Dayananda Sagar College of Engineering

Department of Electronics and Communication Engineering

Assignment

Program: B.E.
Course : Python Programming
Course Code: 19CE6IEPYP
Semester : 6
Section: A
Date: 06/06/2023

A Report on

Use case 6:Driver Drowsiness Detection

Submitted by

USN		NAME
1DS20EC010		Adil Pasha
1DS20EC049		Debesh Pramanick
1DS20EC050		Deekshitha B
1DS20EC058		Gaganasri M N

Submitted to
Dr.Ravindra S

Signature of faculty- incharge

Use Case -6

Driver Drowsiness detection

• Detect the driver whether the drowsy or awake, considering features like status of the eye, mouth (for yawning), different head angles while feeling sleepy. As each blink of eye cannot be considered as eye close, predict the status after analyzing multiple frames.

• Triggering alerts if the driver is sleepy (voice alerts are preferred).

• The dataset considered with different lightning conditions would be a better option.

Overview:

Our Project of Driver Drowsiness Detection uses the MediaPipe library and OpenCV to perform facial landmark detection and drowsiness detection. It captures video from a webcam and processes each frame to detect facial landmarks and calculate various ratios related to the eyes and lips. Based on these ratios, the code determines if the person is drowsy or not and provides warnings accordingly.

Abstract:

Driver error and carelessness are the main contributors to today's traffic accidents. Major driver errors are most frequently caused by fatigue, alcohol, and reckless driving. This project focuses on a computer-based driver sleepiness detection system for the Intelligent Transportation System that looks for unusual behaviour. Road safety depends on driving assistance systems' capacity to assess a driver's level of alertness
safety. By analyzing blink patterns and eye movements, drowsy driving can be identified early enough to prevent crashes. Accidents caused by sleepy driving are becoming more common in modern times. The majority of drivers experience low energy levels as a result of their weariness or fatigue from all the labour. As a result, people frequently experience sleepiness when driving. Accidents are much more likely to occur as a result of these fatigues. The goal of this project is to create a model that can recognise when a driver is getting sleepy or drowsy and to alert the user when this happens. The majority of high-end vehicles have these models built into the vehicle itself, but public transportation vehicles do not have access to this technology. Python is the implementation language we're employing for this project.

Introduction:

A growing number of vocations in today's society require long-term care. In order to respond swiftly to unanticipated situations, drivers need to keep a close eye on the road. Driver fatigue is a major factor in many traffic accidents. Technologies that can identify and warn a driver when they are in a bad psychophysical condition must be developed in order to lower the frequency of fatigue-related auto accidents. The development of such systems, however, faces many difficulties, one of which is the quick and precise assessment of a driver's fatigue symptoms. One technical choice for putting driver sleepiness detection systems into practice is the use of a vision-based technique. We describe an easy-to-use and flexible vision-based solution for bus driver monitoring. The system's modules include eye detection, eye openness estimation, fusion, sleepy measure estimation and classification.

The global accident death rate has increased to 21%. This demonstrates how serious the issue is. A safe device called drowsiness detection can prevent accidents from being brought on by drivers who nod off behind the wheel. The goal of this Python project is to create a drowsiness detection model that can recognise when a driver's eyes are closed for a brief period of time

Methodology Adopted:

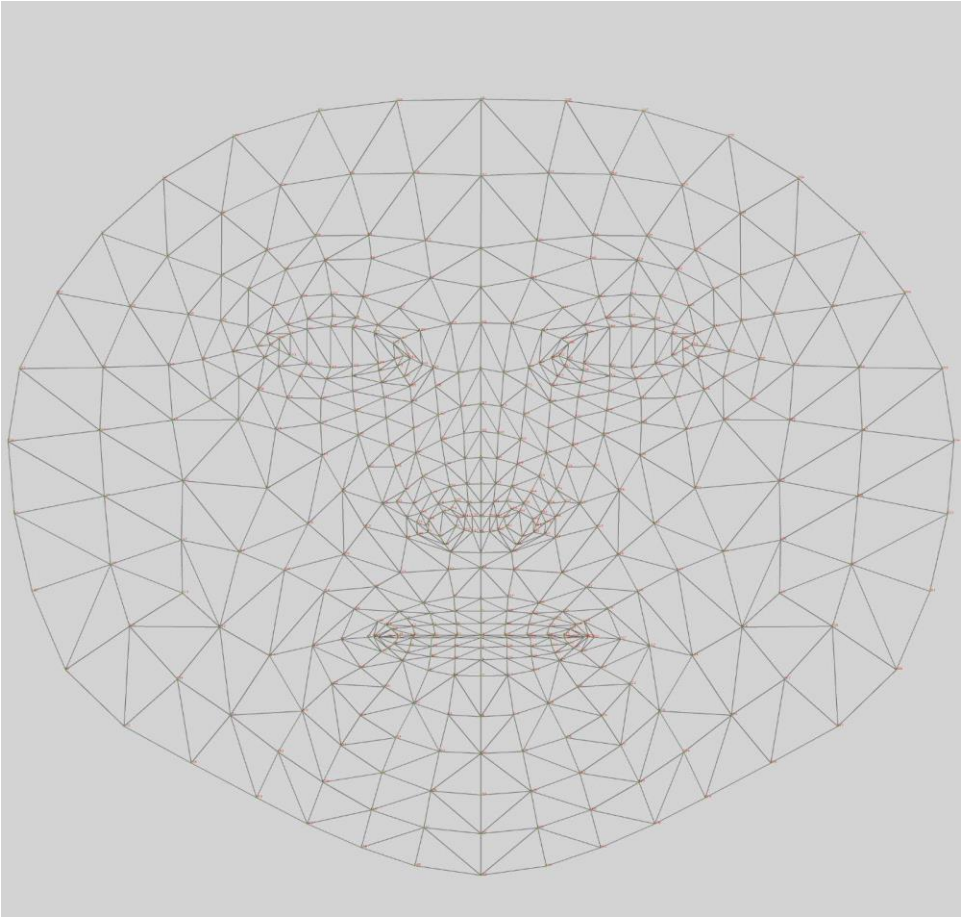
Euclidean distance

This detection can be quickly done using “sshape predictor face landmarks” that mark the essential landmarks on the face. Thus python also provides the flexibility for detecting such serious and significant detection via OpenCV modules. One can easily make their own phone camera for detecting drowsiness

Eye aspect ratio

a drowsy driver alert system that has been developed using such a technique in which the Video Stream Processing (VSP) is analyzed by eye blink concept through an Eye Aspect Ratio (EAR) and Euclidean distance of the eye. Face landmark algorithm is also used .

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$



Algorithm:

1. Import the required libraries: ‘mediapipe’, ‘cv2’, ‘scipy’, ‘threading’, and ‘pyttsx3’.
2. Define helper functions:
    - ‘run_speech(speech, speech_message)’: Use the ‘pyttsx3’ library to convert the provided speech message into audio and play it.
    - ‘draw_landmarks(image, outputs, land_mark, color)’: Draw circles on the image at the specified landmark points using the provided color.
3. Define drowsiness detection functions:
    - ‘euclidean_distance(image, top, bottom)’: Calculate the Euclidean distance between two points in the image using their coordinates.
    - ‘get_aspect_ratio(image, outputs, top_bottom, left_right)’: Calculate the aspect ratio between two sets of landmarks to determine the state of eyes or lips.
4. Set constants and variables:
    - Define configuration parameters such as ‘STATIC_IMAGE’, ‘MAX_NO_FACES’, ‘DETECTION_CONFIDENCE’, and ‘TRACKING_CONFIDENCE’.
    - Define color constants for drawing landmarks.
    - Define landmark lists for lips, eyes, and face.

- Initialize the ‘FaceMesh’ model with the specified configuration parameters.

5. Main loop:
   - Start capturing video frames from the webcam using ‘cv.VideoCapture()’.
   - Process each frame using the ‘FaceMesh’ model to detect facial landmarks.
   - Draw landmarks on the image for visualization purposes.
   - Calculate the aspect ratios for eyes and lips and determine drowsiness based on predefined thresholds.
   - If drowsiness is detected, provide warning messages using text-to-speech.
   - Display the processed image with landmarks, frame count, warnings given, eye ratios, and lip ratio using ‘cv.imshow()’ and ‘cv.putText()’.
   - Continue the loop until the user presses the Esc key.
6. Clean up: Release the video capture and close all OpenCV windows.

