



Control Interface using Gestures for the ABC wheelchair

by

Vivian Hui Zhen Foeng

Project supervisor: Professor David Powers

October 2018

*Submitted to the College of Science and Engineering in partial fulfilment
of the requirements for the degree of Bachelor of Engineering
(Biomedical) / Master of Engineering (Biomedical) at Flinders University,
Adelaide, Australia*

DECLARATION

I certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Signed: Vivian Hui Zhen Foeng

Dated: 15 October 2018

ACKNOWLEDGEMENTS

I would like to extend my appreciation and gratitude to my supervisor, Professor David Powers for his continual guidance and support throughout my project.

I would like to thank Bertha Naveda for her patience, support and advice and RajKunwar Kukreja for his help with the ABC wheelchair.

Finally, I would also like to thank my family and friends for going through this journey with me with their never-ending love, support and patience.

ABSTRACT

Smart wheelchairs offer a solution to wheelchair users that have difficulties controlling a standard power wheelchair, which may be due to cognitive or severe mobility problems. At Flinders University, the development of the ABC wheelchair aims to meet this goal with the use of autonomous navigation and the ability to accept various inputs. Alternative control methods have been explored particularly for those who have limited movement of their upper limbs. Gestures have been identified as a promising option as they are natural to human communication and can be adapted for the capabilities of the user. Face and head gestures could be particularly useful for those who may have limited control over their upper limbs.

This project explores the use of face and head gestures in the control of a smart wheelchair using computer vision for a non-contact alternative. The emphasis of this project is on the user and exploring the factors that affect their use of this system while exploring the factors that influence the effectiveness of this system. This system was implemented using the open-source software OpenFace and an Arduino Mega 2560 for control of the wheelchair. The main challenges that was faced were in user intention and software limitations. These were aimed to be solved by making assumptions surrounding the intention of the user. One assumption made was that they would hold a gesture for a longer time if it was intentional. Software limitations faced were inherent issues with face tracking and computer vision. Through the implementation of this system, further considerations were found and addressed.

It was found that face and head gestures were a viable method of control but lacked in reliability. The performance of the system ranged from 65.23% to 88.28% in informedness, which indicates that while the system most likely can predict the correct gesture with the given information, there are several factors that limit the performance of this system. Individual differences in appearance as well as execution of gestures plays a large role in this. Therefore, more research would need to be conducted into the area of face and head tracking to create a reliable and versatile system. Implementing more varied training data or 3D data could potentially improve these issues. The challenge of deducing user intention can also be improved by integrating various user sensing capabilities such as a brain computer interface (BCI). More studies to formally evaluate this system from a user's perspective could also be future work.

CONTENTS

Declaration.....	2
Acknowledgements.....	3
Abstract.....	4
Contents.....	5
Nomenclature.....	7
1. Introduction.....	8
1.1 Background.....	9
1.1.1 People with Disabilities and their capabilities.....	9
1.1.2 The ABC Wheelchair.....	10
1.1.3 User Interface.....	12
1.1.4 Gestures.....	13
1.2 Relevant Technologies.....	13
1.2.1 Use of Face and Head Gestures in Control Interfaces.....	13
1.2.2 Computer Vision in the recognition of face and head gestures.....	16
1.2.3 Realising a Face and Head Gesture Recognition system.....	24
1.2.4 Most suitable tools.....	35
2. Application and Implementation.....	36
2.1 Project Aims, requirements and Considerations.....	36
2.2 User Input and selection of suitable gestures.....	39
2.3. Face and Head tracking.....	41
2.3.1 3D Camera.....	41
2.3.2 OpenFace.....	43
2.4 Interpretation of Gestures.....	50
2.4.1 Role of Emotion in Gestures.....	50
2.5 Control Interface.....	51
3. Results.....	55
3.1 Face and Head Tracking.....	55
3.1.2 Testing participants.....	63
3.1.2 Scenarios.....	64
3.1.3 Conditions.....	64
3.2 Control interface.....	64
3.2.1 Performance.....	68

3.3 Wheelchair Interface.....	71
4. Discussion.....	74
4.1 Significance of Results.....	74
4.2 Limitations.....	75
5. Conclusion and future work.....	78
References	80
Appendix.....	85
Appendix A – Neutral Tests and execution of gestures	85
A.1 Preliminary Tests done on the author	85
A.2 Subject 1	98
A.3 Subject 2	102
A.4 Subject 3	107
A.5 Subject 4	111
A.6 Subject 5	117
A.7 Subject 6.....	123
A.8 – Individual Parameter sets.....	129
Appendix B– OpenFace Control Interface.....	131
B.1 Openfaceinterpreter.h	131
B.2 Openfaceinterpreter.cpp – Modified FeatureExtraction.cpp	132
Appendix C – Serial Connection for the arduino (Mandal, 2016)	141
C.1 Serial.h	141
C.2 Serial.CPP.....	141
Appendix D – Arduino Code to control the wheelchair (Kukreja, 2018)	144
Appendix E – Standard Testing protocol.....	147

NOMENCLATURE

AAM	Active Appearance Models
ABC	Audiovisual Brain-muscle Computer-Controlled
ABS	Australian Bureau of Statistics
ANN	Artificial Neural Network
API	Application Programming Interface
ASM	Active Shape Models
AU	Action Unit
BCI	Brain Computer Interface
BRFT	Beyond Reality Face Nxt Tracker
CNN	Convolutional Neural Network
CP	Cerebral Palsy
CPU	Central Processing Unit
EEG	Electroencephalogram
EOG	Electrooculogram
FACS	Facial Action Coding System
FOV	Field of Vision
FPS	Frames Per Second
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HCI	Human Computer Interface
IoT	Internet of Things
ISO	International Organization for Standardization
NUI	Natural User Interface
PD	Parkinson's Disease
SCI	Spinal Cord Injury
SDK	Software Development Kit
SVM	Support Vector Machines
VGA	Video Graphics Array

1. INTRODUCTION

Mobility plays a significant role in our lives. It enables us to be independent and increases the opportunities to participate in life's activities. Mobility impairments are quite predominant in Australia. Around 614 200 people reported using mobility aids in the ABS 2015 survey (Australian Bureau of Statistics, 2015a) (Australian Bureau of Statistics, 2015b)

People with disabilities all have differing capabilities and can be limited in control options for mobility aids. In the context of powered wheelchairs, the use of a joystick is traditionally employed for use. For those that have limited movement of their upper limbs, head or chin switches and pneumatic pressure such as a sip-puff switch are also common choices (Fehr et al., 2000). However, this can be fatiguing, mentally taxing, it can compromise functions such as speech or it may not be very intuitive for the user.

Control interfaces are one of the most crucial components in the usability of these devices. It governs whether a device is very simple to use or extremely difficult. As no disability is the same, it makes sense that no mobility solution would be the same. Therefore, it is crucial that wheelchair users have a variety of options to consider when it comes to controlling their personal mobility device, particularly if some methods are fatiguing or difficult for the user.

Smart wheelchairs offer many alternatives for the control of movement and navigation based on this principle, particularly for those that may have difficulties with ordinary controls and navigation. In a survey conducted by Fehr et al. (2000), it was reported that "85% of responding clinicians reported seeing some number of patients each year who cannot use a power wheelchair because they lack the requisite motor skills, strength, or visual acuity". For these reasons, a user may not be able to operate a powered wheelchair reliably or safely with the standard technology currently on them. At Flinders University, the development of the Audio-Visual Brain-muscle Computer-Controlled (ABC) wheelchair aims to address this issue. This device aims to provide the functionalities of a regular powered wheelchair with additional technological features to relieve the cognitive and physical burden on users often in the form of autonomous navigation, assisted control and advanced user interfaces.

The focus of this project is to explore alternative control methods for the ABC wheelchair. Face and head gestures has been identified as a potentially promising control method as gestures are a natural form of communication that people use daily. There are fewer studies on face and head gestures compared to hand gestures, but these are being increasingly explored as upper limb impairments are being considered more.

Providing people with mobility impairments more avenues to control their mobility device would enable them to experience increased independence and could potentially improve their wellbeing and safety.

This thesis will provide an overview of related technologies as well as explore an alternative face and head gesture control interface for the ABC wheelchair. Chapter 1 will discuss the motivation behind this project and explore the relevant work done in this area. Chapter 2 describes the project goals as a whole and how it was implemented. Chapter 3 will lead on from this and presents the results and main findings. This will then be discussed in Chapter 5 and finally concluded.

1.1 BACKGROUND

1.1.1 PEOPLE WITH DISABILITIES AND THEIR CAPABILITIES

There are many disabilities that can affect the motor capabilities of individuals using the wheelchairs. For this project, the focus is primarily on those that have severe disabilities and experience difficulties in controlling their upper limb movements. Quadriplegia is a prominent example of this and can occur through illness or injury. While spinal cord injury is the main cause of quadriplegia, infections, brain tumours and congenital defects are also conditions that can have an effect. (SpinalCord.com, N/A)

Cerebral palsy can affect people in half their body or all limbs as well as the trunk, face and mouth to an extent (Cerebral Palsy Alliance, N/A)., whereas people with SCI are more likely to have more control over their face as the condition is more localised. For people with severe conditions such as spinal cord injury (SCI), quadriplegia, Parkinson's disease (PD) and cerebral palsy (CP) who likely have limited control of their upper limbs, it may be more desirable to use their head and face for control inputs.

As each individual and disability is different, their needs and capabilities will vary. Therefore, it is important to develop a versatile control interface so that users can maximise their existing capabilities.

1.1.2 THE ABC WHEELCHAIR

As suggested by the name, the Audiovisual Brain-Muscle Computer-Controlled Wheelchair is a multi-faceted project that has undergone several stages of development (Robinson, n.d., Asayesh, 2013, Khazab, 2016, Kukreja, 2018). It aims to integrate several different technologies and types of inputs to achieve a safe and reliable wheelchair. In this way, it can help to lessen the physical or cognitive burden on the user.

Depending on the features implemented, smart wheelchairs can range from semi-autonomous and fully autonomous. In this case, the aim for the ABC wheelchair is to create a cost-effective system that can be integrated onto different powered wheelchairs which includes both semi-autonomous as well as fully autonomous functionalities. It should be able to accept user input to control the wheelchair directly as well as destination input, where it can travel to the destination without further input.

Path planning, obstacle avoidance (Yanco, 1998) and wall-following (Kuno et al., 2003) are all examples of the many functionalities a smart wheelchair can provide (Simpson, 2005), the first two of which are being implemented in the ABC wheelchair. Previous work on the wheelchair has involved implementing the control and sensing ability of the wheelchair (Robinson, n.d., Asayesh, 2013) as well as exploring a brain computer interface (BCI) (Khazab, 2016). Currently, the wheelchair can use a combination of inputs to obtain information about the environment around the user as well as obtain user input. A BCI, autonomous navigation and the ability to intake various inputs are being implemented at the time of writing (Kukreja, 2018). The wheelchair is shown in Figure 1.



FIGURE 1: CURRENT ABC WHEELCHAIR AT FLINDERS UNIVERSITY

The ultimate goal for the wheelchair is to create a system that is integrated with the Internet of Things (IoT) to establish a connection and enable data exchange between the wheelchair and other relevant technologies. For example, the act of getting a coffee could become an automated process, in which the user communicates their desire for coffee and this can be made before their arrival.

In terms of the control of the system, inputs from the user can be collected through several means including a brain-computer interface, voice input, eye movement as well as head and facial gestures. A problem that is often encountered with the control of the smart wheelchair using unconventional methods is that it can be difficult to deduce the intention of the user (Ju et al., 2009) (Kuno et al., 2003) (Fine and Tsotsos, 2009). While some systems base assumptions around the intentions of the user in order to decide upon actions, it is becoming more common to utilise a combination of these inputs to determine the intentions of the user with the most

certainty. Thus, multimodal inputs have been gaining popularity for both the control of the device and the function.

The role of this project within the ABC wheelchair project is to explore the use of facial and head gestures for the control of the wheelchair.

1.1.3 USER INTERFACE

A major aspect of the control of a powered or smart wheelchair is the user interface, which is how the user controls the device. There are many aspects to consider when designing the control interface for a user as individual motor, sensory and cognitive skills vary vastly among people with disabilities as discussed previously.

Usability is described by ISO 9241–11 as a three-pronged approach, which describes “ The extent to which a product can be used by specified users to achieve the specified goals with effectiveness, efficiency and satisfaction in a specified context of use” (Dix, 2009). Useability is important as this project can influence an individual’s mobility and independence. In this context, this means the user interface should enable the user to accurately control the wheelchair while minimising the resources involved. Therefore, it is important to aim to minimise the physical and cognitive effort on the user while ensuring that they feel comfortable and happy to use it. A simple and intuitive interface could potentially meet these requirements.

The concept of a natural user interface (NUI) is one that is especially pertinent to this project. The purpose of making an interface as natural as possible is to make it feel more seamless and direct to the user, and in doing so could potentially improve the useability.

There are a few measures that can be taken to ensure a NUI. In this situation, this involves considering gestures that are most intuitive to the user. This could be as simple as using an action involving left, for turning left. Using components of emotional expressions that the user can easily relate to a specific action could also feel more natural for the user. Culture and society can also influence which gestures would feel natural for a user. For example, in Filipino culture it is common for pouting to be used to point (SBS, N/A). This could mean that it would be more natural for a Filipino wheelchair user to use this pouting gesture to indicate where they’d like to go.

1.1.4 GESTURES

Thus far, gestures have been identified as a promising hands-free control method and the idea of using gestures to convey information has been a prevalent topic of interest in many fields. Gestures are movements that can be used to communicate emotion, meaning or intent to other people or the environment. They can be expressed through the movement of various parts of one's body, but is predominantly conveyed using the fingers, hands, arms, head and face (Mitra and Acharya, 2007). They have been analysed in several ways for a variety of reasons. In linguistics, there is an interest in the role of gestures as an essential part of language (Abner et al., 2015) as it is possible to ascertain what one is communicating through gestures. Facial gestures, which are defined as individual movements of the face can also play a role in this when considering expression. Naturally, the relationship between facial gestures and related emotions has also been studied extensively (Ekman et al., 1997) (Ekman, 1993).

A common method for describing and tracking facial gestures is through the Facial Action Coding System (FACS) or a variant of this. It is a systematic method of describing facial actions and was introduced by Ekman and Friesen (1976). The FACS is made up of action units (AU) which describes different facial and head movements. While most of this system is based on the more complex anatomical structure and movement of facial muscles, it is a common method used to describe specific facial movements ranging from small, nearly imperceptible movements to larger obvious movements. Usually the recognition of AUs are performed by trained individuals but the automatic recognition of AUs are being explored further.

Naturally, there are inherent challenges to using gestures as they can be used both intentionally or unintentionally. Therefore, the main challenge in using this in a control interface would be interpreting the user's intention.

1.2 RELEVANT TECHNOLOGIES

1.2.1 USE OF FACE AND HEAD GESTURES IN CONTROL INTERFACES

In general, gestures can be detected through wearable sensing devices (Ben Taher et al., 2016) (Zogg, 2017) (Matthies et al., 2017) or vision-based detection (Ju et al., 2009) (Mitra and Acharya, 2007). Specific examples of methods used to obtain

information from the face and head include tracking biosignals to determine eyebrow movement (Wei et al., 2009), reading accelerometer's to determine head placement (Fatma, 2016) or using computer vision to determine head inclination and facial gestures (Ju et al., 2009). Meaning can then be derived from the information obtained about positioning, angles and movements of the relevant features.

These principles have been applied to a wheelchair previously. A prominent example of gesture detection in this context was demonstrated by the University of Essex's RoboChair, which used a variety of inputs such as bio-signals from the forehead (Wei et al., 2009), EOG readings (Tsui et al., 2007) as well as computer vision to detect head gestures (Jia et al., 2007).

In more recent years, Nasif and Khan (2017) and Wu et al. (2017) both use a similar concept to utilise wearable devices with the attachment of sensors such as accelerometers to measure movements of the head for the control of a wheelchair. These devices are shown in Figure 2.

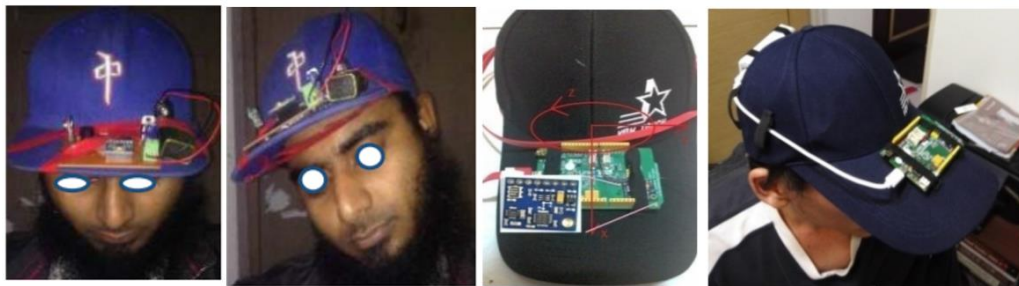


FIGURE 2: WEARABLE DEVICES TO DETECT HEAD GESTURES IN THE CONTROL OF A WHEELCHAIR. THE TWO LEFT IMAGES SHOW THE WEARABLE DEVICE CREATED BY NASIF AND KHAN (2017), WHILE THE TWO RIGHT IMAGES SHOW THE WEARABLE DEVICE CREATED BY WU ET AL. (2017).

However, while Nasif and Khan (2017) uses the interface to control a wheelchair directly, Wu et al. (2017) used additional gyroscope and compass sensors to train and use a classifier that can detect six different head gestures. Both studies are examples of cost-effective methods of controlling a wheelchair using wearable sensors. However, these studies do not address the challenge of intention.

Another interesting example of a wearable sensor was created by Matthies et al. (2017), who developed an interface that uses electrodes to sense a user's facial gestures as shown in Figure 3.

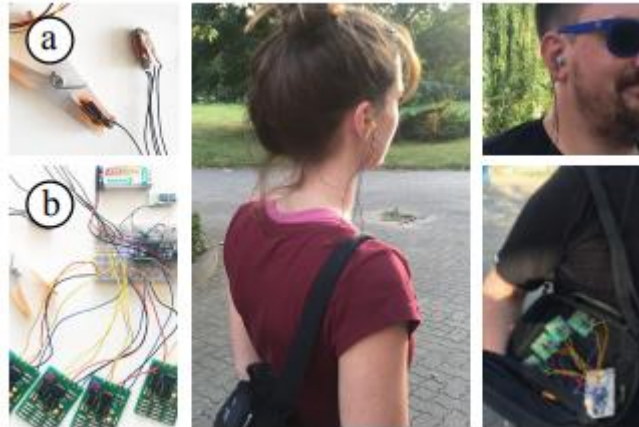


FIGURE 3: WEARABLE ELECTRIC FIELD SENSING DEVICE THAT CAN SENSE FACE-RELATED GESTURES (MATTHIES ET AL., 2017)

These electrodes are placed inside the ear and detects changes in the electric field due to movements of a user's face, which is then transmitted to the appropriate application through Bluetooth. This particular interface is designed to be applied to mobile applications, in order to increase hands-free accessibility but could also be applied to this context. While this device is relatively unobtrusive, there is quite a lot of wiring and componentry involved which may be cumbersome. This interface offers a promising hands-free control method if further development can make it more compact with fewer associated components.

A common issue across wearable devices such as these is that they may be cumbersome for the user. Wearable technology can be bulky and uncomfortable to wear and can often have cables that connect the user to a processor or will need to be set up before use. As mentioned, it is important that the user feels comfortable using the interface and resources such as time required is minimised. Therefore, vision-based technologies are a common direction that many control interfaces are taking (Betke et al., 2002, Kuno et al., 2003, Jia et al., 2007, Ju et al., 2009, Bankar and Salankar, 2015), as it is less intrusive and allows the user to be more comfortable while decreasing the need for others to assist in configuring the device. However, while vision-based techniques are free from these disadvantages, they can be affected by occlusion along with other challenges that are inherent to computer vision. In saying that, computer vision techniques are a favourable alternative as they are non-contact, can be more cost effective and can be quite robust.

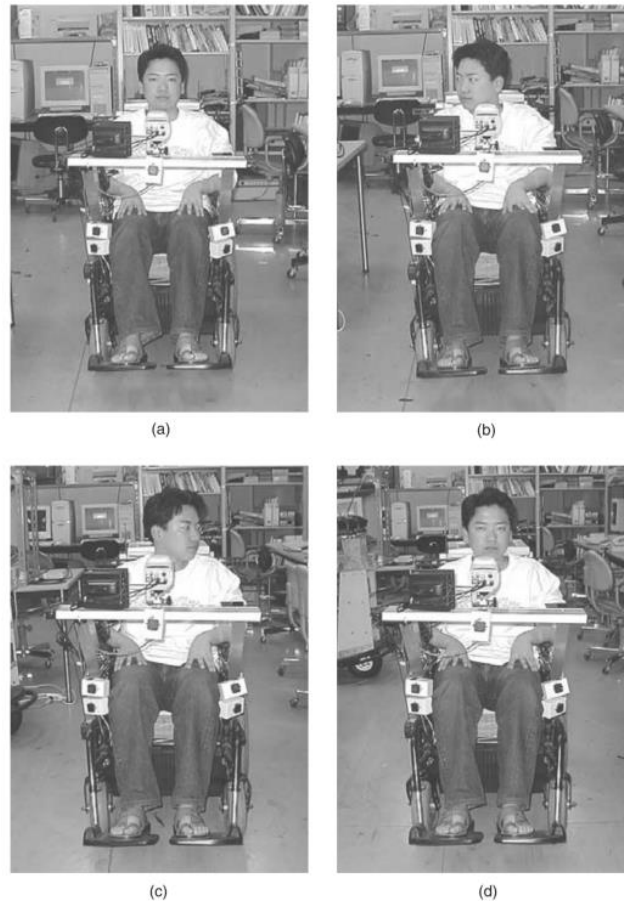
Therefore, the exploration of a non-contact control interface that can enable people with disabilities the adaptability they require for their individual capabilities is desirable. Existing literature and implementations will be analysed to select the most suitable implementation methods.

1.2.2 COMPUTER VISION IN THE RECOGNITION OF FACE AND HEAD GESTURES

Computer vision is currently integrated into the development of many smart wheelchairs and is a popular input method (Leaman and La, 2017). Various methods are used in order to acquire, process and analyse digital media with the goal of understanding and using the information. Image processing and machine learning techniques are heavily involved in the implementation of these methods. It is perceived to be a promising field due to the wide availability of cameras, their small size and the substantial amount of information they can obtain (Simpson, 2005) (Morikawa and Lyons, 2017). Additionally, it is able to receive information in real-time non-invasively at a low cost through being attached to the wheelchair itself. (Morikawa and Lyons, 2017).

Computer vision is very versatile and as such, it can also be used for a variety of applications such as outward-facing vision to aid with navigation and obstacle detection as well as inward-facing vision to detect body gestures. Unsurprisingly, there are many control opportunities available with computer vision. From existing literature, a variety of existing human-computer interfaces that focus on facial or head gestures and use computer vision have been identified.

Kuno et al. (2003) was one of the first research studies that used head gestures as a control input for a smart wheelchair. They used computer vision to detect face direction, in which the user could turn their head to steer the wheelchair and use hand gestures to control the wheelchair remotely. This use of the wheelchair is illustrated in Figure 4.



**FIGURE 4: INTELLIGENT WHEELCHAIR SYSTEM THAT USES FACE DIRECTION TO STEER
(KUNO ET AL., 2003)**

The main issue with this concept was that the intentions of the user were not always clear when they moved their head, i.e. they could be moving their head to look at an object or person. To address this issue, they had to make several assumptions to deduce intention. The first is that slow and steady head movements are made with the intent to steer the wheelchair, which meant quick head movements were ignored. The second assumption made is that the user would likely observe nearby obstacles, without the intention of moving in that direction. Therefore, ultrasonic sensors were used to detect any obstacles that may be close by and used to adjust the sensitivity of the face-turning detection accordingly. While the intention problem is addressed well in this system, it could still pose some difficulties for the user and the people around them. In the trial run, some users felt uneasy with the slow responses of the wheelchair. Therefore, while the system was useable it was not quite user-friendly. Since it is also quite an old project, the hardware and software that was used would not perform as well as it would with modern technology.

Another prominent example is a head gesture-based interface developed for the wheelchair system RoboChair (Jia et al., 2007). This system used the angle of the users face to determine the direction that the wheelchair will be steered. For example, the users head can be detected to be left frontal or right frontal, which indicates to the wheelchair that the user desires to turn left or right, respectively. This is shown in Figure 5.

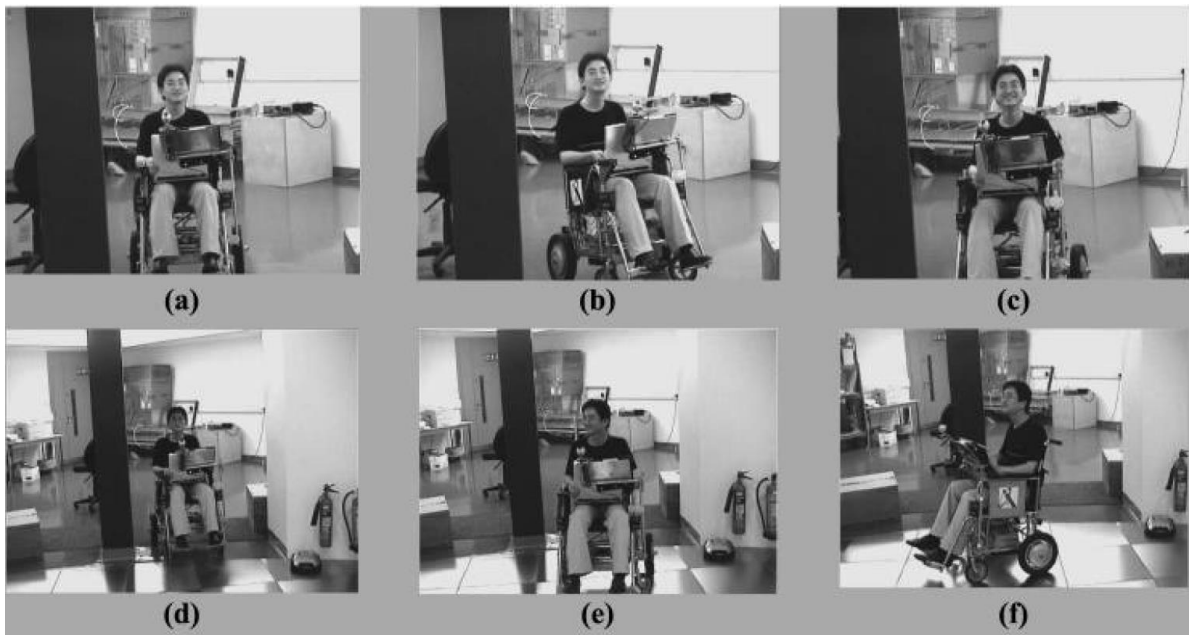


FIGURE 5: ROBOCHAIR IN ACTION, USING HEAD DIRECTION TO STEER THE WHEELCHAIR (JIA ET AL., 2007)

This system appears to have limited safety measures as the only method in which the wheelchair can stop is using a frontal face down gesture. This may not be an appropriate method of stopping in an emergency as it stops slowly and may obstruct the user's vision. The intention problem was addressed in this system by assuming that the user's position in the wheelchair will move when their focus is fixed elsewhere. Therefore, the wheelchair only considers the angle of the users face when it is inside the central field of vision (FOV) of the camera. The angle of the user's face was also a factor, any angle above 45 degrees would not be considered. While this system appeared to have an effective approach, there were many issues encountered that are inherent to computer vision. This included processing speed, variations in lighting and cluttered backgrounds. The tools used in this system included the Viola-Jones face detection and CAMSHIFT face tracking which were tested and compared. Both algorithms are implemented through OpenCV. It was

found that while the CAMSHIFT face tracking algorithm performed better in terms of speed, the Adaboost face detection algorithm was much more robust.

Bankar and Salankar (2015) implemented another system that uses head gestures for the control of a wheelchair. A camera with an embedded processing unit was used for powerful and fast processing. They used the Viola-Jones facial detector in order to detect the face in four directions. These four directions were used to turn left, right, go forward or reverse. While this system is an example of a relatively simple implementation of a head gesture recognition system, the proposal of using a camera with an embedded processing unit may be of further interest for a fast-processing system.

These are just a few examples of research studies that use head gestures for control and while not as prevalent, facial gestures have been demonstrated to be used in a similar way. Several input systems have begun using computer vision to be able to detect and track facial gestures, such as eye blinking, nostril flares, opening and closing of the mouth and facial expressions (Varona et al., 2008, Parmar et al., 2012, Saragih and McDonald, 2017, Rozado et al., 2017). The feasibility of using facial gestures as a user feedback route was explored by Fine and Tsotsos (2009), where they found that it could be a valuable source of input. Eye gaze is the primary focus of many research papers for human control interfaces (Betke et al., 2002, Bartolein et al., 2008, Santos et al., 2014, Bazrafkan et al., 2015). However, there are more control opportunities that can be utilised if the whole face is considered.

FaceSwitch is one such example of an open-source facial gesture detection system created for human computer interaction (Rozado et al., 2017). The control of the interface is placed upon 4 different facial gestures that act as switches, which includes a smile, raised eyebrows, wrinkling the nose and an open mouth as

demonstrated in Figure 6.

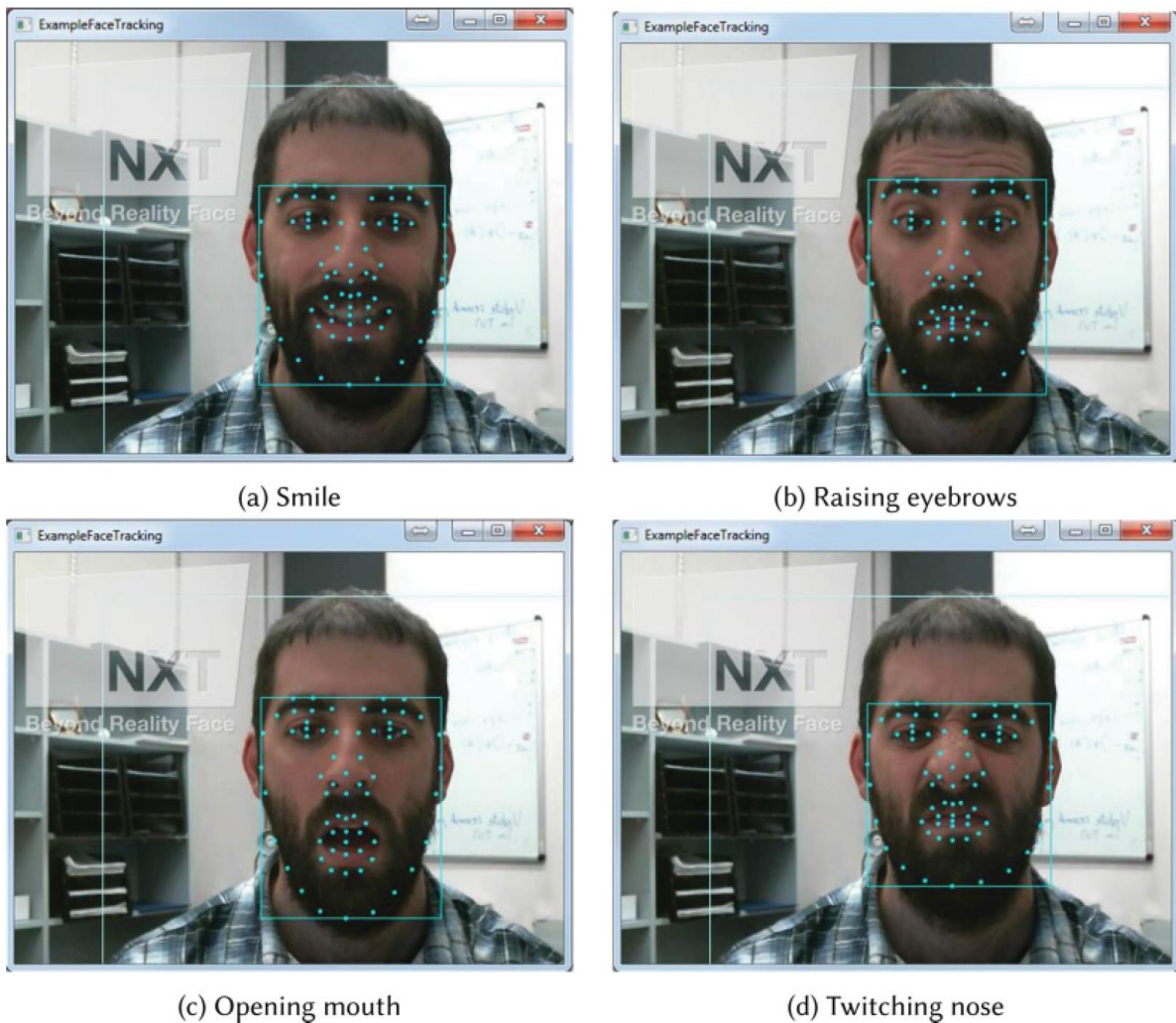


FIGURE 6: THE FOUR GESTURES AVAILABLE FROM FACE SWITCH – A) SMILE, B) RAISING EYEBROWS, C) OPENING MOUTH AND D) TWITCHING NOSE (ROZADO ET AL., 2017)

The aim is to enhance traditional control methods such as eye gaze. In this system, the user's eye gaze acts only as a cursor and magnifying tool, while their facial gestures are used in place of regular mouse commands or keyboard controls. The graphical user interface (GUI) used to select gestures, their thresholds and their commands are shown in Figure 7.

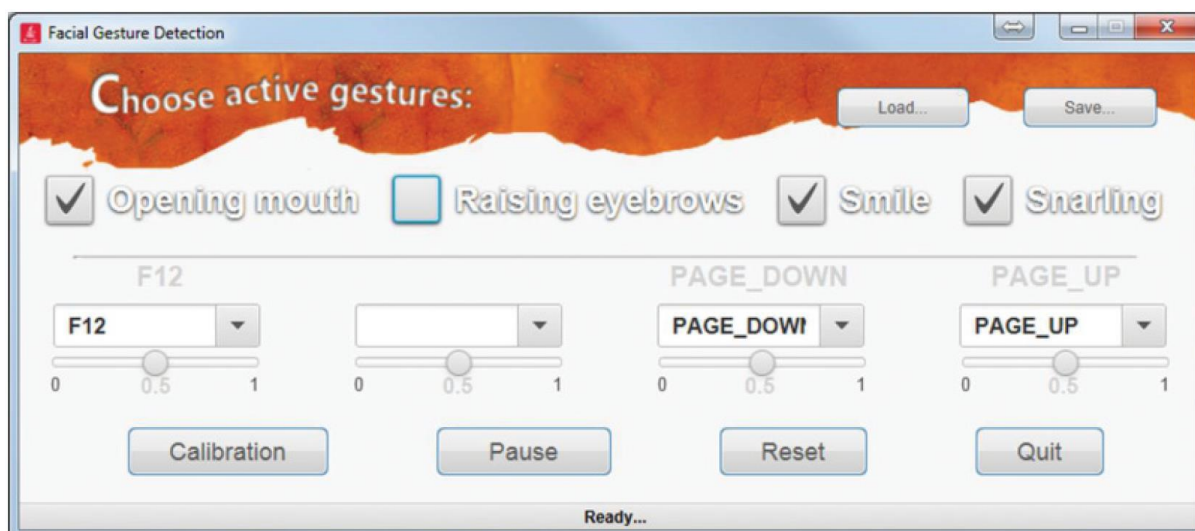


FIGURE 7: GUI OF FACE SWITCH WHERE YOU CAN SELECT GESTURES, THEIR PARAMETERS AND THEIR COMMANDS (ROZADO ET AL., 2017)

This system relies on other tools and applications for the technical aspects, such as the Tobii X2-30 Eye tracker camera that is usually screen-based (Tobii, N/A) and a commercial face tracker, Beyond Reality Face Nxt Tracker (BRFT) in order to track several landmark features on the face. This tracker uses an OpenCV implementation of HaarCascades which is a component of Viola-Jones' system (Tastenkunst, 2018). Other algorithms used were not specified on BRFT's website. However, only a small number of features were able to be tracked with this method which decreased the robustness of the software. It was found that three facial gestures could be detected simultaneously while maintaining reasonable accuracy. Therefore, limitations may have to be placed on the number and type of gestures that the user can be provided with depending on the reliability of the tracker. The 4 gestures were not able to be detected simultaneously with reliability and accuracy. An open mouth and a wrinkled nose were the most recognised gestures. Future work on the system includes the implementation of sensitivity thresholds to enable users with limited motor control of their face to use this system.

Facial gesture recognition has also been investigated in the context of a smart wheelchair (Vazquez-Valencia et al., 2017). A regular webcam with 500x500 pixel resolution was used to capture the user's face from above. Their approach involved the use of several different algorithms to recognise these gestures. Face detection was performed using the Viola-Jones facial detector, and facial feature tracking was

performed with Active Shape Models (ASM) and Active Appearance Models (AAM). However, the number of landmarks were reduced to 12 to improve processing time and this enabled the system to run at 17 frames per second (FPS). A pre-trained Artificial Neural Network (ANN) was then used to classify the gestures. Five gestures were mapped to different actions and used to control their pseudo-wheelchair prototype directly. These are shown in Figure 8. The average accuracy of this system was 85.5% and was considerably affected by variations in lighting.

Gesture / Activation Movement	Command
Head rotation of 70 degrees	Turn left (L)
Head rotation of at least 110 degrees	Turn Right(R)
Eyebrows in up direction	Increase speed (I)
Eyebrows in down direction	Decrease speed (D)
Open Mouth	Stop (S)

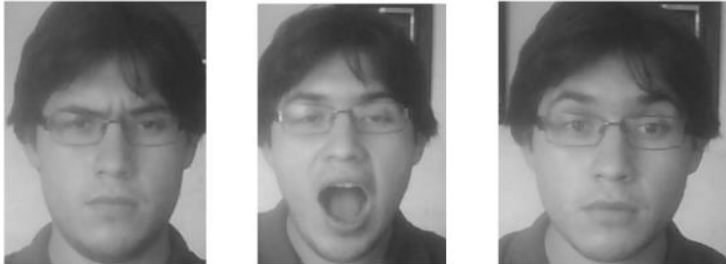


FIGURE 8: FACIAL GESTURES USED IN THE CONTROL OF AN INTELLIGENT WHEELCHAIR (VAZQUEZ-VALENCIA ET AL., 2017)

While systems to track facial gestures and head gestures exist separately, few have explored the use of both facial and head gestures for the control of a human computer interface or wheelchair. This leaves an avenue to be explored for increased control opportunities that are not limited to a few set gestures. Deshmukh et al. (2018) explored the use of face and head gesture recognition for monitoring student interest levels during online learning. However, a relatively small number of gestures were included, and unconventional face gestures were used. Facial gestures used included those associated with sleeping and yawning as well as smiling and head gestures. The face was detected using Haar classifiers and facial features are extracted using an ASM. Similar to the system of Vazquez-Valencia et al. (2017), the features with more significant roles in expressions were identified, and the less significant features were removed accordingly to improve processing speed. Face and head gestures were recognised using different methods. Support Vector Machines (SVM) are used to recognise three facial expressions, while motion

tracking is used to recognise head gestures. The accuracies of both systems were decent, at 97% and 98%. Therefore, a similar method could be investigated to realise a face and head gesture recognition system.

From the review of current face and head gesture computer vision systems, it appears that the selection of gestures is important and can influence how well the system performs. Some gestures may be more difficult to perform or may be more difficult to detect with certainty. Therefore, it is more desirable to have a wide range of facial and head gestures that could be used to control a device.

Deducing intention was also an issue that was shared across many of these studies. It appeared that this was mainly solved by creating assumptions around the use of these gestures. Assumptions made in these studies included slow movements to indicate intentional input and the position and orientation of an individual's head.

The studies implementing the use of facial gestures did not address the intention issue, which is also important to consider as the possibility of mistaking an emotional expression for a gesture is quite likely. However, it is probable that they are relying on the system to be able to differentiate the two.

1.2.2.1 CHALLENGES

In addition to the inherent challenge that gesture-based control presents, there are several challenges that computer vision has yet to overcome, such as efficient processing speed to provide real-time control as well as improved accuracy (Morikawa and Lyons, 2017). The processing speed will be affected by the number of gestures that the system will be able to recognise, the resolution of the data as well as the points tracked. The camera view point can potentially change the appearance of the gestures as well. It is also susceptible to a lot of noise and factors that can affect the input, such as changes in lighting and occlusion.

While the latter challenges have been attempted to be overcome with improvements in software, additional data provided in the form of depth data has been suggested to overcome these limitations. In a study conducted by Chang et al. (2003), it was found that a combination of 2D and 3D data greatly improves the rate of face recognition.

A variety of hardware has been tested in the computer vision field. Regular webcams have been used to obtain information about a user's face, such as blinking and the user's relative head movement (Morikawa and Lyons, 2017). Specialised cameras have also been used such as eye tracking devices that sit upon a user's head (Bartolein et al., 2008). Consumer 3D depth devices such as the Microsoft Kinect and Intel RealSense cameras have been popular choices for obtaining depth data at a low cost (Zhang, 2012, Khoshelham and Oude Elberink, 2012, Draelos et al., 2015). However, it has been found that the Kinect depth technology can be quite noisy (Zhang, 2012), the latest Kinect for Windows v2 has a minimum depth of 0.5 meters which would be unsuitable for face tracking applications at close range (Microsoft, 2017) and has been officially discontinued in 2017 due to its lack of success (Kipman and Lapsen, 2017). Intel's RealSense devices are a promising choice for this application as demonstrated by Silva et al. (2017) and Patil and Bailke (2016) and their successful use of the Intel RealSense SR300 in emotion recognition. Intel's SR300 and D435 cameras have minimum depth ranges of 0.2 meters and the latter camera utilises stereovision models for more accurate depth information (Intel, 2017a, Intel, 2017b). A limitation to many depth devices is that they rely on infrared wavelengths to obtain depth data. As a result, they are unable to perform well under outdoor conditions due to interference by the Sun's own infrared light. The D435 camera potentially has better accuracy in both indoor and outdoor applications as it combines infrared with stereovision techniques to obtain depth information. However, software capabilities are also closely linked to the performance of a device, which will be reviewed in the next section.

1.2.3 REALISING A FACE AND HEAD GESTURE RECOGNITION SYSTEM

Realising a face and head gesture recognition system can range in complexity depending on which tools and techniques are used in such a system. In the succeeding sections, the theory and notable tools and techniques as well as existing implementations are explored.

1.2.3.1 STAGES INVOLVED IN A FACE AND HEAD GESTURE RECOGNITION SYSTEM

Automatic face and head gesture recognition can be a complex task. It shares many similarities with its more heavily researched kin - facial expression analysis, as they use many of the same techniques. For this purpose, facial expression analysis can

be used to illustrate the stages involved in the implementation of a face and head gesture system.

Computer vision and machine learning algorithms and techniques encompass the field of facial expression analysis and tracking and are used with varying degrees of effectiveness. Many combinations of different algorithms and techniques for each stage of the facial expression analysis process have been explored. The specific algorithms and techniques that have been chosen depends on the assumptions that can be made, software and hardware limitations, the application and the requirements of the system.

There are three main stages involved in facial expression analysis. These are detection of the face, feature extraction and classification. The first two stages can be applied directly to the implementation of a face and head gesture recognition system. Classification in this context would aim to recognise individual facial actions. The diagram in Figure 1 shows the basic structure of a facial expression analysis system as well as various methods of performing each step. In addition to these stages, multiple intermediate steps may be necessary such as pre-processing and face-normalisation to ensure the input media is appropriate for use. The need for these intermediate steps depends on the methods used for each stage.

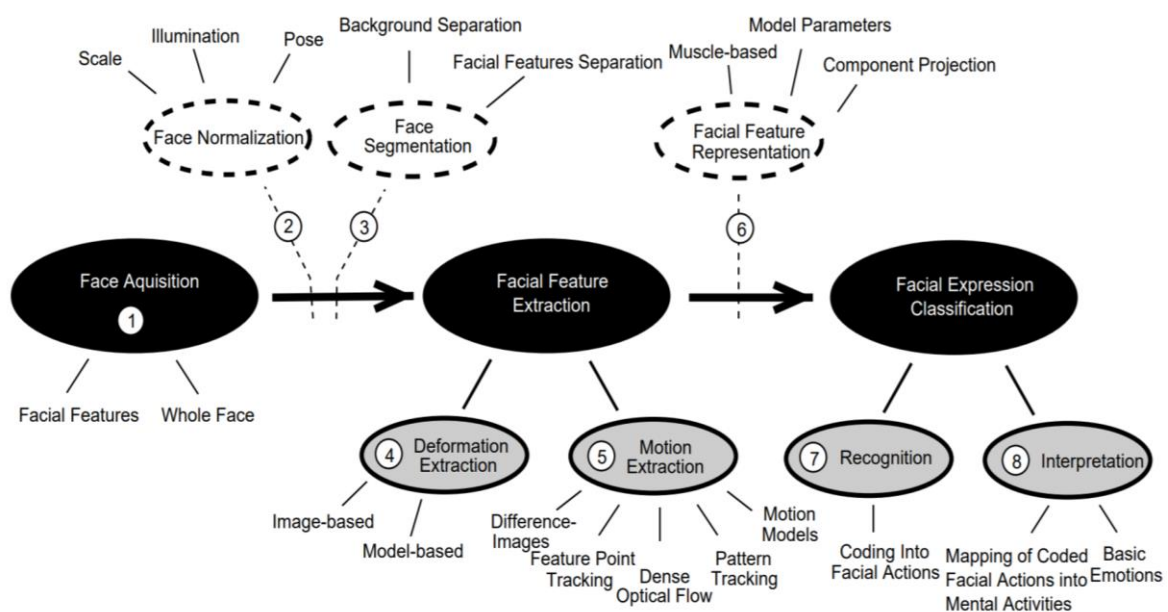


FIGURE 9: STRUCTURE OF FACIAL EXPRESSION ANALYSIS (FASEL AND LUETTIN, 2003)

The first stage involves detecting the face to secure the region of interest. The facial data can then be extracted through one of two methods shown for the second stage. The method used can vary depending on the focus of the study as well as the type of data that is used. The features can then be tracked, and differences or motion can be classified and interpreted to provide recognition of expression or of gestures. For facial gesture recognition, the features can be tracked and classified into the relevant facial action associated with the FACS. This facial information can then be interpreted to find head pose and hence head gestures as well.

