DEVELOPMENT OF CAMERA-BASED DRIVER FATIGUE MONITORING ON RASPBERRY PI 4

MANUSSANAN SAENGPENCHAI

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN AUTOMOTIVE ENGINEERING

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2023

KMITL-2023-EN-M-037-018

THESIS EXAMINATION CERTIFICATION

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

Thesis Title	DEVELOPMENT OF CAMERA-BASED DRIVER FATIGUE MONITORING ON RASPBERRY PI 4
Student	MR. MANUSSANAN SAENGPENCHAI
Student Id	59610034
Degree	MASTER OF ENGINEERING
Program	AUTOMOTIVE ENGINEERING (INTERNATIONAL)
Thesis Advisor	ASSOC.PROF. DR. NATTAWOOT DEPAIWA
Thesis Co-Advisor	DR. CHADCHAI SRISURANGKUL
Thesis Co-Advisor	ASSOC.PROF. DR. MASAKI YAMAKITA
Thesis Reference Number	KMITL-2023-EN-M-037-018
Date	21th April 2023 Time 1.00 pm - 3.00 pm
Place	video call

(ASSOC.PROF.DR.SOMYOT KAITWANIDILAI)

DEAN SCHOOL OF ENGINEERING

14th Mrach, 2023

คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
แบบรับรองความถูกต้องของวิทยานิพนธ์

ชื่ออาจารย์ที่ปรึกษา รศ.ดร.ณัฐวุฒิ เดไปวา

ชื่ออาจารย์ที่ปรึกษาร่วม ดร.ฉัตรชัย ศรีสุรางค์กุล

ชื่อนักศึกษา มนัสนันท์ แสงเพ็ญฉาย

รหัสนักศึกษา 59610034

หลักสูตร วิศวกรรมศาสตร์มหาบัณฑิต

สาขาวิชา วิศวกรรมยานยนต์ (หลักสูตรนานาชาติ)

วิทยานิพนธ์เรื่อง

(ภาษาไทย)

(ภาษาอังกฤษ) DEVELOPMENT OF CAMERA-BASED DRIVER FATIGUE MONITORING ON RASPBERRY PI 4

ข้าพเจ้าในฐานะอาจารย์ที่ปรึกษาวิทยานิพนธ์ขอ**รับรองความถูกต้องของวิทยานิพนธ์ตามข้อบังคับสถาบันว่าด้วย การศึกษาระดับบัณฑิตศึกษา ปี พ.ศ. ๒๕๕๙**

ณัฐวุฒิ เดไปวา

(รศ.ดร.ณัฐวุฒิ เดไปวา)
อาจารย์ที่ปรึกษาวิทยานิพนธ์

คำสั่งคณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ที่ 193/2566
เรื่อง แต่งตั้งกรรมการสอบวิทยานิพนธ์ นายมนัสนันท์ แสงเพ็ญฉาย

- - - - - - - - - - - - - - - - - - - -

ตามที่ นายมนัสนันท์ แสงเพ็ญฉาย รหัสประจำตัว 59610034 หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมยานยนต์ (หลักสูตรนานาชาติ) ได้ทำวิทยานิพนธ์ เรื่อง "DEVELOPMENT OF CAMERA-BASED DRIVER FATIGUE MONITORING ON RASPBERRY PI 4 " โดยมี รศ.ดร.ณัฐวุฒิ เดโปวา เป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ และมี ดร.อัครชัย ศรีสุรางค์กุล , Assoc.Prof. Dr. Masaki Yamakita เป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ทั้งนี้ตามที่ประชุมคณะกรรมการประจำส่วนงานวิชาการ โดยผ่านการเวียนตามบันทึกข้อความที่ อว 7002(4)/e 0328 ในวันที่ 21 มีนาคม 2566 มีมติแต่งตั้งกรรมการสอบวิทยานิพนธ์ ดังนี้

1.	**ดร.ศราวุธ**	**เลิศพลังสันติ**	**ประธานกรรมการ**	**(ผู้ทรงคุณวุฒิภายนอกสถาบัน)**
2.	ดร.อัครชัย	ศรีสุรางค์กุล	กรรมการ	
3.	Assoc.Prof.Dr. Masaki	Yamakita	กรรมการ	
4.	รศ.ดร.ปรีชา	การินทร์	กรรมการ	
5.	รศ.ดร.ณัฐวุฒิ	เดโปวา	กรรมการและเลขานุการ	

ทั้งนี้ ให้ดำเนินการจัดสอบในวันศุกร์ที่ 21 เมษายน พ.ศ. 2566 เวลา 13.00 – 15.00 น. ทางออนไลน์แบบ Video Call

สั่ง ณ วันที่ 30 มีนาคม พ.ศ.2566

(รองศาสตราจารย์ ดร.สมยศ เกียรติ์นิชวิโล)
คณบดี คณะวิศวกรรมศาสตร์

หัวข้อวิทยานิพนธ์	การพัฒนาอุปกรณ์ตรวจจับความเหนื่อยล้าของคนขับรถโดยใช้ราสเบอร์รี่พาย 4
นักศึกษา	นาย มนัสนันท์ แสงเพ็ญฉาย
รหัสประจำตัว	59610034
ปริญญา	วิศวกรรมศาสตร์มหาบัณฑิต
สาขาวิชา	วิศวกรรมยานยนต์ (หลักสูตรนานาชาติ)
พ.ศ.	2566
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ. ดร. ณัฐวุฒิ เดไปวา
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	ดร. ฉัตรชัย ศรีสุรางค์กุล
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	Assoc.Prof. Dr. Masaki Yamakita

บทคัดย่อ

ประเทศไทยเป็นหนึ่งในประเทศที่มีอุบัติเหตุบนท้องถนนมากที่สุดเป็นอันดับต้นๆของโลก หนึ่งในสาเหตุที่สำคัญคือ อุบัติเหตุที่เกิดขึ้นจากคนขับมีอาการหลับใน ซึ่งอุบัติเหตุประเภทนี้มักเกิดขึ้นกับรถโดยสารสาธารณะที่ใช้เดินทางข้ามจังหวัดโดยเฉพาะในช่วงเทศกาลวันหยุดยาวที่ผู้คนมีความต้องการเดินทางเป็นจำนวนมาก ทำให้คนขับต้องแอบทำงานล่วงเวลา ถึงแม้ที่ผ่านมาภาครัฐจะมีกฎหมายเพื่อช่วยลดความเสี่ยงในการเกิดอุบัติเหตุเช่น การควบคุมชั่วโมงการทำงานของคนขับ การตรวจวัดระดับแอลกอฮอล์ก่อนออกเดินทาง การควบคุมระดับความเร็วของรถโดยสาร แต่ทุกปียังคงมีข่าวที่รายงานอุบัติเหตุที่เกิดจากการหลับในของคนขับอยู่ ซึ่งคนขับบางคนก็รู้ตัวเองว่าเริ่มล้ายังสามารถแวะจุดพักรถเพื่อพักผ่อนได้ แต่ในบางคนก็มีอาการหลับในแบบไม่รู้ตัว ดังนั้นโครงงานนี้จึงได้ทำการพัฒนาและอัลกอริทึมที่สามารถใช้งานบนมินิคอมพิวเตอร์ในที่นี่คือราสเบอรรี่พาย 4 (Raspberry Pi 4) โดยสามารถตรวจจับสถานะของคนขับตามเวลาจริง (Real-time) ขึ้น โดยมีหลักการทำงานคือจะมีกล้องขนาดเล็กคอยบันทึกหน้าของคนขับรถไว้ ถ้าคนขับมีอาการกระพริบตาช้ากว่าปกติติดต่อกันในช่วงเวลาหนึ่ง มีการหลับตานานกว่าปกติ อัลกอริทึมจะทำการวิเคราะห์ความล้าของคนขับแล้วทำการส่งเสียงเตือนเพื่อให้คนขับรู้สึกตัว โดยการเลือกใช้ราสเบอรี่พายเพราะมีขนาดเล็ก สามารถติดตั้งได้ที่คอนโซลหน้าของรถทุกๆประเภทไม่ว่าเก่าหรือใหม่โดยอาศัยแหล่งจ่ายไฟภายในรถยนต์ งานวิจัยนี้หวังว่าจะสามารถพัฒนาอัลกอริทึมที่สามารถใช้งานบนราสเบอรรี่พายโดยสามารถทำงานตามเวลาจริงได้ และสามารถช่วยเตือนคนขับเมื่อเกิดความล้าได้ซึ่งจะเป็นประโยชน์และช่วยเพิ่มความปลอดภัยบนท้องถนนได้

Thesis Tittle	Development of Camera-based Driver Fatigue Monitoring on Raspberry Pi 4
Student	Mr. Manussanan Saengpenchai
Student ID	59610034
Degree	Master of Engineering
Program	Automotive Engineering (International Program)
Year	2023
Advisor	Assoc.Prof. Dr. Nattawoot Depaiwa
Co-advisor	Dr.-ing Chadchai Srisurangkul
Co-advisor	Assoc.Prof. Dr. Masaki Yamakita

ABSTRACT

One of the top causes of car accidents today is drowsy driving referred to The National Highway Traffic Safety Administration (NHTSA) of The US car accidents report, this is commonly called as driver fatigue. Driver fatigue is unable to adequately perceive, react and respond to situations on the road. The way to make sure the driver concentrates on the road and does not doze off while driving, the state of the driver in real-time should be known to confirm it. This study develops visual-based driver fatigue monitoring for this purpose. The system uses the Infrared camera for suitable in any light conditions to detect symptoms of the driver such as eyelid distance, eye blink rate, and eye saccadic movement to estimate driver alertness based on extracted symptoms and alarms if needed. The system uses Python language for machine learning analysis and uses Raspberry Pi 4 for processing the data. This system desired to be applied on every type of vehicles without considering size and age and can operate in any situation such as daytime, nighttime, and raining time to improve the public safety and reduce accidents on the road. It will be beneficial for the driver, passenger, and everyone on the road

ACKNOWLEDGEMENT

I would like to express my very great appreciation to Dr.-ing Chadchai Srisurangkul, my research advisor who always assists with research, work, and the care that is provided. Thanks for the equipment support, participant contact, and being a participant, which made collecting the dataset much smoother. Thank you to Asst. Prof. Dr. Chaiwat Nuthong, Assoc. Prof. Dr. Nattawoot Depaiwa and Assoc. Prof. Dr. Masaki Yamakita as co-advisors and this work might not succeed without their patient guidance, enthusiastic encouragement, and useful critiques of this research. Also, I would like to thank Asst. Prof. Dr. Preecha Karin for consistent very kind support through graduation.

I would like to thank Thailand Advance Institute of Science and Technology, Tokyo Institute of Technology (TAIST-Tokyo Tech) and National Science and Technology Development Agency (NSTDA) for providing full scholarship and financial support. I wish to thank Driving and Vehicle Technology Research Team (DVTT), National Metal and Material Technology Center (MTEC) for instrument and materials used for this research succeed.

Finally, I would like to thank my family and my important persons: Pichamol Thirasuppasri, Savitri Payakkapol, and Bongkotchaporn Duangsrikaew for their support and encouragement throughout my research.

Manussanan Saengpenchai

TABLE OF CONTENTS

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF DEFINITIONS

EEG	*Electroencephalogram*
KSS	*Karolinska Sleepiness Scale*
B-ORS	*Behavioral Observers Rating Scale*
D-ORS	*Driving performance Rating Scale*
NHTSA	*National Highway Traffic Safety Administration*
MTEC	*National Metal and Materials Technology Center*
NSTDA	*National Science and Technology Development Agency*
RPI4	*Raspberry Pi 4*
FPS	*Frame per second*
EAR	*Eye aspect ratio*
MAR	*Mouth aspect ratio*
BB	*Bounding box*
ROI	*Rectangle of interest*
TP	*True positive*
TN	*True negative*
FP	*False positive*
FN	*False negative*
P	*Precision*
R	*Recall*

CHAPTER 1

INTRODUCTION

1.1 Research background

According to Thailand's accident reports, road accidents are the leading cause of all forms of accidents. This is consistent with the Ministry of Transport's road accident analysis report that reported 19,904 deaths in 2019. Thailand has the world's ninth highest road fatality rate and the highest in Asia. The top 5 causes of accidents are: Overspeed driving (70.94%), Vehicle or object cut-off suddenly (8.25%), Drowsy driving (7.02%), Defective equipment (3.31%), and Drunk driving (2.84%) [1]. The months with the most accidents are January and April, which is the festival season. The majority of incidents are triggered by the driver's perceived behavior and consciousness.

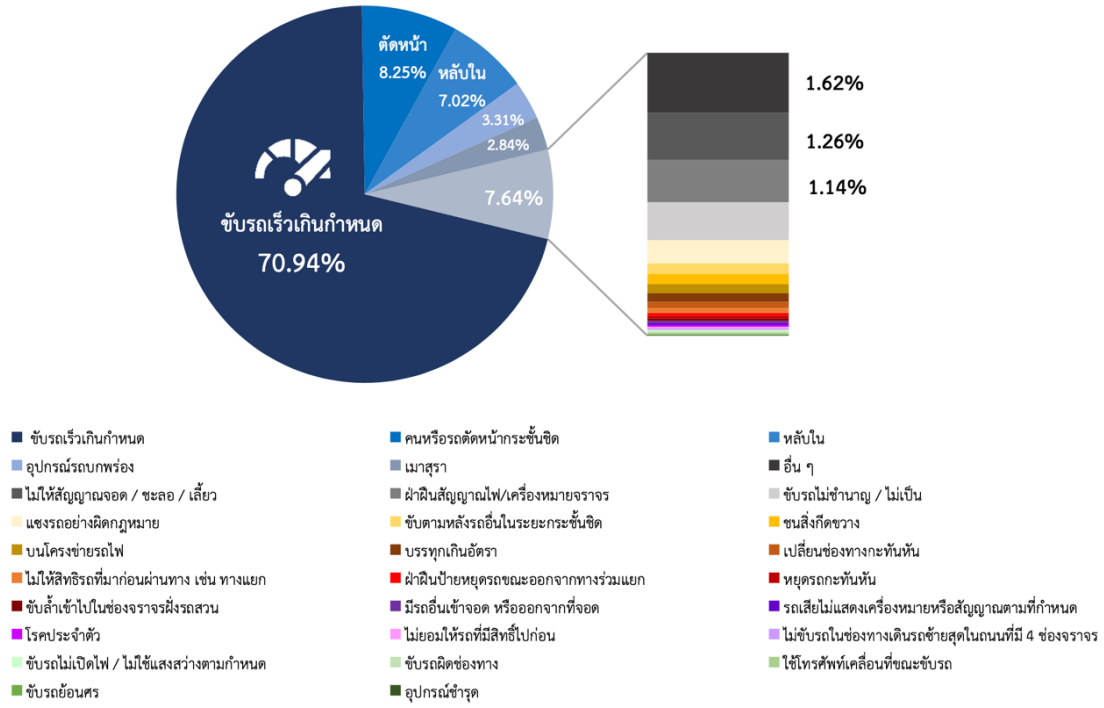


Figure 1.1 Overview of main causes of accidents in 2019

Traffic laws may prevent drivers from speeding and reduce the severity of accidents. However, controlling the driver's consciousness is difficult, many times the driver is unaware. Accidents from falling asleep in trucks and buses are often caused

by the driver having to drive for an extended period. When an accident occurs, the severity and injury rates of accidents and death are high compared to other types of accidents, so the driver's consciousness, awareness or status is great importance [2]. Autonomous driving systems that do not require the driver to drive themselves will become more common in the future, but drivers are not permitted by law to let the system run while they are sleeping or to ignore traffic [3]. A vehicle equipped with a system that tracks the driver's alertness levels should assist the driver in assessing his or her condition or warn the driver to be more mindful and increase the safety of the vehicle on the road. Therefore, nowadays many modern cars are equipped with a system called Driving Pattern Monitoring, which tracks driver activity or driving patterns. Some cars have a front-facing camera that tracks lane lines. If the system detects repeated attempts to drive outside the lane without turning on the turn signal lights or driving swaying around, it assumes that the driver has less control over the vehicle that it is potentially dangerous and will alert the driver to rest. Visual-based Driver Monitoring, which uses a video camera to identify the driver's face symptoms and transmute the video to assess the current situation, is another common method of tracking the driver's status. These safety systems are not yet widely used, but only in newer vehicles and in passenger cars. Many vehicles, including passenger cars, buses, or trucks, are also likely to lack driver-related safety systems.

For the above reason, the aim of this research is to create a system that can detect the driver's status using a camera. This system must be tiny and simple to mount. Avoid interfering with other equipment or vehicle systems. The driver is not required to mount any equipment on the body. It can be mounted on a variety of vehicles, allowing it to be widely used. And create algorithms to detect the status of the driver, which can be processed on a microcomputer using a Raspberry Pi 4 to operate in real time. Capable of detecting the level of driver drowsiness and sounding an alarm if it falls within the limits of an accident. Driving data from volunteers was collected using a driving simulator to improve the accuracy of the established algorithm. The author sincerely hopes that the developed equipment and systems can help to minimize accidents caused by driver fatigue.

1.2 Research objective

1) Study and understand the behavior of fatigue and sleep deprivation.

2) Develop algorithms that can reliably calculate fatigue levels and process them in real-time.

3) Design the driving simulation data collection and build the appropriate environment for use with the driving simulator.

1.3 Thesis outlines

This thesis is divided into 6 chapters. The contents are summarized as followed:

Chapter1: Introduction

This chapter introduces the motives for conducting studies, the general background of driver fatigue monitoring, scope of work, and the objectives.

Chapter2: Literature review

This chapter reviews the fatigue behaviors and symptoms, driver fatigue monitoring methods, fatigue level, dataset collecting, and the necessary algorithm.

Chapter3: Research methodology

This chapter explains the details of system workflow, image quality improvement, detection method, symptoms extraction method, fatigue level decision method, dataset collecting method, and the importance parameters from data collected.

Chapter4: Result

This chapter presents driver detection performance, eye blinking and mouth opening detection, and fatigue level detection in terms of processing performance and accuracy. The relationship between fatigue level and driver facial features behaviors.

Chapter5: Conclusion and recommendations

This chapter concludes the satisfaction and dissatisfaction of this work. The factor should concern data collecting and algorithm development. Recommendation for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Fatigue behaviors and symptoms

Fatigue is the transitory period between wake and sleep, and it can lead to sleep if uninterrupted. To better define fatigue, it is useful to know that it can be divided into physical and mental categories [4]. Muscle fatigue is called the syndrome of decreased muscle performance after stress and is characterized by lower muscle power and motion. The risk of exhaustion can be increased by hard physical work. Muscle fatigue is similar to physical fatigue. The most common symptom of mental fatigue is a general feeling of weariness, inhibition, and impaired function. Physical and mental exhaustion can both affect vigilance and task performance.

Fatigue and drowsiness are caused differently and based on different processes but usually considered together because the effects are the same. A person who is fatigued or exhausted is less alert or attentive and can fall asleep in severe cases [5].

2.1.1 Factor of fatigue

We will examine the factors that cause fatigue by examining the following characteristics of drowsy driving accidents [6].

- Occurring at night or in mid-afternoon, more than 40% of crashes occur between 1 am and 7 am.
- Occur on high-speed highway, straight road. About 70% of crashes occur on rural highways with 55 to 65 mph speed limit.
- Involve only the driver as occupant.
- Involve a single vehicle running off the roadway or rear-end and head-on collisions. No signs of braking the vehicle.

2.1.1.1 Behavioral factor

Sleep deprivation is the most common cause of drowsiness. Many studies have reported the negative impact of sleep deprivation on driving. Sleeping less than 4 hours a night has a negative impact on driving efficiency [6]. The length of the last sleep cycle and the amount of sleep in the previous 24 hours are the two most significant contributing factors in distinguishing between fatigue and non-fatigue related accidents.

Due to circadian rhythms, people feel sleepy during the afternoon and evening. Drivers who drive at night are more likely to be involved in an accident [5].

2.1.1.2 Environmental factor

A situation is shown to be monotonous when the stimuli remain constant or change in a predictable pattern. A variety of studies have implicated the monotony of roadway geometry and environment as a cause of driver drowsiness. Sleep-related accidents may be more frequent on long stretches of major highways, accounting for up to 40% of fatal accidents. Straight road sections reduce driving performance more than curves. The lower vehicle density makes the driver also bored [4].

2.1.2 Indicator of fatigue

Physical such as head, eyes, mouth, or body movement, can indicate fatigue. Table 2.1 describes the fatigue symptoms while driving based on each indicator to explain.

Table 2.1 The physical indicators of fatigue

Indicator	Normal	Fatigue
Head	Look straight ahead.	Head bending/nodding [7]
Eyes	Blink duration < 400ms	Blink duration > 400ms Blink duration > 800ms is assume as micro-sleep [8]
	Eye-blinking about 15-20 times per minutes	Eye-blinking activity radically increased [15]
	Look straight ahead	Looks in other directions for an extended time

Mouth		Yawning
Body movement	Sitting still	Some arms and legs were scratched. Stretching, Slumped [9]

Although there are a variety of physiological measures for measuring alertness, the brain activity signal may be one of the most accurate and reliable [4]. Electrical brain activity is classified by rhythms defined by the delta, theta, alpha and beta frequency bands.

Table 2.2 The Electroencephalography band

Waveband	Frequency	Status
Delta	0.5 to 4 Hz	Occurs during the transition to drowsiness and sleep.
Theta	4 to 7 Hz	Occurs in association with hypnagogic imagery and low levels of alertness.
Alpha	8 to 13 Hz	Occurs during wakefulness
Beta	13 to 30 Hz	Occurs in association with alertness, arousal, and excitement.

The most significant symptom of fatigue appears in the eye after many studies of brain activity monitoring. In terms of EEG band power, blink parameters, and eye movements, drowsiness has been classified into three stages. The first stage of drowsiness is reduced vigilance, which can be represented by increased EEG theta band power and decreased eye movements. The second stage is sleeping proclivity, which is distinguished by longer blink duration and longer lid reopening. Increased blink rate indicates the final stage in which the driver almost completely loses the ability to react to driving events [10].

2.2 Fatigue detection method

This is important to note that a detector of driver fatigue impairment must have a high detection capability with a low risk of false alarm. The most important approaches for fatigue/distraction detection can be divided into four categories [11].

2.2.1 Physiological measures

This method uses physiological activity such as brain activity, heart rate, or skin temperature to distinguish between drowsiness and wakefulness in drivers. The electrodes or probably the headband is used to collect physiological signals. Electrodes are appropriate for clinical applications but since real-world monitoring causes discomfort [10]. For example, from Figure 2.1 the device needs to intrusive with subject.

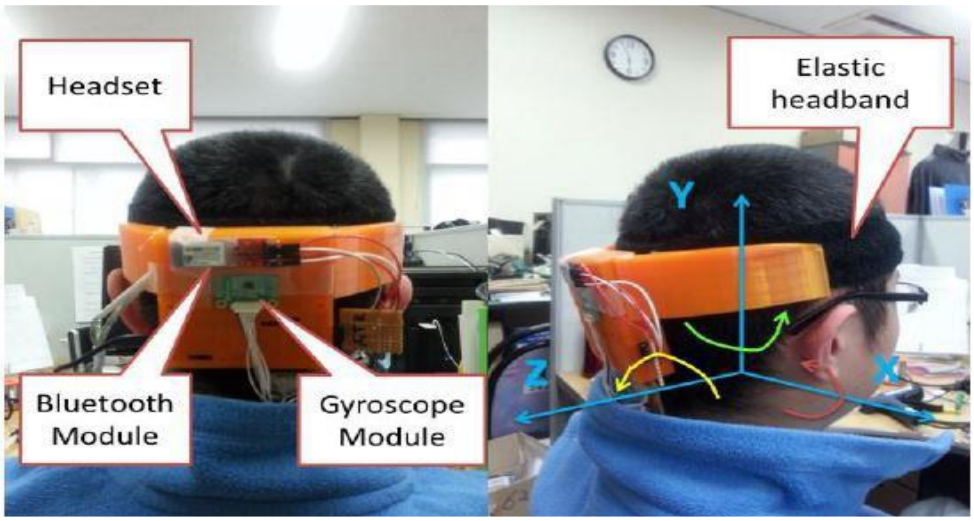


Figure 2.1 EEG headset system for drowsiness detection

2.2.2 Vehicle-based measures

The drowsiness of the driver is assessed using vehicle control systems, which may include steering wheel movements, braking patterns, and lane departure measurements. Steering wheel measurements are more accurate than other vehicle-based methods. Vehicle-based methods may not be as reliable in the accurate detection of sleepiness because they rely on the nature of the road and on the driving abilities of the driver [12].

2.2.3 Behavioral measures

Eye activity attracted attention regarding in-car development for the monitoring of sleepiness of drivers. The driver face monitoring system is a real-time system that investigates a driver's physical and mental condition based on the processing of driver

facial image. Eyelid closure, blinking, gaze direction, yawning, and head movement could all be used to estimate the driver's status [11].

In automotive industry, the driver monitoring system of each manufacture are describe in Table 2.3. The two main monitoring systems based were used are vehicle-based and driver behavioral-based.

Table 2.3 The commercially driver fatigue detection system

Brands	Technology	Monitoring device	Detection parameter	Detection method
Mercedes - Benz (2009)	Attention Assist	Sensor on steering column	Steering wheel movement	Vehicle-based measures
Toyota / Lexus (2006)	Driver monitoring system	Charge-coupled camera	Eye tracking Head motion	Behavioral measures
Volvo (2007)	Driver Alert Control	- Camera - Steering wheel sensor	Road line detection with lane keeping	Vehicle-based measures
Ford (2011)	Driver alert control	Camera	Lane position	Vehicle-based measures
Volkswagen (2015)	Driver Fatigue Detection	Sensor on steering wheel	Steering wheel movement	Vehicle-based measures

2.2.4 Subjective measures

This method uses a questionnaire or an observer to determine the driver's level of fatigue. Karolinska Sleepiness Scale (KSS) is a popular reference fatigue level for estimating fatigue. The KSS scale can conclude 10 levels of fatigue ranging from extremely alert to extremely sleepy. According to a Swedish study, because the KSS has too many levels and is difficult to define by observers, this study attempted to

divine the fatigue level into three levels called Behavioral signs of sleepiness (B-ORS), which are described in Table 2.4.

Table 2.4 The KSS level relate to B-ORS level

Karolinska Sleepiness Scale (KSS)	Behavioral signs of sleepiness (B-ORS)
Lv1 Extremely alert	Lv0 Alert
Lv2 Very alert	
Lv3 Alert	
Lv4 Rather alert	
Lv5 Neither alert nor sleepy	
Lv6 Some signs of sleepiness	Lv1 First sings of sleepiness
Lv7 Sleepy, but no effort to keep awake	
Lv8 Sleepy, but some effort to keep awake	Lv2 Severe sleepiness (Micro-sleep)
Lv9 Very sleepy, great effort to keep awake, fighting sleep	
Lv10 Extremely sleep, cannot keep awake	

The driving performance into three stages based on the B-ORS level as shown in following table.

Table 2.5 The B-ORS level relate to driving performance stage

Behavioral signs of sleepiness (B-ORS)	Driving performance impairment (D-ORS)
Lv0 Alert	Lv0 Alert
Lv1 First sings of sleepiness (short periods of long blink duration, some yawn, some changes in body position)	Lv1 First sings of sleepiness (normal reaction, minor wobbling)
Lv2 Severe sleepiness (Micro-sleep) (half-closed eyes, changes in body position, head nodding)	Lv2 Severe sleepiness (Micro-sleep) (no capability to fight sleepiness, severe reduction in driving performance)

The overall fatigue cause and method of detection relation can describe by the following chart.

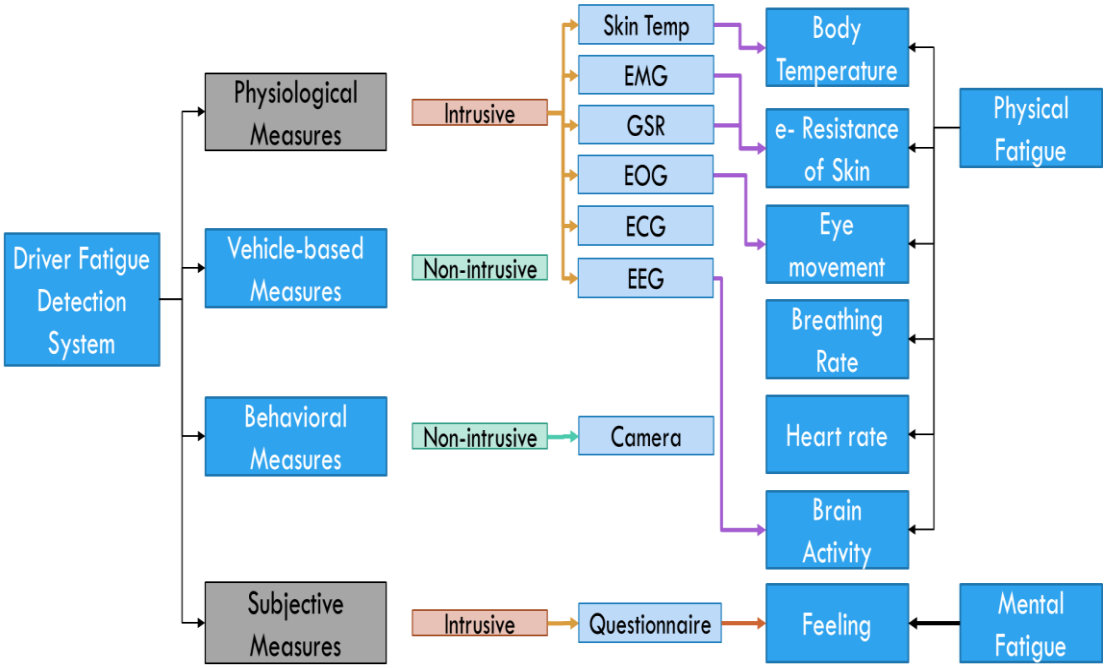


Figure 2.2 The fatigue cause and detection method relation

2.3 Dataset collecting

Collecting the participant fatigue data for the purpose of research, the various participants and the driving scenario environment are the necessary factors to consider.

2.3.1 Participant

Choosing the sample of participants should be a variety of samples with a range of ages suitable for driving (Over 18 years old, refer from Department of Land Transport of Thailand) [6][13]. In general, the sample should be divided into two groups: full rest and partial rest (less than 4 hours) and some case had the no rest at all group [14], with a large sample size required to effectively categorize fatigue characteristics [10].

2.3.2 Scenario

Based on statistical data on the characteristics of accidents involving driver drowsiness, the driving scenario should mainly consist of long and monotonous driving

sections such as straight roads with less traffic and long running distances [13] that are likely to induce driver boredom and fatigue when studying driver drowsiness [6].

2.3.2.1 Real-world driving

The experiment should be strict for the sake of safety in a real-world driving scenario. Testing on a rural road with a low car density, such as in the late hours of the night, as in the Norway study [13]. The participant as the driver is accompanied by an observer as the passenger to determine the level of fatigue.

2.3.2.2 Driving simulator

Driving simulator scenarios are the most popular method of testing because they are safe from accidents and allow the driver to fall asleep while testing. At the very least, the driving simulator scenario settings should be related to National Highway Traffic Safety Administration (NHTSA) basic components by the following criteria [6]:

- Physics of the vehicle and interaction with road surface
- Surrounding environment including other vehicles
- Integration of informative systems (sounds, motion, etc.)
- Integration of control devices (steering wheel, pedal)

2.3.2.3 Kind of data

In many fatigue studies, the data that was collected from the experiment are the driving pattern, driver video recorded, fatigue level by questionnaire and observation.

Most of the data collected in the driver fatigue experiment consisted of vehicle parameter data, eye closure data, and video image during the experiment [6]. In some experiments, level of fatigue (KSS) is also asked during the test [9].

2.4 Drowsiness detection in computer vision

The steps to detection in the camera-based fatigue monitoring system begin with face detection and end with an alarm warning decision algorithm, as shown in the following flowchart.

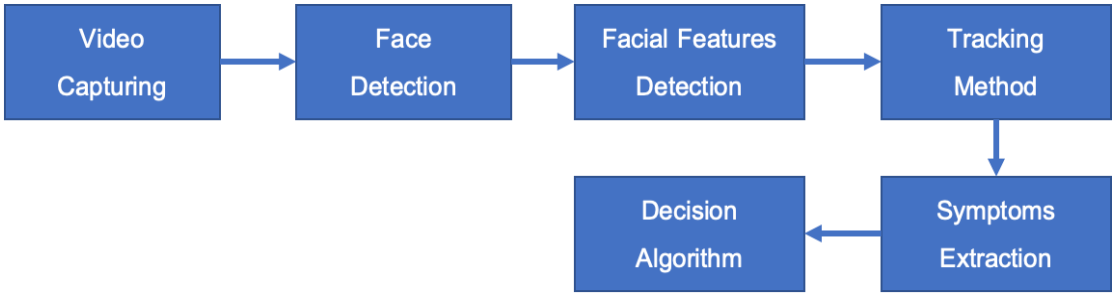


Figure 2.3 The camera-based fatigue monitoring workflow

2.4.1 Face detection algorithm

Face detection is the first step in most driver face monitoring systems [11]. This is a major component of systems that detect fatigue and distraction based on processing of the facial region. Due to the difficulty of directly detecting eyes, the face is detected first, followed by the eyes. The main facial detection problems are:

- In plane / out plane face rotation image of face can vary due to relative camera-face pose.
- Covering part of the face such as mask, sunglasses Facial features such as beards, mustaches, and glasses.
- Faces may be partially occluded by other objects.
- Imaging conditions such as lighting and camera characteristics affect the appearance of a face.
- Real-time processing, which is an important factor in processing large video archives.

There are four types of face detection methods, Knowledge-based, Feature-based, Template Matching, and Appearance-based methods [15][16]. There are many

techniques to detect faces, with the help of these techniques, we can identify faces with higher accuracy. These techniques have almost the same procedure for Face Detection such as OpenCV, Neural Networks, Matlab, etc.

From [17] study, they study Both MATLAB and OpenCV face detection that can be used for creating such prototypes and systems but they have the reason to choose OpenCV. Since, MATLAB is fabricated from java which in turn is fabricated from C. Therefore, when a code is scripted and run-on MATLAB, the computer initializes by interpreting the code and converting it into java and then finally executes the script. Whereas OpenCV uses C/C++ library functions. Which directly provides the computer with the machine language code and hence helps in faster execution. About the cost, MATLAB costs around USD$2150 whereas OpenCV is open source, so it is free of cost. The most efficient way of face detection is HAAR-Cascade based.

For the image results obtained from object detection, the most basic metric to measure the accuracy are [18]:

1) True positive: The model predicts that a face already exists in the frame, that face exists.

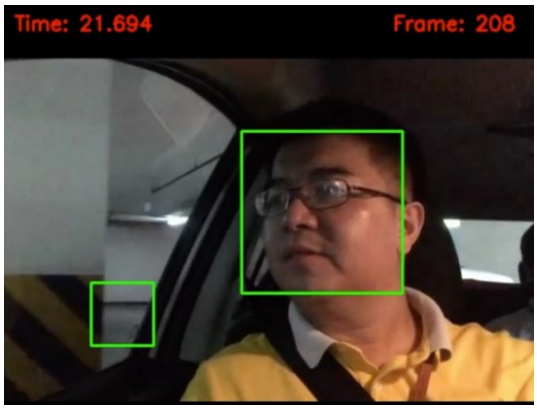
2) False positive: The model predicts that a face exists but there are no faces in the frame.

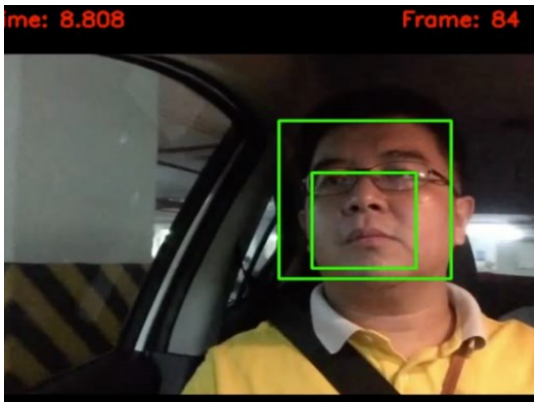
3) False negative: The model predicts that there are no faces, but that there are faces in the frame.

4) True negative: The model predicts that there are no faces and that in the frame there are no faces.

(a) True positive

(b) False negative





(c) False positive (Ghosting), (d) False positive (Face overlap)

Figure 2.4 Image results of face detection

In object detection, True negative (TN) result does not apply as there are infinitive number of bounding boxes that should not be detected in any given image.

The variables that will be used to consider the accuracy of detection are:

1) Precision: The percentage of correct positive prediction

2) Recall: The percentage of correct positive prediction among all given ground truth.

3) F1-score: The value that averages the above two parameters to determine overall performance.

Those 3 variables can be defined in the following equation:

$$P = \frac{TP}{TP+FP} = \frac{TP}{all\ detection} \quad (2\text{-}1)$$

$$R = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truth} \quad (2\text{-}2)$$

$$F1score = \frac{P \times R}{\frac{P+R}{2}} \quad (2\text{-}3)$$

Where: P = Precision, R = Recall,

TP = True Positive, FP = False Positive, FN = False Negative

A good object detector should find all ground-truth objects (FN = 0; high recall) while identifying only relevant objects (FP = 0; high precision). Therefore, a particular object detector can be considered good if its precision stays high as its recall increases,

which means that if the confidence threshold varies, the precision and recall will still be high [18].

2.4.2 Face tracking algorithm

Face tracking is the process of locating a moving face or several faces over a period in a video. The main purpose of these processes is to detect and track the face even in poor viewing conditions in surveillance applications. After face detection, feature extraction and tracking were performed to achieve face tracking [18].

The face tracker then analyzes the subsequent video frames and outputs the location of the initialized face within these frames by estimating the motion parameters of the moving face. This is different from face detection, the outcome of which is the position and scale of one single face in one single frame. More importantly, these faces have the same identity.

Since one face track corresponds to one identity, unlike in frame-based face detection, the workload of intra-shot face matching is greatly reduced [19].

2.4.3 Facial features detection

Facial feature detection in images and video deals with estimating the position of prominent features of a human face on a facial image. Facial feature detection methods utilize information regarding facial geometry, luminance, color and/or shape information of facial features.

The published methods involve a face detection step and seek for facial features only within the detected face region. These methods give more robust results than the second methods that do not involve face detection and search for facial features over the entire image [20].

The detected features are, in most cases, the eyes, the eyebrows, the nose and the mouth. The detected features are, in most cases, the eyes, the eyebrows, the nose and the mouth.

To know the driver's status, observing the driver's facial features and converting them into numerical values that can tell the status is the key. For most of the observational organs are the head, eyes, and mouth. In express the associated of facial features with symptoms indicating fatigue.

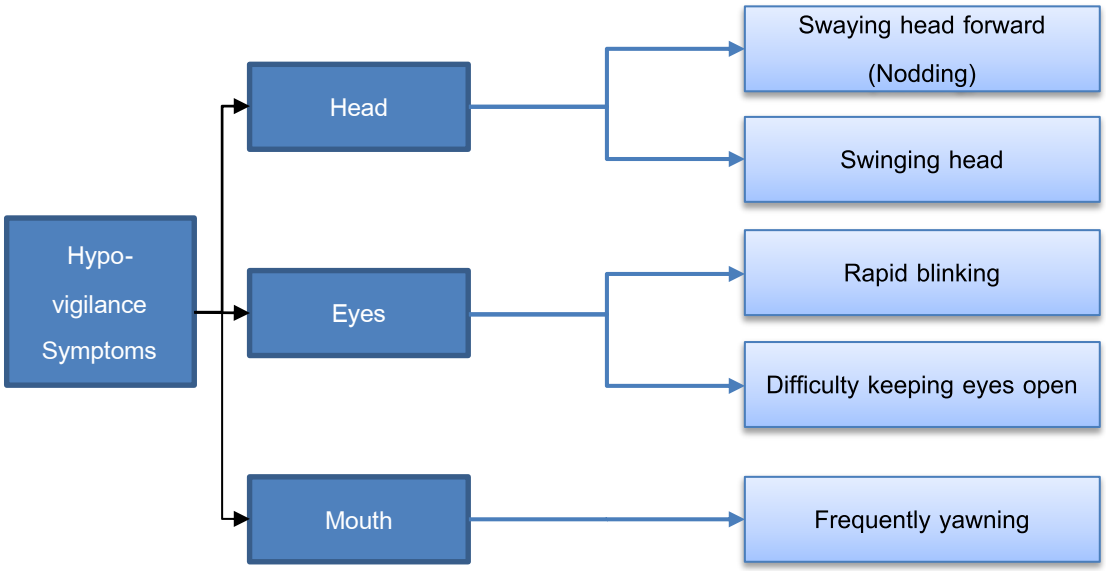


Figure 2.5 Fatigue symptoms with facial features

There are several methods for detecting the features of the face from detecting the whole face to detecting each feature separately which are divided into the following topics.

2.4.3.1 Facial landmark detection

In computer vision, to automatically extract that facial information, the localizations of the fiducial facial key points (as shown in Figure 2.6) are usually a key step and many facial analysis methods are built up on the accurate detection of those landmark points.