Prgram:

```
#  mediapipe fasemesh landmarks:
#https://github.com/google/mediapipe/blob/a908d668c730da128dfa8d9f6bd25d519d006692/mediapipe/modules/face_geometry/data/canonical_face_model_uv_visualization.png

import mediapipe as mp
import cv2 as cv
from scipy.spatial import distance as dis
import threading
import pyttsx3

# Function to run speech using text-to-speech

def run_speech(speech, speech_message):
    speech.say(speech_message)
    speech.runAndWait()

# Function to draw landmarks on the image

def draw_landmarks(image, outputs, land_mark, color):
    height, width = image.shape[:2]
    for face in land_mark:
        point = outputs.multi_face_landmarks[0].landmark[face]

        point_scale = ((int)(point.x * width), (int)(point.y * height))

        cv.circle(image, point_scale, 2, color, 1)

# Function to calculate Euclidean distance between two points

def euclidean_distance(image, top, bottom):
eight, width = image.shape[0:2]
```

```
        point1 = int(top.x * width), int(top.y * height)
        point2 = int(bottom.x * width), int(bottom.y * height)

        distance = dis.euclidean(point1, point2)
        return distance
```

 # Function to calculate aspect ratio between two sets of landmarks

```
def get_aspect_ratio(image, outputs, top_bottom, left_right):
        landmark = outputs.multi_face_landmarks[0]

        top = landmark.landmark[top_bottom[0]]
    bottom = landmark.landmark[top_bottom[1]]

        top_bottom_dis = euclidean_distance(image, top, bottom)

        left = landmark.landmark[left_right[0]]
        right = landmark.landmark[left_right[1]]

        left_right_dis = euclidean_distance(image, left, right)

        aspect_ratio = left_right_dis / top_bottom_dis

        return aspect_ratio
```

 # Constants and configurations

```
face_mesh = mp.solutions.face_mesh
draw_utils = mp.solutions.drawing_utils
landmark_style = draw_utils.DrawingSpec((0, 255, 0), thickness=1, circle_radius=1)
connection_style = draw_utils.DrawingSpec((0, 0, 255), thickness=1, circle_radius=1)
```

```
STATIC_IMAGE = False
MAX_NO_FACES = 2
DETECTION_CONFIDENCE = 0.6
TRACKING_CONFIDENCE = 0.5

COLOR_RED = (0, 0, 255)
COLOR_BLUE = (255, 0, 0)
COLOR_GREEN = (0, 255, 0)
```

 # Lists of specific facial landmarks

```
LIPS = [61, 146, 91, 181, 84, 17, 314, 405, 321, 375, 291, 308, 324, 318, 402, 317, 14, 87,
178, 88, 95,
        185, 40, 39, 37, 0, 267, 269, 270, 409, 415, 310, 311, 312, 13, 82, 81, 42, 183, 78]

RIGHT_EYE = [33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161,
246]
LEFT_EYE = [362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385, 384,
398]
```

