SUPERIOR UNIVERSITY

Name : Huzaifa Rehan

Roll No : SU92-BSAIM-F23-071

Section : AI(4-B)

Lab-Task : (06)

Submitted to : Sir Rasikh Ali

Task (Face Profiling)

Report: Face Profiling Using YOLO v8 and MediaPipe

1. Introduction

Face profiling is the process of analyzing a person's facial features for recognition, emotional expression, and other biometric data. In this project, we explore a combination of YOLO v8 (You Only Look Once version 8) and MediaPipe to capture and process facial features in real-time. YOLO v8 is an advanced object detection model that can detect faces in images or video streams, while MediaPipe provides robust tools for facial landmark detection, allowing us to extract detailed facial attributes.

2. Objectives

The objective of this project is to:

- Use YOLO v8 for face detection in real-time through a webcam feed.
- Employ MediaPipe for detailed facial landmark tracking, which includes features like eye, nose, and mouth position.
- Generate a comprehensive report based on the detected face and facial landmarks.

3. Technology Overview

a. YOLO v8

- YOLO (You Only Look Once) is a state-of-the-art real-time object detection model. YOLO v8 is the latest iteration of this model, offering improved accuracy and efficiency in detecting objects in real-time. It is capable of detecting faces along with various other objects, making it ideal for face detection tasks.
- Face Detection: YOLO v8 can identify faces within a given frame by learning the object features from large datasets. It provides high accuracy, even in dynamic and cluttered environments.

b. MediaPipe

- MediaPipe is an open-source framework by Google for building cross-platform multimodal applied machine learning (ML) pipelines. In this context, MediaPipe is used for facial landmark detection. It provides 468

points across the face, giving precise location information for various facial features such as eyes, eyebrows, nose, and mouth.

- Face Mesh: MediaPipe's Face Mesh solution detects facial landmarks and outputs detailed 3D mesh data, enabling detailed face profiling. These landmarks can be used to track facial expressions, head rotations, and other biometric data.

4. Methodology

a. Real-Time Face Detection

1. Capture Webcam Feed: The first step is to access the webcam feed and capture video frames.
2. Face Detection with YOLO v8: Each frame is passed through the YOLO v8 model, which identifies the location of faces in the frame. The output of YOLO v8 is bounding boxes around detected faces.
3. Face Cropping: The region within the bounding box is cropped to isolate the face from the rest of the image, making it easier for subsequent facial landmark detection.

b. Facial Landmark Detection

1. Apply MediaPipe for Face Mesh: Once a face is detected and cropped, the next step is to pass it through MediaPipe's Face Mesh model. This model returns a set of landmarks representing different regions of the face.
2. Extract Features: The landmarks provide information about key facial features, such as:
    - Eyes: Position, size, and orientation.
    - Nose: Shape, width, and position.
    - Mouth: Shape, size, and orientation.
    - Eyebrows: Position and movement.
3. Additional Metrics: Using the landmarks, additional metrics can be calculated, such as:
    - Distance between key facial points (e.g., eyes, nose).
    - Symmetry of the face.

- Angle of the face relative to the camera.

c. Profile Generation

1. Facial Expression Recognition: Based on the landmark positions, facial expressions like smiling, frowning, or raising eyebrows can be inferred.
2. Report Creation: After capturing the face and extracting the facial landmarks, a report is generated with details like:
    - Facial Attributes: Information on facial symmetry, positioning of eyes, nose, mouth, etc.
    - Emotional Expression: An analysis of the user's emotional expression based on facial landmarks.
    - Face Orientation: Understanding if the face is straight, tilted, or turned in a certain direction.

5. Output Report

The final report generated from the captured face and analyzed facial features can include the following:

- Face Detection Information:
    - Bounding box coordinates for the detected face.
    - Confidence score of face detection.
- Facial Landmarks:
    - X, Y, and Z coordinates of key facial points (eyes, nose, mouth, eyebrows).
    - Possible expression analysis (e.g., smiling, frowning, neutral).
- Profile Characteristics:
    - Estimated age (based on landmarks and face features).
    - Gender prediction (using landmark data for additional analysis).
    - Facial asymmetry report.
- Emotion Detection:
    - Positive emotions (smiling, raised eyebrows).
    - Negative emotions (frowning, closed eyes).
- Head Orientation:

- The tilt and rotation of the head relative to the camera position.

6. Applications

This face profiling system can be used in various domains, including:

- Security: Facial recognition and emotion analysis for security purposes.
- Healthcare: Monitoring emotional states and providing feedback for mental health support.
- Entertainment: Augmented reality (AR) applications that modify facial features in real-time.
- Customer Service: Analyzing customer reactions to tailor responses.