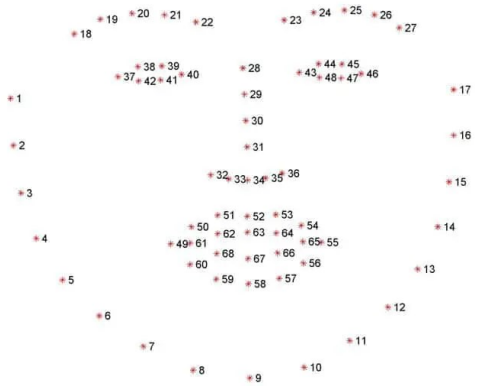


Figure 2.6 Facial landmark location

For example, facial expression recognition and head pose estimation algorithms may heavily rely on the facial shape information provided by the landmark locations. The facial landmark points around eyes can provide the initial guess of the pupil center positions for eye detection and eye gaze tracking.

Facial landmark detection algorithms aim to automatically identify the locations of the facial key landmark points on facial images or videos. This detection is challenging for several reasons. First, facial appearance changes significantly across subjects under different facial expressions and head poses. Second, facial occlusion by other objects or self-occlusion due to extreme head poses would lead to incomplete facial appearance information.

The facial landmark detection algorithms can classify into three major categories: Holistic methods, Constrained Local Model (CLM) methods, and the Regression-based methods.

There are many existing facial landmark algorithms. From [21] study, there are several observations found the Regression based methods achieve much better performances than the Holistic methods and the Constrained local model methods, especially on images with significant variations. They compare the efficiency comparison of leading algorithms that are described by the following table.

Table 2.6 The efficiency comparison of leading algorithm [21]

Methods	Type	Points	FPS
TCDCN	DR	5	58
HyperFace	DR	21	5
Consensus of Exemplars	C	29	1
3DDFA	DR	68	13
CFAN	DR	68	40
CFSS	R	68	25
Supervised Descent Method	R	68	30
3D Regression	R	68	111
Explicit Shape Regression	R	87	345
RCPR	R	194	6
One Milisecond Face Alignment	R	194	1000
Face Alignment 3000fps	R	194	200 / 1500

Where: Type H = Holistic methods, C = Constrained local methods, R = Regression based methods, DR = Deep learning-based Regression methods

The results show that the One Millisecond Face Alignment, the Supervised Descent method, and CFSS are good options considering both the speed and accuracy. There are still a few problems with facial landmark detection. First, the current facial landmark detection and tracking algorithms still have problems on facial images under some conditions, including extreme head poses, facial occlusion, strong illumination, etc. Second, facial landmark detection still heavily relies on the face detection accuracy, which may still fail in certain conditions. Third, the computational cost for some landmark detection and tracking algorithms is still high, that affects the real-time processing requirement.

2.4.3.2 Eye detection

Because the most important symptoms are related to the activity of the eyes, the eye region is always processed for symptom extraction in all driver face monitoring systems. As a result, eye detection is required prior to processing of the eye region [11]. Given an arbitrary face image, the goal of eye detection is to determine the

location of the eyes. Simply in eye detection, the areas where both eyes are located are found or two eyes individually localized.

A lot of work on eye detection area such as eye pupil movement detection, eye feature extraction, eye state detection, eye gaze detection using different techniques both in still images and in video sequences for real-time applications. This type of work is more difficult in computer vision as detection or real-time tracking of small details are highly affected from varying ambient conditions and result may easily fail.

Eye detection is generally dependent on face recognition. With the help of algorithms finding location of faces in images, eye detection methods show better performance. There are new approaches that try to find eyes, without detection of faces in images which also shows great performance [15]. The various of eye detection methods can be shown in Figure 2.7.

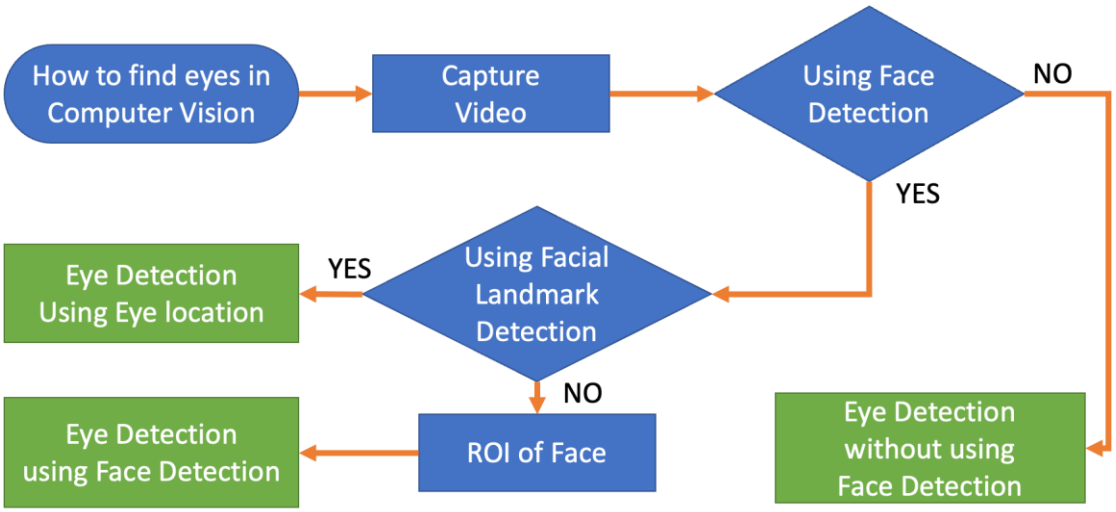


Figure 2.7 Eye detection methods flowchart

2.4.3.3 Head pose estimation

Head pose estimation is a crucial initial task for human face analysis, which is employed in several computer vision systems, such as: facial expression recognition, face recognition, head gesture recognition, gaze recognition, driver monitoring, etc. [22].

In computer vision, face pose estimation is defined as the process of deducing the face orientation from the 3D coordinates in the world space to the 2D coordinates

in the image space [23]. The face is usually modeled as a rigid object, with three DOF in the pose characterized by three rotation angles: pitch, roll and yaw. Process can describe by Figure 2.8.

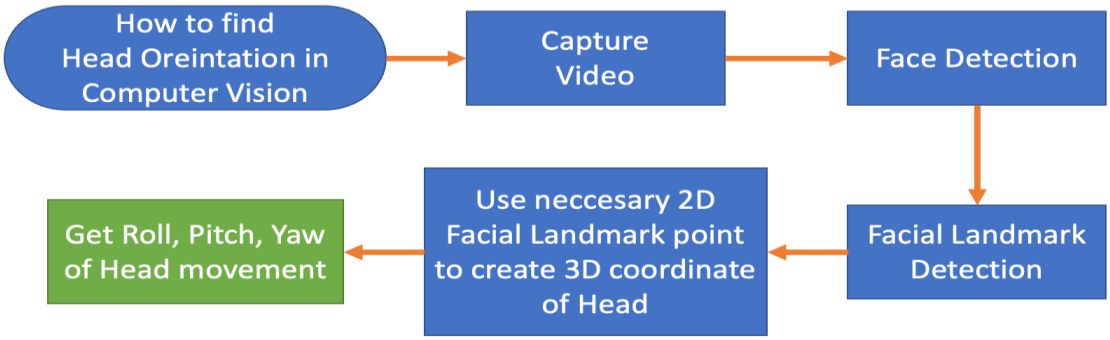


Figure 2.8 Head pose estimation method flowchart

With a human head facing the camera, yaw is the angle of moving the head left and right (rotation around the Y-axis); the pitch is that of moving the head up and down (rotation around the X-axis); and roll is the tilt angle (rotation around the Z-axis) as shown in Figure 2.9.

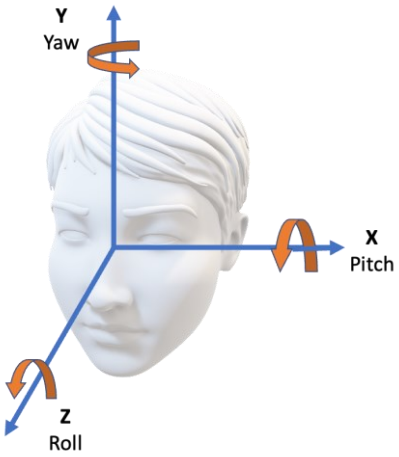


Figure 2.9 Head pose rotation angles

2.4.3.4 Mouth detection

Open or closed mouths are used to measure driver fatigue in some driver's hypo-vigilance detection systems such as yawning. In Computer Vision, there are two methods to detect mouth. First is use feature object detection such as HAAR Casacde trained with mouth image dataset. For more robust, some technique uses HAAR Cascade mouth detection in Rectangle of interest area of face after found face. Second

is to use mouth location from Facial landmark detection. The mouth detection methods can describe in Figure 2.10.

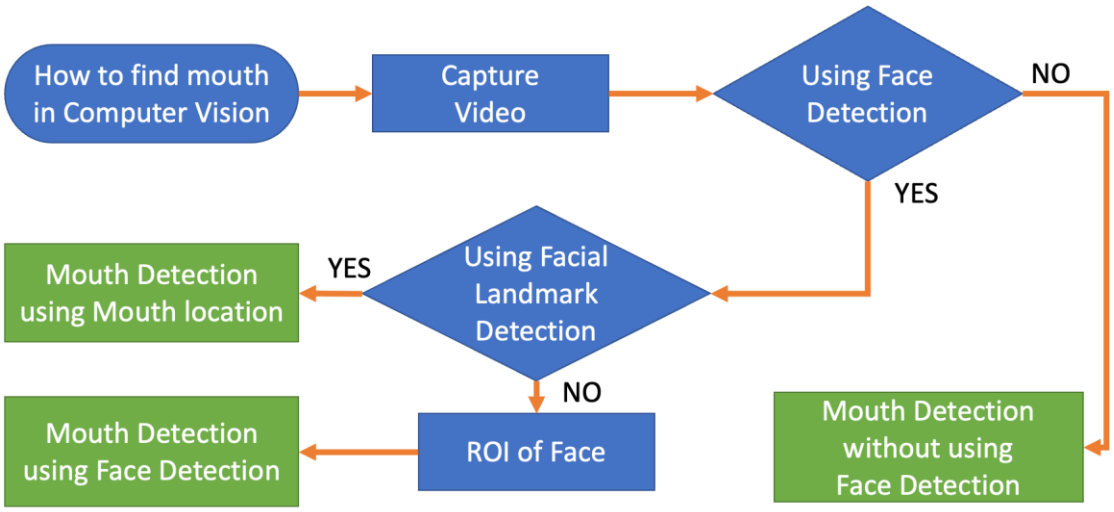


Figure 2.10 Mouth detection methods flowchart

2.4.4 Symptom extraction

In driver face monitoring systems, useful symptoms for fatigue and distraction detection are classified into four broad categories: symptoms related to the eye region, the mouth region, the head, and facial expression. The alertness and fatigue condition relate to Table 2.1.

2.4.5 Decision algorithm

Algorithms of decision-making should be able to detect and take an appropriate decision on driver fatigue and distraction. The shorter length of the decision algorithm can detect driver sleepiness or distraction results in higher system performance. In the meantime, a warning method is extremely important after driver hypo-vigilance detection.

The detection of driver fatigue and distraction based on the extracted symptoms is discussed after symptom extraction. Classification problem is considered as driver state determination. Two classes comprise the simplest form of this classification: consciousness and unconsciousness.

2.5 System design

Although the method for detecting fatigue or distraction is effective for achieving a robust system, current driver face monitoring systems have two major issues: the first is robust and precise detection and tracking of the face and facial components, and the second is robust and precise symptom extraction. The following factors should be considered when developing camera-based fatigue monitoring:

- Develop invariant illumination algorithms for face detection and tracking.
- Develop algorithms to monitor face and face components with skin color and partial occlusion robustness (e.g., glasses, sunglasses)
- Developing robust driver face tracking algorithms in different directions regarding head rotations.
- Fast processing is required to achieve real-time systems.

To develop the driver fatigue monitoring system, two parts that should concern are Hardware part and Algorithm development. The system design should meet the system requirement that describe in Table 2.7.

Table 2.7 The system requirement and system design relation

System Design		
Hardware Design	**System Requirement**	**Algorithm Design**
Camera system with ability to capture in various light condition such as IR camera	Good accuracy Detection	Robust object detection algorithm Need various dataset
High computational processor Sufficient memory and storage	Real-time able processing	Less complexity algorithm
Good Alert system component such as buzzer, LED indicator light	Good Driver Alert System	Accuracy Driver monitoring algorithm Need a lot of dataset
Small system device Can use power supply from in-vehicle socket	Easy to attach on vehicle	

2.5.1 Hardware selection

In this research the Raspberry Pi is the main processor, we need to select the suitable component to make the overall system perfectly operate with usable in real vehicle. The concern points to using in real vehicle is the sizing of the system, we need it so small as we can. The installment positioning, we need it easy to attach and the system can capture the driver in good position. The power supply needs to be easy. In this following we are explain the component of the system.

2.5.1.1 Processor

The Raspberry Pi is very popular for many portable processing projects because of sizing, satisfy performance, and ease of use. In the first of this research, we used Raspberry Pi 3B+ for the process, but we found some problem about processing speed while running image processing. Then Raspberry Pi 4 released with more memory and come with new processor, this is the best performance Raspberry Pi that ever had. So, we describe the specification of two Raspberry model in Table 2.8.

Table 2.8 Raspberry Pi Specification Comparison

	Raspberry Pi 3B+	Raspberry Pi 4B
CPU	Broadcom BCM2837B0 Quad-core Cortex-A53 1.4Ghz	Broadcom BCM2711 Quad-core Cortex-A72 1.5Ghz
RAM	1 GB LPDDR2 SDRAM	4 GB LPDDR4 3200 SDRAM
Wireless	2.4 GHz and 5.0 GHz 802.11b/g/n/ac	2.4 GHz and 5.0 GHz 802.11ac
Bluetooth	4.2 BLE	5.0 BLE
Connectivity	4 x USB 2.0 40-pin GPIO	2 x USB 3.0, 2 x USB 2.0 40-pin GPIO
Display	1 x HDMI (1080p)	2 x Micro HDMI (4k @60FPS)
Power	5V/2.5A DC via micro-USB	5V DC via USB-C (min 3A) 5V DC via GPIO header

In term of performance, the comparison graph in Figure 2.11 and Figure 2.12 explain how fast the Raspberry Pi 4 have compared to the others [24].

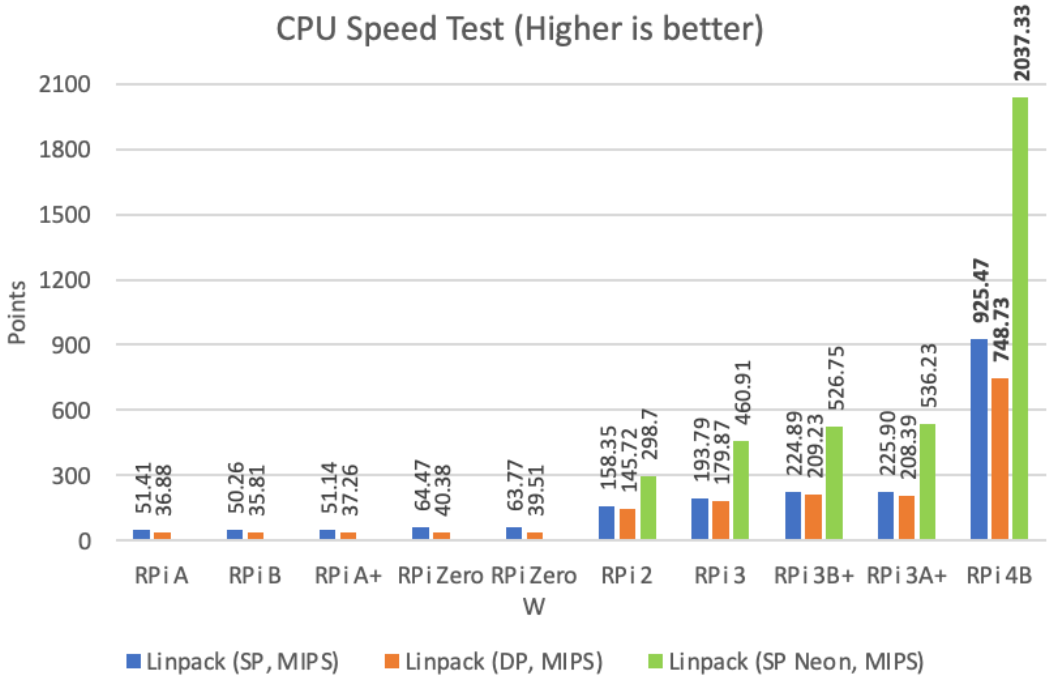


Figure 2.11 The CPU speed test comparison of Raspberry Pi all model

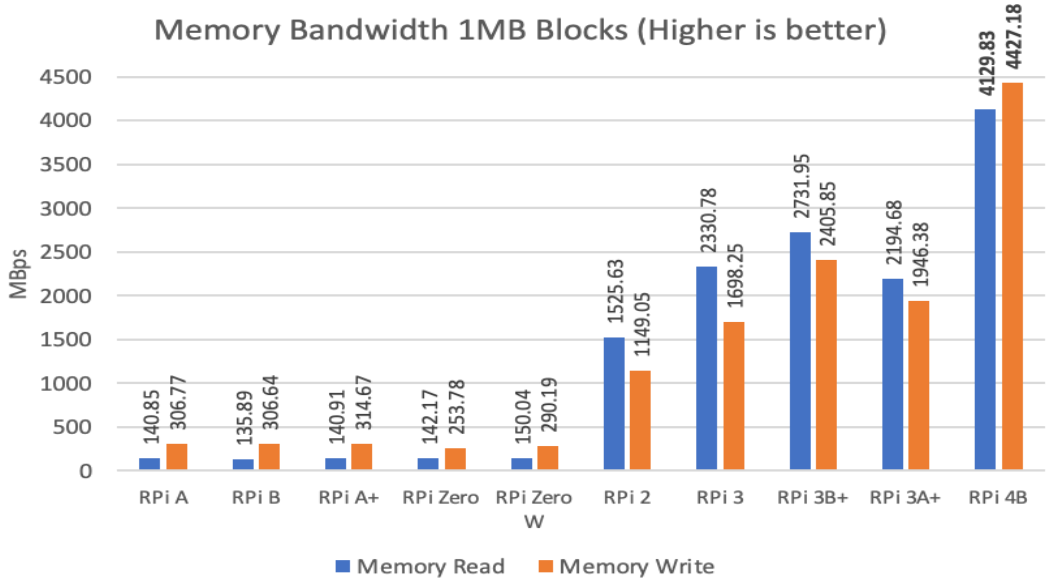


Figure 2.12 The memory bandwidth test comparison of Raspberry Pi all model

2.5.1.2 Camera

The camera module of this system needs to capture the picture in various lighting conditions and work compatible with the Raspberry Pi. Connect camera to Raspberry Pi has two ways, first is connect by camera module cable [25] and the second is connect by USB port [26].

2.5.1.3 Memory card

The memory card plays an important role in Raspberry Pi performance because of the Operating system have installed on it. The high memory bandwidth of memory card can make Raspberry Pi working in good condition.

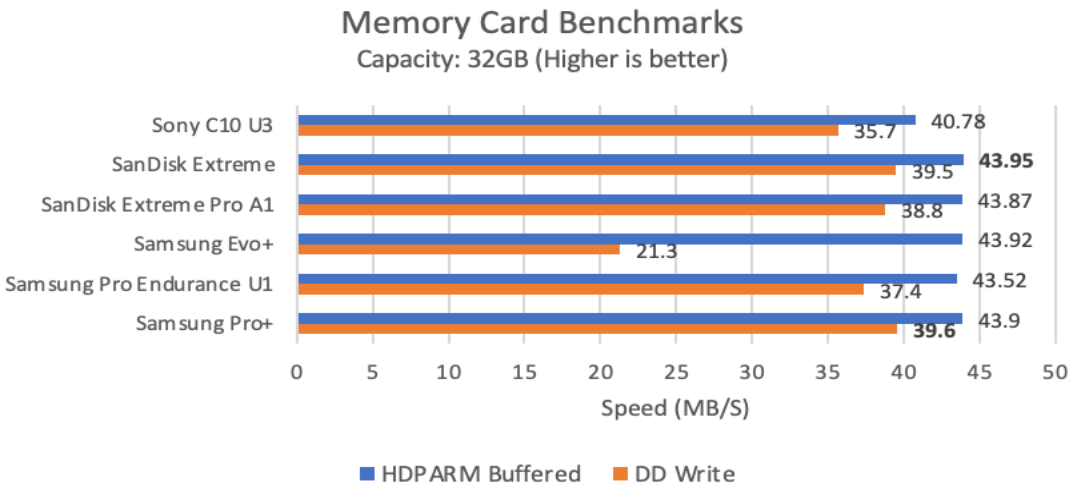


Figure 2.13 Memory card benchmarks comparison on Raspberry Pi 4 model B

From Figure 2.13, two main parameters of this testing are Buffered and Write. First is HDPARM Buffered, displays the speed of reading through the buffer cache to the disk without any prior caching of data [27]. This measurement is an indication of how fast the drive can sustain sequential data reads under Linux, without any filesystem overhead. The second is DD Write, DD is a command-line utility for Unix and Unix-like operating systems where the primary purpose is to copy a file and convert the format of the data during the process. It is used to monitor the writing performance of a disk device on a Linux and Unix-like system. From the Memory Benchmarks the SanDisk Extreme and Samsung Pro+ are the highest performing of this class in both of reading and writing performance. The next figure is the pricing comparison to choose the best worthy in term of performance and pricing.

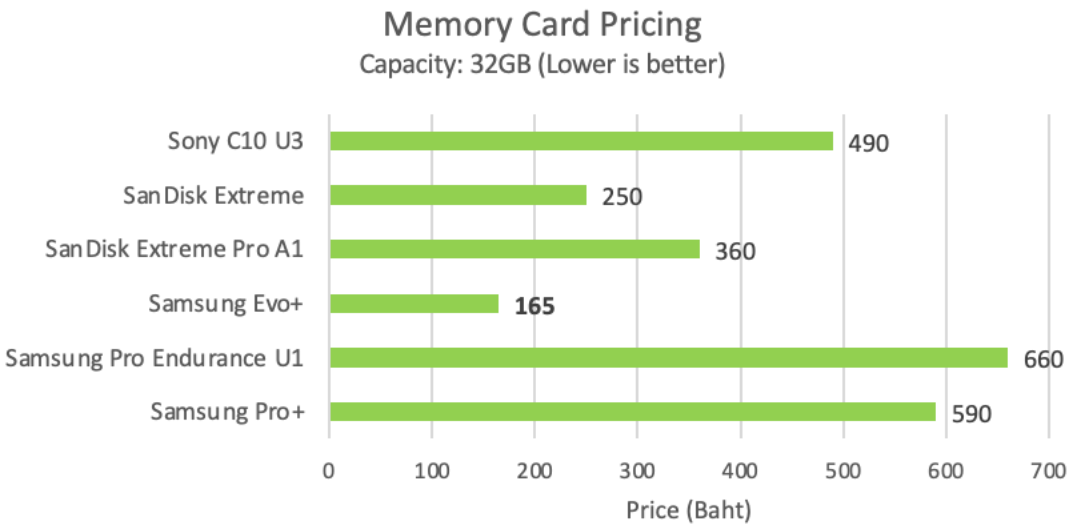


Figure 2.14 The price of memory card group comparison

The cheapest memory card is Samsung Evo+ and the second place is SanDisk Extreme which is the best in performance test. In this case the SanDisk Extreme is worth the money per performance.

2.5.2 Algorithm selection

From the camera-based flowchart in Figure 2.3, the necessary algorithms are face detection, facial feature extraction, and decision algorithm. The algorithm should be accurate and have low complexity to be able to operate in real-time situations. The suitable algorithm will be experiment in the research methodology.

CHAPTER 3

RESEARCH METHODOLOGY

The goal of this research is to develop a device to detect driver fatigue. In which research requires development of both hardware and software algorithms. Hardware development requires usability and installation to be a priority, while being able to support well-developed software together. To develop an algorithm to be accurate, it takes datasets to be tuned and repeatedly tested until the appropriate values are obtained in real life conditions. The data used for testing should be as close as possible to the actual situation. The data we need is Video data in which the driver's face is recorded while driving to view various reactions. The important observation point is that you should be able to see the whole face clearly, see both the eye and mouth because there are important facial features that will be used to develop algorithms for predicting driver symptoms. The following content describes the methods in each topic from collecting data to completing algorithm development.

3.1 Data collection

The information we need most for the development of fatigue detection devices is driving video recordings, especially video recordings of the driver's face clearly.

For data collection, things to consider are the sample group that can really represent the main target audience. The driving situation is very close to real driving. The recording is well recorded, and the target can be clearly seen. The camera mounting position corresponds to the actual device mounting on the car. All this in order to obtain a dataset that can be used to further develop both the algorithm and hardware related to the camera.

There are two sources of data we will use. First, use educational data from trusted sources from The University of Texas at Arlington Real-Life Drowsiness Dataset (UTA-RLDD) [28] dataset. Second is our own data collected according to different situations. The advantage of using the data for education is that we can clearly know

the level of driver fatigue from that file and have more comprehensive information over a wide range of ages, more data with less time to record on our own. While the data collection itself has the advantage that the sample will be more Thai people, which is the main target group for development.

We will combine the advantages of both datasets in the hope that the algorithms developed based on these datasets will enable us to implement them in many situations while maintaining good accuracy.

3.1.1 Participants

For collecting the dataset, we are accepting an unlimited volunteer to participate in the test with the following conditions.

- Age 18 years or older and have a driving license.
- No gender limits.
- Participants are allowed to wear glasses or wear a hat while driving. Sunglasses, masks, or other face coverings may not be worn during the test. (The test will only have one participant with one observer after April 2020)
- If during the test the driver feels uncomfortable or has symptoms of illness. The test can be canceled immediately.

Each of the above conditions is met to meet the required dataset, which is to require a legally driven person. There is no maximum age limit. There are a variety of data such as people who wear glasses or wear a hat while driving. But since we didn't ask the driver to report sleep deprivation before the test, the data of those who are sleepy is less.

In addition, during the data collection period, there was an outbreak of COVID-19 during which the government announced a lockdown. We therefore asked for more volunteers online to record their faces when they were sleepy and send them to Google Drive as another option.

3.1.2 Scenario

For situations where video recording is used, we need an incident that occurs during driving. It is difficult to record the driver while driving long distances due to travel restrictions. Therefore, courtesy of a driving simulator to simulate driving was used in this experiment. The system of driving simulator is described in APPENDIX A.

But if there is no dataset on the real car, it may cause the data to be missing certain events that may occur in real driving, such as the location of the camera on the actual car. Behavior of the driver that acts on the car, such as adjusting the rear window, adjusting the radio, adjusting the air conditioner, etc. There are passengers in the back row that might show up in the frame. sudden changes in lighting conditions, such as driving out of a car park, driving out of the tunnel, etc. Having some datasets recorded on the car should improve the accuracy of driver detection. Courtesy of MTEC's converted electric car test program to record short driver behavior during city test drives.

But in both scenarios, there may not be enough data on people who are sleepy, so participants who voluntarily submitted videos of themselves sleeping drowsy were asked for cooperation. While the government announced the lockdown can't go anywhere accompanying this part of the information as well Therefore, for the situations in which we collect the data ourselves, the following situations are included:

- Real-world driving scenario (as shown in Figure 3.1)
- Simulate driving scenario (as shown in Figure 3.2)
- Self-portrait drowsy situation (as shown in Figure 3.3)

It is expected that the information obtained will be comprehensive for different driving situations. and in various driver statuses from alert to sleepy.

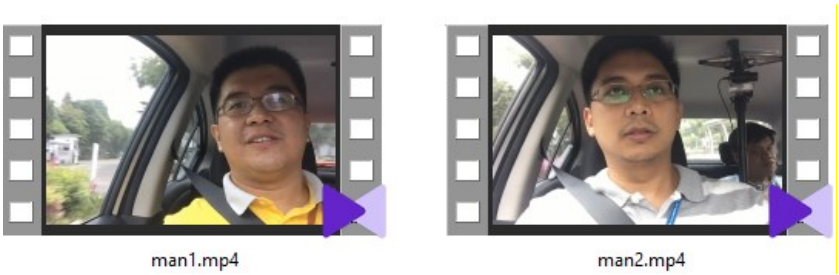


Figure 3.1 Real-driving scenario video recorded dataset

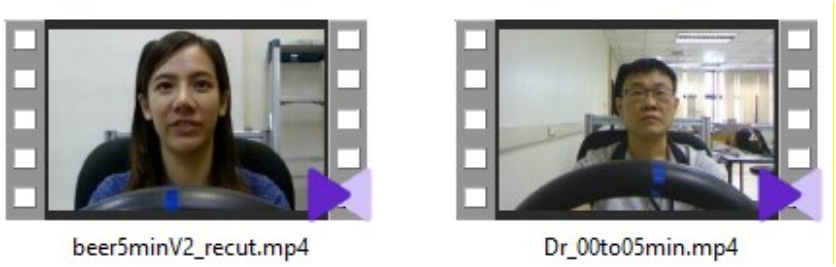


Figure 3.2 Driving simulation video recorded dataset

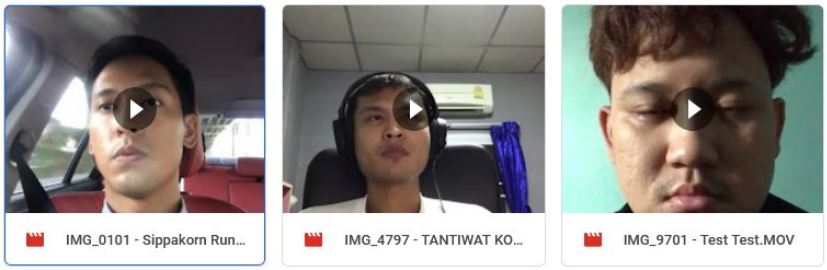


Figure 3.3 Self-portrait drowsy situation video dataset

3.1.3 Route and environmental

For driving route, we set the route on the simulator to find the fatigue of the driver which from reports of road accidents due to dozing. The scene of the accident usually takes place on a long straight road and in light traffic conditions. Therefore, we used City Car Driving, a driving simulation game for an experiment. The game can customize the environment, including lighting conditions, weather conditions, traffic conditions, whether there are pedestrians in the city or not. Adjust road conditions according to driving along the steering wheel side.

The test determined driving conditions based on Right-hand Traffic (RHT). The route consisted mainly of long straight roads on motorways as shown in Figure 3.4. By requiring the participants to drive a loop for 60 minutes or more in the afternoon light conditions, which is another period when the driver's fatigue occurs inferior to

nighttime. It is a time when the body is working hard to digest food and light conditions that make it easier for eyesight to be tired than in the morning or in the evening.

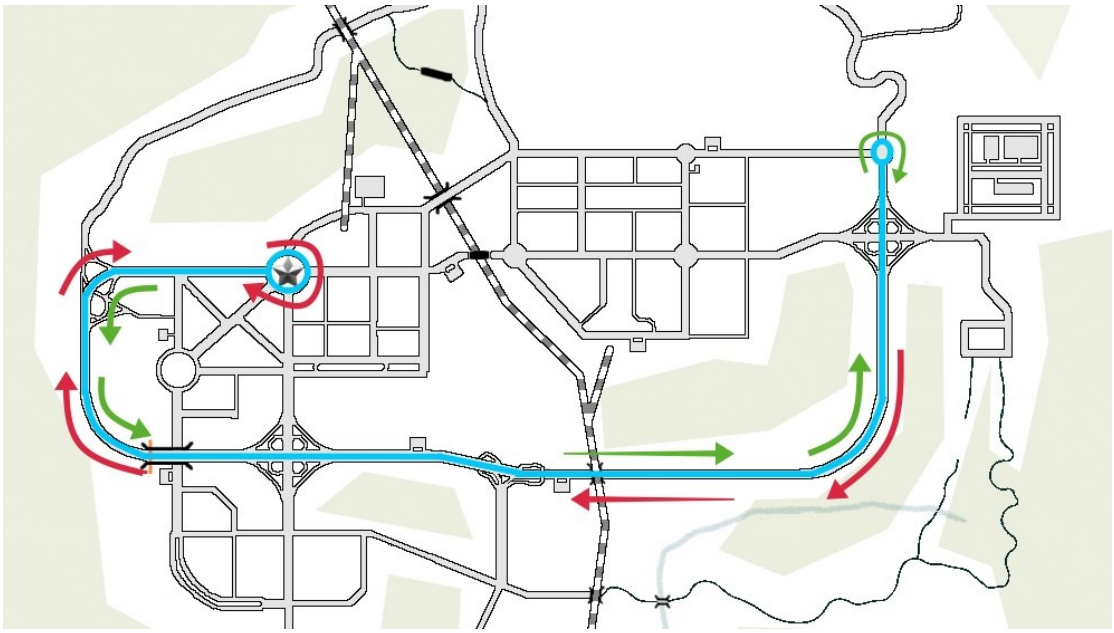


Figure 3.4 Route use in driving simulator

The car used in the simulator is the Toyota Corolla Altis as shown in Figure 3.5, which is the same model as the car used in the real-world driving scenario.

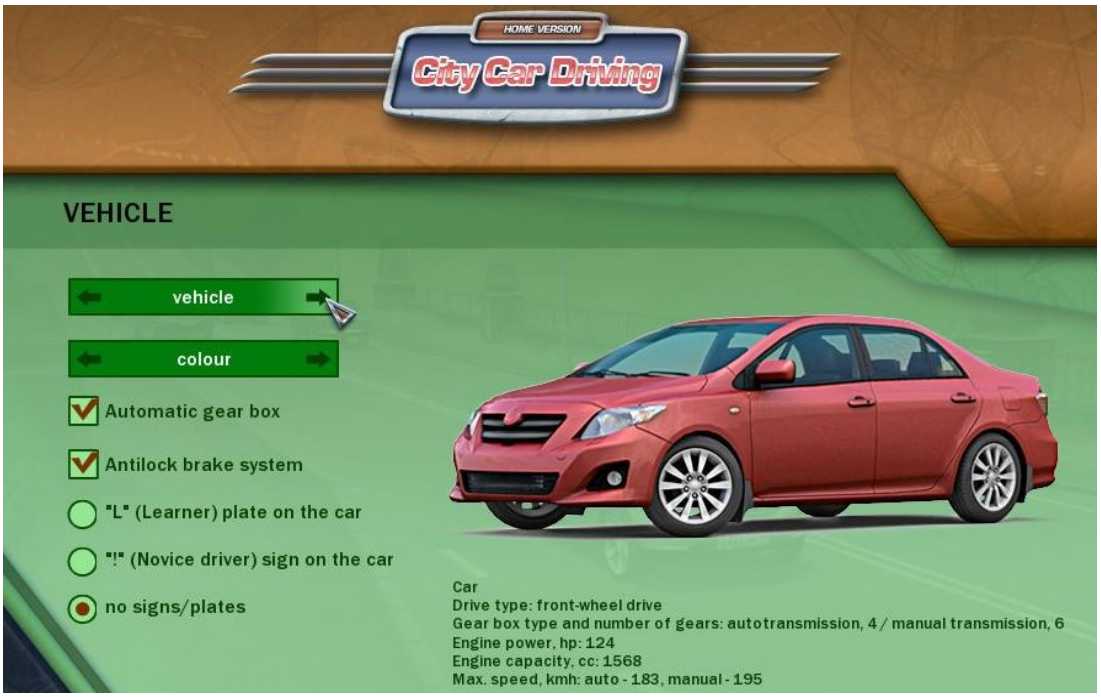


Figure 3.5 Vehicle use in driving simulator

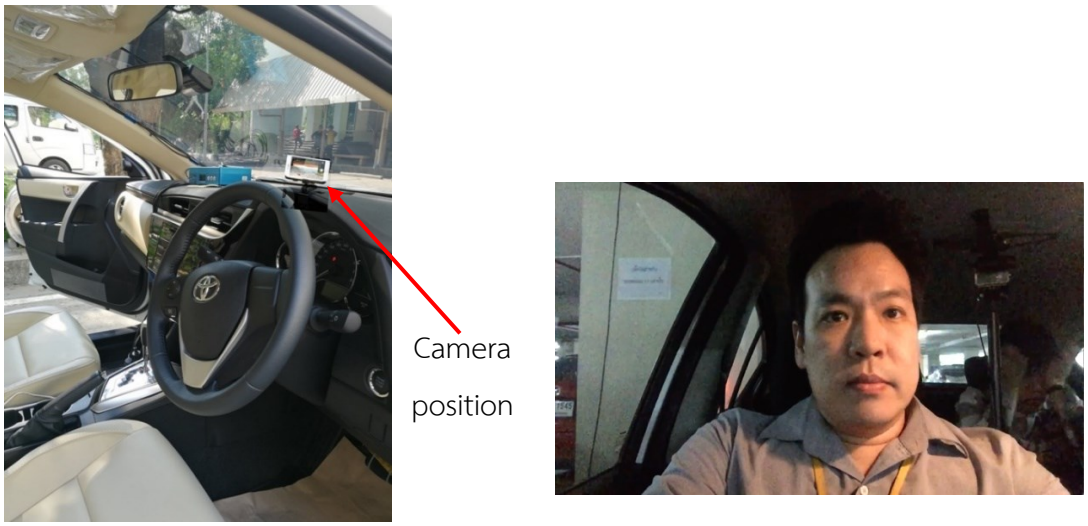
The experimental conditions were set for the participants to comply with as follows:

- Try to drive in the same manner as when driving a real car. Adjust the sitting position to suit the driver.
- Do not use speed while driving more than 120 km/h.
- Drive carefully, try not to hit objects in the game. Drive in lane.

In the experimental room, it is an air-conditioned room that provides comfortable air in a relatively quiet environment. The test period is from 2 p.m. onwards, which is when the body becomes more prone to fatigue. If the participant wanted to turn on the radio, this was allowed.

3.1.4 Data recording

For recording data for real-world driving situations, iPhone 6 is used to record the driver's posture. The installation position is above the dashboard with the camera angle above the steering wheel as shown in Figure 3.6 (a), but not so high that it blocks the driver's visibility.



(a) Camera position (b) Camera perspective

Figure 3.6 Data recording in real-driving condition

For driving simulation condition test from simulator, the webcam will be installed in the same location as on the actual car as shown in Figure 3.7, above the center console and slightly above the steering wheel. During the test, the driver's

fatigue level refers to Karolinska Sleepiness Scale level (KSS level) was questioned every 10 minutes from start driving for study the differential of alertness.

Figure 3.7 Camera position in simulate-driving condition

For situations where the volunteers were recording themselves from home, they were asked to record a video with their faces clearly visible. It doesn't matter if the video is captured vertically or horizontally, but don't let anything obscure any part of the face. When uploading a video, participants will be asked to do a questionnaire to assess their fatigue level and the period when they record the video. The data collection in this research can summary in Table 3.1.

Table 3.1 Dataset collection to use in this research

Data Source	Participants	Condition	Route / Environmental	Data Collect
Self-recorded dataset	Volunteer participants	Real-world driving	Urban area	Portrait Video
		Driving simulation	Light traffic in motorway	Portrait Video / KSS level
		Self-recorded	Living area	Portrait Video / KSS level
Educational dataset	UTA-RLDD [15]	Real-world driving	Urban area	Portrait Video / KSS level
		Self-recorded	Living area	Portrait Video / KSS level

3.2 Hardware developing

For hardware development and design with pre-configured hardware for Raspberry Pi 4 and Pi Camera NoIR V2 which must design an integrated device that can be installed in a car. The camera angle should give an image that comes out comparable to the angle that collects the data. Collecting data on the actual car will allow us to get the right position and distance between the camera and the driver. The things that need to be considered in the hardware design are:

- The device must be easily installed on the vehicle. It can be used with the power supply on the vehicle.
- The installed device can work. Easy to use.
- Video recording without obscuring the driver's vision.

3.3 Algorithm developing

The limitation of mini-computer boards such as the Raspberry Pi 4 is that its processing potential is not high. The design and development of an algorithm should consider two things: accuracy and processing speed. Processing speed affects the system in real time because if it's one second late, it's not present. On the other hand, even if the processing is fast, the system detects too many errors. It may cause the driver to be so annoyed that they do not want to use it anymore. Algorithm development should be balanced between these two things.

Our system focuses on detecting the driver's vital organs, including the driver's face, eyes, and mouth, then converts the values from the images to numerical values for further analysis of the driver's status. In the following section, it describes how the algorithm of a system should be developed before the overall flowchart is summarized in the final section.

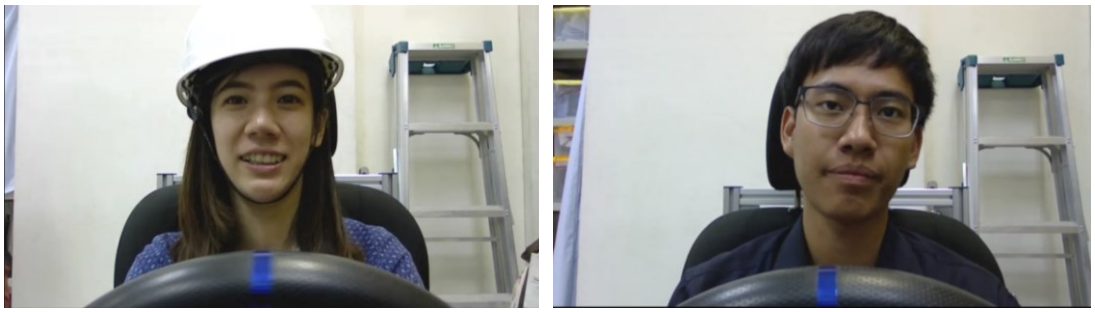
3.3.1 Dataset for object detection algorithms

In this study, we used self-collected datasets and educational datasets from other studies for testing and tuning the detection algorithm. The dataset contains 15 videos with 60 seconds long, 30 FPS. In 15 examples, the situation can be divided as follows.

