SUSTAV ZA NADZOR VOZAČA POMOĆU UGRADBENE RAČUNALNE PLATFORME

JOSIP JURAJ STROSSMAYER UNIVERSITY OF OSIJEK

FACULTY OF ELECTRICAL ENGINEERING, COMPUTING AND INFORMATION TECHNOLOGY

GRADUATE STUDY

DRIVER MONITORING SYSTEM USING AN EMBEDDED COMPUTER PLATFORM

GRADUATION THESIS

RAJESH RIMAL

OSIJEK, 2022

FERIT
FACULTY OF ELECTRICAL ENGINEERING, COMPUTER SCIENCE AND INFORMATION TECHNOLOGY **OSIJEK**

Form D1: Form for appointing members to the Committee for the graduation exam

Osijek, 14 July 2022

To the Committee for final papers and graduation exams

Appointing members to the Committee for the graduation exam

Name and surname:	Rajesh Rimal
Study programme, branch:	Graduate University Study Programme in Automotive Computing and Communications
Student's ID number: **Admission date:**	D-2ACC 13 November 2020
Student's personal identification number:	96884439837
Mentor:	Associate Professor Marijan Herceg, PhD
Co-mentor:	-
Co-mentor from the company:	Zvonimir Kaprocki
Head of the Committee:	Associate Professor Mario Vranješ, PhD
Member 1 of the Committee:	Associate Professor Marijan Herceg, PhD
Member 2 of the Committee:	Associate Professor Ratko Grbić, PhD
Title of the Master's thesis:	Driver monitoring system using an embedded computer platform
Scientific branch:	**Telecommunications and Informatics (scientific field Electrical Engineering)**

Task of the Master’s thesis:	One of the advanced driver assistance systems (Advanced Driver Assistance System) is the driver monitoring system, which detects driver fatigue or insufficient attention based on images obtained from a camera mounted in the dashboard. vehicles. Such systems, using algorithms developed in the field of computer vision, identify a person, estimate the position of the head, measure the duration of closed eyelids, frequency of yawning, etc., and based on the obtained results give an appropriate warning (e.g., sound signal together with a message on the screen dashboard). As part of the thesis, it is necessary to first design such an algorithm on a personal computer and then implement it on a suitable computer platform with a corresponding camera (NVIDIA Jetson Nano, Raspberry Pi, etc.). Furthermore, as part of the diploma work, it is necessary to carry out an appropriate verification of the system's operation on the basis of its own recorded video sequences obtained by directly mounting the platform and camera in a passenger car.
Suggested grade for the written part (Master’s thesis):	Excellent (5)
Short explanation of the grade pursuant to the Requirements for evaluating final papers and Master’s theses:	Application of knowledge and skills acquired during one’s studies: 3 points Achieved results related to the task’s complexity: 3 points Clarity and coherence: 3 points Level of independence: 3 points
Date of the mentor’s grade:	11 July 2022
Mentor's signature allowing the student to submit the final version to the Student Administration Office:	Signature:
	Date: 14 July 2022

FERIT
FACULTY OF ELECTRICAL ENGINEERING,
COMPUTER SCIENCE AND INFORMATION TECHNOLOGY **OSIJEK**

STATEMENT OF ORIGINALITY

Osijek, 14 July 2022

Name and surname:	Rajesh Rimal
Study programme:	Graduate University Study Programme in Automotive Computing and Communications
Student's ID number: **Admission date:**	D-2ACC 13 November 2020
Turnitin similarity rate [%]:	11

I hereby declare that the submitted Master's thesis entitled ***Driver monitoring system using an embedded computer platform*** was done under the mentorship of Associate Professor Marijan Herceg, PhD

And is my own work and that, to the best of my knowledge, it contains no sourses or resources other than ones mentioned and ackowlegded. I also declare that the intellectual content of this thesis is the product of my own work, except in parts where I was provided the assistance of my mentor, co-mentor or other people which is acknowledged.

Student's signature:

TABLE OF CONTENTS

1. INTRODUCTION 1

2. LITERATURE REVIEW 3

2.1. Problem Statement 3

2.2. Previous Research and Outcomes 3

3. METHODS USED FOR DETECTION OF FACE, EYE, MOUTH, AND THEIR STATES 7

3.1. Dataset used 7

3.2. Face Detection 7

3.3. Head Position Estimation 9

3.4. Eye and eye state detection 11

3.4.1. Eye Detection 11

3.4.2. Eye state detection 12

3.5. Mouth and Yawn Detection 13

3.5.1. Mouth Detection 13

3.5.2. Yawn Detection 13

4. EMBEDDED SYSTEM USED 15

4.1. Raspberry Pi 15

4.1.1. Raspberry Pi 3 Model B 15

4.1.2. Raspberry Pi Camera 16

5. ALGORITHM DEVELOPMENT AND IMPLEMENTATION 17

5.1. Objectives 17

5.2. DMS Algorithm 17

5.3. Algorithm for face detection and localization 20

5.4. Algorithm for Detecting Head Position 22

5.4.1. Calculation of Rotation Matrix and Euler angles 22

5.4.2. Head position detection 23

5.5. Algorithm for Detecting Closed Eye 24

5.5.1. Calculation of ADR 25

5.5.2. Eye state detection 25

5.6. Algorithm for Detecting Yawn 26

5.6.1. Calculation of MAR 27

5.6.2. Mouth state detection 27

5.7. System Setup 28

6. VALIDATION AND TESTING OF PROPOSED SOLUTION 30

6.1. Test in PC 31

6.1.1. Reasons for FN 32

6.1.2. Reasons for FP 34

6.2. Validation in Raspberry Pi 36

6.2.1. Execution speed in Pi 36

6.2.2. Accuracy in Raspberry Pi 37

6.3. Comparison with PC results 39

6.4. Testing using Raspberry Pi Camera 40

7. CONCLUSION 44

REFERENCES 45

ABSTRACT 48

BIOGRAPHY 49

LIST OF FIGURES 50

LIST OF TABLES 51

LIST OF ALGORITHMS 52

ABBREVIATIONS 53

ANNEXES i

1. INTRODUCTION

According to World Health Organization (WHO), approximately 1.3 million people are dying worldwide each year due to fatal road accidents of which most cases involved overspeeding, reckless driving, alcohol and drug consumption, and distracted and drowsy driving [1]. In Europe, data published by European Union (EU) show that fatalities in traffic accidents were around 19.800 in 2021 which is 5% more than in 2020 [2]. National Highway Traffic Safety Administration (NHTSA) in the USA 2021 claimed there were about 42.915 injuries in road accidents nationwide which is 10.5% more than in 2020 [3]. These road accidents not only cause the loss of lives but also lifetime injuries and disabilities, property damage, and economic losses. In vehicles, for the control of such accidents, the introduction of the driver assistance systems like anti-locking systems, engine control systems, adaptive cruise control, driver monitoring systems, lane detection, departure warning system, etc is increasing. These systems form the basis of the Advanced Driver Assistance Systems (ADAS). These systems are installed in the vehicle to assist the driver in some cases (ADAS Level 1 or Driver Assistance) or to take full control of the vehicle and perform the driving task under all the road and environment conditions (ADAS Level 5 or Full Automation). The ultimate goal of the ADAS is to assist the driving or take control of the vehicle or alert the driver in dangerous situations. This system reduces the accidents caused by human factors which are estimated to cause more than 90% of accidents [4] that include loss of control of the vehicle due to fatigue, distraction, etc. However, only a few vehicle manufacturers have achieved the ADAS level 3 or are conditionally automated. ADAS level 3 is the automation level in which the dynamic driving task is performed by the automated driving system with the data obtained from monitoring the environment. The dynamic driving task includes the task like steering, braking, accelerating, decelerating, and monitoring the vehicle with additional tasks like lane departure, taking a turn, using signals, responding to events, etc. Level 3 autonomous vehicle is capable of driving in certain driving modes and in case of any incapability of the system to drive, it expects the human driver to respond to the request of intervention [5]. The ultimate goal of these levels is to minimize and control road accidents that are caused by human error. So, in the vehicles that will be manufactured soon or in existing ones, if some systems that can assist or alert the driver in unusual driving conditions are installed, then the road accidents due to driver's action can be reduced.

One of the approaches to reduce road accidents due to driver's action is the Driver Monitoring System (DMS) which is one of the most commonly used or will be compulsorily used in the coming days. These systems are installed in the vehicle that monitors the series of actions or conditions of the driver and warns the driver in case of inattention, distraction, fatigue, deprived sleep, etc. Driving inattention is the degree of non-concentration while driving due to the use of gadgets, fatigue, distraction, etc. Fatigue and lack of sleep cause drowsiness and driving in this condition will be threatening. The fatigued or sleep-deprived driver can fall asleep in the middle of long drives causing accidents that may lead to the loss of life. Also, distraction is another issue that if not warned may lead to accidents. The DMS aims to alert the driver with continuous warning messages and sounds if the fatigue conditions like sleepiness and nodding head down, closing of eyes for a long time, yawning frequently, and the distraction conditions like looking sideways are observed. If these actions are regularly observed from a camera installed on the dashboard of the vehicle or the physiological status of the driver is regularly monitored, alerted in case of abnormalities, or even ask the driver to stop and take rest for some time, the accidents due to drowsiness and distractions can be reduced.

In section 2, a brief discussion of the existing methods of DMS approaches is discussed, in section 3, the dataset used and methods for detecting the face, mouth, eyes, and their states are described. In section 4, a description of the embedded system used is provided. In section 5, the algorithms for the individual, integrated modules of DMS, and system setup are provided. Section 6 presents the testing results of the developed DMS algorithm in PC and Raspberry Pi and a summary is presented in section 7.

2. LITERATURE REVIEW

In this section, a review of the different methods that are existing in the field of driver monitoring is given.

2.1. Problem Statement

The concept of driver monitoring is a new feature and is only used in vehicles of some manufacturers like BMW, Ford, General Motors, Tesla, and Subaru. Although these manufacturers claim that their system will detect and prevent driver drowsiness and inattention, the tests performed by the Consumer Reports team uncovered various deficits in the system and were marked as not safe for driving [6]. The European Union and China are on their way to making DMS mandatory in all new vehicles, while the United States is investing trillions of dollars in all-around safer roads. A general safety regulation passed by the EU Council of Ministers mandates the installation of all the necessary advanced safety systems in all new cars in the EU market. The advanced safety system comprises the camera-based DMS to detect the drowsiness or inattention of the driver and alert the driver in case of any distraction is observed. The regulation will be implemented over four years. First, starting in 2022, the regulation is enforced on all new type-approved cars with certain ADAS levels. However, by the end of 2026, the regulation will be employed for all the newly produced cars of all ADAS levels. This regulation aims to avoid at least 140.000 serious injuries by 2038. The United States on July 1st, 2020, passed the Moving forward Act that will force the installation of the technology that detects inattentive and intoxicated drivers in a newly produced vehicle [7]. Hence, the new types and methods of the DMS will be coming into effect that can be an effective approach to reducing the accidents and fatalities caused.

2.2. Previous Research and Outcomes

Different approaches were discussed and presented in the previous years in the field of DMS. The various existing methods are presented in [8] for the DMS. The first method is the subjective method in which sleepiness is measured by different surveys, questionnaires, tests, and electrophysiological measures, and the obtained data is interpreted to predict the factors that are leading to a vehicle accident. These data provide the means for other methods to focus while detecting and preventing factors that are related to the driver's drowsiness. The second method is the physiological method which provides a more objective method to measure the drowsiness level by measuring the physiological signals from the human body like electrocardiogram (ECG), electroencephalogram (EEG), and electrooculogram (EOG). This method is more reliable and accurate compared to other methods, however, using it in a real-time application in the vehicle is difficult and more intrusive for drivers. The third method is the vehicle-based method which includes the subjective analysis of data based on real pieces of evidence like accident reports, driver survey reports, police reports, etc. following the event and suggests the typical driver's and vehicle's behavior during these events exhibit certain characteristics. These characteristics imply that a vehicle involved in an accident driven by a drowsy driver yields distinct driving patterns that can be measured and then used to detect a possible drowsy driving situation. The next one is the behavioral approach which includes the detection of the face of the drivers, their eyes, and their mouth to predict the abnormalities of the driver based on their state. The states may be face detected or not, turned up/down/left/right/forward, opened or closed eyes and mouth. The final method is a hybrid that includes the fusion of different methods. The most common approach is the behavioral method where different methods are used to detect the face, eyes, and mouth and algorithms for detecting their states and behaviors.

In [9], the authors have proposed a face detection method based on the Viola-Jones (VJ) algorithm that is widely used in many applications including DMS. The authors presented an approach to develop the face detection method that was about 15 times faster than the other methods existing at that time. The algorithm is based on selecting the Haar Features, creating an integral image, AdaBoost training, and cascading classifiers. Although the VJ detector is widely used due to its fast detection speed, effective feature selection, and invariance on scale and location, it possesses some disadvantages also. For example, VJ detectors cannot detect the rotated face, they are sensitive to lighting conditions, and can detect the front face only.

In [10], a real-time face detector is developed with higher accuracy and real-time speed known as FaceBoxes. The main advantage of this method is a real-time execution speed that can be achieved on the central processing units (CPU). The method is using the convolutional neural network (CNN) with two types of convolutional layers; Rapidly Digested Convolutional Layer (RDCL) and the Multiple Scale Convolutional Layers (MSCL) with two and four convolutional layers in each type respectively. Additionally, the anchor densification strategy contributes to making this method efficient and accurate in CPUs. This method claims to have the accuracy of 98.91% and

96.30% in Annotated Faces in-the-wild (AFW)[11] and PASCAL face datasets respectively with 20 frames per second (fps) for video graphics array (VGA)-resolution i.e. 640 x 480 images in CPU and 125 fps in the graphical processing unit (GPU).

In [12], the authors have presented a method known as a single-shot multibox detector (SSD) for object detection with a CNN to produce a fixed-sized gathering of bounding boxes and score if the objects are present in those boxes followed by non-maximum suppression to finally detect the objects. It uses visual geometry group (VGG)-16 as a base network but other networks also can be used. A similar approach to detecting the face can be applied if a model is trained on the face datasets. The main drawback of this method is that it cannot detect small objects and needs a larger number of data to train the model.

In [13], a deep learning approach to the detection and alignment of the face is proposed. The multitask cascaded neural network (MTCNN) model of three stages is proposed which works for face detection and alignment with higher accuracy. The three stages of this model are proposal network (P-Net) to obtain candidate face window and approximate bounding box regression vectors, refine network (R-Net) to refine the windows and sort out the false one, and O-net to finally obtain the bounding box and five facial landmark positions. The speed of this method was obtained to be 99 fps in GPU for VGA-resolution images and only 16 fps on 2.60GHz CPU.

A solution named MediaPipe FaceMesh that detects the face at first and extracts the 468 facial landmarks points are proposed in [14]. This solution is developed by Google and designed to work in smartphones, CPU, and GPU devices. It uses the machine learning algorithms to first detect the face based on BlazeFace [15], crops and inputs the detected face to the face landmark model and extracts the facial landmarks from the facial surfaces based on [16]. This method aims to achieve high performance in devices with both CPU and GPU.

In [17], the author provided the Dlib model trained to detect and identify 68 facial landmarks using Dlib’s pre-trained facial landmark predictor. There are two shape predictor models with one localizing 68 and the other 5 facial landmark points. These models are trained in the i-Bug 300 W dataset [18]. The detection is based on the Histogram of Oriented Gradients (HOG).