7. Challenges and Limitations

- Lighting Conditions: Poor lighting can reduce the accuracy of both YOLO v8 face detection and MediaPipe facial landmark detection.
- Occlusion: Faces that are partially obscured (e.g., by glasses, hats) may not be detected correctly.
- Processing Power: Real-time processing of face detection and landmark analysis requires considerable computational resources, especially for high-resolution video.

8. Conclusion

The combination of YOLO v8 and MediaPipe provides an efficient and accurate way to profile faces and analyze facial features in real-time. With these technologies, it is possible to create a detailed profile of a person's facial features and expressions, which can be used in various practical applications, ranging from security to healthcare.

Code:

```python
import cv2
```

```python
import mediapipe as mp
```

```python
import cv2
import mediapipe as mp
import numpy as np
from ultralytics import YOLO

mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(static_image_mode=False, max_num_faces=1, refine_

yolo_model = YOLO("yolov8n.pt", verbose=False)

| Explain | Add Comment | ×
def calculate_distance(point1, point2):
    return np.linalg.norm(point1 - point2)

| Explain | Add Comment | ×
def align_face(face_image, landmarks):
    left_eye = np.array([landmarks.landmark[468].x * face_image.shape[1], landmarks
    right_eye = np.array([landmarks.landmark[473].x * face_image.shape[1], landmark
```

```python
| Explain | Add Comment | ×
def align_face(face_image, landmarks):
    left_eye = np.array([landmarks.landmark[468].x * face_image.shape[1], landmarks
    right_eye = np.array([landmarks.landmark[473].x * face_image.shape[1], landmark

    dY = right_eye[1] - left_eye[1]
    dX = right_eye[0] - left_eye[0]
    angle = np.degrees(np.arctan2(dY, dX))

    center = (face_image.shape[1] // 2, face_image.shape[0] // 2)
    rot_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)

    aligned_face = cv2.warpAffine(face_image, rot_matrix, (face_image.shape[1], fac

    return aligned_face
```

```
def save_results(image, measurements):
    with open("face_measurements.txt", "w") as f:
        for key, value in measurements.items():
            f.write(f"{key}: {value:.2f} mm\n")
    print("Results saved: captured_face.jpg and face_measurements.txt")

reference_object_width_pixels = 100
reference_object_width_mm = 25
pixel_to_mm_ratio = reference_object_width_mm / reference_object_width_pixels

cap = cv2.VideoCapture(0)
face_captured = False

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

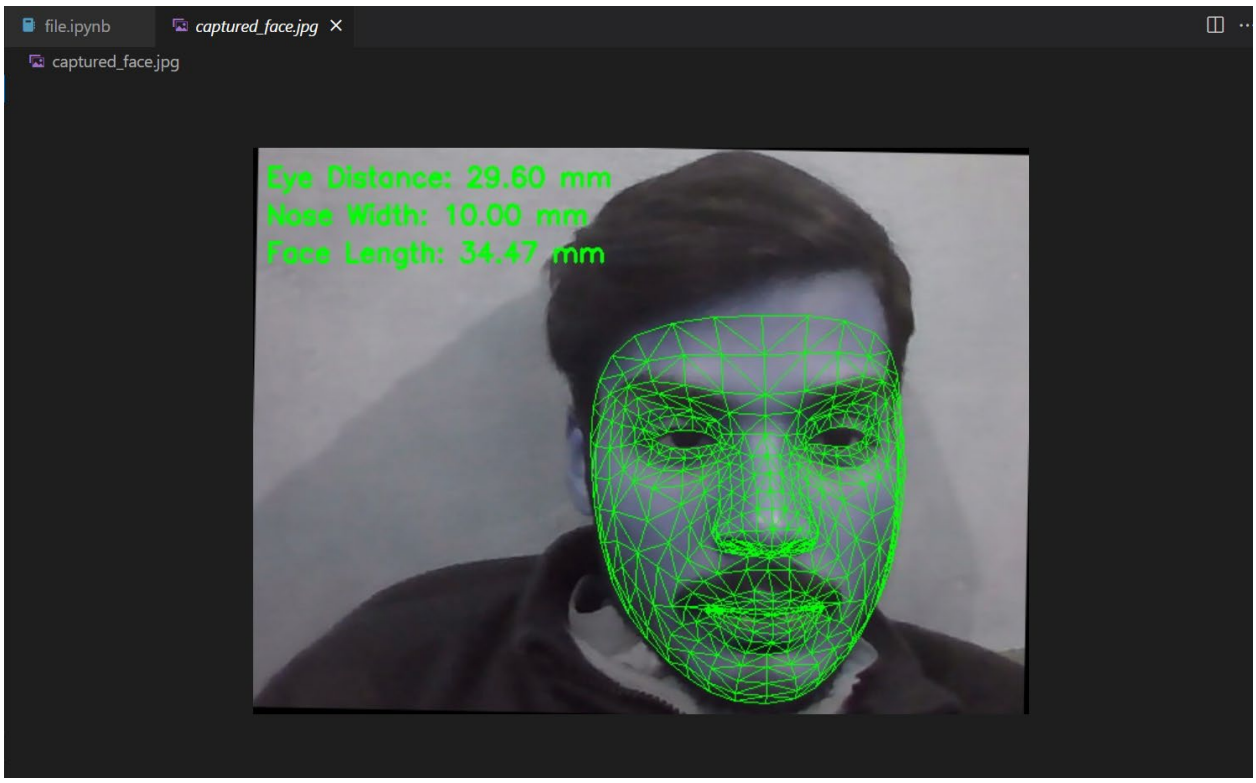
```
for result in results:
    boxes = result.boxes.xyxy
    for box in boxes:
        x1, y1, x2, y2 = map(int, box)