A) Real-world driving condition without passenger (as shown in Figure 3.8 (A))

B) Long distance driving from driving simulator (as shown in Figure 3.8 (B))

(A) Real-world driving (driver only) condition video dataset



(B) Virtual driving from driving simulator video dataset

Figure 3.8 Video dataset for object detection

This dataset will be used to testing the accuracy of detection algorithm in subject 3.3.3 to 3.3.8.

3.3.2 Dataset for fatigue level (KSS level)

Driver fatigue observations rely on both the video dataset from UTA-RLDD which the KSS fatigue level is provided, and the driver self-assessment data based that describe in Table 3.1. This dataset will be used to study the relationship between the behaviors of driver and KSS level. This relationship can describe in subject 3.3.9.

3.3.3 Face detection testing

Face detection is the first and most important step of driver monitoring. In order to work on the Pi4 as efficiently as possible, choosing a face detector to use is something that must be very important. The detector that will be tested in the system and be compared are:

- HAAR Cascade face detector (The Classical Face Detector)
- DLIB face detector (HOG + Linear SVM)
- MTCNN face detector (Neural Networks based)

The parameters that will be used in this testing to select which detector is more efficient are:

1) Processing speed: measured by the FPS obtained while detection is in progress.
2) Accuracy: accuracy of locating people's faces from videos that can consider 3 variable values; Precious, Recall, and F1-Score.

The testing video dataset obtains the drivers' face in every frame, no True negative case in the detection test. The expectation of good detection is to have more true positive events with a minimum of false positive and false negative. The higher value of F1-score is better accuracy. The FPS obtained while detection is high.

3.3.4 Face tracking testing

Face tracking uses a location recognition technique derived from face detection that means Face Detector doesn't have to be active all the time. After the face detector test from the previous section is complete, the acquired face detector will be used with DLIB face tracking. By measuring the performance with the same parameters as the face detection benchmark, whether using face tracking can improve detection performance better than using only face detection.

3.3.5 Facial feature detection testing

In order to know the driver's status, observing the driver's face and converting it to the status value is an important challenge. Commonly used tools in Python for detecting facial organs include the DLIB facial landmark, or a specially trained detector for detecting that organ that described in subject 2.4.3.

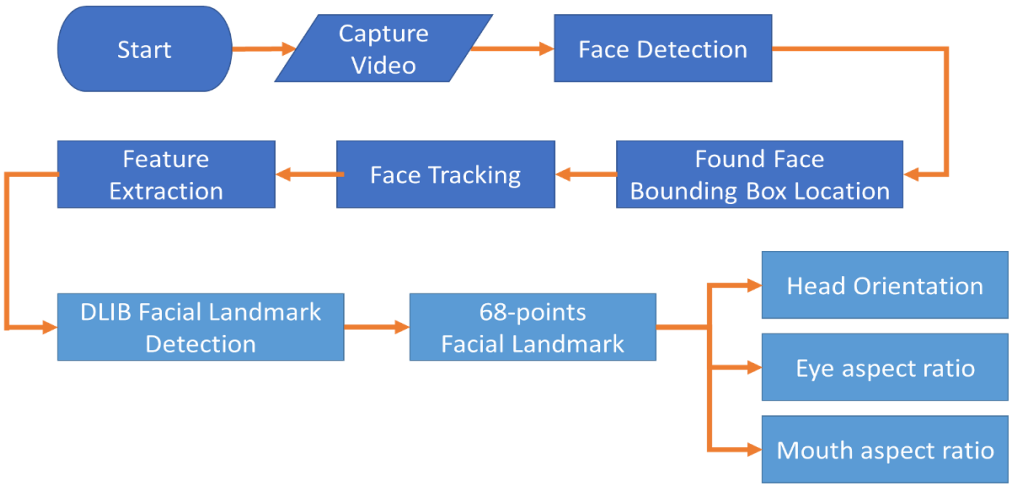
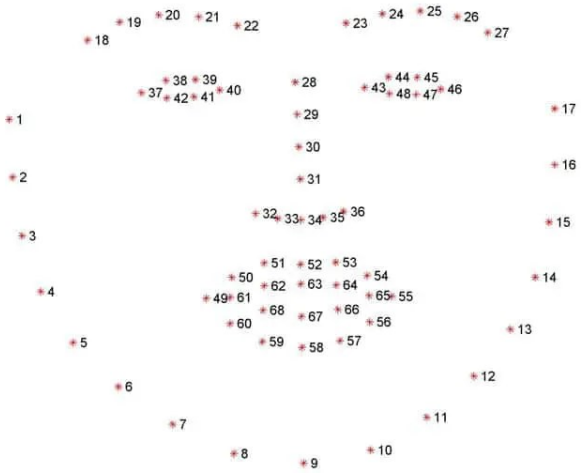


Figure 3.9 Facial feature detection method

This testing measures how much the use of facial feature detector affects processing speed on RPi4. After that, accuracy will be measured.

3.3.5.1 DLIB facial landmark detector accuracy

Accuracy of the DLIB facial landmark detector can be measured by monitoring the position of the 68 points of facial landmarks to determine how accurately the organ position can be plotted. In this study, the plot locations were divided into five parts for easy observation: the chin, eyes, eyebrows, bridge of the nose and mouth, which were important reference locations as shown in Figure 3.10. If any part doesn't match, the percentage of accuracy will be deducted by 20%, and if all positions are correct, it will be 100% accurate.

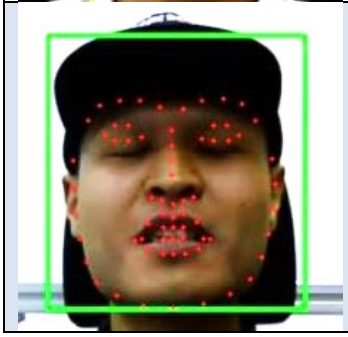


Position	Landmark Points
Eyes	37-42 / 43-48
Eyebrows	18-22 / 23-27
Nose Bridge	28-31
Outer Mouth	49-60
Chin	3-15

Figure 3.10 DLIB facial landmark importance location position

Table 3.2 shows an example of the accuracy measurement of a facial landmark can be demonstrated.

Table 3.2 Example of DLIB facial landmark detector accuracy measurement

Picture	Facial feature landmark position					Facial landmarks accuracy
	Chin	Eyes	Eyebrow	Nose bridge	Mouth	
	OK	OK	OK	OK	OK	100%
	OK	NO	NO	OK	OK	60%

3.3.6 Facial feature extraction

Once the Facial Feature Detector has been tested, the next step is to convert the feature detector gains to the driver status. The topics will be divided into each important facial features as follows.

3.3.6.1 Head orientation testing

Head tilt can be monitored by taking the 68 facial landmarks points then converting them from 2D location on the image to 3D location to obtain the head tilt values for roll, pitch, and yaw respectively. The test is to observe whether the algorithm can output in the right direction and how much computational power is consumed by the framerate change in RPi4. The result picture while process should be like in Figure 3.11.

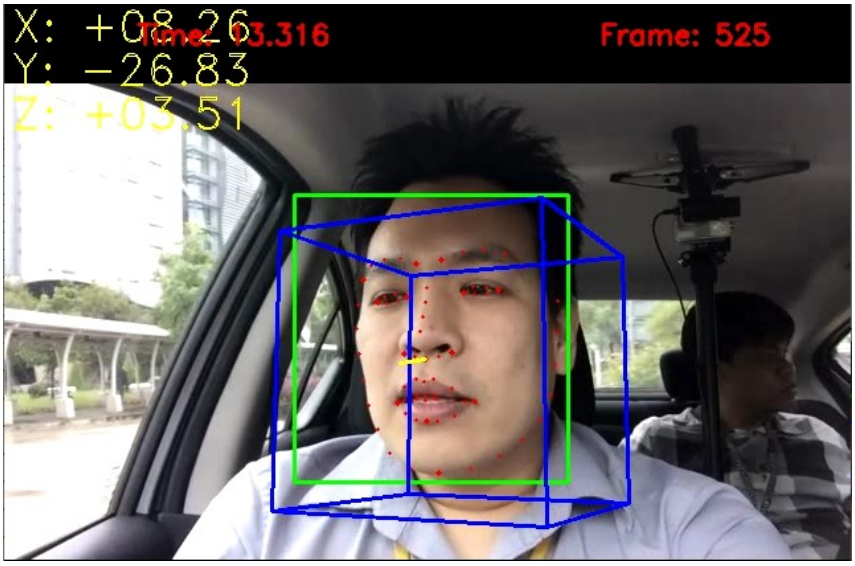


Figure 3.11 Head oreintation testing image result

3.3.6.2 Eye detection testing

As mentioned in section 3.3.5, this section will test eye detector to see whether they can detect and convert the value to the eye state correctly or not compared with ground truth event. Including the processing time from FPS to compare which detector is more suitable for RPi4.

For detecting eye status from DLIB facial landmarks, Eye Aspect Ratio (EAR) values are plotted. EAR values are derived from area calculations with locations from

facial landmarks plotted around both eyes. The plotted area values are then compared with the ground truth event. The EAR value approaches 0 as if the eyes were closed.

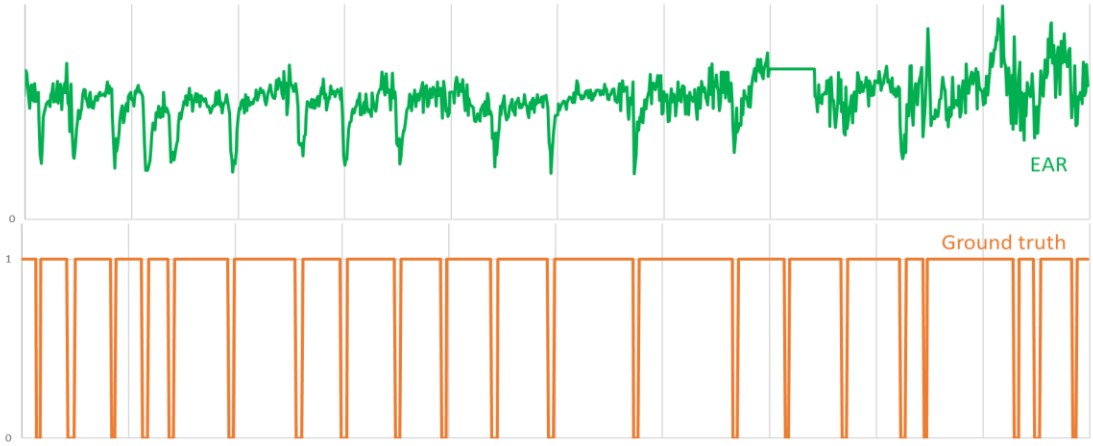


Figure 3.12 Eye detection event accuracy

3.3.6.3 Mouth detection testing

For mouth state detection with the DLIB facial landmark, a similar area detection technique is used for eye detection, which is called the Mouth Aspect Ratio (MAR). By applying the MAR technique to a dataset with various events such as talking, laughing, yawning, etc., This testing is for study that the MAR can be applied well to fatigue detection or not.

3.3.7 Optimization method for better signal clarify

In order to determine the position of the facial features as well as the signals such as EAR, MAR value that will be obtained from the facial detector more clearly. For example, the EAR value is more accurate from a clearer eye image in various environments. This step will try to improve the image quality in various ways to see if it can give a better result or not according to the following topics.

3.3.7.1 Image resolution

The larger the image resolution may bring the more accurate and clear detection. In this experiment, four different video resolutions in 4:3 scale are taken and tested for detection in Table 3.3.

Table 3.3 Video resolution for signal clarify comparison

Video resolution (4:3)	Video resolution standard
640 x 480	VGA
800 x 600	SVGA
1024 x 768	XGA
1280 x 960	

This testing will show how different image resolution produces the signal format of the detector, in terms of signal clarify and processing time.

3.3.7.2 Image equalizer

Another technique that is popularly used to improve image quality is to adjust the lighting of the image. The benefit of adjusting the lighting conditions of the image is to adjust the light in various situations to come out close to the best lighting conditions, such as being able to adjust the image in the dark to see the object better.

For testing the image equalizer methods, the technique came from OpenCV to determine which one is most suitable for the dataset being used. The measurement is based on the accuracy of object detection and the resulting frame rate. The video dataset that will be used in testing are:

1) The original file with resize down to 480p and then apply the enhancement method. The resizing video is similar to the actual algorithm workflow to reduce the processing load.

2) The original files at 720p size have been improved image quality by external programs to make the object clearly visible. The enhanced HD video data is used for better ground truth events.

The EAR value with ground truth of the event is recorded and compared with the signal obtained by each technique of image enhancement, then the satisfied technique in term of accuracy and signal will be summary. The improvement of the image quality technique that was used in the experiment can be divided into sub-topics as follows.

3.3.7.2.1 Histogram equalization technique

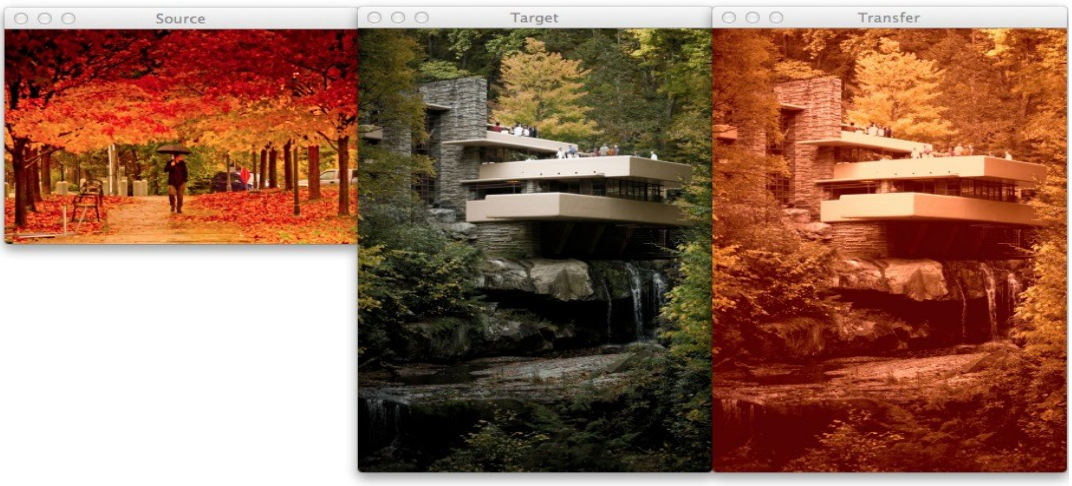
The first technique is image histograms with python-based tools will be investigated as shown in Table 3.4.

Table 3.4 Improving image quality by image histogram equalizer on OpenCV

Method	Parameters	Enhancement condition
cv2.equalHist (source: openCV library)	---	Automatic
Clipping_hist% F (source: stackoverflow)	Clipping hist percent	Trial & error best value
cv2.CLAHE (source: openCV library)	- Clipping limit - Grid size	Trial & error best value
a = 255/img.max() (source: stackoverflow)	Image maximum brightness pixel	Automatic

3.3.7.2.2 Color similarity measure technique

This technique improves the original image to be in the same light as the new image to be used as a reference. Therefore, this technique must ensure that the reference image to be used has the best lighting conditions that are good for detecting the subject well.



(a) the reference image (b) the input image (c) the result image

Figure 3.13 Color similarity measure technique

From Figure 3.13, the image to be used as a reference image for lighting conditions is figure (a), and the image that needs to be adjusted for lighting conditions

is figure (b). After applying this technique, the result will be figure (c) that has been modified to mimic the reference image.

3.3.7.2.3 Gamma correction technique

Gamma in photography is a non-linear operation used to define and decode brightness values in images and videos, that means the gamma value can be adjusted to change the brightness of the image. By adjust the gamma value until found the good lighting condition, Figure 3.14 shows how adjusting the gamma values effects the lighting conditions of the image.

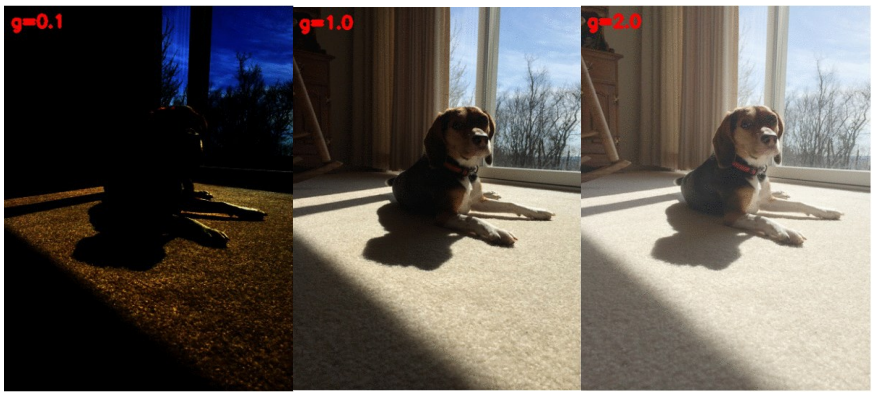


Figure 3.14 Gamma correction technique

3.3.7.3 Crop and zoom technique

The author assumed that larger image resolution would help better object detection, but on the other hand, larger resolution resulted in large file sizes and slow processing. But if using a small image resolution and choosing to enlarge the image only at the location of interest, it is expected that the advantages of both parts will be obtained. The first part is the processing speed as the operation of a small file, and the next is that the signal pattern from the object is as clear as the large image. This technique zoom-in the bounding box area of driver that will be described in the following topics.

3.3.7.3.1 Crop and zoom in object method

The object zoom-in method is enlarging the image after the object has been detected as shown in Figure 3.15. This method ensures that the driver can be detected if the driver is not exactly centered in the frame. For example, bending over to collect things, bending over for radio tuning, etc. While this method may be a bit slower than

the first method because of more step to process as shown in Figure 3.16, but it is expected to provide more accurate.

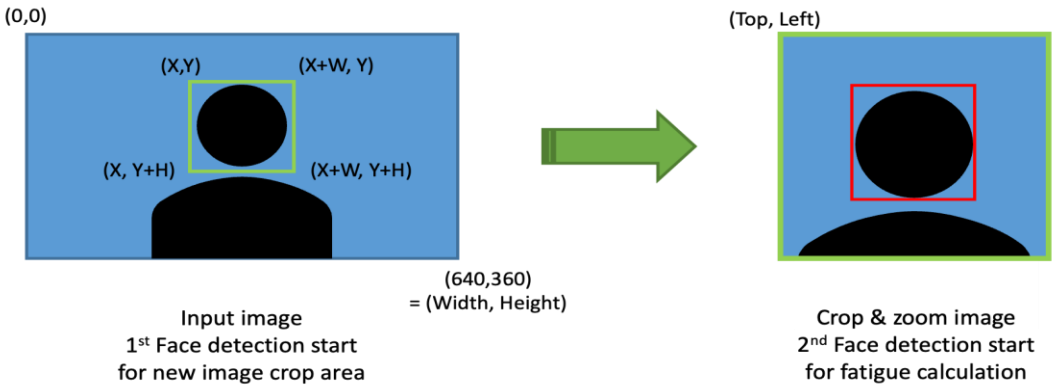


Figure 3.15 Crop and zoom in object method

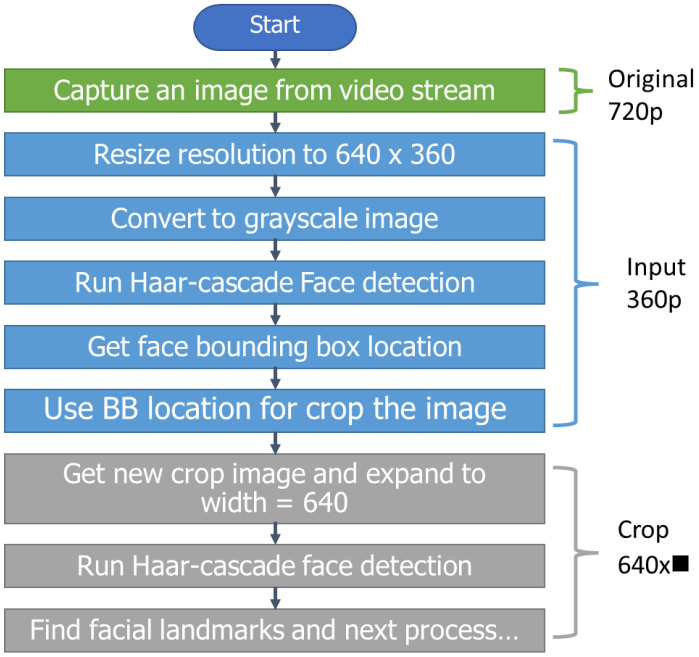


Figure 3.16 Crop and zoom in object flowchart

3.3.8 Symptom extraction

After object detection section, the next step is to convert the acquired signal such as EAR, MAR, or head rotation angles into symptom. The importance of detecting fatigue usually occurs mainly in the eye area followed by the head and mouth area.

One thing to be aware of is the Raspberry Pi itself may not be able to diagnose multiple symptoms at once, although observing multiple symptoms at the same time can analyze the driver status more accurately. Choosing to improve the one important

symptom may be more beneficial for the system. In this research, symptoms of interest to be used in driver fatigue analysis are divided into the following topics.

3.3.8.1 Blinking detection and blink duration

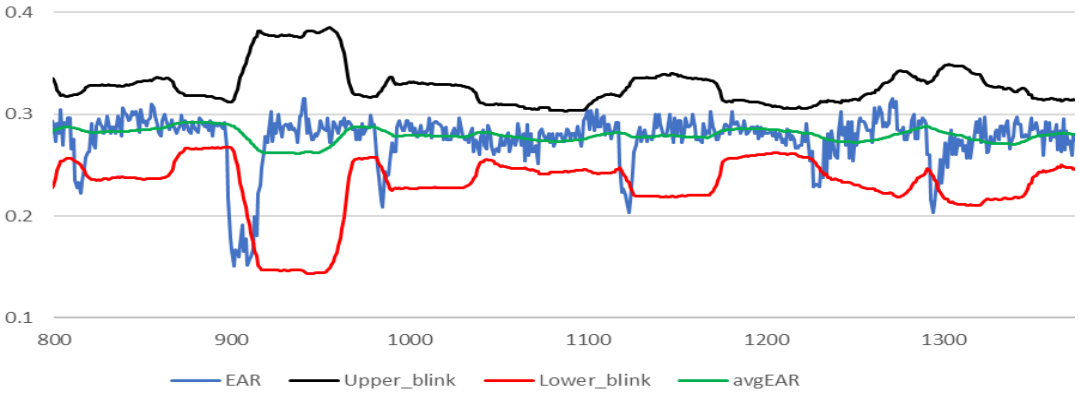
From 2.1.2, says that the symptoms of people who start to fatigue are blinking more often to forcibly waking up, at the same time blinking intervals slowed until the eyes were unable to open. Therefore, blink detection is an important parameter, which well detected can determine the duration of each blink. There are three methods for finding blinking by using EAR values as follows:

Table 3.5 Eye blinking detection based on EAR values methods

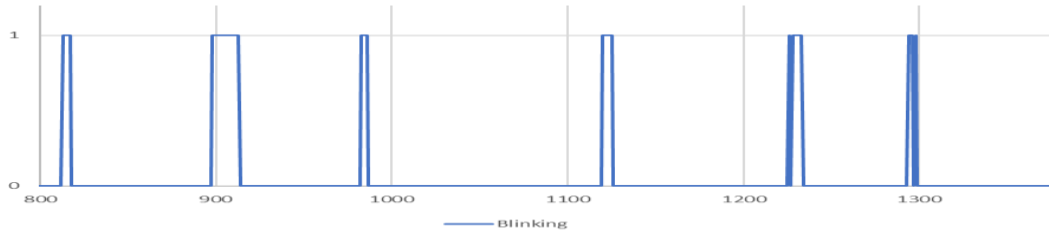
Source	Eye feature extraction	Blink detection	Drowsiness detection
PyimageSearch.com https://pyimagesearch.com/2017/10/23/raspberry-pi-facial-landmarks-drowsiness-detection-with-opencv-and-dlib/	EAR		EAR < 0.3 for 16 frame
PyimageSearch.com https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/	EAR	EAR < 0.3 for 3 frames	
Tereza Soukupova and Jan Cech Real-Time Eye Blink Detection using Facial Landmarks	EAR	EAR < 0.2 for 0.2 second	
		EAR by SVM method	
This research	EAR	EAR low peak detection using Z-score	EAR lowest under Z-score boundaries

From Table 3.5, the method of this research use is also included. Because using values from thresholding alone may be inaccurate. In Tereza and Jan's method [29], it was found that using an EAR < 0.2 for 0.2 second gave a lot of error, so they switched to event training with SVM instead. But since the system of this research is not equipped with additional computational devices such as Google Coral that will do this kind of task better. Other techniques are using the EAR value thresholding and converting it into a blink event.

This research intends to use the Z-Score peak detection to detect blinking event by detect the lower peak of the signal as shown in Figure 3.17, which is a signal indicates eyes closed. Hopefully it will improve blinking detection more accurately than using thresholding alone with less computational power.



(A) Blinking detection using Z-score peak detection from EAR value



(B) Blinking event converted from (A)

Figure 3.17 Eye blinking detection using Z-score peak detection method

3.3.8.2 Yawning detection

Yawning is one of the symptoms that clearly indicate drowsiness. In this study, mouth events were observed using a method similar to the observation of eyes by finding a mouth aspect ratio (MAR). However, mouth tends to have more events than eyes, such as talking, laughing, etc.

This experiment will be investigated by observing the change in the MAR value compared to the time it occurs to be able to determine the yawning event.

3.3.9 Fatigue level and Eye aspect ratio (EAR) relationship

This research focuses mainly on fatigue detection by observing the EAR value. Therefore, in order to make the detection more effective, this section studies the EAR values by level of fatigue to be set the conditions for decision-making of the monitor algorithm.

The video dataset from 3.3.2 will be used for correlation, with each video having a KSS level. The KSS fatigue levels are grouped into 3 groups based on BORS

levels for easier grouping. The parameters used in the study were EAR, blinking rate, and blinking speed.

3.3.10 Driver fatigue monitoring algorithm

After testing the performance of each detection topic as shown as Table 3.6 on the RPi4 and selecting the appropriate one, including finding the relationship between the level of fatigue and the important parameters. The final step in algorithm development is to combine the results of each topic to form the main algorithm as shown as Figure 3.18.

After the main algorithm is obtained, the videos are tested again with both the 3.3.1 videos file and other video files, measuring the accuracy and processing speed to ensure that the main code will still work fine. The parameters to measure are FPS, F1-Score, and ground truth eyes events.

Table 3.6 The algorithm develops section

Algorithms develop section			Tools	Parameter
3.3.3	Face detection		- HAAR - DLIB - MTCNN	- FPS - F1-score
3.3.4	Face tracking		- DLIB tracking	- FPS - F1-score
3.3.5.1	Facial feature landmark		- DLIB facial landmark	- FPS - Landmark location
3.3.6.1	Facial feature extraction	Head orientation	- DLIB facial landmark	- FPS - Roll, pitch, yaw
3.3.6.2		Eye detection	- DLIB facial landmark	- FPS - EAR

			- HAAR eye detector	- Blinking event
3.3.6.3		Mouth detection	- DLIB facial landmark	- MAR - Mouth opening events
3.3.7.1	Optimization for signal clarify	Image resolution	- Resize video command (CV2)	- FPS - EAR signal pattern
3.3.7.2		Image equalizer	- Histogram equalization - Color similarity - Gamma correction - Crop and zoom	- FPS - EAR signal pattern
3.3.8.1	Symptom extraction	Blinking detection	- Thresholding - Z-score peak detection	- Blinking event
3.3.8.2		Yawning detection	- Thresholding	- MAR events

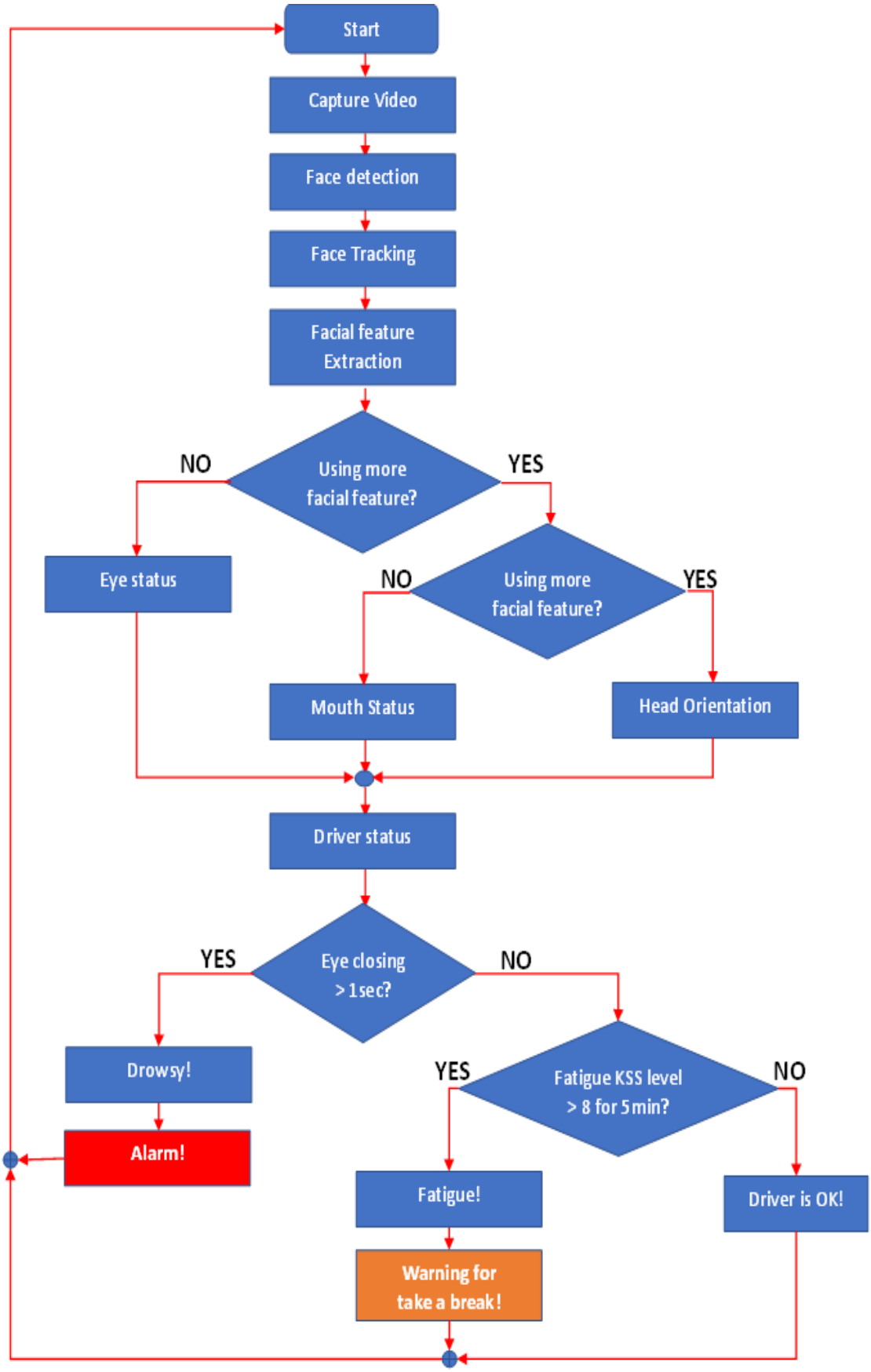


Figure 3.18 This research driver fatigue monitoring algorithm flowchart

CHAPTER 4

RESULTS AND DISCUSSIONS

From the hypothesis for the development of the algorithm in CHAPTER 3 and collecting driver data for studying fatigue behavior. Here are the results, including the problems encountered in the experiments, as well as testing whether the developed algorithm can work effectively.

4.1 Dataset collection

Data collection in this research as previously described, involves three ways of collecting data: from the actual vehicle recorded by iPhone, using a driving simulator recorded by Logitech webcam, and asking to record video of self-portrait sent from home. Fatigue level questionnaires were asked by observers in simulation driving situation and the participants answer in google form from home video data.

4.1.1 Participants

From three sources of video record data, the number of participants is 64 people. Table 4.1 show the number of participants by each condition.

Table 4.1 Join participants in this research

Condition	Men		Women	
	With fatigue level asking	Without questionnaire	With fatigue level asking	Without questionnaire
Real-world driving	-	7	-	5
Virtual driving	8	12	9	9
Self-recorded from home	10	-	4	-
Total	37		27	

4.1.2 Driver behaviors relation with fatigue level

The video data and KSS level questionnaire from recorded will transform to important parameters relate to fatigue level. The interest symptom parameters from videos are blinking rate and blinking duration. These parameters can thresholding the fatigue condition relate to eye value signal.

The following chart describes the symptom parameters related to three fatigue levels in B-ORDS level: alert state (KSS lv.1-5), drowsy state (KSS lv.6-7), and micro-sleep state (KSS lv.8-9). Data recording using various sources to make the data neutral for anybody.

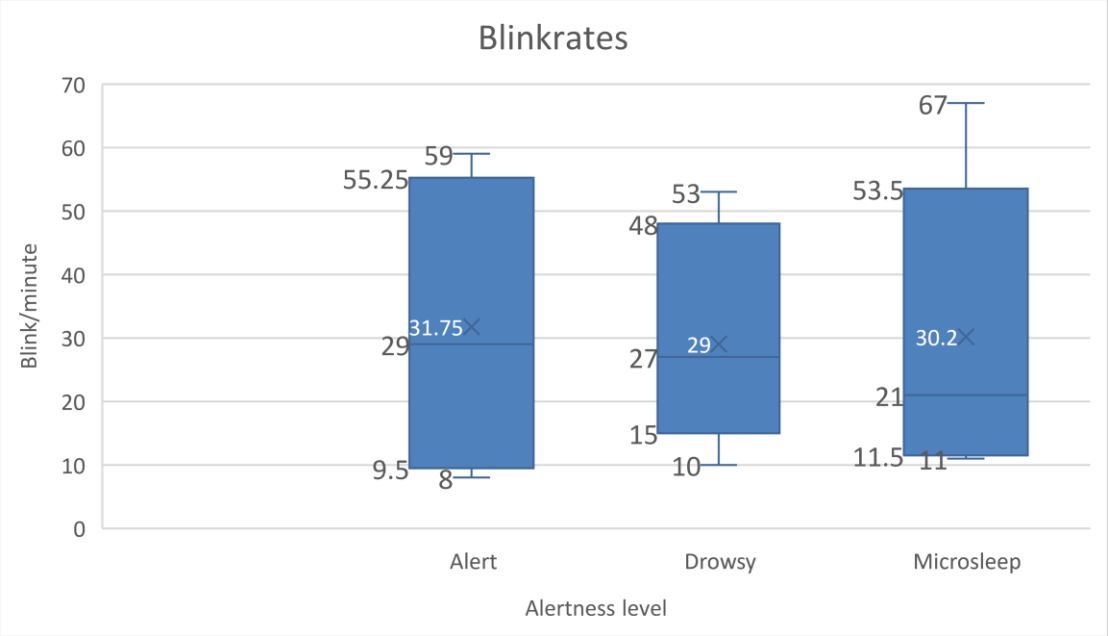


Figure 4.1 Alertness level and blink rate relation

From the Figure 4.1, the blink rate value of the three level of alertness is not much different, the median value is around 30 blinks/min. The Alert state has a much blink rate than Drowsy level because some drivers are more focused and nervous on driving, the eye has response rapid that make blink rate high. Some drivers focus on the road without blinking an eye that makes the blink rate low. The Drowsy state shows more frequent in minimum blink rate, the eye has some fatigue occurred that made driver need to blink more but at the same time less focus the made the blink rate not much higher. The Micro-sleep state shows the blink rate value looks similar to alert

state but in different reason. The eyes have much fatigue that need to sleep but the driver force to awake that made the eye has more blink than drowsy state. Therefore, this parameter cannot show the clearly different value to threshold the driver state. Next parameter is blink duration chart related with alertness level if this parameter is appropriate.

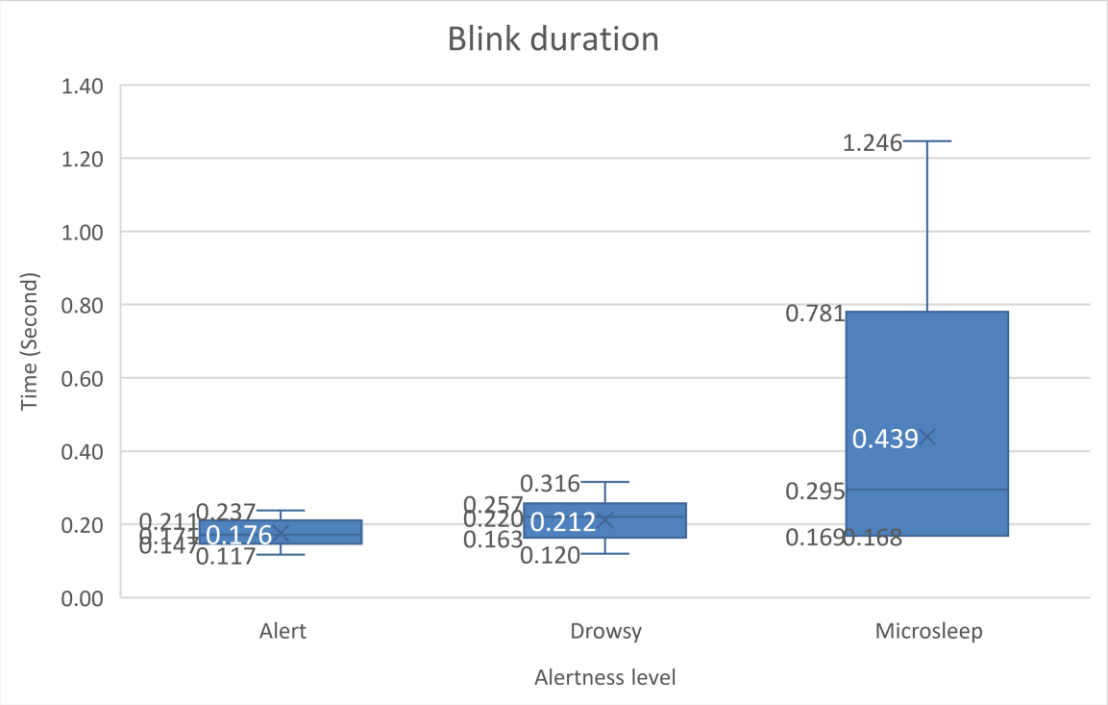


Figure 4.2 Alertness level and blink duration relation

From the Figure 4.2, we can see the clearly different in blink duration time. The blink duration depends on alertness level the short blinking means alertness, the more blinking time mean more sleepiness similar to information from Table 2.1. Therefore, to indicate the driver status the blink duration time more than 0.24 second per blink can determine the driver is in drowsy state and the blink duration more than 0.4 second per blink can determine the driver has a high risk to enter the micro-sleep state. The blink duration value can be used in drowsiness algorithm to indicate the driver status.

From virtual driving dataset who answered the fatigue level in Table 4.1, we can find the relation between the fatigue level and time. The following chart will show the relation between KSS level and time of driving.

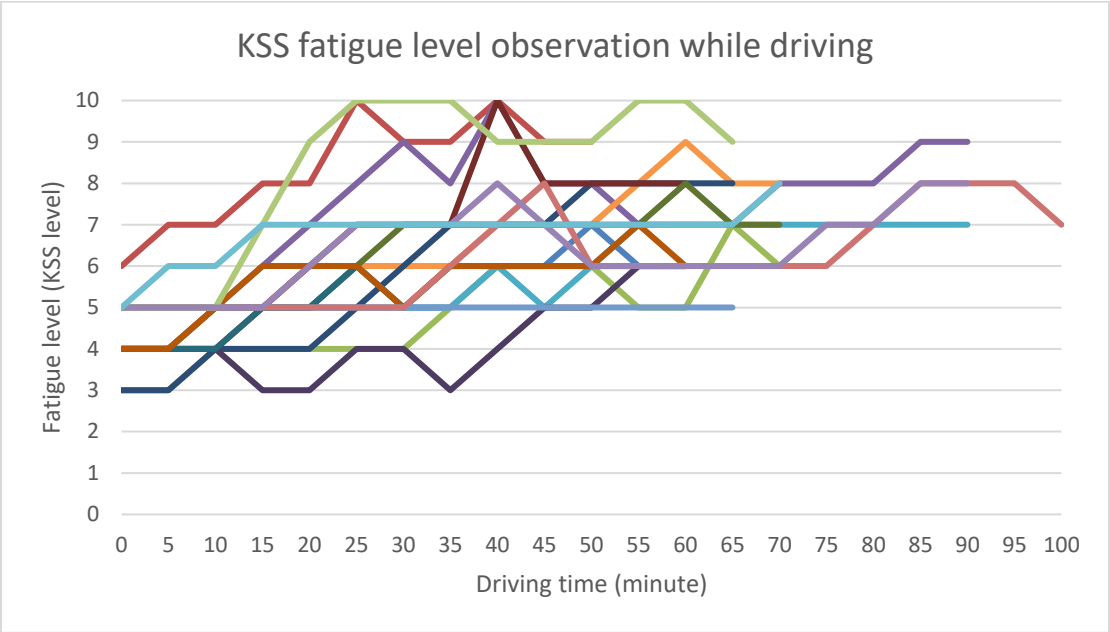


Figure 4.3 KSS level observation from virtual driving dataset

From Figure 4.3, the fatigue level tends to increase depend on longer time. The majority of datasets have level changes after 10 minutes pass and the driver state start from alert and ends in drowsy state. Some drivers can stay in alert state until the end of driving. Some drivers have fall asleep while driving (KSS lv = 10) after 25 minutes pass, then after becoming conscious again they try to keep awake, but the level cannot decrease to drowsy state just stay in micro-sleep state. Therefore, in boredom conditions such as this virtual driving condition the fatigue level tend to increase after 10 minutes pass and if the driver fall into the micro-sleep state, they cannot comeback to more refresh state if they not have a break.