The authors in [19] provided the method of the detection of the blinks using facial landmarks. This paper suggested eye aspect ratio (EAR) as a measure of determining the state of the eye i.e. whether it is closed or open. EAR is the ratio of the height and the width of the eye. The EAR is assumed to be constant when it is open and almost close to zero when it is closed. In [20], the authors have also used a similar concept to determine the state of the eye. This method involves the detection of the eye's region, illumination normalization, binarization with adaptive thresholding, calculating the ratio of height to width of the binarized eye, and cumulative difference in the number of black pixels of the binarized eye region in successive frames. The blink is detected by combining the above-discussed method based on a support vector machine (SVM).

In [21], the authors have presented the DMS in which the facial expression of the driver is captured through the video sequences using a camera placed on the dashboard. The face and eyes were detected using pre-trained detectors based on the VJ algorithm. The algorithm monitors the detected face and eyes to estimate the normal positioned head and eye state. A separate pre-trained detector was used to detect the left and right turn of a face.

In [22], the authors have also used the Viola-Jones algorithm to detect the face at first, then localize the facial points, and then construct the 3D model of the face based on the 2D localized points to estimate the head position. Different methods like simultaneous modeling and tracking (SMAT) were used to localize the 2D points and point iterative pose estimation (POSIT) and random sampling and consensus (RANSAC) were used to construct the 3D model from the localized points. The 3D model developed is used to estimate the pose of the driver. Also, the Percentage of Closure (PERCLOS) parameter was used to detect the eyes state.

The authors in [23] have developed their CNN classification model that detects the state of the eye and decides if the driver is drowsy or not. The model is developed by using the dataset of 48.000 eye images from the MRL eye dataset [24] from Kaggle and training datasets under a CNN model based on ResNet architecture with three convolutional layers. For detecting the eye state in real-time, the eye region is detected and localized using the Google MediaPipe detector from each frame captured by the webcam, cropped, and fed to developed CNN to detect the eye state whose accuracy was obtained as 95%.

3. METHODS USED FOR DETECTION OF FACE, EYE, MOUTH, AND THEIR STATES

In this section, the methods that are used in this thesis will be discussed. In particular, the datasets used, detectors for face, eye, and mouth, and their states will be described and discussed.

3.1. Dataset used

Three different datasets for Driver monitoring were analyzed. The first dataset used is DMD (Driver Monitoring Dataset) [25], created within the framework of the Vision Inspired Driver Assistance System's (VI-DAS) project led by Vicomtech, which is an organization working on applied research in digital technologies. This dataset consists of 20 videos taken from the camera in the driver's dashboard and has 4 different male individuals. Next is the YawDD [26] dataset which consists of 29 videos taken from the camera installed in a driver's dashboard, has both male and female, and each video contains actions like driving silently, yawning, eyes closing, talking, etc. The third dataset is DrivFace [27] which contains images of 4 individuals driving in real-time and has 2 males and females each. The total sample in this dataset is 606. The detailed comparison of the dataset is shown in Annex I.

3.2. Face Detection

For face detection, detectors like FaceBoxes, SSD, MediaPipe FaceMesh, MTCNN, DLib, and VJ algorithm are used and compared. The performance of each detector is based on the time needed to detect a face in each frame and the accuracy of detection of the face. At first, all the detectors were run where Dlib, MTCNN, and VJ were slower compared to other detectors. So, the accuracy was not calculated for these detectors. The accuracy of the detector is measured using parameters like precision, recall, and F1-Score. The detector with the highest F1-Score will be the most accurate. The precision, recall, and F1-Score are calculated as shown in equations 3.1 to 3.3.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

$$F1 - Score = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \quad 3.3$$

where TP represents true positives, FP represents false positives and FN represents false negatives.

Based on the time taken to detect a face in each frame and the F1-Score, MediaPipe FaceMesh is selected for the face detection of the driver which is shown in Table 3.1.

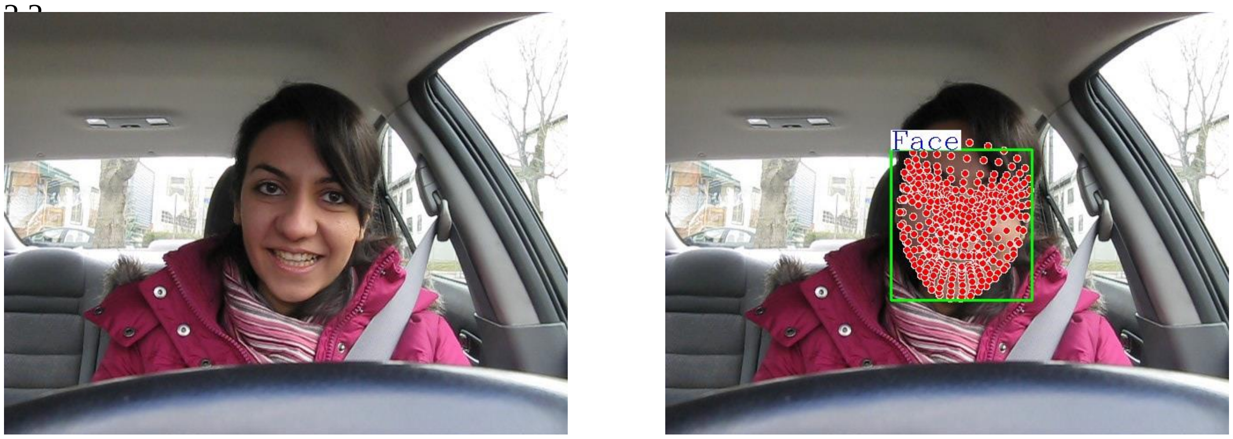
Table 3.1 Comparison of face detectors based on F-1 score and time to detect face

Detectors	Dataset Name	Frames	Time per Frame (ms)	TP	FP	FN	Precision	Recall	F1-Score
FaceBoxes	YawDD	64593	37.736	64586	0	0	1.0000	1.0000	1.0000
	DMD	38574	38.236	38471	55	7	0.9986	0.9998	0.9992
SSD	YawDD	64593	26.685	63314	0	1272	1.0000	0.9803	0.9901
	DMD	38574	31.993	30299	19	8184	0.9994	0.7873	0.8808
MediaPipe FaceMesh	YawDD	64593	9.114	64564	0	22	1.0000	0.9997	0.9998
	DMD	38574	7.098	38383	0	104	1.0000	0.9973	0.9986
Dlib	YawDD	26166	175.4	-	-	-	-	-	-
VJ	YawDD	26116	41.518	-	-	-	-	-	-
MTCNN	YawDD	2741	943.635	-	-	-	-	-	-

The MediaPipe FaceMesh has a high speed and high accuracy in CPU-based devices. The main advantage of using this method over other methods is its high speed, and good accuracy, and most important, it localizes the facial points after detection. Unlike other detectors, it is not needed to use a separate different detector to localize the facial points. The MediaPipe FaceMesh is an end-to-end neural network-based model that estimates 468 3D facial points in real-time. Only with the use of a single camera and without the need for a depth sensor, these 3D facial points are estimated. The input to the model is the RGB frame from a camera or a stream of frames. From the frame, the face is detected using the BlazeFace [15]. The BlazeFace detector is a lightweight detector that produces the face bounding box with several landmarks like eyes center, nose tip, etc. These landmarks are used to align the face bounding box rectangle horizontally with the line joining the center of the eyes. The rectangle bounding box is cropped from the original image and resized from 256x256 in the full model to 128x128 in the smallest model. This will be the input for the mesh prediction which produces the vector of 3D landmark coordinates.

For the mesh prediction, the model uses the straightforward residual neural network architecture. In first layers, the more aggressive subsampling is used and the majority of the computation is dedicated to its shallow part i.e. the layers closer to the input layer. As a result, the neurons' receptive fields begin to cover large areas of the input image comparatively early. When such a receptive field reaches the image boundary, the model can rely on its relative location in the input image (due to convolution padding). As a result, the neurons in the deeper layers are likely to distinguish between mouth-relevant and eye-relevant features. This leads to the construction of a high-level and low-dimensional mesh representation. Thus, a mesh is turned into the coordinates in the last few layers of the network. The training of this model is done in the 30K in-the-wild mobile camera photos taken using different sensors in different lighting conditions.

In Figure 3.1, the results of the MediaPipe FaceMesh algorithm are shown.



(a) (b)

Figure 3.1: FaceMesh face detection [26] (a - Original Image, b – Image of detected face in the green bounding box and 468 localized facial landmarks in red dots)

3.3. Head Position Estimation

Head position estimation is the determining the position or orientation of the head with respect to the camera. The head position estimation used in this thesis is based on the Perspective-n-Point (PnP) concept. If the camera is placed at a fixed position, there are two types of motion of an object with respect to the camera; translation motion i.e. moving the object from its current 3D location (x, y, z) to some new 3D location (x', y', z'), and rotational motion where the object can rotate about camera's X, Y and Z axes that can be represented by Euler angles (roll, yaw, and pitch) which is shown in Figure 3.2. In this thesis, the head position is calculated only based on the yaw and pitch of the head from a normal position. The normal position is the position where the driver looks forward towards the road. The yaw is the turn of the head in the left and right direction while the pitch is the turn of the head in the up and down direction and the roll is the turn of the head in the down-right or down-left direction.

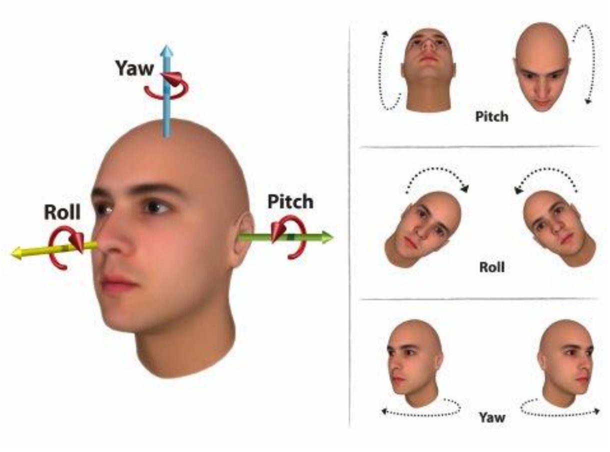


Figure 3.2 Head orientation showing pitch, roll, and yaw movements [28]

If the two coordinates system are available i.e. 2D coordinates (u,v) from the image plane and 3D world coordinates (x, y, z), then using the intrinsic parameter of the calibrated camera, the rotational and translational vectors, also known as extrinsic parameters, are calculated. The PnP problem is governed by Equation 3.4.

$$s\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & \gamma & u_0\\0 & f_y & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} \tag{3.4}$$

Where u and v are the coordinates of the image plane, f_x and f_y are the focal length of the camera, γ is the skew parameter, u_0 and v_0 are the principal points, and s is the scale factor for the image point as the depth is unknown. The r_{11} to r_{33} and t_1 to t_3 are the required rotational and translational vector elements. The rotational vector is converted into a rotational matrix and from the obtained rotational matrix, Euler angles can be obtained as pitch (θ), yaw(ψ), and roll (φ) of a driver. The rotational matrix obtained is in the form represented by Equation 3.5. Also, the rotations of ψ, θ, and φ radians about the 3 principle axes *X, Y,* and *Z* can be defined by standard definitions as represented in Equations 3.6 to 3.8 respectively.

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{3.5}$$

$$\mathbf{R_x}(\boldsymbol{\psi}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{3.6}$$

$$\mathbf{R_y}(\boldsymbol{\theta}) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.7}$$

$$\mathbf{R_z}(\boldsymbol{\varphi}) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

Where **R** is the obtained rotation matrix, **$\mathbf{R_x}$(ψ), $\mathbf{R_x}$(θ)**, and **$\mathbf{R_x}$(φ)** are the rotation about the *X, Y,* and *Z*-axis respectively. By equating the elements of the rotation matrix R with the elements of the dot product of Equations 3.6, 3.7, and 3.8, the Euler angles can be calculated.

3.4. Eye and eye state detection

3.4.1. Eye Detection

The eye-detection method used in this thesis is also based on MediaPipe's Face Mesh. Once the face is detected and localized, out of 468 3D facial points, 16 points are used to locate each eye. The facial points that surround the left eye are 33, 246, 161, 160, 159, 158, 157, 173, 133, 155, 154, 153, 145, 144, 163, 7 and the right eye are 362, 398, 384, 385, 386, 387, 388, 466, 263, 249, 390, 373, 374, 380, 381, 382. Once these points are located its will be used in further processing of the frames for the driver monitoring algorithm. The detection of the eyes and the localization of the points that surrounds the eyes is shown in Figure 3.3.

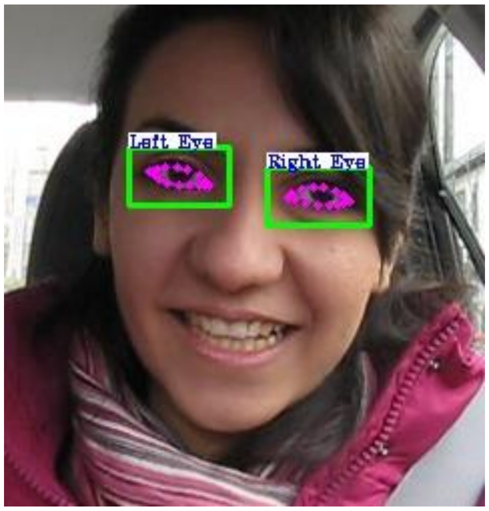


Figure 3.3: Detected eyes in green bounding boxes and localized pink points [26]

3.4.2. Eye state detection

The eyes state detection is based on a similar concept as EAR, but here the area of the eyes and the distance between them is used as input parameters. This method for detection of the eye state is also scale-invariant as the EAR method. The eye facial points are used to calculate the area over the distance ratio (ADR) value in each frame. The ADR is calculated using the formula stated in Equation 3.9.

$$ADR = \frac{\sqrt{S1} + \sqrt{S2}}{2 * d} \qquad 3.9$$

where *S1* and *S2* represent the area of the left and right eye respectively and *d* represents the distance between the center of the two eyes as shown in Figure 3.4. The ADR value calculated is compared with the threshold ADR value to detect the eye state i.e. whether closed or open.

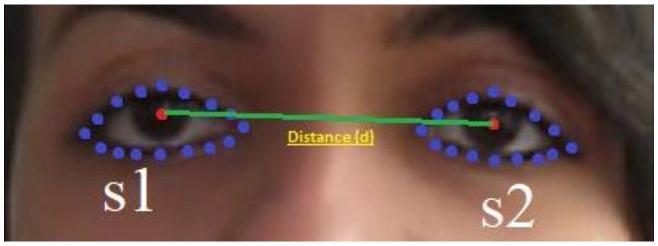


Figure 3.4 Eye Area over the Distance calculation [26]

3.5. Mouth and Yawn Detection

3.5.1. Mouth Detection

Similar to eyes detection, the mouth detection method used in this thesis also uses the same facial landmark points from MediaPipe FaceMesh. 20 facial landmarks can be obtained from the 468 points that surround the mouth. With these facial points, the mouth is detected and localized. The facial landmarks that surrounds the mouth are 62, 183, 42, 41, 38, 12, 268, 271, 272, 407, 292, 325, 319, 403, 316, 15, 86, 179, 89, 96. The detection of the mouth and the localization of the points that surround the mouth is illustrated in Figure 3.5.