1.2.3.1.1 DETECTION

Detection of a face involves the detection of the location and occasionally the size of the facial region, which can be used in later stages. There are multiple approaches to detecting faces in an unconstrained background, which would be the most common situation encountered with a smart wheelchair.

The most prominent face detection algorithm is the Viola-Jones facial detector due to its robustness, real-time processing and efficiency (Viola and Jones, 2004). This algorithm does however have a few limitations; it detects only frontal faces and requires extensive training which means a large data set is needed. However, due to its effectiveness and wide-spread availability, this algorithm and its features are currently employed in many systems (Jia et al., 2007, Varona et al., 2008, Ju et al., 2009, Villaroman, 2013) and is currently the only algorithm used for facial detection in OpenCV (OpenCV, 2018a). Viola-jones facial detector uses a feature invariant approach as it searches for a light nose bridge and a dark eye area and can reach face detection speeds of 15 FPS (Viola and Jones, 2004).

Degtyarev and Seredin (2010) stated that “many researchers [consider] Face Detection tasks ... to be almost solved”. This may be partly due to the significant role the Viola-Jones facial detector has played in many systems, as well as the absence of face detection surveys after 2001. Therefore, to conclude which algorithms were the most effective, they tested a variety of face detection algorithms that were currently available on different platforms. This included the Viola-Jones facial detector on OpenCV 1.0, FaceOnIt by the Idiap research institute, VeriLook 4.0 by Neurotechnology among others. Testing concluded that VeriLook 4.0 had the best performance and processing time, while the Viola-Jones facial detector was second.

(Degtyarev and Seredin, 2010) However, VeriLook 4.0 is a commercial product which means that Viola-Jones would be a good selection for face detection in terms of cost-effectiveness, availability and performance.

However, there are many more recent approaches that are could be more effective as discussed by Chrysos et al. (2017). Template matching and appearance-based methods which matches the face with standard models or learned models, respectively could be a possibility (Yang et al., 2002). Other potential feature-invariant approaches includes a colour-based approach, a prominent example of which was demonstrated by Hsu et al. (2002). However, colour-based methods commonly encounter issues in variations of lighting conditions and variations in facial colours. While this face detection algorithm performed well with varying lighting conditions, head poses and varying skin colours, the system was not very fast. For each second, around five 640x480 pixel images were processed on average. A reduction in image quality improved the speed, with an average of twenty-four 150x220 pixel images able to be processed each second with the Champion image database. (Hsu et al., 2002) While the Viola-jones facial detector is promising, a similar approach to this is also a viable option for a facial detection system.

ANNs have also been explored and investigated for facial detection, the most popular of which is the Convolutional Neural Network (CNN). It operates by reducing the number of possible locations for the face through a series of feature identifiers and requires training with several negative and positive samples. Li et al. (2015) proposed a system that can process 14 FPS VGA images on CPU and 100 FPS on GPU, where the run time could be adjusted according to the accuracy needed. This provided a very fast run time on GPU but provided a similar processing rate to Viola-Jones facial detector on CPU. This particular implementation evaluated the images at low resolution to deliver higher processing speeds and analysed difficult areas at a higher resolution, which is a potential method of cutting down processing time.

Kawato (2002) also proposed an alternative face detection method, which uses the area between the eyes to locate the face. "Between-the-Eyes" is a template matching technique that can run at real-time, processing 27 FPS. It uses skin-colour to extract a facial region which undergoes several intermediate processes until it is able to be accurately matched with the area between the eyes. However, this system

often fails when the user's forehead is obstructed by hair or glasses and has not been tested with many non-frontal faces. The fast processing speed of this system would be desirable, particularly for this application. However, this system is not robust to occlusion.

From the review of various facial detector implementations, it is clear a compromise must usually be made between the detection capabilities and the processing time. It appears that the processing speed of these detectors depends on the CPU and its computational power as well as the particular algorithm used.

1.2.3.1.2 FACIAL FEATURE EXTRACTION

This stage involves identifying and extracting facial landmarks and the associated data including changes in appearance due to movement. Similar to facial detection, there are many approaches for landmark localisation, extraction and the tracking of features, as shown in Figure 9. Landmark localisation involves the detection of significant regions and features on the face. More landmarks are necessary for facial expression analysis compared to tracking facial gestures as facial expression involves the entire face.

There are two main types of features that can be extracted: geometric-features or appearance-based features. Geometric-based features include permanent facial components such as the eyes, mouth, nose as well as any permanent wrinkles. Appearance-based features include those that are transient, such as wrinkles and furrows (Fasel and Luetttin, 2003). While the computational effort may be higher, a hybrid approach where both geometric and appearance features are used could yield better results (Tian et al., 2005). The face can also be analysed through a holistic or local approach, which examines either the face as a whole entity or individual features of the face.

These features can be extracted and tracked with either focussing on the deformation or motion of the relevant features. Deformation extraction focusses on changes in the shape of geometric-based features as well as the appearance of appearance-based features. This is typically compared with a neutral or standard model to discern changes. Gabor filters are a prominent example of this, in which they are able to represent changes in the face using descriptive coefficients that vary in scale and orientation (Tian et al., 2005). In contrast, motion extraction focusses on

the perceived differences in facial features due to movements. This commonly involves using sequences of frames in order to track changes. A prominent example of the motion extraction technique is optical flow which tracks movement as observed in image sequences (Fasel and Luetttin, 2003). The data extracted from this process would then need to be converted to a useful form in order to be applied to the next stage of analysis.

1.2.2.1.3 RECOGNITION/CLASSIFICATION

The last stage of facial expression analysis involves using the extracted features in order to classify expression or in this case, facial gestures based on the aforementioned FACS. This can be performed with machine learning techniques and algorithms such as ANNs, which can be trained to classify these AU.

Another challenge that is faced in this situation is that the classification stage generally requires training with an appropriate dataset that matches what the user would like to classify, which in this case is different facial and head gestures. Datasets that are large enough to supply this sort of data are generally databases that are labelled with emotional expression and not AU. Datasets that are labelled with these facial and head gestures is another possibility that can be explored. However, the process of obtaining and labelling a dataset can be a large one.

In order to obtain a dataset that can be used for training the classification of AU, the dataset will have to be coded manually by a trained FACS expert. There have been few studies relating to creating an automatic AU recognition system and there are few datasets available that have these labels. The few that do include datasets such as SEMAINE, BP4D and DISFA, which have currently been implemented to detect AU in the open-source software OpenFace (Baltrušaitis et al., 2015).

1.2.3.1.4 CHALLENGES

There are a few crucial factors that need to be considered in the development of an effective face and head gesture recognition system. The pose of the person and angle that the camera is placed would affect the data that is interpreted. In addition, facial expressions are often spontaneous while intentional facial movements are not. Differentiating the two types of movements may pose an issue and may contribute to challenges in deducing the user's intention. This may be recognised in part by the intensity of a particular movement or where it occurs. The detection threshold for

intensity of movement could be used in order to separate intentional gestures from unintentional gestures. This would need to be adapted to suit a user's individual facial movement capabilities. Other input modalities could be used as an additional factor to support the recognition of intentional and unintentional movement, such as EEG signals.

There are also limitations in the databases of faces available to train these algorithms as many do not have provide enough variation, particularly those using 3D information. This may compromise the robustness of the facial expression analysis system. Interpersonal facial differences and degree of expressiveness are also a factor that may affect the facial expression analysis system. The dynamics of expression are important to consider as well, as expressions may change from individual to individual and could affect the expression that is analysed at that point. Occlusion of the user's face may also play a role in the operation of the system. (Tian et al., 2005)

1.2.3.2 EXISTING IMPLEMENTATIONS

Many suitable algorithms and techniques for use in face and head gesture recognition have been implemented as open-source software, which is software that is free and accessible for public use. It is constantly being improved, which reduces the need for the user to continually improve it. Using these could provide an efficient way of implementing the first stages of facial and head gesture recognition.

Table 1 shows prominent implementations and tools that could be used to create a face and head gesture recognition system. These are all open-source and compatible with C++ or Python which are the desired programming languages to use in this project. There are a few existing commercial products which analyse facial expression and AU such as Affectiva (Affectiva, N/A) and FaceReader (Noldus, N/A)

The following tools and libraries were concluded as suitable software as they fit the criteria of being open-source and compatible with C++ or python in visual studio. Table 1 shows a summary of open-source tools and libraries that can be used to implement a face and head gesture recognition system. These were identified to have varying capabilities as well as compatibilities.

TABLE 1: OPEN-SOURCE TOOLS AND LIBRARIES THAT ARE COMPATIBLE WITH C++ OR PYTHON IN VISUAL STUDIO

SDK/Libraries	Description	Relevant Tools	Algorithms Used	Hardware Compatibility
Intel RealSense SDK for windows (2016 R3) (Intel, N/A) Accessible from: https://software.intel.com/en-us/realsense-sdk-windows-eol	<ul style="list-style-type: none"> Discontinued version of the intel RealSense SDK Has computer vision functionalities 	<ul style="list-style-type: none"> Face Tracking and Recognition (78 landmark points) Pose Detection (Face orientation in degrees) User background Segmentation Hand Tracking and Gesture Recognition 	<ul style="list-style-type: none"> Unknown 	3D cameras - Intel's F200, SR300 and R200 Cameras
Intel® RealSense™ Cross Platform API (IntelRealSense, N/A) Accessible from: https://github.com/IntelRealSense/librealsense/tree/v1.12.1	<ul style="list-style-type: none"> Mainly encompasses camera capture functionalities No computer vision algorithms Not an official Intel product 	<ul style="list-style-type: none"> Depth, colour, infrared and fisheye streaming (depending on camera) Multi-camera capture Synthetic streaming i.e. depth aligned to colour 	<ul style="list-style-type: none"> N/A 	3D cameras - Intel's F200, SR300, R200, LR200 and ZR300 Cameras C++, Python, Java

<p>Intel RealSense SDK 2.0 (Intel RealSense, 2018) Accessible from: https://github.com/IntelRealSense/librealsense/releases</p>	<ul style="list-style-type: none"> • Development stage • Extended camera capture functionalities 	<ul style="list-style-type: none"> • Can generate and visualise a textured 3D point cloud • Can render and save video streams • Can remove background from video (segmentation) • RealSense-Viewer: quick access to camera to explore data or record • Depth and colour streaming 	<ul style="list-style-type: none"> • Unknown 	<p>3D cameras - Intel's D400 series and the SR300 Cameras</p>
<p>OpenCV (OpenCV, 2018b) Accessible from: https://github.com/opencv/opencv/releases/tag/3.4.0</p>	<ul style="list-style-type: none"> • Computer vision and machine learning library 	<ul style="list-style-type: none"> • Face detection • Face landmark detection 	<ul style="list-style-type: none"> • Viola-Jones Facial Detector (Viola and Jones, 2004) • CAMSHIFT (Bradski, 1998) 	<p>2D camera</p>
<p>Dlib (Dlib, 2018) Accessible from: http://dlib.net/</p>	<ul style="list-style-type: none"> • Toolkit for making real world machine learning and data analysis applications in C++ 	<ul style="list-style-type: none"> • Facial landmark detector • Segmentation (regions) 	<ul style="list-style-type: none"> • One Millisecond Face Alignment with an Ensemble of Regression Trees (Kazemi and Sullivan, 2014) • Efficient Graph-Based Image Segmentation (Felzenszwalb and Huttenlocher, 2004) 	<p>2D camera</p>

<p>OpenFace (Baltrušaitis et al., 2016) Accessible from: https://github.com/TadasBaltrušaitis/OpenFace</p>	<ul style="list-style-type: none"> • Open-source facial behaviour analysis toolkit 	<ul style="list-style-type: none"> • Facial landmark detector • Facial landmark and head pose tracking • Gaze tracking • Facial AU Recognition • Facial Feature extraction 	<ul style="list-style-type: none"> • Constrained Local Neural Fields (Baltrušaitis et al. 2013) • Cross-dataset learning and person-specific normalisation (Baltrušaitis et. Al, 2015) 	2D camera
<p>Kinect for Windows SDK 2.0 (Microsoft, N/A-b) Accessible from: https://www.microsoft.com/en-au/download/details.aspx?id=44561</p>	<ul style="list-style-type: none"> • Discontinued • A variety of development tools encompassing audio and visual functionalities 	<ul style="list-style-type: none"> • 25 point skeleton for a total of six people (each person has 25 skeletal joints) • Thumb tracking, end of hand tracking, open and closed hand gestures • Gesture detection and tracking • Face API – detection, face tracking, modelling 	<ul style="list-style-type: none"> • Unknown 	Kinect v2 Sensor
<p>Microsoft Face Tracking SDK for Kinect for Windows (Microsoft, N/A-a) Accessible from: https://msdn.microsoft.com/en-us/library/jj130970.aspx</p>	<ul style="list-style-type: none"> • Discontinued • Used in conjunction with Kinect for windows SDK 	<ul style="list-style-type: none"> • Deduces head pose and facial expressions • Provides info on: <ul style="list-style-type: none"> ○ Tracking status ○ 2D points ○ 3D head pose ○ AUs 	<ul style="list-style-type: none"> • Unknown 	Kinect Sensor – retail edition
<p>Point Cloud Library (PCL) (PointCloudLibrary, 2018)</p>	<ul style="list-style-type: none"> • For 2D/3D image and point cloud processing • Supports OpenNI 	<ul style="list-style-type: none"> • Segmentation • Keypoint detection 	<ul style="list-style-type: none"> • Unknown 	2D/3D camera

<p>Accessible from: https://github.com/PointCloudLibrary/pcl</p>				
<p>FaceTracker (Saragih and McDonald, 2017) Accessible from: https://github.com/kylemcdonald/FaceTracker</p>	<ul style="list-style-type: none"> • Library for deformable face tracking 	<ul style="list-style-type: none"> • Deformable face tracking 	<ul style="list-style-type: none"> • Face Alignment through Subspace Constrained Mean-Shifts (Saragih et al., 2009) 	<p>2D camera</p>

1.2.4 MOST SUITABLE TOOLS

From the literature, there are several stages involved in the implementation of a face and head gesture recognition system. Different techniques would have to be used depending on the hardware and software used to implement the system due to compatibility with specific cameras and data types.

As discussed, there are inherent challenges in computer vision that could potentially be improved through additional inputs such as 3D depth. As discussed previously, the Intel RealSense cameras SR300 and D435 could potentially provide a more accurate system with the use of the Intel RealSense SDK's. However, this is a less cost-effective option.

Otherwise, there are a few possible implementations using a regular webcam. OpenCV provides a robust face detector through the Viola-Jones detector, while Dlib provides a landmark detector with 68 landmarks.

Face detection is generally performed well with Viola-Jones and its OpenCV implementation, but the latter two stages of face landmark extraction and classification could be improved. Many of these stages have been implemented into freely available open-source software.

OpenFace provides an open-source implementation of various facial behaviour analysis tools. Research and implementation of various algorithms and techniques have been conducted through a joint effort from research groups at the Carnegie Mellon University and the University of Cambridge. Several different datasets were used to train and evaluate landmark detection as well as the AUs.

These are all able to be used with any 2D camera, including a standard webcam. This makes it most suitable for this application.

These particular cameras and the software discussed have their own advantages and disadvantages which will be explored further in Chapter 3.

2. APPLICATION AND IMPLEMENTATION

This chapter defines the intended application and project aims. It also describes the implementation of the system in various stages of the project.

2.1 PROJECT AIMS, REQUIREMENTS AND CONSIDERATIONS

From reviewing the literature, it was clear that there were few explorations of both face and head gestures in the use of a control interface. A combination of both face and head gestures should enable versatility for the user to choose which gestures they'd feel most comfortable using.

Therefore, the primary aim of this project was to explore and investigate the use of face and head gestures in a control interface for the ABC wheelchair. This involves the following:

- Investigating the best gestures
- Implementing an interface that can recognise face and head gestures
- Interpreting the intentional gestures into commands
- Interfacing this system with the ABC wheelchair
- Investigating how well these gestures perform
- Observing improvements that could be made and implementing them if possible
- Determining feasibility of this system

There are several stages that this project can be divided into, as shown in the following flow chart in Figure 10. The requirements and considerations for each of these stages will be detailed in the rest of this section.

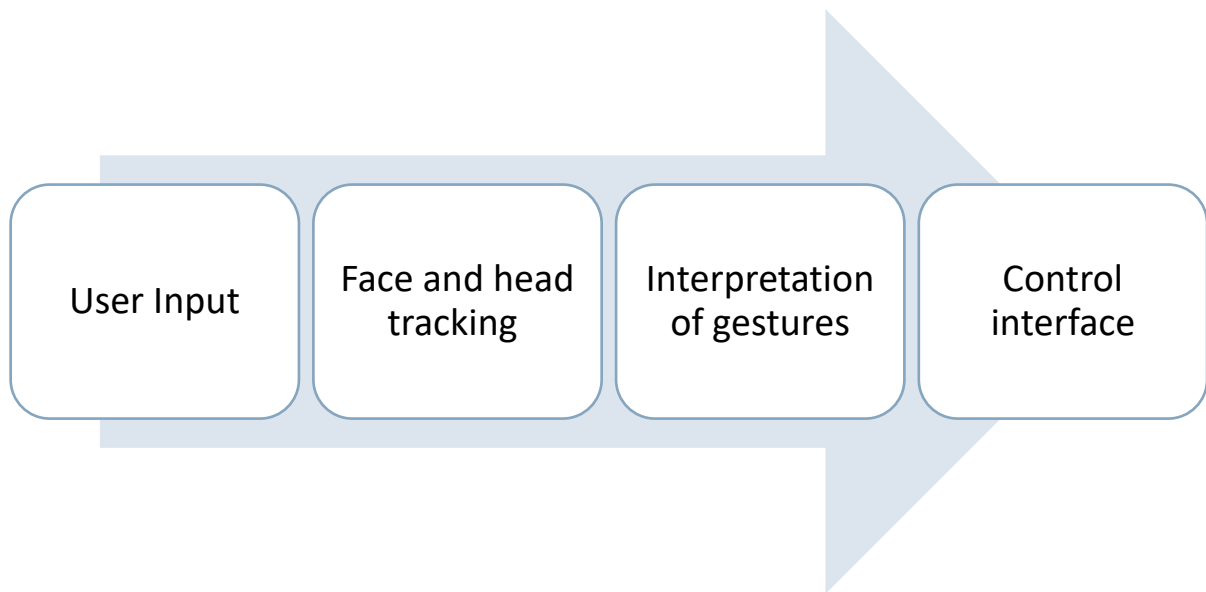


FIGURE 10: FLOW CHART OF STAGES INVOLVED IN THE IMPLEMENTATION OF A FACE AND HEAD GESTURE RECOGNITION SYSTEM

As the focus is on exploring an adaptable and user-friendly interface for a mobility device, the desirable characteristics of this system include:

- Versatility
- Intuitiveness
- Useability (Effective, efficient, satisfactory)
- Reliable
- Safe

These characteristics can be applied to most stages of this project.

In terms of user input, suitable face and head gestures should be selected based on how intuitive they are, and how distinct they are from regular emotional expression in order to be reliable.

The chosen system for the face and head tracker should meet the following criteria:

- Must function in real-time
- Should be consistent
- Should track only one face at a time

The performance of the system can be dependent on a few factors such as the likelihood of intentional gestures being detected and the ease of performance and

how it feels for the user. These will be explored in section 2.3 and 2.4 which focus on the interpretation of gestures and control interface respectively.

The control interface itself aims to integrate the desirable characteristics mentioned. In this case, there are several requirements. It must produce feedback so that the user is aware of gestures they are sending or not sending as well as the current state of the wheelchair. Safety features must be implemented where possible as it is a mobility device and the wrong input can be dangerous for the user as well as the people around them.

There are a few limitations that have been identified, that will need to be addressed where possible in the following stages of development. A major issue that was identified in gesture recognition was one surrounding difficulty deciphering the user's intention. Ways that this has been addressed in the past have been with the implementation of assumptions or multi-modal inputs. Therefore, several assumptions will likely need to be made in the implementation of the system to increase the likelihood of an intentional gesture being detected.

Other factors that could affect the system include the inherent challenges presented with computer vision, limitations with the chosen software as well as user integration. While computer vision has many advantages, it also comes with inherent disadvantages which includes differences in lighting conditions and obstruction.

The software may be affected by individual differences in appearance as well as the position of the face. The performance of detection and tracking could also vary person to person.

With any user-centred product, it is important that the end users are involved in the design process to ensure that it designed in a way that will benefit the user. However, in this case, the ABC wheelchair is still in its initial prototyping stages and the resources required to bring in suitable participants would be far too great as people with severe disabilities would need to be identified and recruited to test this prototype. This would cost money for compensation and considerable time to organise and would require the prototype to be very reliable and safe to use, which may not be guaranteed at the conclusion of the project. Furthermore, the aim of this project is to explore and investigate the use of face and head gestures in the control interface of the ABC wheelchair so involving the end users in this way was


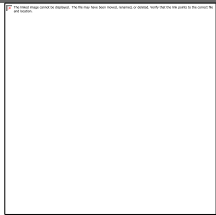


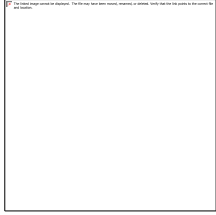
considered outside of the project scope. Since this is the case, it is important to consider the users perspective throughout the development wherever possible. However, this is a limitation that will affect the useability of the device.

2.2 USER INPUT AND SELECTION OF SUITABLE GESTURES

The method in which user input is given is important, as discussed in section 1.1.3. Choosing gestures that aligns with the project aims and requirements can influence the effectiveness of this type of mobility aid. This project also provides an opportunity to investigate which gestures would be better suited to this application.

Initially, a wide range of gestures were identified for potential use in this system as it is important that it is versatile as the abilities of people with disabilities vary greatly. The array of suitable face and head gestures shown in Table 2 and Table 3 were selected for several reasons. Namely, the gesture was easy to perform, and it can be easily distinguished from an emotional expression. A factor that was considered was that some may not be as culturally or societally appropriate as others, and some people may feel varying levels of comfort performing different gestures. However, this becomes less of an issue with a larger array of options.

TABLE 2: SELECTED FACIAL GESTURES TO EXPLORE

Gesture (AU)	Example	Gesture (AU)	Example
Raise Eyebrows AU01 – Inner Brow Raiser AU02 – Outer Brow Raiser		Press lips together AU23 – Lip Tightener	
Wink (L and R) AU46 – Wink	 	Lip Suck AU28 – Lip Suck	








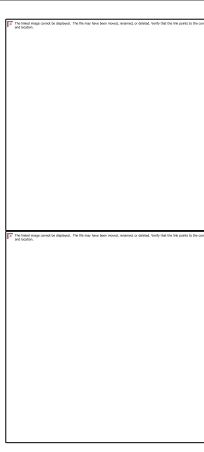








<p>Flare Nostrils N/A</p>		<p>Pout AU18 – Lip Puckerer</p>	
<p>Wrinkle Nose AU09 – Wrinkle Nose</p>		<p>Tongue out N/A</p>	
<p>Puff cheeks N/A</p>		<p>Open mouth AU25 – Lips Part AU26 – Jaw Drop AU27 – Mouth Stretch</p>	
<p>Smirk (L and R) N/A</p>		<p>Move mouth (L and R) N/A</p>	

TABLE 3: SELECTED HEAD GESTURES TO EXPLORE

Gesture (AU)	Example	Gesture (AU)	Example
<p>Rotate Head to the Left AU51 – Head Turn Left</p>		<p>Tilt Head Down AU54 – Head Down</p>	

<p>Rotate Head to the Right AU52 – Head Turn Right</p>		<p>Nod AU53 – Head Up AU54 – Head Down</p>	 
<p>Shake Head AU51 – Head Turn Left AU52 – Head Turn Right</p>	 	<p>Tilt head to the left AU55 – Head Tilt Left</p>	
<p>Tilt Head Up AU53 – Head Up</p>		<p>Tilt head to the right AU56 – Head Tilt Right</p>	

As seen in the tables, not all the gestures can be classified through AUs. Therefore, alternative ways of detecting these gestures would need to be explored. However, this is limited by the tools and software available to implement the detection of these gestures as shown in the next section.

2.3. FACE AND HEAD TRACKING

2.3.1 3D CAMERA

In the literature review, it was evaluated that computer vision would be the best technique to use as it requires less equipment and is more robust in the information that can be detected. However, there are various methods that can be used to implement a face and head gesture recognition system using computer vision. Several avenues were explored; one of which involved the use of 3D data with the Intel RealSense D435 and another which involved the use of a standard webcam.

Originally, the Intel RealSense 3D short-range cameras were identified as the most suitable choice for this application as 3D depth could potentially improve the accuracy of the system and it has been shown to yield decent results, as demonstrated with the Intel RealSense SR300 in the past (Silva et al., 2017) (Patil and Bailke, 2016).



FIGURE 11: INTEL REALSENSE SR300 (INTEL, 2016)

This has likely been chosen for its depth sensing technology, short-range capabilities (0.2-1.5m) as well as for the various capabilities of the Intel RealSense SDK 2016 (Intel, 2016). However, a similar system had already been achieved with the SR300 and its associated SDK. Face gestures for use in an Augmentative and Alternative Communication (AAC) device was implemented by Rich-Perrett (2018) at Flinders University who used the SR300 and the related SDK 2016. However, it was concluded that the newer Intel RealSense D435 camera could potentially be used to improve the accuracy of their work. A limitation with the SR300 was that it was not as reliable outdoors as the infrared light from the sun would disrupt the infrared light from the camera (Rich-Perrett, 2018).



FIGURE 12: INTEL REALSENSE D435 (INTEL, 2018A)

Therefore, the D435 camera was mainly explored for this purpose. The Intel RealSense Depth Camera D435 retails for \$179 USD and contained improved depth sensing technology which can be used outdoors more reliably. The range for this camera is 0.2-10m (Intel, 2018a). There were issues in obtaining the camera as it was in high demand and placed on backorder in early 2018 (Intel RealSense

Support, 2018). As the camera was explored further, it was found that while the physical specifications appeared to be excellent for this application, there were limitations in terms of its available software and compatibility. The SR300 and its official SDK were discontinued in late 2017 (Intel, 2018b) and was not compatible with the D435. The D435 was only compatible with the new Intel RealSense SDK 2.0, which includes only basic functionalities and no middle-ware unlike the previous SDK 2016. Intel released the associated SDK 2.0 onto GitHub to enable developers to build functionalities. However, this SDK only provided basic functionalities that enabled it to obtain data and measurements (Intel RealSense, 2018). Both SDK's were implemented but were deemed not useful for this application. The use of machine learning and other available tool was considered as shown in Figure 13.

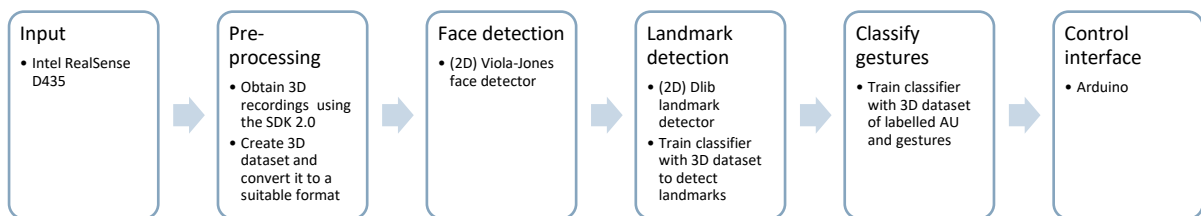


FIGURE 13: ALTERNATIVE FLOW CHART FOR THE IMPLEMENTATION OF D435 USING THE INTEL REALSENSE SDK 2.0 AND MACHINE LEARNING TECHNIQUES

However, the process behind collecting and labelling a 3D dataset is a sizeable effort and labelling them with accurate AU descriptors would require a trained FACS expert. The transitions between 2D data and 3D data would need to be integrated as well. Using other tools and techniques in some stages would only utilise the 2D data and not the 3D data, which would defeat the purpose of using a 3D camera. While this could be an excellent avenue to explore, this was beyond the scope of this project.

2.3.2 OPENFACE

After evaluating the advantages and disadvantages of using machine learning with the D435, it was decided that a standard webcam would be best for this project and scope. According to the Australian Bureau of Statistics (2016), affordability was identified as one of the main factors that impact the use of aids and equipment for people with disabilities. Therefore, this project should be as inexpensive where it can afford to be. As a result, the hardware that was chosen for use was the standard

webcam. One of the overall goals for the ABC wheelchair was also to create a cost-effective device, which this choice supports. It also allows the system to be more easily accessible as well and removes a potential source of incompatibility in the future.

There were a few possible implementations that could be used with a standard webcam. The implementation of the OpenCV face detector and Dlib face tracker appeared to be a promising choice. Therefore, an example of the OpenCV face detection and Dlib face tracking was implemented and compared to the output of OpenFace as shown in Figure 14 and Figure 15 respectively.

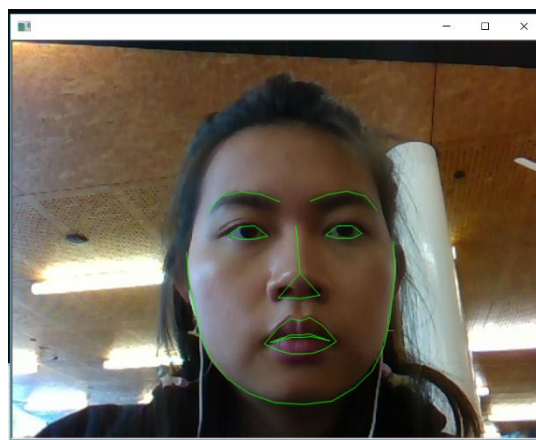


FIGURE 14: DLIB IMPLEMENTATION OF ITS FACE TRACKING SOFTWARE (DLIB, 2018)

However, it was clear that OpenFace was much more robust in terms of the features extracted. It also allowed the user to choose which algorithms they would like to use for face detection and landmark detection. While both implementations could track in real-time and appeared to track the face and detect the landmarks well, OpenFace was chosen because of its versatility.

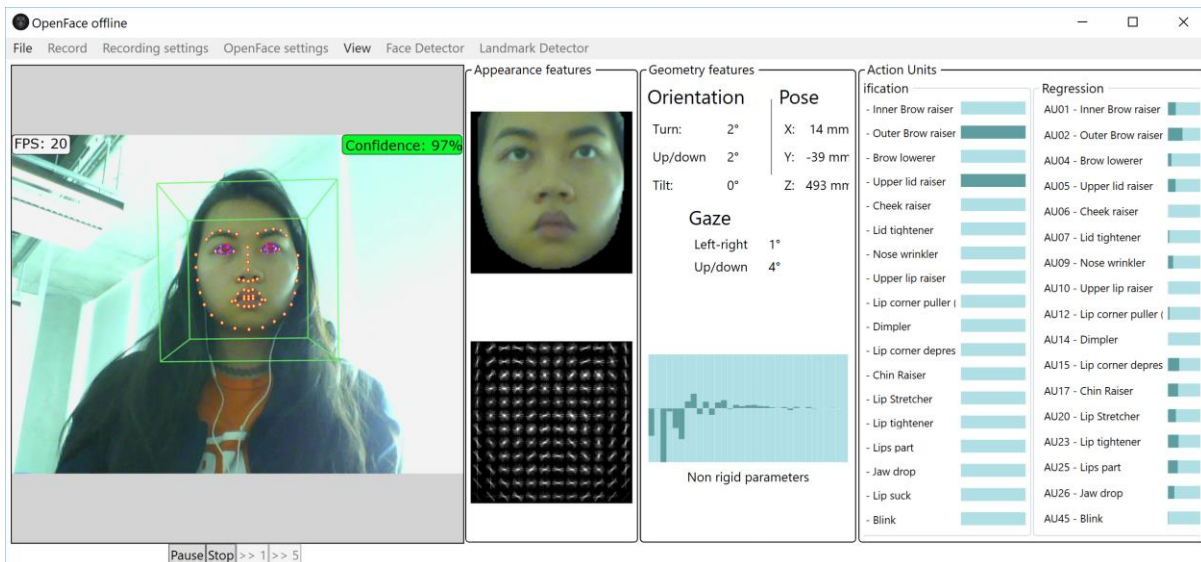


FIGURE 15: OPENFACE APPLICATION (BALTRUSAITIS ET AL., 2018)

OpenFace includes several different modules within the source-code; Figure 15 shows the graphical user interface with every feature that it can obtain. There are also executables within the OpenFace source-code that can be run separately for the extraction of various features with different sources of input. This can be modified from Visual Studio 2015 in its native language C++.

The features that can be extracted from OpenFace include orientation/pose, gaze, landmark positions as well as AUs. The AUs have been trained with Support Vector Machine classification as well as Support Vector Regression on three different datasets. The number of AUs that can be identified in this system are limited due to the training data that was used. As explained previously, it is difficult to obtain a dataset that has AU labels. Measures have been taken to decrease the effect of lighting, occlusion and pose. However, these challenges are not be solved easily and the system is still under constant improvement.

In this application, the various features mentioned can be used to explore the recognition of the gestures that have been selected.

Pose/Orientation

The orientation describes the yaw, pitch and roll of the head, given by turn, up/down and tilt respectively as shown in Figure 16.

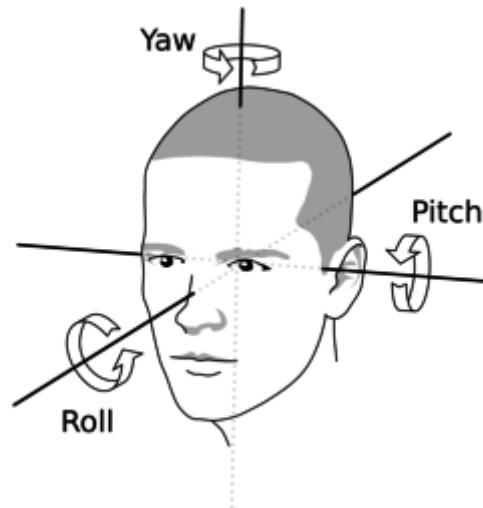


FIGURE 16: DIAGRAM OF YAW (TURN), PITCH (UP/DOWN) AND ROLL (TILT) (JANTUNEN ET AL., 2016)

The values obtained from the yaw, pitch and roll values could be used in the detection of a gesture. The gestures that could be obtained include:

1. Rotate Head to the Left
 - i. AU51 – Head Turn Left
2. Rotate Head to the Right
 - i. AU52 – Head Turn Right
3. Tilt Head Up
 - i. AU53 – Head Up
4. Tilt Head Down
 - i. AU54 – Head Down
5. Tilt Head Left
 - i. AU55 – Head Tilt Left
6. Tilt Head Right
 - i. AU56 – Head Tilt Right

AUs

There are two main outputs for the AUs; based on different classification methods. One classifies the AU as present or not, or on an intensity scale of 0-5. The latter will be the predominant focus of this system as these are measurable. However, it is noted that the gesture lip suck is solely based on the former classification.

The 18 AUs that can be detected are shown in Table 4. As seen, only a few AUs relate to the selected gestures that were identified earlier, such as AU01 and AU02 for raising eyebrows. Additional gestures could also be tested to investigate whether these can be used in the face and head gesture system.

TABLE 4: AUs THAT OPENFACE CAN DETECT

AU	Action
AU01	Inner Brow Raiser
AU02	Outer Brow Raiser
AU04	Brow Lowerer
AU05	Upper Lid Raiser
AU06	Cheek Raiser
AU07	Lid Tightener
AU09	Nose Wrinkler
AU10	Upper Lip Raiser
AU12	Lip Corner Puller
AU14	Dimpler
AU15	Lip Corner Depressor
AU17	Chin Raiser
AU20	Lip Stretcher
AU23	Lip Tightener
AU25	Lips Part
AU26	Jaw Drop
AU28	Lip Suck (Classification Only)
AU45	Blink

5 of the selected gestures could be identified through the AUs that OpenFace can detect, with the exception of AU27.

1. Raise Eyebrows
 - i. AU01 – Inner Brow Raiser

- ii. AU02 – Outer Brow Raiser
2. Wrinkle Nose
 - i. AU09 – Wrinkle Nose
3. Lip Suck
 - i. AU28 – Lip Suck
4. Open mouth
 - i. AU25 – Lips Part
 - ii. AU26 – Jaw Drop
 - iii. AU27 – Mouth Stretch
5. Press lips together
 - i. AU23 – Lip Tightener

Other gestures associated with the other AUs were also identified to assess the feasibility of using them in this system. These included the following:

1. Brow Lowerer (AU04)
2. Widening Eyes (AU05)
3. Squint (AU07)
4. Widen Mouth (AU14)
5. Frown (AU15, AU17)

This means that AUs 6, 10, 12, 45 which are cheek raiser, upper lip raiser, lip corner puller and blink respectively were not used.

2.3.2.1 TESTING

The reliability and role of the AUs and pose in this system can be explored through several tests. Testing was performed on the author and an additional 6 participants in order to quantify the effects of gesture execution on the AUs and how this compares to a neutral expression. Using this information, the interpretation of gestures can be performed and ranges of appropriate thresholds for the detection of gestures can be found.

There were a few stages of testing involved to assess the feasibility of these gestures in the use of this system. Preliminary testing was performed on the author and involved recording a neutral expression for 1.5 minutes in order to get a range of values. The 10 different gestures were then executed individually for 1.5 minutes at a

time for short and long durations, every few seconds. This was done to see if there are any differences between the detection of AUs when gestures are performed quickly or while holding the gesture. The result of this could influence the type of control implemented into the system.

The values obtained from this test can be used to determine which gestures OpenFace can detect best. This can be used to verify which gestures should be used in this system and how further tests should be conducted as only these gestures would be explored. It is important to note that an assumption made with this test is that other users would have similar relationships between their gestures and neutral baseline.

This was explored through further testing of other people. 6 participants were included in the testing who differed in age, gender and ethnic backgrounds. It was required that each participant was over 18 years old, so that they can give consent to participate. They were informed of what the study would involve and that they were participating as volunteers and could stop at any time without consequence. The participants ranged from 22 to 70 years of age, and multiple ethnicities and different genders were included to ensure the data obtained is inclusive of variations within these demographics. Out of the sample size, 4 participants were female and 2 were male. All participants were relatively familiar with technology and were recruited through a direct approach. Ethics were not sought as the data was predominantly used to discern what would work best for the system and did not involve any personal data.

They were asked to perform different gestures and a neutral expression, much like the previous testing procedure. However, only the chosen gestures were investigated to observe how the gestures and their parameters differ person-to-person. The standard protocol used is shown in Appendix E.

The video files for these tests were recorded, and the AUs as well as landmarks were input into a csv file. This was then processed through MATLAB in order to obtain graphs of the intensity of the relevant AUs sorted by the gesture performed. The results of these tests and observations made will be discussed further in the results section.

2.4 INTERPRETATION OF GESTURES

This stage of development determines the framework in which the gestures will be interpreted and how intentional movements can be distinguished from unintentional movements. Using the results gained from the previous tests, the parameters can be implemented.

Several assumptions have been made about intentional gestures which forms the basis for this system. The implementation of these should increase the likelihood of an intentional gesture being detected. These include:

- An intentional gesture will generate a similar range of intensities for the AUs each time
- An intentional gesture will occur for longer than an unintentional movement of the face

In terms of code, this would mean a gesture would be detected if the AUs lie within a certain range and held for longer than 2 seconds. Therefore, checks will be implemented to ensure that a gesture is only detected under the aforementioned conditions. An underlying assumption made in this however, is that OpenFace produces similar results under the same conditions with the same individual.

2.4.1 ROLE OF EMOTION IN GESTURES

The role of AUs in emotion was also considered as it is important not to confuse gestures with emotional expression. The relevant AUs are highlighted in bold. It can be seen in Table 5 that several of these AUs are involved in emotion. This can affect the reliability of the system as an emotional expression could be confused for a gesture.

While it can be assumed that emotional expressions generally wouldn't involve the same intensities involved in intentional gestures, this could be a source of error. Therefore, a measure that can be taken to reduce the likelihood of a gesture being sent as a command if they are actually displaying emotion or other expressions, is to ensure that it is only sent if one gesture is detected at a time. This is particularly applicable to the emotional expressions fear and surprise as they involve AUs that are used to detect the raised eyebrows or an open mouth. For disgust, further measures could also be taken.

TABLE 5: THE AU INVOLVED IN EMOTION (IMOTIONS, N/A)

Happiness / Joy	6 + 12	Cheek Raiser, Lip Corner Puller
Sadness	1 + 4 + 15	Inner Brow Raiser, Brow Lowerer, Lip Corner Depressor
Surprise	1 + 2 + 5 + 26	Inner Brow Raiser, Outer Brow Raiser, Upper Lid Raiser, Jaw Drop
Fear	1 + 2 + 4 + 5 + 7 + 20 + 26	Inner Brow Raiser, Outer Brow Raiser, Brow Lowerer, Upper Lid Raiser, Lid Tightener, Lip Stretcher, Jaw Drop
Anger	4 + 5 + 7 + 23	Brow Lowerer, Upper Lid Raiser, Lid Tightener, Lip Tightener
Disgust	9 + 15 + 16	Nose Wrinkler, Lip Corner Depressor, Lower Lip Depressor
Contempt	12 + 14 (on one side of the face)	Lip Corner Puller, Dimpler

Therefore, a summary of the features that will be implemented are:

- Detect gestures to occur within a certain range
- Count the number of occurrences (10+ times)
- Only allow one gesture to be detected at a time

The way in which the user interacts with the wheelchair will now be explored in the following section.

2.5 CONTROL INTERFACE

Using the considerations from the previous section, the control interface was implemented to acquire information from OpenFace and detect gestures.

This was implemented on Visual Studio 2015 in C++, through modifying the open-source code OpenFace. There were several executables that had various capabilities which could have been used as the basis for development. However, the Feature Extraction executable suited the requirements of the project the best as it was able to extract a variety of features from sequences of images. It also tracks one face at a time and can extract features in real-time which was two of the main criteria identified. This was modified and renamed Open Face Interpreter to suit this project.

The interface was initially written to provide the following:

- Option to choose gestures for various abilities
- Gesture detection
- Safeguards in place to ensure that movements are intentional
- Safeguards in place to ensure that only one command is sent at a time or none if more than one gesture is occurring simultaneously

The initial implementation is shown in the block diagram in Figure 17.