In this study we choose to focus on how to detect the driver in danger state as micro-sleep state with the clue we got from the dataset: have longer blink duration with some insane eye blink rate (over 60 blinks per minute).

4.1.3 Data collection problem

For problems encountered in collecting datasets, some cause will describe here to prevent the future work or other work from the disturbed thing including:

1) Camera angle and camera distance

In data collecting, there is a problem with the camera tilting for one person, making the dataset that gets the camera angle not quite like a real driving. In collecting data from home, often the data is recorded vertically, which is not the same as the camera angle used in the experiment.

2) Gestures

In collecting data, the behavior of the driver, especially the real driving caused the recording of the driver's face to come off the frame somewhat. Collecting data from the simulator was found that some people slouch while driving, causing some faces to be obscured by the steering wheel, which may affect the accuracy of face detector.

3) Various of fatigue datasets

As the majority of the data collected on volunteers were not hired or paid, most of the fatigue levels collected tended to be similar. For collecting data with high levels of fatigue due to an unforced request that the subjects had to go to sleep before the driving test began. This makes the data for people who are sleepy quite a few. Some important data are not recorded such as age, driving experience, etc.

However, the experiment using the afternoon time after lunch showed that some people fell asleep during the test, which was hypothetical. Including collecting data on sleepy people from home and using datasets for education is enough to help solve this problem of diversity.

4.2 Detection algorithm

In algorithm development, the first step for driver monitoring is the driver face detector. Accuracy is necessary for any detection algorithm, but the processing speed is important for Raspberry Pi too. The following are the components of object detection algorithm in concern with accuracy and processing speed.

In this research, we have tested the part of algorithm on MacBook Pro with same environment as Pi. If the result works well on MacBook Pro, then the part of algorithm will apply on Pi and display the result on Pi, if the result on MacBook is not working well, we only show result on MacBook and no need to test on Pi.

4.2.1 Dataset for object detection

The dataset for object detection record in 3 situation and can divide into 4 categories as explained in 3.1.1. While testing, the driver's movement and light condition affect the accuracy of object detection, for more effectively we divide the video categories by driver movement and light condition are ease to compare to driving situation.

Table 4.2 Dataset for object detection in term of driver movement

Driver movement and light condition		Driving situation similarity
Stable movement **Frontal looking** **Bright light condition**	**4**	Driving on straight road in daytime
Stable movement **Frontal looking** **Dim light condition**	**4**	Driving on straight road in low light place
Frequent movement **Look around** **Bright light condition**	**4**	Driving on urban road in daytime
Frequent movement **Look around** **Dim light condition**	**3**	Driving on urban road in low light place
Total	**15**	

4.2.2 Face detection

For testing the processing speed can be measured by the average number of framerates obtained. If the detector can process faster, the value should be closer to the original file. The detector with the highest framerates is considered to use less resources and could process in near real-time. The following figure shows the framerates that three detectors process compared to each other.

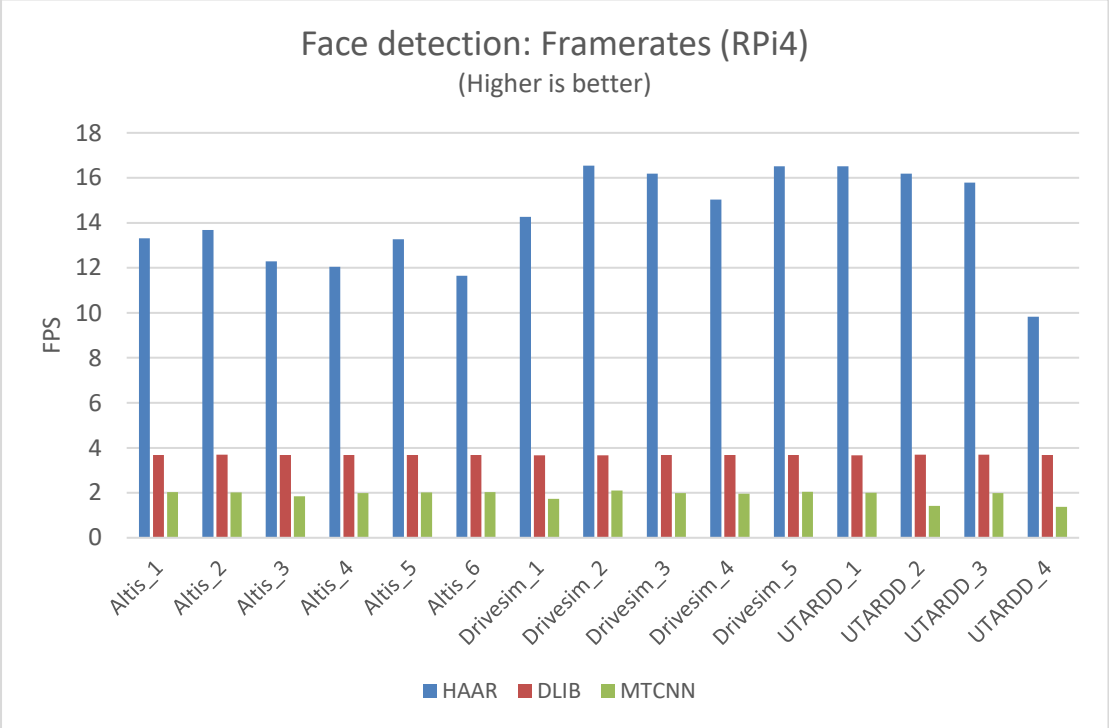


Figure 4.4 Face detection framerates comparison process on RPi4

From the framerates result in Figure 4.4, it was found that HAAR Cascade gave the highest framerate around 14.2 FPS. It is 4 times than the DLIB face detector around 3.68 FPS. MTCNN, a Neural Network detector, gave the lowest frame rate around 1.90 FPS due to high power in computation.

On accuracy topic, since the driver's face is present in every frame of video data, there are no true negative (TN) cases in this test. The expectation is that the true positive (TP) events must be maximized in the test while the false positive (FP) and false negative (FN) events are kept to a minimum. The following are the results of the experiments of each detector to see how well faces are detected.

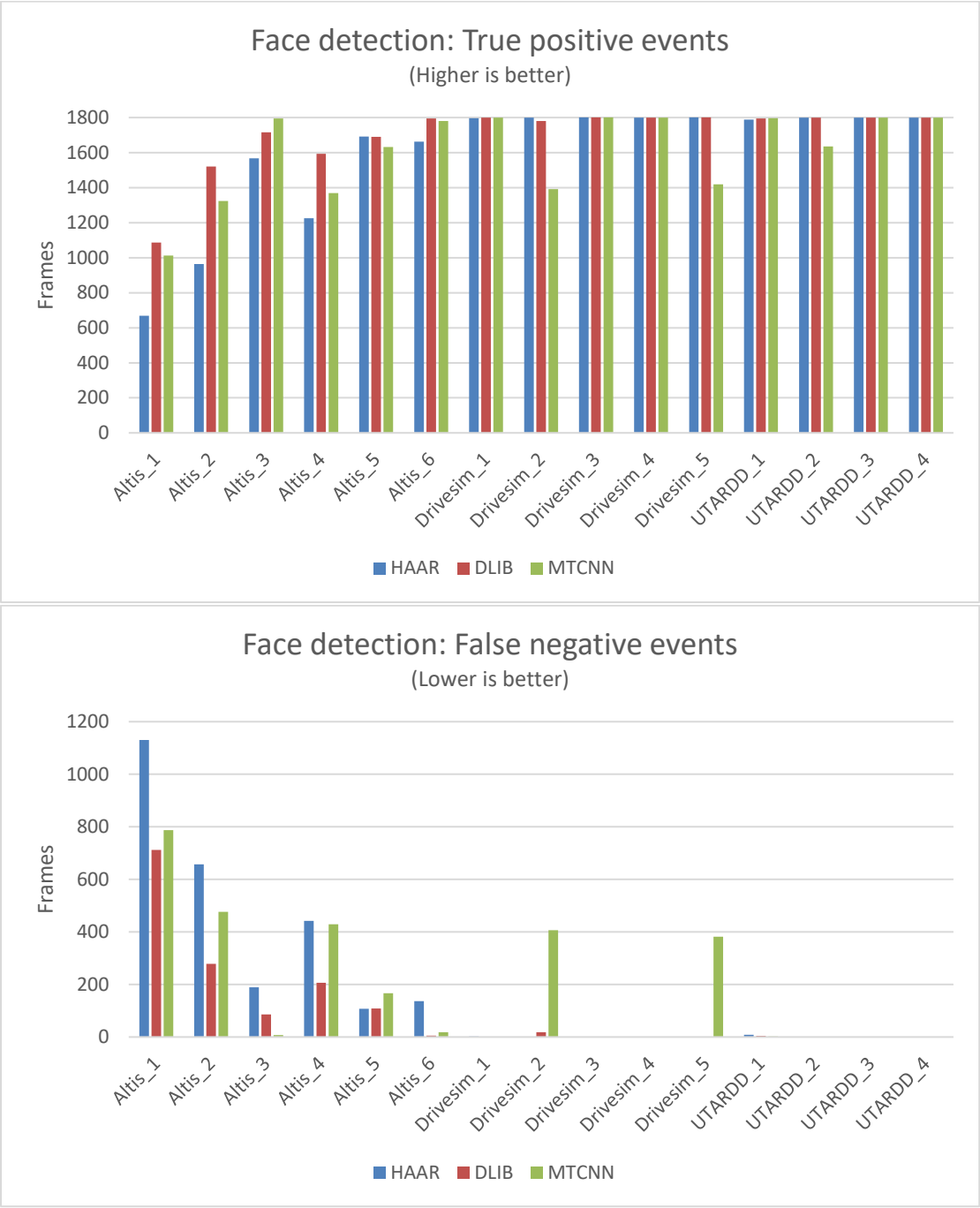


Figure 4.5 Face detection object detection comparison

From Figure 4.5, it was found that in most scenarios DLIB face detector delivered the most frames with TP events and the least FN events compared to the other two detectors in the same video, but with the other two detectors can do quite well. Next stage is to find the precision, recall and F1-score values to show the overall accuracy of the detector clearly.

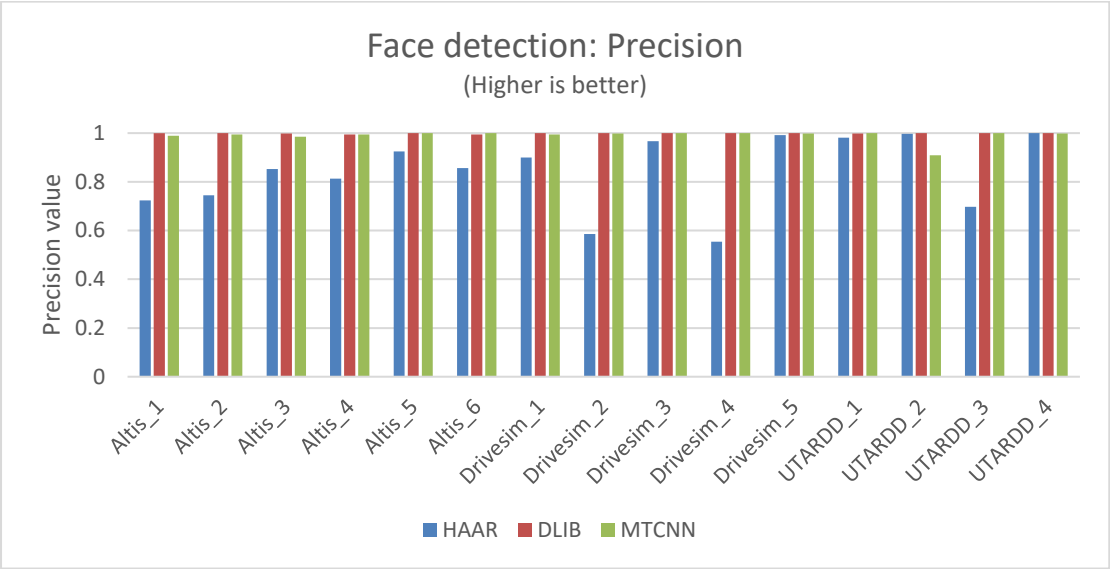


Figure 4.6 Face detection precision value comparison

Figure 4.6 show that both DLIB and MTCNN provide high precision value in most situations, but HAAR performs well in some bright light and frontal looking situations. HAAR Cascade's precision value is low because it not only detects many true positive events but also many false positive events as well.

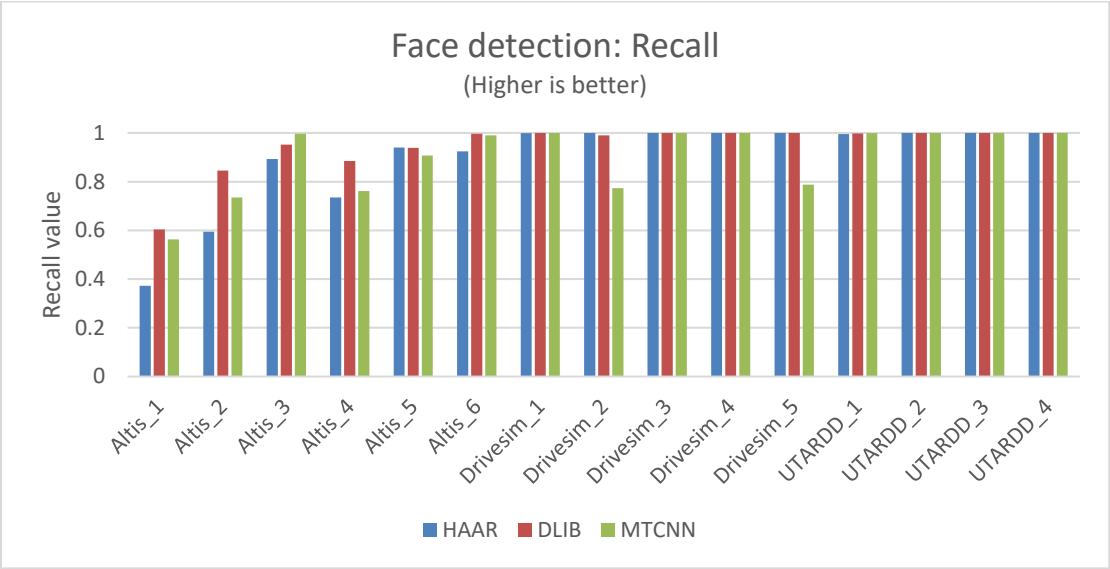


Figure 4.7 Face detection recall value comparison

In testing, the highest recall values came from DLIB, which performed best in all situations. MTCNN came in second, and HAAR came in last place. DLIB and MTCNN can detect the face in various angles, that is the reason of less false negative event and considered a weak point of HAAR because to detect in very tilt angle.

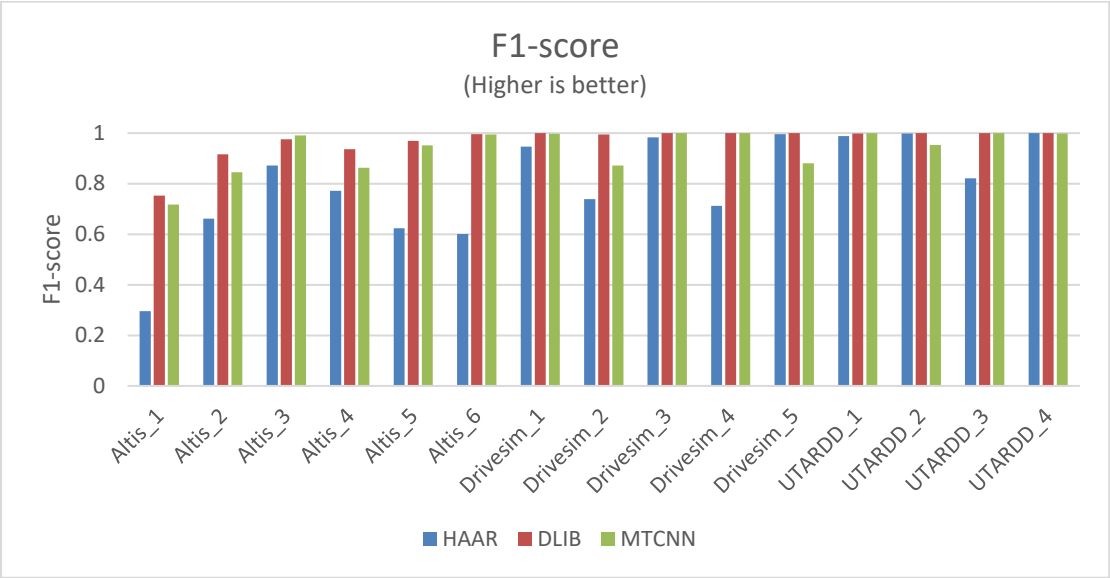


Figure 4.8 Face detection F1-score comparison

Based on the results of the F1-score, which overall measures which detector performs the most accurate. DLIB is the best-performing detector covering a wide range of tilting angles. Second is MTCNN, which performs best various angle facing but still misses some frontal face detection, and the last place is HAAR Cascade which detects frontal faces quite well, but slant face detection is poor. The following table shows the average value of all parameters from 15 videos dataset.

Table 4.3 The comparison of face detector performance in all conditions

Subject	Face Detector		
	HAAR Cascade	DLIB	MTCNN
Framerates (FPS)	14.21	3.68	1.90
True Positive events	1597	1704	1610
False Negative events	178	94	178
False Positive events (Ghosting)	317	1	15
False Positive events (Face overlap)	45	1	1
Precision	0.839	0.999	0.990
Recall	0.897	0.947	0.901
F1-score	0.800	0.969	0.937

From Table 4.3, it clear that in terms of processing speed, none of them can compete with HAAR Cascade. In terms of accuracy, both DLIB and MTCNN are very accurate. But HAAR wasn't bad either, being able to give more than 70% accuracy. The following table will describe the accuracy value depending on the situation refer from Table 4.2 to better understand the face detector characteristic.

Table 4.4 Face detection accuracy comparison in each condition

Precision			Recall			F1-score		
Stable movement / Frontal looking / Bright light condition *4 videos - (Driving on straight road in daytime)*								
HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN
0.760	1.000	0.998	1.000	0.997	0.943	0.849	0.999	0.967
Stable movement / Frontal looking / Dim light condition *4 videos - (Driving on straight road in low light place)*								
HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN
0.913	1.000	0.977	1.000	1.000	0.947	0.950	1.000	0.958
Frequent movement / Look around / Bright light condition *4 videos - (Driving on urban road in daytime)*								
HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN
0.876	0.997	0.995	0.887	0.958	0.937	0.808	0.976	0.962
Frequent movement / Look around / Dim light condition *3 videos - (Driving on urban road in low light place)*								
HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN	HAAR	DLIB	MTCNN
0.798	1.000	0.995	0.636	0.796	0.735	0.527	0.879	0.838

From Table 4.4, the HAAR can detect well in frontal looking in stable movement condition even in dim light. If the object has frequent movement and looks around, HAAR can work well in bright light than dim light condition.

DLIB can perform well in any situation with highest accuracy but also has a chance to miss some face detection if the object has frequent movement and looks around in dim place. MTCNN look like DLIB with less accuracy around 3.2%.

We can summary that if the object looks frontal and has less movement, the detector can do it well even in dim light. If the object has more movement and looks around, the brightness is a necessary factor of accuracy. In this research, we are looking for an algorithm to detect drowsy driving. The majority of drowsy accidents occur on long straights where the driver has to look straight ahead more consistent with the first two scenarios from Table 4.4, which every detector can work well in this situation. In this section, we will choose two detectors to develop in the next section: HAAR with the fastest processing speed and DLIB with the highest accuracy.

4.2.3 Face tracking

Face tracking is a popular technique used to help better detect the position of a face by remembering the position of the face in subsequent frames after the detector has detected in the first frame. Eliminating the need to search for a new face position in every frame, which reduces the processing load and helps in the accuracy of bringing information on the target face to be processed further because it can be confident that it is the same person's face in case of more people entering the frame.

The following flowchart presents the working principle of face tracking from the beginning to finish. There are two important parameters in the process: face tracking status and tracking quality. The first one remembering the position of the face. If there is no value, then the face position is not remembered, the face search must be performed first. The second parameter is the value that tells whether the face is still in its original position. If the driver moves a lot from its original position or there is something obscuring the face at that moment, the value will be as low as near zero.

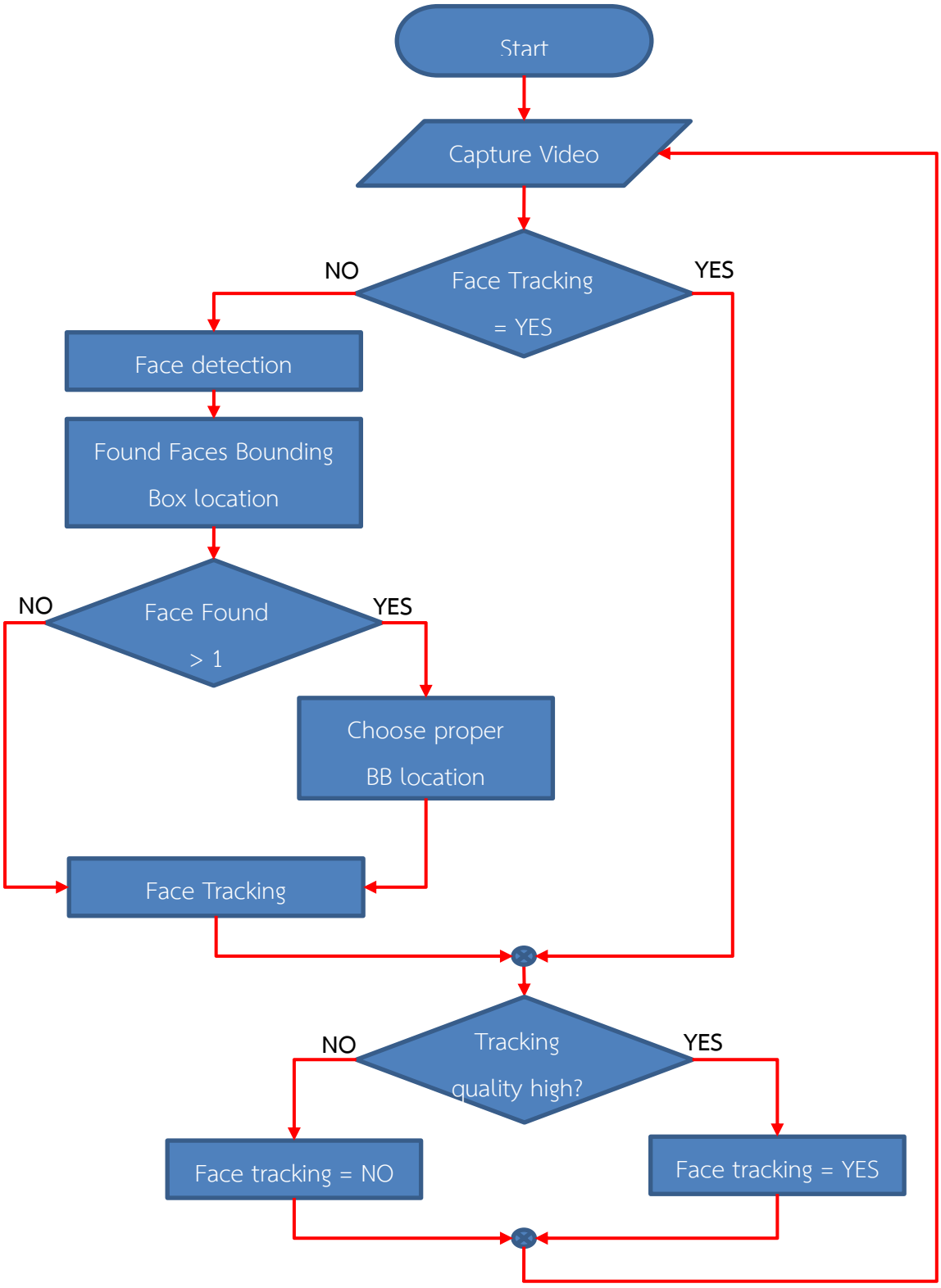
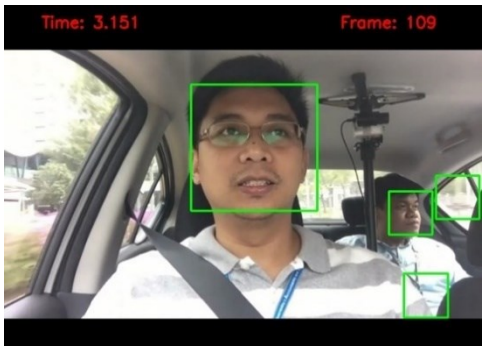


Figure 4.9 Face Tracking Flowchart

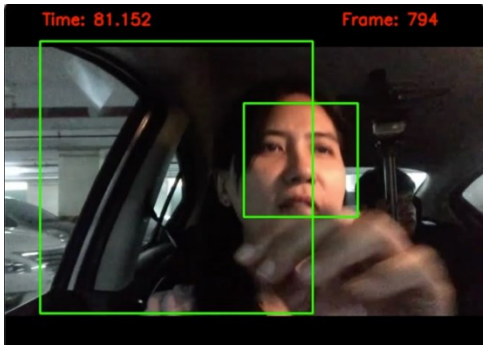
From Figure 4.9, in the event that more than one bounding box (BB) of face position is found such as Figure 4.10(A), selecting the position to be recognized is an

important step. Even if there is only one face position, how can ensure that it is the correct face position such as Figure 4.10(B).

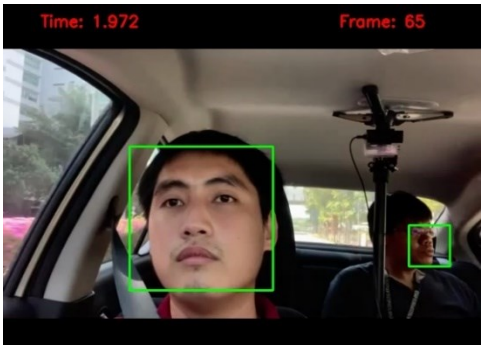


(A) Multiple bounding boxes

(B) Wrong location bounding box



(C) Face overlapping BB



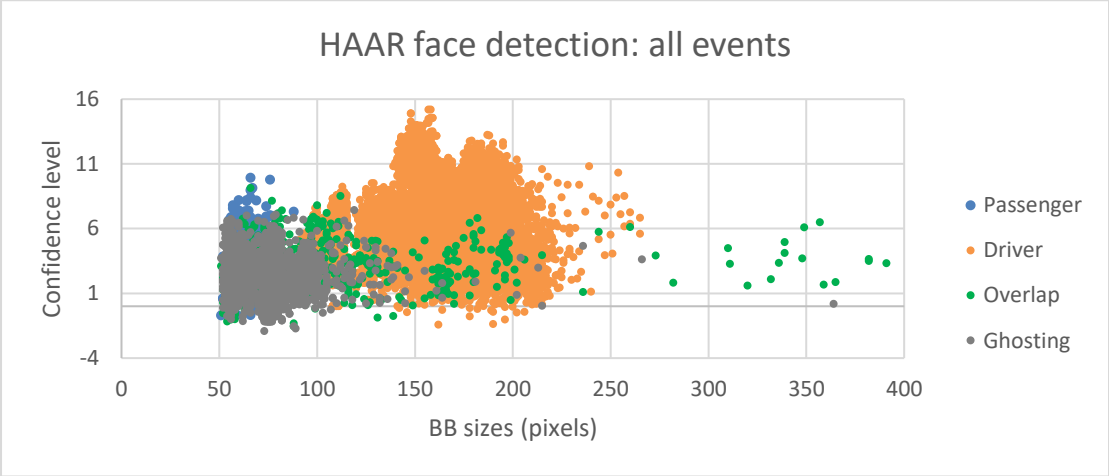
(D) Correct position

Figure 4.10 The Bounding box location of detected object

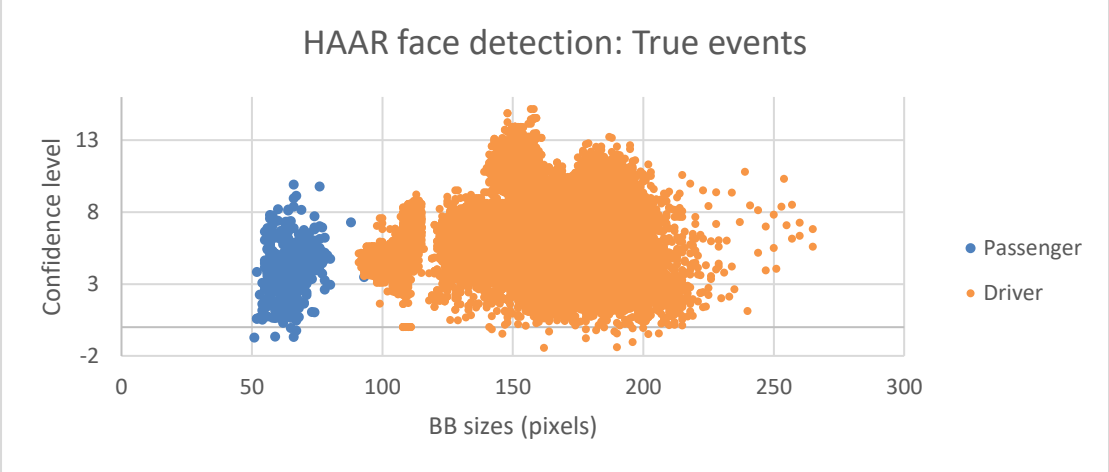
Human face tracking is generally a method of choosing where to recognize face location from the largest bounding box size in the case of multiple bounding box locations such as Figure 4.10(D). However, the biggest bounding box is not assured that are the needed object such as Figure 4.10(C).

The chance to choose the correct position, the necessary parameters of each face detector should be studied to define the characteristic of each detected event. The HAAR and DLIB have the same parameters in every BB, box size and confident level. About the BB sizes, both detectors have the same kind of data: width and height of BB in pixels. The BB shape of HAAR and DLIB is square shape. The confidence level of BB is float value. HAAR confident level value can be a negative number while DLIB is only positive number. If we can find the relation of BB size and confident level of

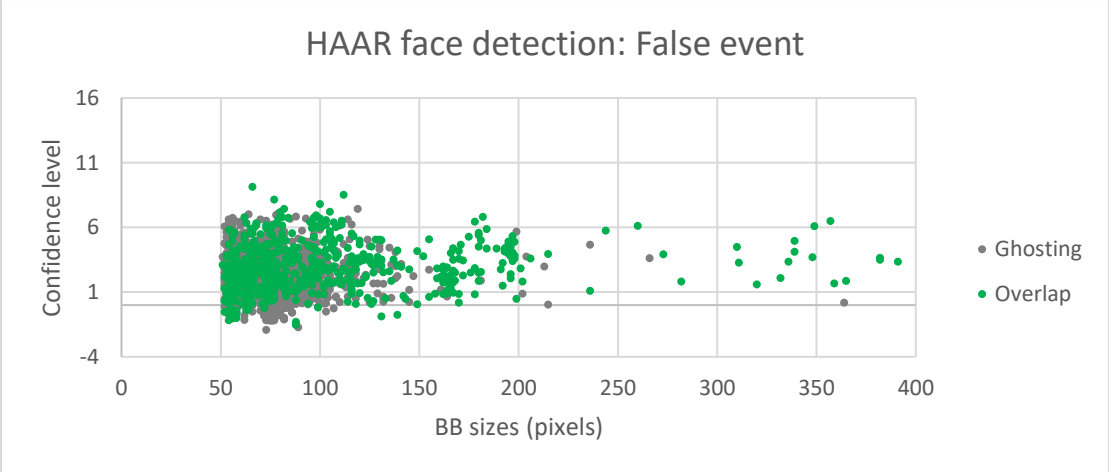
the true positive and false detection events, we can separate the value to make face tracking more accurate position.



(A) All events of bounding box in scatter chart



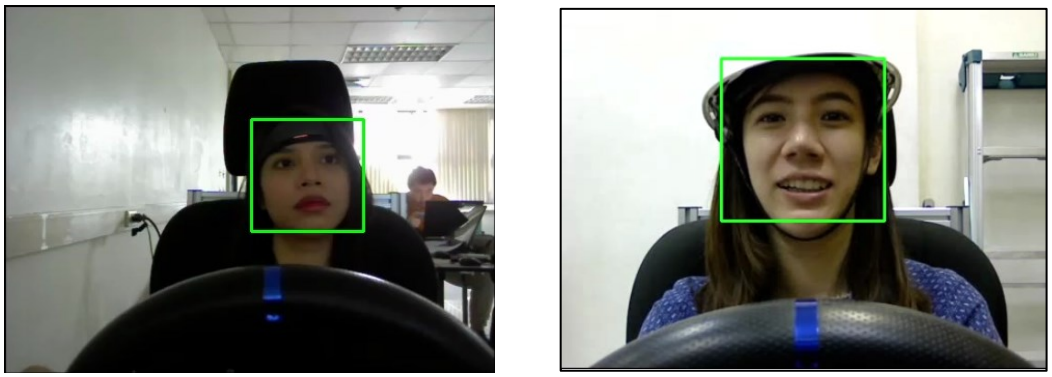
(B) True events between driver and passenger in scatter chart



(C) False events in scatter chart

Figure 4.11 The HAAR face detection bounding box event comparison

From Figure 4.11, the detection events between true and false events can divide by bounding box size. The false events have mainly BB size between 50 to 100 pixels and the confidence level in positive value as shown in Figure 4.11(C). The true events detection can clearly classify the driver and passenger by BB size that can see two group of BB size as shown in Figure 4.11(B). From the driver events, we can see two small group of events because of two videos of dataset have far distance of camera record setup around 90 centimeters while the others have camera distance around 40 to 70 centimeters that similar to the distance in real-driving recording distance.

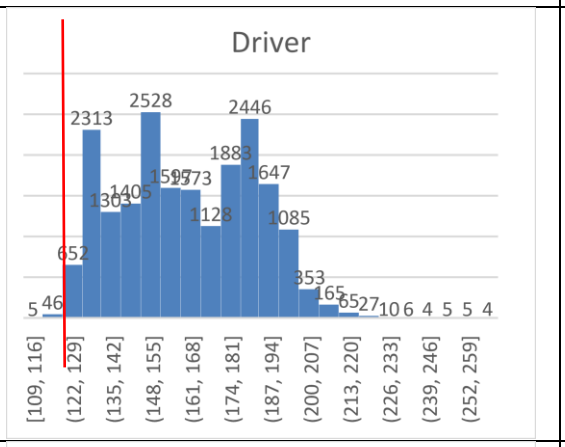
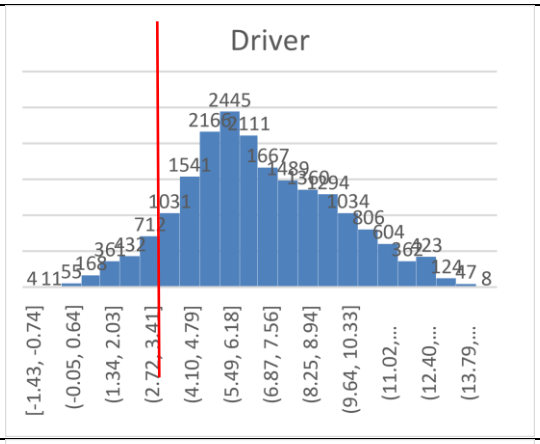
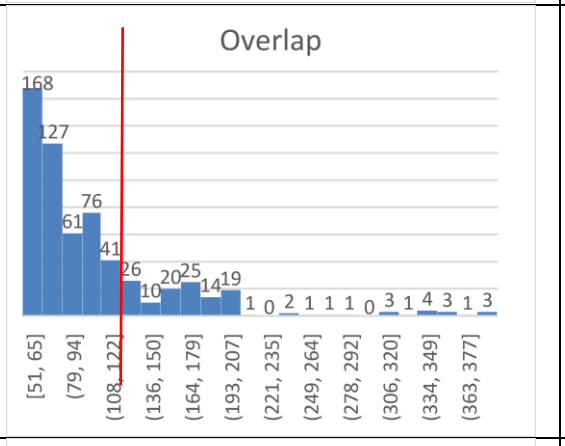
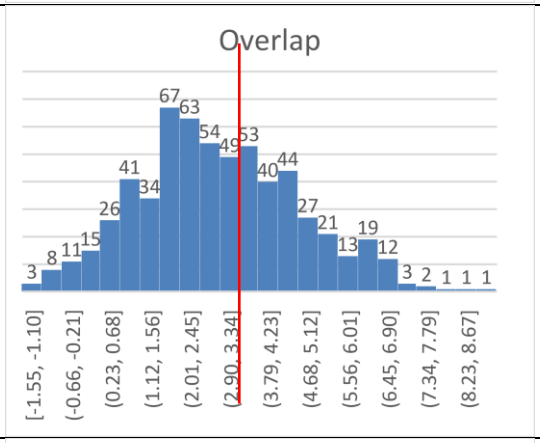
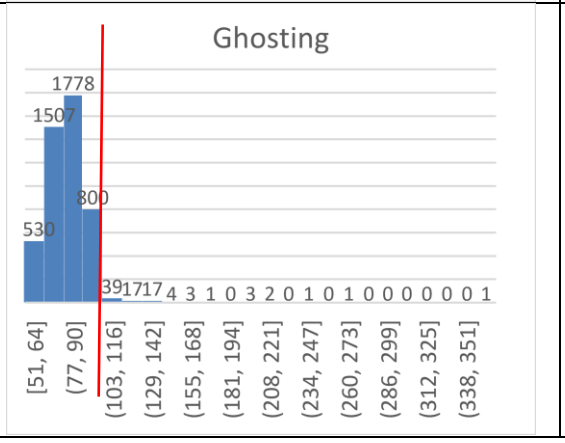
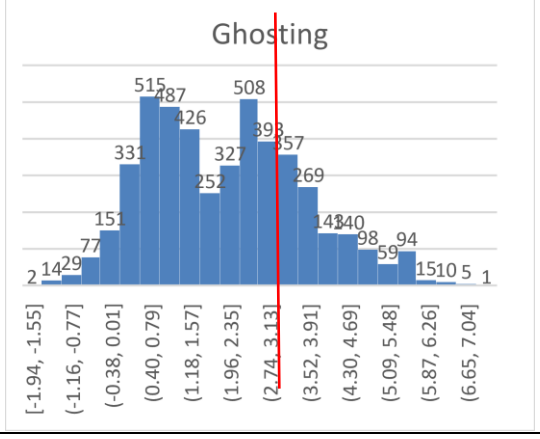


(A) Far distance camera video (B) Normal distance camera video

Figure 4.12 Camera distance between two group of driver events

For make the algorithm suitable for real-driving conditions, we need to cut off some far distance camera dataset. Then using new data to analysis which is the best value to detect the driver. Normally, much research uses linear separability to classify the necessary data from all data, but from this research we need the quadrant cut because this has both parameters between BB sizes and confidence level. For more driver event detection, the following table shows the frequency of events in both parameters.

Table 4.5 The HAAR bounding box of all events in histogram chart

Bounding box sizing	Confidence level
	
	
	

From Table 4.5, we found the Overlap and Ghosting events have characteristic in right-skewed distribution of BB sizing by smaller than driver event that can easily classify. Therefore, threshold value of BB sizing starts from 109 pixel that can classify the amount of event data are driver event 100%, passenger event 0.35%, overlap event 42.76%, and ghosting event 18.89%.

From confidence level, every event has characteristics in normal distribution except ghosting event that is double-peaked distribution. For less false events detection, in this research choose thresholding confidence level more than 2.74 that just after the second peak of ghosting event. Therefore, threshold value of confidence level starts from 2.74 that can classify the amount of event data are driver event 94.21%, passenger event 69.68%, overlapping event 46.71%, and ghosting event 29.78%.