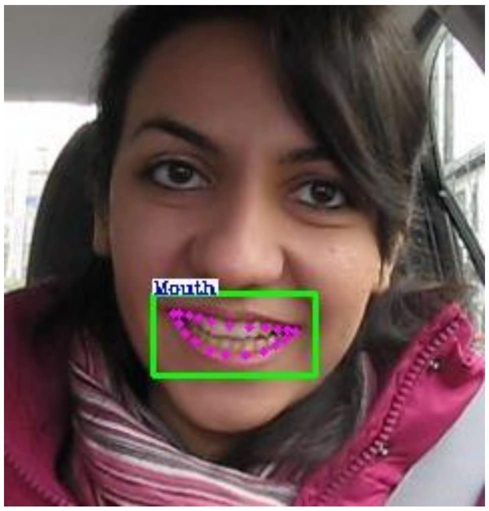


Figure 3.5 Detected mouth in the green bounding box and localized pink points [26]

3.5.2. Yawn Detection

Yawn detection proposed in this thesis is based on the calculation of the mouth aspect ratio (MAR) which represents the ratio of the height to the width of the mouth. It is similar to the calculation of EAR and is calculated using the formula stated in Equation 3.10

$$MAR = \frac{|(p4 - p16)| + |(p6 - p14)|}{2 * |(p0 - p10)|} \qquad 3.10$$

where *p0, p4, p6, p10, p14,* and *p16* represent the coordinates represented in (*x*-coordinate, *y*-coordinate) of the facial landmarks that surround the mouth. Out of 20 points, only 6 are taken to calculate the MAR value. The arrangement of those 6 points is shown in Figure 3.6.

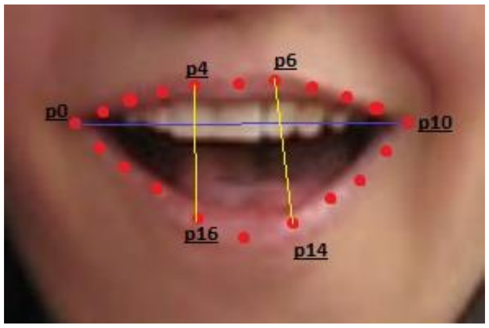


Figure 3.6 Arrangement of points for calculation of MAR [26]

4. EMBEDDED SYSTEM USED

In this thesis, the proposed DMS is implemented in Raspberry Pi.

4.1. Raspberry Pi

Raspberry Pi is a small credit-card-sized single-board computer developed by Raspberry Pi Foundation. The first-generation Raspberry Pi Model B was launched in February 2012. It became extremely popular due to its price (around $35) and small size. After the popularity of the first-generation model, there were several generations of Raspberry Pi released. The board comes with a Broadcom system on a chip (SoC) consisting integrated ARM processor (CPU) and a built-in GPU. The processor clock speed is moving from 700 MHz to 1.4 GHz for Pi 3B+ or 1.5 GHz for Pi 4. Secure Digital (SD) cards are used for the storage of the operating system and programs. There are up to four USB ports and an HDMI port mounted on the board. Using GPIO pins standard protocols such as I2C are supported. All B models have an Ethernet connection, while the Pi 3 and Pi Zero W have Wi-Fi 802.11n and Bluetooth.

4.1.1. Raspberry Pi 3 Model B

The Raspberry Pi 3 Model B was released in 2016 with a 1.2 GHz 64-bit quad-core processor and built-in 802.11n Wi-Fi and Bluetooth capabilities. This model is equipped with Broadcom BCM2837 SoC with an integrated quad-core ARM Cortex-A53 CPU with a clock speed and Broadcom VideoCore IV GPU. The RAM available is 1 GB. The network support includes 10/100 Ethernet and 2.4GHz Wi-Fi 802.11n. It is also equipped with the Bluetooth 4.1 Classic. The available Input/Output are 4 USB 2.0, HDMI, 3.5 mm analog connection, Camera Serial Interface (CSI), and Display Serial Interface (DSI). The board is supplied by a 5V power supply through a USB micro connector. The board also consists of a 40 GPIO pin-header with I2C and SPI support [29]. The image of the model used is provided in Figure 4.1

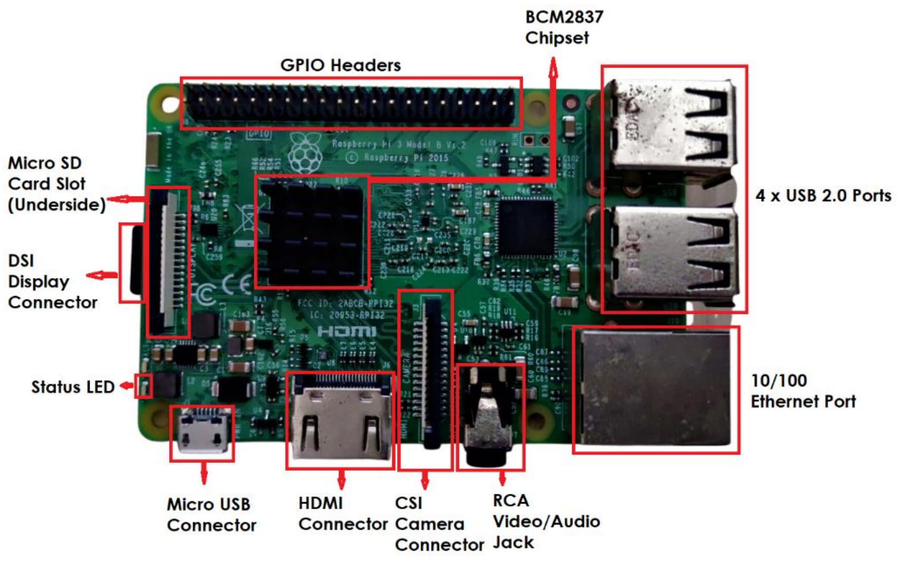


Figure 4.1 Raspberry Pi 3 Model B board specifications

4.1.2. Raspberry Pi Camera

The Raspberry Pi Camera module used in this thesis is a high-definition camera that is compatible with both Raspberry Pi models A and B. This camera connects to the board via the CSI connector with the 15-pin ribbon cable. It is a 5-megapixel camera with 2592 x 1944 still picture resolution. The maximum image transfer rate is 1080p at 30 frames per second (fps) and 720p at 60 fps. The additional feature includes automatic exposure control, white balance, band filter, 50/60 Hz luminance detection, and black level calibration [30].

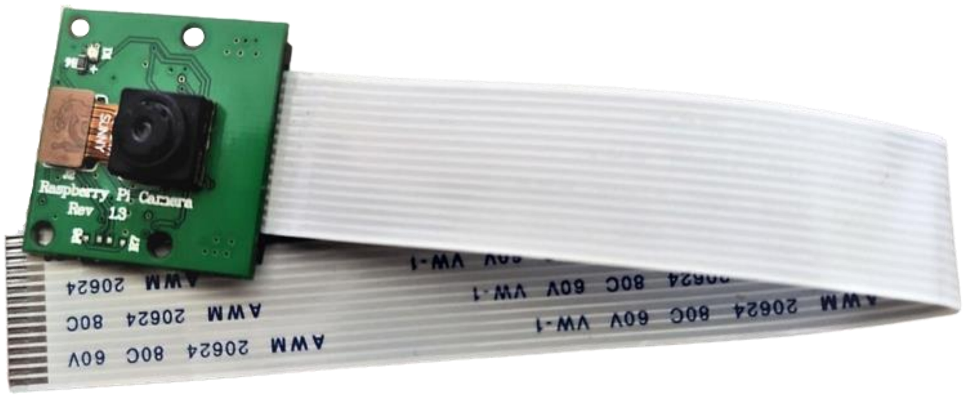


Figure 4.2 Raspberry Pi Camera Module

5. ALGORITHM DEVELOPMENT AND IMPLEMENTATION

5.1. Objectives

The main objective of this thesis is to develop a DMS in a PC environment and test it in real-time in Raspberry Pi. This ADAS involves the system of continuous monitoring of the driver and raises the alarm if any inattention or drowsiness is observed. Inattention in driving refers to turning sideways and not paying attention to the road, and drowsiness refers to the condition where the driver is fatigued and feels sleepy. The drowsiness can be measured by the frequency of yawning, frequency of blinking the eyes, looking down for a long time, eyes closed for a long time, etc. As a part of the graduate thesis, a behavioral method of detecting the drowsiness or inattention of a driver is proposed. The specific objectives of the project are:

a. Develop the algorithm to detect the face, eyes, and mouth of the driver.
b. Based on the detected face, eyes, and mouth, develop the algorithm that detects the head position (looking forward, turning left/right/up/down), the eyes state i.e. opened and closed, and warn the driver if it is closed for a long time. Furthermore, detecting the yawing, detecting the frequency of yawning, and warning the driver if the frequency is high.
c. Integrate the individual algorithms to develop a working DMS.
d. Implement the integrated module to the embedded system (Raspberry Pi) and test the developed model and optimize it to run in real-time conditions.

5.2. DMS Algorithm

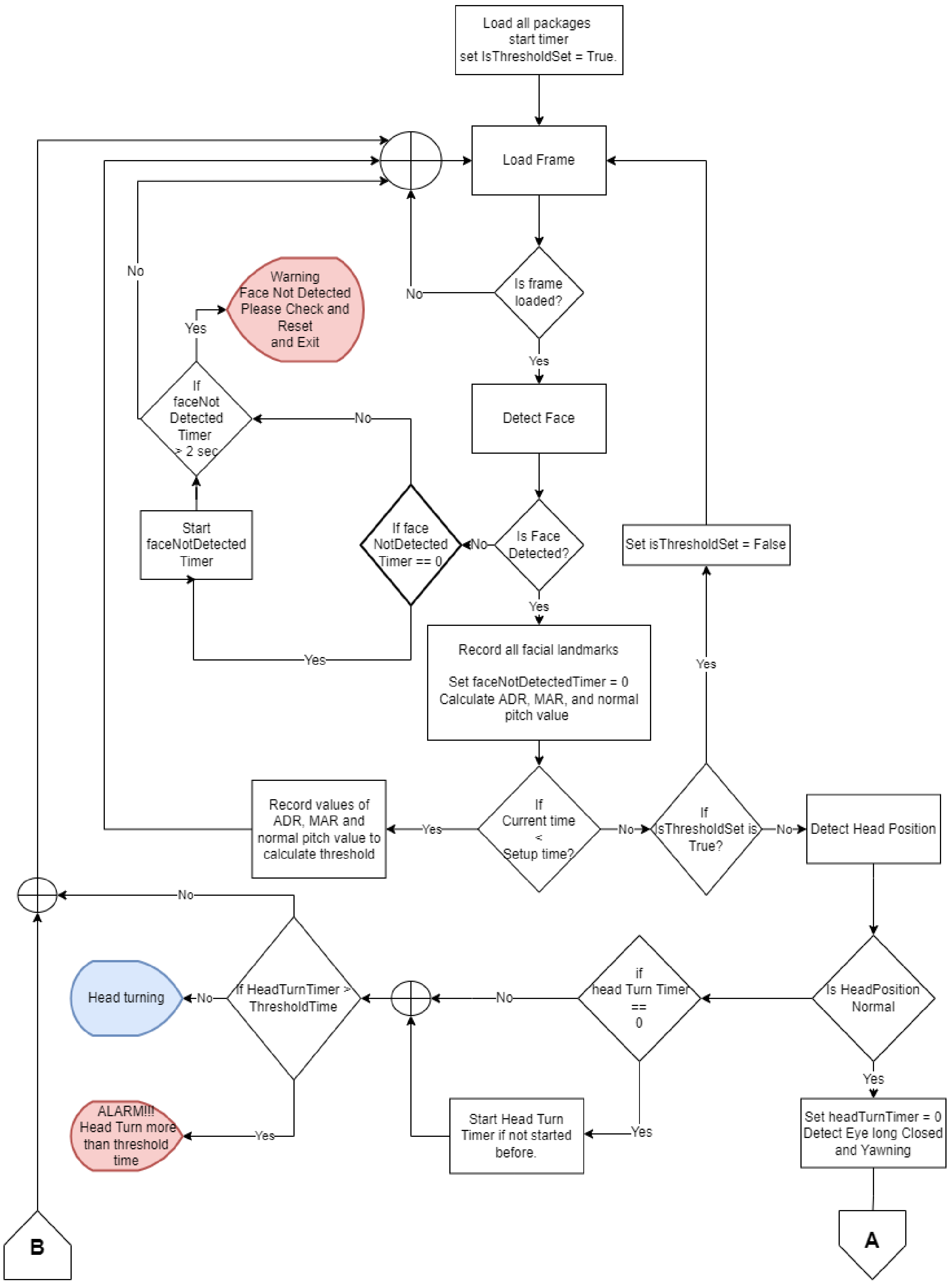
For detecting the face, eyes, and mouth and their states, as discussed in sections 3.2 to 3.5, MediaPipe FaceMesh is used. The complete algorithm of DMS is shown in Figure 5.1. The DMS takes the frame from the camera or video sequence and displays appropriate warning signs if any abnormal behavior is detected in the driver. The system starts with loading a frame from a camera or recorded video sequence and starts a timer. When the frames are loaded then, the first step is to detect the face using the algorithm *DETECTFACE*. If the face is detected, then the facial landmark points are localized and recorded for further use and if not, then the next frame is loaded. Also, the face not detected counter is started and the system warns the driver with the warning message "Face is not detected. Check and reset". The recorded landmarks are used to calculate the pitch, ADR, and MAR values using functions *HEAD_POSITION_DATA*, *COMPUTE_ADR*, and

COMPUTE_MAR respectively. The calculated values are stored till the time elapsed is less than 10 seconds. In the frame in which the time elapsed is more than 10 seconds, threshold MAR, ADR, and face normal pitch are calculated. This is done once only exactly after 10 seconds is over.

Moreover, if the time elapsed is more than 10 seconds and the threshold is set, the algorithm detects the head position as described using the function *DETECT_HEAD_POS*. If the head position is not in normal position, the head turn timer counter is started. The timer increases time in each frame and resets when the head is back to its normal position. If the head is turned for more than or equal to the threshold time, then the alarm is raised as "ALARM!!! Head Turn more than threshold time". If the head position is in a normal state, the eyes and mouth state are detected.

The next detection is the yawn detection, where the calculated MAR in each frame is compared to the threshold MAR. If the calculated MAR is greater than the threshold MAR, the algorithm detects the mouth state as open. However, for a yawn to complete, the mouth should gradually open for more than or equal to the threshold time. The yawn is detected if the mouth is found open for consecutive frames for at least a threshold time. When in the frame the mouth is detected as open, the yawn time counter starts and continues to increase time in the consecutive frames till the mouth is in an open state. If the yawn last for more than or equal to threshold time, the yawn is detected. The frequency of yawning is calculated and warned the driver with the message "Yawning too much. Please rest" if the driver is yawning more than the predefined value. The yawn detection is done using the function *DETECT_YAWN*.