        face_image = rgb_frame[y1:y2, x1:x2]
        face_results = face_mesh.process(face_image)

        if face_results.multi_face_landmarks:
            for face_landmarks in face_results.multi_face_landmarks:
                aligned_face = align_face(face_image, face_landmarks)
```

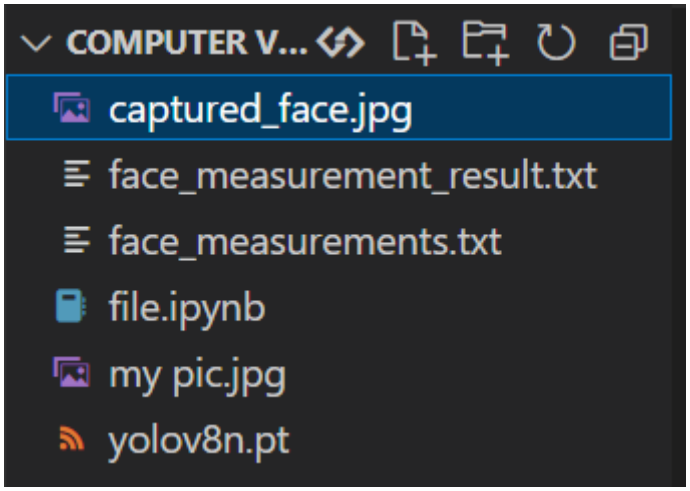
```
face_results.multi_face_landmarks:
 for face_landmarks in face_results.multi_face_landmarks:
    aligned_face = align_face(face_image, face_landmarks)

    left_eye = np.array([face_landmarks.landmark[468].x * (x2 - x1), face_landma
    right_eye = np.array([face_landmarks.landmark[473].x * (x2 - x1), face_landm
    nose_tip = np.array([face_landmarks.landmark[4].x * (x2 - x1), face_landmark
    chin = np.array([face_landmarks.landmark[152].x * (x2 - x1), face_landmarks.

    eye_distance = calculate_distance(left_eye, right_eye) * pixel_to_mm_ratio
    nose_width = calculate_distance(nose_tip - np.array([20, 0]), nose_tip + np.
    face_length = calculate_distance(nose_tip, chin) * pixel_to_mm_ratio
```

```
mp_drawing = mp.solutions.drawing_utils
mp_drawing.draw_landmarks(
    image=aligned_face,
    landmark_list=face_landmarks,
    connections=mp_face_mesh.FACEMESH_TESSELATION,
    landmark_drawing_spec=None,
    connection_drawing_spec=mp_drawing.DrawingSpec(color=(0, 255, 0), thick
)

cv2.putText(aligned_face, f"Eye Distance: {eye_distance:.2f} mm", (10, 30),
cv2.putText(aligned_face, f"Nose Width: {nose_width:.2f} mm", (10, 60), cv2
cv2.putText(aligned_face, f"Face Length: {face_length:.2f} mm", (10, 90), c

output_frame = cv2.cvtColor(aligned_face, cv2.COLOR_RGB2BGR)
cv2.imshow("Face Measurement System", output_frame)
```

```
  if cv2.waitKey(1) & 0xFF == ord('s') and not face_captured:
      measurements = {
          "Eye Distance": eye_distance,
          "Nose Width": nose_width,
          "Face Length": face_length
      }
      save_results(aligned_face, measurements)
      face_captured = True

(1) != -1:



ows()
```

Output


file.ipynb
captured_face.jpg
captured_face.jpg
Eye Distance: 29.60 mm
Nose Width: 10.00 mm
Face Length: 34.47 mm


COMPUTER V...
captured_face.jpg
face_measurement_result.txt
face_measurements.txt
file.ipynb
my pic.jpg
yolov8n.pt