 # Lists of specific facial landmarks

```
LEFT_EYE_TOP_BOTTOM = [386, 374]
LEFT_EYE_LEFT_RIGHT = [263, 362]


RIGHT_EYE_TOP_BOTTOM = [159, 145]
RIGHT_EYE_LEFT_RIGHT = [133, 33]

UPPER_LOWER_LIPS = [13, 14]
LEFT_RIGHT_LIPS = [78, 308]

FACE = [10, 338, 297, 332, 284, 251, 389, 356, 454, 323, 361, 288, 397, 365, 379, 378,
400, 377, 152, 148, 176, 149, 150, 136, 172, 58, 132, 93, 234, 127, 162, 21, 54, 103, 67,
109]
```

```
# Initializing the face mesh model

face_model = face_mesh.FaceMesh(static_image_mode=STATIC_IMAGE,
                                max_num_faces=MAX_NO_FACES,
                                min_detection_confidence=DETECTION_CONFIDENCE,
                                min_tracking_confidence=TRACKING_CONFIDENCE)

# Opening the webcam

capture = cv.VideoCapture(0)

frame_count = 0
warnings_given = 0
min_frame = 6
min_tolerance = 5.0

speech = pyttsx3.init()

# Main loop

while True:
    # Read the frame from the webcam
    result, image = capture.read()

    if result:
        image_rgb = cv.cvtColor(image, cv.COLOR_BGR2RGB)
        outputs = face_model.process(image_rgb)

                if outputs.multi_face_landmarks:

        # Draw landmarks on the face

                draw_landmarks(image, outputs, FACE, COLOR_GREEN)

                draw_landmarks(image, outputs, LEFT_EYE_TOP_BOTTOM, COLOR_RED)
                draw_landmarks(image, outputs, LEFT_EYE_LEFT_RIGHT, COLOR_RED)

        # Calculate the aspect ratio of the left eye

                ratio_left = get_aspect_ratio(image, outputs, LEFT_EYE_TOP_bottom,
        LEFT_EYE_LEFT_RIGHT)

                draw_landmarks(image, outputs, RIGHT_EYE_TOP_BOTTOM, COLOR_RED)
                draw_landmarks(image, outputs, RIGHT_EYE_LEFT_RIGHT, COLOR_RED)

        # Calculate the aspect ratio of the right eye
```

```
        ratio_right = get_aspect_ratio(image, outputs, RIGHT_EYE_TOP_BOTTOM,
RIGHT_EYE_LEFT_RIGHT)

# Calculate the average aspect ratio of both eyes
        ratio = (ratio_left + ratio_right) / 2.0

        if ratio > min_tolerance:
            frame_count += 1
        else:
            frame_count = 0

        if frame_count > min_frame:
            warnings_given += 1
            message = 'Drowsy Alert: It seems you are sleeping... please wake up'
            t = threading.Thread(target=run_speech, args=(speech, message))
            t.start()

        draw_landmarks(image, outputs, UPPER_LOWER_LIPS, COLOR_BLUE)
        draw_landmarks(image, outputs, LEFT_RIGHT_LIPS, COLOR_BLUE)

# Calculate the aspect ratio of the lips
        ratio_lips = get_aspect_ratio(image, outputs, UPPER_LOWER_LIPS,
LEFT_RIGHT_LIPS)
        if ratio_lips < 1.8:
            message = 'Drowsy Warning: Please focus on the road'
            p = threading.Thread(target=run_speech, args=(speech, message))
            p.start()

 # Add text information on the image
        BLACK = (255, 255, 255)
        font = cv.FONT_HERSHEY_SIMPLEX
        font_size = 1.1
        font_color =BLACK
        font_thickness = 2
        x, y = 30, 30
# Add text information on the image

      img_text = cv.putText(image, str(frame_count), (30,30), font, font_size,
      font_color, font_thickness, cv.LINE_AA)
              img_text = cv.putText(image, str(warnings_given), (100, 30), font,
      font_size, font_color, font_thickness, cv.LINE_AA)
              img_text = cv.putText(image, str(round(ratio, 3)), (30, 60), font,
      font_size, font_color, font_thickness, cv.LINE_AA)
              img_text = cv.putText(image, str(round(ratio_lips, 3)), (30, 90), font,
      font_size, font_color, font_thickness, cv.LINE_AA)

      cv.imshow("FACE MESH", image)
```

```
        if cv.waitKey(1) & 255 == 27:
            break


    capture.release()
    cv.destroyAllWindows()
```

Code Description:

The code can be divided into several sections:

- Importing Required Libraries: The necessary libraries are imported, including mediapipe, cv2 (OpenCV), scipy, threading, and pyttsx3 (text-to-speech library).
- Helper Functions: Two helper functions are defined:
    - run_speech(): This function uses the pyttsx3 library to convert text to speech and speaks the provided message.

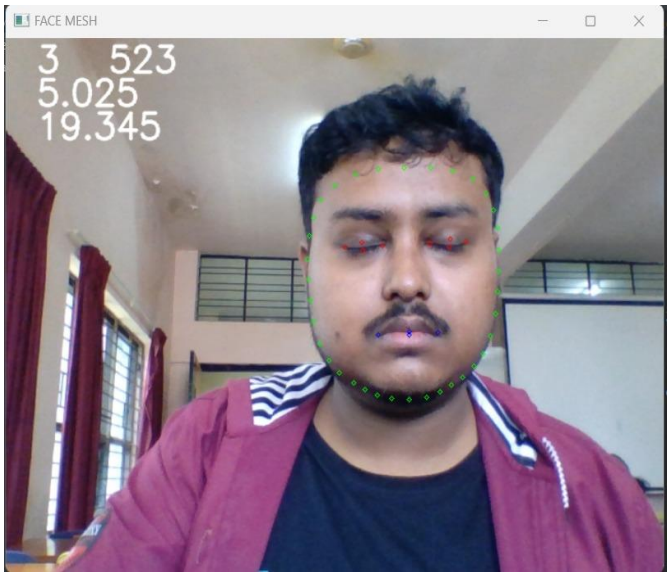
 draw_landmarks(): This function takes an image, the outputs from facial landmark detection, a list of landmarks to be drawn, and a color. It draws circles on the image at the specified landmarks.