Finally, the last detection involves the detection of the eyes state and warning the driver if the eye is found closed for a long period i.e. greater than the threshold time. The function *DETECT_EYE_LONG_CLOSED* is used to detect eye state and raise a warning if needed. The ADR value in each frame is compared to the threshold ADR and the eye state is detected as closed if the ADR value is less than the threshold. The eye close counter is started which only resets if the eye is in an open state. So, if the eye is found closed in a close state for more than Threshold Time, then the alarm is raised as "Eye Closed more than threshold time"


Load all packages
start timer
set IsThresholdSet = True.
Load Frame
Is frame loaded?
No
Yes
Detect Face
Is Face Detected?
No
Yes
If face NotDetected Timer == 0
Yes
Start faceNotDetected Timer
If faceNot Detected Timer > 2 sec
No
Yes
Warning
Face Not Detected
Please Check and Reset
and Exit
Record all facial landmarks
Set faceNotDetectedTimer = 0
Calculate ADR, MAR, and normal pitch value
If Current time < Setup time?
Yes
Record values of ADR, MAR and normal pitch value to calculate threshold
No
If isThresholdSet is True?
Yes
Set isThresholdSet = False
No
Detect Head Position
Is HeadPosition Normal
Yes
Set headTurnTimer = 0
Detect Eye long Closed and Yawning
A
if head Turn Timer == 0
Yes
Start Head Turn Timer if not started before.
No
If HeadTurnTimer > ThresholdTime
No
Head turning
Yes
ALARM!!!
Head Turn more than threshold time
No
B

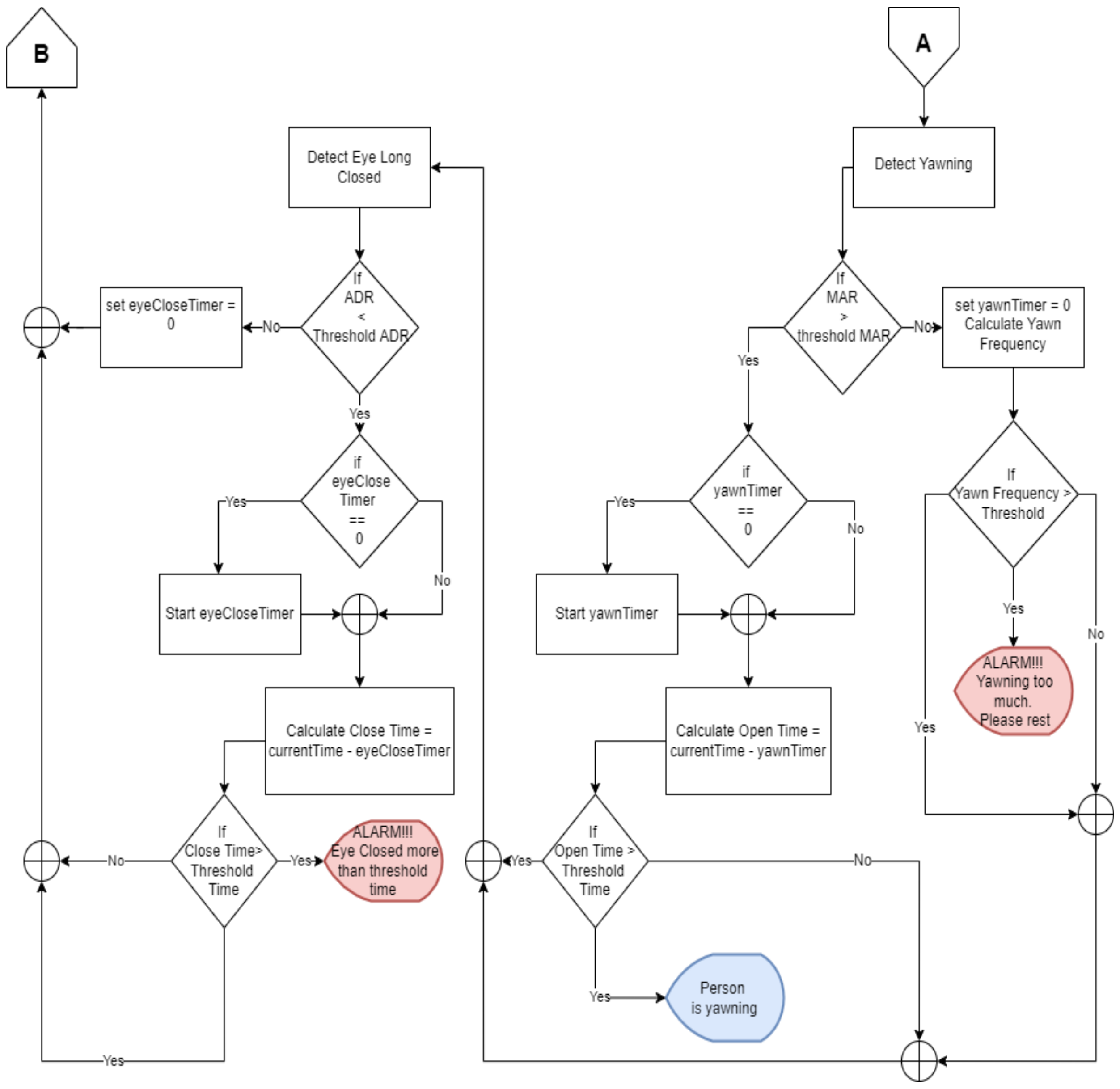

B
A
Detect Eye Long Closed
Detect Yawning
If ADR < Threshold ADR
If MAR > threshold MAR
set eyeCloseTimer = 0
set yawnTimer = 0 Calculate Yawn Frequency
No
Yes
if eyeCloseTimer == 0
if yawnTimer == 0
If Yawn Frequency > Threshold
Start eyeCloseTimer
Start yawnTimer
ALARM!!! Yawning too much. Please rest
Calculate Close Time = currentTime - eyeCloseTimer
Calculate Open Time = currentTime - yawnTimer
If Close Time> Threshold Time
If Open Time > Threshold Time
ALARM!!! Eye Closed more than threshold time
Person is yawning

Figure 5.1 DMS flow chart

5.3. Algorithm for face detection and localization

The function *DETECTFACE* takes a frame from the video sequence, detects the face if there is any, and localizes the facial points. Furthermore, it also consists of a branch that checks the condition of the setup is complete or not. If the setup is complete, the function *DETECT_HEAD_POS* is called to detect the head position. If the head position is normal, *DETECT_YAWN* and *DETECT_EYE_LONG_CLOSED* are called for detection of eye closure and yawn. The brief working of the algorithm *DETECTFACE* is given in Algorithm 1.

Algorithm 1 Face detection and localization

Algorithm 1: DETECTFACE

```
Input   :  The individual frame from the video sequence
Output  :  Detected face with landmarks coordinates if the face is available
begin
 1  idxLeftEye ←[33, 246, 161, 160, 159, 158, 157, 173, 133, 155, 154, 153, 145, 144, 163, 7]
 2  idxRightEye ←[362, 398, 384, 385, 386, 387, 388, 466, 263, 249, 390, 373, 374, 380, 381, 382]
 3  idxMouth ← [62, 183, 42, 41, 38, 12, 268, 271, 272, 407, 292, 325, 319, 403, 316, 15, 86, 179, 89, 96]
 4  startTime ← time.time()
 5  cap ← cv2.VideoCapture (FilePath) ("0" for camera)
 6  while True:
 7      ret, image ← cap.read()
 8      if not ret:
 9          break
10      else:
11          image ←  cv2.resize(image,(640,480))
12          results ← face_mesh.process(image)
14          if results.multi_face_landmarks:
15              for face_landmark in result.multi_face_landmark:
16                  landmarks ← load landmark coordinates (x,y,z)
17                  eyes_left ← calculate coordinates of the left eye as per idxLeftEye
18                  eyes_right ←calculate coordinates of the right eye as per idxRightEye
19                  mouth ← calculate coordinates of mouth as per idxMouth
20                  MAR ← CALCULATIONS.COMPUTE_MAR(mouth)
21                  ADR ← CALCULATIONS.COMPUTE_ADR (eyes_left, eyes_right)
22                  pitch, yaw, roll ← HEAD_POSITION.HEAD_POSITION_DATA (image.shape, landmarks)
23                  if (time.time()-startTime)< self.setup_time:
24                      YAWN_DETECTOR.SETUP_MAR(MAR)
25                      EYE_LONG_CLOSED_DETECTOR.SETUP_ADR (ADR)
26                      HEAD_POSITION.SETUPPITCH(pitch, yaw, roll)
27                  else:
28                      text = HEAD_POSITION.DETECT _HEADPOS(image, pitch, yaw, roll, frame_count, threshold)

                        if  text == "Normal Position":
29                          EYE_LONG_CLOSED_DETECTOR. DETECT_EYELONGCLOSED(image, ADR, frame_count, threshold)
```

```
                    YAWN_DETECTOR.DETECT_YAWN (image, MAR, frame_count, threshold)
            else:
                print("Face Not Detected")
end
```

5.4. Algorithm for Detecting Head Position

The head position detection is based on the PnP concept and involves the coordinates of the face from the image plane, the real 3D coordinates of the face, and the camera intrinsic parameters. The OpenCV's *solvePnP* function is used to calculate the rotational matrix, from which Euler's angles are calculated and compared to the threshold to warn the driver if any abnormal position of the head is found.

5.4.1. Calculation of Rotation Matrix and Euler angles

For the calculation of the rotation vector using *solvePnP* for each frame, 36 points scattered around are chosen for which the coordinates (*u*,*v*) from the image are obtained. The 3D coordinates are obtained from the generic face model proposed by MediaPipe licensed under Apache License, Version 2.0. The 3D coordinates (x, y, z) are obtained for the same 36 landmark points as in the image plane. The next parameter in *solvePnP* is the camera matrix and distortion matrix. The camera matrix is given by the intrinsic camera parameters as stated in Equation 3.4. The focal lengths f_x and f_y are calculated as the width of the image frame, a skew parameter γ is set to 0, and principal points u_0 and v_0 are calculated as half of the width and height of the image frame respectively. The camera matrix is given by Equation 5.1.

$$Camera\ Matrix\ = \begin{bmatrix} w & 0 & w/2 \\ 0 & w & h/2 \\ 0 & 0 & 1 \end{bmatrix} \qquad 5.1$$

where w and h represent image width and height respectively. The second matrix, the distortion matrix consists of the coefficients which are assumed to be zero. These coefficients are considered zero assuming the camera used has no distortion. After all the parameters of the *solvePnP* are known, the calculation returns rotation and translation vector. The rotation vector is converted to a rotation matrix using OpenCV's *Rodrigues* function. From the rotational matrix, 3 angles are obtained using OpenCV's *RQDecomp3x3* function. This function calculates the Euler angles described in section 3.3 The obtained angles are pitch, yaw, and roll respectively measured in degrees. The calculation of these angles is shown in Algorithm 2

Algorithm 2 Calculation of Pitch, Yaw, and Roll

```
Algorithm 2:   HEAD_POS_ESTIMATOR.HEAD_POSITION_DATA
Input    :   Facial landmarks (landmarks)
Output   :   Calculated pitch, yaw, and roll
begin
   1   height, width ← image.shape()
   2   camera_matrix ← [width, 0, height/2],
                        [0, width, width/2],
                        [0, 0, 1]]
   3   dist_matrix ← 4x1 null matrix
   4   metric_landmarks, pose_transform_mat ← get_metric_landmarks (landmarks.copy(),
       pcf)
   5   model points ← calculate 36 3D coordinates from metric_landmarks
   7   image_points ← calculate 36 2D coordinates from landmarks
   8   success,rVec, tVec ← cv2.solvePnP (model_points, image_points, camera_matrix,
                              dist_matrix, flags=cv2.SOLVEPNP_ITERATIVE)
   9   rotationMatrix, jac ← cv2.Rodrigues(rotation_vector)
  10   angles, mtxR, mtxQ, Qx, Qy, Qz ← cv2.RQDecomp3x3(rotationMatrix)
  11   pitch, yaw, roll ← angles[0], angles[1], angles[2]
  12   return pitch, yaw, roll
end
```

5.4.2. Head position detection

Once, calculating the pitch, yaw and roll are done, the next task is to estimate the position of the head of the driver based on these 3 values. When the camera is straight in front of the driver, the normal yaw and roll angles are considered as 0^0. The left or right turn indicates the value of yaw less than 0^0 (negative values) or more than 0^0 (positive values) respectively. Similarly, the down-left or down-right indicate the value of roll less than 0^0 (negative values) or more than 0^0 (positive values) respectively. However, the normal pitch of the face depends on how high the face is from the camera. To know the normal pitch of the face, the algorithm runs for 10 seconds before the detection of position starts. After 10 seconds the algorithm estimates the normal pitch value of the driver and sets the threshold for all kinds of turns. The criteria of the head position estimation are checked in each frame and positions are estimated as listed in Table 5.1.

Table 5.1 Head position estimation conditions

Position	Criteria
Left Turn	$Yaw_{current\_frame} <= -35^0$
Right Turn	$Yaw_{current\_frame} >= 35^0$
Down-Left Turn	$Roll_{current\_frame} <= -35^0$
Down-Right Turn	$Roll_{current\_frame} >= 35^0$
Up Turn	$Pitch_{current\_frame} <=$ Normal pitch - 35^0
Down Turn	$Pitch_{current\_frame} >=$ Normal pitch + 35^0
Normal Position	Looking forward and all the above conditions are false.

Once the driver's head position is detected other than in normal position, the timer counter starts and if the head is turned continuously even after *Threshold Time* as calculated in Equation 5.2, the algorithm produces the alarm "ALARM!!! Head Turn more than threshold time". The algorithm stops the warning, and timer counter and resets it when the head position is found in normal position.

$$Threshold\ Time = \frac{AverageFPS}{F} \tag{5.2}$$

Where *AverageFPS* represents the mean value of fps observed during the setup period and *F* is the factor that should be known by the user so that *Threshold Time* is always approximately equal to 2 seconds.

5.5. Algorithm for Detecting Closed Eye

The algorithm is based on the calculation of ADR as Equation 3.9 and its comparison to the threshold value of ADR to determine whether the eye is in an open or closed state. The open eye will have the maximum ADR value and the closed eye will have the minimum ADR value close to zero. This concept is used to estimate the state of the eye. If the eye is in a closed state, the algorithm then starts the time counter. If the eye is closed for a sufficiently long time than

Threshold Time from Equation 5.2, the algorithm produces the warning message “Eye Closed more than threshold time” to the driver.

5.5.1. Calculation of ADR

In each frame, the ADR value is calculated from the 32 facial points that surround the eyes as represented in Algorithm 3 using Equation 3.9.

Algorithm 3 Calculation of ADR

```
Algorithm 3:   CALCULATIONS.COMPUTE_ADR
Input     :    Eyes facial landmarks point coordinates (16 (x,y) for each eye i.e.
               leftEyeCoordinates and rightEyeCoordinates arranged in a list)
Output    :    Calculated ADR value
begin
    1  | left_eye_area ← Calculate the area of the left eye
    2  | right_eye_area ← Calculate the area of the right eye
    3  | left_eye_center ←  Calculate the center of the left eye
    4  | right_eye_center ← Calculate the center of the right eye
    5  | eyes_distance ← Calculate the distance between the center of the left and right eye
    7  | ADR ← (sqrt(left_eye_area) + sqrt(right_eye_area)) / (2 * eye_distance)
    8  | return ADR
end
```

5.5.2. Eye state detection

For the eye state detection, the ADR value calculated in section 5.5.1 must be compared to some threshold value. If the value of ADR is below the threshold value, the eye is declared as closed otherwise it is open. However, setting a threshold is a problem because the ADR value depends upon the facial structure and differs from individual to individual. To calculate the adaptive threshold of ADR, the algorithm *DETECT_FACE* runs for 10 seconds (setup period) to record the value of ADR of a person at the beginning of the program. Based on this recorded ADR, a threshold is calculated which becomes the threshold of that individual. In case the driver is changed, the threshold needs to be set up again. The threshold is calculated using 15% of the highest ADR values recorded during the setup period. These highest ADR values represent the open eye state. The threshold is calculated using the formula as represented in Equation 5.3

$$ADR\ threshold = 0.90 * averageOpenEyeADR \qquad 5.3$$

where *averageOpenEyeADR* represents the average of the 15% of recorded ADR values arranged in descending order during the setup period. This means that 15% of the data on open eye states are used to find the average ADR value of the open eye of an individual. Then, the threshold is set using this average ADR.