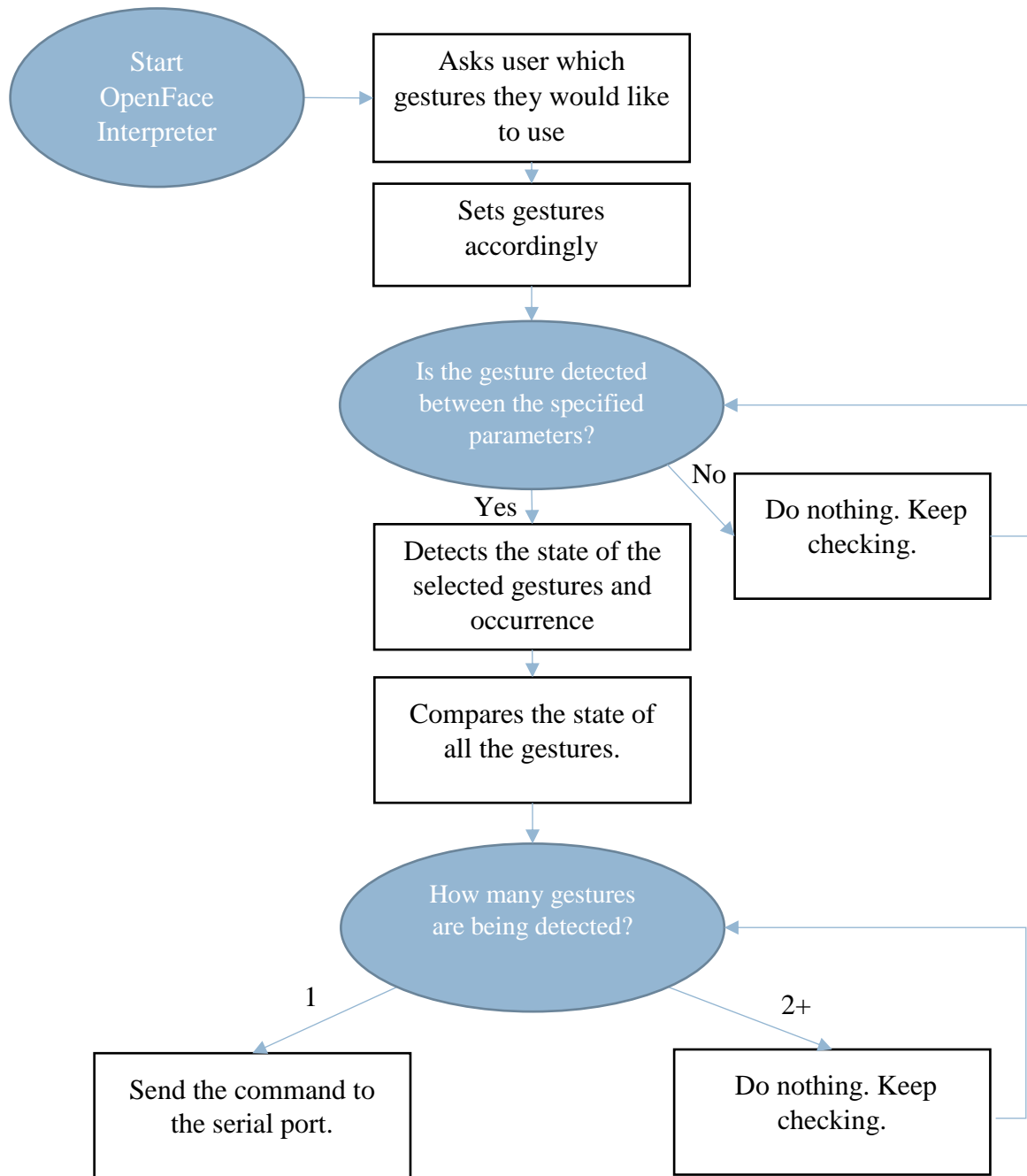


FIGURE 17: INITIAL IMPLEMENTATION OF OPEN FACE INTERPRETER

The initial gestures and the associated values that were implemented are shown in Table 6.

TABLE 6: INITIAL GESTURES AND VALUES IMPLEMENTED INTO OPEN FACE INTERPRETER

Gesture	Relevant feature	Range of detection
Raise Eyebrows	AU01	3.5+
	AU02	3.5+

Wrinkle Nose	AU09	3.5+
Open Mouth	AU25	3.5+
	AU26	3.5+
Lip Suck	AU28	1
Tilt Left	Tz	20 to 100
Tilt Right	Tz	-100 to 20

As the interface is heavily reliant on OpenFace and its performance, it is important that the way it works was understood so issues can be accounted for. There were a few limitations identified, one of which is that facial pose and orientation can have quite a big effect on the accuracy of detecting AUs. Ways to decrease these effects were considered and implemented.

This was then implemented with the wheelchair, using the existing Arduino Mega 2560 controller that had been integrated into the control of the wheelchair. This involved implementing serial communication between OpenFace and the Arduino. User evaluation of the operation of the wheelchair was also obtained to determine the ease of use and feasibility of the system.

3. RESULTS

This chapter will present results of how well the face and head tracking fared with different people and under different situations and conditions. The resulting implementation of the control interface and wheelchair interface will also be illustrated and evaluated.

3.1 FACE AND HEAD TRACKING

The face and head tracking system was explored by the author in order to gain insight to what parameters need to be put in place depending on different individuals and situations.

Preliminary testing of the author showed that there were 5 distinct gestures that would have higher chances of being detected in this application. These were raising eyebrows, wrinkling nose, frowning, opening mouth and lip sucking. The minimum and maximum values of the AU involved in the 10 gestures were compared under neutral conditions and with execution of the gestures. The data obtained was processed through Matlab and tabulated as shown in Table 7. The differences between the highest value of the neutral baseline and the gestures were calculated. This gives an indication of which AUs differ most from the neutral baseline when a gesture is performed. The AUs that had a significant difference of 1 and more have been highlighted in bold.

TABLE 7: MINIMUM AND MAXIMUM INTENSITIES OF AU IN NEUTRAL CIRCUMSTANCES AND IN GESTURES (N=1)

Neutral			Gestures					
			Short action		$\Delta max_{short-neutral}$	Long action		$\Delta max_{long-neutral}$
AU	Min	Max	Min	Max		Min	Max	
'AU01'	0	1.32	0	3.2	1.88	0	3.9	2.58
'AU02'	0	1.82	0	3.87	2.05	0	4.36	2.54
'AU04'	0	1.45	0	1.71	0.26	0	1.87	0.42
'AU05'	0	2.18	0	2.27	0.09	0	2.97	0.79
'AU07'	0	2.04	0	2.65	0.61	0	2.72	0.68
'AU09'	0	0.88	0	2.02	1.14	0	2.26	1.38
'AU14'	0	1.3	0	3.69	2.39	0	3.32	2.02
'AU15'	0	1.36	0	3.2	1.84	0	3.59	2.23

'AU17'	0	1.32	0	2.06	0.74	0	2.09	0.77
'AU23'	0	0.86	0	1.82	0.96	0	1.37	0.51
'AU25'	0	1.17	0	2.85	1.68	0	3.67	2.5
'AU26'	0	1.42	0	4.3	2.88	0	4.59	3.17
'AU28'	0	0	0	1	1	0	1	1

In this case, it appears that the AUs AU01, AU02, AU09, AU14, AU15, AU25 and AU26 differed the most from the neutral baseline. This means that the corresponding gestures were most likely to be detected:

- Raise eyebrows
- Wrinkle nose
- Frown
- Widen mouth
- Open mouth
- Lip suck

The following gestures were unlikely to be detected:

- Widening eyes
- Pursing lips
- Lowering eyebrows
- Squinting

Therefore, it can be concluded that these four gestures should not be used in the implementation of the gesture recognition system. However, the other 6 gestures can be investigated further to observe how intensities of each AU vary between different gestures.

The gesture Raise Eyebrows was investigated first. The data from the previous short and long tests could be graphed as shown in Figure 18 and Figure 19. It was found that the intensity was generally around 3-4 for the short actions and 2-4 for the longer actions. This indicates the range in which raise eyebrows was likely to occur. The gesture of wrinkling the nose appeared to be activate the AUs that are related to raising eyebrows, but at a smaller intensity as shown in Figure 20 . However, it is important to ensure that the range implemented to detect an intentional eyebrow

raise is outside of the intensity when wrinkling nose occurs. This means that the range where an intentional eyebrow raise would occur is between 2 and 4.

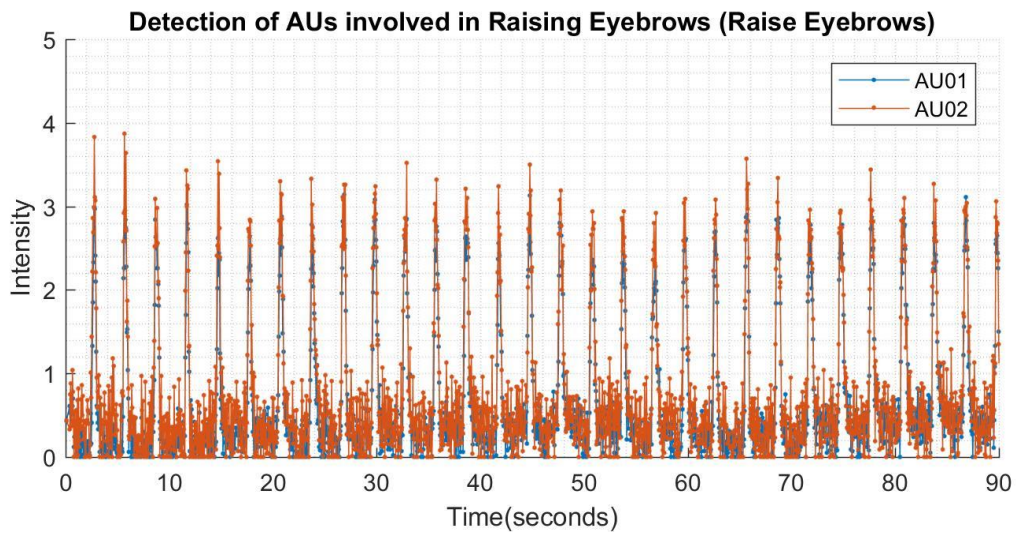


FIGURE 18: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

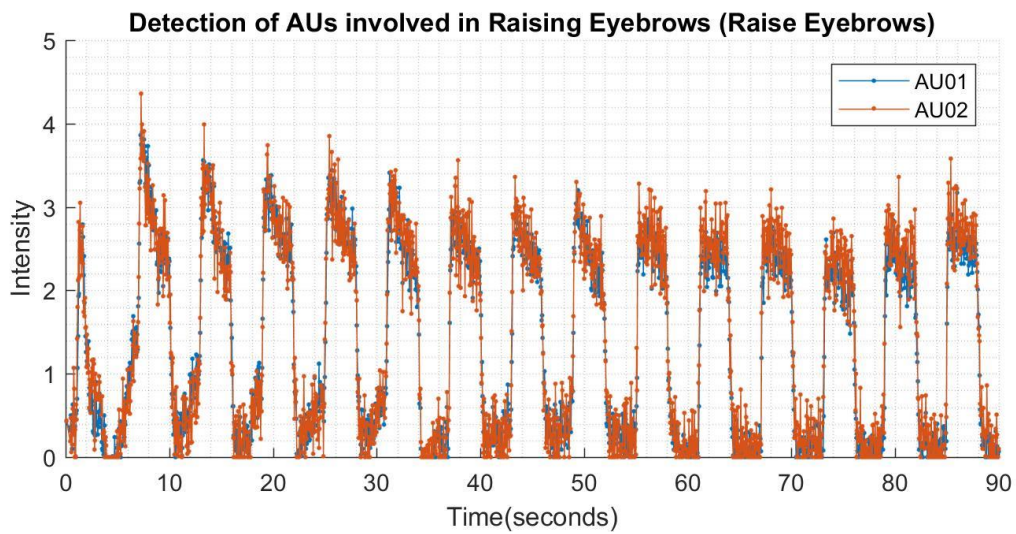


FIGURE 19: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

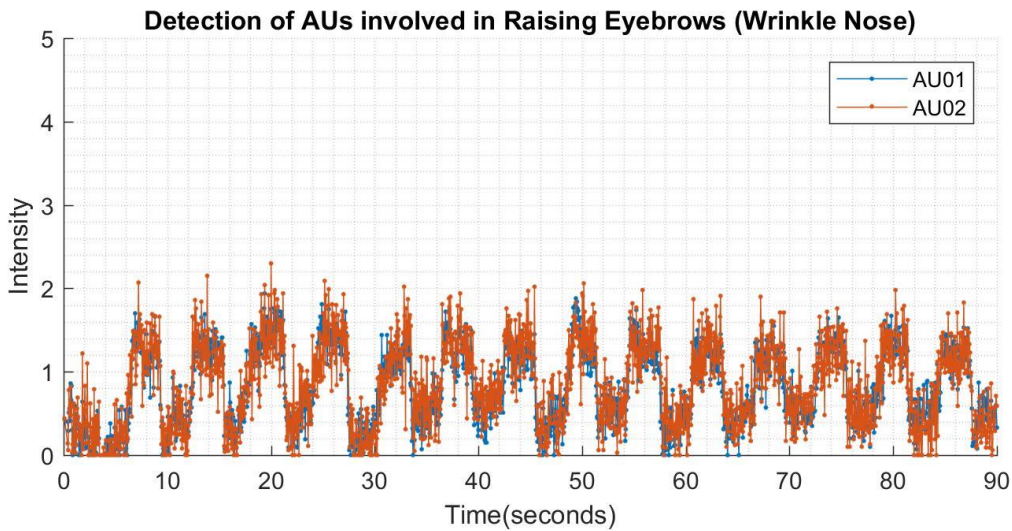


FIGURE 20: DETECTION OF AU01 AND AU02 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME.

Wrinkle nose was investigated next in a similar fashion. This gesture exhibited a lower intensity compared to Raise Eyebrows but was consistent as shown in Figure 21.

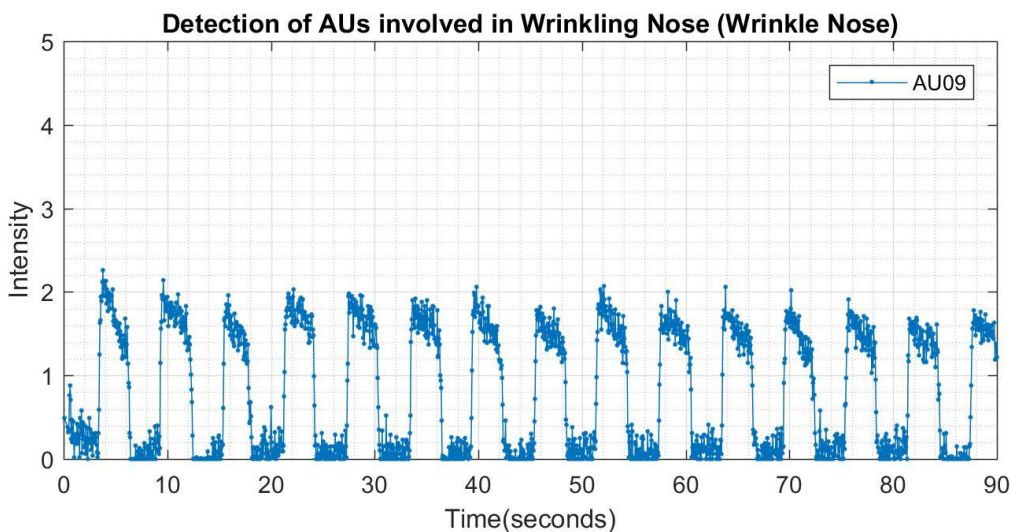


FIGURE 21: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME.

Open mouth was then investigated and illustrated in Figure 22 and Figure 23. The intensities involved in this gesture were quite high and consistent which suggests that this gesture would be easy to detect. It also shows that AU25 is more strongly detected compared to AU26. There were a few other gestures that appeared to be

activate these AUs, but the intensities were not significant. These are shown in Appendix A to allow for a direct comparison.

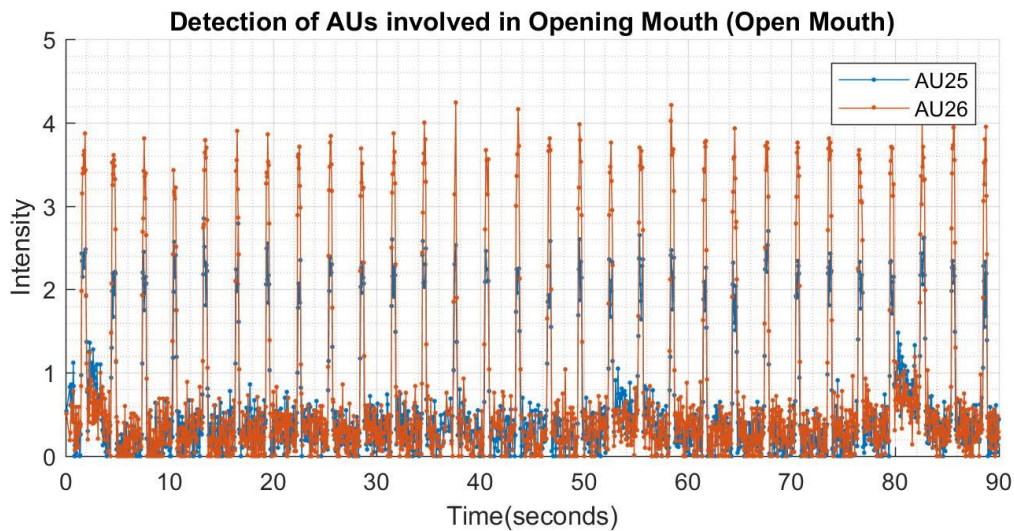


FIGURE 22: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT DURATIONS OF TIME.

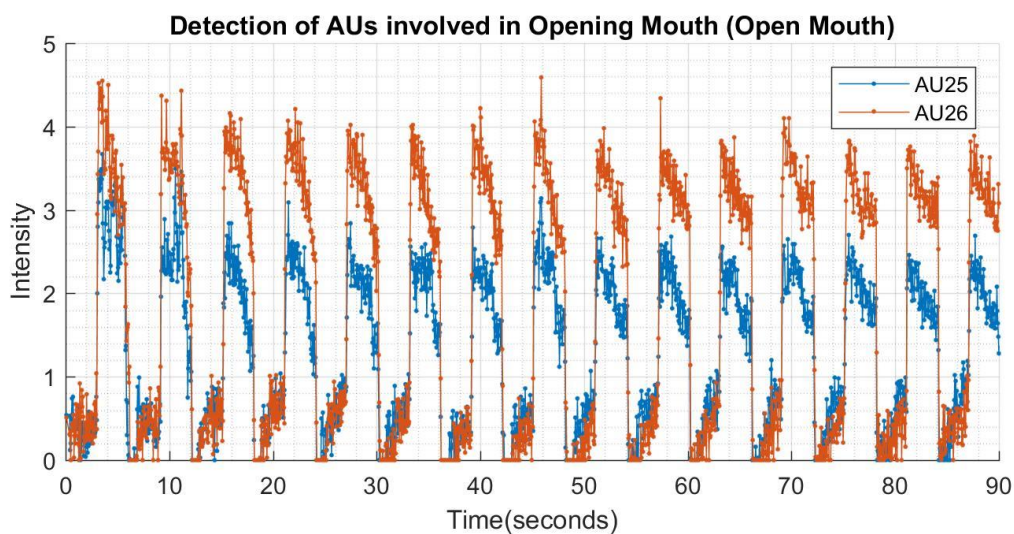


FIGURE 23: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG DURATIONS OF TIME.

The gesture widen mouth was then investigated. However, it was found that the AU in widen mouth was being detected in other actions such as lip suck at similar intensities. Figure 24 and Figure 25 show that the intensities generally range around 3 when widen mouth is performed for both short and long durations of time. When

the gesture lip suck is performed, it can be observed that AU14 is also detected slightly lower but also around the same range. These graphs can be observed in Appendix A. This is too similar and means that either widening mouth or lip suck should be used and not both as it is likely they will be detected at the same time. Widen mouth is a gesture that could potentially be mistaken in a smile, which means lip suck would probably be the more appropriate gesture to use.

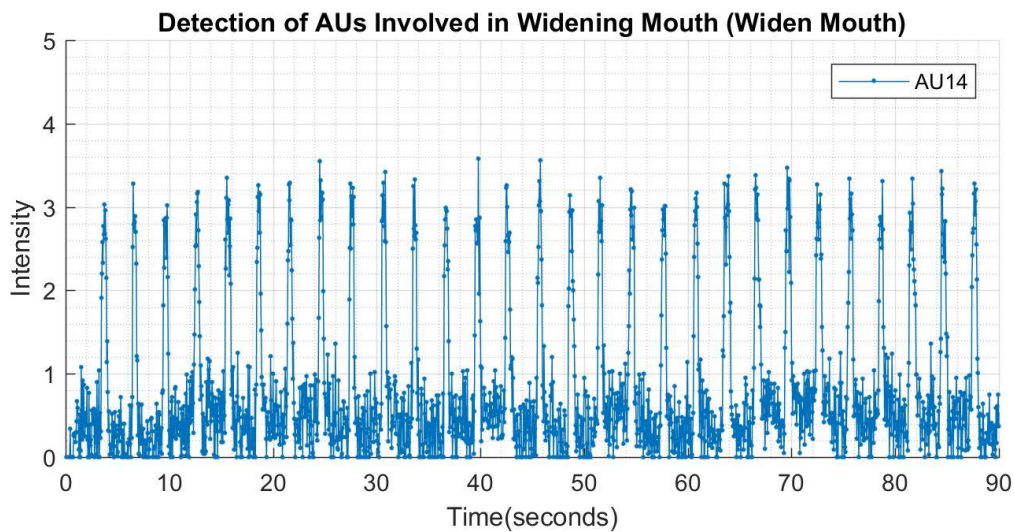


FIGURE 24: DETECTION OF AU INVOLVED IN WIDENING MOUTH WHEN WIDEN MOUTH GESTURE IS PERFORMED FOR SHORT DURATIONS OF TIME

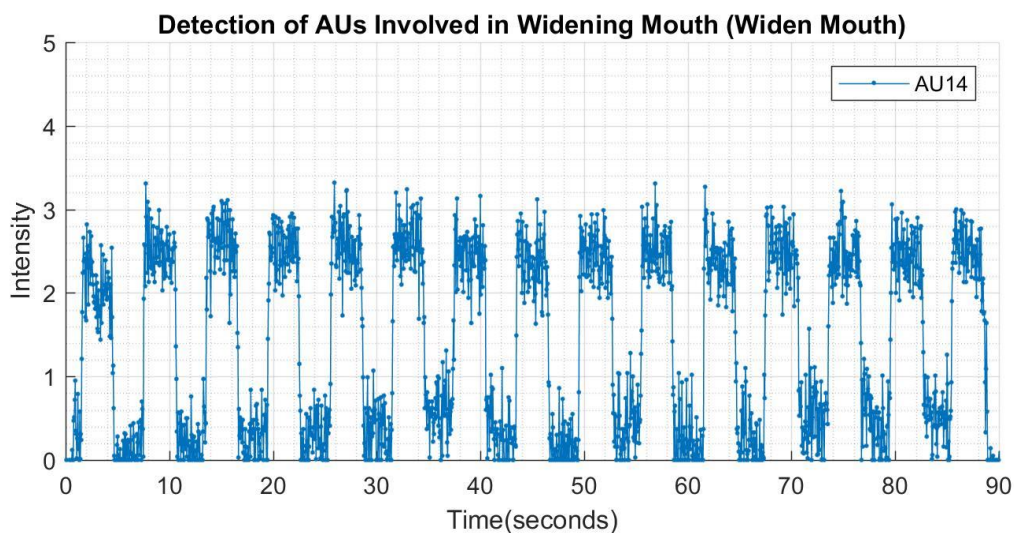


FIGURE 25: DETECTION OF AU14 WHEN WIDEN MOUTH GESTURE IS PERFORMED FOR LONG DURATIONS OF TIME

Frowning was also investigated, where it was found that AU15 occurs periodically in frowning, while AU17 did not appear to be related. During the gesture lip suck, it appeared that AU17 was detected frequently while AU15 occurred at a lower intensity. The gestures wrinkling nose and widening mouth also appear to activate these AUs. From this, it can be concluded that AU15 would be the more reliable choice for detecting frowning. However, due to the similarities of intensities it shares with other gestures such as lip suck, it would probably not be ideal for use as a gesture. These comparisons can be seen in Appendix A.

Lip suck was investigated as well, where it was found that it was quite reliable as long as it is performed in short periods of time as shown in Figure 27. It appeared to decrease in intensity when held for longer periods of time as shown in Figure 27. This AU was not activated in any other gesture, which is promising in this context as this gesture would hopefully not attract any false positives.

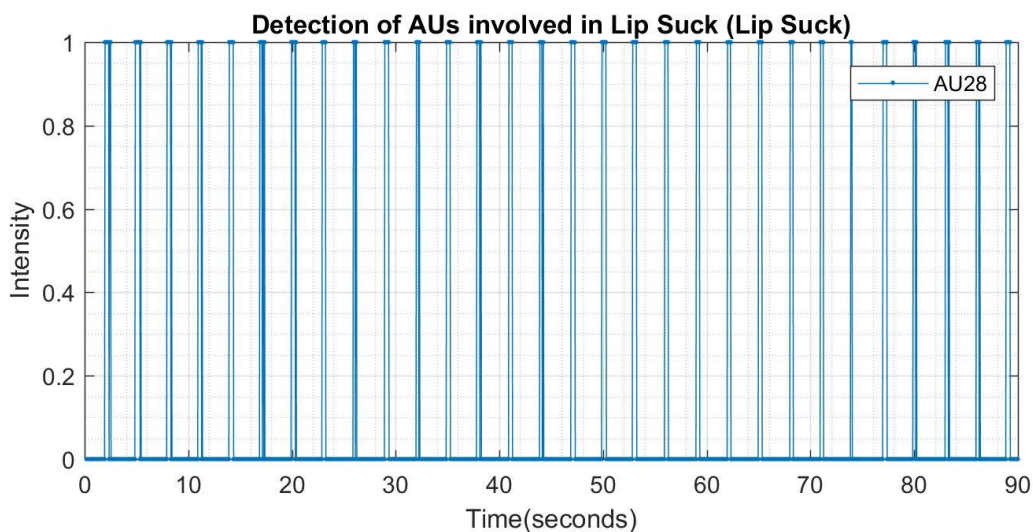


FIGURE 26: DETECTION OF AU28 WHEN LIP SUCK GESTURE IS PERFORMED FOR SHORT DURATIONS OF TIME

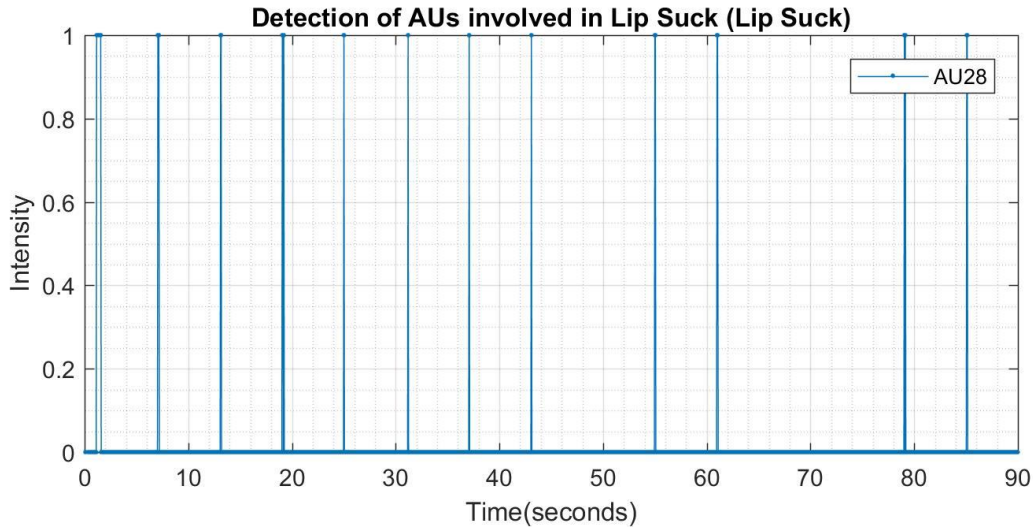


FIGURE 27: DETECTION OF AU28 WHEN LIP SUCK GESTURE IS PERFORMED FOR LONG DURATIONS OF TIME

From these tests, it was clear that individual parameters needed to be set for each gesture and that different AU's are affected by each gesture. Because of this, frowning and widening mouth were not included in the final set of gestures that were implemented. Initial parameters were also found for these particular set of gestures using the graphs, which is shown in Table 8. The absolute lower limit was obtained from the highest value found in the neutral tests. The ranges for the short actions and long actions were estimated directly from the graphs. These parameters can be used in the implementation of the gesture recognition system.

TABLE 8: PARAMETERS FOR EACH GESTURE, USING THE MAX VALUE FOR NEUTRAL FOR THE ABSOLUTE LOWER LIMIT AND ESTIMATING THE RANGES OF THE SHORT AND LONG ACTIONS

AUs	Absolute lower limit	Limits for short actions		Limits for long actions	
		Min	Max	Min	Max
AU01	1.32	2.6	3.4	1.6	4.1
AU02	1.82	2.6	4	1.6	4.5
AU09	0.88	1.2	2.2	1	2.6
AU25	1.17	1.4	3	1.4	3.8
AU26	1.42	3.2	4.4	2.3	4.7
AU28	0	0	1	0	1

A common theme between the long activations of gestures was also observed. Lip suck was a prominent example, but it appeared that all the AUs decreased in intensity over time. This could be due to difficulties in holding the same gesture for a

prolonged period of time or more likely, a decrease in sensitivity. This would need to be accounted for.

3.1.2 TESTING PARTICIPANTS

Similar tests were performed on 6 different able-bodied participants in order to observe how they would perform the selected gestures relating to the AUs. It was found that each subject performed the gestures differently, each utilising different AUs. Some participants were also unable to perform certain gestures due to difficulties in coordination and physical movement. This underlines the need for a versatile system. The range of detection found for some gestures appeared to be relatively similar, which means a general parameter could possibly be implemented into the system. These are shown in Appendix A.8. Otherwise, an individualised range for the detection of gestures may be better suited for each subject.

TABLE 9: SUMMARY OF RANGE OF DETECTIONS ACROSS PARTICIPANTS 1-6

AUs	Subject 1		Subject 2		Subject 3		Subject 4		Subject 5		Subject 6	
	<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	2	3.8	2.2	4.4	2	4.5	2	4.9	2	3.6	1.8	5
AU02	2	3.8	2.2	4.2	2	4.5	1.8	4.2	1.8	4.1	1.5	4.5
AU09	0.8	2.2	2.4	3.3	2	3.4	0.8	1.7	1.9	3.4	0.8	2.7
AU25	2	4.6	2.4	4.95	1.6	4.8	1	3.9	2.8	5	1.4	3.9
AU26	1.6	3.85	3.4	4.8	2	4.7	2	5	3	5	3	5
AU28	-	1	-	1	-	1	-	1	-	0	-	1

The average range across all participants as found from Table 9 is:

- AU01: 2 – 4.36
- AU02: 1.88 – 4.36
- AU09: 1.45 – 2.78
- AU25: 1.86 – 4.53
- AU26: 2.5 – 4.73
- AU28: Not applicable, can only be 0 or 1

This demonstrates the range that the system would have to be calibrated to. It was also found that there is variation in intensities for the AU activated for each gesture. Lip suck was also not reliably detected after the first few detections. It was also noted that lip suck was not registered in the system when subject 5 performed the

gesture. This could be due to difficulties in tracking the landmarks of the face due to facial variations.

An initial prototype was made of the control interface that involved the gestures that could be performed easily and were distinct from expression and emotions. This included raise eyebrows, wrinkle nose, open mouth and lip suck. Preliminary measures that were taken included the detection of occurrence (how many times the gesture was detected consecutively) and the threshold (any AU intensity above 3.5 would be considered detected). However, this was not very reliable and useful and so was investigated further through the following procedure.

3.1.2 SCENARIOS

Several different scenarios were tested, one in which conversation was occurring and another where general movements of the face were recorded. During conversation it was clear that a high number of false positives were being detected. This was compared with the video and it was found that it occurred mainly when the head moved to a non-forward-facing position. This is a limitation that is common in facial tracking systems. When this occurred, the AU was shown to reach intensity level 5. Therefore, it was concluded that an upper limit would need to be found for each intentional gesture and a safeguard put into place to accept only inputs when the user is forward facing.

3.1.3 CONDITIONS

There has been no noticeable decrease in performance under different lighting conditions. This program has been used in a variety of lighting conditions including low light, artificial light and natural lighting.

3.2 CONTROL INTERFACE

Using the information gained from exploring the face and head tracking, the control interface was implemented to have the conditions mentioned in section 2.4.

Initial testing of the program showed that lip suck was being mistaken for the gesture open mouth most of the time. This was partly due to the fact that the program was originally written to accept one AU which was chosen based on the preliminary tests. However, it was clear that after testing this was not the case for everyone and that AU for jaw drop, was also occurring with lip suck, albeit at a smaller intensity.

Therefore, after testing it was concluded that there were several adjustments that needed to be made to make the system more user friendly, increase the likelihood of the detection of an intentional gesture and to reduce the detection of false positives. These included:

- Inclusion of default gestures chosen for commands with natural interface considerations, i.e. tilt head left for going left and vice versa.
- Implementing all the parameters for each gesture – additional AU involved
- Reducing the occurrence of false positives by ensuring gestures are only detected when directly facing forward.
- Lip suck was not being picked up over time, so the number of occurrences was decreased to 5 as opposed to 10 for this gesture.

The modified system is shown in the block diagram in Figure 28.

The range of detection used was determined previously in section 3.1. The following gestures are detected using the following AUs and extracted features as shown in Table 10. The up/down and side/side values are used to check whether the user is facing forward. Gestures are only detected when Tx and Ty are within these bounds.

TABLE 10: FINAL GESTURES AND PARAMETERS IMPLEMENTED

Gesture	Relevant feature	Range of detection
Raise Eyebrows	AU01	2.6 to 4.1
	AU02	2.6 to 4.5
Wrinkle Nose	AU09	1.2 to 2.6
Open Mouth	AU25	1.4 to 3.8
	AU26	3.2 to 4.7
Lip Suck	AU28	1
Tilt Left	Tz	20 to 100
Tilt Right	Tz	-100 to 20
Up/down	Tx	-15 to 15
Side/side	Ty	-15 to 15

The final block diagram of the interface is shown in Figure 28.

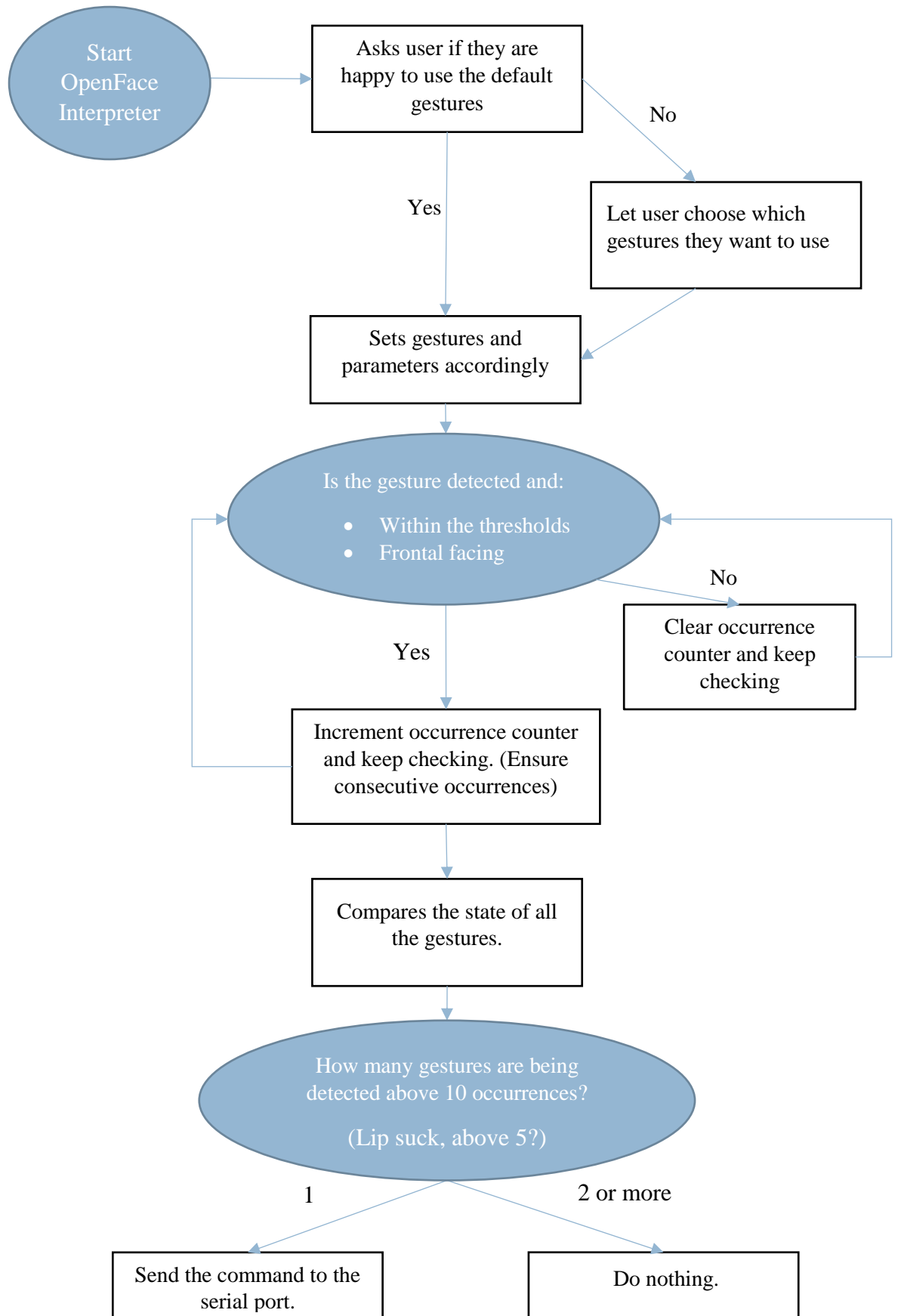


FIGURE 28: MODIFIED IMPLEMENTATION OF OPEN FACE INTERPRETER APPLICATION

Once the command is sent to the serial port, the corresponding action is activated in the wheelchair. The user has the option of choosing the default gestures for the commands or choosing their own as illustrated in Figure 29 and Figure 30 respectively. The code inside Open Face Interpreter is shown in Appendix B.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\vivia> cd Documents\OpenFace\OpenFace2.04Viv2\OpenFace-OpenFace_2.0.4\x64\Release
PS C:\Users\vivia\Documents\OpenFace\OpenFace2.04Viv2\OpenFace-OpenFace_2.0.4\x64\Release> .\OpenFaceInterpreter -device
0
Connection Established

Hello, are you happy to use the following gestures for these commands (Y/N)?

STOP: Open Mouth
FORWARD: Raise Eyebrows
BACKWARD: Wrinkle nose
LEFT: Tilt head left
RIGHT: Tilt head right
y

These are the gestures chosen:
STOP: open mouth
FORWARD: raise eyebrows
BACKWARD: wrinkle nose
LEFT: tilt head left
RIGHT: tilt head right

```

FIGURE 29: CONTROL INTERFACE FOR OPENFACEINTERPRETER IN THE DEFAULT GESTURE CASE

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\vivia> cd Documents\OpenFace\OpenFace2.04Viv2\OpenFace-OpenFace_2.0.4\x64\Release
PS C:\Users\vivia\Documents\OpenFace\OpenFace2.04Viv2\OpenFace-OpenFace_2.0.4\x64\Release> .\OpenFaceInterpreter -device
0
Connection Established

Hello, are you happy to use the following gestures for these commands (Y/N)?

STOP: Open Mouth
FORWARD: Raise Eyebrows
BACKWARD: Wrinkle nose
LEFT: Tilt head left
RIGHT: Tilt head right
n

These are your available gestures:
0. Raise Eyebrows
1. Wrinkle nose
2. Open mouth
3. Lip suck
4. Tilt head left
5. Tilt head right

What would you like to use for STOP ?
3
What would you like to use for FORWARD ?
2
What would you like to use for BACKWARD ?
0
What would you like to use for LEFT ?
4
What would you like to use for RIGHT ?
5

These are the gestures chosen:
STOP: lip suck
FORWARD: open mouth
BACKWARD: raise eyebrows
LEFT: tilt head left
RIGHT: tilt head right

```

FIGURE 30: CONTROL INTERFACE FOR OPENFACEINTERPRETER WHERE USER CHOOSES GESTURES

3.2.1 PERFORMANCE

The performance of this recognition system was then tested on the author with the parameters found from the previous gestures tests and placed into a contingency table to compare the number of predicted gestures versus the actual gesture. This was tested while facing forward to minimise the effects of orientation on the gesture recognition and for consistency. Each test was performed 10 times each. The result of this is shown in Table 11. It was found that raise eyebrows was detected while head tilt to the left occurred. It is noted that multiple gestures were detected when one gesture was being performed at times, which resulted in more than 10 actual identified gestures.

TABLE 11: CONFUSION MATRIX FOR CORRECTLY AND INCORRECTLY IDENTIFIED GESTURES

		Actual						
		<i>Nothing</i>	<i>Raise Eyebrows</i>	<i>Wrinkle Nose</i>	<i>Open Mouth</i>	<i>Lip Suck</i>	<i>Tilt Left</i>	<i>Tilt Right</i>
Predicted	<i>Nothing</i>	10	0	1	1	0	0	0
	<i>Raise Eyebrows</i>	0	10	0	0	0	0	0
	<i>Wrinkle Nose</i>	0	0	10	0	0	0	0
	<i>Open Mouth</i>	1	1	0	8	0	0	0
	<i>Lip Suck</i>	0	0	0	0	10	0	0
	<i>Tilt Left</i>	0	5	0	0	0	10	0
	<i>Tilt Right</i>	0	1	0	0	0	0	10

This was analysed further with Matlab to give the following scores of informedness, which is the “probability of an informed decision” and markedness, which is defined by “how marked a condition is for the specified predictor” (Powers, 2011). These are measures of how well the system predicts the gestures. RandAccuracy provides the ratio between the number of classes that were predicted accurately against the total number of cases. The values shown are all around 87% which indicates that in this test, the classifier showed quite promising results.

- Informedness: $0.8828 = 88.28\%$
- Markedness: $0.8707 = 87.07\%$
- RandAccuracy: $0.8718 = 87.18\%$

This was also tested on subject 5 to observe differences in the classification. Ethics were not sought as the information acquired was for system quality testing purposes. The confusion matrix of their gestures was obtained from this test. It was shown that it was not as reliable, particularly for the open mouth gesture which was not detected in all 10 tests.

TABLE 12: CONFUSION MATRIX FOR CORRECTLY AND INCORRECTLY IDENTIFIED GESTURES – SUBJECT 3

		Actual						
		<i>Nothing</i>	<i>Raise Eyebrows</i>	<i>Wrinkle Nose</i>	<i>Open Mouth</i>	<i>Lip Suck</i>	<i>Tilt Left</i>	<i>Tilt Right</i>
Predicted	<i>Nothing</i>	5	3	1	1	0	0	0
	<i>Raise Eyebrows</i>	0	10	0	0	0	0	0
	<i>Wrinkle Nose</i>	0	0	10	0	0	0	0
	<i>Open Mouth</i>	10	0	0	0	0	0	0
	<i>Lip Suck</i>	0	5	0	0	10	0	0
	<i>Tilt Left</i>	0	0	0	0	0	10	0
	<i>Tilt Right</i>	2	0	0	0	0	0	8

In this case, the three performance measures taken were significantly lower. This means that for this subject, the gesture classification performed significantly worse. This could be due to various factors, including a mismatched range of detection, issues tracking the face or inconsistent gesture performance by the individual.

- Informedness: $0.6523 = 65.23\%$
- Markedness: $0.7337 = 73.37\%$
- RandAccuracy: $0.7067 = 70.67\%$

However, in the graphs obtained of the gestures it appeared that open mouth for this subject was detected to occur within quite a wide range which suggests that the tracking capability of OpenFace could be the source of error as opposed to the classification framework or the subject. The AU intensities for short and long actions for this subject are shown in Figure 31 and Figure 32. It is observable that AU25 and AU26 differ vastly in intensity with the performance of the same gesture.