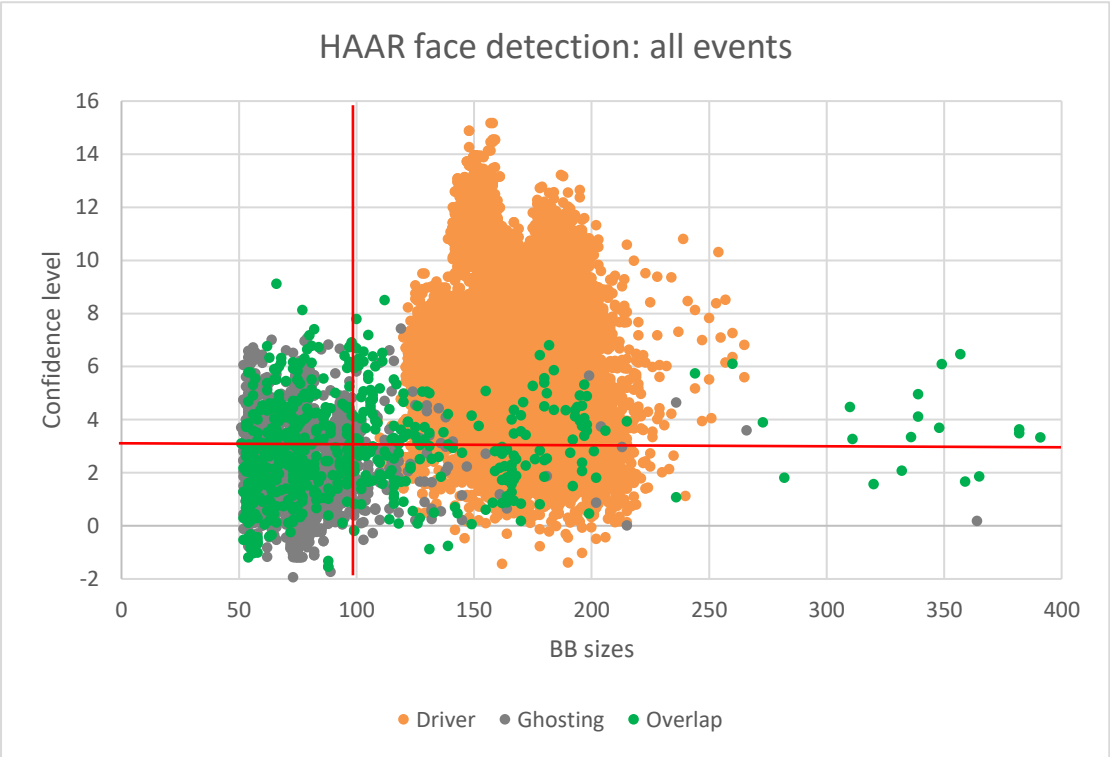
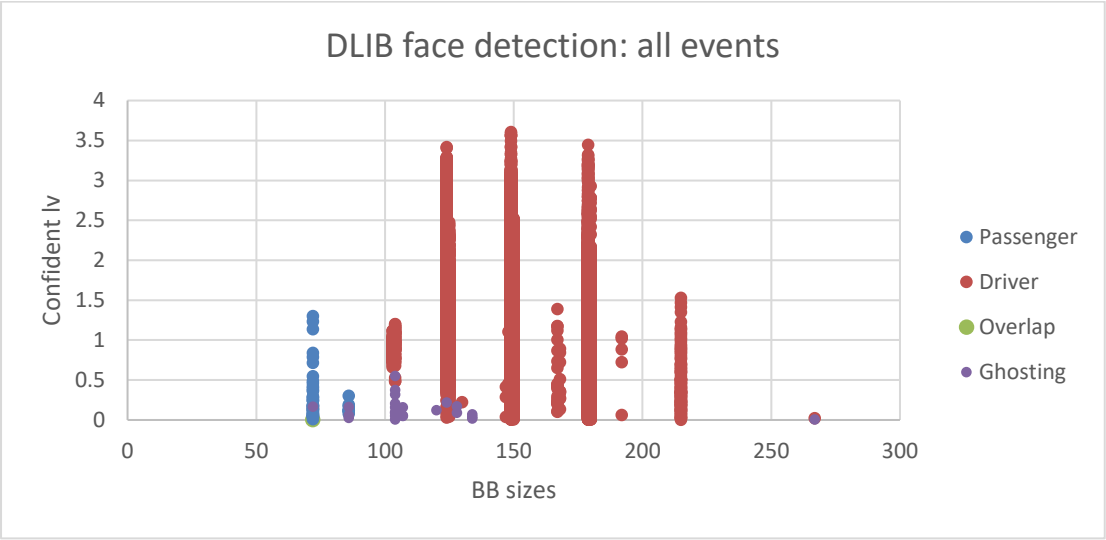
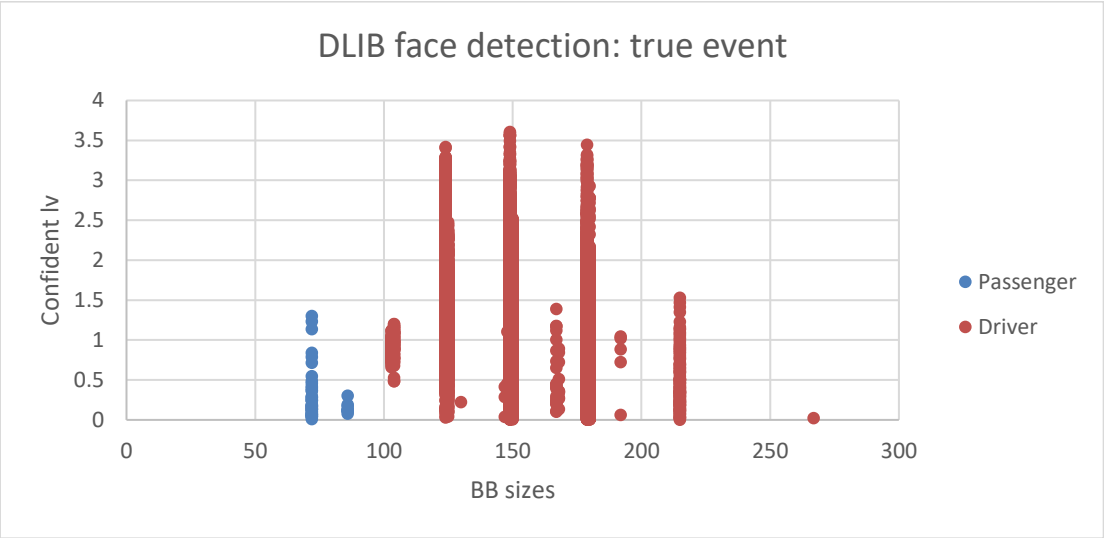


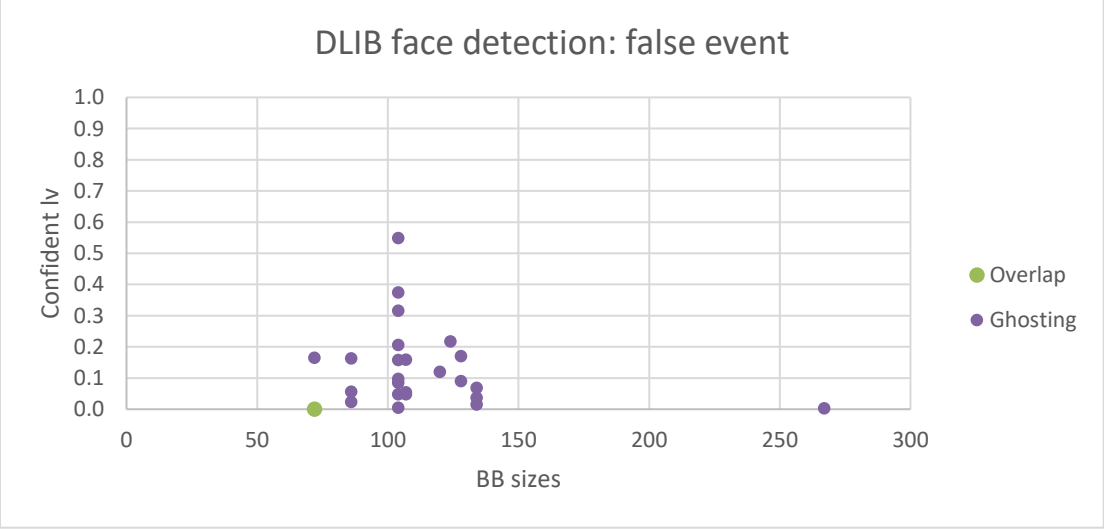
Figure 4.13 The quadrant cut of HAAR detection event



(A) All events of bounding box in scatter chart



(B) True events bounding box in scatter chart

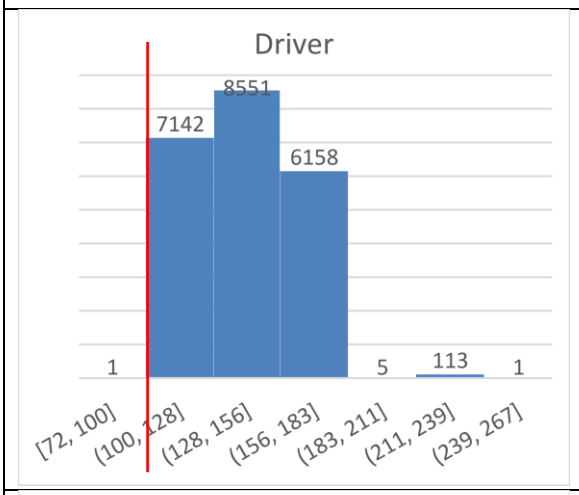
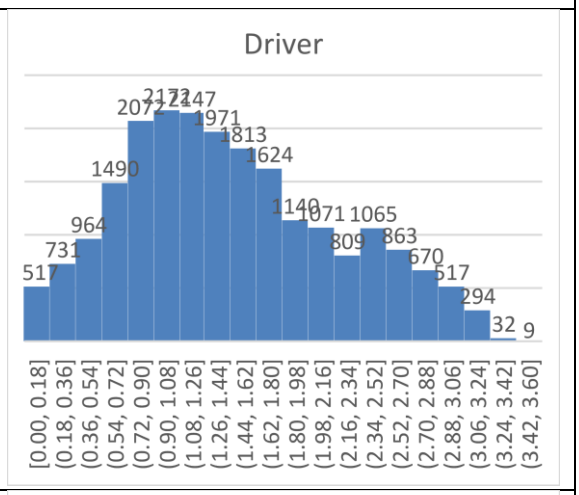
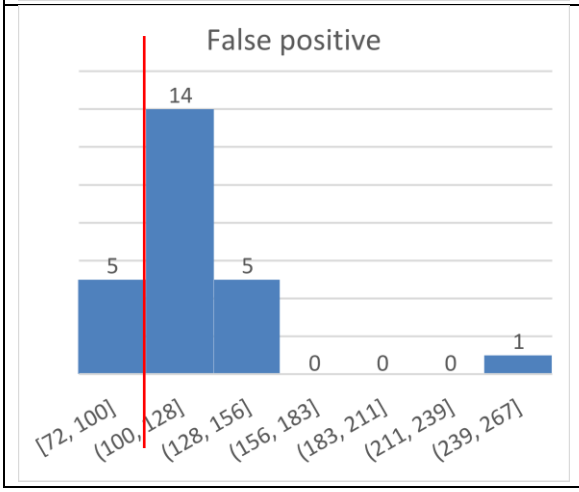
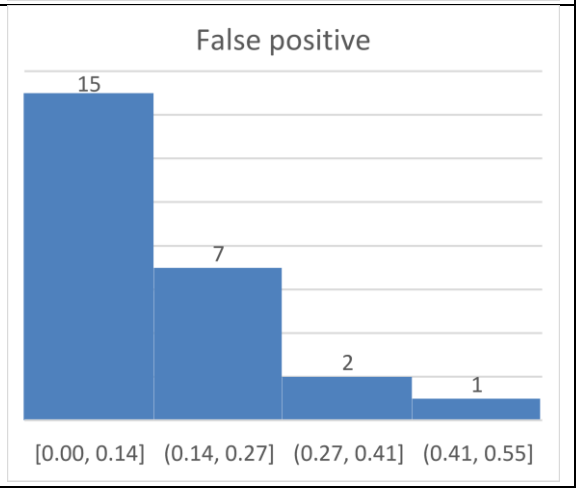


(C) False events bounding box in scatter chart

Figure 4.14 The DLIB face detection bounding box event comparison

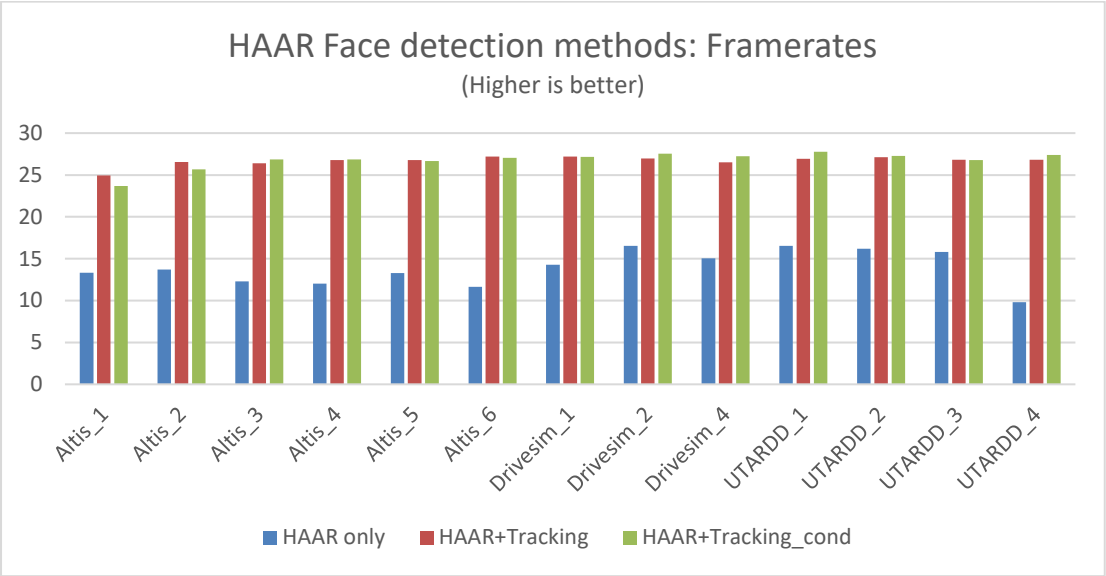
From chart in Figure 4.14(A), the BB sizing is not distributed in any pixel that mean the DLIB face detection has trained in specific BB sizing but not limit in confidence level that has distribute value characteristic.

Table 4.6 The DLIB bounding box of all events in histogram chart

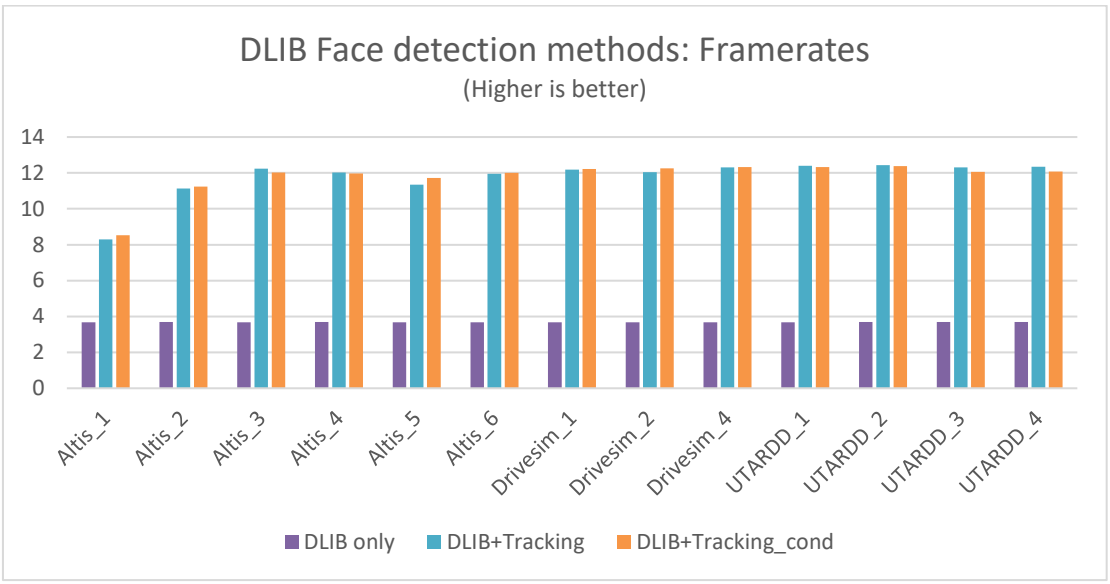
Bounding box sizing	Confidence level
	
	

From Table 4.6, because of the amount of event has too much gap that have not significant to classify by confidence level. The BB sizing below 100 pixels has false positive event than driver event that can be acceptable to thresholding event by using the BB sizing start from 100 pixels.

Next step is testing the face tracking in two methods; 1) face detection using face tracking from the biggest bounding box and 2) face detection using face tracking with parameter thresholding then compare two methods with face detection only. This comparison makes for better development of detection algorithms in RPi4.



(A) HAAR face detection with face tracking framerate comparison



(B) DLIB face detection with face tracking framerate comparison

Figure 4.15 Face tracking framerate improvement comparison

From Figure 4.15, using face tracking can increase framerates significantly. HAAR face detection with face tracking can increase framerate 192% and DLIB face detection have huge framerate increasing about 3 times (320%) compared to not use face tracking. In face tracking comparison, using parameter thresholding does not have significant framerates change.

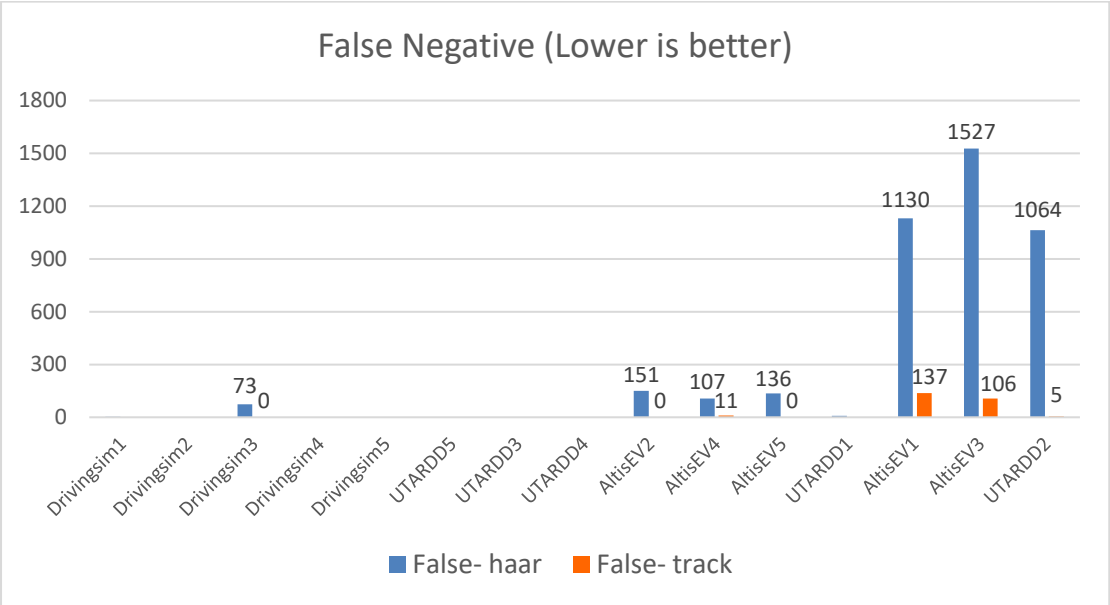
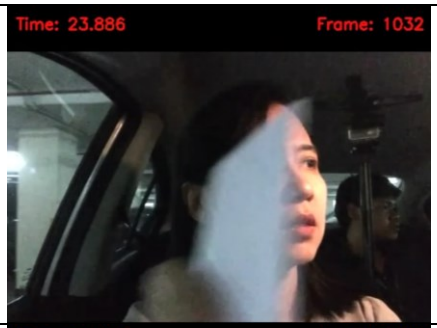
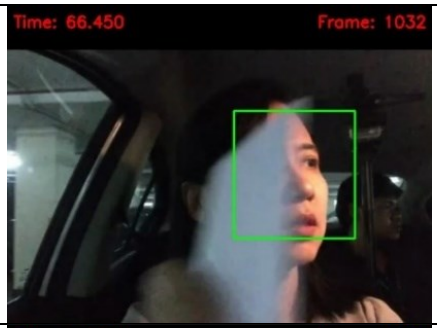
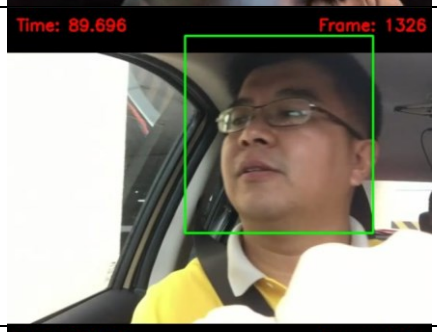


Figure 4.16 Face tracking in False negative events comparison

Table 4.7 Face Tracking and Face Detection frame-to-frame comparison

Face detection only	Face detection with face tracking
	
	
	

For frames that are undetectable using Face tracking, most of them are due to the presence of too many obscuring objects for the Tracking Quality value to be too small to recognize the faces. Although in some situations there is no object obscuring the face, there is a chance that HAAR itself will not be able to detect the face as shown in the following figure.

Therefore, for the further development of the algorithm, the driver face detection part will use face tracking with confidence level checking as the main development.

4.3 Drowsiness algorithm

After coming up with a way to detect faces as accurately as possible, the next step is to analyze the symptoms from the faces to detect the status of the driver. From Figure 3.9, it explains how in Computer Vision there some methods are used to analyze organs before converting the obtained values into driver symptoms at that time. In this study, three important organs were analyzed: the head, eyes, and mouth. The following is an experiment with each extractor to see what symptoms it can detect and how it performs when processed on RPi4.

4.3.1 Facial features detection

DLIB Facial Landmarks is an important tool for facial features detection. This extractor used to predict the position of facial landmarks in 68 locations. In the study, 68 points could be plotted to determine head tilt, eye blinking, and yawning. The working principle in this experiment can be explained by the following flowcharts.

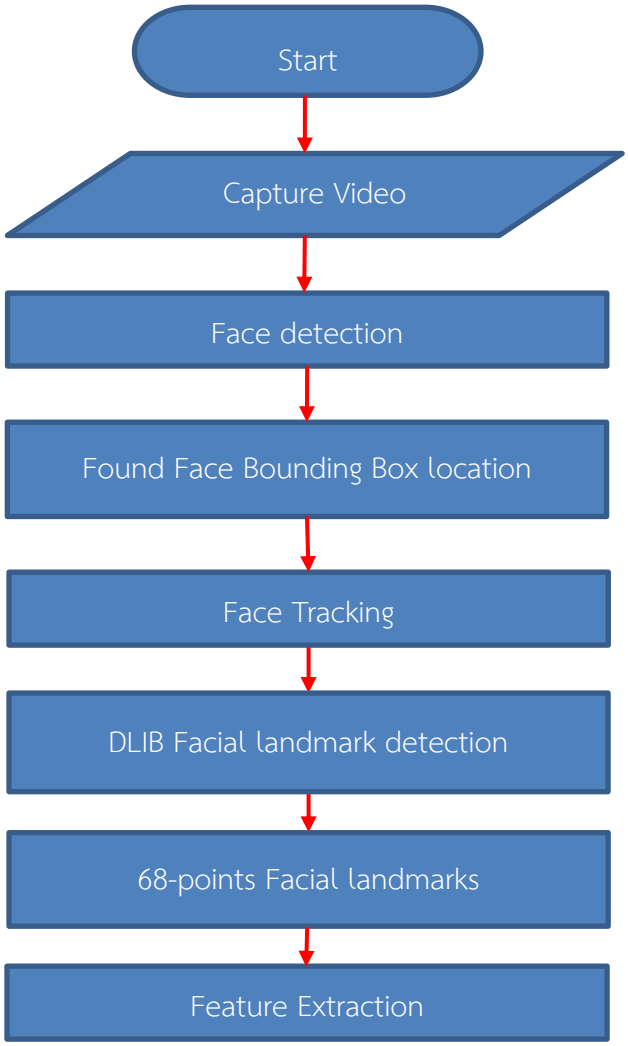


Figure 4.17 DLIB Facial Landmark Flowchart

The addition of this process must be considered in the Performance section, to see how well it is suitable for use in the main algorithm, which is divided into two sections, processing speed and accuracy.

Processing speed is slow or fast can be seen from the frame rate obtained by processing. The following figure shows a comparison of frame rates between 1) face detection alone, 2) face detection using face fracking and 3) face detection use face tracking with facial landmark detection is used.

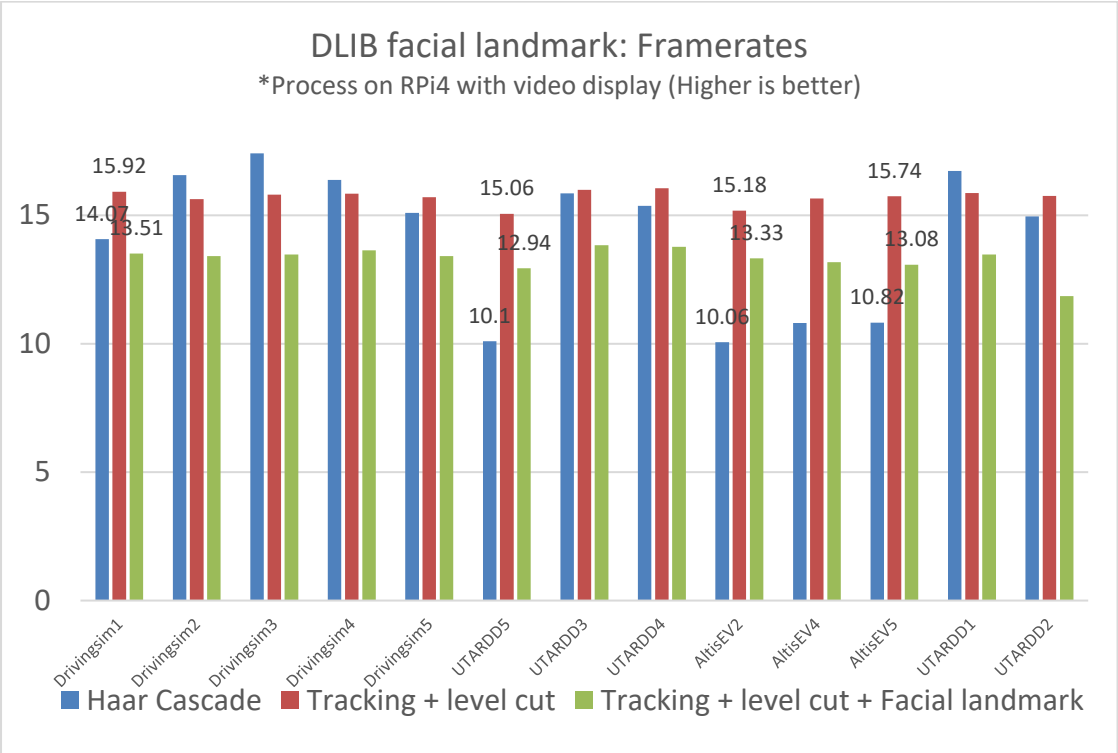
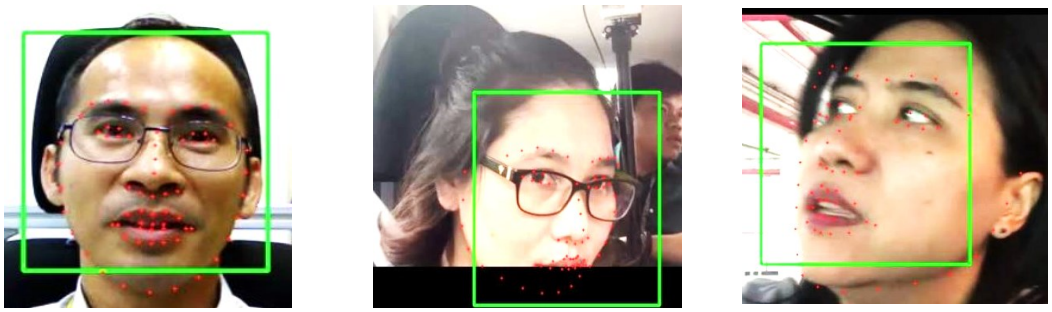


Figure 4.18 DLIB facial extractor framerates performance

From Figure 4.18, the use of DLIB facial extractor results in lower framerates compared to face detection but there are cases that perform better than using the HAAR face detection alone. This is a tradeoff to perform more functions. The framerate obtained by acquiring a Facial Landmark was 6.80% slower than using face Tracking with level cut, but still 5.13% faster than using pure HAAR face detection, which is not a significant difference. The result process on RPi4 with video display that made the framerate lower than the framerate comparison in Figure 4.15(A).

The DLIB facial landmark accuracy measurement uses the five organ landmarks observation principles as shown in Table 3-1 to facilitate the percentage accuracy. For the example to be used to observe the accuracy here, 7 examples are selected in the following situations. 2 examples from driving simulator, 3 examples from city driving situation, and 2 examples from self-portrait recorded.



(A) Corrected facial landmark position

(B) Wrong facial landmark position

Figure 4.19 DLIB facial landmark detected position

The example from Figure 4.19(A) shows how the correct position detection can be achieved. Although some parts of the driver's face are off frame, the DLIB facial landmark itself tries to predict the position of the chin to match the rest of the face that is not obscured.

Sometimes the DLIB facial landmark is unable to accurately predict the facial feature landmarks that show in Figure 4.19(B). We should investigate the results of the experiment in each sample. But preliminary hypothesis that it is probably caused by low light conditions, or drivers often turn into critical angles which is the same reason that makes face detection fail.

Table 4.8 DLIB facial landmark prediction results from observation

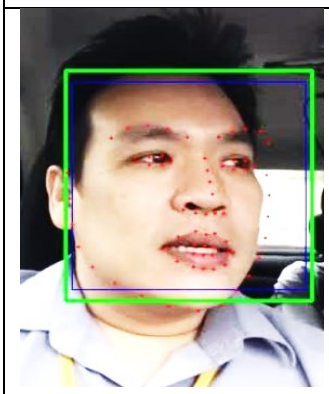
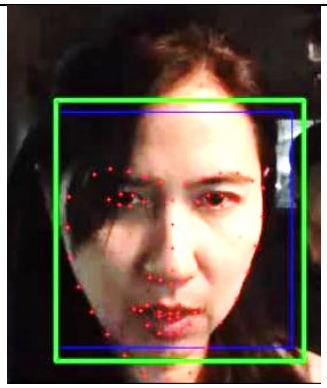
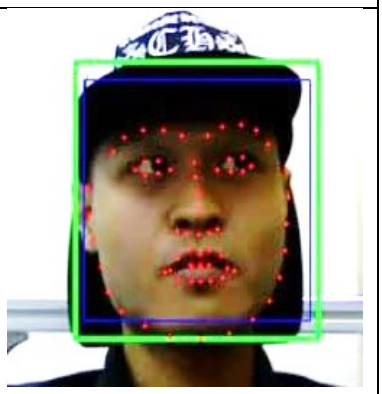
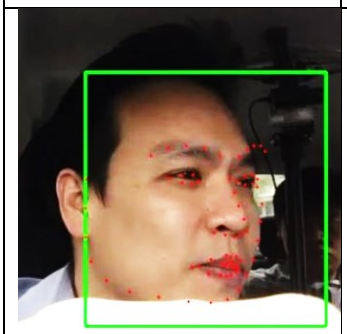
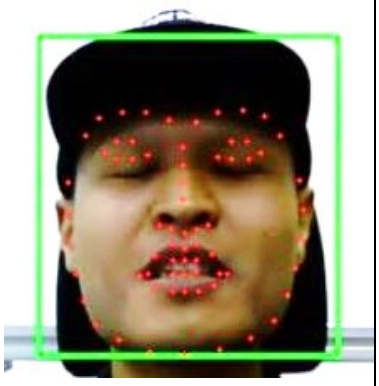
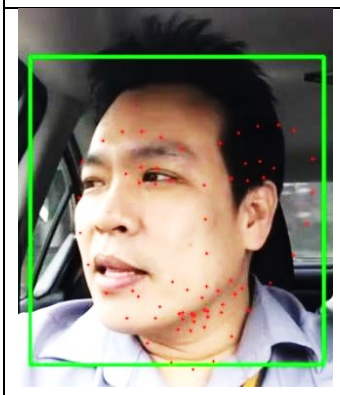
Situation (Head movement /Light condition)	*Dataset Name*	*Face Detector runtimes*	*Facial Landmark Position*
A: Frontal / Bright	*Drivingsim#2*	*1*	*Good*
A: Frontal / Bright	*Drivingsim#4*	*1*	*Good* *Eye position: Not Good*
B: Mix / Bright	*Altis #2*	*1*	*Not Good*
B: Mix / Dim	*Altis #4*	*4*	*Good*
B: Mix / Bright	*Altis #5*	*1*	*Good*
B: Mix / Bright	*UTARDD #1*	*1*	*Good*
C: Frontal / Dim	*UTARDD #2*	*1*	*Good*

From Table 4.8 above, it is stated how many cycles face detection runs as well, because face detector runs one cycle to get the original bb position, after that face tracker will always try to insert a position based on that initial position. If the face detector has started working several times, it means that the driver may turn the face away from the camera angle that difficult to detect or have something covering the face for a long time until the tracking quality value is too low.

Determining whether the case was good or bad predicted the position was based on observations from the video files by the researcher. Most of the examples have a single round of face detection. The driver's face does not have moved from the original position much, making the facial landmarks positioning good. For cases where the position is poor, it is necessary to study what factors contribute to the wrong position prediction. Including cases where the face detector works often.

The following table will show the image in the poor case videos start from the first frame and the following frame, where some frames the landmarks position is starting to distort to determine what factors are involved. There are 3 examples of cases that do not predict the position well.

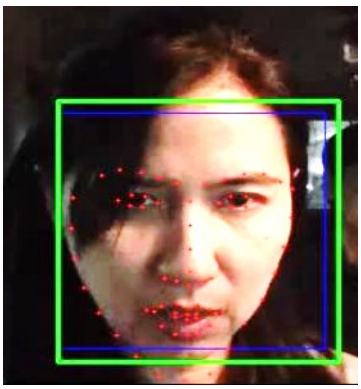
Table 4.9 The image from poor facial landmark detection case

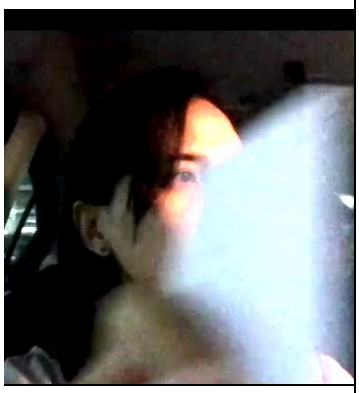
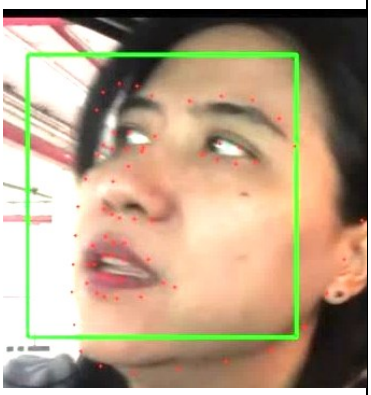
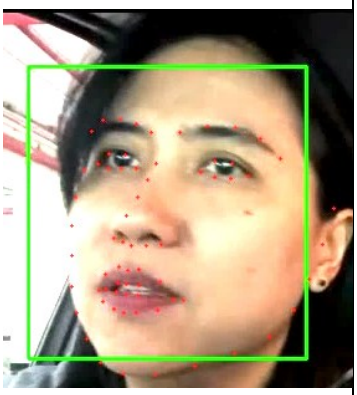
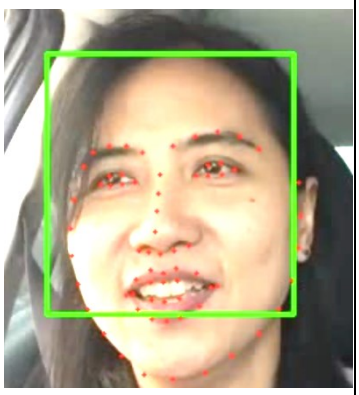
Case: AltisEV#2	Case: AltisEV#5	Case: Drivingsim#4
		
Frame: 1 Confidence level: 3.81 Tracking quality: 30.67 Landmark: 100%	Frame: 1 Confidence level: 7.09 Tracking quality: 21.88 Landmark: 100%	Frame: 1 Confidence level: 5.44 Tracking quality: 30.67 Landmark: 100%
		
Frame: 148 Tracking quality: 11.17 Landmark: 90%	Frame: 661 Tracking quality: 11.60 Landmark: 10%	Frame: 1226 Tracking quality: 18.96 Landmark: 80%
		
Frame: 656 Tracking quality: 17.75 Landmark: 0%	Frame: 774 Tracking quality: 15.89 Landmark: 90%	Frame: 1397 Tracking quality: 13.89 Landmark: 90%

From Table 4.9, all case has miss alignment position after the facial landmark detect for a while especially the case that have frequent movement such as Altis#2 and Altis#5. The frontal looking stable movement case as Drivingsim#4 has a few miss alignments as seen in frame 1226 when the driver closing eye, but in this case, the driver has no eyebrow that maybe made the DLIB facial landmark detector has no clue about eye and eyebrow location. Finally, this table can summary that the tracking quality value in frontal looking much higher than side looking and it has no relation with facial landmark position.

Therefore, from experiments in many cases where the facial landmark locations were wrong, it was found that two factors affecting the accuracy. First is subject had been facing into a critical angle often that make landmark detector predict wrong location. Second is the driver's face is obscured in key reference points such as eyebrows, as in Drivingsim#4 case that has been explained before, making the prediction of eye position distorted. In the case of Altis#EV5 that the face detection has run 4 rounds, we found something interesting.

Table 4.10 The facial landmark detected with several face detection

		
Frame: 1 Confidence level: 7.09 Tracking quality: 21.88 Landmark: 100%	Frame: 661 Confidence level: 7.09 Tracking quality: 11.60 Landmark: 10%	Frame: 774 Confidence level: 7.09 Tracking quality: 15.89 Landmark: 90%

		
Frame: 998 Confidence level: - Tracking quality: - Landmark: -	Frame: 1004 Confidence level: 7.96 Tracking quality: 11.39 Landmark: 100%	Frame: 1034 Confidence level: - Tracking quality: - Landmark: -
		
Frame: 1402 Confidence level: 5.21 Tracking quality: 8.70 Landmark: 90%	Frame: 1456 Confidence level: 5.21 Tracking quality: 13.07 Landmark: 100%	Frame: 1743 Confidence level: 6.65 Tracking quality: 17.78 Landmark: 100%

From Table 4.10, the several face detection cases make the facial landmark position precious alignment even the driver has turn to side-looking several times. When face detection starts again, miss alignment facial landmark position will correct again. Therefore, if we settle the amount of frame to restart face detection or call “refreshing frame” to refresh the face location then facial landmark position will be more accurate in long-term. In the following flowchart show the refresh frame technique with refresh every 100 frames.