The threshold is set immediately in the next frame once the setup period is over. After this, the ADR of each frame is compared to the threshold value. If the ADR value is less, the timer counter starts and stops only when the eye is opened in the following frames. If the time for which the eye is closed is greater than *Threshold Time*, the warning message is displayed to the driver. The eye-long closed detection is presented in Algorithm 4.

Algorithm 4 Detection of eye long-closed state

```
Algorithm 4: EYELONGCLOSEDDETECTOR.DETECT_EYELONGCLOSED
Input   :  Calculated ADR value of each frame after the setup period is over (ADR).
Output  :  State of Eyes
begin
  1   If is_threshold_set is False:
  2       setup_adr_list.sort(reverse=True)
  3       length ← len( setup_adr_list )
  4       length_for_threshold ← 0.15 * length
  5       threshold ← 0.90 * (sum(setup_adr_list[:length_for_threshold])/
      length_for_threshold)
  6       is_threshold_set ← True
  7   else:
  8       if ADR < threshold:
  9           if close_timer_counter == 0:
 10               close_timer_counter ← time.time()
 11           else:
 12               end_time ← time.time()
 13               if end_time – close_timer_counter > time_threshold:
 14                   print(“Wake Up!!!!”)
 15       else:
 16           close_timer_counter ← 0
end
```

5.6. Algorithm for Detecting Yawn

The yawn detection algorithm consists of two steps where the first step is the calculation of the MAR and the second step consist of a recording of the MAR value of the individual during the setup period, calculating the threshold value, and comparing the calculated MAR value with a threshold MAR value.

5.6.1. Calculation of MAR

In each frame, the MAR value is calculated from the 6 facial points out of 20 points that surround the mouth as represented in Algorithm 5 using Equation 3.9.

Algorithm 5 Calculation of MAR

```
Algorithm 5:   CALCULATIONS.COMPUTE_MAR
Input    :   Mouth facial landmarks point coordinates (20 (x,y) for mouth i.e.
             mouthCoordinates arranged in a list)
Output   :   Calculated MAR value
begin
    1   | a ← abs(dist.euclidean(mouth[4], mouth[16]))
    2   | b ← abs(dist.euclidean(mouth[6], mouth[14]))
    3   | c ← abs(dist.euclidean(mouth[0], mouth[10]))
    4   | MAR ← ((a + b) / (2.0 * c)) * 100
    5   | return ← MAR
end
```

5.6.2. Mouth state detection

For the mouth state detection i.e. open or closed, the MAR value calculated as in section 5.6.1 is compared with the threshold value of MAR. Similar to the ADR value, the MAR also depends upon the facial structure and hence differs from person to person. So, to set the threshold that is adaptive to the individual, the MAR value will be recorded in the initial setup period. Once the setup period is over, the threshold is calculated in the next frame. The threshold is calculated by using the minimum value of MAR i.e. MAR of a closed mouth as represented in Equation 5.4

$$MAR\ threshold = 4 * minMAR \tag{5.4}$$

where *minMAR* is the minimum MAR value recorded during the setup period. If the MAR value is greater than the MAR threshold, then the mouth is in an open state otherwise it is closed. For yawning, the mouth has to be in an open state for at least *Threshold Time* seconds which is calculated as Equation 5.2.

When the mouth is detected as open, the timer counter starts and only stops once the mouth is detected as a closed state. If the mouth is detected open even after 2 seconds the algorithm detects that the person is yawning. The yawn detection method is shown in Algorithm 6.

Algorithm 6 Yawn detection

Algorithm 6: YAWNDETECTOR.DETECT_YAWN

```
Input   :   Calculated MAR value of each frame after the setup period is over (MAR).
Output  :   State of the mouth
begin
   1  | if is_threshold_set is True:
   5  |       threshold ← 4 * min(setup_mar_list)
   6  |       is_threshold_set ← False
   7  | else:
   8  |       if MAR > threshold:
   9  |             if open_timer_counter == 0:
  10  |                   open_timer_counter ← time.time()
  11  |             else:
  12  |                   end_time ← time.time()
  13  |                   if end_time – open_timer_counter > time_threshold:
  14  |                         print("Yawning")
  15  |       else:
  16  |             open_timer_counter ← 0
end
```

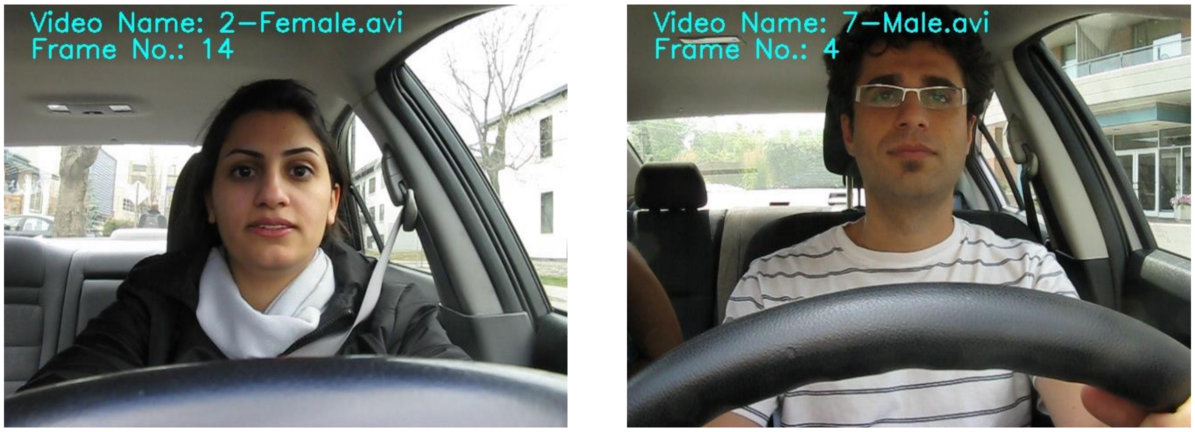
5.7. System Setup

The developed DMS application contains different modules like *driverMonitoring.py* as the main module, *faceDetect.py* for face detection, and *calculations.py* for calculation of ADR and MAR. Other modules include *headPostionEstimation.py* for estimating the head position of the driver, *faceGeometry.py* which contains the 3D real-world coordinates of a generic model, and *eyeLongClosedDetection.py* for detecting the eye state and raising warning if necessary. The final module is the *yawnDetection.py* to detect the mouth state and yawn. The application was run on 64-bit Windows OS with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz PC environment. For the application to run on PC, the packages like *mediapipe*, *opencv-python*, *scipy*, *numpy*, etc. are installed. These packages are installed using the *pip install <package-name>* command. Then after the DMS application can be run through the command prompt using the command *python driverMonitoring.py -f <video-path>* and can be quitted by key *'q'*.

In Raspberry Pi, the installation of the packages is similar to PC using *pip3 install <package-name>* except for the mediapipe package which can be installed using the command *sudo pip3 install mediapipe-rpi3* for Raspberry Pi 3 and *sudo pip3 install mediapipe-rpi4* for the Raspberry Pi 4. The additional dependency packages if needed can be found in [31]. The DMS application in Raspberry Pi can be run from *python3 driverMonitoring.py -f <video-path> -s <skip-frames>* and can be quitted by key *'q'*. For running the application through a webcam or Pi camera, *0* is written as an argument in place of *<video-path>*.

6. VALIDATION AND TESTING OF PROPOSED SOLUTION

The testing and validation of the proposed solution are presented in this section. After the algorithms for each module i.e. face and head position detection, eye and eye long-closed detection, and mouth and yawning detection are developed, the individual modules are integrated to form a single DMS. The integrated module is tested on both the PC environment and Raspberry Pi. The system is tested using the YawDD dataset. Altogether 15 video sequences (8 males and 7 females) with 33.218 frames of resolution 640 x 480 recorded at 30 fps are used from this dataset. The example of the frames in this dataset is shown in Figure 6.1. The brief detail of the dataset can be found in Annex I.



(a) (b)

Figure 6.1 Examples of the image in the YawDD dataset [26] (a – Image from a video sequence named 2-Female.avi, b - Image from a video sequence named 7-Male.avi) [26]

The video sequences are run and the individual frame is labeled as TN, TP, FN, and FP. The meaning of this labeling is represented in Table 6.1. After labeling each frame, the precision, recall, and F1 score are calculated to test the accuracy of the system. These labels are labeled by the comparison of the original frames i.e. ground truth data with the predicted frame's data.

Table 6.1 Labels description

Labels	Description
TN	Predicted a normal condition in a frame and it is correct
TP	Predicted an abnormal condition in a frame and it is correct

FN	Predicted a normal condition in a frame and it is incorrect
FP	Predicted an abnormal condition in a frame and is incorrect

The normal condition represents the ordinary driving conditions like the face is present in a frame, the eye is in a normal state i.e. not closed for a long time, no yawning, and no head turns in any direction while driving. Whereas the abnormal condition represents the conditions like face not present in a frame, eyes closed for a long time, yawning, and turning in either direction (up/down/left/right).

6.1. Test in PC

In the PC environment, the overall accuracy measured in terms of the F1-Score of the system was obtained as 0.91 as shown in Table 6.2. The F1-Score of the videos ranges from 0.56 to 1 which means the algorithm can predict the driver's behavior in most of the videos as it was designed. However, in some of the driver's video sequences, the accuracy was obtained very low due to the high number of FN or FP.

Table 6.2 PC environment test results in terms of recall, precision, and F1-score

S.No.	Video Name	TP	TN	FP	FN	Recall	Precision	F1 Score
1	1-Female.avi	511	1937	0	0	1.00	1.00	1.00
2	2-Female.avi	180	1668	0	19	0.90	1.00	0.95
3	3-Female.avi	527	2156	60	0	1.00	0.90	0.95
4	5-Female.avi	123	1516	75	119	0.51	0.62	0.56
5	6-Female.avi	442	708	157	0	1.00	0.74	0.85
6	8-Female.avi	703	1502	75	91	0.89	0.90	0.89
7	11-Female.avi	281	1015	128	0	1.00	0.69	0.81
8	1-Male.avi	420	1866	0	0	1.00	1.00	1.00
9	2-Male.avi	701	1536	85	0	1.00	0.89	0.94
10	3-Male.avi	242	1552	14	19	0.93	0.95	0.94
11	4-Male.avi	259	1429	0	43	0.86	1.00	0.92
12	7-Male.avi	338	1721	62	0	1.00	0.85	0.92
13	9-Male.avi	353	1292	111	0	1.00	0.76	0.86

14	10-Male.avi	355	1776	0	0	1.00	1.00	1.00
15	11-Male.avi	175	1310	47	0	1.00	0.79	0.88
Overall		5610	22984	814	291	0.95	0.87	0.91

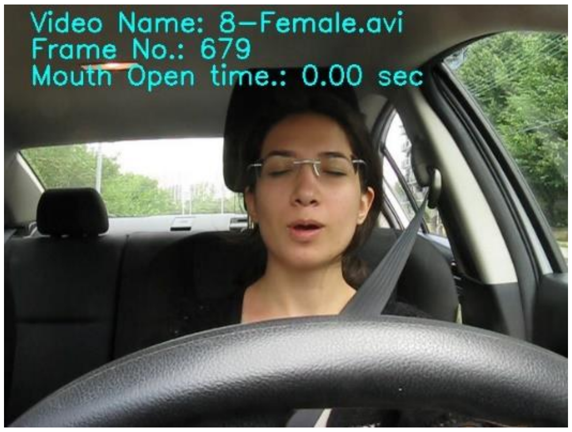
6.1.1. Reasons for FN

FNs were observed in some video sequences where the maximum number counts during the yawning detection and rest during head position estimation and eye long-closed detection. A high number of FNs were observed in yawning detection as the algorithm failed to detect the yawn.

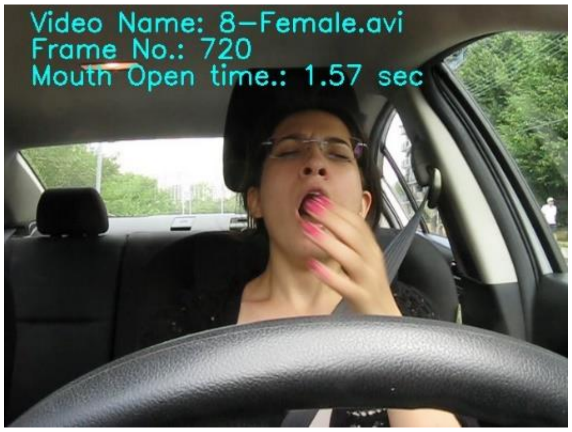
The main reasons for the FN are

- Due to the covering of the mouth during yawning. If a person starts yawning, the algorithm starts the timer counter and notes down the start frame number. The yawn is only detected if the person's mouth is open as in yawning for at least 2 seconds for around 30 fps video i.e. approximately 60 frames. If the person opens their mouth for yawning for at least or more than 2 seconds, the algorithm at the end of the yawn i.e. when the mouth is predicted as closed, detects the person has just yawned from the start frame to the frame preceding the current frame. However, if the person closes the mouth during yawning, the mouth is not detected after some frames which means the mouth is in a closed state. So yawn is not detected as the timer counter resets to zero and the yawning is not detected due to the less time between the start of the yawn and the frame where the mouth is closed. Also, if the mouth is closed by hand from the start of the yawing, here the algorithm predicts the normal frames, and FNs are observed in these cases.

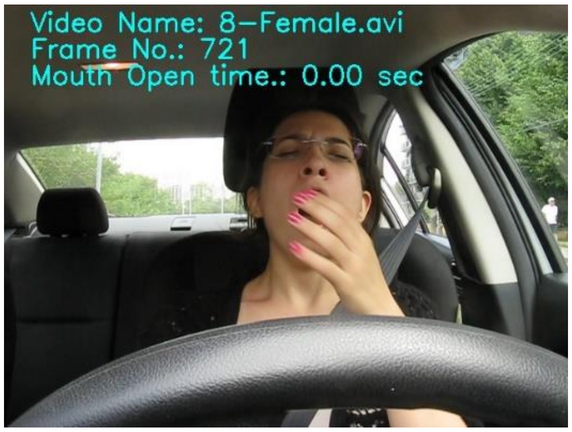
  Similar conditions of covering the mouth with hands while yawing were observed in video sequences named "5-Female.avi" and "8-Female.avi" as shown in Table 6.2. Figure 6.2 shows the FNs observed when an individual from video sequence "8-Female.avi" covered her mouth with hands during yawing and the algorithm did not predict those frames as the individuals were yawning. The mouth was open from the 679th frame and in the 721st frame the mouth was closed by hand and the time elapsed was only 1.57sec and couldn't detect the yawn as the mouth open was not detected for more than the threshold time which is almost 2 seconds.