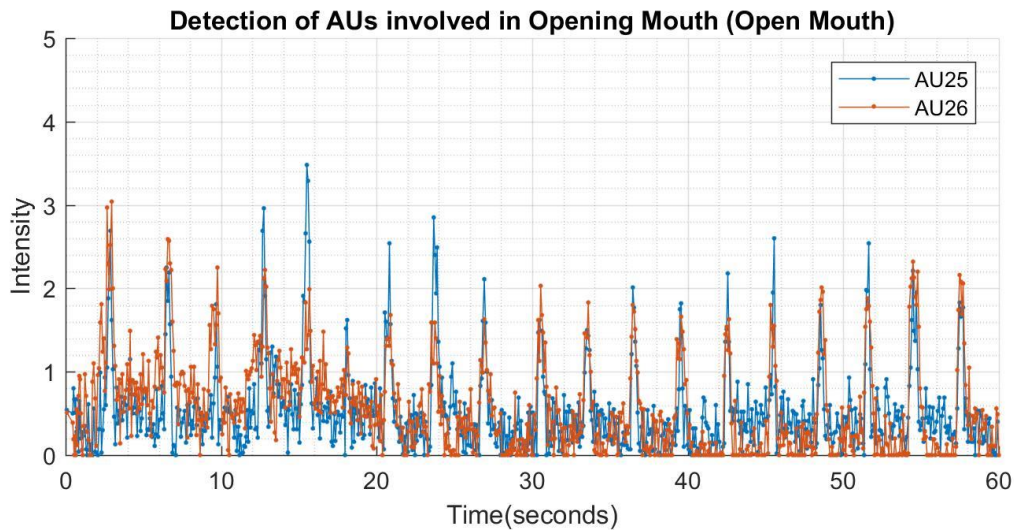


FIGURE 31: DETECTION OF AU25 AND AU26 DURING THE GESTURE OPEN MOUTH PERFORMED PERIODICALLY FOR A SHORT PERIOD OF TIME ON SUBJECT 3

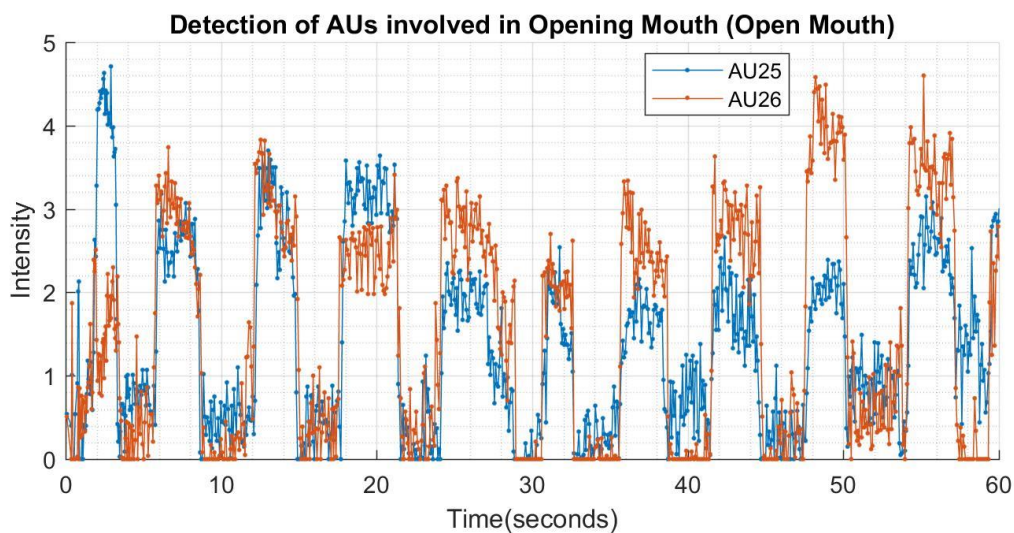


FIGURE 32: DETECTION OF AU25 AND AU26 DURING THE GESTURE OPEN MOUTH PERFORMED PERIODICALLY FOR A LONG PERIOD OF TIME ON SUBJECT 3

It appears that the ideal gestures to use in this application vary from person to person. Some gestures are unlikely to be detected for some compared to others based on how consistently OpenFace can track their features. In this case, subject 3 was not able to use open mouth as it was very unreliable while this was not the case for others.

FaceSwitch (Rozado et al., 2017) found that the most easily recognised gestures were open mouth and wrinkling nose, based on their system using Beyond Reality Face Tracker. However, it appears that for this program at least, it is quite variable.

3.3 WHEELCHAIR INTERFACE

The wheelchair uses a joystick control that can be controlled through the Arduino. The joystick is controlled by sending different values to forward/backward and left/right that represent the extent of power it should be given. Changing these values can be used to control the wheelchair through gestures. Information is sent to the serial port and received by the Arduino to follow through with the appropriate action. The serial communication (Mandal, 2016) and Arduino control (Kukreja, 2018) was integrated into this project as shown in Appendix C and D respectively.

This system can take inputs from face and head gestures and use these for a stop, forward, backward, left and right command. Once a gesture is executed, it is sent to the wheelchair through the serial port if the correct conditions are in place.

This was tested with the wheelchair and confirmed to be working. A secondary control had been implemented as a safety feature (Kukreja, 2018), where the joystick can override any input sent through the gesture recognition system. For an additional safety feature, a physical switch could also be implemented and connected to the Arduino. Therefore, the user would be able to control whether the wheelchair accepts inputs from the face and head gesture control system. In this way, the face and head gesture control system could be switched on or off and could also be used as an emergency stop switch.

On the wheelchair, there were a few errors with the detection of the gestures. Some gestures did not appear to be detected well but were detected reliably on another run under the same conditions. The program also lagged and froze occasionally when continuous input was given. Without altering the tools used behind the facial behaviour analysis, a method in which the system could be improved is to adjust the range of detection for the different gestures.

This interface was tested on a fellow colleague to obtain a subjective evaluation of the system. The results are as follows:

How safe did you feel?

Very unsafe	Somewhat unsafe	Neutral	Safe	Very safe
1	2	3	4	5

How intuitive did the input method feel?

Very unintuitive	Somewhat unintuitive	Neutral	Intuitive	Very intuitive
1	2	3	4	5

How accurate were the controls?

Very inaccurate	Somewhat inaccurate	Neutral	Accurate	Very accurate
1	2	3	4	5

How responsive were the controls?

Very unresponsive	Somewhat unresponsive	Neutral	Responsive	Very responsive
1	2	3	4	5

Which controls did you find easiest to use?

Forward (eyebrows), Tilting

How comfortable was the input method to use?

Very uncomfortable	Somewhat uncomfortable	Neutral	Comfortable	Very uncomfortable
1	2	3	4	5

How tired do you feel?

Very tired	Somewhat tired	Neutral	Not really tired	Not tired at all
1	2	3	4	5

Did the input method give you any muscle fatigue?

Yes	Somewhat	Neutral	Not really	No
1	2	3	4	5

Any other comments?

It's very good but holding the gestures felt too long and the left and right controls could be changed to turn slower. With more training it could potentially be very good.

Based on this feedback, it appears there are quite a few improvements that could be made to increase the reliability and ease of control. This could involve lowering the number of occurrences necessary so that the delay in gesture to action is decreased. Adjusting the parameters as mentioned could also increase the reliability.

However, for both these methods, there would be a trade-off between the certainty that a gesture is intentional and the system being more responsive. However, they felt relatively safe using the device, which is important in a device like this.

It is also common in NUIs to undergo training in order to feel more comfortable with the controls. In this case, it is probable that with repeated use of these gestures and commands the interface would feel more intuitive for the user.

4. DISCUSSION

4.1 SIGNIFICANCE OF RESULTS

In this thesis, it has been demonstrated that it is possible to use both face and head gestures in the control of a smart wheelchair with a simple implementation of open-source software OpenFace and a standard webcam. 6 different gestures were implemented and examined from the user's perspective as well as through a technical perspective. These were Raise Eyebrows, Wrinkle Nose, Open Mouth, Lip Suck, Tilt Left and Tilt Right. It was found that some gestures were more reliable and easier to perform while others weren't. This varied depending on the person, but generally tilting the head to the left or right appeared to be detected well. These two gestures are the only head gestures of the implemented set and rely on head pose to obtain the values necessary. The differences in reliability could be due to individual differences in facial features, which can affect the tracking and other processes behind the detection of AUs.

The assortment of gestures was particularly useful when it was found that some gestures function better or were more easily detected depending on the individual. This enables the user to pick and choose which gestures work with their own capabilities and suit them best. This is particularly important in this application as this device is aimed towards those who have limited movement. This aspect of the design worked well as it made the system versatile, which was one of the main criteria that was identified in the project requirements.

Default gestures were implemented as a way to lessen the load on the user if they did not want to go through the trouble of picking and remembering certain gestures for the command of the wheelchair. These were chosen based on what would seem natural to the user. This included tilting left to go left, tilting right to go right, raising eyebrows as forward, wrinkling nose as reverse and open mouth as stop.

User feedback stated that it did not feel unintuitive or intuitive. However, it's common for NUIs like this one to require user training to become more familiar with the controls as demonstrated by Rozado et al. (2017). While this reduces the useability, as it would be ideal for the interface to be natural enough not to warrant training, this

is a common occurrence with NUIs. After an initial learning curve, the input method should feel very natural for the user.

The informedness of the gesture recognition system was found to range between 65.23% and 88.28%. This range is satisfactory, but this means that the prediction of the gesture classification system is making informed decisions.

OpenFace has its limitations like any other piece of software. However, in this implementation it proved to be an effective toolkit for obtaining facial behaviour in a low-cost and accessible way. While proper evaluation of different facial behaviour analysis toolkits was outside of the scope of this paper, this implementation proves that expensive hardware is not necessary to implement a face and head gesture classifier. However, this project demonstrates the successful application of OpenFace to a challenging mobility problem. This can be used as a basis to encourage others to explore similar ideas.

4.2 LIMITATIONS

There are a few limitations that have been identified with this system, regarding the software as well as the practical usage.

Facial expression and movements are inherently difficult to quantify and measure. While a lot of progress has been made in this field, it is still relatively far from being accurate. However, OpenFace performs relatively well in this area as it was able to be used in this application effectively with a few adjustments. While this software is continuously being improved and developed by Tadas Baltrusaitis (Baltrusaitis et al., 2018) and other researchers, there are several improvements that can be made to improve reliability and accuracy - particularly in the case of where the person is not frontal facing. Methods that could potentially be used to improve the software is the use of more training data, particularly those that include more diverse poses and AU labelling. A 3D model database could be particularly useful to improve landmark tracking of a non-frontal facing person. More specialised datasets could also be created for this purpose, which could be obtained through the Intel RealSense cameras. In this way, the software could potentially classify gestures directly rather than through the indirect pathways of determining AU and the gestures they relate to.

Other limitations of software include the inherent challenges in computer vision, namely relating to occlusion and lighting. If a camera is unable to pick up certain features because they are hidden or by objects or unclear due to lighting, it can be difficult to ascertain the relevant features.

While it has been proven that this system can be used, this particular system is limited to be a proof of concept as it is not reliable enough for an individual to use for their mobility needs. More improvements to the system and further testing would need to be performed for it to be reliable and safe enough for use. It is also important for the end users to be involve in the development process. However, this would be costly in terms of time and resources and would be better suited to a larger funded project.

Intention was a major challenge faced in this study, and other studies on gesture recognition. This thesis addressed the intention challenge through the implementation of assumptions but further measures to deduce intention could include the use of multi-modal inputs. However other assumptions could be introduced, but this would limit the ways that a user can express a command. Extra information could be provided from the other aspects of the wheelchair such as the BCI in order to differentiate an intentional action from an unintentional action. Further studies could also be done to explore the ways people exhibit intentional gestures as opposed to unintentional movements.

In practical use, it may be difficult for the user to remember the gestures chosen for each command. This can be considered a limitation as it may be difficult for those with cognitive impairments or bad memory. However, this could be improved by making the system more intuitive and training control of the wheelchair with the user. This could also be helped through the addition of a legend that displays which gestures are for which command. In terms of intuitiveness, this may not be the best option. However, the aim of this project was to provide alternative options in terms of control which has been implemented.

In this project, there are several stages that could be improved. Formal experiments to explore user satisfaction could have been performed as well as further comparisons between different tests to obtain a bigger picture of the performance of the system. This could also influence the way the system operates, as it is likely that

further factors would need to be considered for people with disabilities in practical use.

5. CONCLUSION AND FUTURE WORK

The continued exploration of smart wheelchairs and alternative methods of control provides many benefits, not only to the subsets of their technological fields but also for people with mobility impairments. The technology could make regular powered wheelchairs easier and safer to use as well as provide alternative methods of control. However, this technology could particularly benefit those that cannot operate standard powered wheelchairs due to their cognitive or physical impairments.

In this thesis, one alternative method of control was explored – the use of face and head gestures in the control of a smart wheelchair. This was chosen due to the relatively underexplored combination of face and head gestures in the use of control interfaces as well as its versatility and intuitiveness for the user. It was concluded to be a viable interface and while there are quite a few improvements that could be made, the opportunities that this technology opens up are countless.

The main challenges faced in this implementation included the difficulties in deducing user intention as well as limitations faced in terms of the software. Exploring methods to deduce the intention of the user can be explored further through practical research as it is likely that regular movement can be mistaken for a gesture. It could also be improved further through the use of additional inputs to increase the amount of information obtained from the user. The current implementation of BCI could potentially be used in this way. With these improvements, the likelihood of predicting the user's intention correctly could be increased. Challenges faced in software included those that relate to computer vision as well as difficulties faced in face tracking. Lighting, occlusion and individual variances in facial appearance and movements influence the performance. This could potentially be improved through the integration of 3D data, such as those from the Intel RealSense cameras or through the creation of more specialised datasets that can address some of these issues.

In terms of future work, there are a few parallels between control interfaces for smart wheelchairs as well as for computer access. While the control interfaces of a smart wheelchair may not always be designed to connect the user with a computer, the applications are still highly applicable. This could also potentially be applied to any HCI and could be used to control a computer. This is one of the next stages that will

be explored as it would provide an alternative method for control for a wide variety of applications.

For the proper development of a functional interface for a smart wheelchair, it is important that the end users are involved. This way, the wheelchair and its interface can be evaluated at every stage of development to ensure the most natural and intuitive interface. In addition to involvement of the end users, more test participants could also have been involved in the evaluation of performance and the function of the wheelchair.

The results of developing a system that can recognise facial and head gestures could also potentially be used in the context of interpreting emotions. In terms of the wheelchair, this could lead to more intuitive control options and safety measures. Emotional responses could be considered from the AUs that are present at one time. This could be used in the wheelchair interface to predict and provide suggestions for immediate action. For example, if the user is looking fearful, it could provide options to call 000, a friend or family member. In other applications, this could lead to developments into measuring human responses to stimuli, such as during online learning.

In conclusion, while this technology is still in its early stages, there are many opportunities that this technology can bring. Improving existing technology in the field of facial analysis and learning more about the people that can use this technology and exploring how this information might be used are just a few of the next steps that can be taken.

REFERENCES

- ABNER, N., COOPERRIDER, K. & GOLDIN-MEADOW, S. 2015. Gesture for Linguists: A Handy Primer. *Language and linguistics compass*, 9, 437-451.
- AFFECTIVA. N/A. *Emotion SDK* [Online]. Affectiva. Available: <https://www.affectiva.com/product/emotion-sdk/> [Accessed 10 October 2018].
- ASAYESH, S. 2013. *Electronics Design of Brain Controlled Wheelchair*. Master of Engineering (Electronics), Flinders University.
- AUSTRALIAN BUREAU OF STATISTICS 2015a. A Profile of People With Disability in Australia. In: AUSTRALIAN BUREAU OF STATISTICS (ed.). Australia.
- AUSTRALIAN BUREAU OF STATISTICS 2015b. Use of Aids and Equipment by People With Disability in Australia. In: AUSTRALIAN BUREAU OF STATISTICS (ed.). Australia.
- AUSTRALIAN BUREAU OF STATISTICS 2016. 4430.0 - Disability, Ageing and Carers, Australia: Summary of Findings, 2015. In: STATISTICS, A. B. O. (ed.). Canberra: Australian Bureau of Statistics.
- BALTRUŠAITIS, T., MAHMOUD, M. & ROBINSON, P. Cross-dataset learning and person-specific normalisation for automatic Action Unit detection. 2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 4-8 May 2015 2015. 1-6.
- BALTRUŠAITIS, T., ROBINSON, P. & MORENCY, L. P. OpenFace: An open source facial behavior analysis toolkit. 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), 7-10 March 2016 2016 Lake Placid, NY, USA. 1-10.
- BALTRUSAITIS, T., ZADEH, A., LIM, Y. C. & MORENCY, L. OpenFace 2.0: Facial Behavior Analysis Toolkit. 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), 15-19 May 2018 2018. 59-66.
- BANKAR, R. T. & SALANKAR, S. S. Head Gesture Recognition System Using Gesture Cam. 2015 Fifth International Conference on Communication Systems and Network Technologies, 4-6 April 2015 2015 Gwalior, India. 535-538.
- BARTOLEIN, C., WAGNER, A., JIPP, M. & BADREDDIN, E. 2008. Easing Wheelchair Control by Gaze-based Estimation of Intended Motion. *IFAC Proceedings Volumes*, 41, 9162-9167.
- BAZRAFKAN, S., KAR, A. & COSTACHE, C. 2015. Eye Gaze for Consumer Electronics: Controlling and commanding intelligent systems. *IEEE Consumer Electronics Magazine*, 4, 65-71.
- BEN TAHER, F., BEN AMOR, N., JALLOULI, M., BEN HAMMOUDA, A. & DGHIM, O. 2016. A collaborative and voice configured electric powered wheelchair control system based on EEG and head movement. *3rd International Conference on Automation, Control Engineering and Computer Science (ACECS-2016)*.
- BETKE, M., GIPS, J. & FLEMING, P. 2002. The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10, 1-10.
- BRADSKI, G. R. Real time face and object tracking as a component of a perceptual user interface. *Applications of Computer Vision, 1998. WACV '98. Proceedings.*, Fourth IEEE Workshop on, 19-21 Oct 1998 1998 Princeton, New Jersey, USA. 214-219.
- CEREBRAL PALSY ALLIANCE. N/A. *Types of cerebral palsy* [Online]. Cerebral Palsy Alliance Research Foundation Cerebral Palsy Alliance Research Foundation Available: <https://research.cerebralpalsy.org.au/what-is-cerebral-palsy/types-of-cerebral-palsy/> [Accessed 20 March 2018].
- CHANG, K., BOWYER, K. & FLYNN, P. Face recognition using 2D and 3D facial data. *ACM Workshop on Multimodal User Authentication, 2003 Santa Barbara, California*. 25-32.

- CHRYSOS, G. G., ANTONAKOS, E., SNAPE, P., ASTHANA, A. & ZAFEIRIOU, S. 2017. A Comprehensive Performance Evaluation of Deformable Face Tracking "In-the-Wild". *International Journal of Computer Vision*.
- DEGTYAREV, N. & SEREDIN, O. 2010. Comparative Testing of Face Detection Algorithms. In: ELMOATAZ, A., LEZORAY, O., NOUBOUD, F., MAMMASS, D. & MEUNIER, J. (eds.) *Image and Signal Processing: 4th International Conference, ICISP 2010, Trois-Rivières, QC, Canada, June 30-July 2, 2010. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- DESHMUKH, S. P., PATWARDHAN, M. S. & MAHAJAN, A. R. 2018. Feedback based real time facial and head gesture recognition for e-learning system. *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*. Goa, India: ACM.
- DIX, A. 2009. Human-Computer Interaction. In: LIU, L. & ÖZSU, M. T. (eds.) *Encyclopedia of Database Systems*. Boston, MA: Springer US.
- DLIB. 2018. *Dlib* [Online]. Available: <http://dlib.net/> [Accessed 16 January 2018].
- DRAELOS, M., QIU, Q., BRONSTEIN, A. & SAPIRO, G. Intel realsense = Real low cost gaze. 2015 IEEE International Conference on Image Processing (ICIP), 27-30 Sept. 2015. 2520-2524.
- EKMAN, P. 1993. Facial expression and emotion. *American psychologist*, 48, 384.
- EKMAN, P. & FRIESEN, W. V. 1976. Measuring facial movement. *Environmental psychology and nonverbal behavior*, 1, 56-75.
- EKMAN, P., ROSENBERG, E. & EDITORS 1997. *What the Face Reveals: Basic and Applied Studies of Spontaneous Expression Using the Facial Action Coding System (FACS)*.
- FASEL, B. & LUETTIN, J. 2003. Automatic facial expression analysis: a survey. *Pattern Recognition*, 36, 259-275.
- FEHR, L., LANGBEIN, W. E. & SKAAR, S. B. 2000. Adequacy of power wheelchair control interfaces for persons with severe disabilities: a clinical survey. *Journal of rehabilitation research and development*, 37, 353.
- FELZENSZWALB, P. F. & HUTTENLOCHER, D. P. 2004. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59, 167-181.
- FINE, G. & TSOTSOS, J. 2009. Examining the feasibility of face gesture detection using a wheelchair mounted camera.
- HSU, R.-L., ABDEL-MOTALEB, M. & JAIN, A. K. 2002. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 696-706.
- IMOTIONS. N/A. *Facial Action Coding System (FACS) - A Visual Guidebook* [Online]. Available: <https://imotions.com/blog/facial-action-coding-system/> [Accessed 10 October 2018].
- INTEL. 2016. *Intel® RealSense™ Camera SR300* [Online]. Available: <https://software.intel.com/sites/default/files/managed/0c/ec/realsense-sr300-product-datasheet-rev-1-0.pdf> [Accessed 9 October 2018].
- INTEL. 2017a. *Intel® RealSense™ Camera SR300* [Online]. Intel Software Developer Zone: Intel. Available: <https://software.intel.com/en-us/realsense/sr300> [Accessed 27 December 2017].
- INTEL. 2017b. *Intel® RealSense™ Depth Camera D400-Series* [Online]. Intel Software Developer Zone. Available: <https://software.intel.com/en-us/realsense/d400> [Accessed 27 December 2017].
- INTEL. 2018a. *Intel® RealSense™ Depth Camera D435* [Online]. Intel. Available: <https://click.intel.com/intelr-realsensetm-depth-camera-d435.html> [Accessed 5 October 2018].
- INTEL. 2018b. *Product Change Notification* [Online]. Mouser. Available: https://www.mouser.com/PCN/Intel_Corporation_PCN115963_00.pdf [Accessed].
- INTEL. N/A. *Intel RealSense SDK for windows (discontinued)* [Online]. Available: <https://software.intel.com/en-us/realsense-sdk-windows-eol> [Accessed 20 January 2018].
- INTEL REALSENSE. 2018. *Intel® RealSense™ SDK 2.0* [Online]. Available: <https://github.com/IntelRealSense/librealsense/releases> [Accessed 18 January 2018].

- INTEL REALSENSE SUPPORT. 2018. *When will Realsense D435 be available?* [Online]. Intel RealSense communities. Available: <https://communities.intel.com/thread/123518> [Accessed 5 October 2018].
- INTELREALSENSE. N/A. *Intel RealSense Cross Platform API* [Online]. Github. Available: <https://github.com/IntelRealSense/librealsense/tree/v1.12.1> [Accessed 6 January 2018].
- JANTUNEN, T., MESCH, J., PUUPPONEN, A. & LAAKSONEN, J. T. 2016. On the rhythm of head movements in Finnish and Swedish Sign Language sentences. *Speech Prosody 2016*. Boston, USA: ResearchGate.
- JIA, P., GRAY, J. O., HU, H. H., LU, T. & YUAN, K. 2007. Head gesture recognition for hands-free control of an intelligent wheelchair. *Industrial Robot: An International Journal*, 34, 60-68.
- JU, J. S., SHIN, Y. & KIM, E. Y. 2009. Vision based interface system for hands free control of an Intelligent Wheelchair. *J Neuroeng Rehabil*, 6, 33.
- KAZEMI, V. & SULLIVAN, J. One millisecond face alignment with an ensemble of regression trees. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 23-28 June 2014 2014 Columbus, Ohio, USA. 1867-1874.
- KHAZAB, F. 2016. *Analysis of Potential Types of Brain-Computer Interface Technology for a Severely Locked-in Patient*. Masters by Research, Thesis (Masters).
- KHOSHELHAM, K. & OUDE ELBERINK, S. 2012. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. 12, 1437-54.
- KIPMAN, A. & LAPSEN, M. 2017. Exclusive: Microsoft Has Stopped Manufacturing the Kinect. In: WILSON, M. (ed.). *Fast Co Design: Fast Co Design*.
- KUKREJA, R. S. 2018. *AudioVisual (Brain) Controlled Computer Wheelchair* Adelaide, Australia: Flinders University.
- KUNO, Y., SHIMADA, N. & SHIRAI, Y. 2003. Look where you're going [robotic wheelchair]. *Robotics & Automation Magazine, IEEE*, 10, 26-34.
- LEAMAN, J. & LA, H. M. 2017. A Comprehensive Review of Smart Wheelchairs: Past, Present, and Future. *Human-Machine Systems, IEEE Transactions on*, 47, 486-499.
- LI, H., LIN, Z., SHEN, X., BRANDT, J. & HUA, G. A convolutional neural network cascade for face detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015 Boston. 5325-5334.
- MANDAL, M. K. 2016. *Serial Communication with an Arduino using C++ on Windows* [Online]. Manash's Blog. Available: <https://blog.manash.me/serial-communication-with-an-arduino-using-c-on-windows-d08710186498> [Accessed 19 September 2018].
- MATTHIES, D., A STRECKER, B. & URBAN, B. 2017. EarFieldSensing: A Novel In-Ear Electric Field Sensing to Enrich Wearable Gesture Input through Facial Expressions. *Conference on Human Factors in Computing Systems* Denver, Colorado.
- MICROSOFT. 2017. Human Interface Guidelines v2.0. 2017. Available: <https://developer.microsoft.com/en-us/windows/kinect/hardware>.
- MICROSOFT. N/A-a. *Face Tracking* [Online]. Microsoft. Available: <https://msdn.microsoft.com/en-us/library/jj130970.aspx> [Accessed 21 January 2018].
- MICROSOFT. N/A-b. *Kinect for Windows SDK 2.0* [Online]. Microsoft. Available: <https://www.microsoft.com/en-au/download/details.aspx?id=44561> [Accessed 18 January 2018].
- MITRA, S. & ACHARYA, T. 2007. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37, 311-324.
- MORIKAWA, C. & LYONS, M. J. 2017. Design and Evaluation of Vision-based Head and Face Tracking Interfaces for Assistive Input.
- NASIF, S. & KHAN, M. A. G. Wireless head gesture controlled wheel chair for disable persons. 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 21-23 Dec. 2017 2017. 156-161.

- NOLDUS. N/A. *FaceReader: Action Unit Module* [Online]. Available: <https://www.noldus.com/facereader/action-unit-module> [Accessed 10 October 2018].
- OPENCV. 2018a. *Face Detection using Haar Cascades* [Online]. OpenCV: OpenCV. Available: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html [Accessed 28 December 2017].
- OPENCV. 2018b. *OpenCV 3.4.0* [Online]. OpenCV. Available: <https://github.com/opencv/opencv/releases/tag/3.4.0> [Accessed 14 January 2018].
- PARMAR, K., MEHTA, B. & SAWANT, R. 2012. Facial-feature based Human-Computer Interface for disabled people.
- PATIL, J. V. & BAILKE, P. Real time facial expression recognition using RealSense camera and ANN. 2016 International Conference on Inventive Computation Technologies (ICICT), 26-27 Aug. 2016 2016. 1-6.
- POINTCLOUDLIBRARY. 2018. *Point Cloud Library* [Online]. Available: <https://github.com/PointCloudLibrary/pcl> [Accessed 21 January 2018].
- POWERS, D. M. W. 2011. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, 2, 37-63.
- RICH-PERRETT, L. 2018. Development of a Face Tracking Switch Access Solution for Clients with Severe and Multiple Disabilities. Flinders University, College of Science and Engineering.
- ROBINSON, M. n.d. *Autonomous Brain Controlled Computer Interface (ABC) Wheelchair*. Flinders University.
- ROZADO, D., NIU, J. & LOCHNER, M. 2017. Fast Human-Computer Interaction by Combining Gaze Pointing and Face Gestures. *ACM Trans. Access. Comput.*, 10, 1-18.
- SANTOS, R., SANTOS, N., JORGE, P. M. & ABRANTES, A. 2014. Eye Gaze as a Human-computer Interface. *Procedia Technology*, 17, 376-383.
- SARAGIH, J. & MCDONALD, K. 2017. *FaceTracker* [Online]. Available: <https://github.com/kylemcdonald/FaceTracker> [Accessed 22 January 2018].
- SARAGIH, J. M., LUCEY, S. & COHN, J. F. Face alignment through subspace constrained mean-shifts. 2009 IEEE 12th International Conference on Computer Vision, Sept. 29 2009-Oct. 2 2009 2009 Kyoto, Japan. 1034-1041.
- SBS. N/A. *Filipino Culture* [Online]. SBS. Available: <https://culturalatlas.sbs.com.au/filipino-culture/filipino-culture-communication> [Accessed 9 October 2018].
- SILVA, V., SOARES, F., ESTEVES, J. S., FIGUEIREDO, J., SANTOS, C. & PEREIRA, A. P. 2017. Happiness and Sadness Recognition System—Preliminary Results with an Intel RealSense 3D Sensor. In: GARRIDO, P., SOARES, F. & MOREIRA, A. P. (eds.) *CONTROLO 2016: Proceedings of the 12th Portuguese Conference on Automatic Control*. Cham: Springer International Publishing.
- SIMPSON, R. C. 2005. Smart wheelchairs: A literature review. *The Journal of Rehabilitation Research and Development*, 42.
- SPINALCORD.COM. N/A. *Quadriplegia / Tetraplegia* [Online]. SpinalCord.com: SpinalCord.com. Available: <https://www.spinalcord.com/quadriplegia-tetraplegia> [Accessed 10 October 2018].
- TASTENKUNST. 2018. *Beyond Reality Face SDK - v4.0.0 (BRFv4)* [Online]. Github. Available: https://github.com/Tastenkunst/brfv4_win_examples [Accessed 20 January 2018].
- TIAN, Y.-L., KANADE, T. & COHN, J. F. 2005. Facial expression analysis. *Handbook of face recognition*, 247-275.
- TOBII. N/A. *Tobii Pro X2-30 eye tracker* [Online]. Tobii. Available: <https://www.tobiipro.com/product-listing/tobii-pro-x2-30/> [Accessed 20 March 2018].
- TSUI, C. S. L., PEI JIA, J. Q., GAN, J. Q., HUOSHENG HU, J. Q. & KUI YUAN, J. Q. 2007. EMG-based hands-free wheelchair control with EOG attention shift detection.
- VARONA, J., MANRESA-YEE, C. & PERALES, F. J. 2008. Hands-free vision-based interface for computer accessibility. *Journal of Network and Computer Applications*, 31, 357-374.

- VAZQUEZ-VALENCIA, J. E., MARTIN-ORT, M., OLMOS-PINEDA, I., OLVERA-LOPEZ, J. A. & PINTO-AVENDANO, D. E. 2017. Automatic gesture recognition for wheelchair control. *Proceedings of the XVIII International Conference on Human Computer Interaction*. Cancun, Mexico: ACM.
- VILLAROMAN, N. 2013. *Face Tracking User Interfaces Using Vision-Based Consumer Devices*. Master of Science, Brigham Young University.
- VIOLA, P. & JONES, M. J. 2004. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57, 137-154.
- WEI, L., HU, H. & YUAN, K. 2009. Use of forehead bio-signals for controlling an Intelligent Wheelchair.
- WU, C. W., YANG, H. Z., CHEN, Y. A., ENSA, B., REN, Y. & TSENG, Y. C. Applying machine learning to head gesture recognition using wearables. 2017 IEEE 8th International Conference on Awareness Science and Technology (iCAST), 8-10 Nov. 2017 2017 Taichung, Taiwan. 436-440.
- YANCO, H. A. 1998. Wheellesley: A robotic wheelchair system: Indoor navigation and user interface. *In: MITTAL, V. O., YANCO, H. A., ARONIS, J. & SIMPSON, R. (eds.) Assistive Technology and Artificial Intelligence: Applications in Robotics, User Interfaces and Natural Language Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- YANG, M.-H., KRIEGMAN, D. J. & AHUJA, N. 2002. Detecting faces in images: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 34-58.
- ZHANG, Z. 2012. Microsoft Kinect Sensor and Its Effect. *IEEE MultiMedia*, 19, 4-10.
- ZOGG, C. 2017. Magic off the cuff. *EmpaQuarterly*. Switzerland: EMPA.