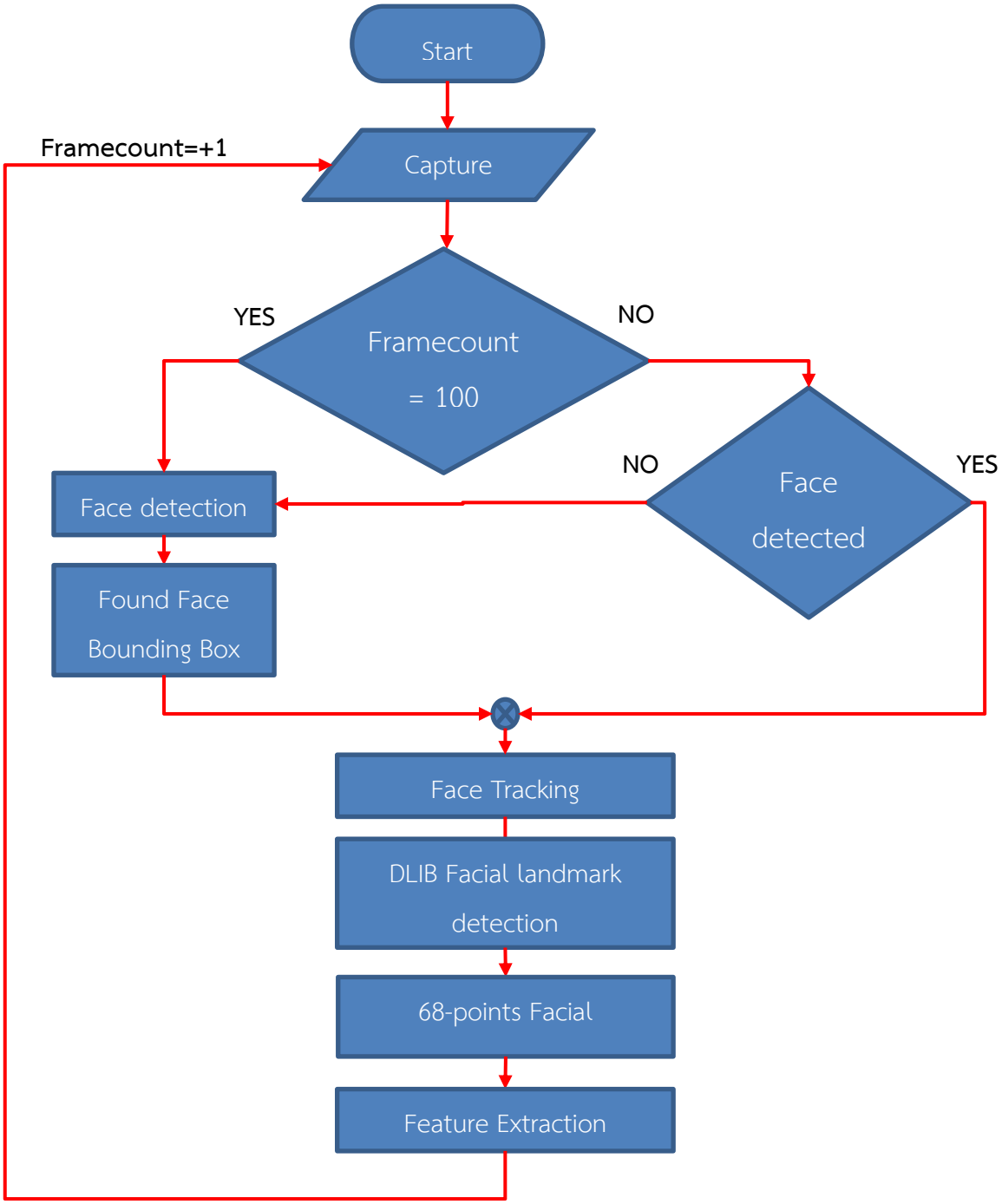
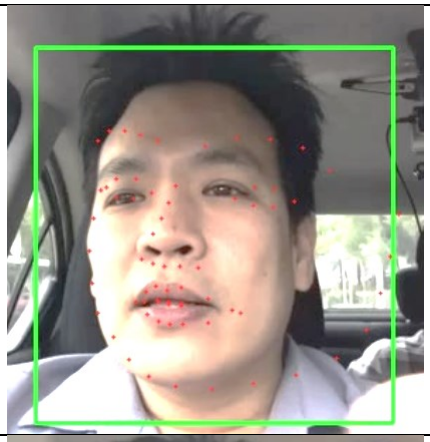
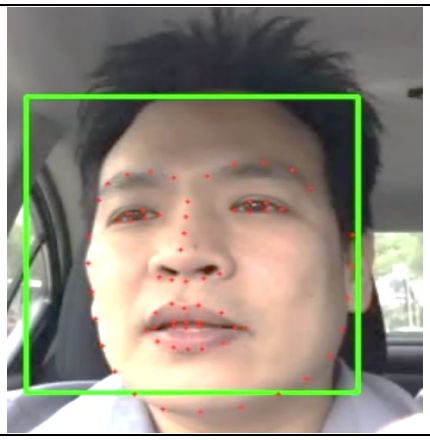
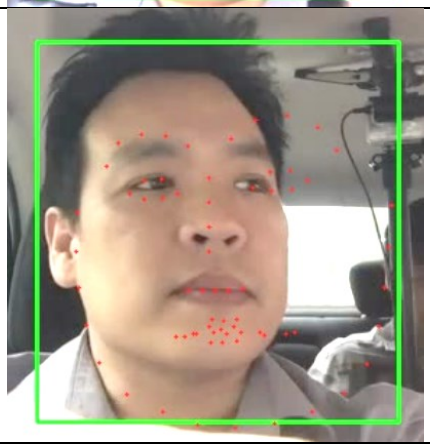
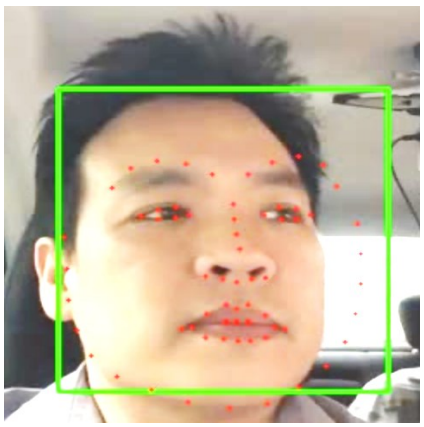


Figure 4.20 Facial landmark detection with refresh frame technique

Table 4.11 Facial landmark frame comparison between normal detection and refresh frame detection

Without refresh frame	Frame	With refresh frame
	505	
	1698	

Therefore, in order to detect organ position more accurately than ever, it is assumed that if using the face detection frame refreshing technique, the face detector will be forced to reactivate every time after a preset number of frames has been reached in order for face tracking to be update the latest face location more frequently to significantly reduce errors in predicting facial landmarks.

4.3.2 Facial feature extraction

When the facial landmark position of driver has been detected, the next thing is extracting the position to be facial feature. The following topic are necessary facial features that can convert to driver status. The using amount of facial landmark points depends on each facial feature.

4.3.2.1 Head extraction

Head extraction using 16 points of 68 points DLIB facial landmark to create rectangle box of the driver's head and using 5 points around nose tip as a reference point. The head extraction can get roll, pitch, and yaw angles of head rotation. The 13 examples are used in head extraction testing. The following chart shows the testing result in terms of processing speed and accuracy.

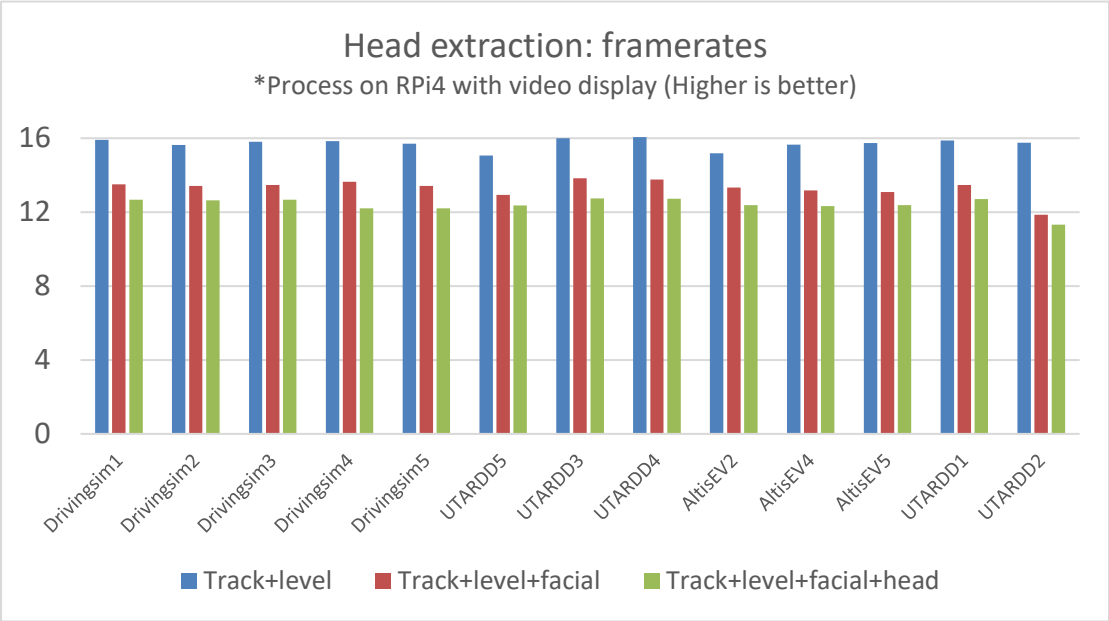


Figure 4.21 Head extraction comparison on framerates performance

From Figure 4.21, using head extraction drain framerate performance from facial landmark detection -8.8% (around 1.1FPS). In terms of accuracy, from observation the roll, pitch, and yaw angles have not relate much with the real angle of head detection. Therefore, using head extraction in this algorithm will make the framerate decrease and the roll, pitch, and yaw are not much accuracy to predict the nodding of driver. It is not significant to predict driver drowsiness if the driver does not fall asleep first.

4.3.2.2 Eye extraction

The eye extraction method uses 6 points of each eye from 68 points DLIB facial landmark to calculate the eye area, both eye area will calculate to the eye value call Eye Aspect Ratio (EAR) that can determine driver status. The 13 examples are used in eye extraction testing. The following chart are the testing results in terms of processing speed and accuracy.

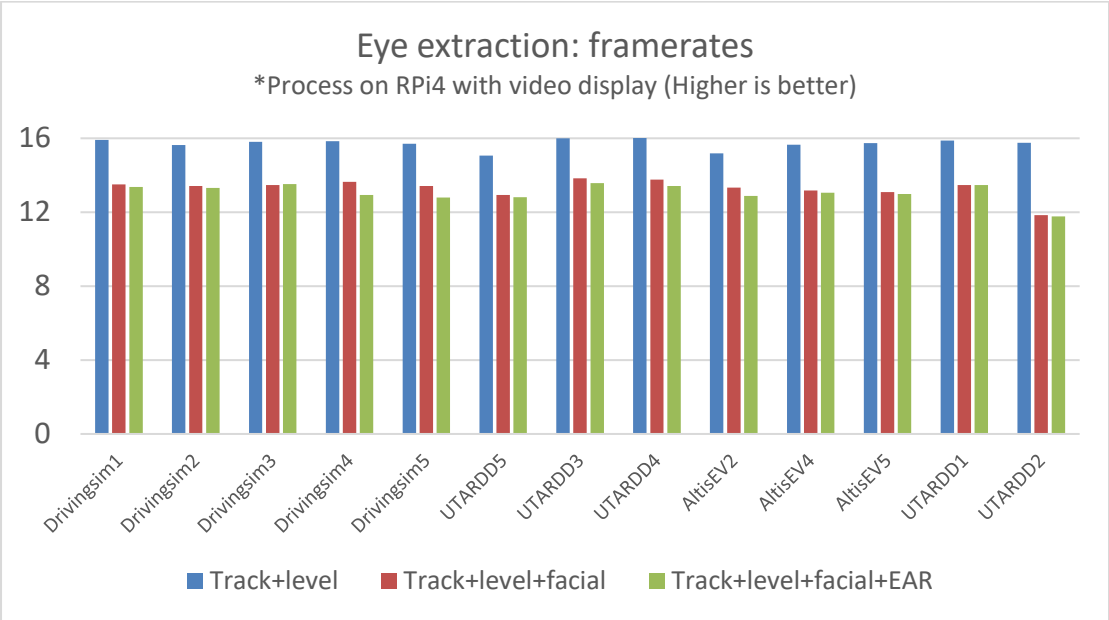


Figure 4.22 Eye extraction comparison on framerates performance

From Figure 4.22, eye extraction has similar framerate as facial landmark detection that mean this does not drain processing speed in main algorithm. In accuracy, the following figure will show the EAR signal and the blinking ground truth event comparison to estimate the EAR signal can really determine the eye status.

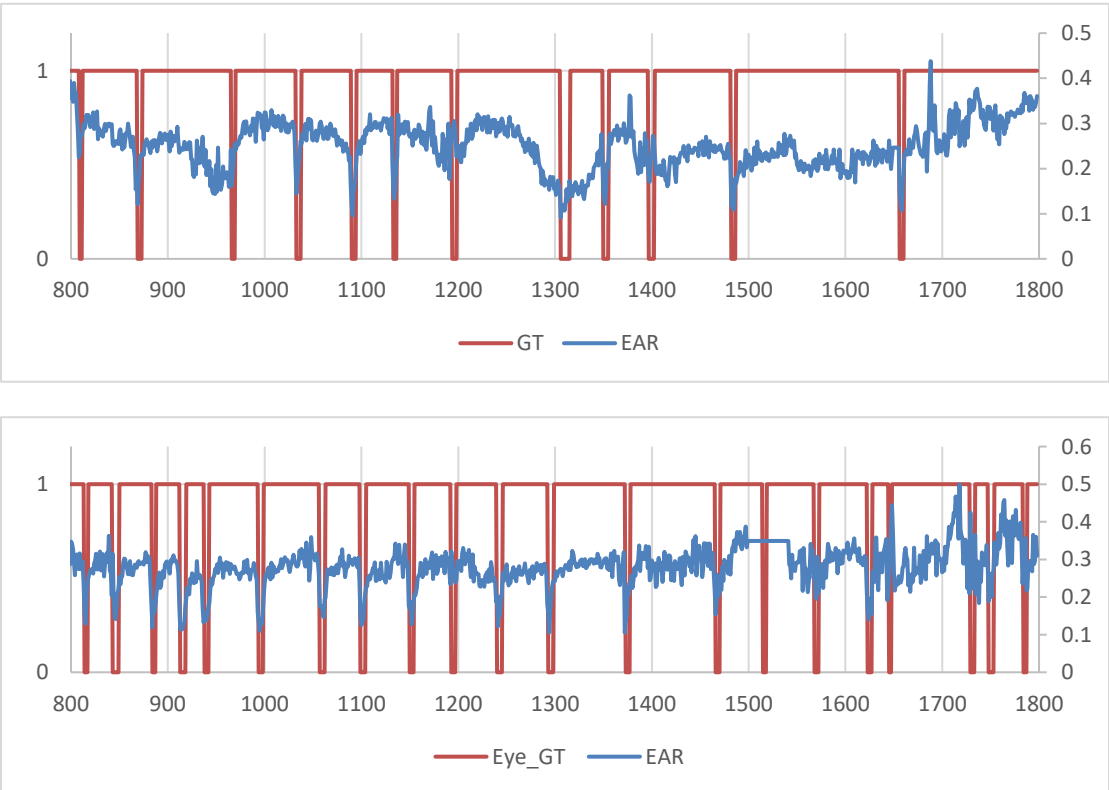


Figure 4.23 EAR signal and blinking ground truth event comparison

Figure 4.23 show the potential of EAR signal that can detect the eye blinking, the lower value decrease on the blinking zone that mean it has relation between EAR value and eye blinking. Therefore, eye extraction will be used in the main algorithm to detect the eye event without drain processing speed.

4.3.2.3 Mouth extraction

Mouth extraction using the same technique as eye extraction with only 5 points of DLIB facial landmark detection around inner mouth position. The framerate is the same as using facial landmark detection.

For more information, the accuracy topic will be described in topic 4.3.4.3. Therefore, mouth extraction does not affect the drain of processing performance and has potential to detect the yawning. The mouth extraction will be used in the main algorithm.

4.3.3 Optimization method for feature landmark signals

The significant feature for predicting status of driver come in area values, the accurate value should concern in algorithm development. The following method is trying to improve the signal quality for better symptoms detection.

4.3.3.1 Image resolution

The first method is image resolution, because the signal depends on the area of facial feature. Sometime the small picture cannot clarify the eye area between open and closing eye. In this topic assume that the bigger image resolution can get the better signal in the same event. This testing has compared signal from the image in 5 various resolutions from 480P to 960P in blinking event.

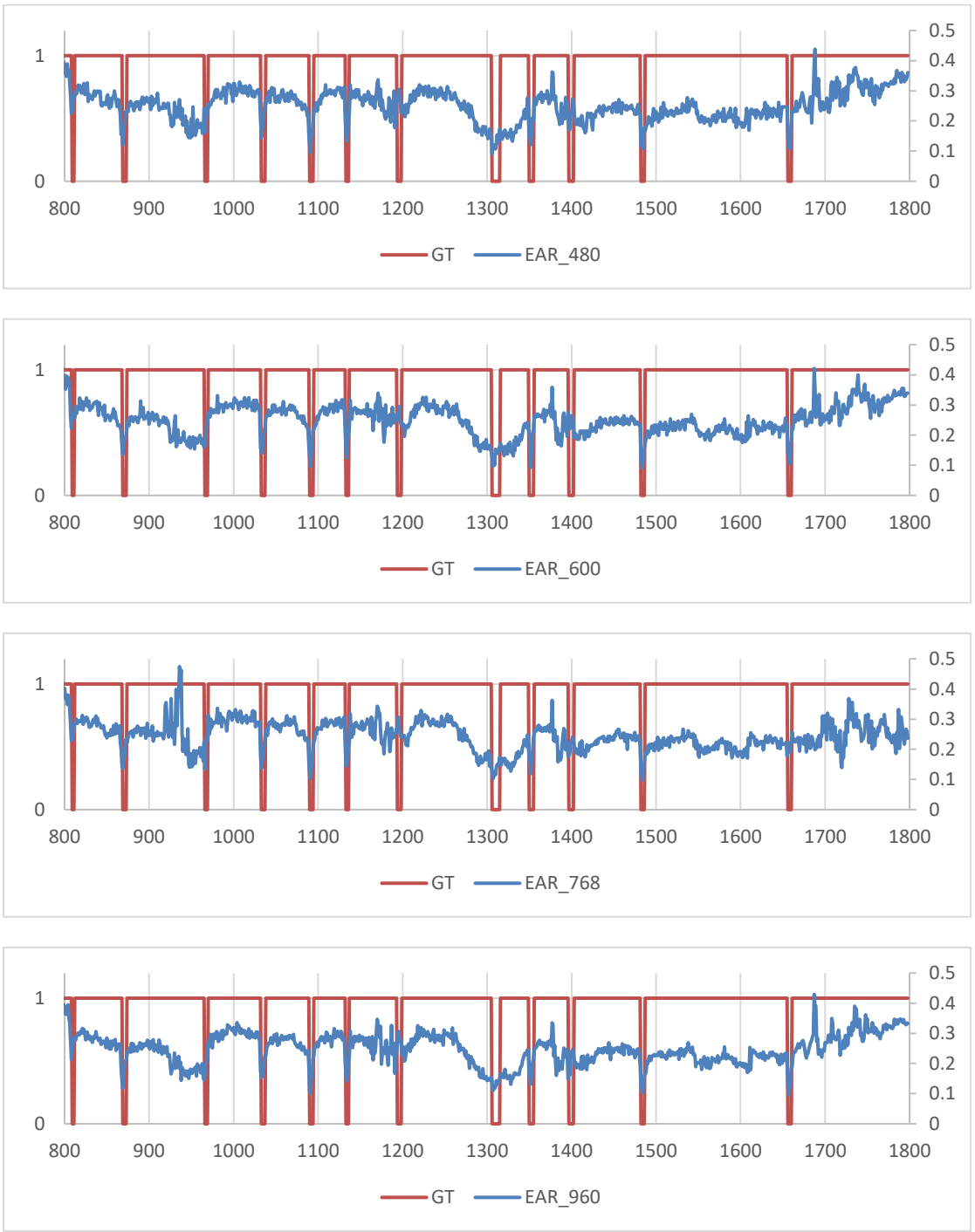


Figure 4.24 EAR signal of blinking events in various image resolution

From image resolution comparison, some driver faces have miss detection in small image but can detected in bigger image. From Figure 4.24, the EAR signal in many cases have clear and smooth signal in biggest image as 960P than small image, but not all the case such as 768P is not have better signal than 600P. Therefore, the bigger resolution can reduce the false positive event of face detection and can get some clarify value signal but not at all. The next step is about the lighting condition because

in this testing, the various dataset in bright condition gets better signal in low light condition.

4.3.3.2 Image equalization

In the previous testing results, the low light condition dataset has not clarified signal even in bigger image. In real-world driving, light conditions can immediately change depending on weather and route. The next signal optimization method is image equalization. The matching technique can get clarify image that maybe can improve the detection and signal quality.

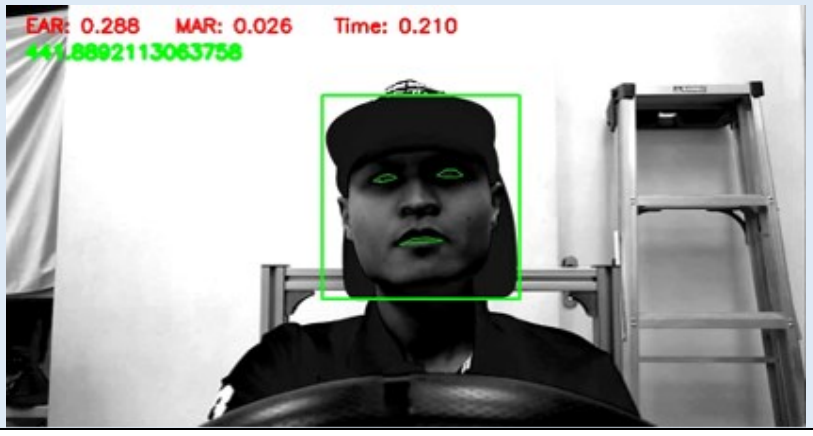
Table 4.12 Image equalizer techniques for signal optimization

	Technique	Parameters
1	cv2.equalHist	---
2	Clipping_histogram	Clipping=20%
3	Clipping_histogram	Clipping=25%
4	cv2.CLAHE	Clip_limit=3.5, Gridsize=10x10
5	cv2.CLAHE	Clip_limit=4.0, Gridsize=10x10
6	a = 255/img.max()	img.max()
7	Color transfer	Good grayscale source image
8	Gamma correction	g=1.5
9	Gamma correction	g=2.0

Nine techniques of image equalizer in Python using in the various video dataset. The parameters of each technique are from the recommend value from online source, and trial & error value.

The following tables show the example of each technique comparison, the information from table are image result and EAR signal pattern between post process from equalizer (480P) and manual adjust light by researcher (720P).

Table 4.13 Image results from each image equalizer

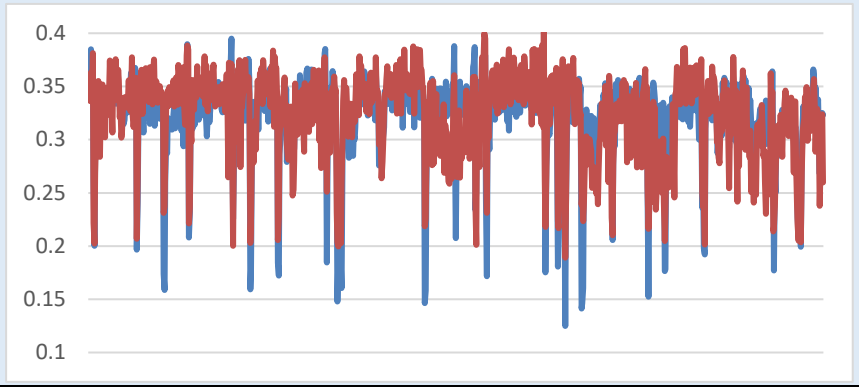
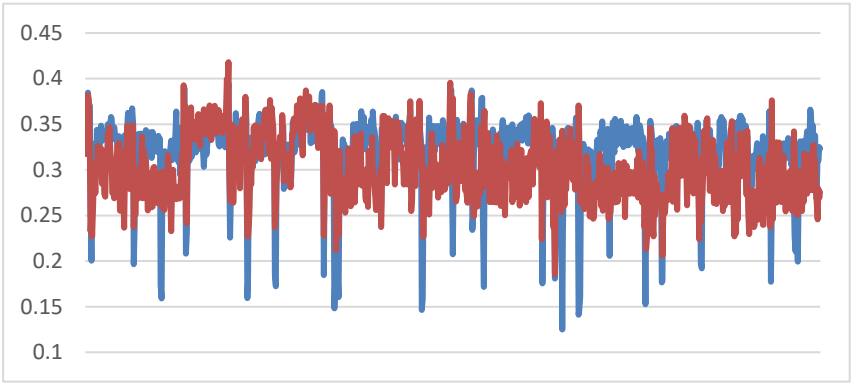
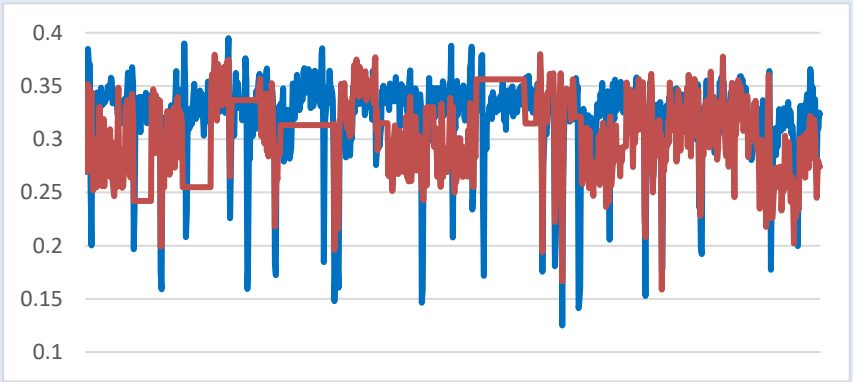
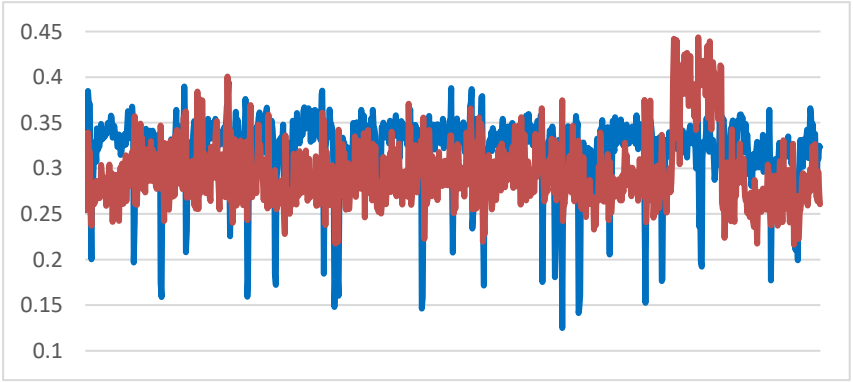
Technique	Image results
cv2.equalHist (Auto adjust)	
Clippinng_hist clipping=20%	
Clippinng_hist clipping=25%	

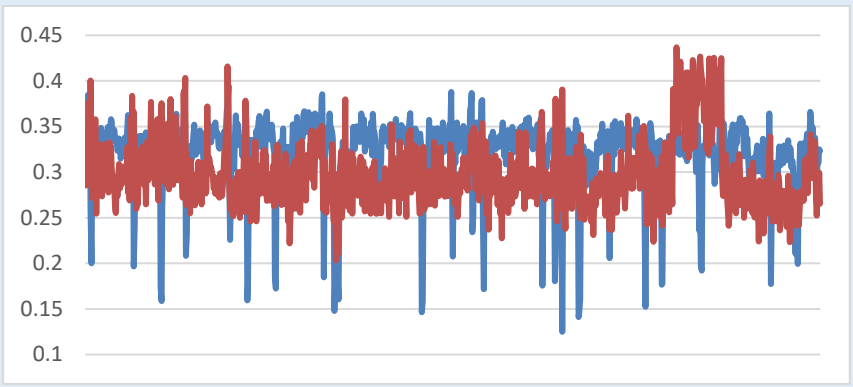
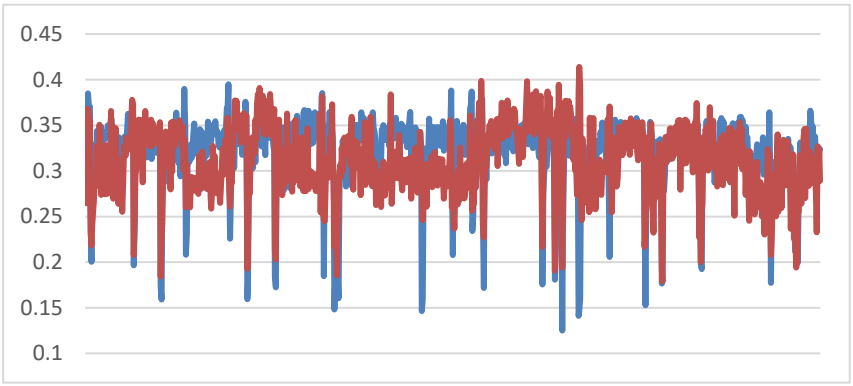
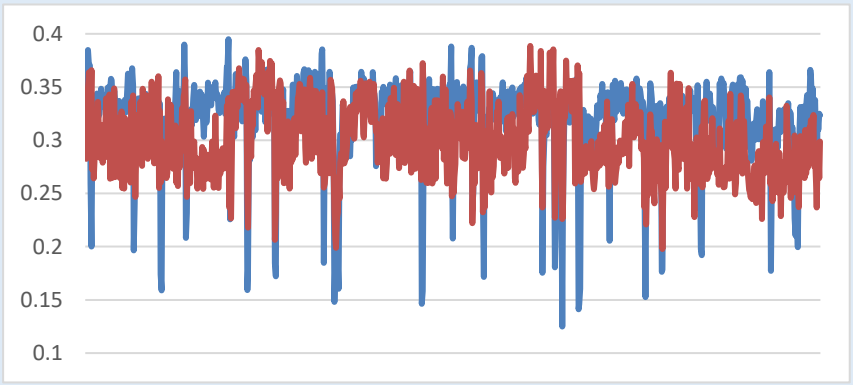
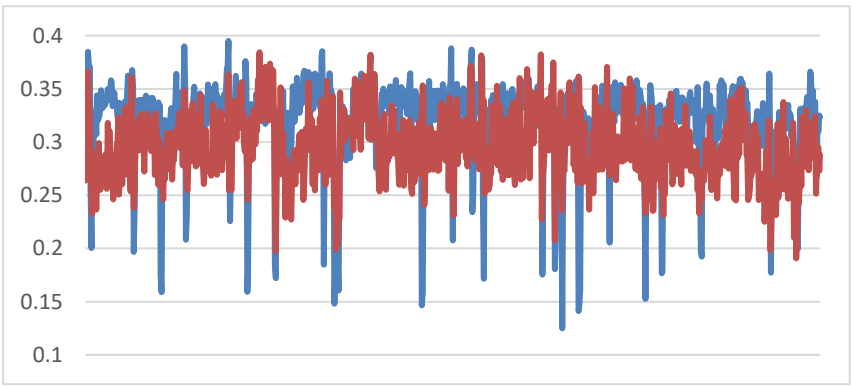
cv2.CLAHE clip limit =3.5	
cv2.CLAHE clip limit =4.0	
Alpha **=255/img.max()** **(Auto adjust)**	

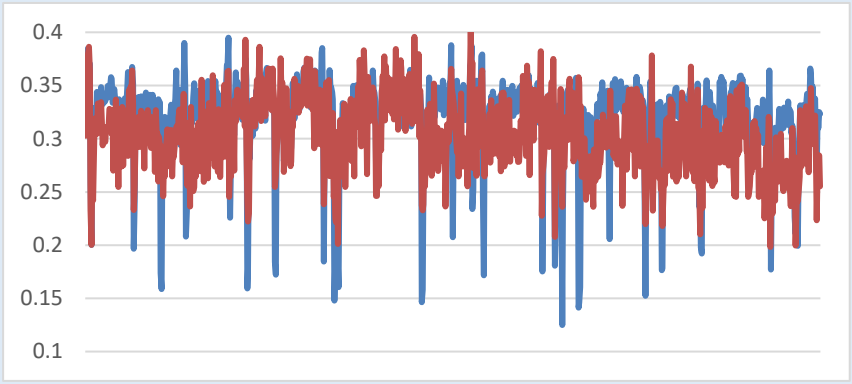
Color transfer	
Gamma correction g=1.5	
Gamma correction g=2.0	

From Table 4.13, show the result image after equalized by each technique. Auto parameters are cv2.equalHist and Alpha value, others are manual adjust value parameter. The next table is EAR signal pattern generated from each equalized image.

Table 4.14 EAR pattern signal from each image equalizer

Technique	EAR pattern signal
cv2.equalHist (Auto adjust)	
Clippinng_hist clipping=20%	
Clippinng_hist clipping=25%	
cv2.CLAHE clip limit =3.5	

cv2.CLAHE clip limit =4.0	
Alpha =255/img.max (Auto adjust)	
Color transfer	
Gamma correction g=1.5	

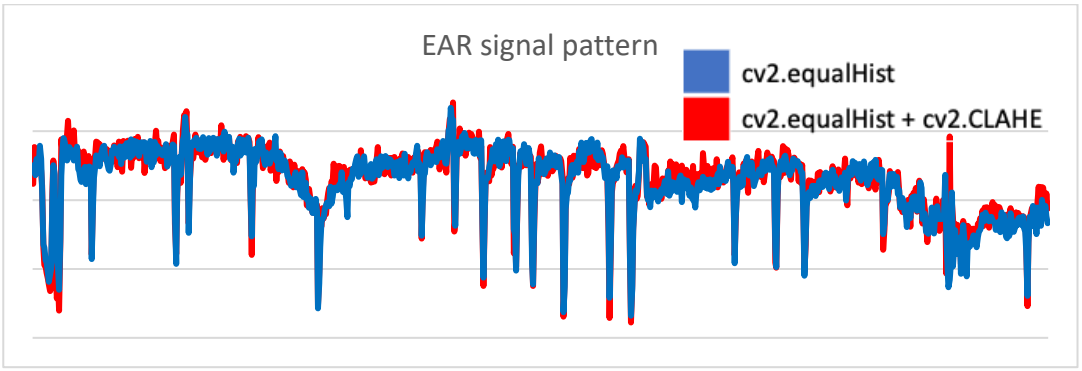
Gamma correction g=2.0	

From Table 4.14, signal pattern on cv2.calcHist is look closer to 720P signal pattern that mean this technique can improve signal quality.

For more methods to make the pattern look better, the next experiment is combining two methods of image equalizer to see the signal pattern result. The equalizer method to use in this experiment are cv2.equalHist and cv2.CLAHE that are easy to use because they are OpenCV-based.

(A)Image from cv2.equalHist

(B)Image from cv2.equalHist +CLAHE



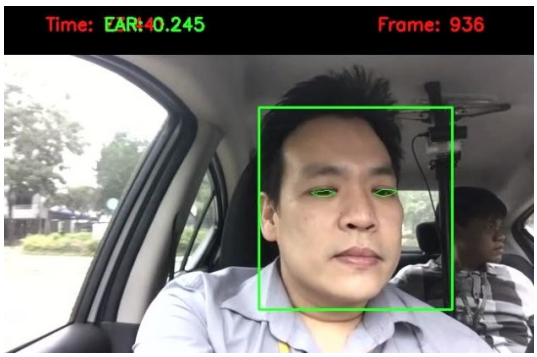
(C)EAR signal pattern comparison between (A) and (B) equalizers

Figure 4.25 The result between using one and two equalizers

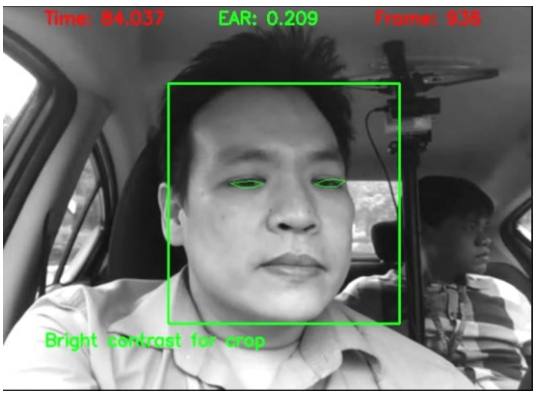
Figure 4.25 (A) and (B) display the image results from both techniques, (B) image look brighter than (A). (C) image display EAR signal pattern comparison, the combine equalizer makes the signal pattern sharper than using cv2.equalHist. Therefore, for better signal the combination will be used in the main algorithm.

4.3.3.3 Object zoom-in technique

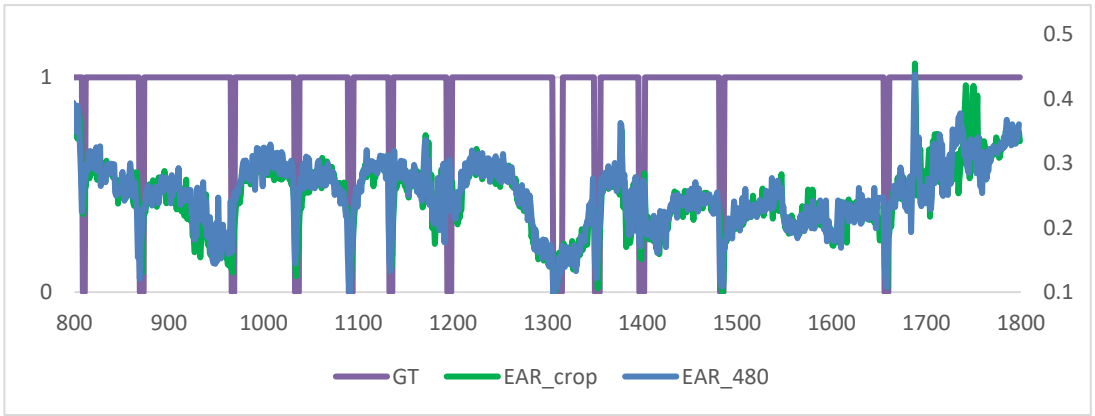
From result in 4.3.3.1 that the bigger resolution has better signal pattern, but the bigger resolution means the slow in processing time. The next technique to make the signal pattern more clarifies is the object zoom-in method. This method has two steps of detection, first detect the driver's face, and get bounding box area, then expand the image around bounding box. Second, using face detection to detect the face in new cropping area and then detect the facial landmark to get the EAR signal. The following table displays the image result and signal pattern to check that can make signal good.



(A) Normal detection



(B) Object zoom-in detection



(C) The signal pattern comparison between normal and zoom-in detection

Figure 4.26 Object zoom-in method image result and EAR signal pattern

From testing result, EAR signal pattern from object zoom-in technique can see more clarify in peak zone than normal detection that can see in Figure 4.26. Therefore, object zoom-in technique will be applied in detection method for better signal quality.

4.3.4 Symptom extraction

From the facial feature topic in 4.3.2, eyes and mouth are selected for explore the driver's status. The EAR signal from the eye can generate to indicate the eye status. The MAR signal uses the same technique as EAR but detects on mouth events. The previous topic tried to optimize these signals to the most clarify status indicator. This topic is trying to use these signals to be the driver status. In this research, the following will describe our method and results.

4.3.4.1 Fatigue level and EAR relations

From topic 4.1.2, we get the eye blinking relationship with fatigue level in term of blink rate and blink duration. The blink duration is necessary for driver status detection. In symptoms extraction, EAR value is the eye status indicator. To indicate the eye status, the EAR detection will apply to dataset in topic 4.1.2 for EAR characteristic study.

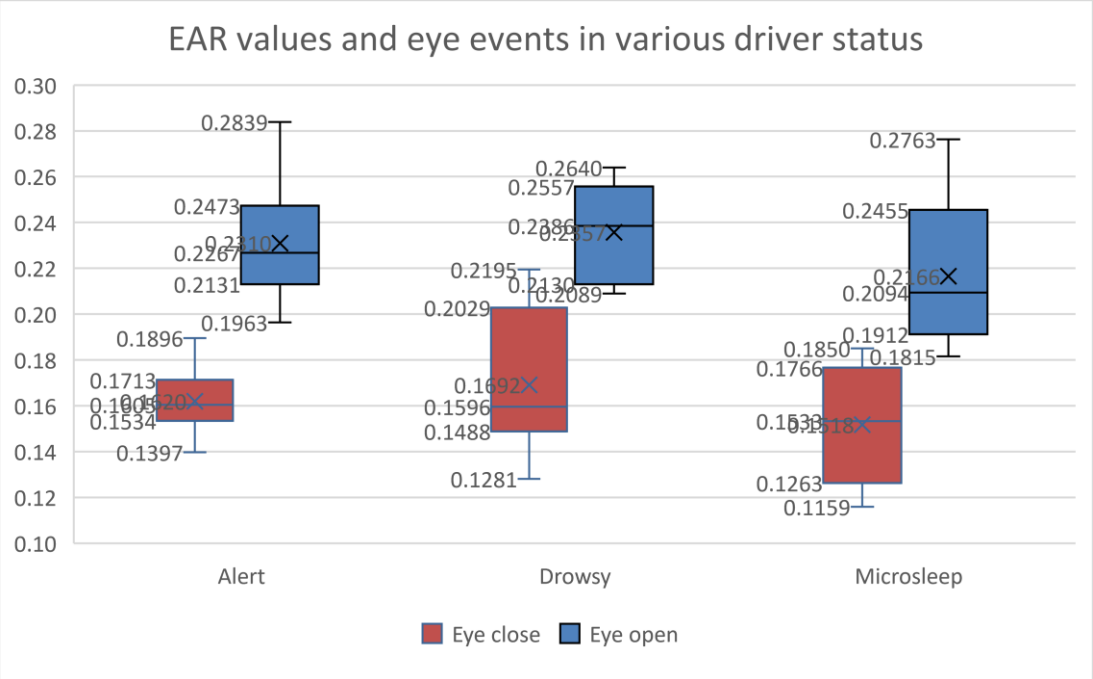
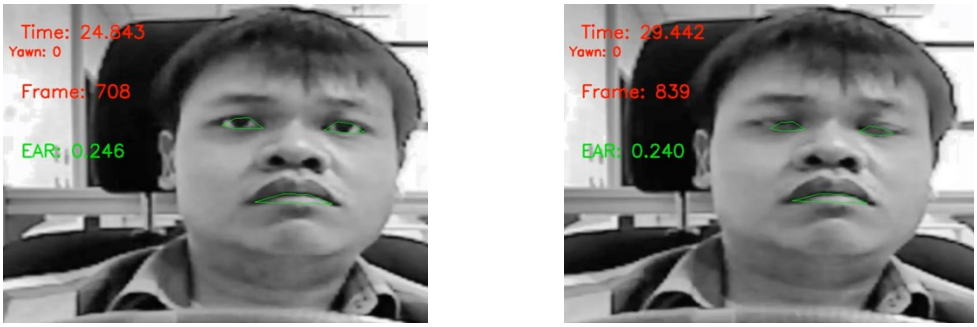


Figure 4.27 EAR value and eye events in various driver status

From Figure 4.27, we have two group of EAR value for eye events in various fatigue level. EAR value between eye-closing status and eye-opening status is can classified into two group, most value of eye closing is under 0.2 that similar to the value in Tereza & Jan's work [29]. In eye-opening value, EAR value is more decreasing depending on more fatigue level because the eye got fatigue that make eyelid heavy and difficult to lift. Some high EAR value in eye-closing event because the facial landmark detection problem, when eye is closing, the facial landmark detector detects the eyelid shape instead of eyeball such as Figure 4.28 (B).