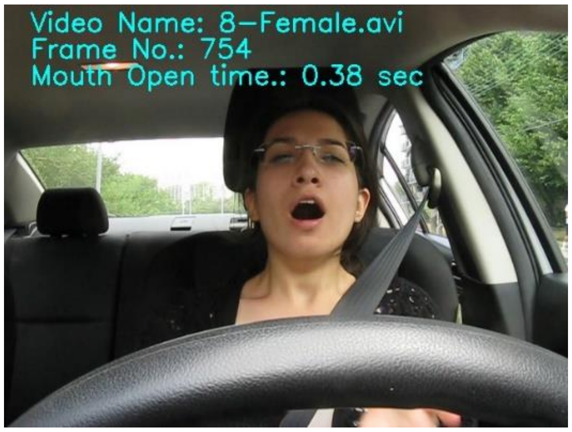
(a)



(b)



(c)



(d)

Figure 6.2 FNs observed due to covering the mouth by hand [26]. (a –Yawn starts from Frame 679, b- Frame 720 where the mouth is still detected as open for 1.57 sec, c – Frame 721 where the mouth is closed by hand and timer resets, d – arbitrary Frame 754 where the mouth is detected open again for 0.38 sec after mouth was uncovered.

- Due to the short yawn, which is not detected by the algorithm. The algorithm starts the timer counter but cannot exceed 2 seconds hence not labeled as yawning. This condition is observed in a video sequence named “2-Female.avi” and “4-Male.avi”. In Figure 6.3, from 313th to 331st frame, the mouth is detected as open i.e. approximately 1.84 seconds which is less than 2 seconds.

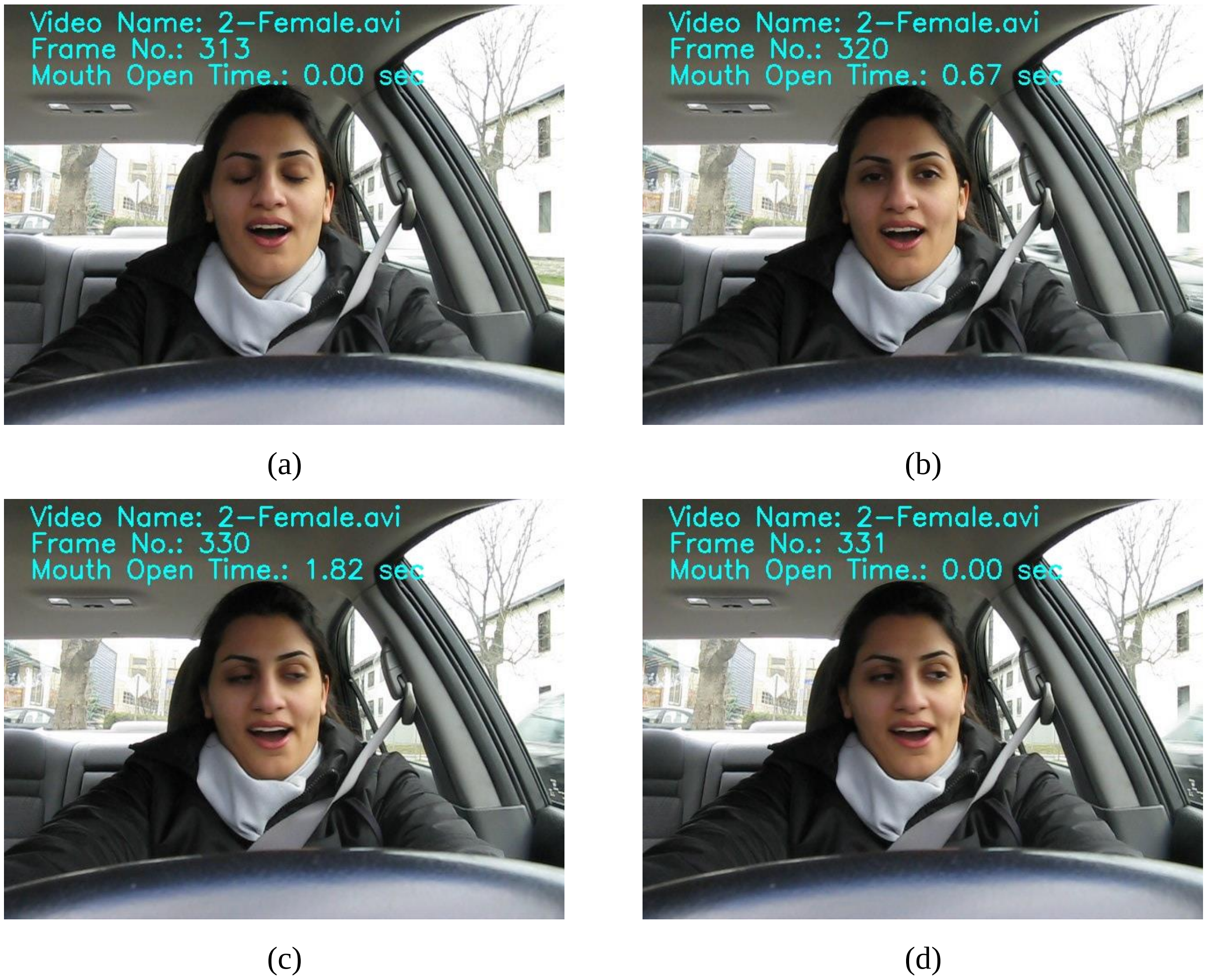


Figure 6.3 FNs observed due to short yawn time [26] (a – Yawn start from frame 313, b – Mouth open detected for 0.67 sec at frame 320, c – Mouth open detected for 1.82 sec at Frame 330, d – Mouth detected as closed in Frame 331 so that yawn is not detected)

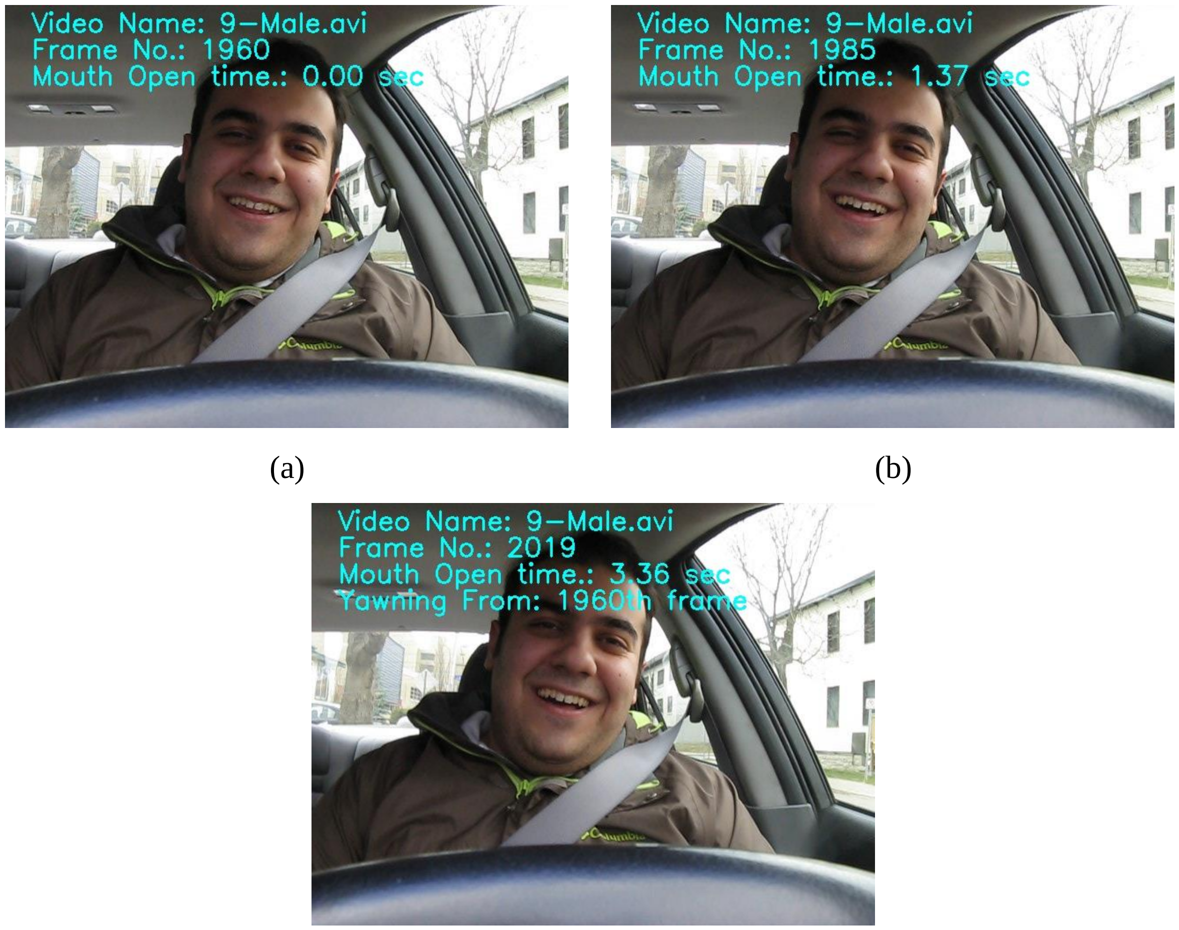
- The FNs were also observed due to setting incorrect thresholds for pitch, MAR, and ADR. For example, the algorithm calculates the threshold MAR to predict whether the mouth is open or closed. If the threshold calculated is high then it actually should have been, the algorithm cannot detect the yawn when an individual is yawning.

6.1.2. Reasons for FP

The algorithm detected the abnormalities in some frames when there is none. During this condition, FPs are detected. For example, if a person is laughing and the algorithm detects it as yawning.

The main reasons for the FP are:

- The FP's in most cases were observed when an individual was laughing. During laughing the person's mouth is detected as open. So, if the mouth is open for more than a threshold time, the algorithm counted it as a yawn. For instance, the video sequences "3-Female.avi", "5-Female.avi", "6-Female.avi", "8-Female.avi", and "11-Female.avi", "2-Male.avi", "7-Male.avi", "9-Male.avi" and "11-Male.avi". Figure 6.4 shows the yawning detected during laughing for the video "9-Male.avi".



(a) (b)

(c)

Figure 6.4 FPs observed due to laughing[26] (a- Frame 1960 from which yawn is detected as started, b – Frame 1985 after 1.37 seconds and mouth is still open, c – Person is detected as yawning even after 3.36 sec at Frame 2019)

- The next reason for the FP’s is the calculation of the threshold during the setup period. If the threshold set is very low and while comparing the calculated values with the threshold values, the FP’s may be detected.

The execution speed of the developed DMS in the PC was not a problem, as the speed was obtained to be 32.03 fps for VGA-resolution images and up to 60.29 fps for 160 x 120 resolution images which can be shown in Table 6.3.

Table 6.3 Execution Speeed in PC

S.No.	Video Name	Image size (640x480) Execution Speed (fps)	Image size (160x120) Execution Speed (fps)
1	1-Female.avi	31.88	59.44
2	2-Female.avi	33.40	63.32
3	3-Female.avi	33.47	63.42
4	5-Female.avi	32.80	61.84
5	6-Female.avi	33.59	57.11
6	8-Female.avi	33.70	61.5
7	11-Female.avi	34.73	62.26
8	1-Male.avi	30.13	61.47
9	2-Male.avi	29.80	63.3
10	3-Male.avi	30.35	55.76
11	4-Male.avi	30.92	55.83
12	7-Male.avi	31.46	60.29
13	9-Male.avi	31.65	56.97
14	10-Male.avi	31.40	60.65
15	11-Male.avi	31.20	61.17
Average		32.03	60.29

6.2. Validation in Raspberry Pi

The same set of video sequences that were tested in the PC environment is tested in Raspberry Pi. In Raspberry Pi also, the test was considering both factors, execution speed and accuracy.

6.2.1. Execution speed in Pi

The initial speed for a 160 x 120 frame was found to be around 4-5 fps which was very low for a system designed to be working in a real-time environment. Also, any abnormalities in the video sequences were detected using a series of consecutive frames at a certain time which is at least 2 seconds. This means the decision of any abnormalities is not detected from one frame but a series of consecutive frames. This advantage is used to speed up the execution of the video in Raspberry Pi. During one frame the detection of the state of the eyes and the mouth, and head position is done, and some consecutive frames are skipped where no detection is done. The state of these skipped consecutive is ignored as this will not hamper the detection in most of the cases. In Raspberry Pi, for the developed system to be working at least at 24 fps, 5 and 6 frames were skipped and the testing is performed. While skipping 5 and 6 frames, the average execution speed was obtained as 24.07 and 26.69 fps respectively which is shown in Table 6.4. The threshold time for the detection of eye long closed or yawning or head position is calculated using Equation 5.2 where factor *F* is estimated as 13.5 to make the threshold time approximately equal to 2 seconds.

Table 6.4 Execution Speed in Raspberry Pi

S.No.	Video Name	5 Frames Skipped	6 Frames Skipped
		Execution Speed (fps)	Execution Speed (fps)
1	1-Female.avi	23.96	26.35
2	2-Female.avi	24.06	26.82
3	3-Female.avi	24.18	26.78
4	5-Female.avi	24.11	26.61
5	6-Female.avi	24.02	26.43
6	8-Female.avi	24.07	26.66
7	11-Female.avi	24.21	26.65
8	1-Male.avi	23.99	26.72
9	2-Male.avi	24.04	26.79
10	3-Male.avi	24.13	26.82
11	4-Male.avi	24.12	26.84
12	7-Male.avi	24.04	26.81
13	9-Male.avi	24.00	26.73
14	10-Male.avi	24.11	26.64
15	11-Male.avi	24.03	26.70
Average		24.07	26.69

6.2.2. Accuracy in Raspberry Pi

As described in section 6.2.1, the test in Raspberry Pi is performed in the video sequences by skipping 5 and 6 frames to validate the model developed in the PC. Similar to the PC environment, the F1-Score for each video sequence is calculated by labeling the frames as TP, TN, FP, and FN respectively according to Table 6.1. The recall, precision, and the F1-score are calculated from those labeling. The overall accuracy of the system measured in the F1-score was obtained as 0.91 and 0.92 while 5 frames and 6 frames are skipped respectively as shown in Table 6.5 and Table 6.6. The minimum accuracy was calculated as 0.82 and 0.69 respectively for 5 and 6 frames skipped algorithms whereas, in some video sequences, the maximum accuracy of 1.00 was also obtained in both cases.