APPENDIX

APPENDIX A – NEUTRAL TESTS AND EXECUTION OF GESTURES

A.1 PRELIMINARY TESTS DONE ON THE AUTHOR

Neutral

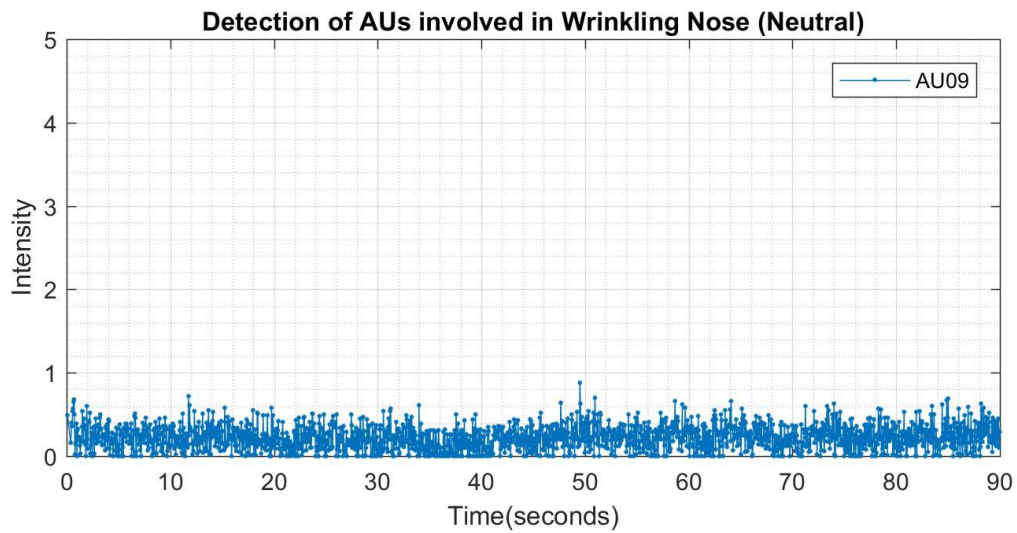


FIGURE 33: DETECTION OF AU09 WHILE HOLDING A NEUTRAL POSITION

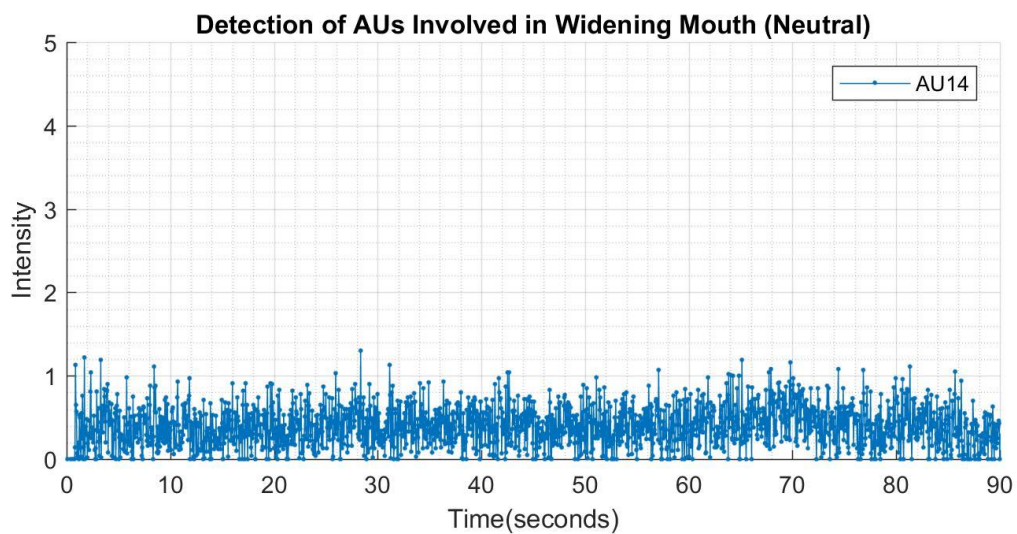


FIGURE 34: DETECTION OF AU14 WHILE HOLDING A NEUTRAL POSITION

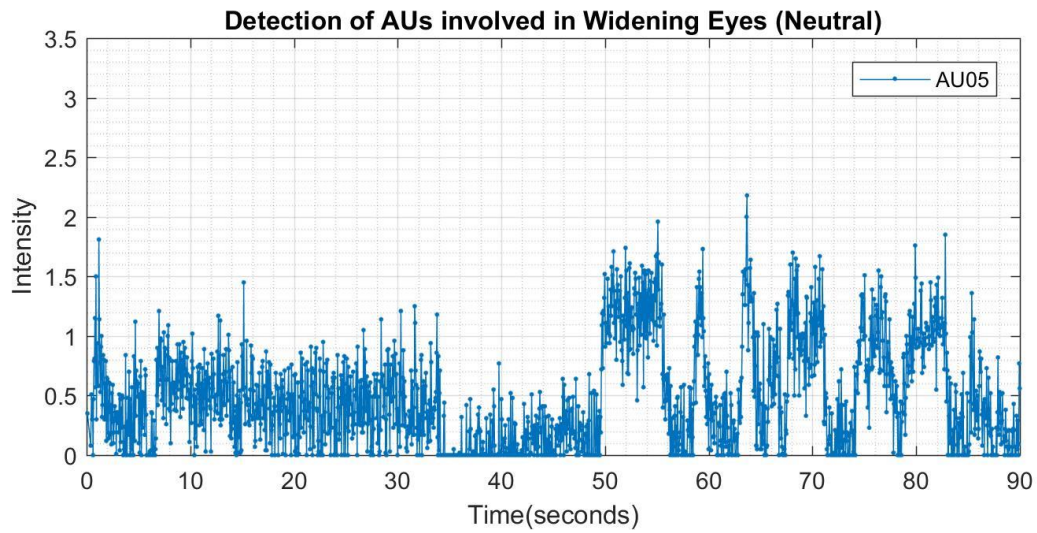


FIGURE 35: DETECTION OF AU05 WHILE HOLDING A NEUTRAL POSITION

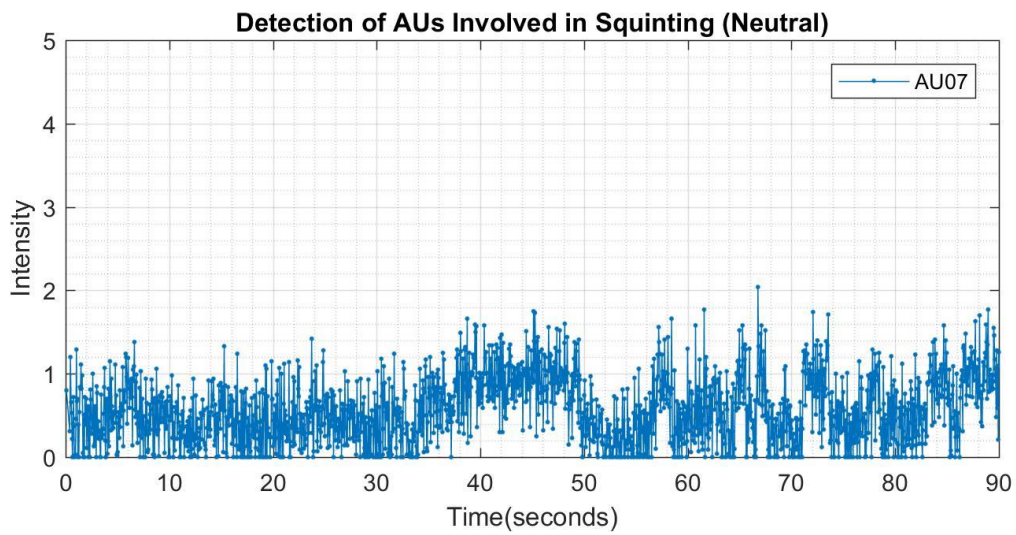


FIGURE 36: DETECTION OF AU07 WHILE HOLDING A NEUTRAL POSITION

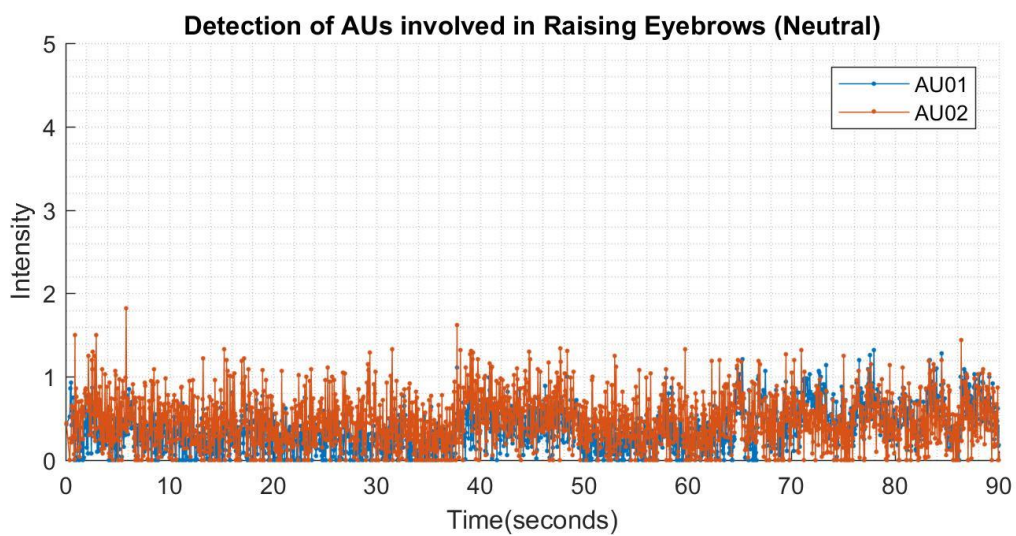


FIGURE 37: DETECTION OF AU01 AND AU02 WHILE HOLDING A NEUTRAL POSITION

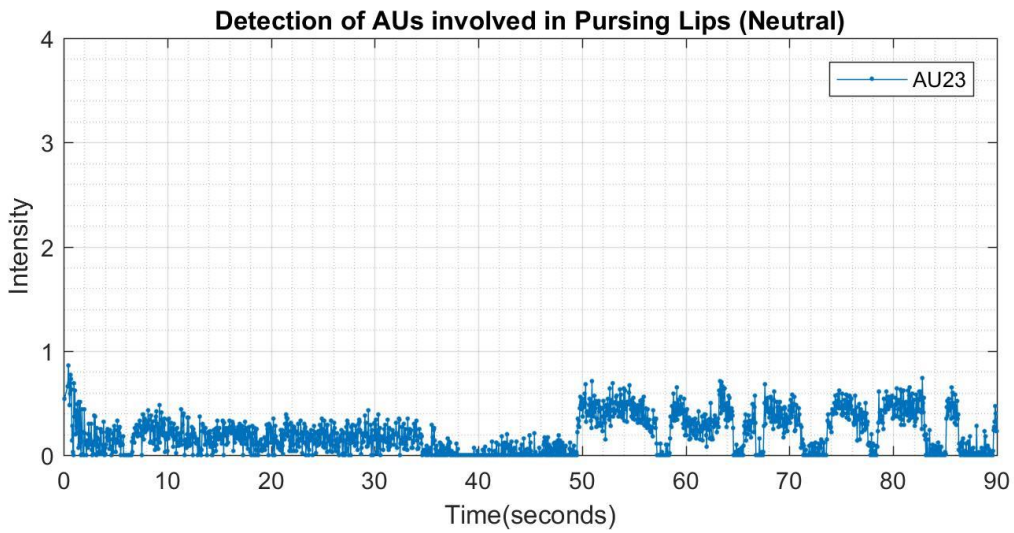


FIGURE 38: DETECTION OF AU23 WHILE HOLDING A NEUTRAL POSITION

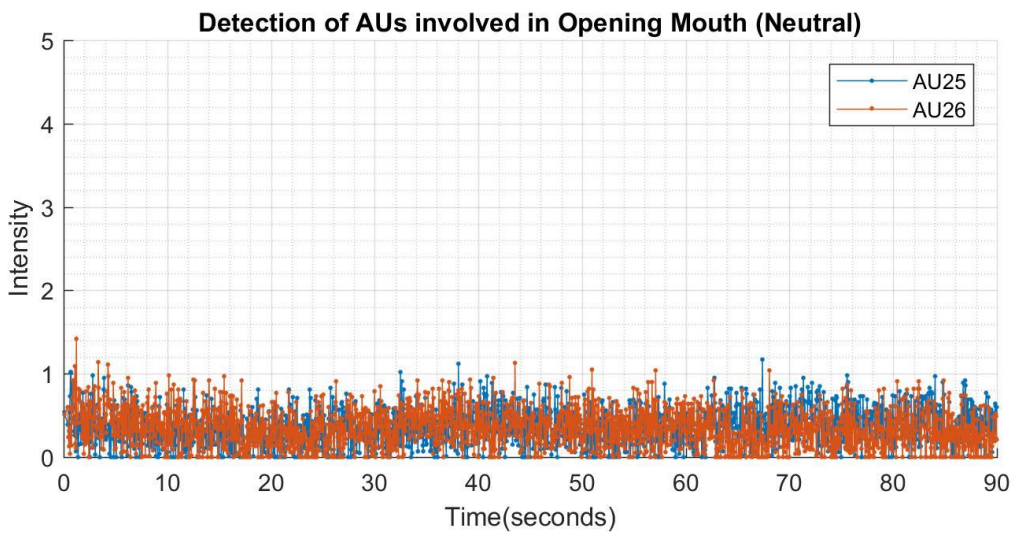


FIGURE 39: DETECTION OF AU25 AND AU26 WHILE HOLDING A NEUTRAL POSITION

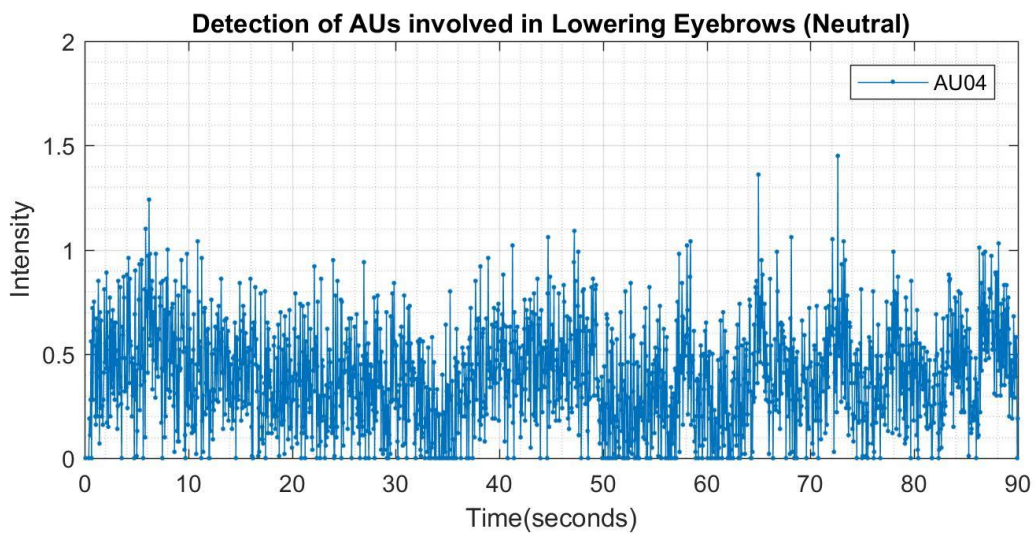


FIGURE 40: DETECTION OF AU04 WHILE HOLDING A NEUTRAL POSITION

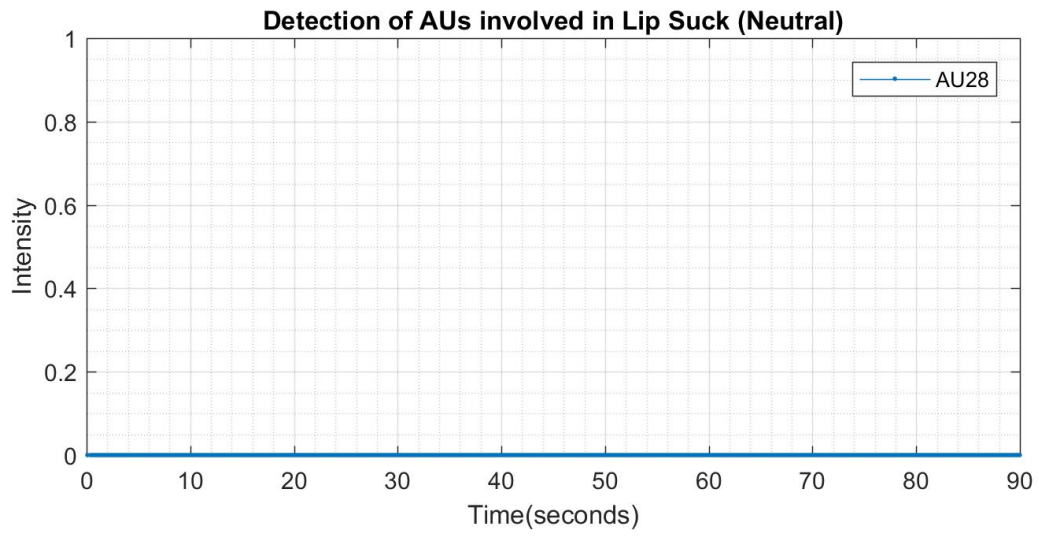


FIGURE 41: DETECTION OF AU28 WHILE HOLDING A NEUTRAL POSITION

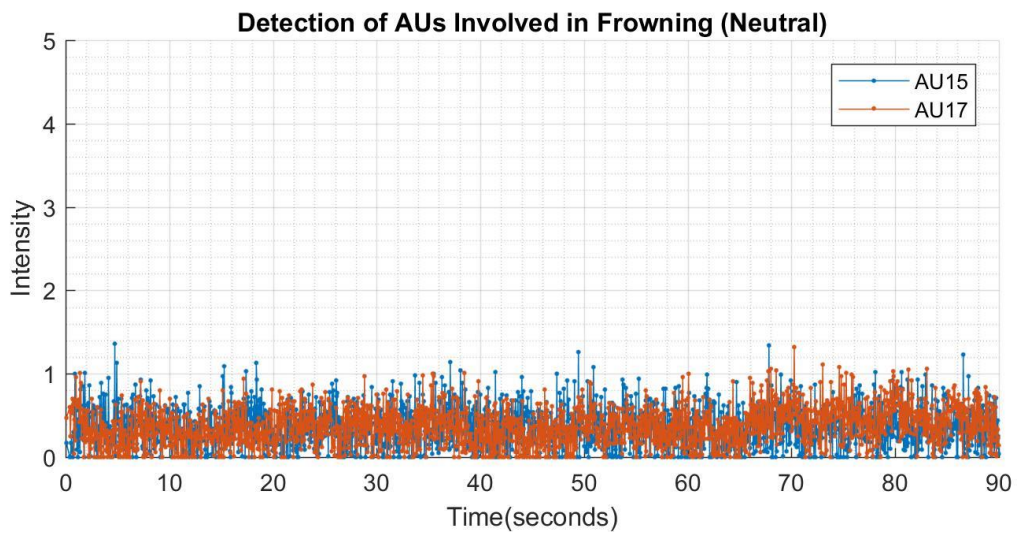


FIGURE 42: DETECTION OF AU15 AND AU17 WHILE HOLDING A NEUTRAL POSITION

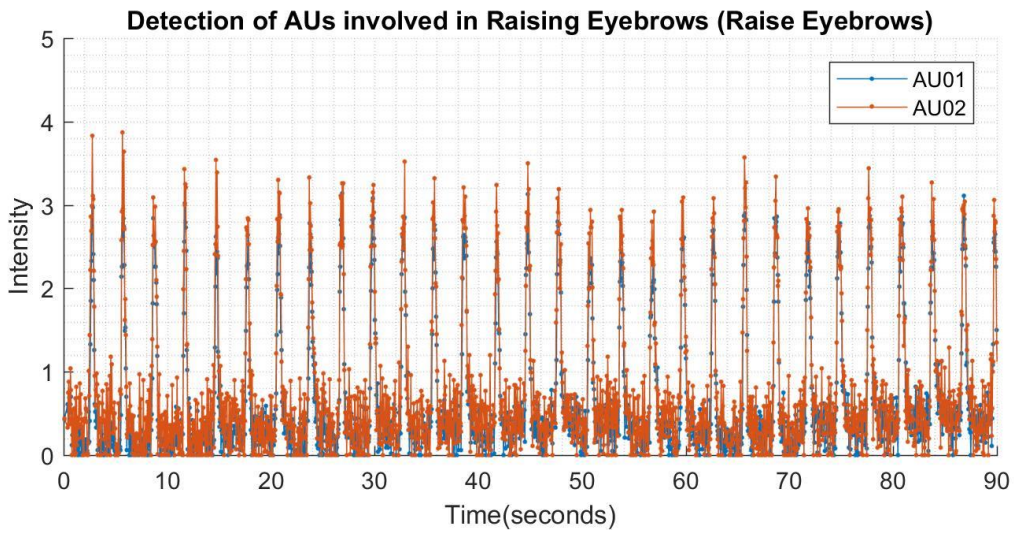


FIGURE 43: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

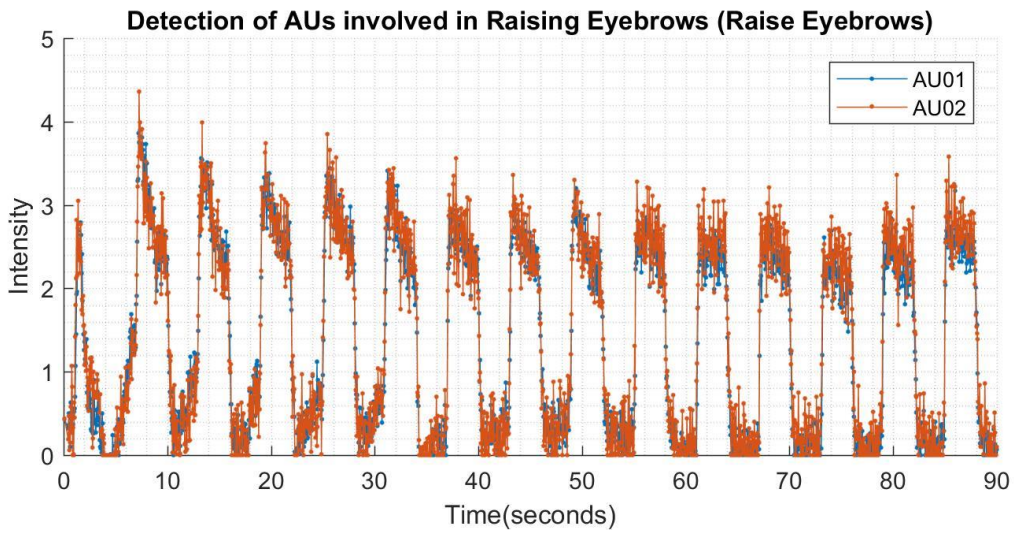


FIGURE 44: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

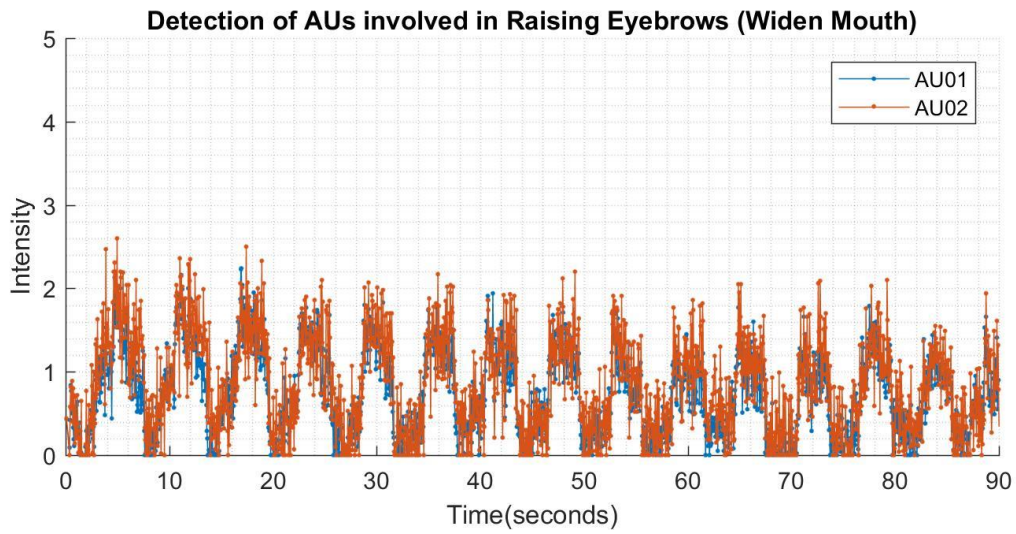


FIGURE 45: DETECTION OF AU01 AND AU02 DURING THE WIDEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

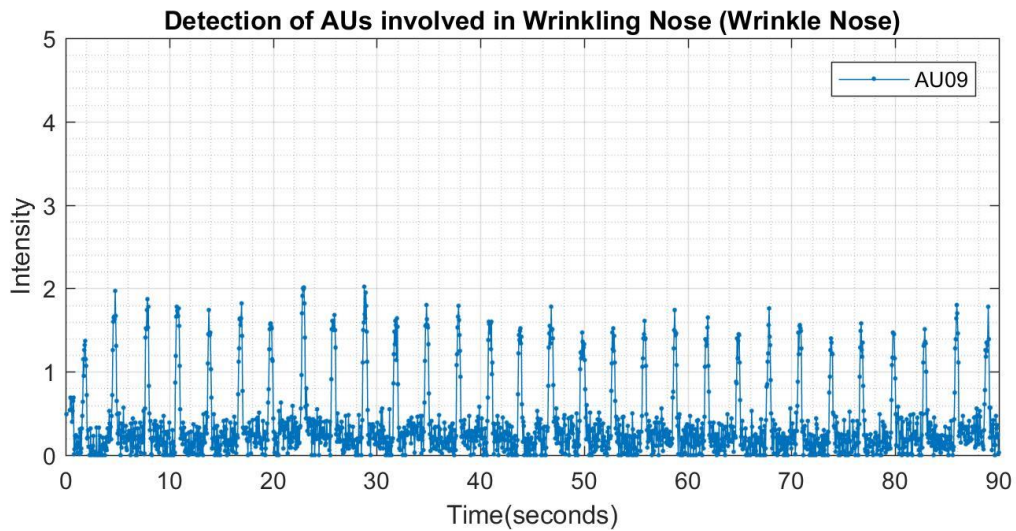


FIGURE 46: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

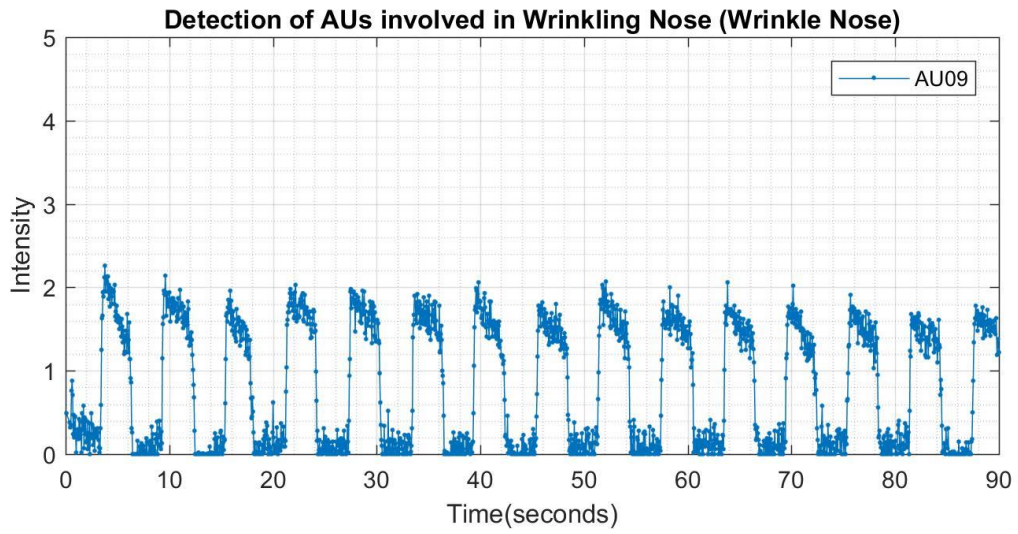


FIGURE 47: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

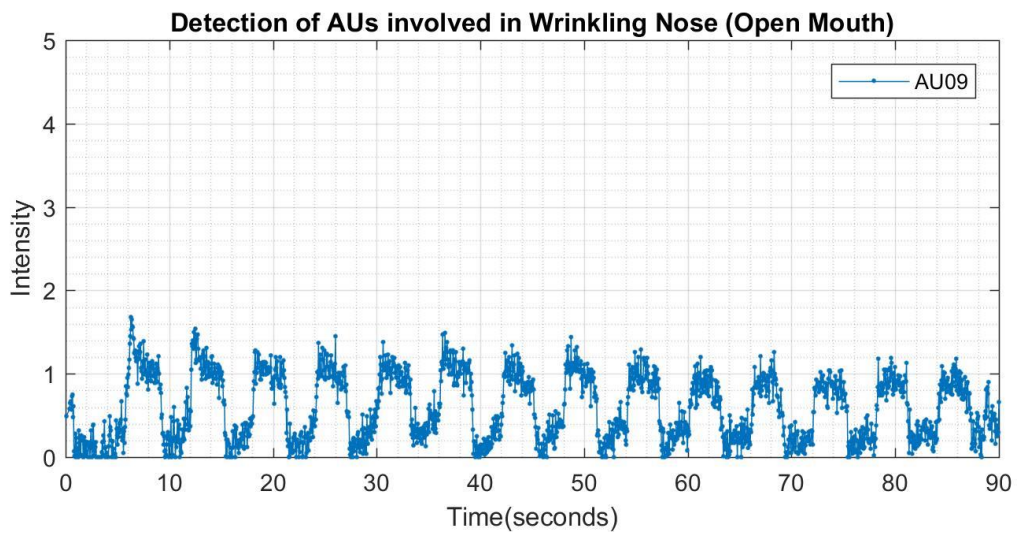


FIGURE 48: DETECTION OF AU09 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

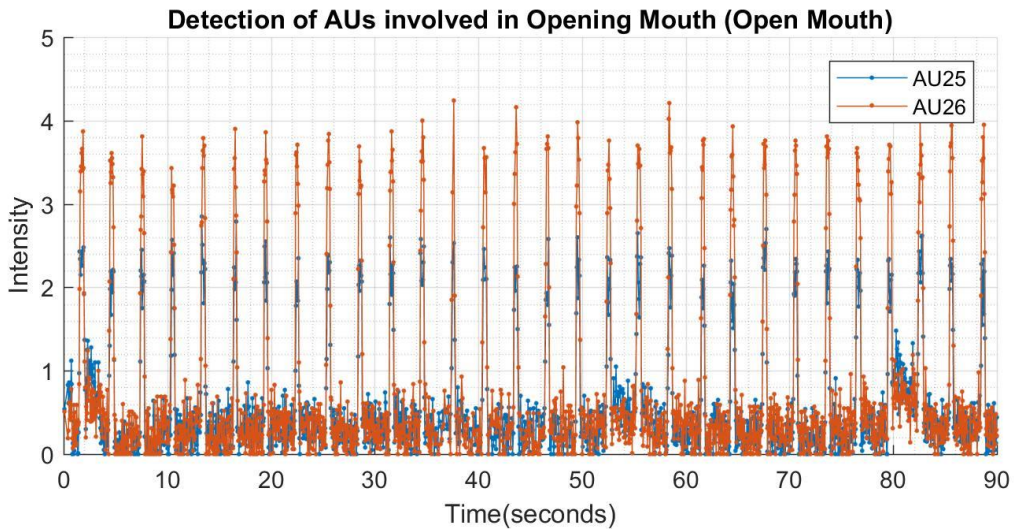


FIGURE 49: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

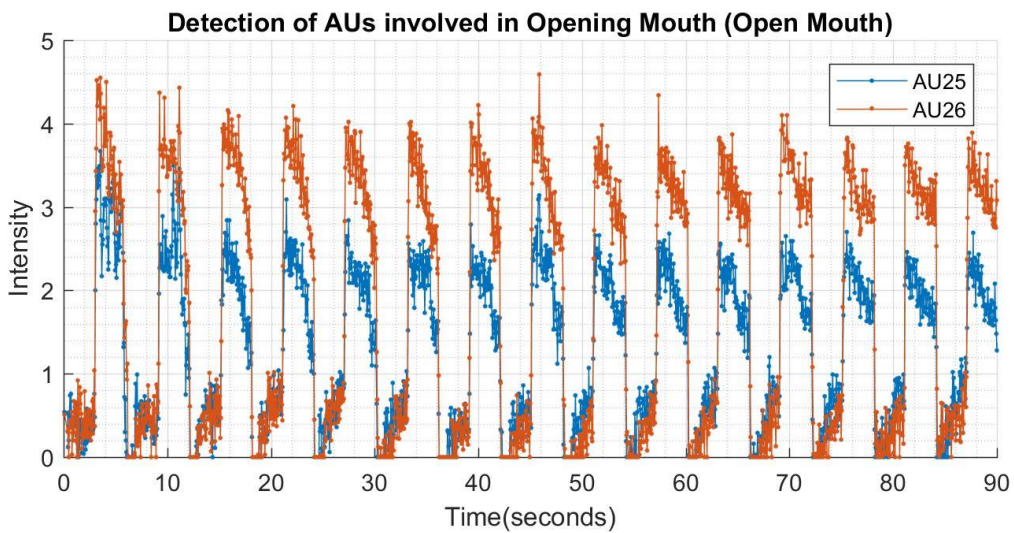


FIGURE 50: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

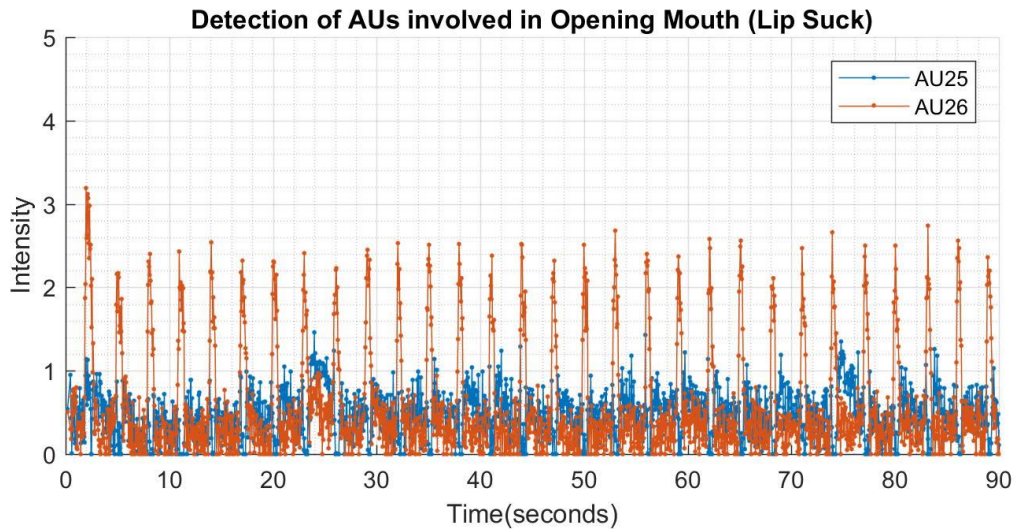


FIGURE 51: DETECTION OF AU25 AND AU26 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

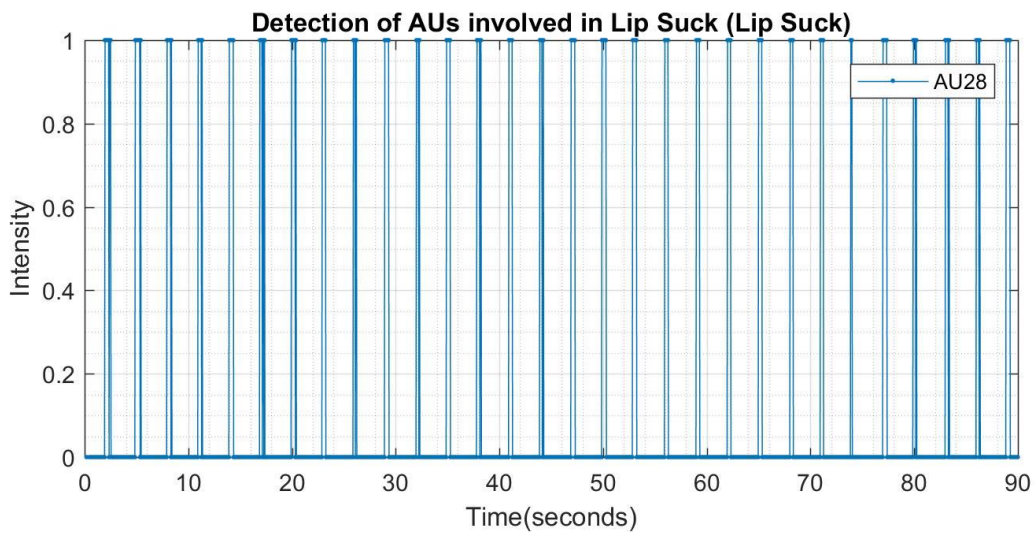


FIGURE 52: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

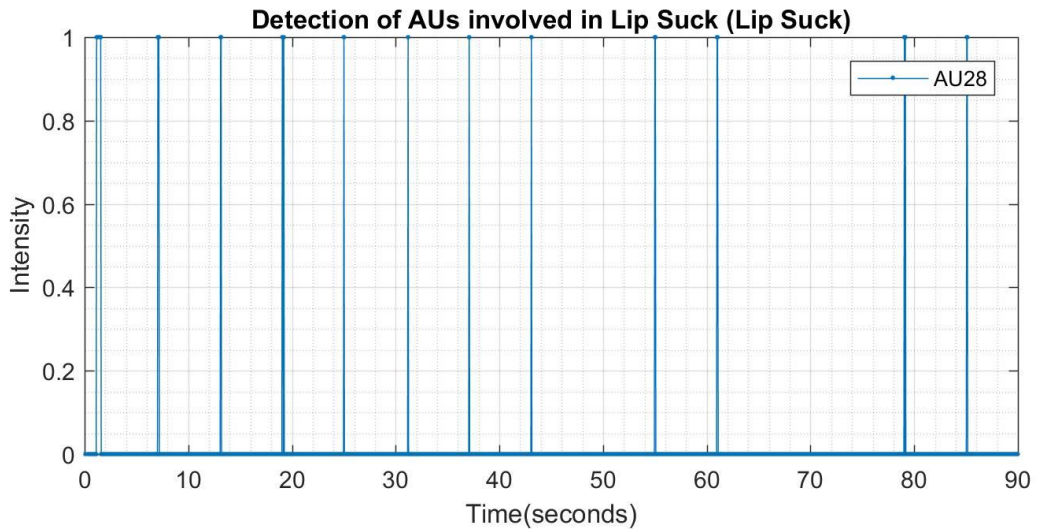


FIGURE 53: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

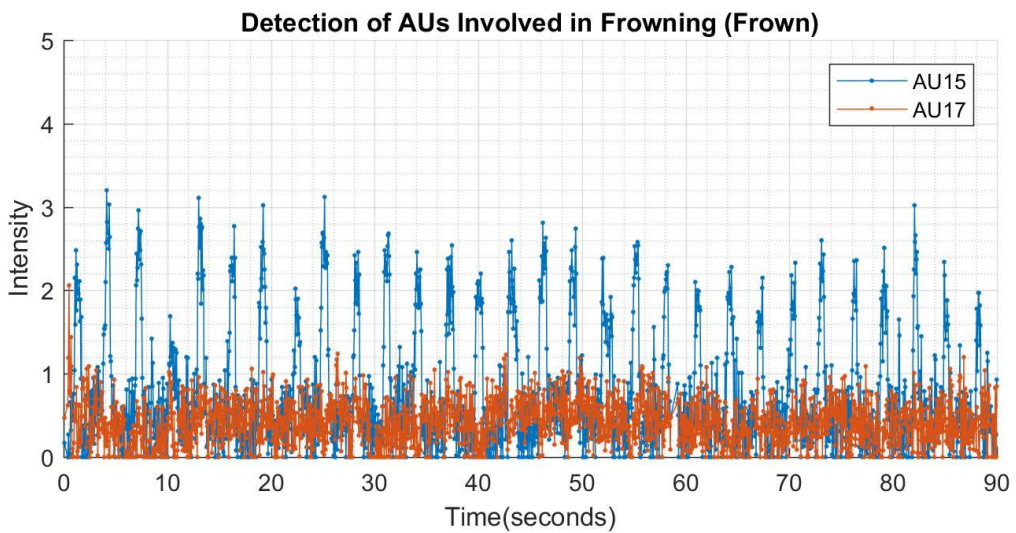


FIGURE 54: DETECTION OF AU15 AND AU17 DURING THE FROWN GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

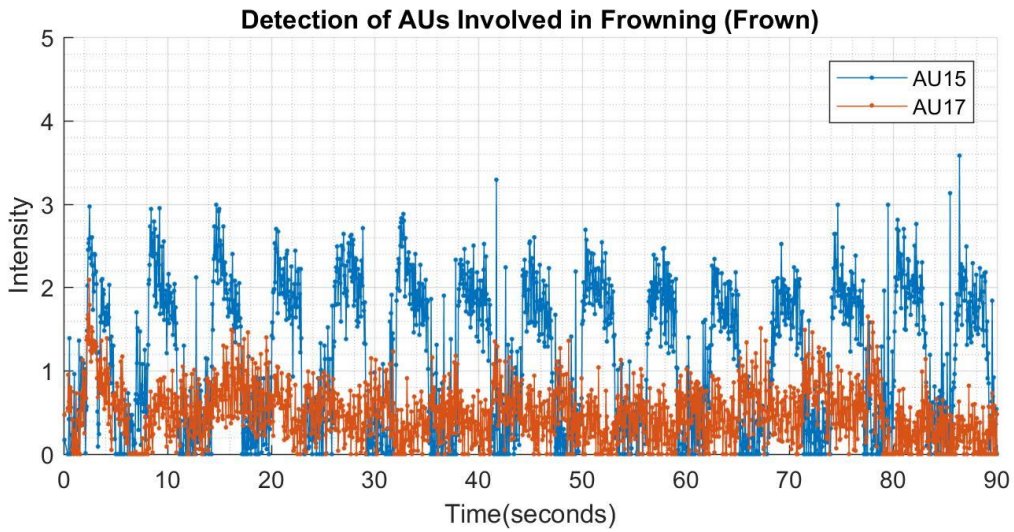


FIGURE 55: DETECTION OF AU15 AND AU17 DURING THE FROWN GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

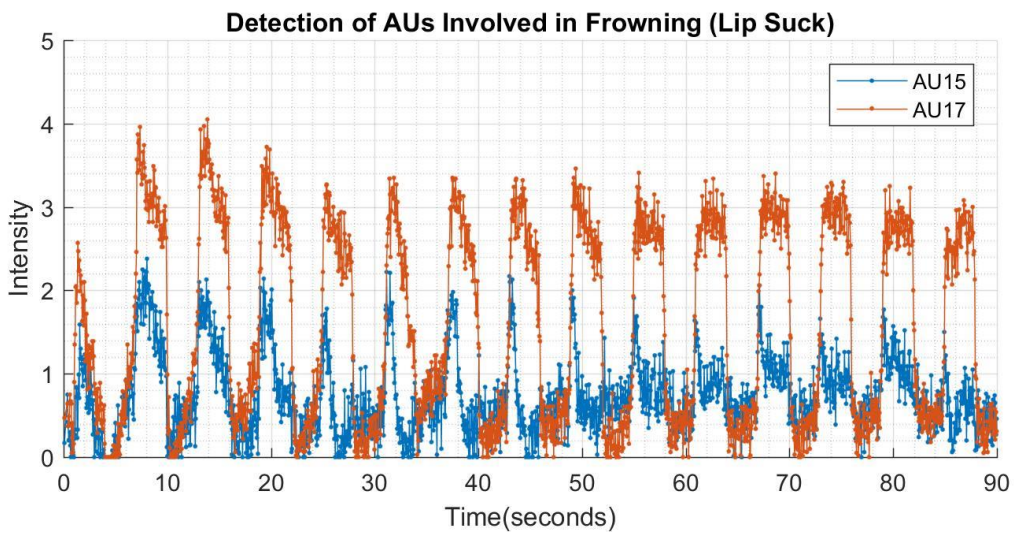


FIGURE 56: DETECTION OF AU15 AND AU17 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

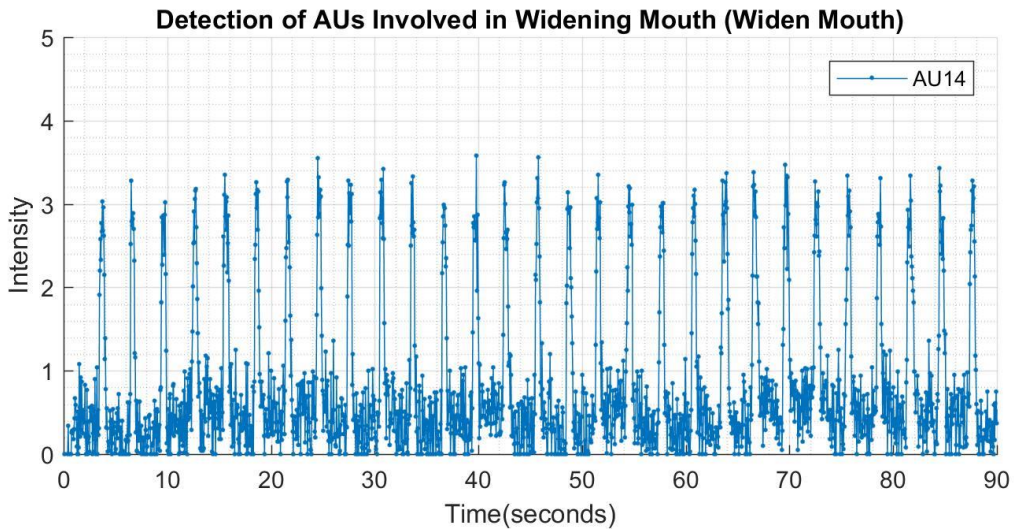


FIGURE 57: DETECTION OF AU14 DURING THE WIDEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

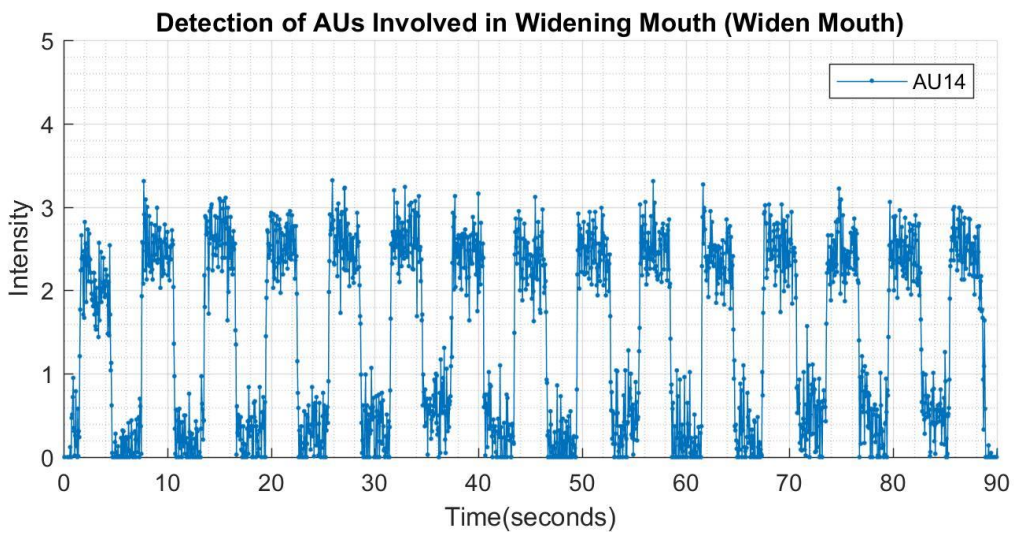


FIGURE 58: DETECTION OF AU14 DURING THE WIDEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

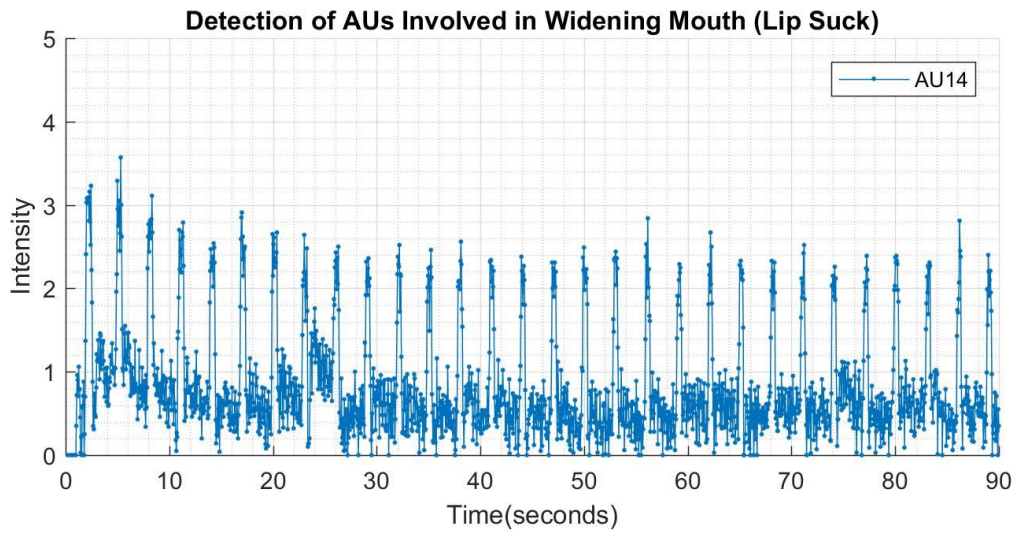


FIGURE 59: DETECTION OF AU14 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

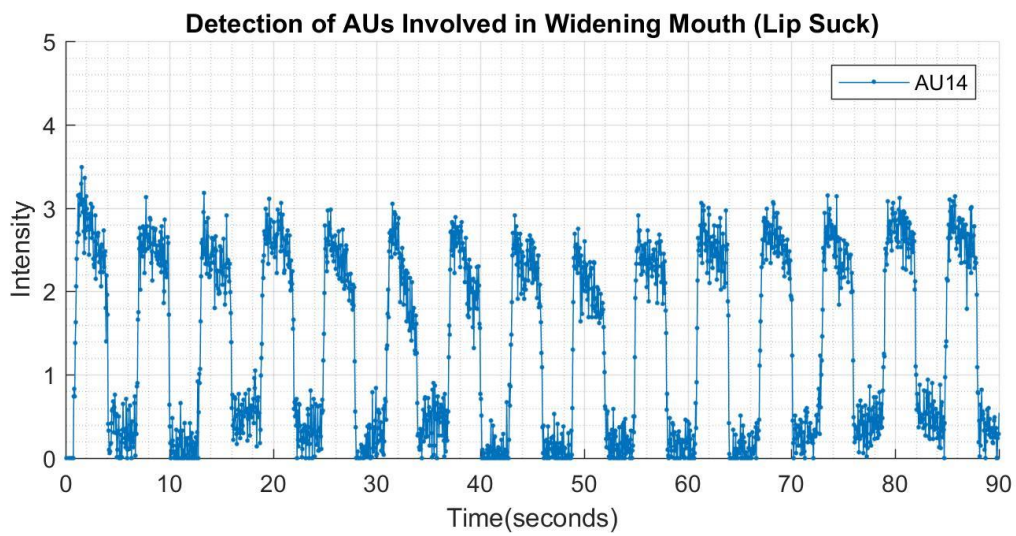


FIGURE 60: DETECTION OF AU14 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

A.2 SUBJECT 1

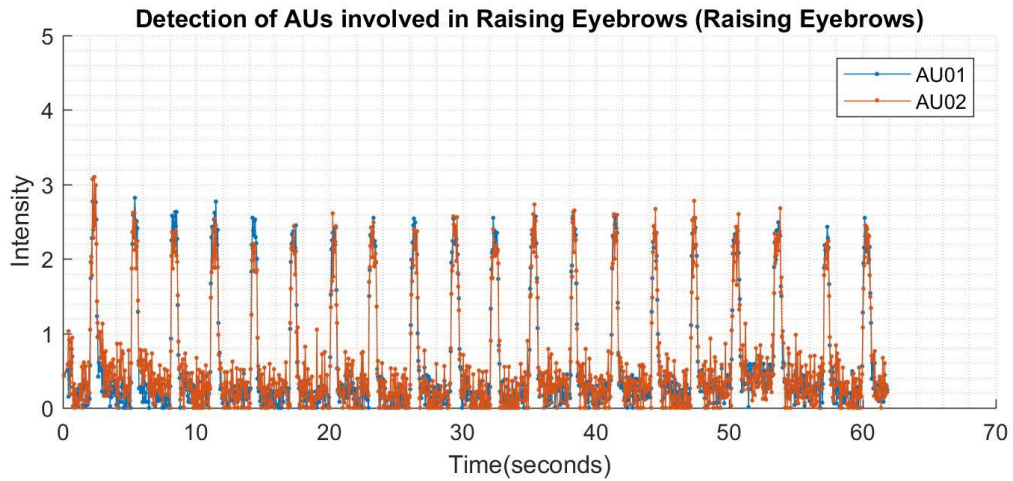


FIGURE 61: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

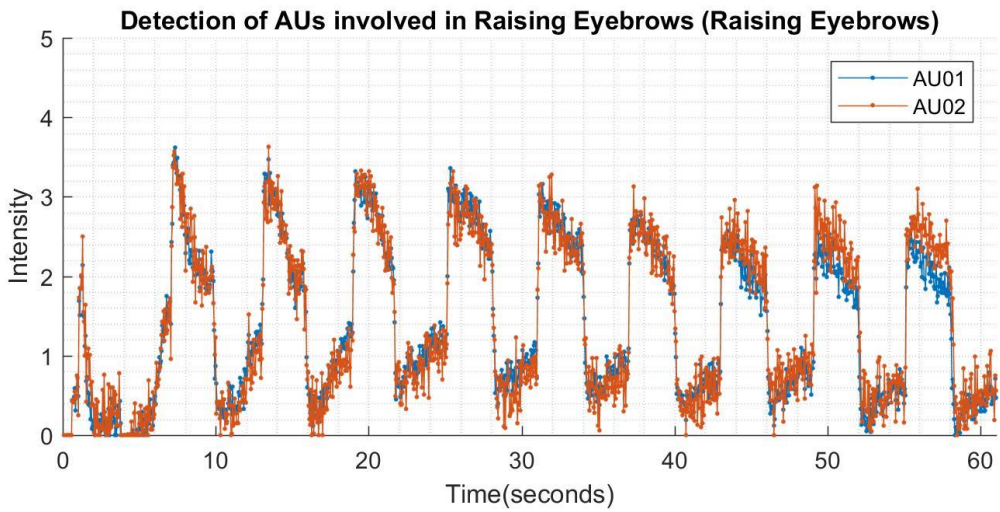


FIGURE 62: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

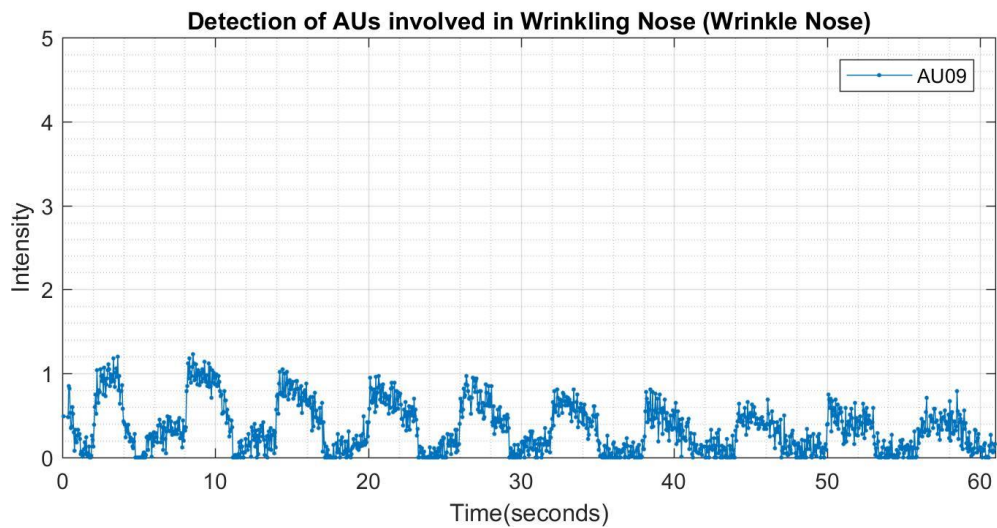


FIGURE 63: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

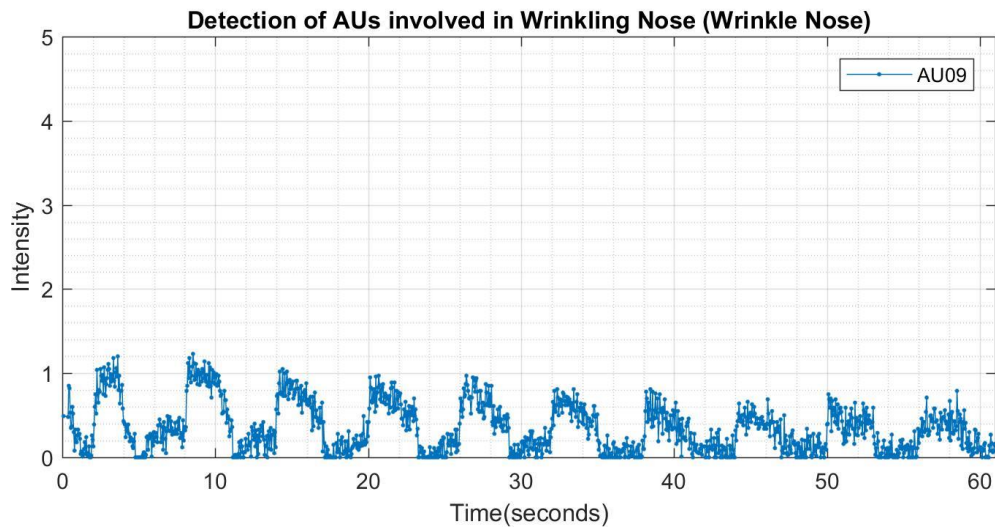


FIGURE 64: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

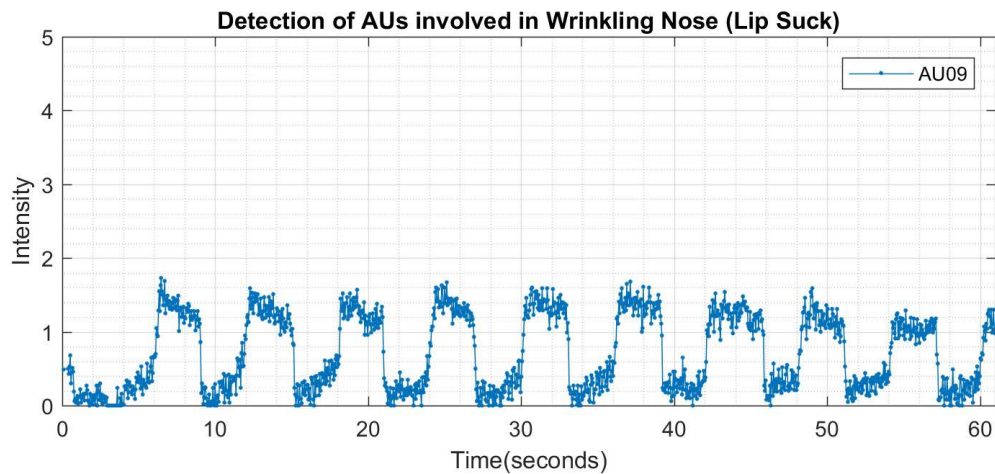


FIGURE 65: DETECTION OF AU09 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

