Face Profiling Image Report

M.Junaid 109(4B)

Table of Contents

1. Introduction

The given Python code performs facial feature detection using the OpenCV library. The primary goal is to measure various facial features such as the Interpupillary Distance (IPD), face width, face height, and the face ratio. The code processes an image, detects faces and eyes, and visualizes measurements like IPD and face proportions directly on the image.

2. Libraries Used

- OpenCV (cv2): For image processing tasks like face and eye detection, drawing shapes, and displaying results.
- NumPy (np): For numerical calculations, particularly when calculating distances.
- Math: Used for mathematical functions like square roots (though not directly used in this particular code).
- sys: Included but not used in this code.

3. Functions and Workflow

calculate_distance(point1, point2)

```
def calculate_distance(point1, point2):
    return np.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)
```

Purpose: This function computes the Euclidean distance between two points using the formula:

distance = $\sqrt{((x1 - x2)^2 + (y1 - y2)^2)}$

It is used to calculate the distance between the centers of the two eyes (Interpupillary Distance or IPD).

measure_facial_features(image)

```
def measure_facial_features(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    if len(faces) == 0:
        print("No face detected in the image!")
        return
    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = image[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray)
        eye_centers = []
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
            eye_center = (x + ex + ew//2, y + ey + eh//2)
            eye_centers.append(eye_center)
            cv2.circle(image, eye_center, 2, (0, 0, 255), -1)
        if len(eye_centers) >= 2:
            ipd = calculate_distance(eye_centers[0], eye_centers[1])
            face_width = w
            face_height = h
            face_ratio = face_height / face_width
            cv2.putText(image, f'IPD: {ipd:.2f}px', (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
            cv2.putText(image, f'Face Width: {face_width}px', (10, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
            cv2.putText(image, f'Face Height: {face_height}px', (10, 90),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
            cv2.putText(image, f'Face Ratio: {face_ratio:.2f}', (10, 120),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
            cv2.line(image, eye_centers[0], eye_centers[1], (255, 0, 0), 2)
            face_center_x = x + w//2
            cv2.line(image, (face_center_x, y), (face_center_x, y+h), (0, 255, 255), 2)
```

Purpose: This function handles the main logic of facial feature extraction from an image. It performs the following:

- Convert to Grayscale: The image is converted to grayscale for better performance during face detection.
- Face and Eye Detection: It uses the Haar Cascade Classifiers to detect faces and eyes within the image.
- Feature Calculation: Calculates IPD, face width, face height, and face ratio (height/width).
- Visualization: Draws rectangles, lines, and displays calculated measurements on the image.
- Error Handling: If no face is detected, a message is printed and function exits.

4. Detailed Breakdown of Key Sections

- Face Detection: Uses OpenCV's pre-trained Haar Cascade Classifier to detect and mark faces.

- Eye Detection: Isolates and detects eyes within the detected face region and marks their centers.
- Distance and Proportions: Calculates IPD and face dimensions (height and width).
- User Interaction: Displays image and allows saving result if 's' key is pressed.

5. Code Output