Table 6.5 Raspberry Pi test results while skipping 5 frames

S.No.	Video Name	TP	TN	FP	FN	Precision	Recall	F1-Score
1	1-Female.avi	103	398	0	0	1.00	1.00	1.00
2	2-Female.avi	34	349	0	7	1.00	0.83	0.91
3	3-Female.avi	94	470	0	0	1.00	1.00	1.00
4	5-Female.avi	30	343	0	10	1.00	0.75	0.86
5	6-Female.avi	80	158	19	0	0.81	1.00	0.89
6	8-Female.avi	90	334	0	33	1.00	0.73	0.85
7	11-Female.avi	41	239	0	11	1.00	0.79	0.88
8	1-Male.avi	81	387	0	0	1.00	1.00	1.00
9	2-Male.avi	105	287	45	0	0.70	1.00	0.82
10	3-Male.avi	49	311	5	3	0.91	0.94	0.92
11	4-Male.avi	52	291	0	8	1.00	0.87	0.93
12	7-Male.avi	77	326	28	0	0.73	1.00	0.85
13	9-Male.avi	74	258	23	0	0.76	1.00	0.87
14	10-Male.avi	51	359	0	0	1.00	1.00	1.00
15	11-Male.avi	70	235	0	0	1.00	1.00	1.00
Overall		1031	4745	120	72	0.90	0.93	0.91

Table 6.6 Raspberry Pi test results while skipping 6 frames

S.No.	Video Name	TP	TN	FP	FN	Precision	Recall	F1-Score
1	1-Female.avi	83	330	0	0	1.00	1.00	1.00
2	2-Female.avi	28	286	0	6	1.00	0.82	0.90
3	3-Female.avi	81	385	0	0	1.00	1.00	1.00
4	5-Female.avi	23	271	13	8	0.64	0.74	0.69
5	6-Female.avi	61	146	1	0	0.98	1.00	0.99
6	8-Female.avi	108	255	0	24	1.00	0.82	0.90
7	11-Female.avi	44	180	10	0	0.81	1.00	0.90
8	1-Male.avi	73	310	0	0	1.00	1.00	1.00
9	2-Male.avi	81	246	30	1	0.73	0.99	0.84
10	3-Male.avi	24	263	3	12	0.89	0.67	0.76
11	4-Male.avi	44	237	0	7	1.00	0.86	0.93

Table 6.6 Raspberry Pi test results while skipping 6 frames

S.No.	Video Name	TP	TN	FP	FN	Precision	Recall	F1-Score
12	7-Male.avi	66	274	14	0	0.83	1.00	0.90
13	9-Male.avi	62	194	34	0	0.65	1.00	0.78
14	10-Male.avi	70	288	0	0	1.00	1.00	1.00
15	11-Male.avi	54	189	0	4	1.00	0.93	0.96
Overall		902	3854	105	62	0.90	0.94	0.92

6.3. Comparison with PC results

The test results of the PC and the Raspberry Pi in both cases are similar for particular video sequences. In Figure 6.5, the test results are represented by F1-Score along the y-axis and the video sequence name along the x-axis. The x-axis label represents the video sequence name in the abbreviated form. For example, F1 represents 1-Female.avi and M1 represents 1-Male.avi. The difference between the F1 score in some video sequences is due to the threshold value. The threshold value is set by the data recorded in the setup period. During this period, the ADR, MAR, and pitch are recorded to calculate the threshold value. So, on the PC, all the frames are processed without skipping, and in Raspberry Pi, the frames are skipped and processed. This led to the difference in threshold value in the same video while running on the PC and Raspberry Pi. In the Raspberry Pi also, the difference in threshold was found in some video sequences while skipping 5 and 6 frames which led to the difference in F1-Score.

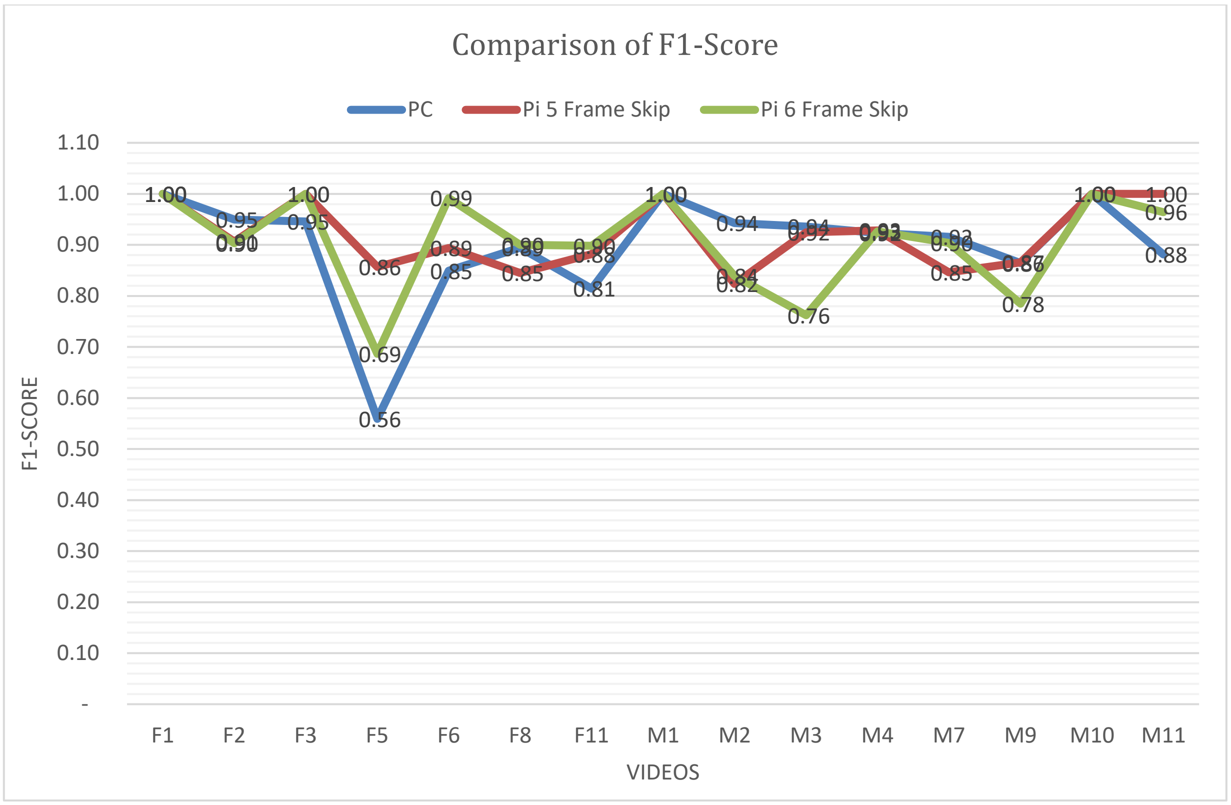


Figure 6.5 Comparison of the F1-Score of PC and Raspberry pi

6.4. Testing using Raspberry Pi Camera

The DMS developed is finally tested in real-time using a Raspberry Pi camera. The test was performed to demonstrate the working of the DMS inside the office. An algorithm with 5 frame skipping is used here in testing. Figure 6.6 shows the random frame where the individual is looking toward the camera and no abnormalities are observed. In Figure 6.7, the individual mouth started to open from frame 1890 and the yawning is detected after 2 seconds has been elapsed in frame 1940. In Figure 6.8, some of the positions of the head are shown. The algorithm raises the warning if the individual turns his head in any direction other than looking forward for more than 2 seconds. Finally, in Figure 6.9, the individual's eyes are detected as closed from frame 520, and the warning of eye closure is raised after 2 seconds in frame 565. The warning is turned off once the person returns to the normal position.

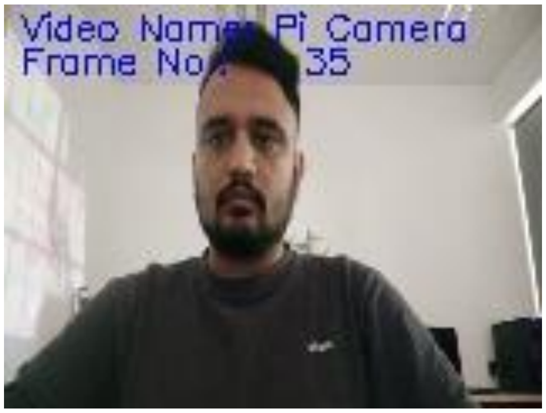
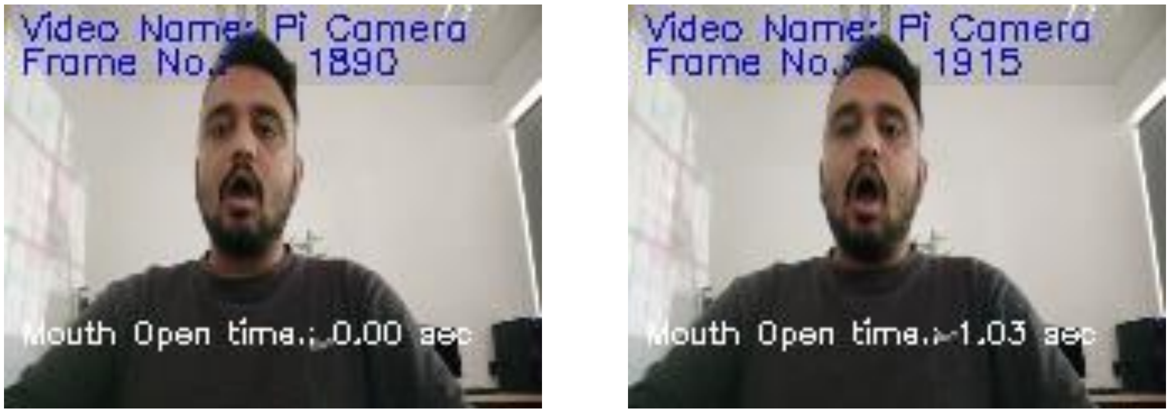
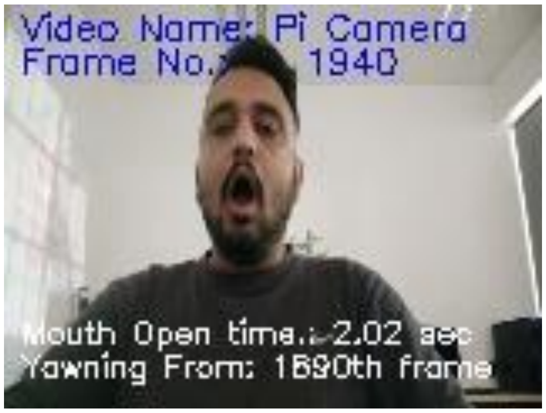


Figure 6.6 Normal Frame

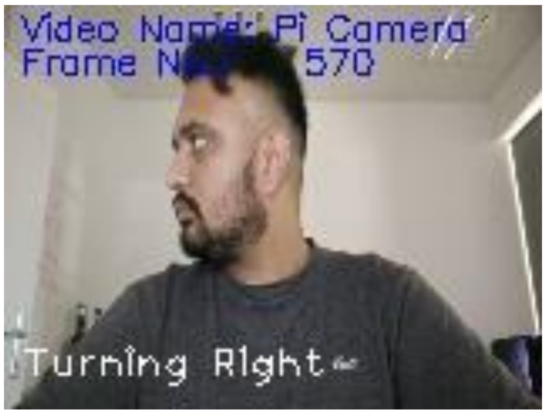


(a) (b)

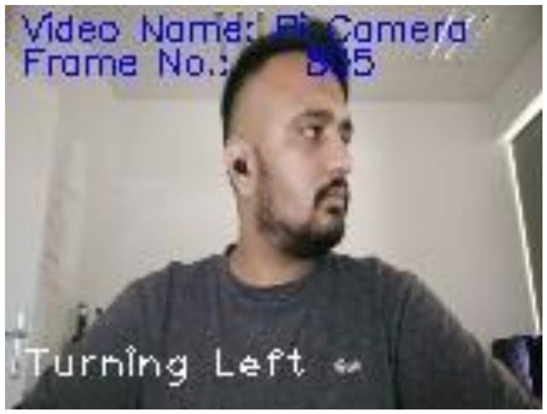


(c)

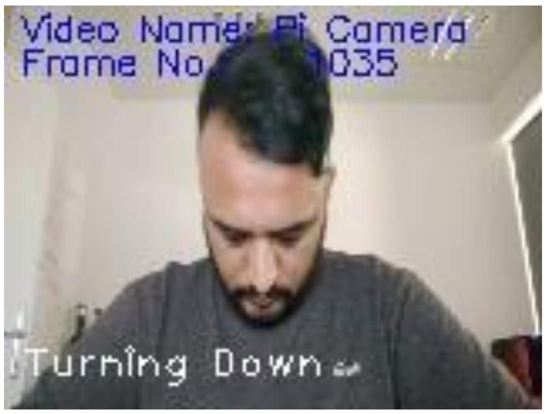
Figure 6.7 Yawning detection (a – Individual starts to yawn at frame 1890, b – Arbitrary frame 1915 after 1.03 secs, c – Yawning detected after 2 seconds in frame 1940)



(a)

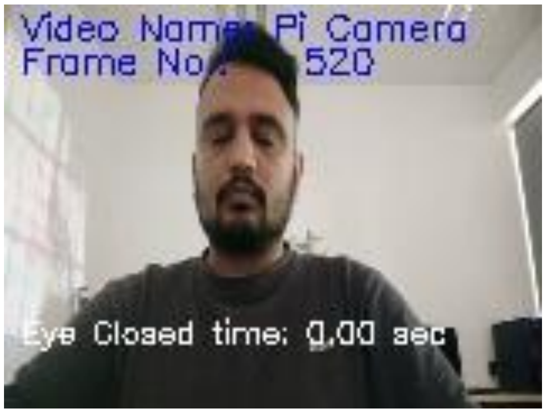


(b)

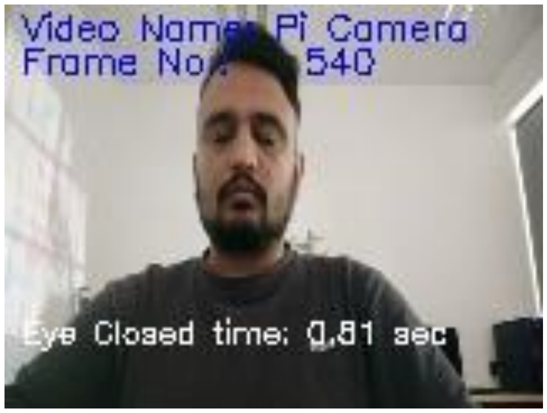


(c)

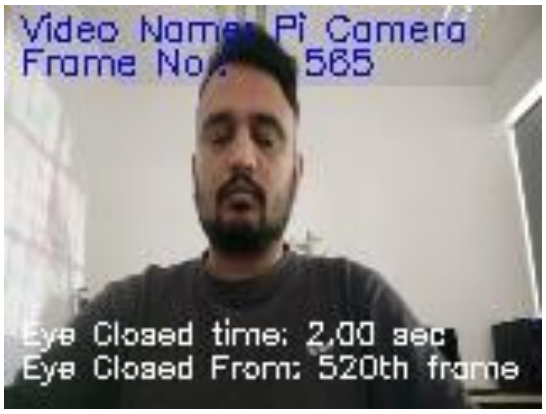
Figure 6.8 Head Position Estimation (a –Turning right, b-Turning left, c- Turning down)



(a)



(b)



(c)

Figure 6.9 Eye long-closed detection (a - Individual close eye at frame 520, b – Arbitrary frame 540 after 0.81 secs, c – Eye long-closed detected after 2 seconds in frame 565)

7. CONCLUSION

In this thesis, a DMS was developed in the PC environment and the same system was validated using the Raspberry Pi. The implementation of this system is based on the behavioral approach of drowsiness detection where the state of eyes and the mouth, and head position are used to detect drowsiness or inattentiveness, regularly monitor them, and alarm in case of abnormality. The system was created and tested using the DMD and YawDD dataset. A CNN-based model known as MediaPipe FaceMesh was used for the face detection and localization of the facial points. The concept of ADR and MAR is used to predict the eye and the mouth state from the facial points that surround the eye and mouth as mentioned in Equations 3.9 and 3.10. Furthermore, the concept of Euler angles based on the rotational matrix is used to estimate the head position from the 36 - 2D and 3D facial points surrounding the face. The algorithm also calculates the adaptive threshold for MAR, ADR, and mean position of a face which will vary from individual to individual. The DMS tested on 15 video sequences of the YawDD dataset obtained the accuracy of 91% on PC and the same tested in the Raspberry Pi, the accuracy was 91% and 92% in two cases. i.e. skipping 5 and 6 frames in the video sequence as discussed in section 6.2.2. In PC the execution speed varied from a minimum average of 32.03 fps to a maximum of 60.09 fps. The overall execution speed was obtained as 24.07 and 26.69 fps in Raspberry Pi. The results are satisfactory with the prospect of further improvement in a more adaptive threshold setting in all the cases. Also, the developed system can detect the face and facial states during the daytime or when there is enough light as the normal RGB camera or the RGB frames of the videos during the daytime were used. This limitation can be corrected by the use of IR cameras or night vision cameras. Similarly, the model that can recognize the facial expression and behaviors like laughing, talking, eating, smoking/vaping, etc. and an additional feature of gaze detection in the eyes to detect unnecessary distractions while the use of a phone could be used in integration with this developed system to increase the accuracy and performance of the real-time working DMS.