(A) EAR value when eye-opening (B) Missing EAR when eye-closing

Figure 4.28 EAR value with eye events

Therefore, the EAR value can classify eye status by EAR value thresholding into two events. The thresholding value between eye-opening and eye closing is 0.2, but in some cases the higher value can be caused from facial landmark detection problem.

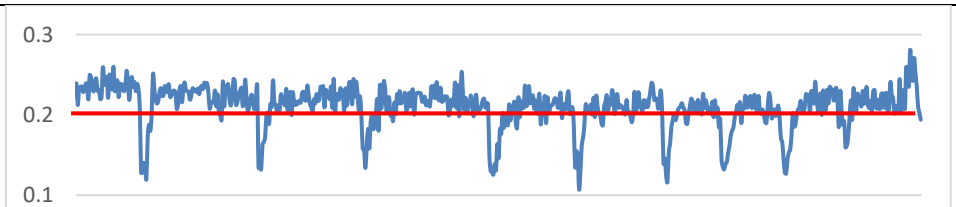
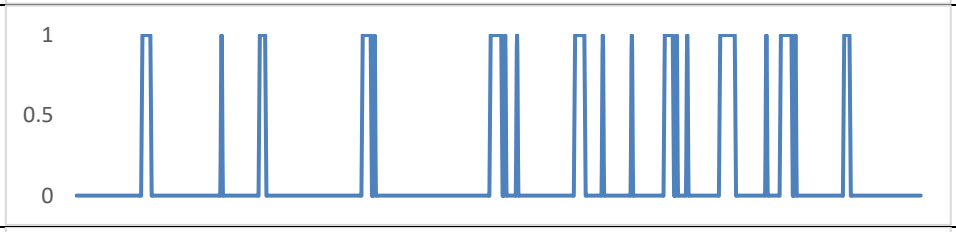
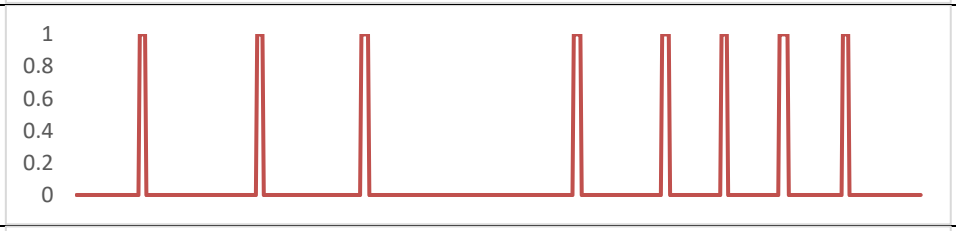
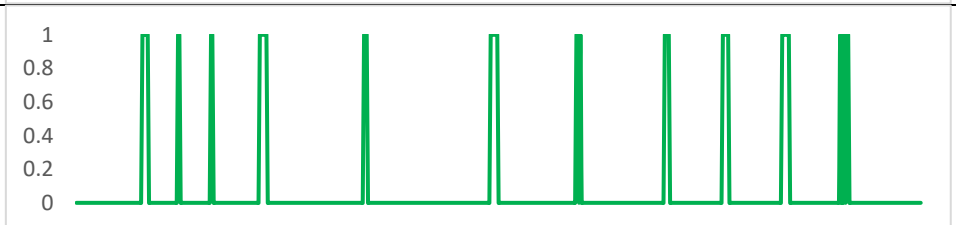
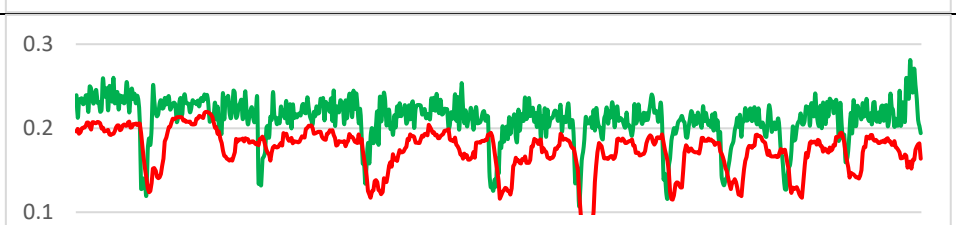
4.3.4.2 Blink detection comparison between thresholding vs. peak Z-Score

EAR signal is the most important for detect eye status, other research has their own methods to detect eye blinking by using EAR signal as see in Table 3.5. Much research uses threshold value to classify between open eye and close eye, some research training the EAR value then making eye status prediction models from machine learning.

From previous topic in Figure 4.27, EAR value below 0.2 can decide that the driver closing eye. But from the same chart, the EAR value is decreasing by more fatigue level or maybe some people have small eyeball that can see minimum EAR value around 0.196 in eye-opening group.

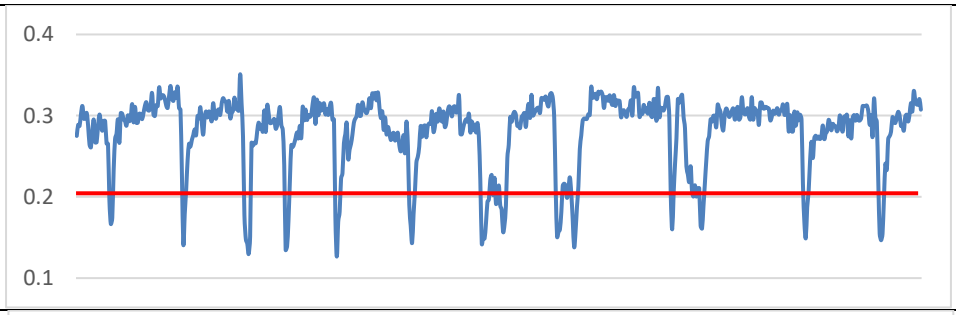
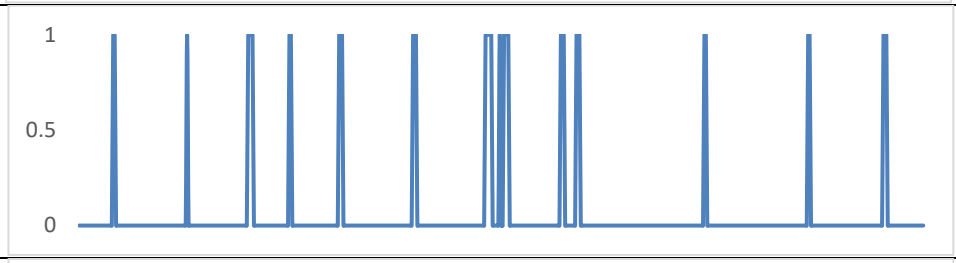
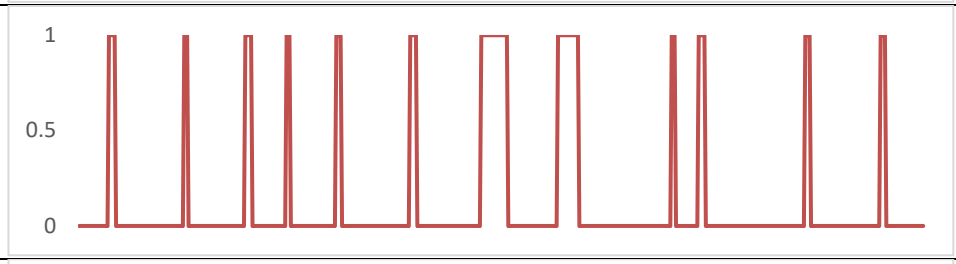
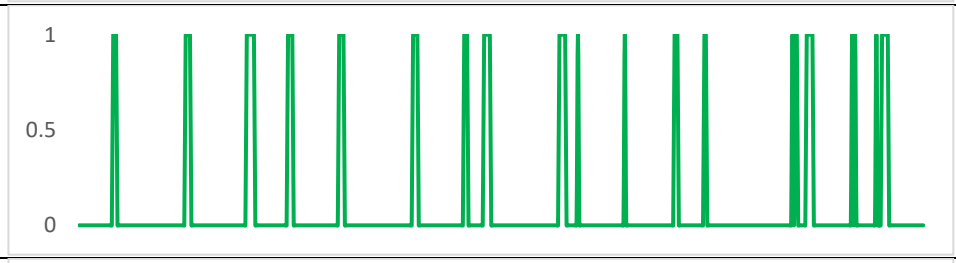
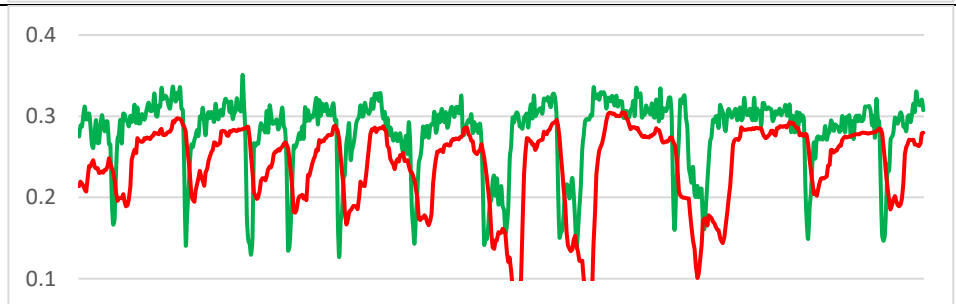
Hence, in this research, we try using two methods to detect eye blinking. First method is using fix EAR value thresholding around 0.2 in more than 0.15 second (around 3 consecutive frame) refer to chart Figure 4.2 that are suitable value for classify eye status. The second method is using Peak Z-score to detect the lower peak events that are estimated to be eye-closing events, this method suitable for the people who have small eyeballs. The blinking detections signal and result of each method compared to ground truth event are displayed in the following table in various fatigue levels.

Table 4.15 The blinking signal detection comparison in alert state

Dataset	Example of blinking signal in alert state
EAR 0.2 signal	
EAR 0.2 Blink (15)	
Ground truth (8)	
Z-score Blink (11)	
Z-score signal	
Result	EAR thresholding: True positive detect: 8 / Fasle positive detect: 7
	Z-score: True positive detect: 8 / Fasle positive detect: 3

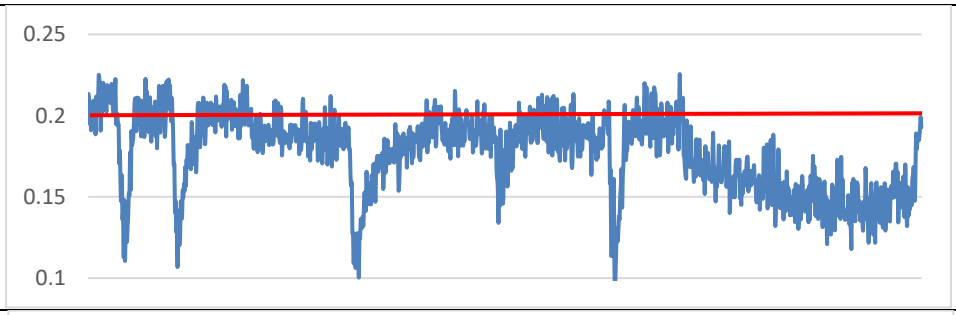
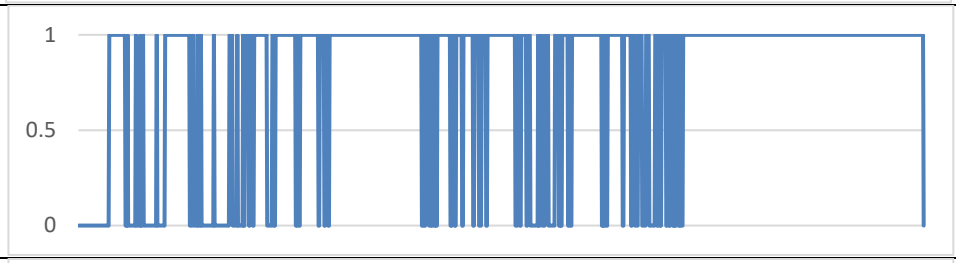
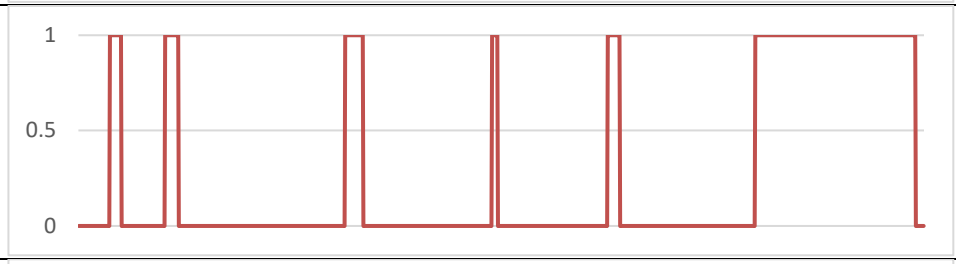
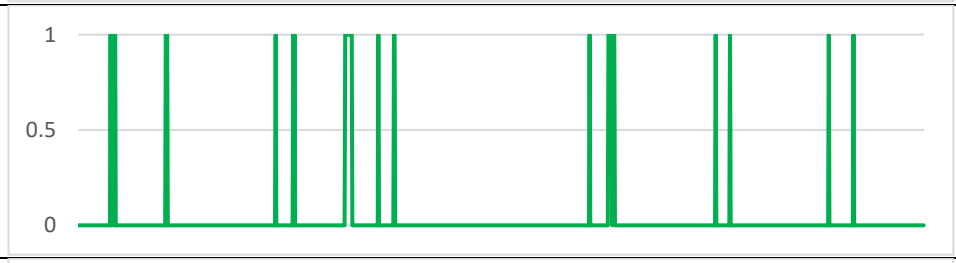
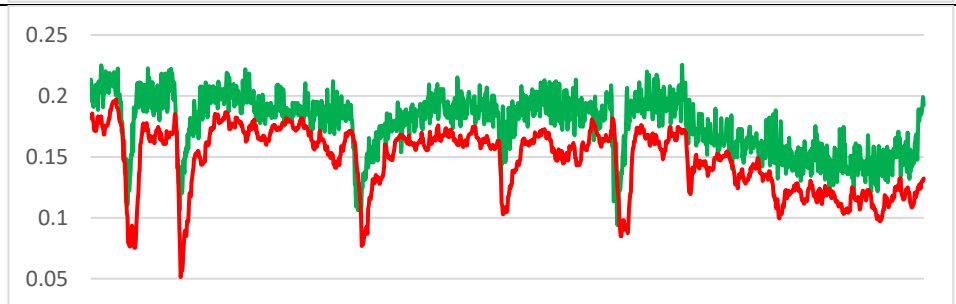
The EAR signal from alert state example in Table 4.15 has constant frequent blinking but has small eyeballs that make EAR value when driver being eye-close have close to under 0.2 that bring EAR blinking threshold method have more false positive detect.

Table 4.16 The blinking signal detection comparison in drowsy state

Dataset	Example of blinking signal in drowsy state
EAR 0.2 signal	
EAR 0.2 Blink (14)	
Ground truth (12)	
Z-score Blink (17)	
Z-score signal	
Result	EAR thresholding: True detect: 11 / F positive: 3 / F negative: 1
	Z-score: True detect: 12 / False positive: 5

The EAR signal from drowsy state example in Table 4.16 have clearly value between eye-open and eye-close that make both detect method more accurate. But in some blink, the driver tries to wake up while eye-closing that make EAR value have W shape.

Table 4.17 The blinking signal detection comparison in microsleep state

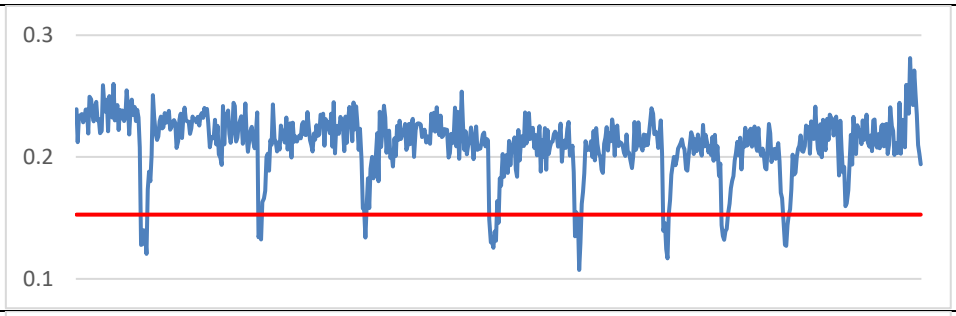
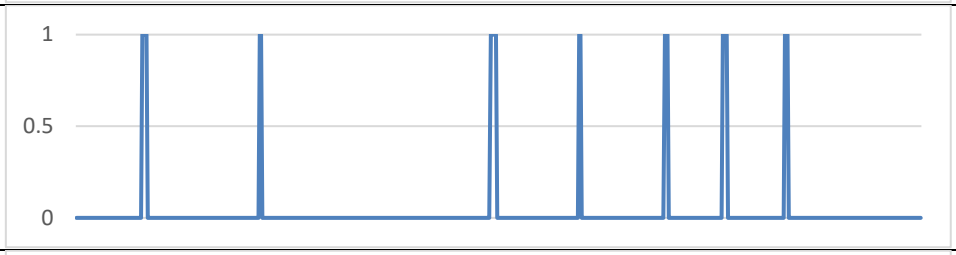
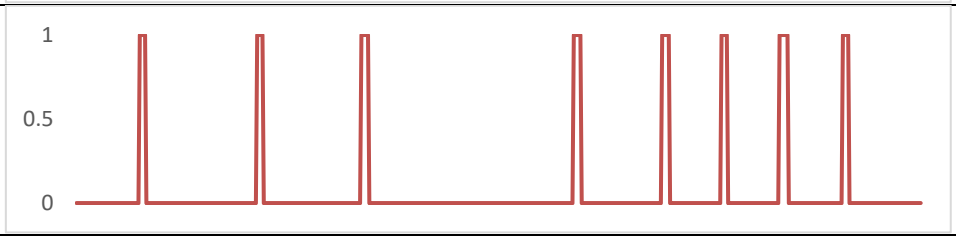
Dataset	Example of blinking signal in microsleep state
EAR 0.2 signal	
EAR 0.2 Blink (61)	
Ground truth (6)	
Z-score Blink (15)	
Z-score signal	
Result	EAR thresholding: True detect: 6 / False positive: 55
	Z-score: True detect: 5 / False positive: 9 / False negative: 1

The EAR signal from microsleep state example in Table 4.17 see that when driver have much fatigue, the EAR value has low to under 0.2 whether eye-closing or eye-opening that make EAR thresholding method have much false positive blinking detection. While the Z-score peak detection method has false negative detection because it has no peak due to long blinking.

The advantage of EAR thresholding method is this method can detect long-blinking event accuracy because of fix value threshold but have some mistake detection if the driver has small eyes or when normal blinking that EAR value has sudden change that make blink rate not much accurate. While the Z-score peak detection is good at normal blink detection because the peak from sudden value change can clearly detect but when long-blinking event this method does not work at all. In drowsiness detection, long-blinking is very necessary more than normal blinking because that means the driver nearly to sleep state that difficult to get awake again.

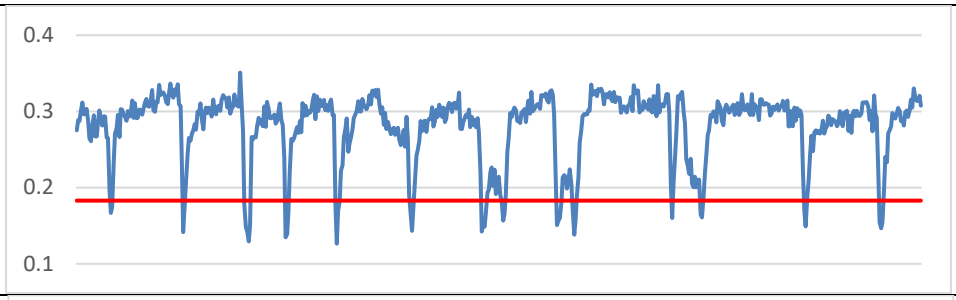
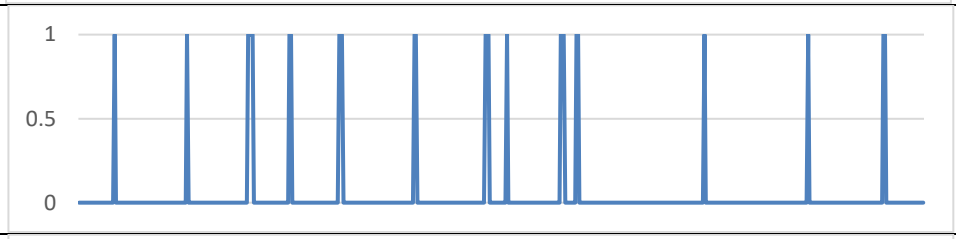
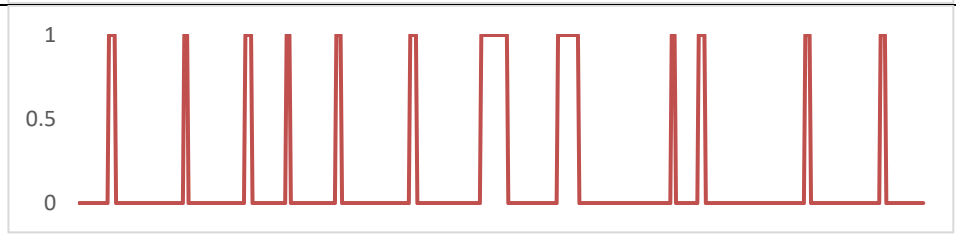
Therefore, we combine both advantages of both method by calculating the lower boundary value from Z-score method then using that value at the fixed thresholding value that can suit for individual detection. The following table shows the results of hybrid method compared to the previous results of EAR 0.2 fix thresholding value method and Z-score peak detection method in 3 states of fatigue of driver.

Table 4.18 The blinking signal detection in alert state from hybrid method

Dataset	Example of blinking signal in alert state
Hybrid signal	
EAR 0.2 Blink (7)	
Ground truth (8)	
Result	EAR thresholding: True positive detect: 8 / Fasle positive detect: 7
	Z-score: True positive detect: 8 / Fasle positive detect: 3
	Hybrid: True positive detect: 6 / Fasle positive detect: 1 Fasle negative detect: 1

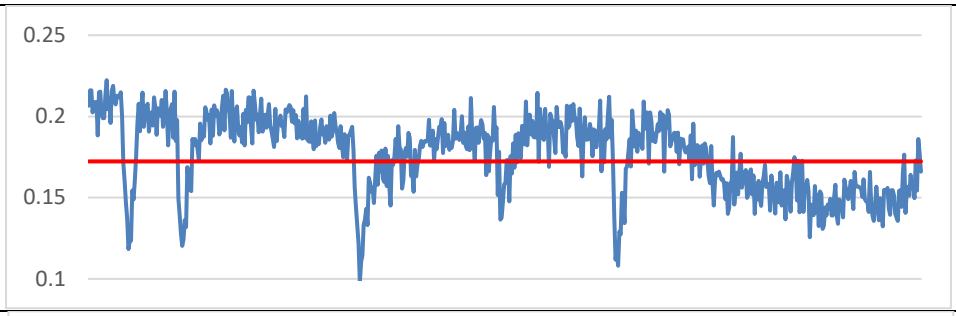
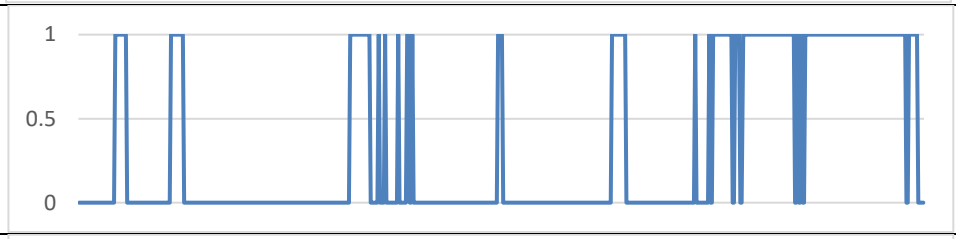
From T*able* 4.18, the new method can decrease the false positive event that can detect blinking more accurate but from the third blink that this method missing detection because the value below the new thresholding value (0.152) in this example is only 1 frame that mean this method difficult to detect very shot blinking event.

Table 4.19 The blinking signal detection in drowsy state from hybrid method

Dataset	Example of blinking signal in drowsy state
Hybrid signal	
Hybrid Blink (13)	
Ground truth (12)	
Result	EAR thresholding: True detect: 11 / False positive: 3 / False negative: 1
	Z-score: True detect: 12 / False positive: 5
	Hybrid: True detect: 11 / False positive: 2 / False negative:1

From Table 4.19, this hybrid method can also decrease the false positive events like in alert state. The false negative event is the same reason from previous table because the very short blinking is not suitable for this method. The W shape signal from this example cannot be detected correctly by this method.

Table 4.20 The blinking signal detection in microsleep from hybrid method

Dataset	Example of blinking signal in microsleep state
Hybrid signal	
Hybrid Blink (19)	
Ground truth (6)	
Result	EAR thresholding: True detect: 6 / False positive: 55
	Z-score: True detect: 5 / False positive: 9 / False negative: 1
	Hybrid: True detect: 6 / False positive: 13

From Table 4.20, this method can detect long blinking event in sixth blink but have some false positive events that divide the one long blinking event into two long blinks. If the driver falls into microsleep state, this method can certainly detect the long blinking events.

From driver behaviors information in 4.1.2, we know that blink rate is difficult to classify the fatigue level of driver that why we are not concern in blinking event so much but the blink duration is the most significant factor that can divide the fatigue status that why we need to find the best method to detect the long blinking event as much as possible. The hybrid method can do better in various fatigue states, can decrease false positive events, and can detect long blinking events that combine two advantages of EAR thresholding and Z-score peak detection method. Therefore, the

blink detection using EAR value in this research focuses on blink duration by using fixed thresholding value from Z-score.

4.3.4.3 Yawning using Respect Area Method

Mouth events can use respect area calculation like blinking detection by using eye aspect area. The mouth events are difficult to define what is driver doing because when driving the eye need to focus on the road that the eye area should full open as much as possible, but the mouth events are various because the driver can do whatever they want such as talking, singing, eating, drinking, or yawning. From our dataset, when the driver wants to yawn, they use their hand cover their mouth by instinct. We have all mouth events that are usable to define the various event from this dataset. There are two kinds of data, not yawning event and yawning event.

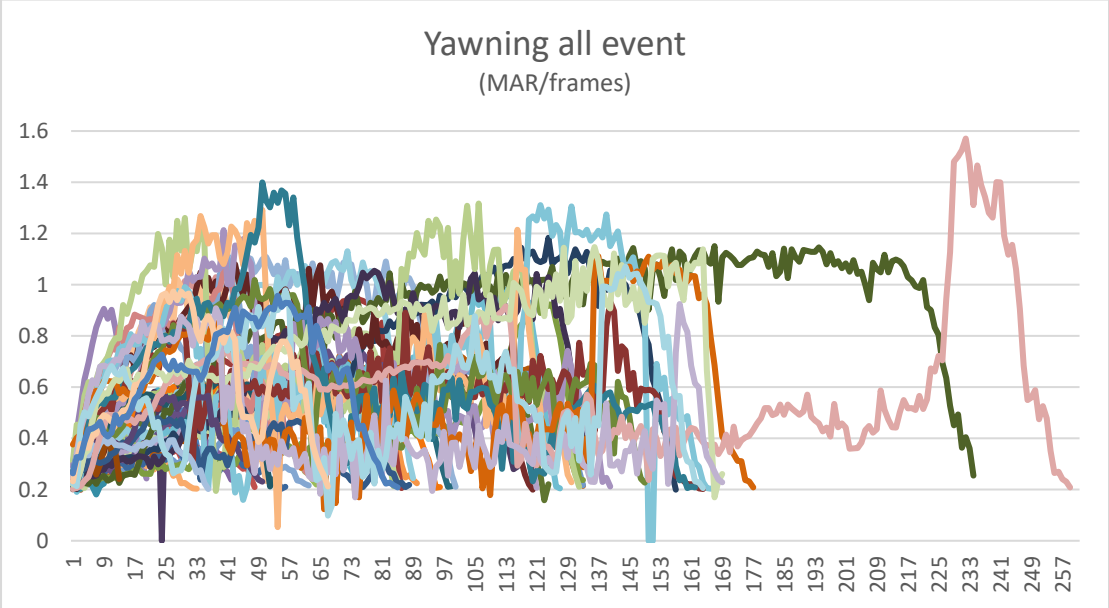


Figure 4.29 MAR value of yawning events in all mouth-opening event

From the yawning detection, we found three kinds of yawing. First is yawning without cover in front of mouth. Second is yawing with some cover in front of mouth. And third is light yawning, the yawning event that body nature need more oxygen when get fatigue.

Table 4.21 The MAR value in all yawning events

Mouth Events	Min MAR value		Avg MAR value		Max MAR value		Time duration (second)		Frames duration	
	avg	min	avg	min	avg	min	avg	min	avg	min
Non-cover mouth	0.201	0.158	0.585	0.350	0.838	1.183	4.782	2.306	101	50
Mouth cover	0.163	0.053	0.644	0.459	1.181	1.570	7.33	2.838	157	62
Light yawning	0.212	0.181	0.359	0.212	0.450	0.906	1.65	0.416	35	9

From Table 4.21, light yawning event can classify from other yawning events by time duration. Normal yawning between non-cover and cover mouth events can classify by MAR value because non-cover mouth can detect mouth clearly that make the value higher. In the other open-mouth events; in this experiment have classified into three events: talking, non-talking, and object cover mouth event. The object cover mouth is missing detection events by having some object cover mouth that make mouth facial detection missing position then the MAR value raise up. The following table is MAR value of all mouth-opening events.

Table 4.22 The MAR value in all mouth-opening not yawning events

Mouth Events	Min MAR value		Avg MAR value		Max MAR value		Time duration (second)		Frames duration	
	avg	min	avg	min	avg	min	avg	min	avg	min
Talking	0.108	0	0.216	0.128	0.314	0.210	2.298	0.049	29	2
Non talking	0.175	0	0.224	0.107	0.273	0.201	1.251	0.044	29	2
Mouth cover	0.111	0	0.220	0.120	0.346	0.210	0.901	0.043	20	2

From Table 4.22, this information is for one round of mouth-opening event that time and frame duration is not so long for talking event. Not yawning events is difficult to classify because the value is look similar but it not necessarily for fatigue detection because there is a huge gap of value from yawning events.

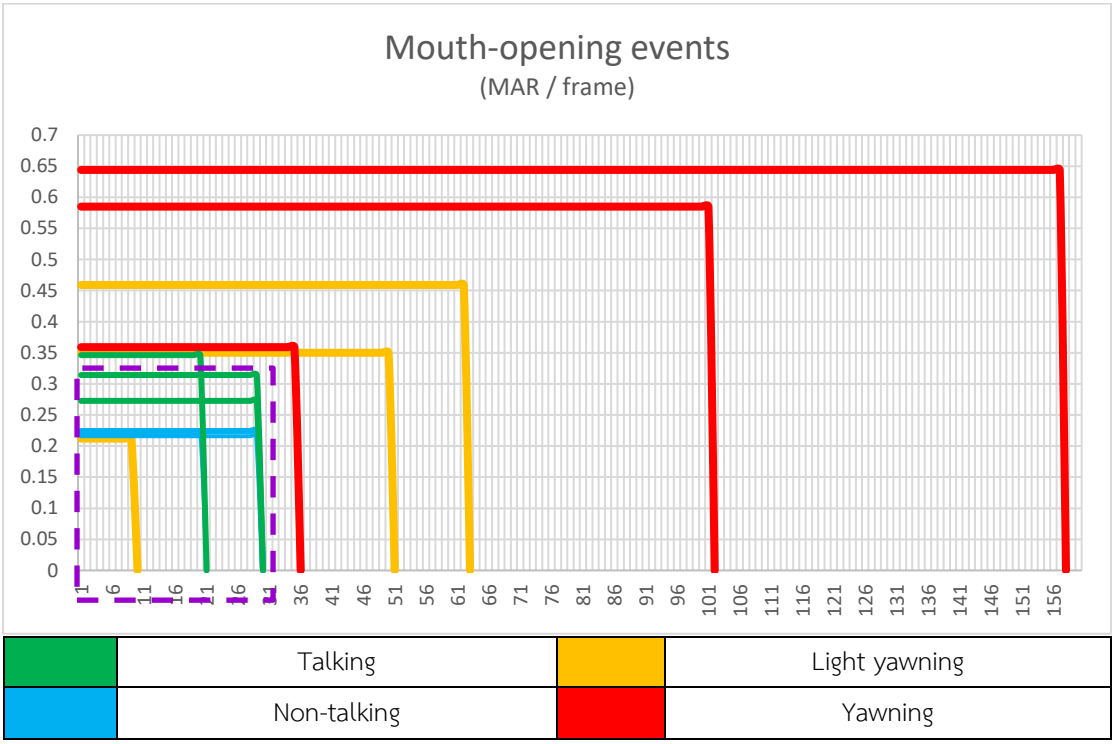


Figure 4.30 All mouth opening events in one round mouth-opening

From Figure 4.30, the purple dash box area is contained all non-yawning events that can be ease to thresholding between yawning event and others. Therefore, using MAR value can detect the yawning event by MAR value and frame duration thresholding. The recommended value for yawning event is MAR value more than 0.35 for 30 consecutive frames.

For drowsiness detection, yawning is one of the signs that fatigue occurred to the driver but not the most significant especially if driver have worn mask or cover mouth every time, they have yawn not like eyes that driver need to stay focus on the road that cannot hide from the camera.

4.4 Final code

From facial features to symptom extraction, in this research try to using any facial features then convert to value to calculate the driver status. Head extraction is not accurate for calculating the head angle. Mouth extraction can get MAR value to detect mouth events and can classify yawning events separate from talking but difficult for various situation such as driver have worn face mask due to COVID-19 situation

moreover in our dataset, when driver need to yawn, there cover their mouth automatically by hand that make yawning detection is not the first choice to detect driver drowsiness. Eye extraction can get EAR that can tell eye status between open eye and closing eye. In this research have tried two methods to detect eye blinking events that finally get the best method even have some false positive event in detect blink but can detect long blinking event that most necessary for fatigue detection. Blinking detection in terms of blink duration is the important role in this research final algorithm.

The final algorithm has two functions for driver monitoring. First is alarm to warn the driver immediately when there is too long blink duration that dangerous for driving and that can identify the driver get into microsleep state. Second is to predict the driver status every minute by using blink rate and blink duration events. Refer from Figure 4.2, the blink duration can divide fatigue level. If the algorithm calculates driver status and found that there fall into drowsy state continuously for 5 minutes, the algorithm can warn driver to take a brake because refer to Figure 4.3, the fatigue level can increase every 5 minutes of driving.

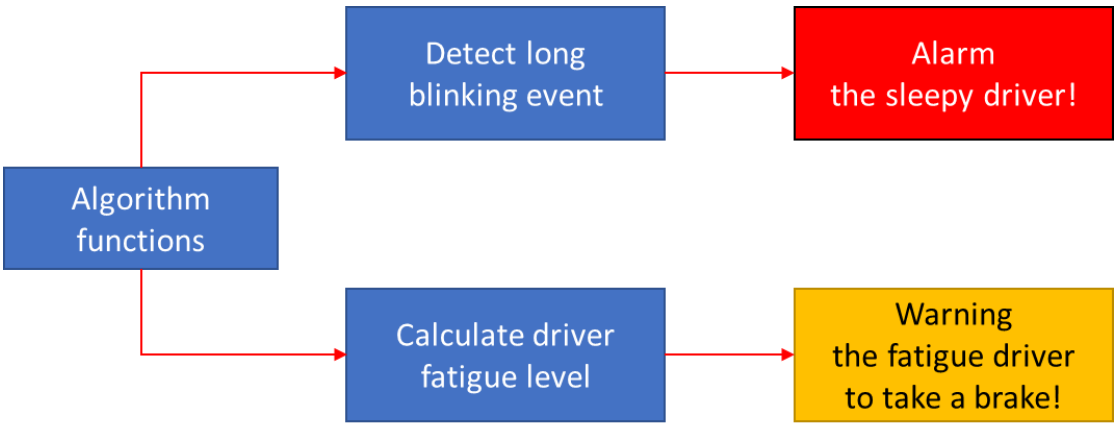


Figure 4.31 Final algorithm function of this research

After many experiments in this research, the following flowchart is the final version have improved from Figure 3.18.