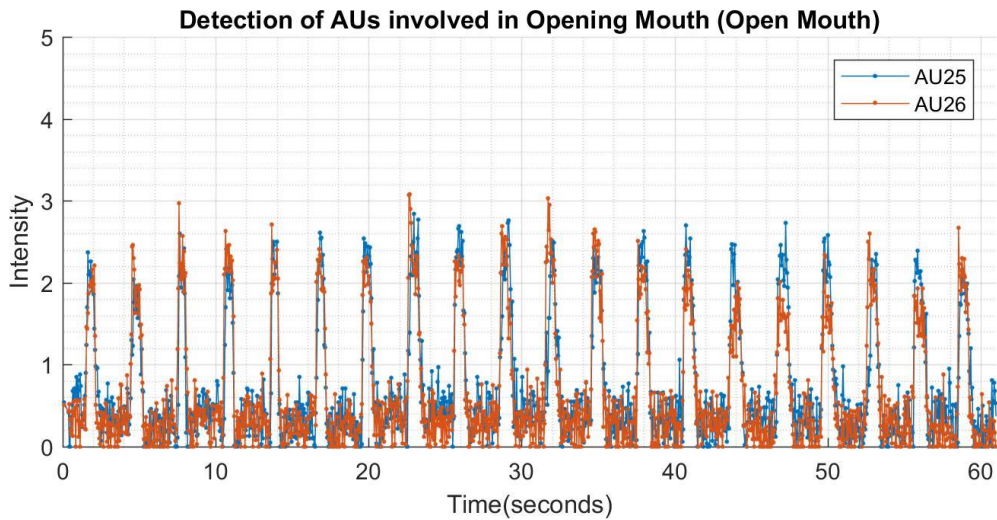


FIGURE 66: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

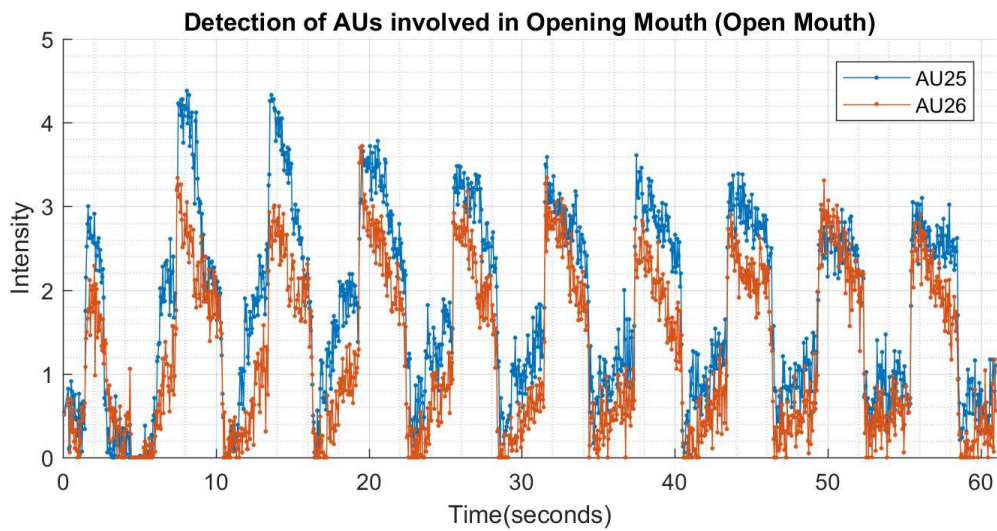


FIGURE 67: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

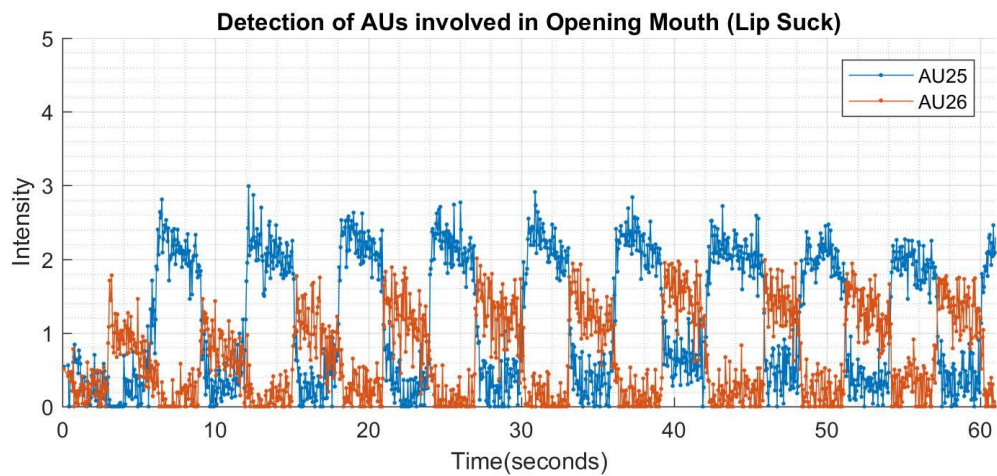


FIGURE 68: DETECTION OF AU25 AND AU26 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

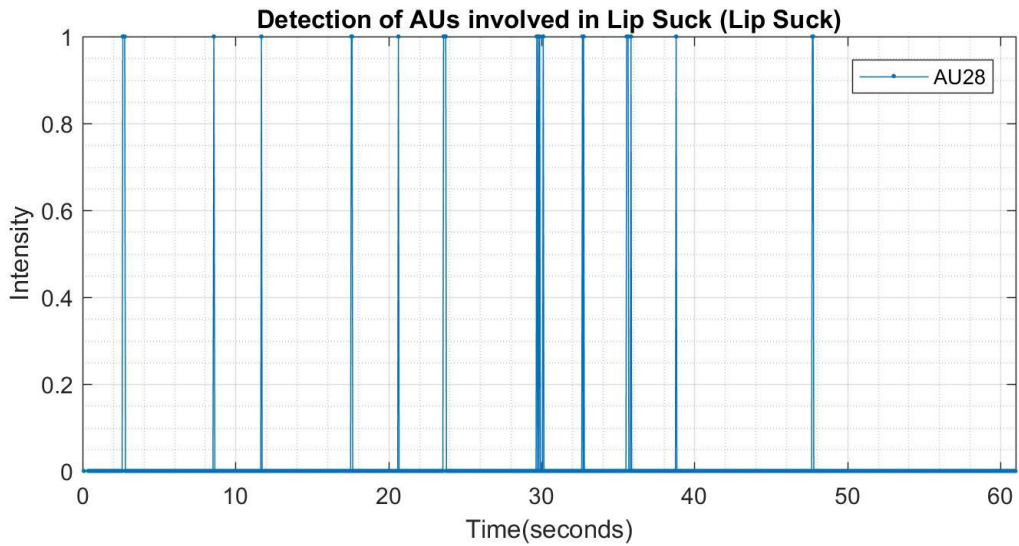


FIGURE 69: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

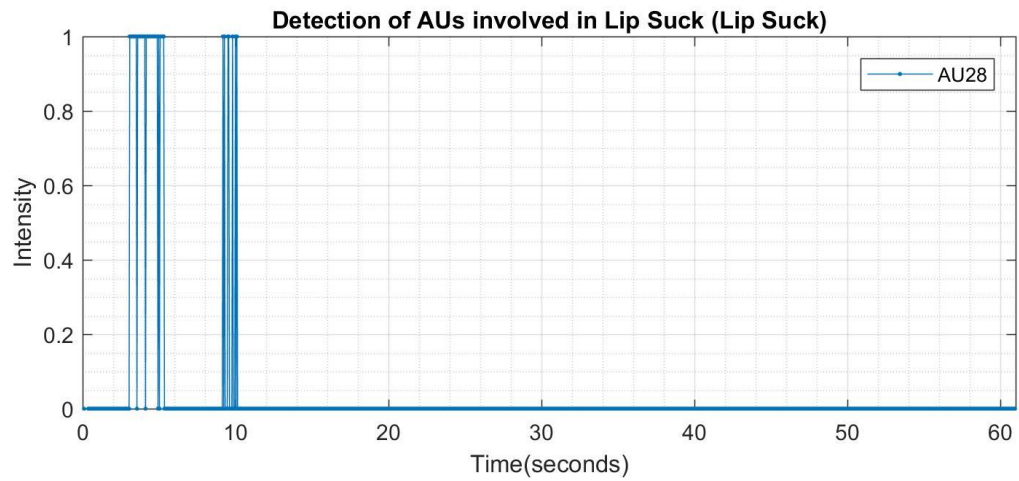


FIGURE 70: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

A.3 SUBJECT 2

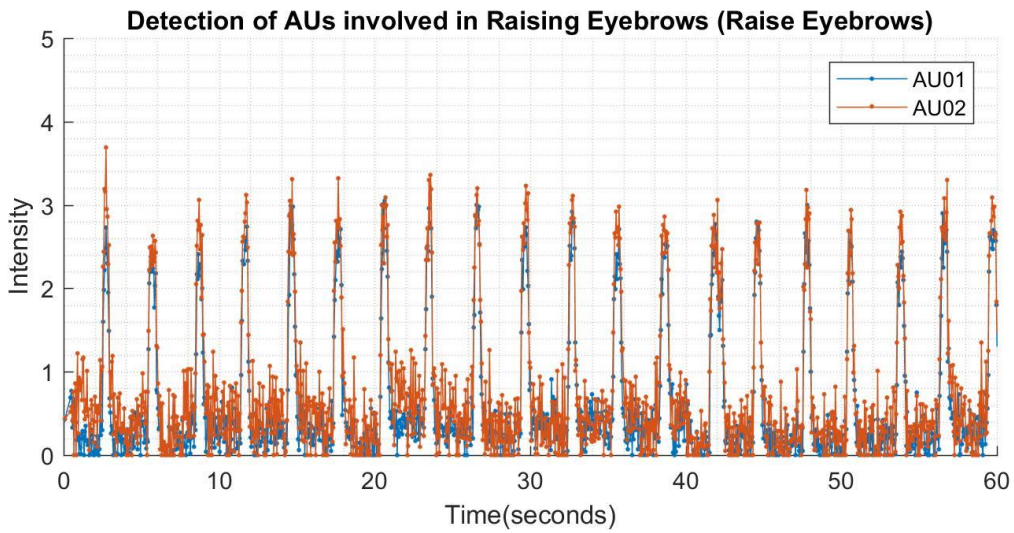


FIGURE 71: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

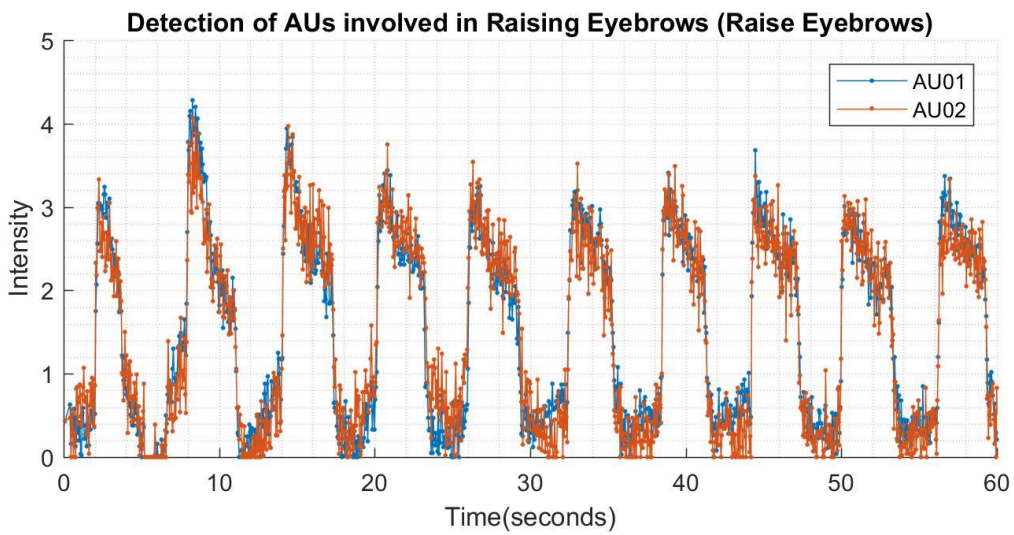


FIGURE 72: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

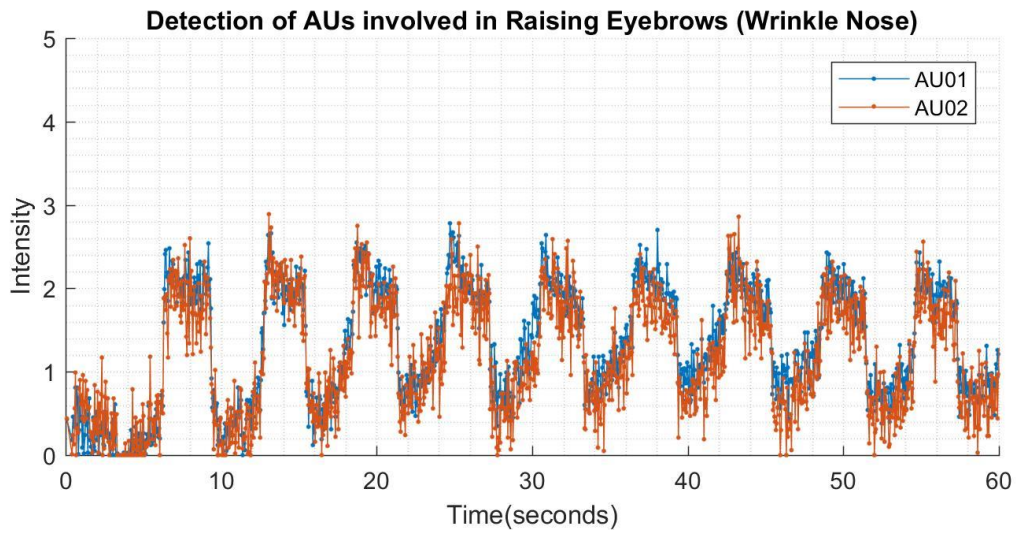


FIGURE 73: DETECTION OF AU01 AND AU02 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

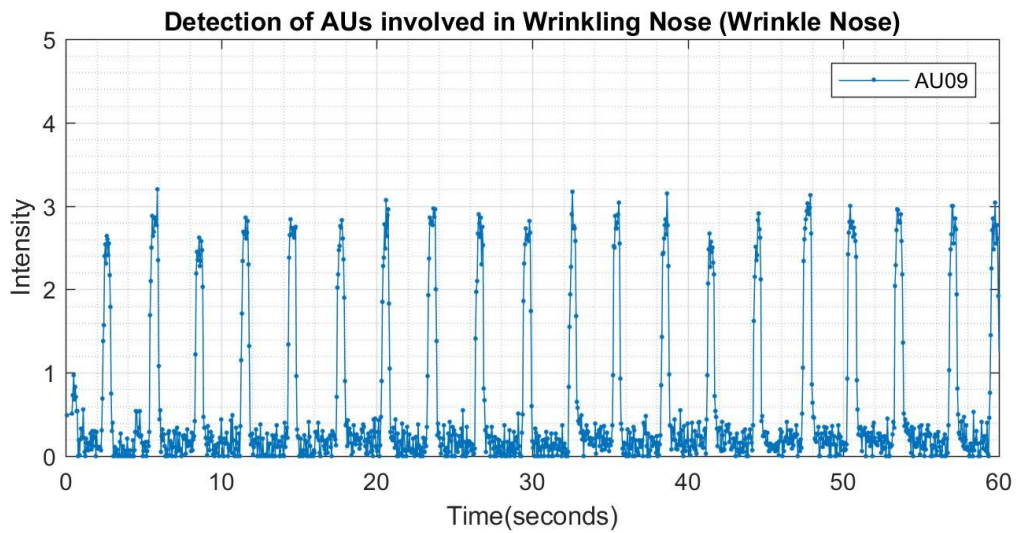


FIGURE 74: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

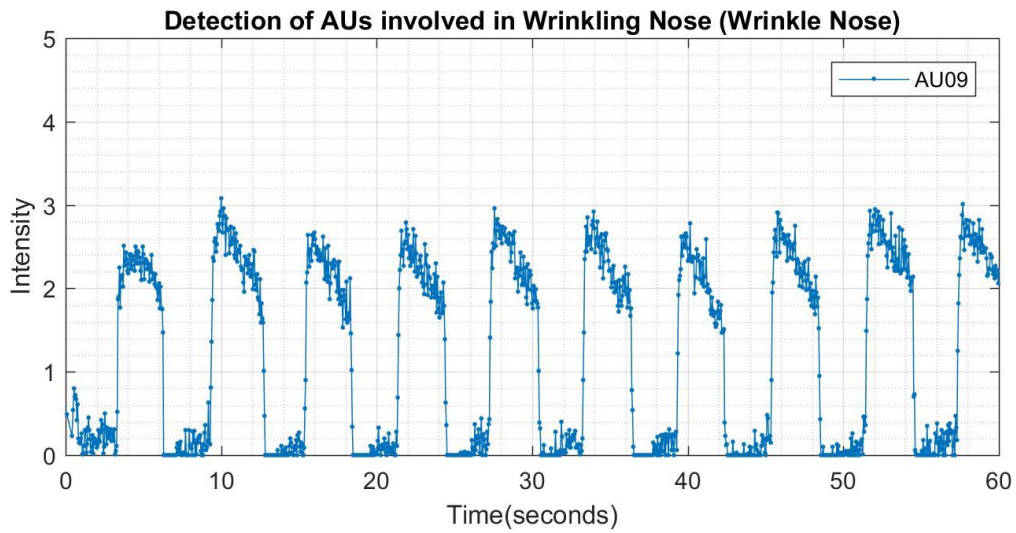


FIGURE 75: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

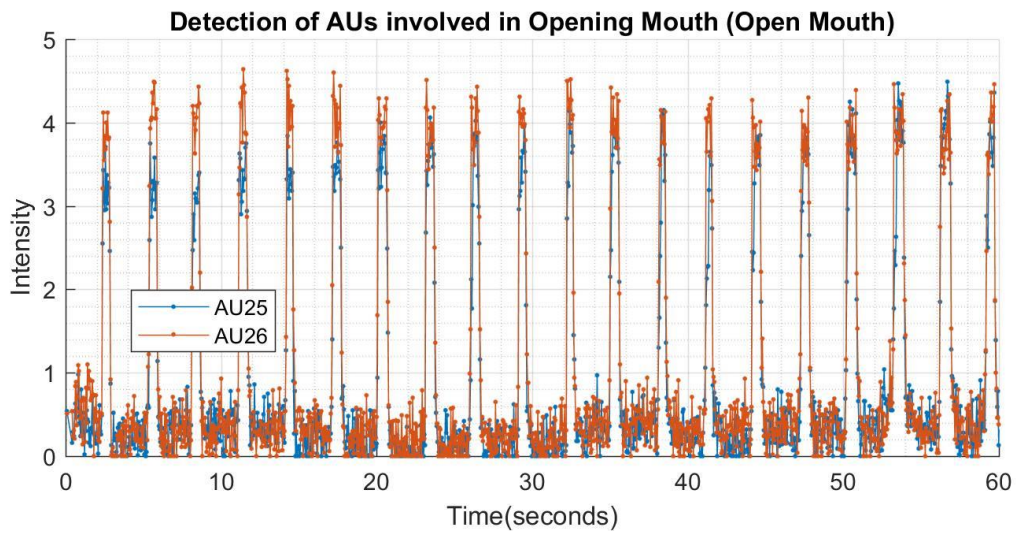


FIGURE 76: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

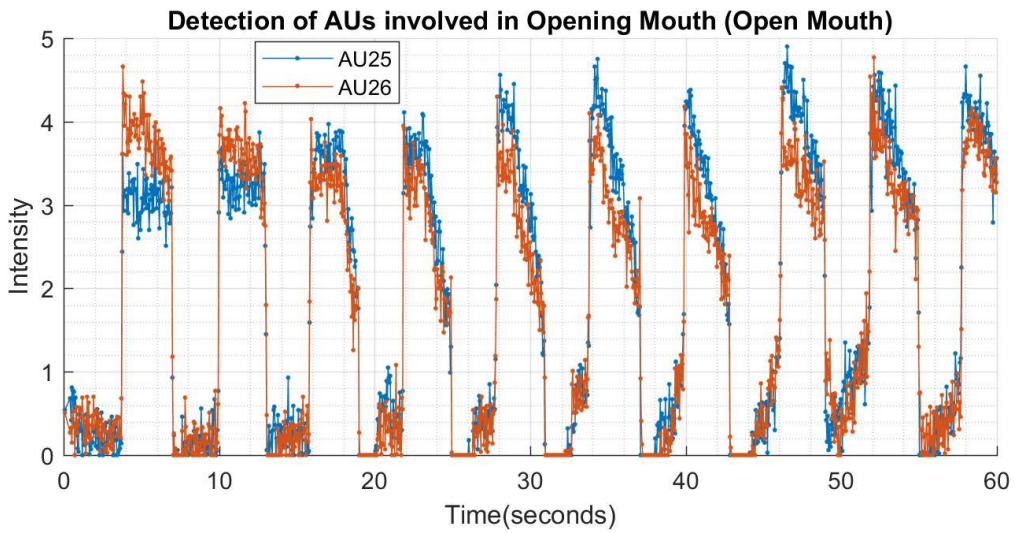


FIGURE 77: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

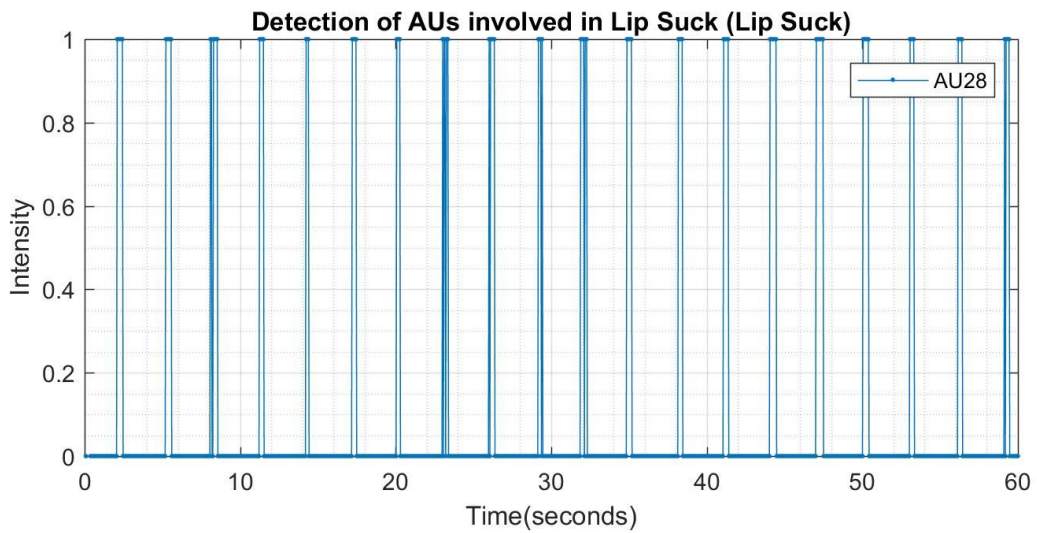


FIGURE 78: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

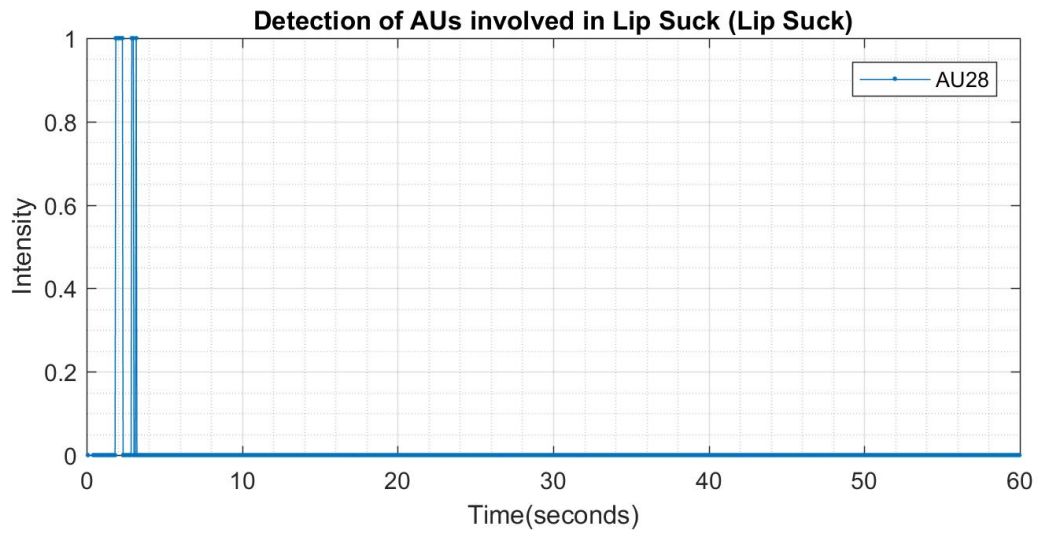


FIGURE 79: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

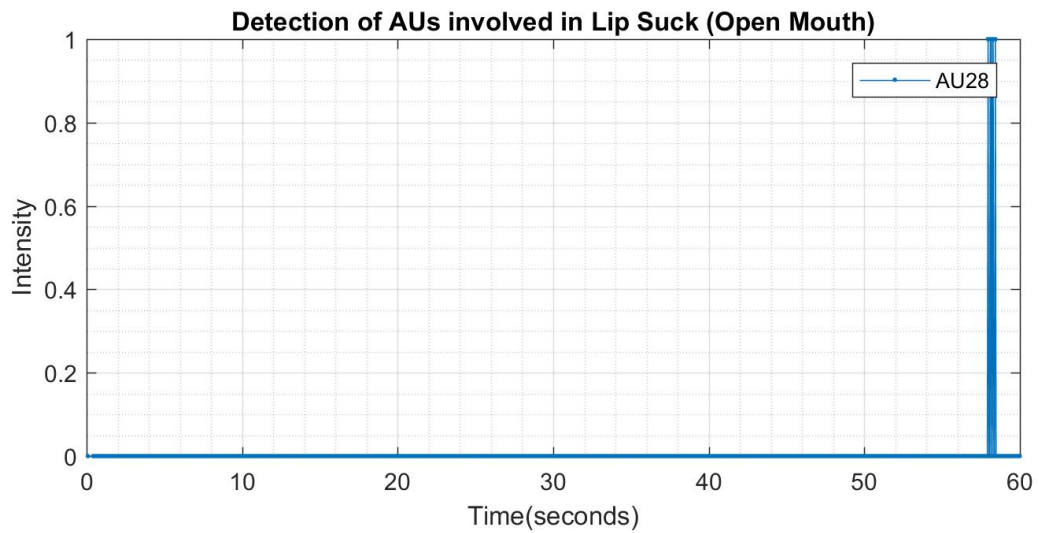


FIGURE 80: DETECTION OF AU28 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

A.4 SUBJECT 3

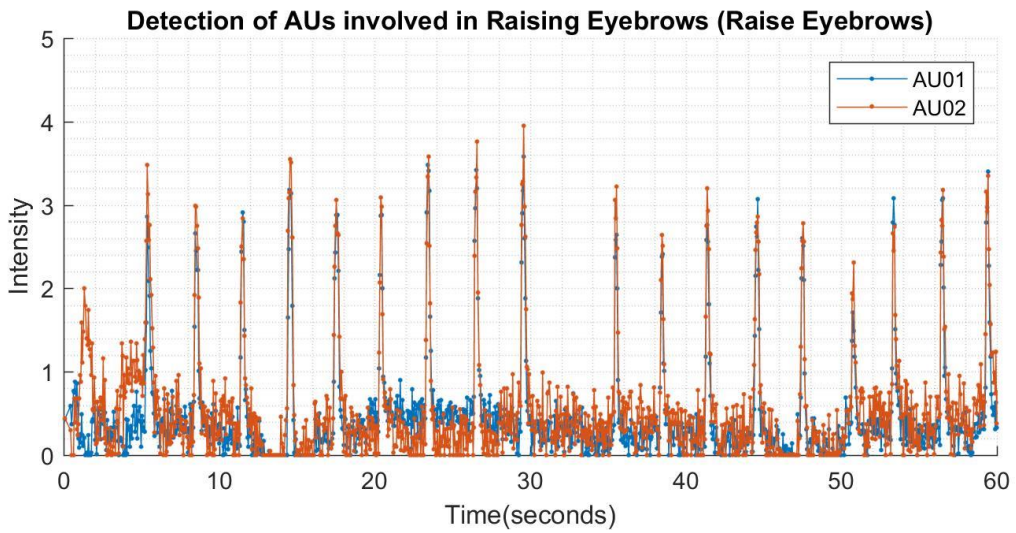


FIGURE 81: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

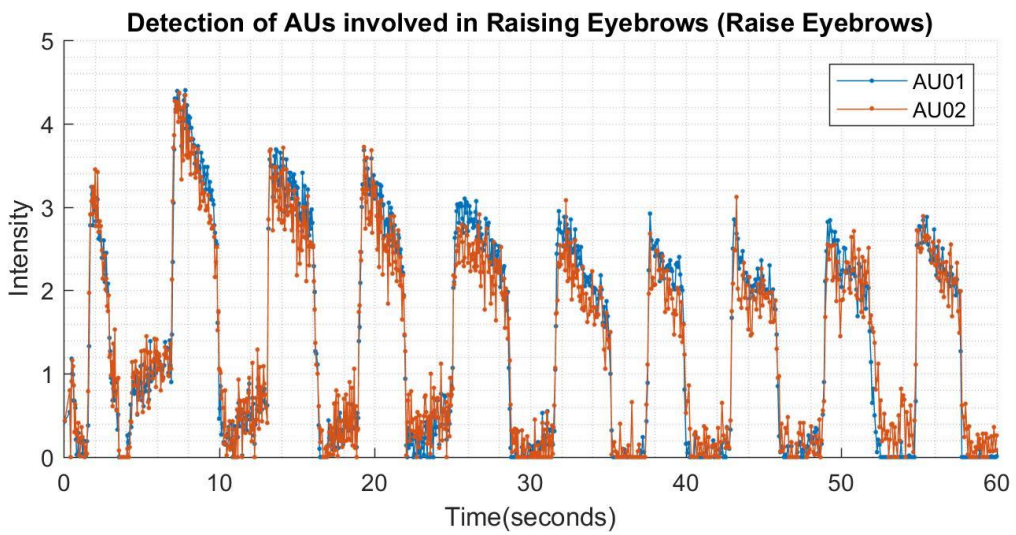


FIGURE 82: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

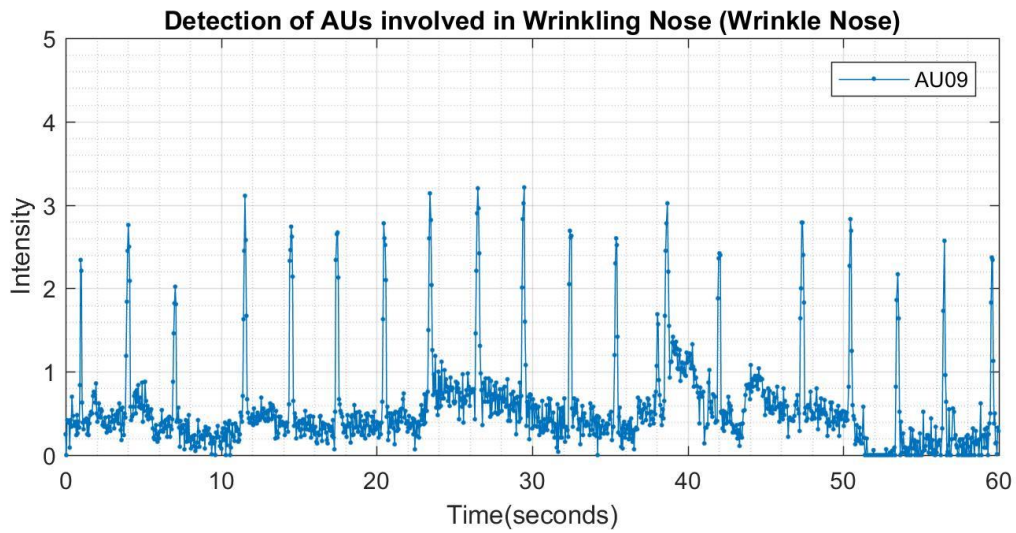


FIGURE 83: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

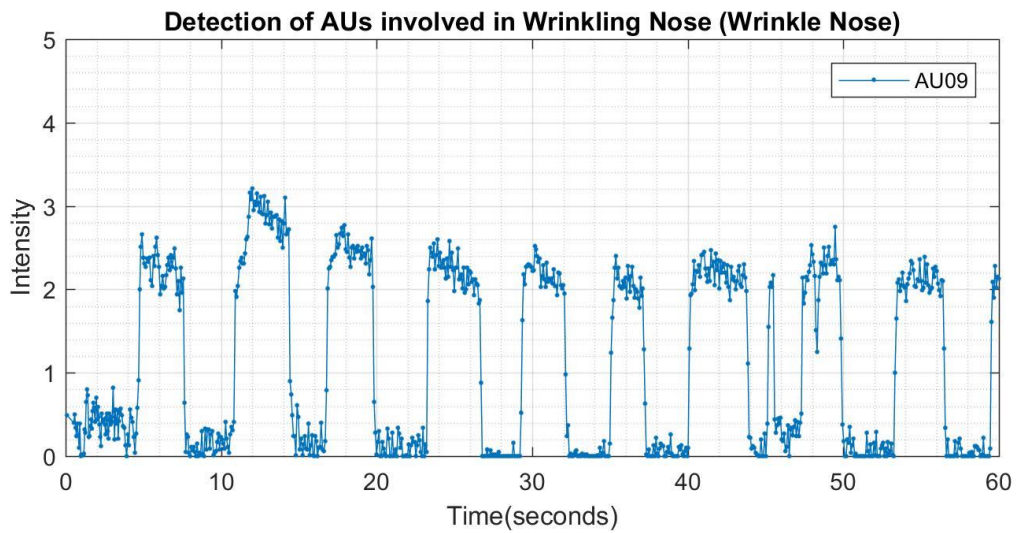


FIGURE 84: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

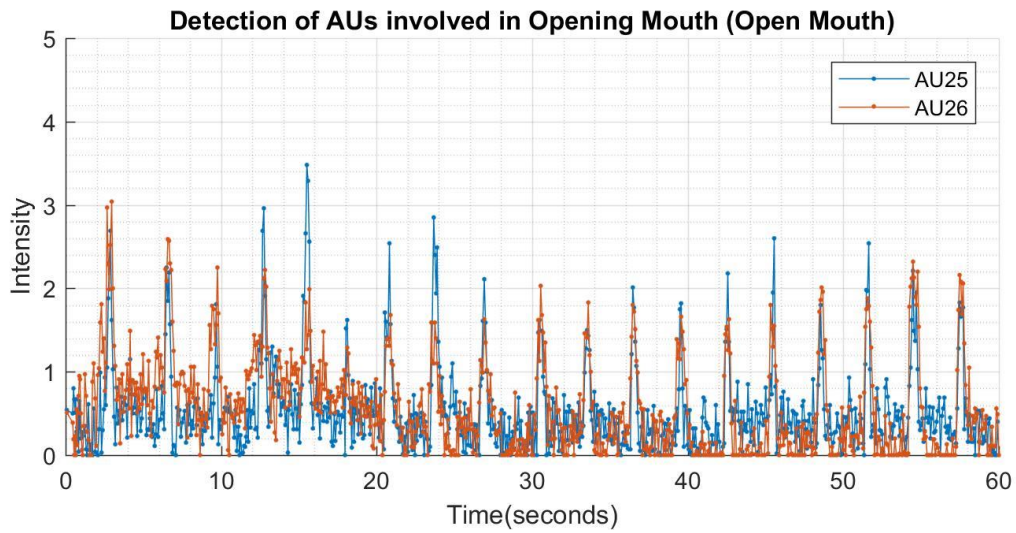


FIGURE 85: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

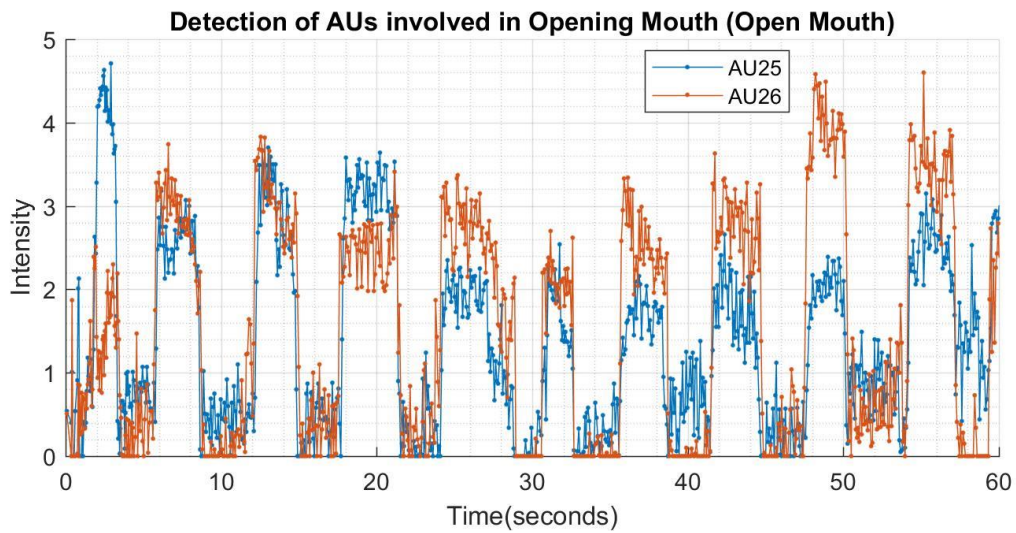


FIGURE 86: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

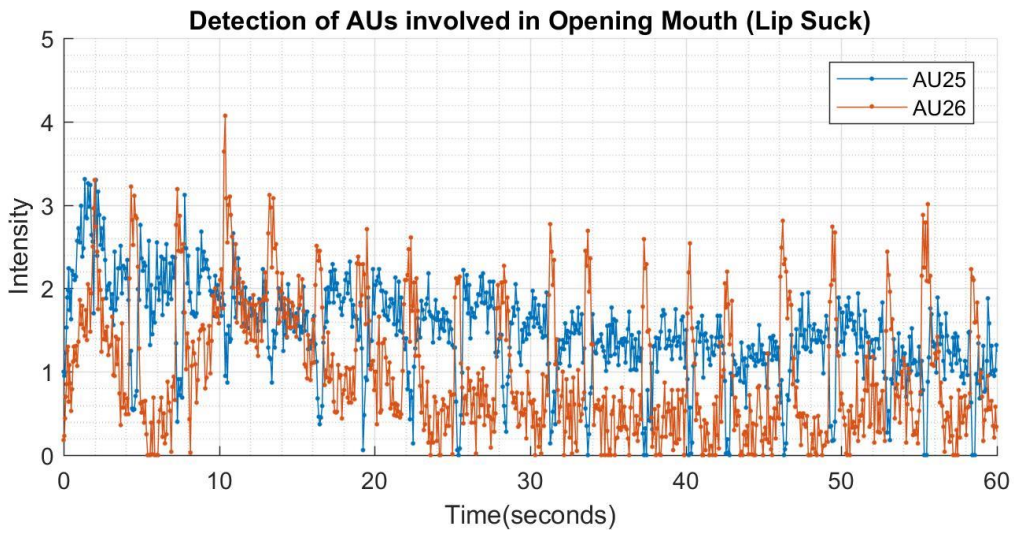


FIGURE 87: DETECTION OF AU25 AND AU26 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

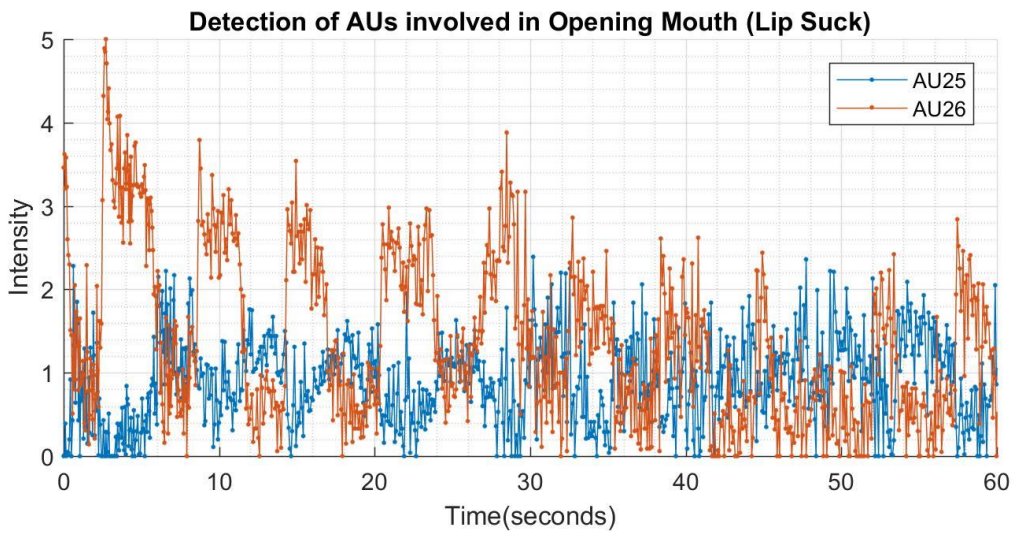


FIGURE 88: DETECTION OF AU25 AND AU26 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

