow("Eye Distance (CM)", frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

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

❖ Run



```
2025-12-12 00:37:40.911179: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation o
rders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1765492668.263640   14148 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors.
W0000 00:00:1765492668.276935   14148 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors.
  click points
Point selected: (226, 155)
Point selected: (533, 155)
Calibration complete → cm per pixel = 0.027882736156351794
W0000 00:00:1765492700.533478   20228 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is only supported for the square ROI. Provide IMAGE_DIMENSIONS or use PROJECTION_MATRIX.
```