REFERENCES

[1] "Road traffic injuries," *World Health Organization*, Jun. 21, 2021. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (accessed May 15, 2022).

[2] Directorate-General for Mobility and Transport, "Road safety in the EU: fatalities in 2021 remain well below pre-pandemic level," Mar. 28, 2022. Accessed: Jun. 15, 2022. [Online]. Available: https://transport.ec.europa.eu/news/preliminary-2021-eu-road-safety-statistics-2022-03-28_en

[3] National Center for Statistics and Analysis, "Early estimate of motor vehicle traffic fatalities in 2021," National Highway Traffic Safety Administration, Washington, DC, DOT HS 813 283, Apr. 2022. Accessed: Jun. 19, 2022. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813283

[4] S. Singh, "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey," National Highway Traffic Safety Administration, Washington, DC, DOT HS 812 115, Feb. 2015. Accessed: Jun. 20, 2022. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115

[5] SAE International and SAE J3016, "SAE International Technical Standard Provides Terminology for Motor Vehicle Automated Driving Systems," Oct. 02, 2014. https://www.sae.org/binaries/content/assets/cm/content/news/press-releases/pathway-to-autonomy/automated_driving.pdf (accessed Jun. 29, 2022).

[6] K. Barry, "Driver Monitoring Systems Can Help You Be Safer on the Road," *Consumer Reports*, Feb. 17, 2022. https://www.consumerreports.org/car-safety/driver-monitoring-systems-ford-gm-earn-points-in-cr-tests-a6530426322/ (accessed May 15, 2022).

[7] F. Lyrheden, "Driver Monitoring (DMS) on its way to becoming mandatory in vehicles around the world," *Smart Eye*, Sep. 29, 2020. https://smarteye.se/blogs/driver-monitoring-dms-on-its-way-to-become-mandatory-in-vehicles-around-the-world/ (accessed May 15, 2022).

[8] A. Čolić, O. Marques, and B. Furht, *Driver Drowsiness Detection*. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-11535-1.

[9] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[10] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "FaceBoxes: A CPU Real-time Face Detector with High Accuracy," *arXiv:1708.05234 [cs]*, Dec. 2018, Accessed: Jan. 28, 2022. [Online]. Available: http://arxiv.org/abs/1708.05234

[11] Xiangxin Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, Jun. 2012, pp. 2879–2886. doi: 10.1109/CVPR.2012.6248014.

[12] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, vol. 9905, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[13] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks,” *IEEE Signal Process. Lett.*, vol. 23, no. 10, pp. 1499–1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[14] Google LLC, “MediaPipe Face Mesh,” *MediaPipe*. https://google.github.io/mediapipe/solutions/face_mesh (accessed Jan. 19, 2022).

[15] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, “BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs,” 2019, doi: 10.48550/ARXIV.1907.05047.

[16] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, “Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs,” 2019, doi: 10.48550/ARXIV.1907.06724.

[17] D. King, “Dlib-Models,” *Github*. https://github.com/davisking/dlib-models (accessed Nov. 21, 2021).

[18] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “300 Faces in-the-Wild Challenge: The First Facial Landmark Localization Challenge,” in *2013 IEEE International Conference on Computer Vision Workshops*, Sydney, Australia, Dec. 2013, pp. 397–403. doi: 10.1109/ICCVW.2013.59.

[19] T. Soukupova, “Real-Time Eye Blink Detection using Facial Landmarks,” p. 8.

[20] W. O. Lee, E. C. Lee, and K. R. Park, “Blink detection robust to various facial poses,” *Journal of Neuroscience Methods*, vol. 193, no. 2, pp. 356–372, Nov. 2010, doi: 10.1016/j.jneumeth.2010.08.034.

[21] L. Masanovic, M. Vranjes, R. Dzakula, and Z. Lukac, “Driver monitoring using the in-vehicle camera,” in *2019 Zooming Innovation in Consumer Technologies Conference (ZINC)*, Novi Sad, Serbia, May 2019, pp. 33–38. doi: 10.1109/ZINC.2019.8769377.

[22] L. M. Bergasa, J. M. Buenaposada, J. Nuevo, P. Jimenez, and L. Baumela, “Analysing Driver’s Attention Level using Computer Vision,” in *2008 11th International IEEE Conference on Intelligent Transportation Systems*, Beijing, China, Oct. 2008, pp. 1149–1154. doi: 10.1109/ITSC.2008.4732544.

[23] M. Elham Walizad, M. Hurroo, and D. Sethia, “Driver Drowsiness Detection System using Convolutional Neural Network,” in *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, Apr. 2022, pp. 1073–1080. doi: 10.1109/ICOEI53556.2022.9777182.

[24] R. Fusek, “Pupil localization using geodesic distance,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11241 LNCS, pp. 433–444, 2018, doi: 10.1007/978-3-030-03801-4_38.

[25] J. D. Ortega *et al.*, “DMD: A Large-Scale Multi-modal Driver Monitoring Dataset for Attention and Alertness Analysis,” in *Computer Vision – ECCV 2020 Workshops*, vol. 12538, A. Bartoli and A. Fusiello, Eds. Cham: Springer International Publishing, 2020, pp. 387–405. doi: 10.1007/978-3-030-66823-5_23.

[26] Abtahi, Shabnam, Omidyeganeh, Mona, Shirmohammadi, Shervin, and Hariri, Behnoosh, "YawDD: Yawning Detection Dataset." IEEE DataPort, Aug. 01, 2020. doi: 10.21227/E1QM-HB90.

[27] K. Diaz-Chito, A. Hernández-Sabaté, and A. M. López, "A reduced feature set for driver head pose estimation," *Applied Soft Computing*, vol. 45, pp. 98–107, Aug. 2016, doi: 10.1016/j.asoc.2016.04.027.

[28] E. N. Arcoverde Neto *et al.*, "Enhanced real-time head pose estimation system for mobile device," *ICA*, vol. 21, no. 3, pp. 281–293, Apr. 2014, doi: 10.3233/ICA-140462.

[29] Raspberry Pi Foundation, "Raspberry Pi 3 Model B," *Raspberry Pi*. https://www.raspberrypi.com/products/raspberry-pi-3-model-b/ (accessed Jun. 14, 2022).

[30] Raspberry Pi Foundation, "Raspberry Pi Documentation-Camera," *Raspberry Pi*. https://www.raspberrypi.com/documentation/accessories/camera.html (accessed Jun. 16, 2022).

[31] N. Singh, "MediaPipe Python Package (Unofficial) for RaspberryPi OS(32 bit) on Raspberry Pi 3 / 4," *PyPi*, Nov. 03, 2021. https://pypi.org/project/mediapipe-rpi3/#description (accessed Jul. 07, 2022).

ABSTRACT

In this thesis, Driver Monitoring System (DMS) is proposed based on the behavioral approach where the system detects the fatigue and inattention of the driver based on the images obtained from the camera mounted on the vehicle's dashboard. The driver monitoring dataset (DMD) and YawDD datasets were used to develop the model on a personal computer (PC) and the same model was implemented in the Raspberry Pi. The convolutional neural network (CNN)-based method MediaPipe Facemesh was used to detect a face and the localization of the landmarks. Also, the methods like the use of area over the distance ratio (ADR), mouth aspect ratio (MAR), and Euler angles were used to detect the eye, mouth state, and head position. The algorithm also calculates the adaptive threshold for MAR, ADR, and mean position of a face which will vary from individual to individual. The developed system is also tested using the Raspberry Pi camera.

Keywords: MediaPipe Facemesh, DMS, ADR, MAR, Euler angles, threshold, Raspberry Pi

BIOGRAPHY

RAJESH RIMAL

He was born on 30th Jan 1994 in Arjundhara-9, Jhapa district from Nepal. He completed his primary and secondary school at Deep Jyoti Vidya Mandir in Jhapa, Nepal in 2010. In 2012, he completed his higher secondary education at College for Higher Education (COHED) in Jhapa, Nepal. In the same year, he got admitted to the undergraduate study of Mechanical Engineering at Pulchowk Campus, Institute of Engineering (IOE) which he completed in 2016. In December 2017, he was awarded a "Bachelor's Degree in Mechanical Engineering" by Tribhuvan University. After completing his studies, he worked in automotive companies like Ashok Leyland and MAN Bus and Trucks as a Service Engineer for 4 years.

In 2020, he enrolled in the graduate study program named Automotive Computing and Communications in the Faculty of Electrical Engineering, Computer Science, and Information Technology (FERIT) at Josip Juraj Strossmayor University of Osijek.

Osijek, 14th July 2022

LIST OF FIGURES

Figure 3.1: FaceMesh face detection [26] (a - Original Image, b – Image of detected face in the green bounding box and 468 localized facial landmarks in red dots)... 9
Figure 3.2 Head orientation showing pitch, roll, and yaw movements [28]................................ 10
Figure 3.3: Detected eyes in green bounding boxes and localized pink points [26] 12
Figure 3.4 Eye Area over the Distance calculation [26]... 12
Figure 3.5 Detected mouth in the green bounding box and localized pink points [26]................ 13
Figure 3.6 Arrangement of points for calculation of MAR [26]... 14
Figure 4.1 Raspberry Pi 3 Model B board specifications 16
Figure 4.2 Raspberry Pi Camera Module 16
Figure 5.1 DMS flow chart 20
Figure 6.1 Examples of the image in the YawDD dataset [26] (a – Image from a video sequence named 2-Female.avi, b - Image from a video sequence named 7-Male.avi) [26] 30
Figure 6.2 FNs observed due to covering the mouth by hand [26]. (a –Yawn starts from Frame 679, b- Frame 720 where the mouth is still detected as open for 1.57 sec, c – Frame 721 where the mouth is closed by hand and timer resets, d – arbitrary Frame 754 where the mouth is detected open again for 0.38 sec after mouth was uncovered. 33
Figure 6.3 FNs observed due to short yawn time [26] (a – Yawn start from frame 313, b – Mouth open detected for 0.67 sec at frame 320, c – Mouth open detected for 1.82 sec at Frame 330, d – Mouth detected as closed in Frame 331 so that yawn is not detected).. 34
Figure 6.4 FPs observed due to laughing[26] (a- Frame 1960 from which yawn is detected as started, b – Frame 1985 after 1.37 seconds and mouth is still open, c – Person is detected as yawning even after 3.36 sec at Frame 2019) 35
Figure 6.5 Comparison of the F1-Score of PC and Raspberry pi 40
Figure 6.6 Normal Frame.. 41
Figure 6.7 Yawning detection (a – Individual starts to yawn at frame 1890, b – Arbitrary frame 1915 after 1.03 secs, c – Yawning detected after 2 seconds in frame 1940)............................... 41
Figure 6.8 Head Position Estimation (a –Turning right, b-Turning left, c- Turning down)......... 42
Figure 6.9 Eye long-closed detection (a - Individual close eye at frame 520, b – Arbitrary frame 540 after 0.81 secs, c – Eye long-closed detected after 2 seconds in frame 565)........................ 43

LIST OF TABLES

Table 3.1 Comparison of face detectors based on F-1 score and time to detect face 8
Table 5.1 Head position estimation conditions.. 24
Table 6.1 Labels description.. 30
Table 6.2 PC environment test results in terms of recall, precision, and F1-score....................... 31
Table 6.3 Execution Speeed in PC... 36
Table 6.4 Execution Speed in Raspberry Pi.. 37
Table 6.5 Raspberry Pi test results while skipping 5 frames 38
Table 6.6 Raspberry Pi test results while skipping 6 frames 38

LIST OF ALGORITHMS

Algorithm 1 Face detection and localization 21
Algorithm 2 Calculation of Pitch, Yaw, and Roll 23
Algorithm 3 Calculation of ADR 25
Algorithm 4 Detection of eye long-closed state 26
Algorithm 5 Calculation of MAR 27
Algorithm 6 Yawn detection 28

ABBREVIATIONS

WHO	World Health Organization
EU	European Union
NHTSA	National Highway Traffic Safety Administration
ADAS	Advanced Driver Assistance Systems
DMS	Driver Monitoring System
ECG	Electrocardiogram
EEG	Electroencephalogram
EOF	Electrooculogram
VJ	Viola-Jones
CNN	Convolutional Neural Network
RDCL	Rapidly Digested Convolutional Layer
MSCL	Multiple Scale Convolutional Layers
SSD	Single Shot Multibox Detector
CPU	Central Processing Unit
GPU	Graphical Processing Unit
VGG	Visual Geometry Group
VGA	Visual Graphic Array
MTCNN	Multitask Cascaded Neural Network
HOG	Histogram of Gradients
EAR	Eye Aspect Ratio

SVM	Support Vector Machine
SMAT	Simultaneous Modeling and Tracking
POSIT	Point Iterative Pose Estimation
RANSAC	Random Sampling and Consensus
PERCLOS	Percentage of Closure
DMD	Driver Monitoring Dataset
ADR	Area over Distance Ratio
MAR	Mouth Aspect Ratio
SoC	System on Chip
SD card	Secure Digital Card
CSI	Camera Serial Interface
DSI	Display Serial Interface
TN	True Negative
FN	False Negative
TP	True Positive
FP	False Positive
Fps	Frames per second
VI-DAS	Vision Inspired Driver Assistance Systems

ANNEXES

Annex I: Dataset details

Description		Dataset Name		
		DMD	Yaw DD	DrivFace
Resolution		1280 x 720	640 x 480	640×480
FPS		29.76	30.00	-
Type		Video	Video	Image
Format		MP4	AVI	JPG
No of Videos		20.00	29	-
Total no of Frames		192509.00	Each above 1400	606.0
Video Length		Each video is 1 min to 9 min	Each video around 1 min	-
No of Participants		5	29	4
Participants	Male	Yes	Yes	Yes
	Female	No	Yes	Yes
Other People in the Car		No	Yes	Yes
Other People's face detected		No	No	Yes
Scenarios	Car	Yes	Yes	Yes
	Simulator	Yes	No	No
Environment Condition	Sunny	Yes	Yes	Yes
	Cloudy	Yes	No	No
	Raining	Yes	No	No
	Night	No	No	No
Driving Mode	Stationary	Yes	Yes	No
	Driving	Yes	No	Yes
Camera Position	Car Dashboard	Yes	Yes	Yes
	Car front-Rear view mirror	No	Yes	No
Driver Wearing Glass	Prescription Glass	Yes	Yes	Yes
	Sunglasses	No	Yes	No

	No Glasses	Yes	Yes	Yes
Mouth Position		Normal-Talking-Yawning	Normal-Talking-Yawning	Normal-Talking-Yawning
Researchers are allowed to use pictures in their paper		Yes	Yes/No	Yes