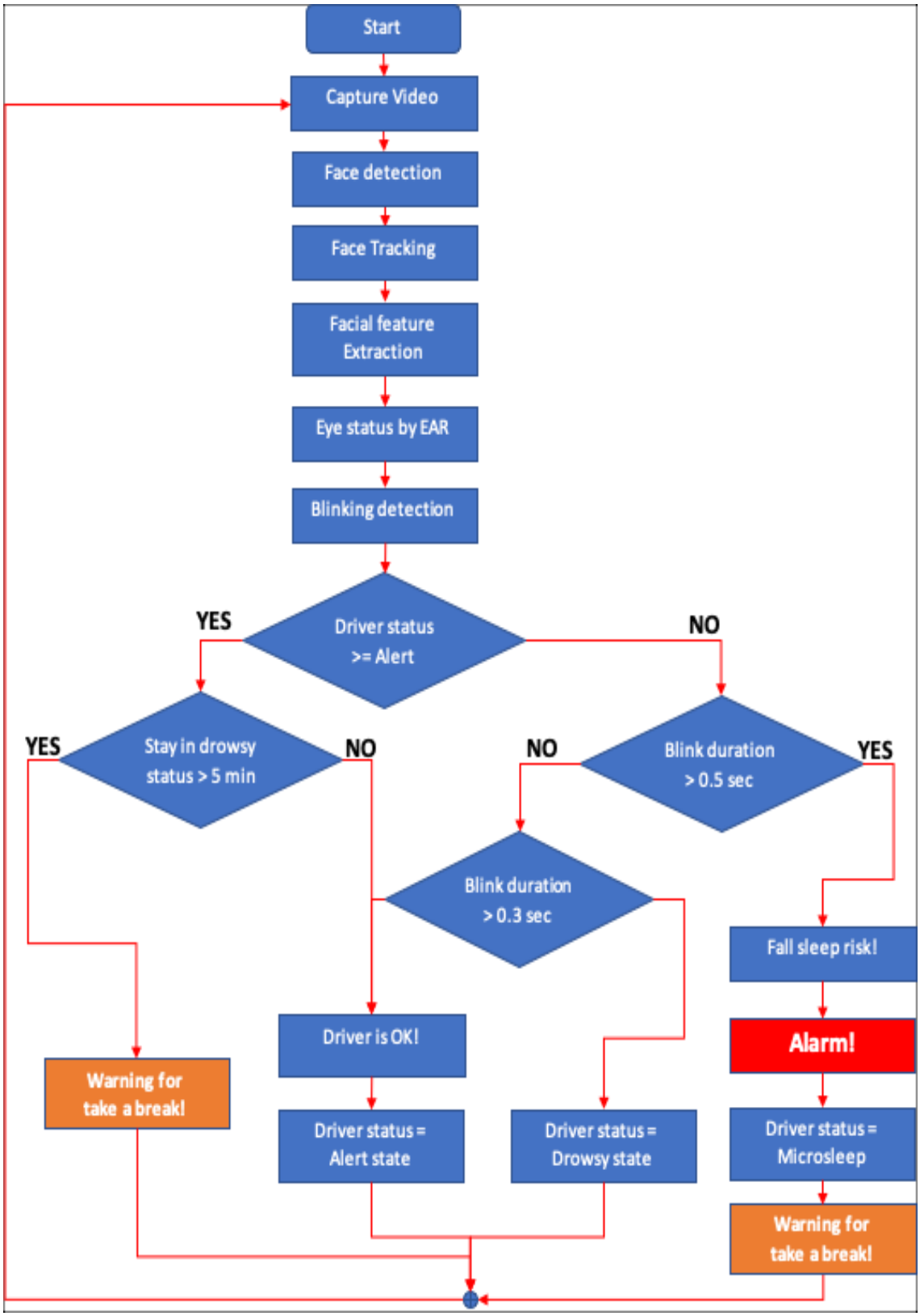
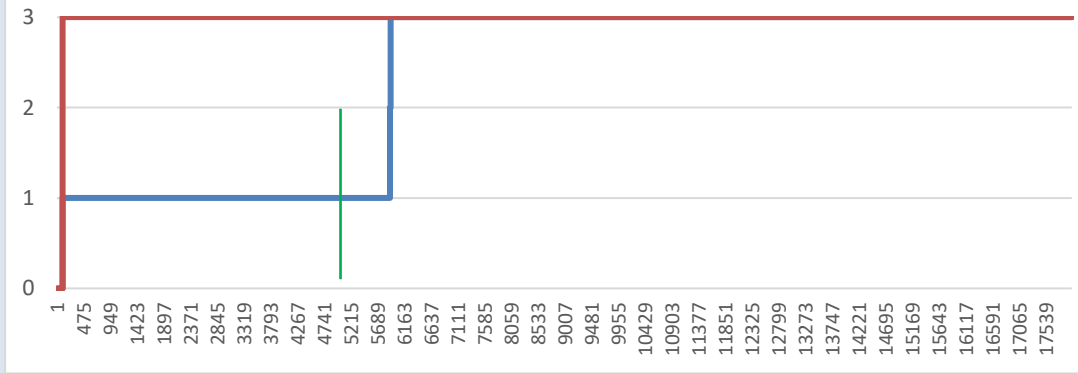


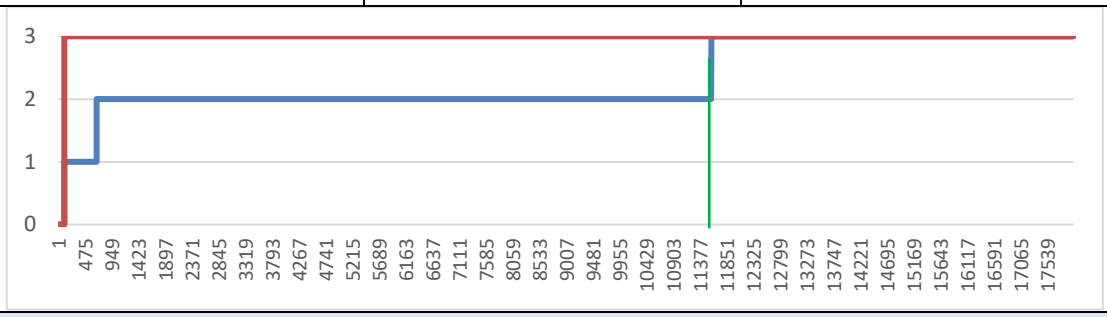
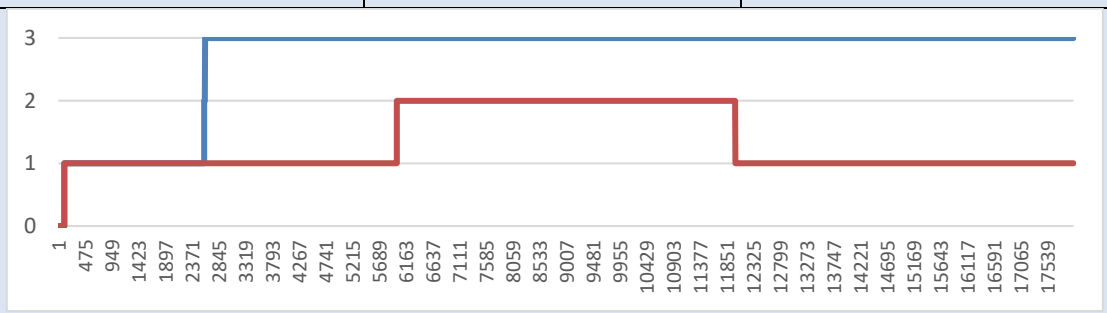
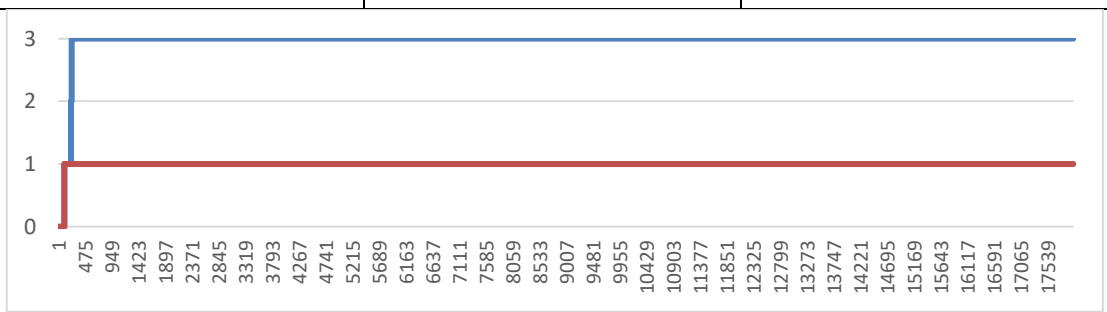
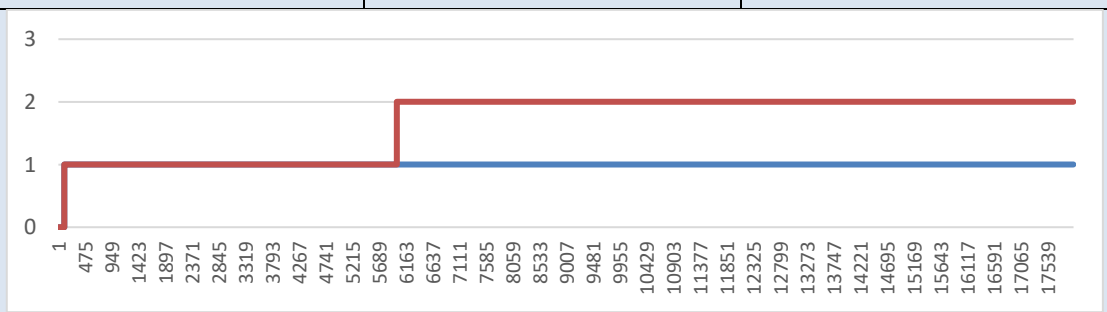
Figure 4.32 Final version of this research algorithm flowchart

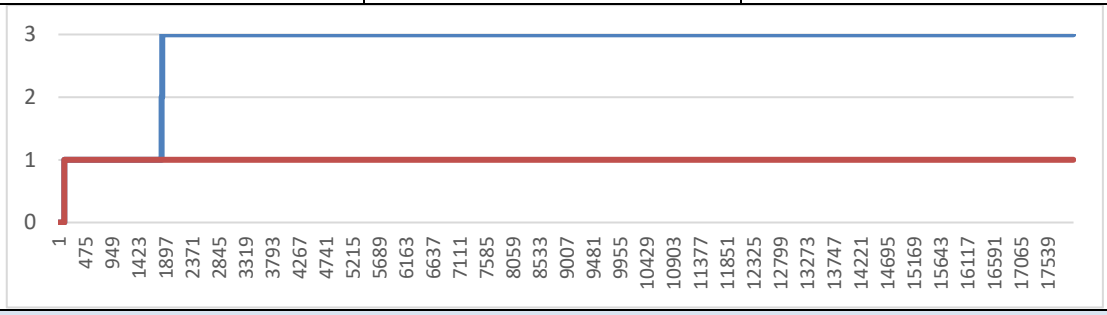
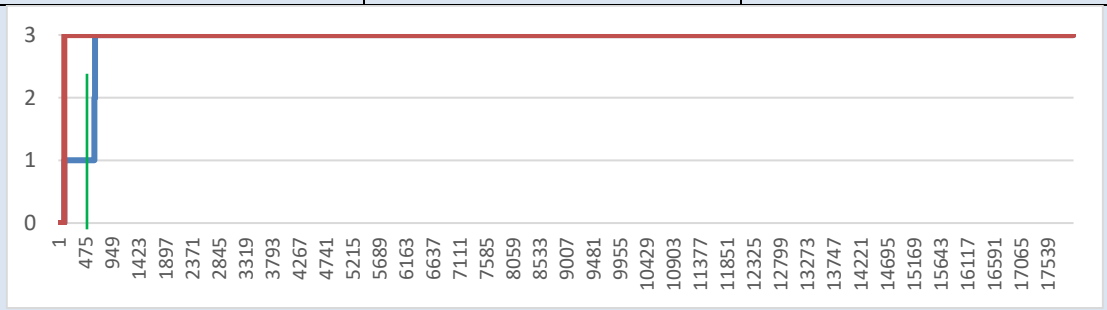
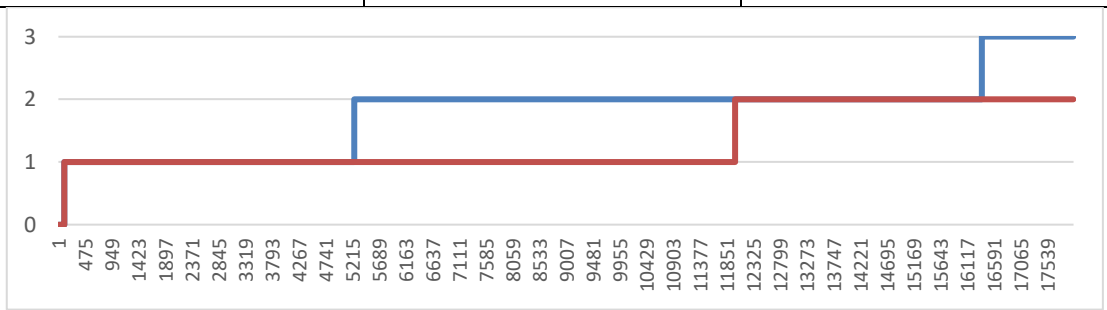
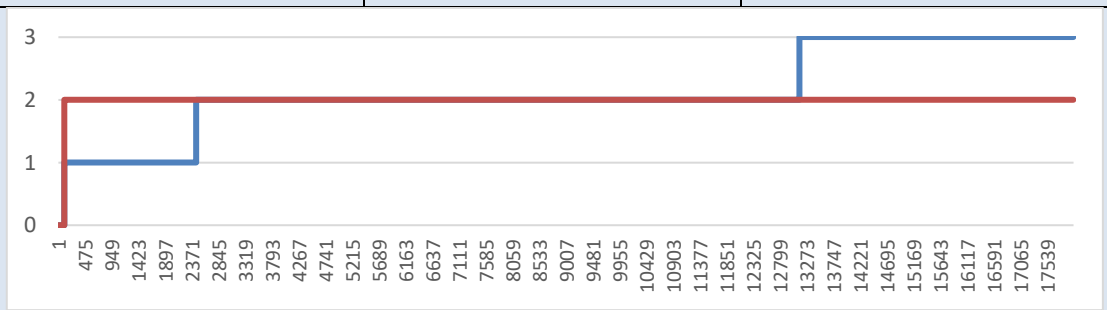
4.5 Validation

For validate the accuracy of the drowsiness monitoring from this research, validation by using the 10 videos dataset that contain same fatigue level and different fatigue level within 10 minutes duration. The result is the driver status in each minute. If which video has a drowsy event, the algorithm should alarm the driver. If which video have stay in drowsy fatigue for more than 5 minutes, the algorithm should display the warning message to driver. The following table contains signal of fatigue level by 0 = unknown, 1 = alert state, 2 = drowsy state, and 3 = microsleep state. Each driver has answered their own fatigue level in KSS level so convert to B-ORDS.

Table 4.23 Final code validation result

Final code state signal	Ground truth state signal	Fall sleep event
Dataset: Drivingsim#1 (30 – 40 minute) / FPS = 17.57		
KSS = lv 5 (Alert)	KSS = lv 5 (Alert)	KSS = lv 6 (Drowsy)
		
Dataset: Drivingsim#2 (20 – 30 minute) / FPS = 17.55		
KSS = lv 8 (Drowsy)	KSS = lv 9 (Microsleep)	KSS = lv 9 (Microsleep)
		

Dataset: Drivingsim#3 (30 – 40 minute) / FPS = 17.48		
KSS = lv 9 (Microsleep)	KSS = lv 8 (Microsleep)	KSS = lv 9 (Microsleep)
		
Dataset: Drivingsim#4 (20 – 30 minute) / FPS = 17.68		
KSS = lv 5 (Alert)	KSS = lv 6 (Drowsy)	KSS = lv 5 (Alert)
		
Dataset: Drivingsim#5 (10 – 20 minute) / FPS = 17.15		
KSS = lv 4 (Alert)	KSS = lv 4 (Alert)	KSS = lv 4 (Alert)
		
Dataset: Drivingsim#6 (20 – 30 minute) / FPS = 16.98		
KSS = lv 5 (Alert)	KSS = lv 6 (Drowsy)	KSS = lv 7 (Drowsy)
		

Dataset: Drivingsim#7 (10 – 20 minute) / FPS = 16.62		
KSS = lv 5 (Alert)	KSS = lv 5 (Alert)	KSS = lv 5 (Alert)
		
Dataset: Drivingsim#8 (35 – 45 minute) / FPS = 16.95		
KSS = lv 9 (Microsleep)	KSS = lv 9 (Microsleep)	KSS = lv 9 (Microsleep)
		
Dataset: Drivingsim#9 (10 – 20 minute) / FPS = 17.64		
KSS = lv 5 (Alert)	KSS = lv 5 (Alert)	KSS = lv 6 (Drowsy)
		
Dataset: Drivingsim#10 (30 – 40 minute) / FPS = 17.55		
KSS = lv 7 (Drowsy)	KSS = lv 7 (Drowsy)	KSS = lv 7 (Drowsy)
		

From Table 4.23, the final code can display the driver fatigue level that calculate on eye blinking event-based compared to questionnaire KSS fatigue level. The results of the two methods were slightly consistent, perhaps the drivers didn't think they were too sleepy when they closed their eyes for more than 0.3 seconds as a result, there didn't reply that he was in a KSS above level 5. But the incident with falling asleep, the algorithm can result in entering a microsleep state. When processed on Raspberry Pi 4, it was found that this algorithm can be processed with framerate about 17 FPS. This is considered as less processing power as using the normal HAAR Cascade face detection algorithm refers to Figure 4.4, that be able to process in near real-time.

This algorithm may not be able to accurately predict driver fatigue in all conditions, but at least it can help warn drivers when they fall asleep and remind drivers to take a break.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

This research aims to develop a visual-based fatigue detection device. The goal is to develop a device that does not interfere with the driver's driving, small size, easy to install, and can be used in real-time. It was intended to be developed to be used on Raspberry Pi 4 because it is a minicomputer that is small and has quite good processing performance compared to other minicomputers. There are 2 main subjects to be studied: first, how to develop algorithms to process quickly and accurately, which is the heart of algorithm development that works on a not powerful minicomputer. Second is a study on fatigue, because each person has different fatigue and the definition of fatigue itself is unclear. To develop an algorithm for detecting fatigue, researchers should understand fatigue characteristics, parameters that can indicate fatigue and the degree of fatigue preceding.

Fatigue levels are divided into 10 levels according to the KSS level, but for driving we can simply classify fatigue into 3 levels according to the B-ORS level. In this experiment, courtesy of MTEC, they were allowed to record video of the actual driver on the vehicle. Makes us know the general nature of driving that the distance from the driver to the front console is about how much distance, camera mounting position, driving style of each driver, and light condition in the cabin while driving. The use of driving simulators for collecting long-distance driving data has also been courtesy. The route can be simulated as a long straight highway to increase the chances that the driver is easily tired and sleepy. The collected datasets for both sections, as well as the UTARLDD and DROZY fatigue study datasets, will be used to determine key parameters that can indicate the driver's fatigue level so that the values obtained are ethnically diverse.

The study found that blink duration can clearly separate groups of drivers according to their fatigue level. The driver in the Alert level had a very fast blink time of less than 0.2 seconds. Drivers in the drowsy level will begin blinking slower, starting from 0.15 seconds to 0.31 seconds per blink. Drivers who initiated the microsleep level

had the slowest blink per second, averaging 0.42 seconds per blink, and the slowest at more than 1 second per blink which is considered very dangerous for driving. In terms of the blink rate, although it was found that drivers who entered the microsleep level blinked the most frequently, the number of blinks was not much different from other fatigue groups. Whether to blink frequently or not is an individual matter for some drivers. Some drivers are alert, but blink frequently cause this parameter is not used.

Development of a fatigue detection algorithm will be divided into two sections. The first is to develop an algorithm for detecting the desired object, then follow by developing of algorithm for calculating fatigue. As for object detection developments, it was found that the HAAR cascade was the fastest to process, but every frame in succession detection had some frames that the detection was wrong. Therefore, it is further improved by using face tracking to help memorize the correct position by analyzing the characteristics of the correct frame. The answer was that the BB frame size and the confidence value should be used to help filter before detecting to make the detection more accurate. In addition, using face tracking also helps to increase the frame rate as well.

After detecting the desired position, the next step is to detect the facial landmarks to be calculated as numbers and then can be used to predict the status of the driver. Head rotation detection was found to be unable to tell the direction exactly as it was and slowed down the framerate. Capturing the size of the eyes to bring a numerical value to indicate the state of opening and closing the eyes can be done well and does not consume processing power. Capturing the size of the mouth, which uses the same technique as the eyes, works fine and consumes a little more frame rate, so it's still considered interesting. But the value obtained from the calculation of those areas found that sometimes it is so little that it looks no different while opening or closing eyes and closing or opening mouth. Therefore, various methods were tried to make the signal clearer. The first experiment is to try processing on multiple image sizes, with the result that the larger image size actually helps the signal to be clearer, but also causes the frame rate to decrease as well. Therefore, tried a method that enlarges the image only when detecting the driver's face and found that it can really help the

signal to be clearer without reducing the frame rate as well. Lighting conditions are another factor that determines whether an object can be detected clearly or not. Therefore, several image enhancement methods have been tried as suggested and found that OpenCV's cv2.equalHist tool works well and automatically image adjust therefore, has chosen to use this method as the main. And found that using cv2.CLAHE, a technique that makes the object stand out from the background, further enhances the signal pattern even though the frame rate decreases slightly. In this research, we choose to use together.

After finding the right way to improve the image and signal quality. Then, the calculated values were applied to the available datasets to determine the relationship between fatigue and EAR values. It was found that the EAR value of eye opening was different for each fatigue level and the EAR value decreased with increasing fatigue level. The EAR value that split between eyes open and closed eyes was 0.2, consistent with most studies. Next, we tried to find ways to use the EAR value to detect blinking, with 2 methods. The first method is to fix the EAR value at 0.2, with the lower value being all eyes closed event. And the second way is to use the Z-score to calculate the bottom edge of the data and then detect the peak value of the bottom edge, which is considered closed eyes. Both methods have their advantages and disadvantages, with the Z-score detecting normal blinking better, but it can't detect long blinks that are at risk of falling asleep. Fixing a value of 0.2 is more likely to detect a normal blinking error, but it can detect prolonged eye closure, which can help warn the driver against falling asleep. Therefore, the advantages of both methods are combined by calculating the EAR fix value from the Z-score to obtain the appropriate value for each individual. The experimental results showed that normal blink detection was good, less mistakes, and importantly, it can better detect long blinking events, which are the most important events for driver monitoring.

The final algorithms at least can help warn drivers when they fall asleep and remind drivers to take a break in near real-time that can make drive safer.

REFERENCES

[1] สำนักงานนโยบายและแผนการขนส่งจราจร สำนักแผนความปลอดภัย, “รายงานการวิเคราะห์สถานการณ์อุบัติเหตุทางถนนของกระทรวงคมนาคม พ.ศ.2562,” 2019.

[2] กรมการขนส่งทางบก, “รายงานฉบับสมบูรณ์ โครงการศึกษาและวิเคราะห์ปัญหาอุบัติเหตุที่เกิดจากรถบรรทุกขนาดใหญ่ในประเทศไทย,” *Pneuma*, vol. 12, no. 1, pp. 97–115, 2008, doi: 10.1163/157007490x00142.

[3] ร. ด. ภ. บุตรอินทร์, “ประเด็นท้าทายทางด้านกฎหมายในยุค IOTs : ศึกษากรณีความรับผิดจากการใช้งาน Smart Car (Legal Challenges in the IOTs : Case study of Smart Car’s Liability),” 2020.

[4] S. K. L. Lal and A. Craig, “A critical review of the psychophysiology of driver fatigue,” *Biol. Psychol.*, vol. 55, no. 3, pp. 173–194, 2001, doi: 10.1016/S0301-0511(00)00085-5.

[5] W. Vanlaar, H. Simpson, D. Mayhew, and R. Robertson, “Fatigued and drowsy driving: A survey of attitudes, opinions and behaviors,” *J. Safety Res.*, vol. 39, no. 3, pp. 303–309, 2008, doi: 10.1016/j.jsr.2007.12.007.

[6] A. Eskandarian, R. Sayed, P. Delaigue, A. Mortazavi, and J. Blum, “Advanced Driver Fatigue Research,” vol. 07, no. April, p. 211, 2007.

[7] J. Van Den Berg, “Sleepiness and head movements,” *Ind. Health*, vol. 44, no. 4, pp. 564–576, 2006, doi: 10.2486/indhealth.44.564.

[8] T. Danisman *et al.*, “Drowsy Driver Detection System Using Eye Blink Patterns,” 2018.

[9] A. Anund, C. Fors, D. Hallvig, T. Åkerstedt, and G. Kecklund, “Observer Rated Sleepiness and Real Road Driving: An Explorative Study,” *PLoS One*, vol. 8, no. 5, 2013, doi: 10.1371/journal.pone.0064782.

[10] A. Chowdhury, R. Shankaran, M. Kavakli, and M. M. Haque, “Sensor Applications and Physiological Features in Drivers’ Drowsiness Detection: A Review,” *IEEE Sens. J.*, vol. 18, no. 8, pp. 3055–3067, 2018, doi: 10.1109/JSEN.2018.2807245.

[11] M. H. Sigari, M. Fathy, and M. Soryani, “A Review on Driver Face Monitoring System for Fatigue and Distraction Detection,” *Int. J. Veh. Technol.*, vol. 2013, pp. 73–100, 2013, doi: 10.1155/2013/263983.

[12] M. Ngxande, J. R. Tapamo, and M. Burke, “Driver drowsiness detection using Behavioral measures and machine learning techniques: A review of state-of-art

techniques," *2017 Pattern Recognit. Assoc. South Africa Robot. Mechatronics Int. Conf. PRASA-RobMech 2017*, vol. 2018-Janua, pp. 156–161, 2017, doi: 10.1109/RoboMech.2017.8261140.

[13] D. Sandberg, A. Anund, C. Fors, G. Kecklund, J. G. Karlsson, and M. Wahde, "The Characteristics of Sleepiness During Real Driving at Night — A Study of Driving Performance , Physiology and Subjective Experience," no. Stage 1, doi: 10.5665/sleep.1270.

[14] J. Jiménez-Pinto and M. Torres-Torriti, "Optical flow and driver's kinematics analysis for state of alert sensing," *Sensors (Switzerland)*, vol. 13, no. 4, pp. 4225–4257, 2013, doi: 10.3390/s130404225.

[15] Z. Orman, A. Battal, and Erdem Kemer, "A Study On Face, Eye Detection And Gaze Estimation," *Int. J. Comput. Sci. Eng. Surv.*, vol. 2, no. 3, pp. 29–46, 2011, doi: 10.5121/ijcses.2011.2303.

[16] W. Koehrsen, "Face Detection For Beginners," *Medium*. pp. 1–9, 2018, Accessed: May 15, 2021. [Online]. Available: https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9.

[17] K. Goyal, K. Agarwal, and R. Kumar, "Face Detection and Tracking: Using OpenCV," *Proc. Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2017*, vol. 2017-Janua, no. April 2017, pp. 474–478, 2017, doi: 10.1109/ICECA.2017.8203730.

[18] A. Sadiq, A. Rabhi, S. Faculty, and A. Mouloudi, "Face Tracking: state of the art," no. November, 2015, doi: 10.1109/ICoCS.2015.7483308.

[19] D. Le and X. Wu, "FACE DETECTION, TRACKING, AND RECOGNITION FOR BROADCAST VIDEO," pp. 1–14, 2008, [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.333.2754&rep=rep1&type=pdf.

[20] S. Asteriadis, N. Nikolaidis, and I. Pitas, "Facial Feature Detection Algorithms: A Review," *Adv. Face Image Anal. Tech. Technol.*, no. January, pp. 42–61, 2010, doi: 10.4018/978-1-61520-991-0.ch003.

[21] Y. Wu and Q. Ji, "Facial Landmark Detection: A Literature Survey," *Int. J. Comput. Vis.*, vol. 127, no. 2, pp. 115–142, 2019, doi: 10.1007/s11263-018-1097-z.

[22] A. Saeed, A. Al-Hamadi, and A. Ghoneim, "Head pose estimation on top of haar-like face detection: A study using the Kinect sensor," *Sensors (Switzerland)*, vol.

15, no. 9, pp. 20945–20966, 2015, doi: 10.3390/s150920945.

[23] Irfan Alghani Khalid, “Head Pose Estimation using Python,” *Towardsdatascience*, 2021. https://towardsdatascience.com/head-pose-estimation-using-python-d165d3541600#:~:text=Head pose estimation is one,to the road or not.

[24] R. (MagPi) Zwetsloot, “Raspberry Pi 4 specs and benchmarks — The MagPi magazine,” *MagPi*, 2018. https://magpi.raspberrypi.org/articles/raspberry-pi-4-specs-benchmarks%0Ahttps://magpi.raspberrypi.org/articles/raspberry-pi-3bplus-specs-benchmarks.

[25] Alasdair Allan, “Introducing the Raspberry Pi Cameras,” *Raspberry Pi Documentation*, 2021. https://www.raspberrypi.com/documentation/accessories/camera.html#camera-modules.

[26] JolleJolles and K. Green, “Working with USB webcams on your Raspberry Pi,” 2021. https://raspberrypi-guide.github.io/electronics/using-usb-webcams.

[27] J. Geerling, “microSD Card Benchmarks,” *Http://Pidramble.Com*, 2020. https://www.pidramble.com/wiki/benchmarks/microsd-cards.

[28] R. Ghoddoosian, M. Galib, and V. Athitsos, “A realistic dataset and baseline temporal model for early drowsiness detection,” *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2019-June, pp. 178–187, 2019, doi: 10.1109/CVPRW.2019.00027.

[29] T. Soukupová and J. Cech, “Eye Blink Detection Using Facial Landmarks,” *Res. Reports C.*, pp. 1–8, 2016.

APPENDIX A

Table 1 MTEC Driving Simulator Specification

Simulator Component	*Hardware*
Computer	*Asus GL12CX-TH002T* *Intel Core i9-9900K, 16GB DDR4* *Nvidia Geforce RTX2080*
Monitor	*Samsung CJ890* *Ultra-wide curved monitor 49” 144Hz*
Steering & pedal	*Thrustmaster T300RS GT edition* *28cm Ø steering wheel 1080° force feedback,* *Accelerator pedal, Clutch pedal,* *Brake pedal with progressive resistance*
Driver Seat	*Mazda 3 (BK) Driver Adjustable Seat*

Figure 1 MTEC driving simulator

APPENDIX B

講演番号 212

文献番号 20195212

Study of a Visual Driver Drowsiness Monitoring Algorithms Based on Raspberry Pi 3

Saengpenchai Manussanan [1] **Srisurangkul Chadchai** [2]

1) International College, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520 Thailand (E-mail: ic@kmitl.ac.th)
2) National Metal and Materials Technology Center, Pathum Thani, 12120 Thailand

ABSTRACT: Nowadays, one of the main causes of car accidents is drowsy driving. This study developed a visual Driver Drowsiness Monitoring device to verify driver's concentration while driving. Raspberry Pi 3 is used as the processor run by Python language. However, the drawback of Raspberry Pi is an insufficiency of RAM. Thus, only suitable algorithms will make the system work well. This study developed the system by comparing two face detection algorithms between Haar-Cascade and Multi-task Cascaded Convolutional Networks (MTCNN) based on accuracy and processing speed in real-time conditions. Then, the algorithm which provides better result will be chosen.

KEY WORDS:**drowsiness detection, computer vision, machine learning** Human Engineering (C2)

1. INTRODUCTION

Thailand is one of the top countries with the highest road traffic death rate. One of the main accidents caused by drowsy driving. [1]. These types of accidents often occur with public buses and minivans traveling across provinces, especially during the long holiday season in which people travel very much. Even in the past, the government enforced laws to reduce the risk of accidents, there is still news reporting the accident caused by the driver's sleep every year.

A precise and scientific definition for fatigue has not been presented yet. [2]. However, the most popular way is based on eye region detection. The main reason is the main symptoms of fatigue and distraction appear in the eyes. At present, the fatigue detection cameras are not widespread and are still expensive because manufacturers have to design circuit boards and hardware themselves.

This project is developing a real-time video camera fatigue detection system based on a common hardware platform. This device is developed to be small in order to be easily installed in many kinds of vehicles. The algorithms can work in real-time conditions, this study tries to deploy two face detection algorithms between Haar Feature-based Cascade and Multi-task Cascaded Convolutional Neural Networks (MTCNN) on the same system. Then, compare their accuracy and the processing speed.

2. METHODOLOGY

2.1. The principles of driver fatigue monitoring

First of all, when the device power up, the camera starts recording video. The processing unit then starts analyzing to find faces in the video, which is called face detection function. After confirming the position of the face, facial landmark localization will begin to locate facial components such as eyes, nose, mouth, and chin. This procedure is used to locate the eyes, as known as the eye detection process. When eyes are found, the system will check wheather it gets any fatigue sign from the eyes or not. If the system finds the vigilance level of the driver is lower than a standard value, the device will alarm to warn the driver until the system finds vigilance level of the driver back to normal stage. (see Fig. 1)

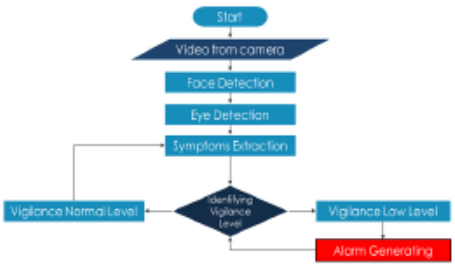


Fig. 1 General flow chart of driver fatigue monitoring

2.2. Face detection

In most of the driver face monitoring systems, face detection is the first part of image processing operations. Face detection is considered as very important part of the system due to the difficulty of eye detection. There are various mentioned problems related to face detection including, there are: 1) position and orientation of the face, 2) lighting conditions, 3) presence of glasses, makeup, beards, mustache, and 4) real-time processing. Good algorithm can deal with these problems except lighting conditions that due to the hardware problem.

In this project, two face detection algorithms were studied to compare the pros and cons in Raspberry Pi3.

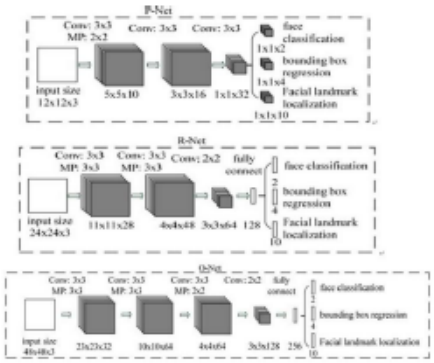
2.2.1. Haar Feature-based Cascade

This technique was introduced in 2001 by Paul Viola and Michael Jones.[3]. This technique is based on Machine Learning process. Haar Cascades Classifier was trained through image analysis by Haar-like Features and Adaboost. The image dataset is divided into two groups. First is positive image, which is wanted target image to be detected (Face photos). And, the second is negative image, which is not the target image (Non-face photos). In Raspberry Pi, this algorithm is available on OpenCV (Open Source Computer Vision) library for free with both pre-trained model and cascade training by your own. It is not complex computational, more accuracy and faster in real-time processing.

2.2.2. Multi-task Cascaded Convolutional Neural Networks

The MTCNN model comes with high accuracy achieved with a deep neural network. [4]. There are 3 networks: The P-Net, the R-Net, and the O-Net. (see Fig. 2) Each multiple layers allows for higher precision, as each network can fine-tune the results of the previous one. In addition, this model employs an image pyramid to find both large and small face. Even though this may produce an overwhelming amount of data. This algorithm needs to install OpenCV, Tensorflow or Caffe.

Fig. 2 MTCNN Structure



2.3. Eye detection

To detect the eyes, the location of the eyes is needed to be known first. The Facial Landmarks Detector will be used to find the location of face components such as eyebrows, nose, mouth, chin and eyes.[5]. The pre-trained facial landmark detector is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face. (see in Fig. 3) For less computational in the system, the facial landmarks localization algorithm will be extracted only in the eye regions. This algorithm is available on Dlib library.

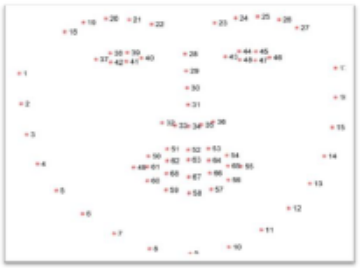


Fig. 3 The 68 Facial landmark coordinates from iBUG300-W [6]

2.4. Symptoms extraction

After eyes detection, the system needs some way to checks the vigilance of the driver. From the Fig.2, the dots can be drawn together to find the area of the eye. In this system refer to Adrian's work [7], the eye blink speed and the eye aspect ratio used to validate the driver.

2.4.1. Eyeblink speed

Eyeblink speed is the time between opening and closing of the eyelid during one blink event. Referring to Blinking Suppresses the Neural Response to Unchanging Retinal Stimulation [8] and Average duration of a single eye blink [9], humans take one blink about 100-400 milliseconds. If eye blink rate is more than 1000 milliseconds these can be assumed that the person has entered the semi-sleep state.

2.4.2. Eye aspect ratio (EAR)

From based on the work by Soukupová and Čech (10), the equation to calculate the distances between the upper and lower eyelids is EAR.

$$EAR = (\|p_2 - p_6\| + \|p_3 - p_5\|)/(2\|p_1 - p_4\|)$$

Where p1 to p6 are facial landmark locations. (see in Fig. 4)

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator with two sets of vertical points.

Typically, EAR of humans during eye open is more than 0.23 and decrease to 0.00 when the eye is closed. This point can be used to determine the state of the eyes.

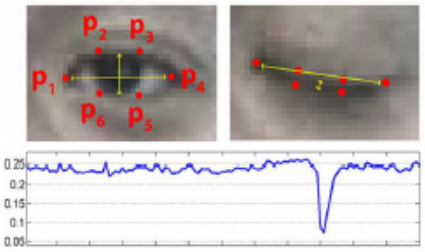


Fig. 4 A visualization of eye landmarks when then the eyes are open and close. Then plotting the eye aspect ratio over time.

3. EXPERIMENT

First, install the video recorder in the position where the actual device will be installed. The distance between the camcorder and the driver should be in the range of 40 centimeters and 70 centimeters. The position of the camcorder should be installed without any obstacle to make the good vision of the camera. (see in Fig. 5)

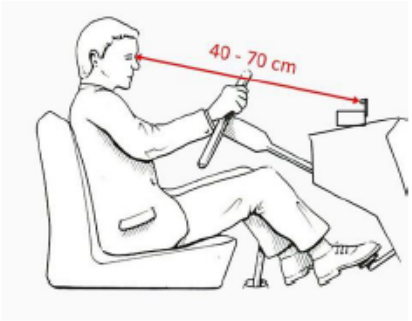


Fig. 5 The recommend distance between the driver and device.

Run the recoded videos from both condition, scenario setting and real-life conditions, with the trained algorithms on a Raspberry Pi 3 device. The scenario setting means driving in constant light condition and real-life condition means driving in light varying condition. The system of the device is similar, the difference is only two algorithms. In this study, everything is set in similar condition as much as possible.

Several events are set in the experiment for both condition such as looking straight forward, looking left, looking right, closing one eye, closing both eyes, and head nodding. (see Table 1) The accuracy is measured by the efficiency of the event detection in a period of time. For example, in 100 seconds if the system can detect the drivers behaviors in variety of event with no mistake, the accuracy can be estimated to be 100%.

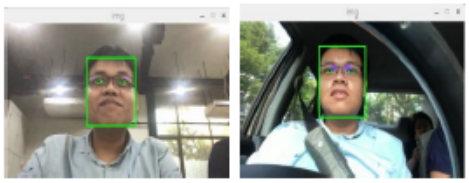


Fig. 6 Two conditions of this experiment between scenario and real-life conditions

4. RESULT

From the experiment, the data was collected from participants and then average the data to determine Detected rate and False alarm. Detected rate is calculated from the time of eye detectable time per full time of the events. False alarm is number of time that the alarm occurred without any drowsy events.

Table 1 The tests result from Haar-cascade detector

Conditions	Scenario		Real-life	
Events	Detected Rate	False Alarm	Detected Rate	False Alarm
Look straight	88.7%	3.25	93.5%	6.75
Look left & right	62.0%	2.50	43.6%	0.75
Closing eye <1sec	75.0%	0.50	89.4%	0.00
Closing eyes >1sec	78.0%	0.25	95.4%	0.25
Squint	78.1%	1.50	98.0%	0.50
Head nodding	70.9%	2.00	77.9%	0.50

Table 2 The device tests result from MTCNN detector

Conditions	Scenario		Real-life	
Events	Detected Rate	False Alarm	Detected Rate	False Alarm
Look straight	99.3%	-	98.5%	-
Look left & right	37.1%	-	40.3%	-
Closing eye <1sec	70.0%	-	100%	-
Closing eyes >1sec	96.3%	-	99.0%	-
Squint	100%	-	100%	-
Head nodding	96.3%	-	95.3%	-

Table 3 Comparison between two detector

Detector	Frame rate (FPS)		Accuracy	
	Scenario	Real-life	Scenario	Real-life
Haar	7.1325	6.8625	75.45%	82.96%
MTCNN	1.5975	1.7400	83.16%	88.85%

5. CONCLUSION

From the result, the Haar-cascade detector has much more frame rate than the MTCNN about 4.5 times. That's mean the haar-cascade has less delay and the system is more suitable to Raspberry Pi than the MTCNN. Besides, Haar-cascade can detect drowsinees faster than MTCNN, thats mean it is more safety for the driver. The slow frame rate makes the calculation of the EAR more slowly and can not handle the driver's symptoms at all. Therefore is the reason that the false alarm value of MTCNN cannot be verified according to Table 1.

About the detected rate, the good system should detect the driver's face in look-straight condition accurately because it is a normal driving gesture. Both algorithms can detect very well with high detected rate. The MTCNN is more accurate than the Haar-cascade around 6-8% because MTCNN can detect partial object, while the Haar-cascade cannot. When part of the driver's face is hidden during turning the steering wheel, the Haar-casade cannot even detect the face continuously while MTCNN can detect the face and eye. The closing eye event is also important. The algorithm that can detect closing eye event properly will lead to less fault alarm. The MTCNN can do better in both scenario and real-life and the Haar can do very well too. But when driver look at the other way, the MTCNN detected rate is lower than the Haar because of wrong localization of the eye. The haar has no problem in localization but the mistake occurs because it cannot detect the face well when look to left or right

Nevertheless, this study is developing the algorithm which can work properly on Raspberry Pi. The result of this study shows that the Haar-cascade detection is more suitable with Raspberry Pi but MTCNN is not suitable at all.

REFERENCES

(1) Ministry of Transport: Road Accidents Prediction Analyzation Annual Report (2014).

(2) Mohamad H. Sigari, Muhamad R. Pourshahabi, Mohsen Soryani, and Mahmood Fathy : A Review on Driver Face Monitoring Systems for Fatigue and Distraction Detection vol. 64 (International Journal of Advanced Science and Technol) (2014).

(3) Paul Viola and Michael Jones : Rapid Object Detection Using a Boosted Cascade of Simple Features (Mitsubishi Electric Research Laboratories) (2004).

(4) Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao : Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks vol.23 (IEEE Signal Processing Letters) (2016).

(5) Vahid Kazemi and Josephine Sullivan : One Millisecond Face Alignment with an Ensemble of Regression Trees (The IEEE Conference on Computer Vision and Pattern Recognition) (2014).

(6) Christos Sagonas, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic : 300 Faces in-the-Wild Challenge: The First Facial Landmark Localization Challenge (The IEEE International Conference on Computer Vision) (2013).

(7) Adrian Rosebrock : Drowsiness detection with OpenCV (https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/) (2017).

(8) Davina Bristow, John D. Haynes, Richard Sylvester, Christopher D. Frith and Geraint Rees : Blinking Suppresses the Neural Response to Unchanging Retinal Stimulation vol.15 (Current Biology) (2005).

(9) Daniel Ramot : Average duration of a single eye blink (BioNumber ID:100706) (2001).

(10) Tereza Soukupov´a and Jan Cech : Real-Time Eye Blink Detection using Facial Landmarks (21st Computer Vision Winter Workshop) (2016).

AUTHOR BIOGRAPHY

Author: Mr. Manussanan Saengpenchai

Date of Birth: 24th May 1993

Place of Birth: Bangkok, Thailand

Undergraduate and Graduate Education:

Master's degree in Automotive Engineering,
King Mongkut's Institute of Technology Ladkrabang, Bangkok, 2022

Bachelor's degree in Mechanical Engineering,
Mahidol University, Nakhon Pathom, 2014

Presentations and Publications:

[1] Saengpenchai Manussanan, and Srisurangkul Chadchai "Study of a Visual Driver Drowsiness Monitoring Algorithms Based on Raspberry Pi 3", 2019, The 28th JSAE Annual Congress (Spring), May 22-24, 2019, Kanagawa, Japan