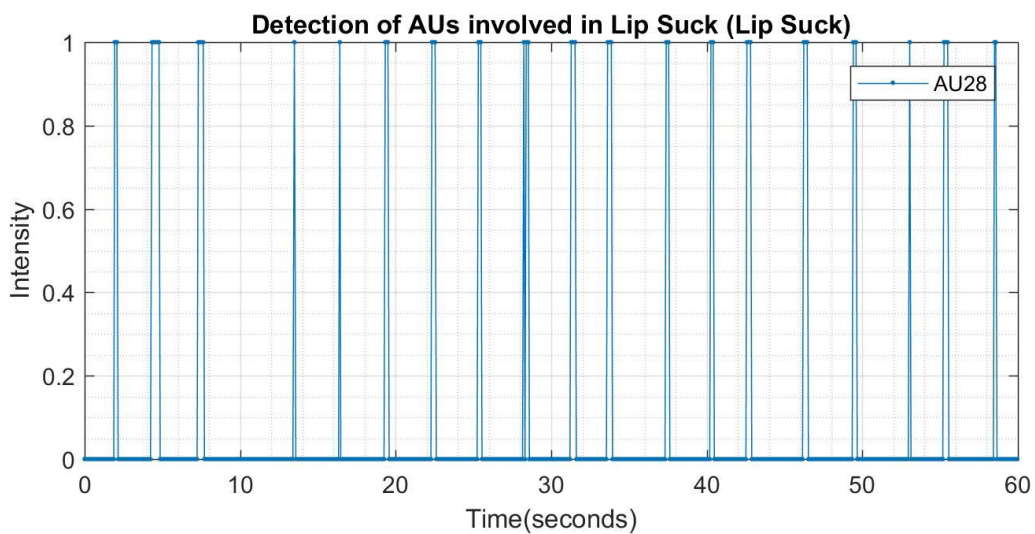


FIGURE 89: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

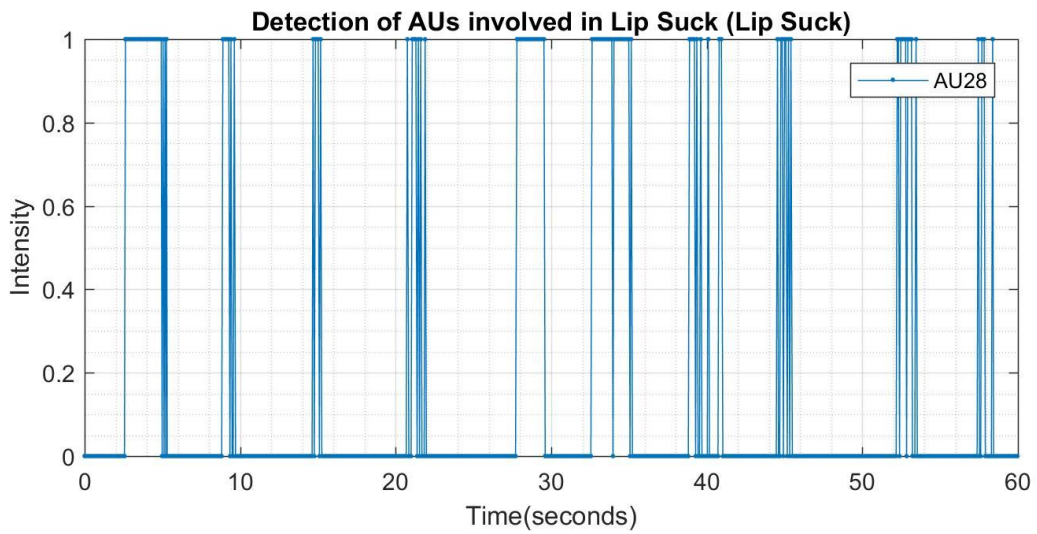


FIGURE 90: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

A.5 SUBJECT 4

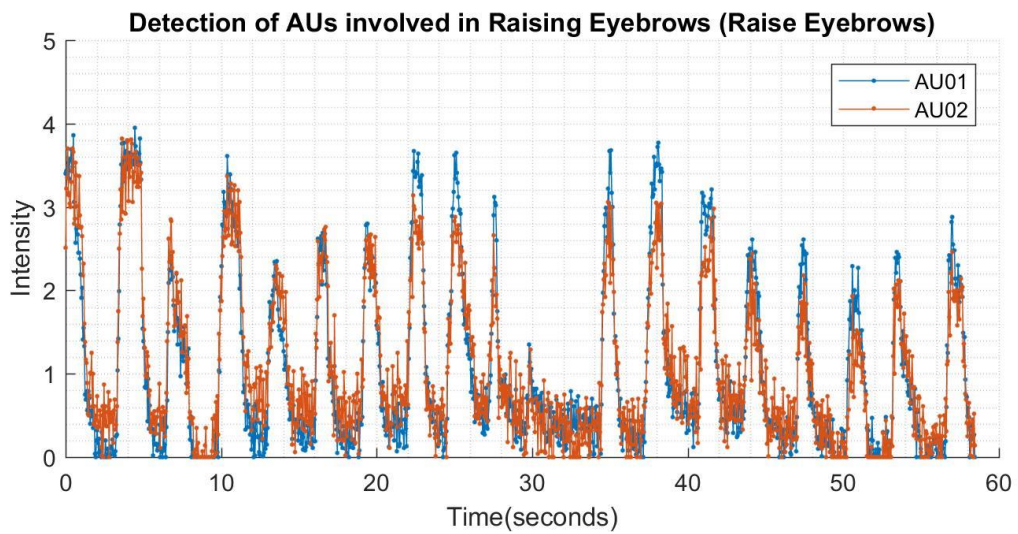


FIGURE 91: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

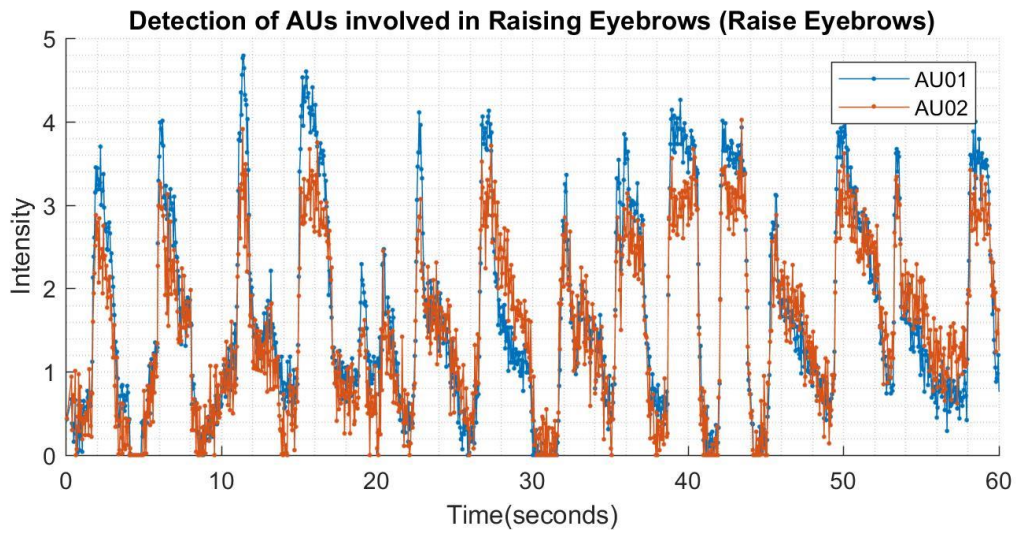


FIGURE 92: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

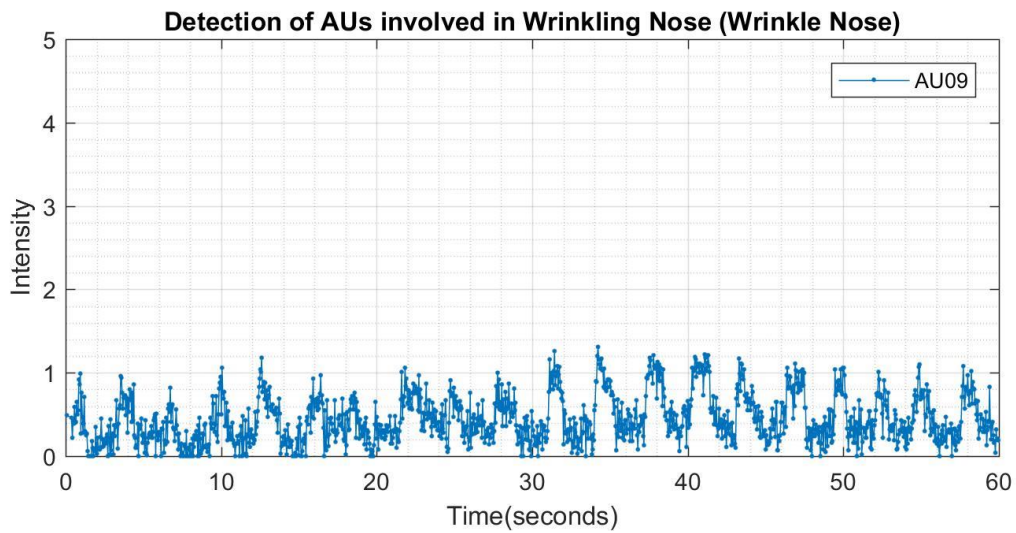


FIGURE 93: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

long

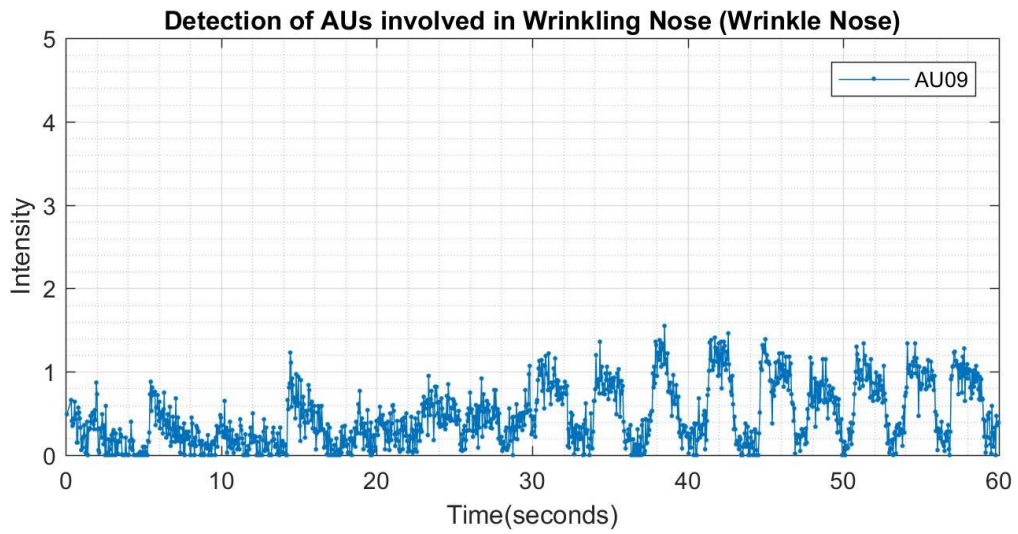


FIGURE 94: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

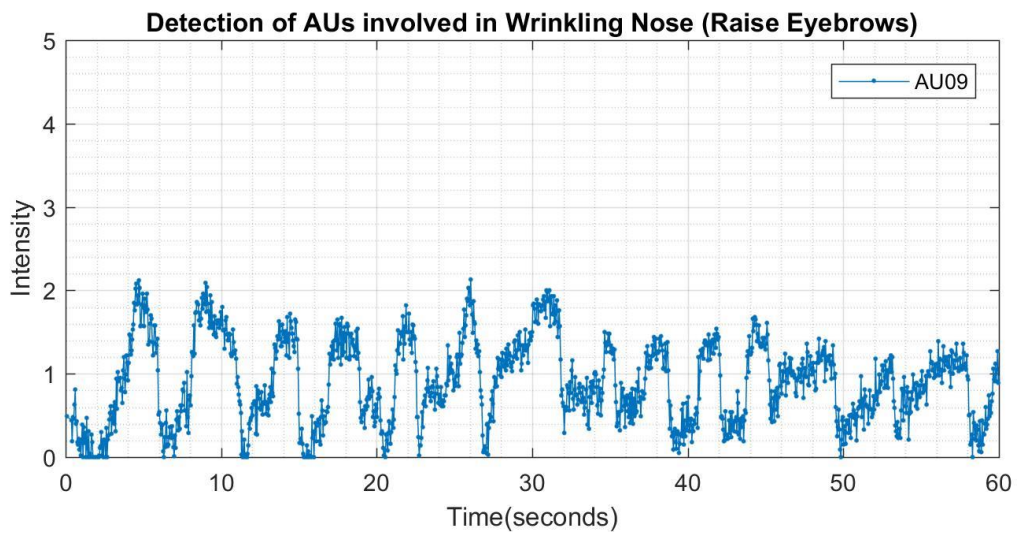


FIGURE 95: DETECTION OF AU09 DURING THE RAISE EYEBROWS GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

:

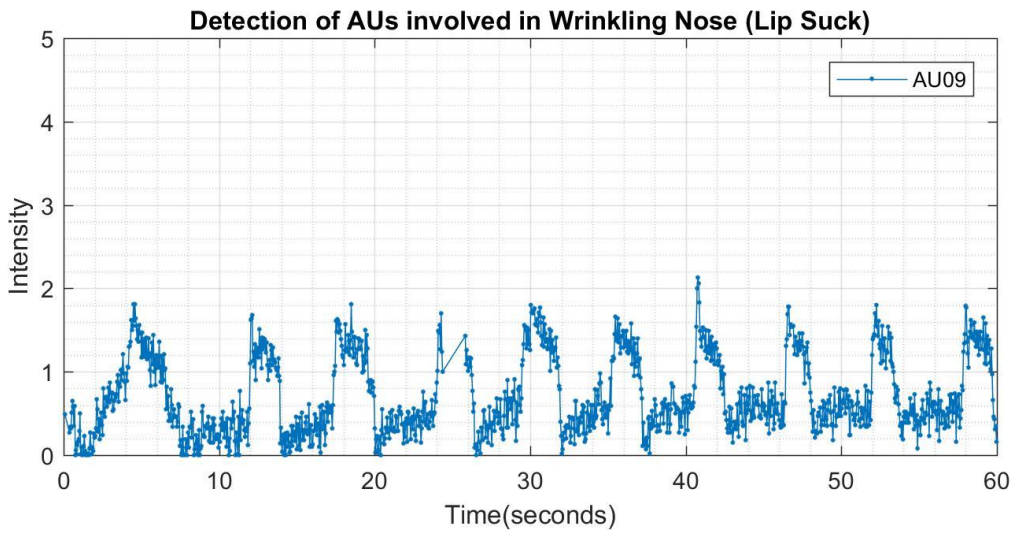


FIGURE 96: DETECTION OF AU09 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

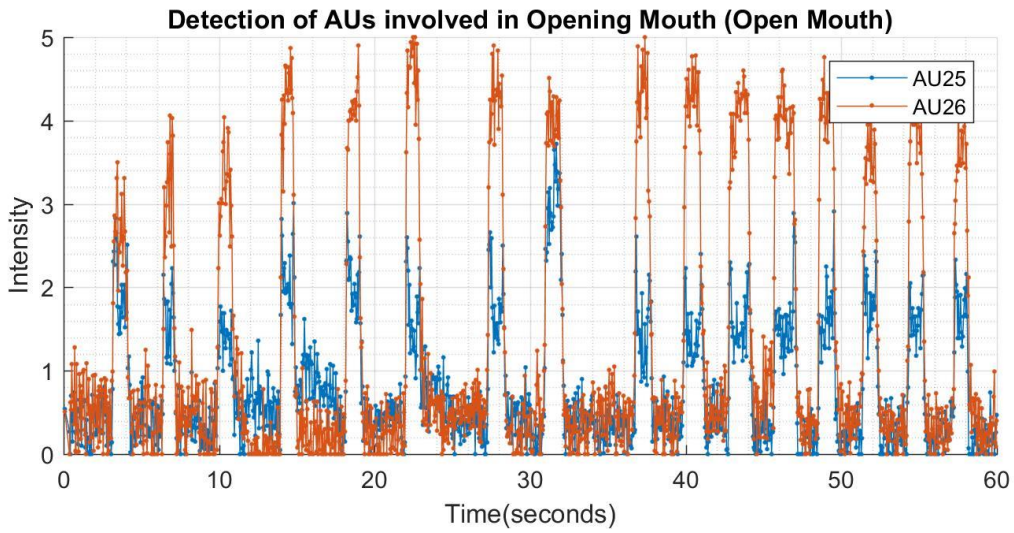


FIGURE 97: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

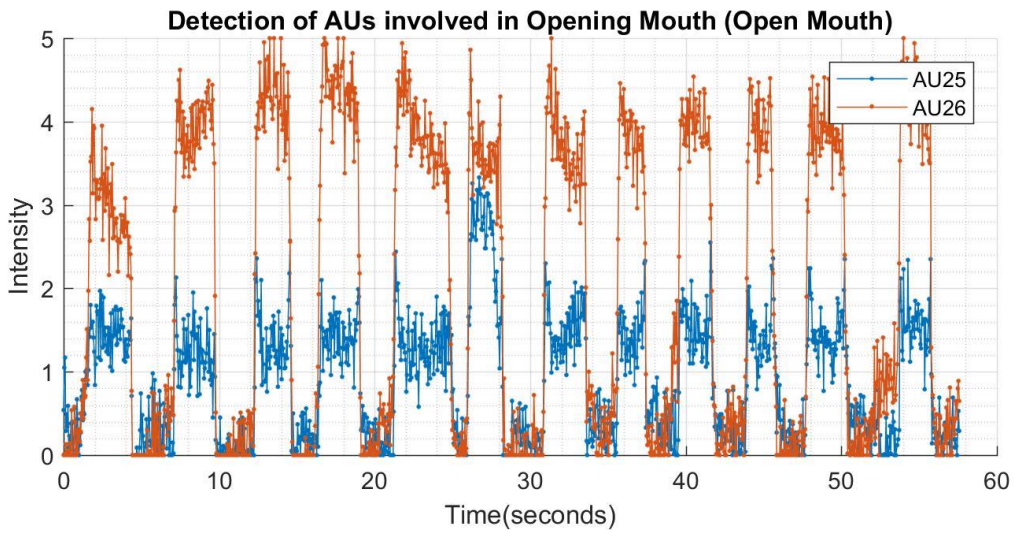


FIGURE 98: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

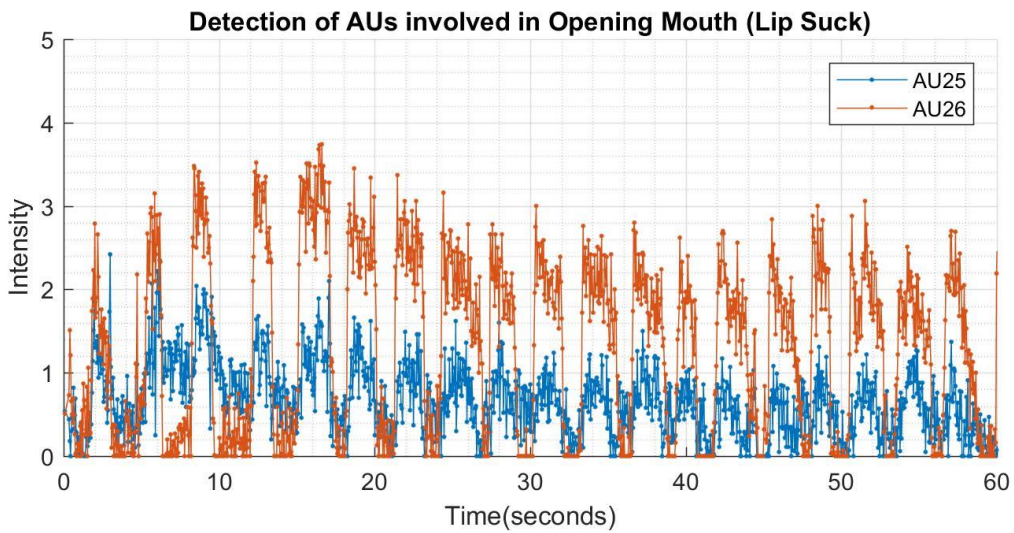


FIGURE 99: DETECTION OF AU25 AND AU26 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

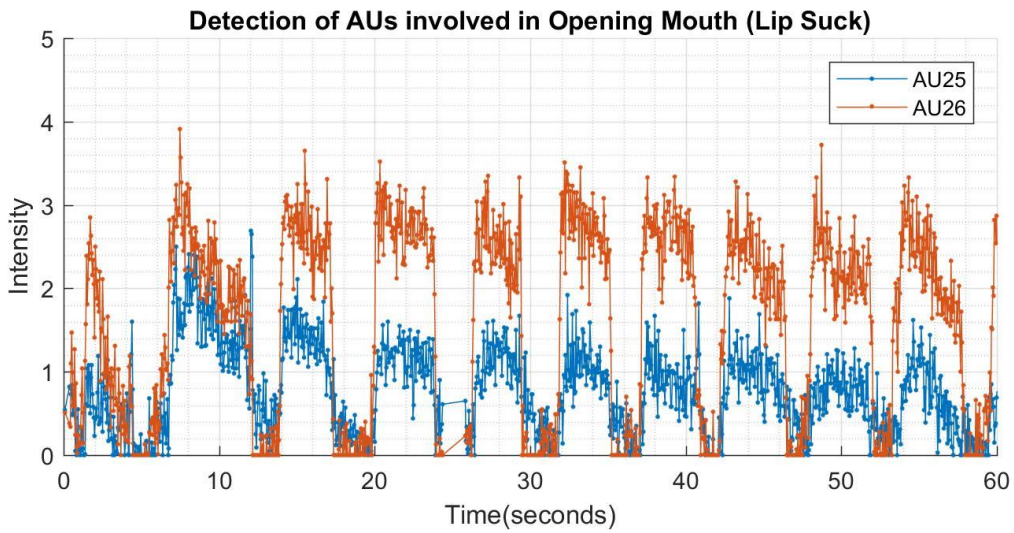


FIGURE 100: DETECTION OF AU25 AND AU26 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

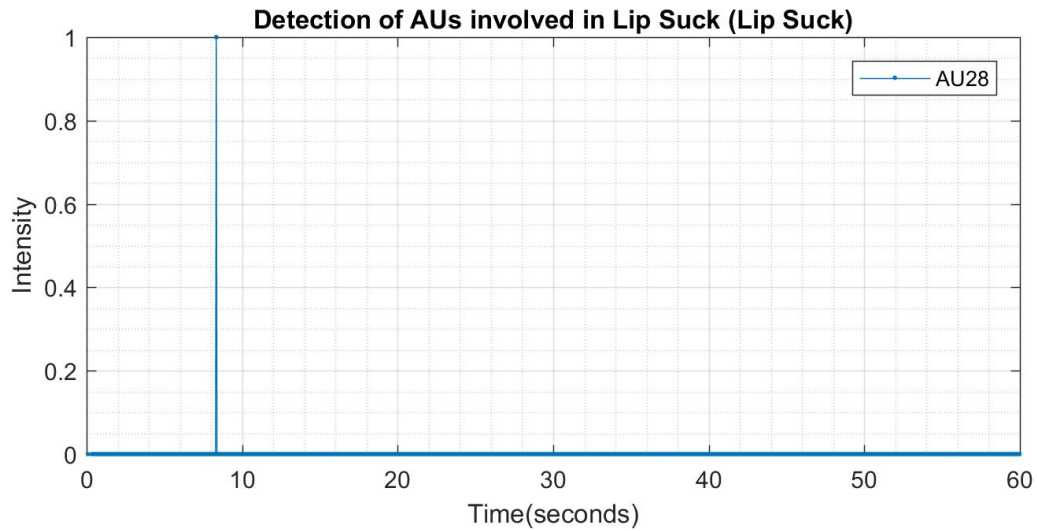


FIGURE 101: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

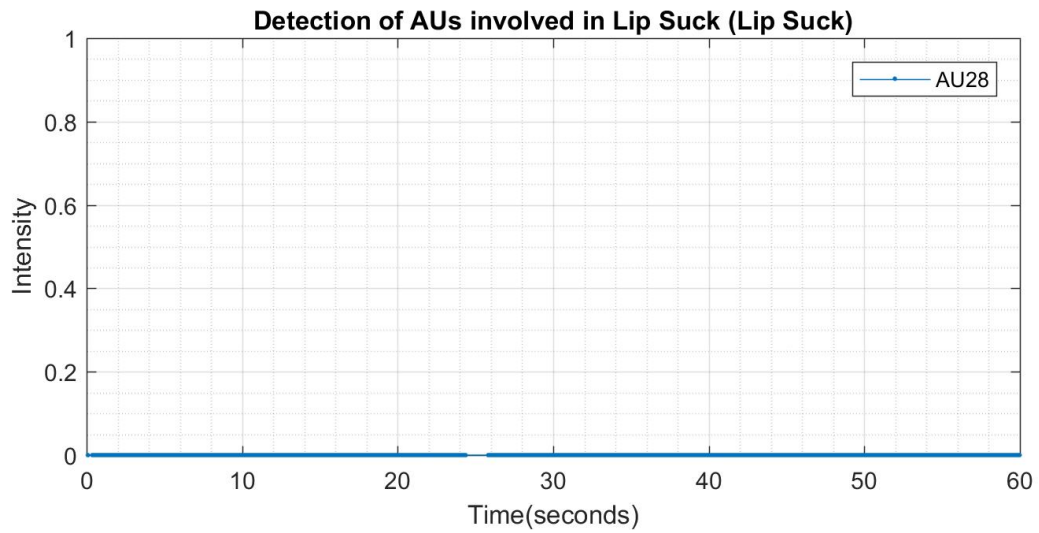


FIGURE 102: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

A.6 SUBJECT 5

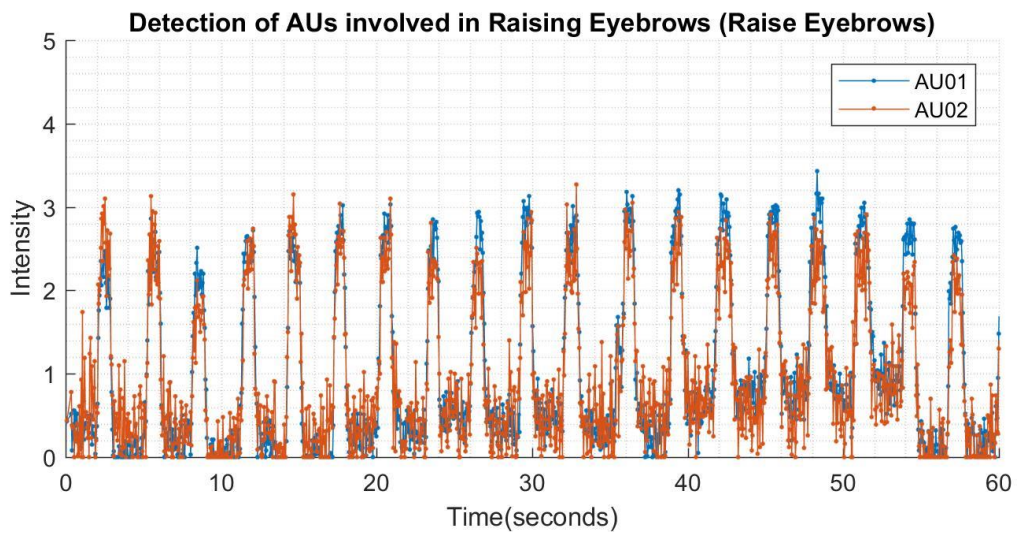


FIGURE 103: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

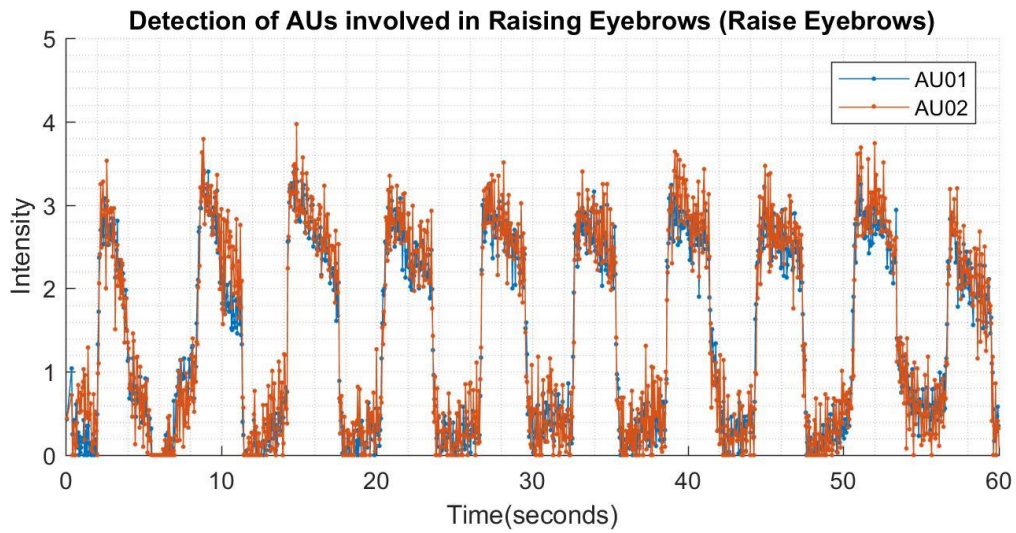


FIGURE 104: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

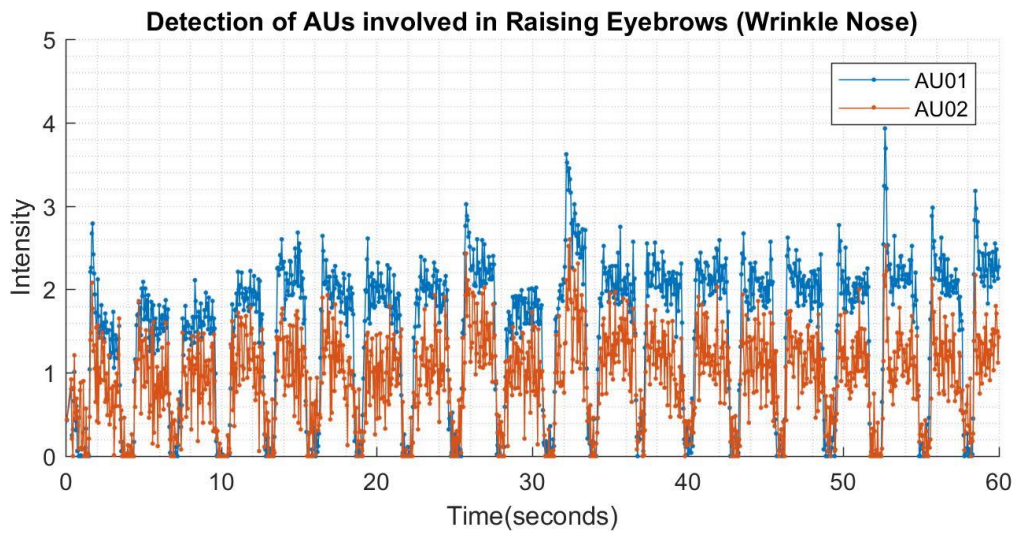


FIGURE 105: DETECTION OF AU01 AND AU02 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

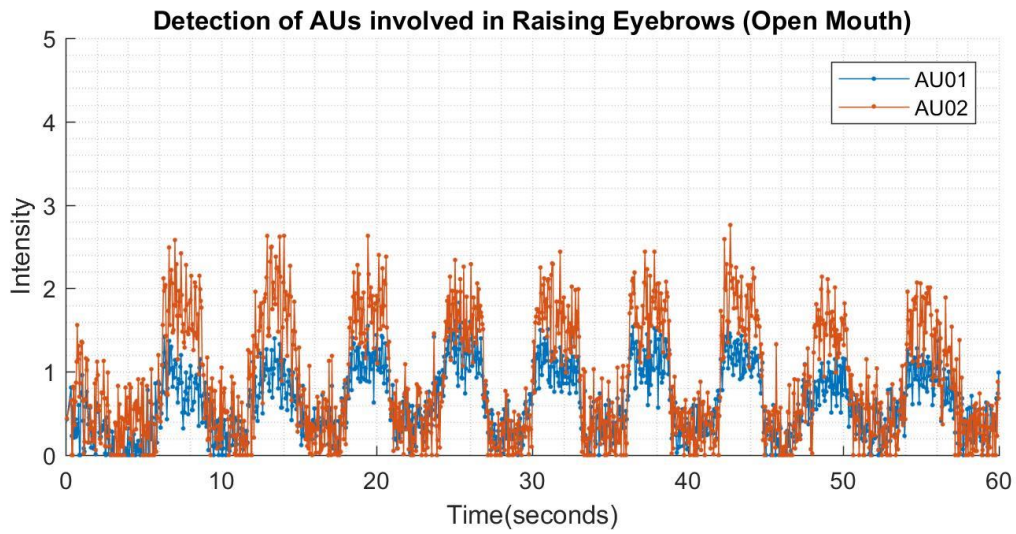


FIGURE 106: DETECTION OF AU01 AND AU02 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

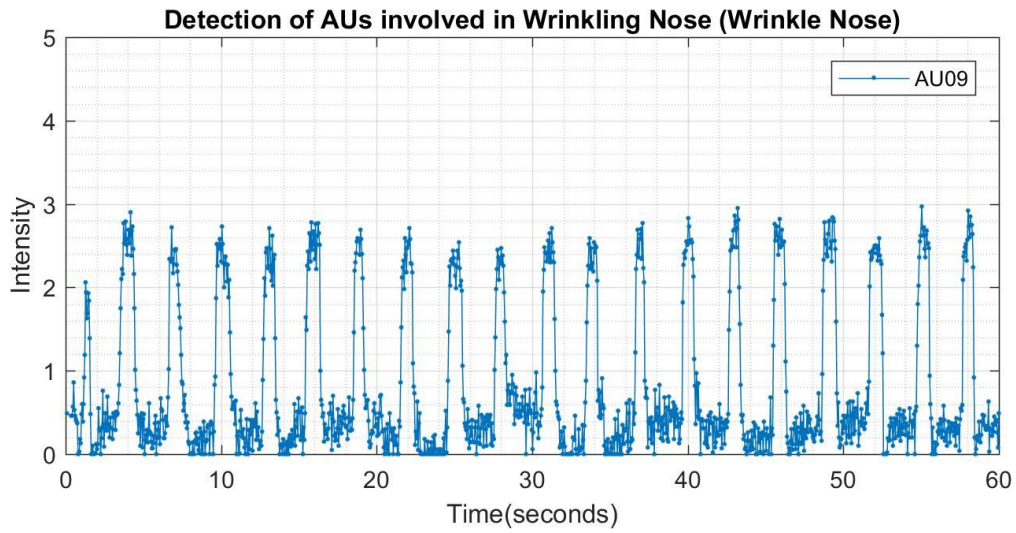


FIGURE 107: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

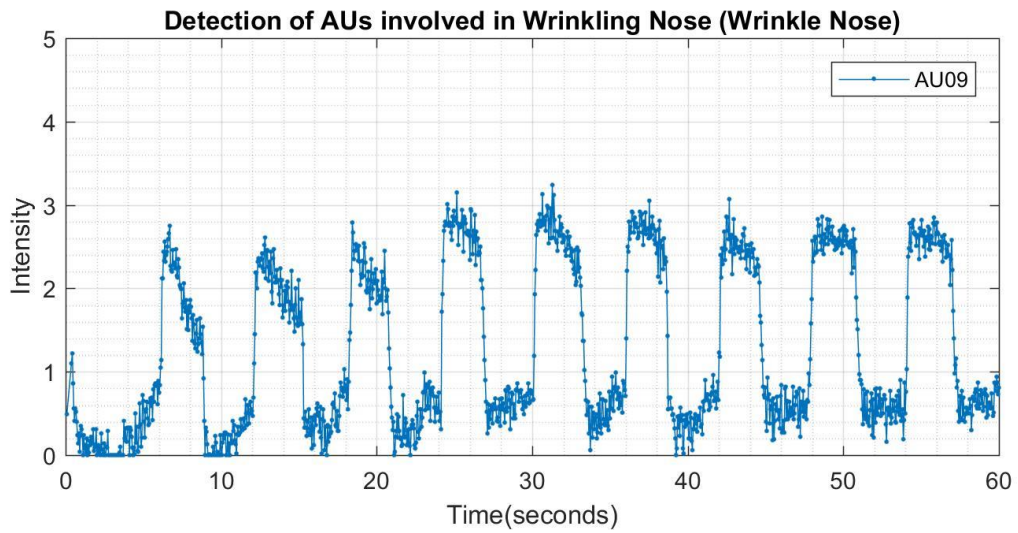


FIGURE 108: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

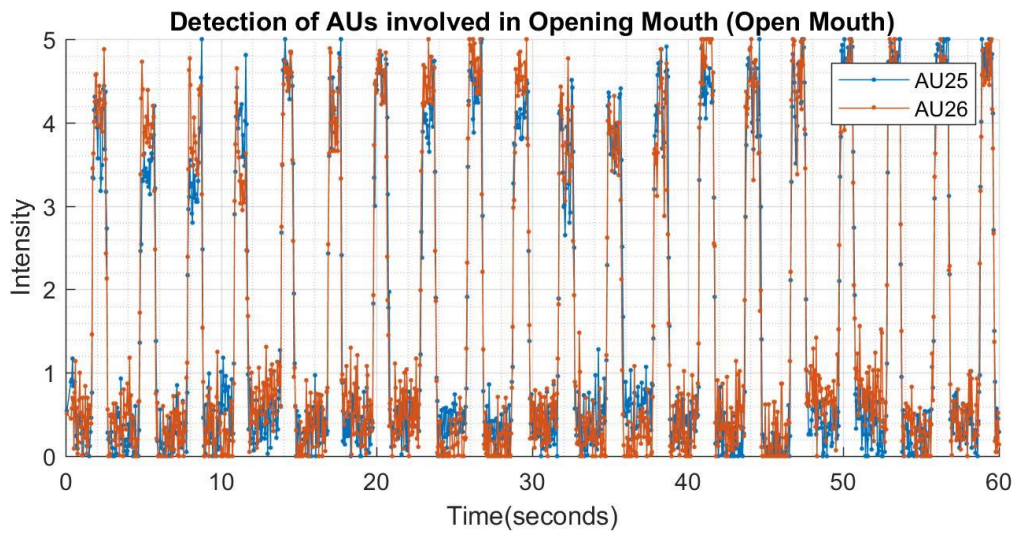


FIGURE 109: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

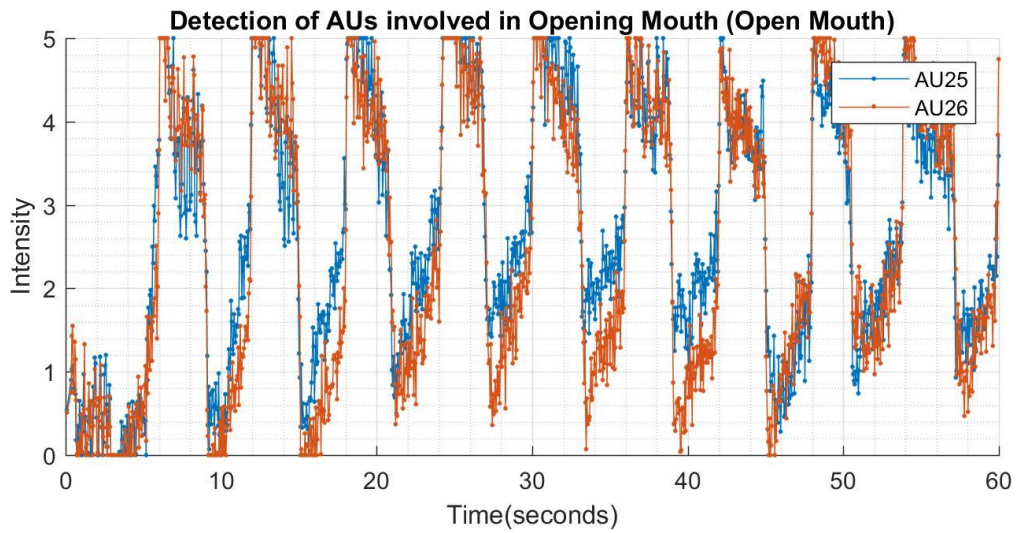


FIGURE 110: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

:

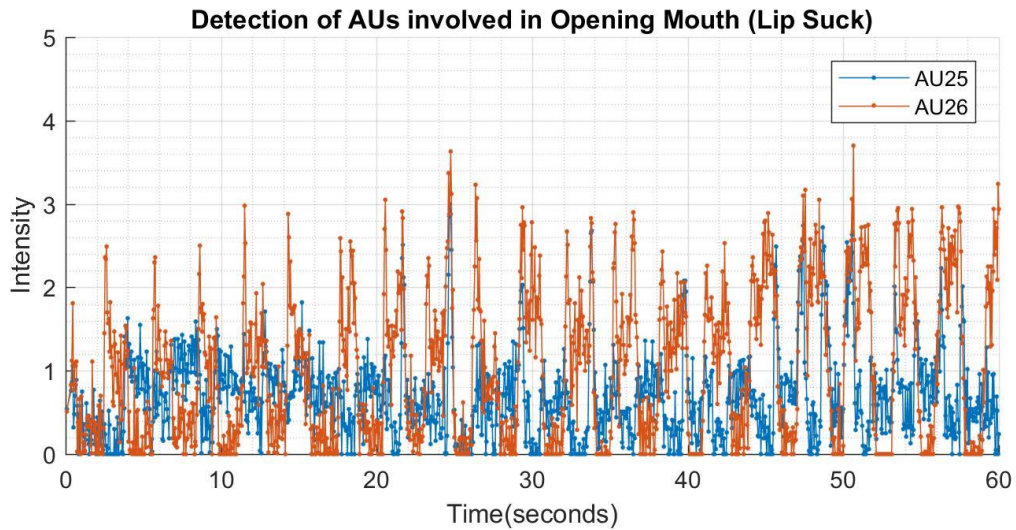


FIGURE 111: DETECTION OF AU25 AND AU26 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

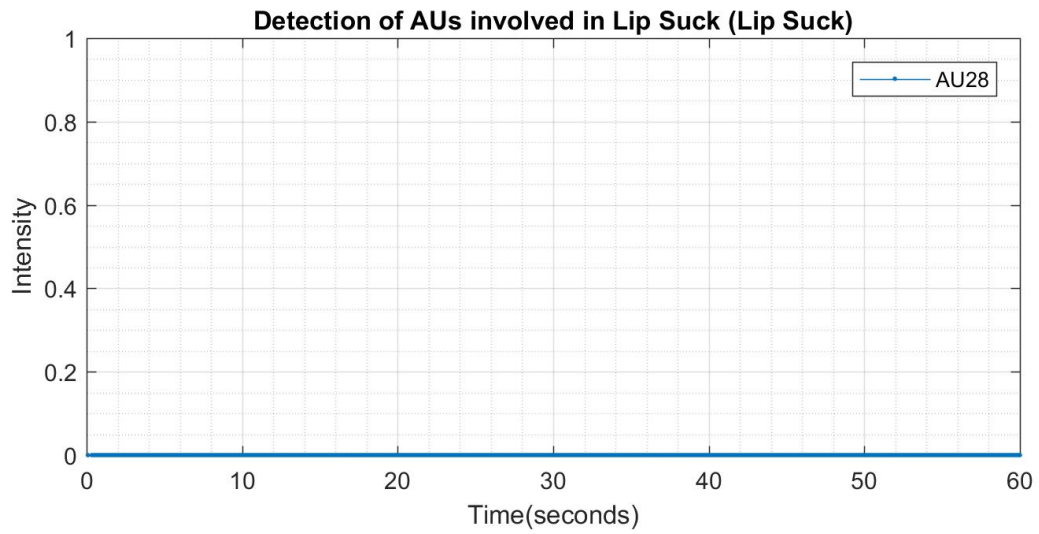


FIGURE 112: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

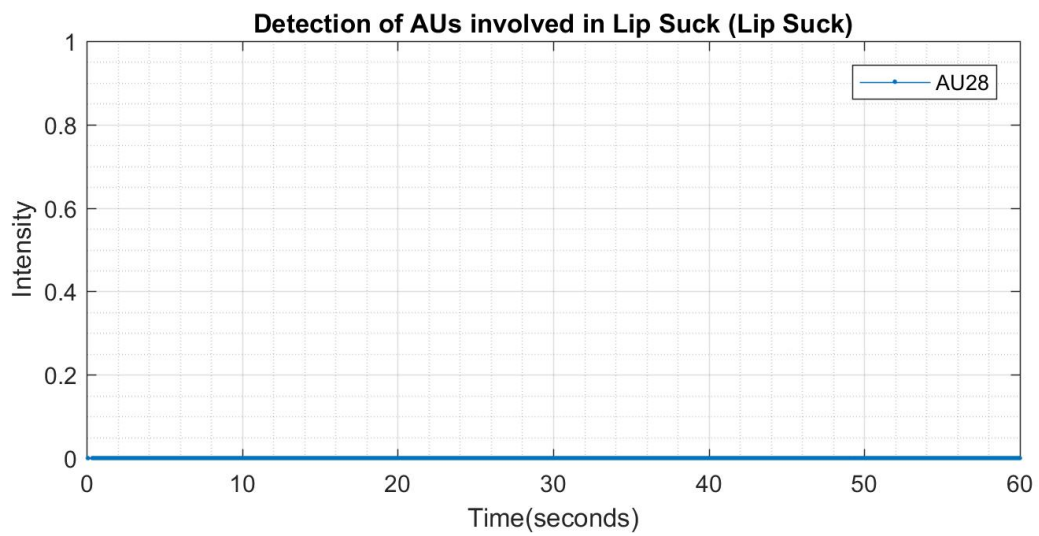


FIGURE 113: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

A.7 SUBJECT 6

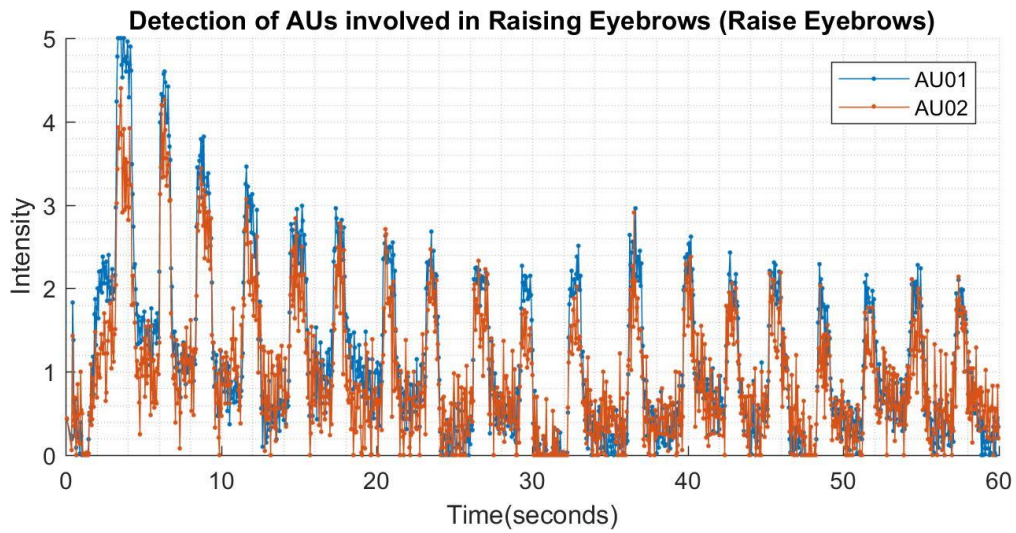


FIGURE 114: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

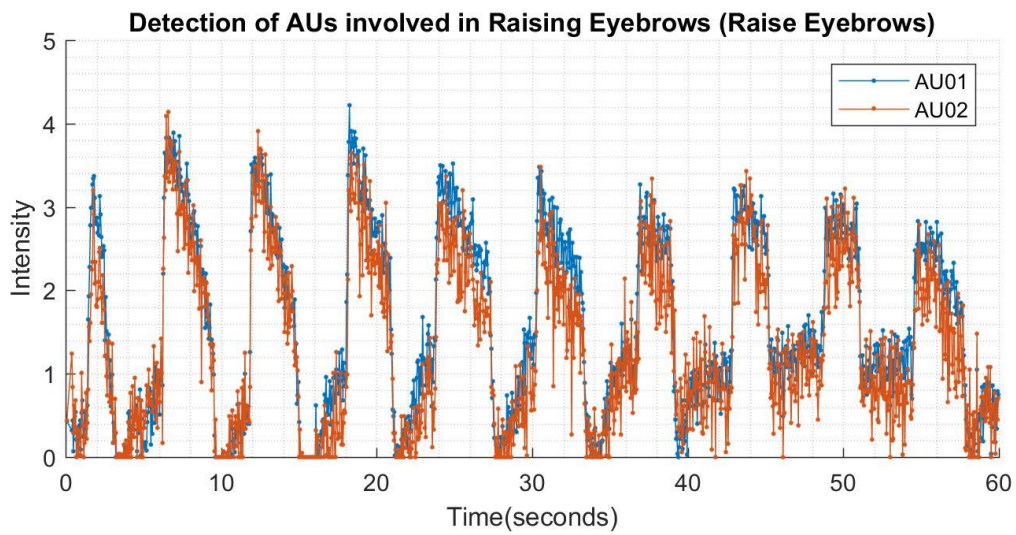


FIGURE 115: DETECTION OF AU01 AND AU02 DURING THE RAISE EYEBROW GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

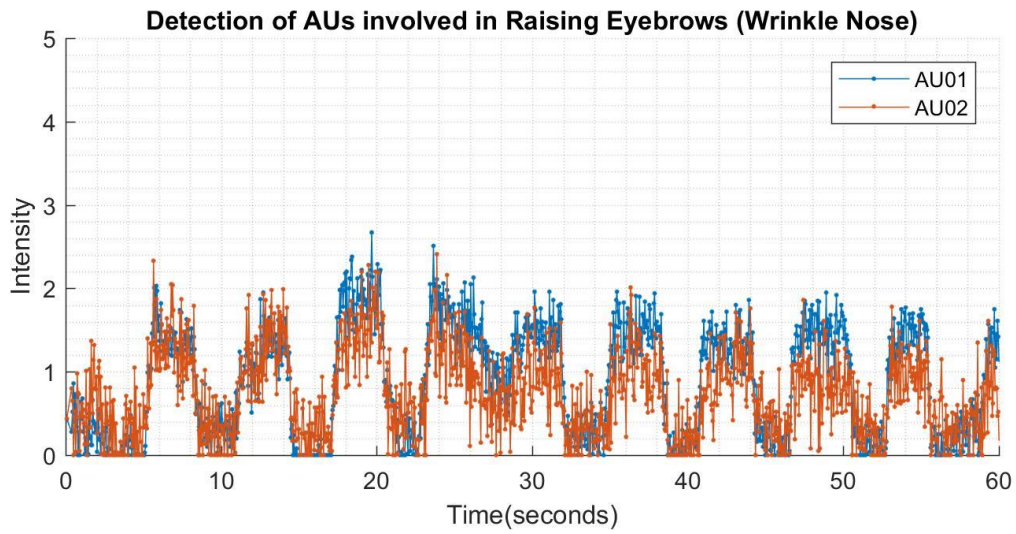


FIGURE 116: DETECTION OF AU01 AND AU02 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