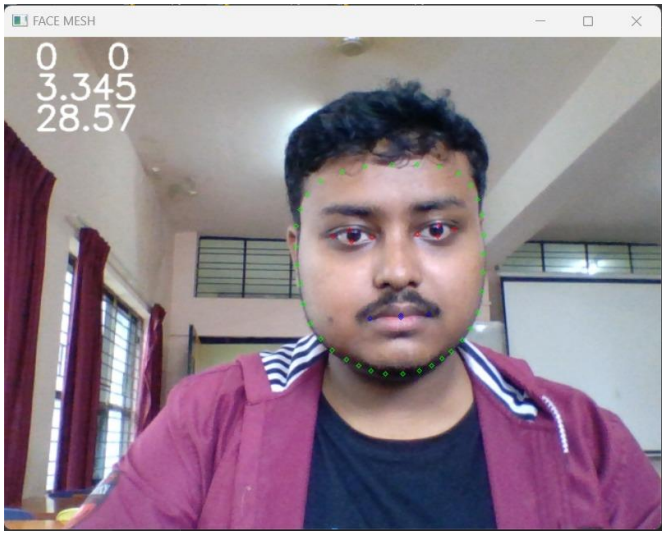
- Drowsiness Detection Functions:
    - euclidean_distance(): This function calculates the Euclidean distance between two points in an image using their coordinates.
    - get_aspect_ratio(): This function calculates the aspect ratio between two sets of facial landmarks (e.g., eye landmarks, lip landmarks) to determine if the eyes are closed or if the lips are open.

- Constants and Variables:
  - Constants: These include configuration parameters such as STATIC_IMAGE, MAX_NO_FACES, DETECTION_CONFIDENCE, and TRACKING_CONFIDENCE. These parameters control the behavior and accuracy of the face mesh detection.
  - Color Constants: These define RGB values for different colors used for drawing landmarks on the image.
  - Landmark Lists: These lists specify the indices of specific landmarks related to the lips, eyes, and face in the face mesh model.
- FaceMesh Model Initialization: An instance of the FaceMesh model from mediapipe is created with the specified configuration parameters.
- Main loop:
  - Video Capture: The code starts a loop to read frames from the webcam using cv.VideoCapture().
  - Face Mesh Processing: Each frame is processed using the FaceMesh model (face_model.process()). The model detects facial landmarks for all faces present in the frame.
  - Landmark Drawing: The code calls the draw_landmarks() function to draw landmarks on the image for visualization purposes.
- Drowsiness Detection: The code calculates the aspect ratios for both eyes and lips and determines if the person is drowsy based on predefined thresholds. If drowsiness is detected, it provides a warning message using text-to-speech.
- Displaying Results: The code displays the processed image with drawn landmarks, frame count, warnings given, eye ratios, and lip ratio using cv.imshow() and cv.putText().
- Loop Termination: The loop continues until the user presses the Esc key (27 in ASCII code).
- Cleanup: After the loop is terminated, the video capture is released and all OpenCV windows are closed using capture.release() and cv.destroyAllWindows().

Result:


FACE MESH
0 0
2.912
1.572


FACE MESH
3 523
5.025
19.345


FACE MESH
0 0
3.345
28.57

Conclusion:

The model for detecting tiredness can do this by observing the driver's eye movement. To pinpoint the region of interest, the model makes use of the eye's aspect ratio. The aspect ratio of the eye can be calculated using the EAR function. The alert is generated if the detection counter value exceeds the driver code-set threshold value. The main objective of this project's development is to lessen the amount of collisions brought on by fatigued drivers.

The newly designed driver abnormality monitoring system is capable of quickly spotting signs of intoxication, exhaustion, and unsafe driving habits. Drowsy driving-related accidents may be prevented using the proposed technology. If the camera generates a greater resolution, even if the driver wears glasses, the device works well in low light conditions. Various self-developed image processing methods are used to collect information about the head and eye positions. The technology can determine if the eyes are open or closed during the monitoring. A warning signal is given when the eyes are closed for an extended period of time. Continuous eye closures are used to determine the driver's alertness state.

References:

[1] https://github.com/google/mediapipe
The official documentation for the MediaPipe library, which provides various cross-platform solutions for perception tasks, including face detection and tracking.

[2] https://developers.google.com/mediapipe/solutions/vision/face_landmarker/ The MediaPipe Face Landmarker task lets you detect face landmarks and facial expressions in images and videos

[3] https://docs.opencv.org/ OpenCV is a popular computer vision library that provides tools and functions for image and video processing.

[4] https://pyttsx3.readthedocs.io/en/latest/ pyttsx3 is a Python library for text-to-speech conversion.

[5] https://youtu.be/uIcKIRtSdCs Drowsiness Detection, using Mediapipe FaceMesh a video reference.

[6] https://github.com/sant-elam/DrowsinessDetection

[7] https://github.com/sant-elam/FaceMesh