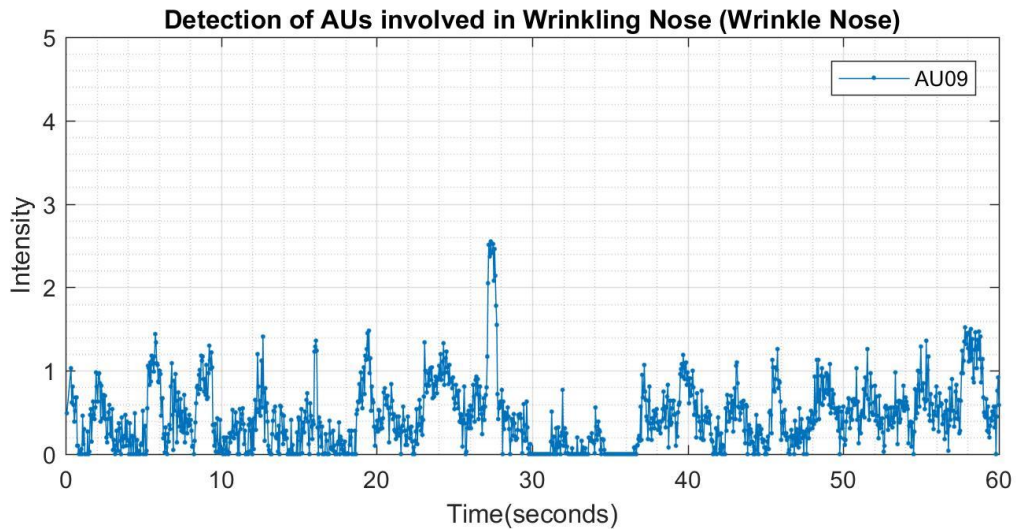


FIGURE 117: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

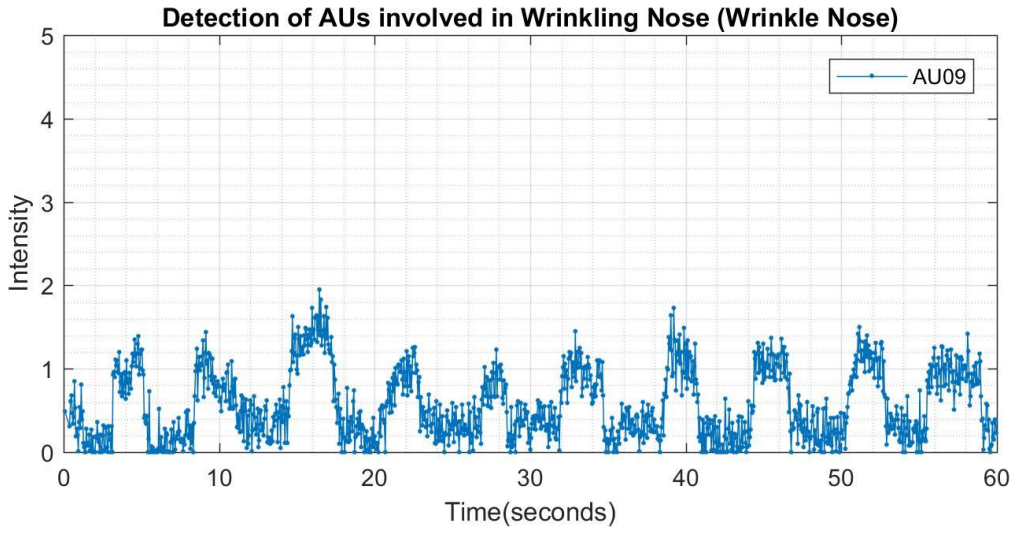


FIGURE 118: DETECTION OF AU09 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

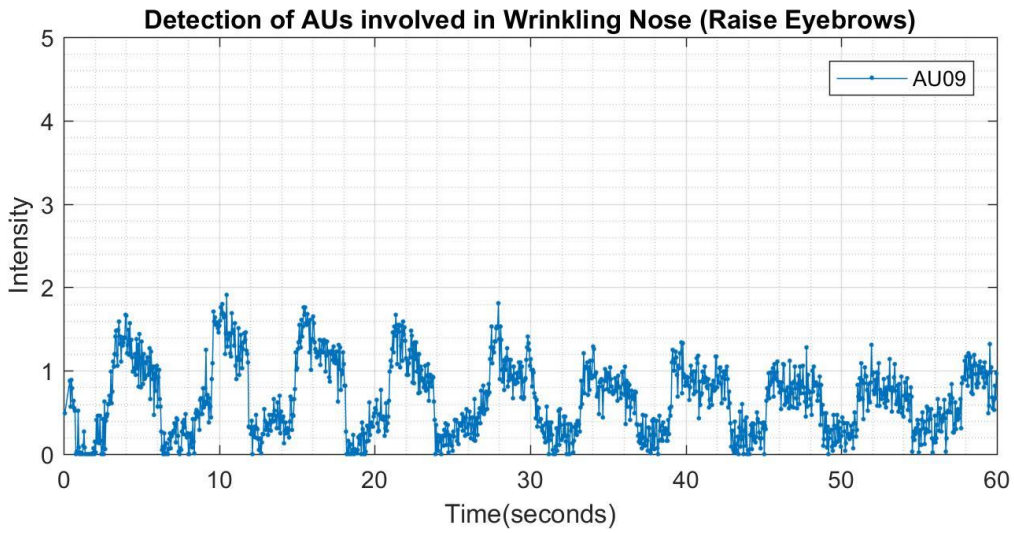


FIGURE 119: DETECTION OF AU09 DURING THE RAISE EYEBROWS GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

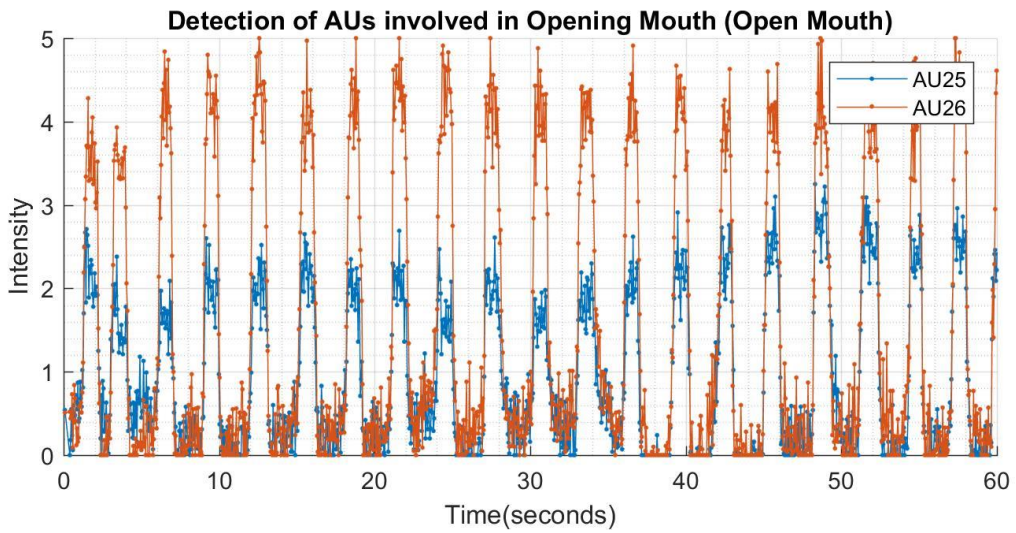


FIGURE 120: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

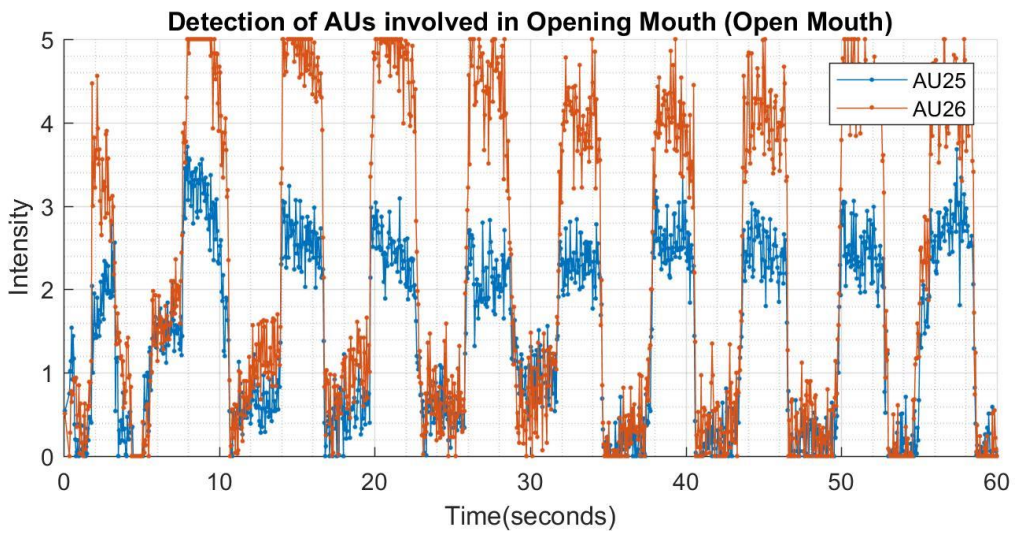


FIGURE 121: DETECTION OF AU25 AND AU26 DURING THE OPEN MOUTH GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

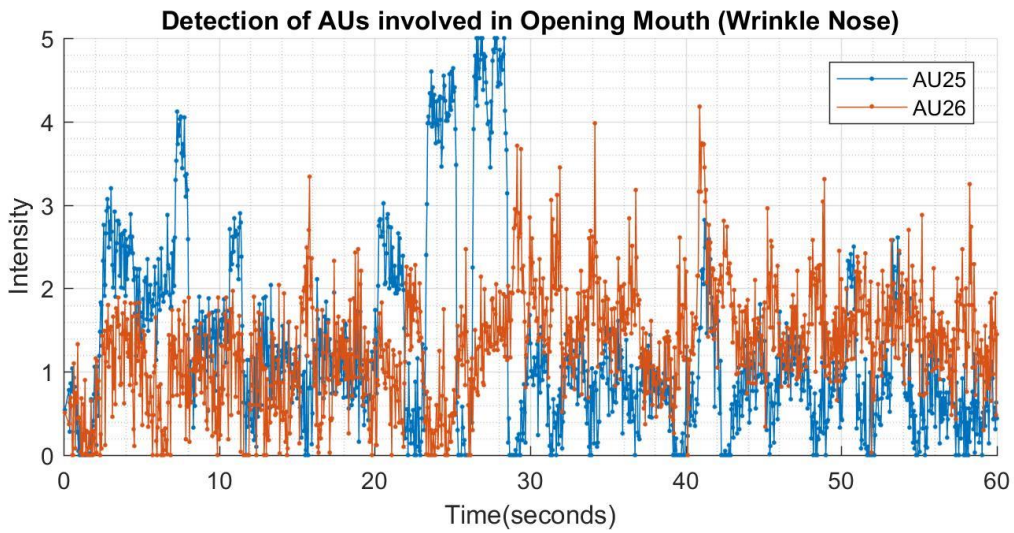


FIGURE 122: DETECTION OF AU25 AND AU26 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

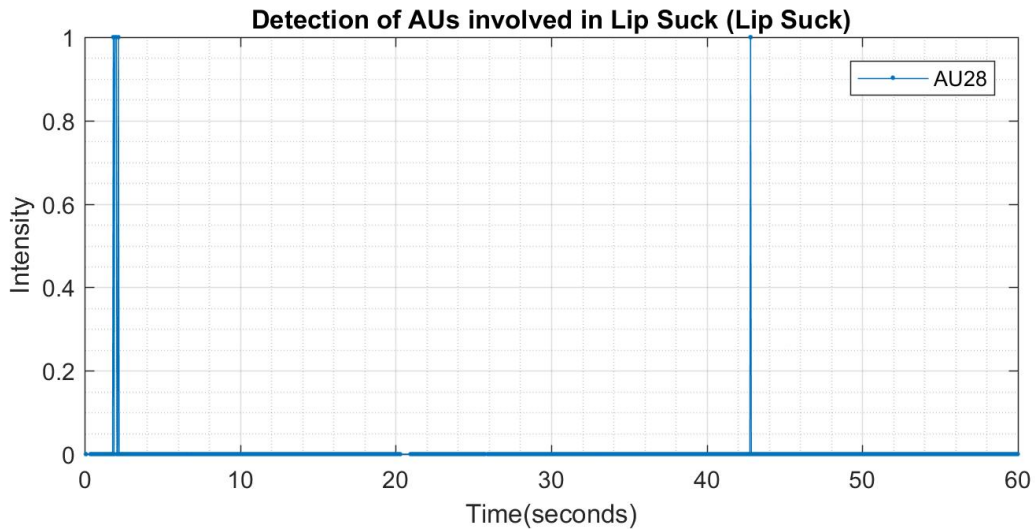


FIGURE 123: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

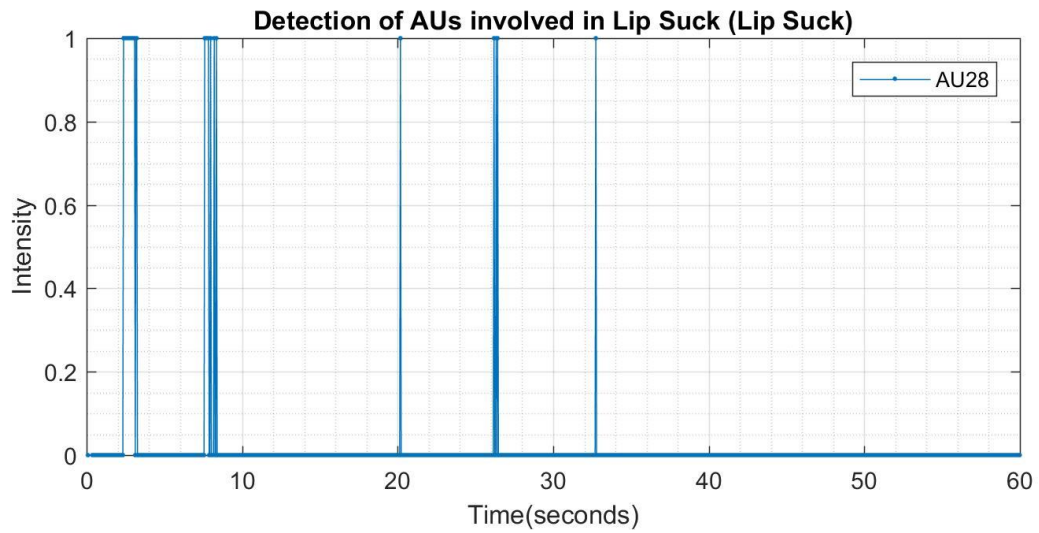


FIGURE 124: DETECTION OF AU28 DURING THE LIP SUCK GESTURE PERFORMED PERIODICALLY FOR LONG PERIODS OF TIME

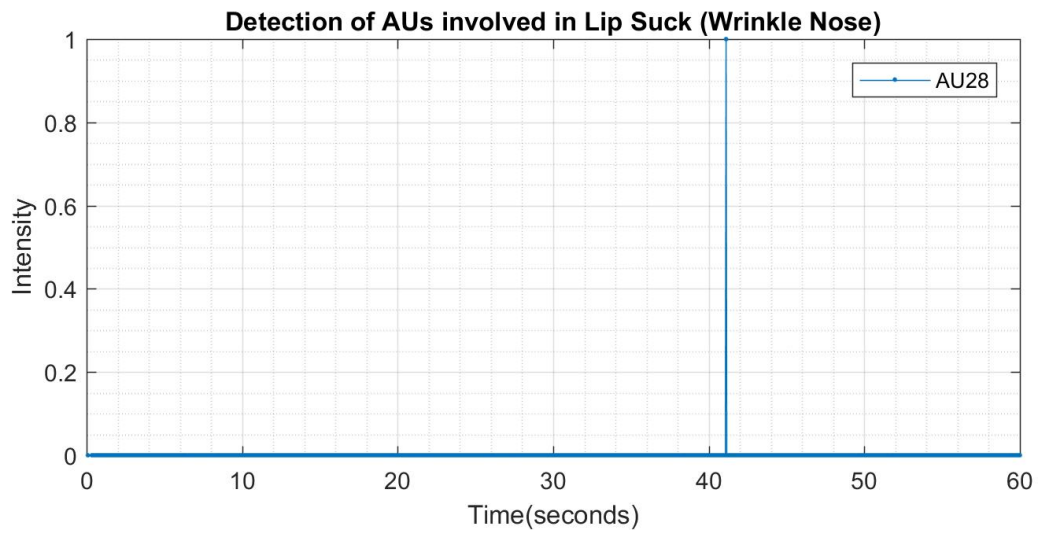


FIGURE 125: DETECTION OF AU28 DURING THE WRINKLE NOSE GESTURE PERFORMED PERIODICALLY FOR SHORT PERIODS OF TIME

A.8 – INDIVIDUAL PARAMETER SETS

The absolute lower limits were taken from the neutral position and the limits for the short and long actions were estimated from the relevant graphs.

Subject 1

Action units	Absolute lower limit	Limits for short actions		Limits for long actions	
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	0.65	2	3	1.6	3.8
AU02	1.1	2	3.2	1.6	3.8
AU09	0.57	0.8	2.2	0.4	1.4
AU25	0.86	1.8	3.1	2	4.6
AU26	0.88	1.6	3.2	1.4	3.85
AU28	0	0	1	0	1

Subject 2

Action units	Absolute lower limit	Limits for short actions		Limits for long actions	
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	1.02	2.2	3	1.6	4.4
AU02	1.58	2.2	3.8	1.6	4.2
AU09	0.97	2.4	3.3	1.4	3.2
AU25	1.11	2.4	4.6	2.4	4.95
AU26	1.2	3.4	4.8	1.6	4.8
AU28	0	0	1	0	1

Subject 3

Action units	Absolute lower limit	Limits for short actions		Limits for long actions	
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	1.22	2	3.7	1.6	4.5
AU02	1.51	2	4.1	1.4	4.5
AU09	0.87	2	3.3	1.8	3.4
AU25	1.58	1.6	3.6	1.5	4.8
AU26	2.65 (~1.8)	1.6	3.2	2	4.7
AU28	0	0	1	0	1

Subject 4

Action units	Absolute lower limit	Limits for short actions		Limits for long actions	
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	0.83	2	4	1.5	4.9
AU02	1.46	1.8	4	1.5	4.2
AU09	0.75	0.8	1.4	0.8	1.7

AU25	1.4	1	3.9	1	3.4
AU26	1.6	2	5	2	5
AU28	0	0	1	0	0

Subject 5

Action units	Absolute lower limit	Limits for short actions		Limits for long actions	
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	1.63	2	3.6	1.5	3.5
AU02	2.27	1.8	3.4	1.5	4.1
AU09	1.2	1.9	3.1	1.2	3.4
AU25	2.04	2.6	5	2.8	5
AU26	2.04	2.9	5	3	5
AU28	1	0	0	0	0

*This subject moved around during the neutral test, which contributes to the high neutral intensities

Subject 6

Action units	Absolute lower limit	Limits for short actions		Limits for long actions	
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	1.5	1.8	5	1.6	4.3
AU02	1.81	1.5	4.5	1.3	4.4
AU09	1.31	0.8	2.7	0.6	2.1
AU25	2.47	1.2	3.3	1.4	3.9
AU26	1.85	3	5	2.6	5
AU28	0	0	1	0	1

*This subject had difficulties wrinkling their nose

Subject 7

Action units	Absolute lower limit	Limits for short actions		Limits for long actions	
		<i>Min</i>	<i>Max</i>	<i>Min</i>	<i>Max</i>
AU01	1.32	2.6	3.4	1.6	4.1
AU02	1.82	2.6	4	1.6	4.5
AU09	0.88	1.2	2.2	1	2.6
AU25	1.17	1.4	3	1.4	3.8
AU26	1.42	3.2	4.4	2.3	4.7
AU28	0	0	1	0	1

APPENDIX B— OPENFACE CONTROL INTERFACE

B.1 OPENFACEINTERPRETER.H

```
#ifndef _OPENFACEINTERPRETER_h_
#define _OPENFACEINTERPRETER_h_

#include <string>
#include <vector>

//Gesture class that stores different gestures and information about them
struct Command {
    //Available gestures
    const std::vector<std::string> gestures = { { "raise eyebrows"},{ "wrinkle nose"},{ "open mouth"},{
"lip suck"},{ "tilt head left"},{ "tilt head right"} };
    //Range of detection for each gesture
    std::vector<std::pair<double, double>> limits{ {1.4,4.5},{1.4,2.5},{2.1,4},{0,1},{20,100},{-100,-20}}; //
lower and upper limits for AU01, AU09, AU25, AU28, tiltL and tiltR
    std::vector<std::pair<double, double>> limits2{ {2,4.5},{0,0},{2,4.7}}; // lower and upper limits for
AU02, nothing, AU26
    std::string name;
    std::string command;
    std::string commandName;
    int selection;
    int occurrence;
    double lowlim;
    double highlim;
    double lowlim2;
    double highlim2;
    bool send = false;

public:
    void setGesture(std::string, int);
    void detectGesture(const std::vector<std::pair<std::string, double> >&, const
std::vector<std::pair<std::string, double> >&, cv::Vec6d);
    std::string Command::checkCommand();
    void sendCommand(std::string c);

};

#endif
```

B.2 OPENFACEINTERPRETER.CPP – MODIFIED FEATUREEXTRACTION.CPP

```
////////////////////////////////////
// Copyright (C) 2017, Carnegie Mellon University and University of Cambridge,
// all rights reserved.
//
// ACADEMIC OR NON-PROFIT ORGANIZATION NONCOMMERCIAL RESEARCH USE ONLY
//
// BY USING OR DOWNLOADING THE SOFTWARE, YOU ARE AGREEING TO THE TERMS OF THIS
// LICENSE AGREEMENT.
// IF YOU DO NOT AGREE WITH THESE TERMS, YOU MAY NOT USE OR DOWNLOAD THE SOFTWARE.
//
////////////////////////////////////

// OpenFaceInterpreter.cpp : Defines the entry point for the feature extraction console application.

// Local includes
#include "LandmarkCoreIncludes.h"

#include <Face_utils.h>
#include <FaceAnalyser.h>
#include <GazeEstimation.h>
#include <RecorderOpenFace.h>
#include <RecorderOpenFaceParameters.h>
#include <SequenceCapture.h>
#include <Visualizer.h>
#include <VisualizationUtils.h>
#include "OpenFaceInterpreter.h"
#include "SerialPort.h"

//Added for arduino code
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <string>

#ifndef CONFIG_DIR
#define CONFIG_DIR "~"
#endif

#define INFO_STREAM( stream ) \
std::cout << stream << std::endl

#define WARN_STREAM( stream ) \
std::cout << "Warning: " << stream << std::endl

#define ERROR_STREAM( stream ) \
std::cout << "Error: " << stream << std::endl

static void printErrorAndAbort(const std::string & error)
{
    std::cout << error << std::endl;
}

#define FATAL_STREAM( stream ) \
printErrorAndAbort( std::string( "Fatal error: " ) + stream )

using namespace std;

vector<string> get_arguments(int argc, char **argv)
{
    vector<string> arguments;

    // First argument is reserved for the name of the executable
```

```

    for (int i = 0; i < argc; ++i)
    {
        arguments.push_back(string(argv[i]));
    }
    return arguments;
}

//Code for arduino
//String for getting the output from arduino
char output[MAX_DATA_LENGTH];

/*Set portname (must contain these backslashes)*/
char *port_name = "\\.\COM4";

//String for incoming data
char incomingData[MAX_DATA_LENGTH];

//Initialise arduino
SerialPort arduino(port_name);

//Code for the command of wheelchair
void Command::setGesture(std::string c, int g) //takes the value for gesture and assigns it to the appropriate
command.
{
    selection = g;
    name = gestures[g];
    lowlim = limits[g].first;
    highlim = limits[g].second;
    lowlim2 = limits2[g].first;
    highlim2 = limits2[g].second;
    occurrence = 0;
    commandName = c;
    if(c == "forward") command = "w";
    if (c == "backward") command = "s";
    if (c == "left") command = "a";
    if (c == "right") command = "d";
    if (c == "halt") command = "q";
}

//Detects gesture and keeps track of consecutive occurrences
void Command::detectGesture(const std::vector<std::pair<std::string, double> >& au_r, const
std::vector<std::pair<std::string, double> >& au_c, cv::Vec6d pose)
{
    //Keeps track of which features are detected within range
    boolean occurred1 = false;
    boolean occurred2 = false;

    double Tx = pose[3] / 3.14 * 180; //Tx - up/down
    double Ty = pose[4] / 3.14 * 180; //Ty - rotation

    if (selection == 0) // raise eyebrows (AU01 and AU02)
    {
        for (size_t idx = 0; idx < au_r.size(); idx++)
        {
            if (au_r[idx].first == "AU01")
            {
                if (au_r[idx].second < highlim && au_r[idx].second > lowlim)
                {
                    occurred1 = true;
                }
            }
            if (au_r[idx].first == "AU02")
            {
                if (au_r[idx].second < highlim2 && au_r[idx].second > lowlim2)
                {
                    occurred2 = true;
                }
            }
        }
    }
}

```

```

    }
    }
}
else if (selection == 1) // Wrinkle nose (AU09)
{
    for (size_t idx = 0; idx < au_r.size(); idx++)
    {
        if (au_r[idx].first == "AU09")
        {
            if (au_r[idx].second < highlim && au_r[idx].second > lowlim)
            {
                occurred1 = true;
                occurred2 = true;
            }
        }
    }
}
else if ( selection == 2) // Open Mouth (AU25 and AU26)
{
    for (size_t idx = 0; idx < au_r.size(); idx++)
    {
        if (au_r[idx].first == "AU25")
        {
            if (au_r[idx].second < highlim && au_r[idx].second > lowlim)
            {
                occurred1 = true;
            }
        }
        if (au_r[idx].first == "AU26")
        {
            if (au_r[idx].second < highlim2 && au_r[idx].second > lowlim2)
            {
                occurred2 = true;
            }
        }
    }
}
else if (selection == 3)
{
    for (size_t idx = 0; idx < au_c.size(); idx++)
    {
        if (au_c[idx].first == "AU28")
        {
            if (au_c[idx].second == highlim)
            {
                occurred1 = true;
                occurred2 = true;
                std::cout << name << ": " << occurrence << "\n";
            }
        }
    }
}
else if (selection == 4 || selection == 5)
{
    double tilt = pose[5]/3.14*180; //to degrees
    if (tilt < highlim && tilt > lowlim)
    {
        occurred1 = true;
        occurred2 = true;
    }
}

//Gesture occurrence check
if (occurred1 && occurred2 && Ty > -15 && Ty < 15 && Tx > -15 && Tx < 15)
{
    occurrence++;
}

```

```

    }
    else {
        occurrence = 0;
    }
}

```

//Check if more than one command is occurring - if not then return its command
std::string Command::checkCommand()

```

{
    //Checks if lip suck is above 5, all other gestures are checked above 10
    if ((occurrence > 5) && (selection == 3))
    {
        send = true;
        return command;
    }
    else if (occurrence > 10)
    {
        send = true;
        return command;
    }
    else {
        send = false;
        return " ";
    }
}

```

//Send command to arduino

void Command::sendCommand(std::string c)

```

{
    std::string input_string = c;

    std::cout << "Sending " << name << "\n";
    ///Getting input
    //std::getline(std::cin, input_string);

    //Creating a c string
    char *c_string = new char[input_string.size() + 1];
    //copying the std::string to c string
    std::copy(input_string.begin(), input_string.end(), c_string);
    //Adding the delimiter
    c_string[input_string.size()] = '\n';
    //Writing string to arduino
    arduino.writeSerialPort(c_string, MAX_DATA_LENGTH);
    //Getting reply from arduino
    arduino.readSerialPort(output, MAX_DATA_LENGTH);
    //printing the output
    puts(output);
    //freeing c_string memory
    delete[] c_string;
}

```

int main(int argc, char **argv)

```

{
    //Creates constructors for each command (with exception of test - used for testing)
    Command forward;
    Command backward;
    Command left;
    Command right;
    Command halt;
    Command test;

    vector<string> arguments = get_arguments(argc, argv);

    // no arguments: output usage

```

```

if (arguments.size() == 1)
{
    cout << "For command line arguments see:" << endl;
    cout << " https://github.com/TadasBaltrusaitis/OpenFace/wiki/Command-line-arguments";
    return 0;
}

//check if arduino is connected
if (arduino.isConnected()) std::cout << "Connection Established" << endl;
else std::cout << "ERROR, check port name\n";

//User selection of gestures to use
char choice;
std::cout << "\nHello, are you happy to use the following gestures for these commands (Y/N)? \n\n"
<< "STOP: Open Mouth \n"
<< "FORWARD: Raise Eyebrows \n"
<< "BACKWARD: Wrinkle nose \n"
<< "LEFT: Tilt head left \n"
<< "RIGHT: Tilt head right"
//<< "TEST: Lip suck"
<< endl;

while (!(std::cin >> choice) || ((choice == 'y') || (choice == 'Y') || (choice == 'n') || (choice == 'N')))
{
    std::cin.clear();
    std::cin.ignore(256, '\n');
    std::cout << "ERROR: please input a Y or N \n";
}

if ((choice == 'y') || (choice == 'Y'))
{
    halt.Command::setGesture("halt", 2);
    forward.Command::setGesture("forward", 0);
    backward.Command::setGesture("backward", 1);
    left.Command::setGesture("left", 4);
    right.Command::setGesture("right", 5);
    test.Command::setGesture("test", 3);
}
else if ((choice == 'n') || (choice == 'N')) //Enables user to choose gestures
{
    int userInput[5] = { 15,15,15,15,15 };
    std::string commands[] = { "STOP", "FORWARD", "BACKWARD", "LEFT", "RIGHT" };
    int i = 0;
    std::cout << "These are your available gestures: \n"
<< "0. Raise Eyebrows\n"
<< "1. Wrinkle nose\n"
<< "2. Open mouth\n"
<< "3. Lip suck\n"
<< "4. Tilt head left\n"
<< "5. Tilt head right\n" << endl;

    while (i < 5) //Error checking for unacceptable inputs
    {
        boolean duplicate = false;
        std::cout << "What would you like to use for " << commands[i] << " ? \n";
        while (!(std::cin >> userInput[i]) || (userInput[i] < 0 || userInput[i] > 5))
        {
            std::cin.clear();
            std::cin.ignore(256, '\n');
            std::cout << "ERROR: you must enter a number between 0-4, please try
again. " << endl;
        }
        for (int k = 0; k < i; k++)
        {
            if (userInput[k] == userInput[i])
            {

```



```

        std::cout << "That gesture has been chosen, please try again. " <<
endl;

        duplicate = true;
        break;
    }
}
if (!duplicate)
{
    i++;
}
}
//Set gestures for commands according to user input
halt.Command::setGesture("halt", userInput[0]);
forward.Command::setGesture("forward", userInput[1]);
backward.Command::setGesture("backward", userInput[2]);
left.Command::setGesture("left", userInput[3]);
right.Command::setGesture("right", userInput[4]);
}
std::cout << "\nThese are the gestures chosen: \n"
<< "STOP: " << halt.name << "\n"
<< "FORWARD: " << forward.name << "\n"
<< "BACKWARD: " << backward.name << "\n"
<< "LEFT: " << left.name << "\n"
<< "RIGHT: " << right.name << endl;

// Load the modules that are being used for tracking and face analysis
// Load face landmark detector
LandmarkDetector::FaceModelParameters det_parameters(arguments);
// Always track gaze in feature extraction
LandmarkDetector::CLNF face_model(det_parameters.model_location);

if (!face_model.loaded_successfully)
{
    cout << "ERROR: Could not load the landmark detector" << endl;
    return 1;
}

// Load facial feature extractor and AU analyser
FaceAnalysis::FaceAnalyserParameters face_analysis_params(arguments);
FaceAnalysis::FaceAnalyser face_analyser(face_analysis_params);

if (!face_model.eye_model)
{
    cout << "WARNING: no eye model found" << endl;
}

if (face_analyser.GetAUClassNames().size() == 0 && face_analyser.GetAUClassNames().size() == 0)
{
    cout << "WARNING: no Action Unit models found" << endl;
}

Utilities::SequenceCapture sequence_reader;

// A utility for visualizing the results
Utilities::Visualizer visualizer(arguments);

// Tracking FPS for visualization
Utilities::FpsTracker fps_tracker;
fps_tracker.AddFrame();

while (true) // this is not a for loop as it might also be reading from a webcam
{

    // The sequence reader chooses what to open based on command line arguments provided
    if (!sequence_reader.Open(arguments))
        break;
}

```

```

INFO_STREAM("Device or file opened");

if (sequence_reader.IsWebcam())
{
    INFO_STREAM("WARNING: using a webcam in feature extraction, Action Unit
predictions will not be as accurate in real-time webcam mode");
    INFO_STREAM("WARNING: using a webcam in feature extraction, forcing
visualization of tracking to allow quitting the application (press q)");
    visualizer.vis_track = true;
}

cv::Mat captured_image;

Utilities::RecorderOpenFaceParameters recording_params(arguments, true,
sequence_reader.IsWebcam(),
sequence_reader.fx, sequence_reader.fy, sequence_reader.cx, sequence_reader.cy,
sequence_reader.fps);
if (!face_model.eye_model)
{
    recording_params.setOutputGaze(false);
}
Utilities::RecorderOpenFace open_face_rec(sequence_reader.name, recording_params,
arguments);

if (recording_params.outputGaze() && !face_model.eye_model)
    cout << "WARNING: no eye model defined, but outputting gaze" << endl;

captured_image = sequence_reader.GetNextFrame();

// For reporting progress
double reported_completion = 0;

INFO_STREAM("Starting tracking");
while (!captured_image.empty())
{
    // Converting to grayscale
    cv::Mat_<uchar> grayscale_image = sequence_reader.GetGrayFrame();

    // The actual facial landmark detection / tracking
    bool detection_success =
LandmarkDetector::DetectLandmarksInVideo(captured_image, face_model, det_parameters, grayscale_image);

    // Do face alignment
    cv::Mat sim_warped_img;
    cv::Mat_<double> hog_descriptor; int num_hog_rows = 0, num_hog_cols = 0;

    // Perform AU detection and HOG feature extraction, as this can be expensive only
compute it if needed by output or visualization
    if (recording_params.outputAlignedFaces() || recording_params.outputHOG() ||
recording_params.outputAUs() || visualizer.vis_align || visualizer.vis_hog || visualizer.vis_au)
    {
        face_analyser.AddNextFrame(captured_image,
face_model.detected_landmarks, face_model.detection_success, sequence_reader.time_stamp,
sequence_reader.IsWebcam());

        face_analyser.GetLatestAlignedFace(sim_warped_img);
        face_analyser.GetLatestHOG(hog_descriptor, num_hog_rows,
num_hog_cols);
    }

    // Work out the pose of the head from the tracked model
    cv::Vec6d pose_estimate = LandmarkDetector::GetPose(face_model,
sequence_reader.fx, sequence_reader.fy, sequence_reader.cx, sequence_reader.cy);

    //Detect gesture according to set parameters

```

```

        forward.Command::detectGesture(face_analyser.GetCurrentAUsReg(),
face_analyser.GetCurrentAUsClass(), pose_estimate);
        backward.Command::detectGesture(face_analyser.GetCurrentAUsReg(),
face_analyser.GetCurrentAUsClass(), pose_estimate);
        left.Command::detectGesture(face_analyser.GetCurrentAUsReg(),
face_analyser.GetCurrentAUsClass(), pose_estimate);
        right.Command::detectGesture(face_analyser.GetCurrentAUsReg(),
face_analyser.GetCurrentAUsClass(), pose_estimate);
        halt.Command::detectGesture(face_analyser.GetCurrentAUsReg(),
face_analyser.GetCurrentAUsClass(), pose_estimate);
        test.Command::detectGesture(face_analyser.GetCurrentAUsReg(),
face_analyser.GetCurrentAUsClass(), pose_estimate);

//check if any commands have occurred 10+ times
forward.Command::checkCommand();
backward.Command::checkCommand();
left.Command::checkCommand();
right.Command::checkCommand();
halt.Command::checkCommand();

//converts bool to int to count number of gestures occurring
int gesturesOccurring = (int)forward.send + (int)backward.send + (int)left.send +
(int)right.send + (int)halt.send;
//std::cout << " gestures: " << gesturesOccurring;

//if only one command is on at a time, then send command
if (gesturesOccurring == 1 )
{
    //send command based on input
    if (forward.send == true)
    {
        forward.Command::sendCommand("w");
    }
    else if (backward.send == true)
    {
        backward.Command::sendCommand("s");
    }
    else if (left.send == true)
    {
        left.Command::sendCommand("a");
    }
    else if (right.send == true)
    {
        right.Command::sendCommand("d");
    }
    else if (halt.send == true)
    {
        halt.Command::sendCommand("q");
    }
}

// Keeping track of FPS
fps_tracker.AddFrame();

// Displaying the tracking visualizations
visualizer.SetImage(captured_image, sequence_reader.fx, sequence_reader.fy,
sequence_reader.cx, sequence_reader.cy);
visualizer.SetObservationFaceAlign(sim_warped_img);
visualizer.SetObservationHOG(hog_descriptor, num_hog_rows, num_hog_cols);
visualizer.SetObservationLandmarks(face_model.detected_landmarks,
face_model.detection_certainty, face_model.GetVisibilities());
visualizer.SetObservationPose(pose_estimate, face_model.detection_certainty);
visualizer.SetObservationActionUnits(face_analyser.GetCurrentAUsReg(),
face_analyser.GetCurrentAUsClass());
visualizer.SetFps(fps_tracker.GetFPS());

```

```

        // detect key presses
        char character_press = visualizer.ShowObservation();

        // quit processing the current sequence (useful when in Webcam mode)
        if (character_press == 'q')
        {
            break;
        }

        // Reporting progress
        if (sequence_reader.GetProgress() >= reported_completion / 10.0)
        {
            cout << reported_completion * 10 << "% ";
            if (reported_completion == 10)
            {
                cout << endl;
            }
            reported_completion = reported_completion + 1;
        }

        // Grabbing the next frame in the sequence
        captured_image = sequence_reader.GetNextFrame();
    }

    INFO_STREAM("Closing output recorder");
    open_face_rec.Close();
    INFO_STREAM("Closing input reader");
    sequence_reader.Close();
    INFO_STREAM("Closed successfully");

    // Reset the models for the next video
    face_analyser.Reset();
    face_model.Reset();
}

return 0;
}

```

APPENDIX C – SERIAL CONNECTION FOR THE ARDUINO (MANDAL, 2016)

C.1 SERIAL.H

```
#pragma once

#ifndef SERIALPORT_H
#define SERIALPORT_H

#define ARDUINO_WAIT_TIME 2000
#define MAX_DATA_LENGTH 255

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

class SerialPort
{
private:
    HANDLE handler;
    bool connected;
    COMSTAT status;
    DWORD errors;
public:
    SerialPort(char *portName);
    ~SerialPort();

    int readSerialPort(char *buffer, unsigned int buf_size);
    bool writeSerialPort(char *buffer, unsigned int buf_size);
    //bool writeSerialPort(char c, unsigned int buf_size);
    bool isConnected();
};

#endif // SERIALPORT_H
```

C.2 SERIAL.CPP

```
#include "SerialPort.h"

SerialPort::SerialPort(char *portName)
{
    this->connected = false;

    this->handler = CreateFileA(static_cast<LPCSTR>(portName),
        GENERIC_READ | GENERIC_WRITE,
        0,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        NULL);
    if (this->handler == INVALID_HANDLE_VALUE) {
        if (GetLastError() == ERROR_FILE_NOT_FOUND) {
            printf("ERROR: Handle was not attached. Reason: %s not available\n",
portName);
        }
        else
        {
            printf("ERROR!!!");
        }
    }
    else {
```

```

DCB dcbSerialParameters = { 0 };

if (!GetCommState(this->handler, &dcbSerialParameters)) {
    printf("failed to get current serial parameters");
}
else {
    dcbSerialParameters.BaudRate = CBR_9600; //set baud rate
    dcbSerialParameters.ByteSize = 8; //8 data bits
    dcbSerialParameters.StopBits = ONESTOPBIT; //1 stop
    dcbSerialParameters.Parity = NOPARITY; //no parity
    dcbSerialParameters.fDtrControl = DTR_CONTROL_ENABLE;

    if (!SetCommState(handler, &dcbSerialParameters))
    {
        printf("ALERT: could not set Serial port parameters\n");
    }
    else {
        this->connected = true;
        PurgeComm(this->handler, PURGE_RXCLEAR | PURGE_TXCLEAR);
        Sleep(ARDUINO_WAIT_TIME);
    }
}
}

SerialPort::~SerialPort()
{
    if (this->connected) {
        this->connected = false;
        CloseHandle(this->handler);
    }
}

int SerialPort::readSerialPort(char *buffer, unsigned int buf_size)
{
    DWORD bytesRead;
    unsigned int toRead;

    ClearCommError(this->handler, &this->errors, &this->status);

    if (this->status.cbInQue > 0) {
        if (this->status.cbInQue > buf_size) {
            toRead = buf_size;
        }
        else toRead = this->status.cbInQue;
    }

    if (ReadFile(this->handler, buffer, toRead, &bytesRead, NULL)) return bytesRead;

    return 0;
}

bool SerialPort::writeSerialPort(char *buffer, unsigned int buf_size) //obtains a char string and returns true
when sent
{
    DWORD bytesSend; //number of bytes to send

    if (!WriteFile(this->handler, (void*)buffer, buf_size, &bytesSend, 0)) { //has to send string
        ClearCommError(this->handler, &this->errors, &this->status);
        return false;
    }
    else return true;
}

```

```
bool SerialPort::isConnected()
{
    return this->connected;
}
```

APPENDIX D – ARDUINO CODE TO CONTROL THE WHEELCHAIR (KUKREJA, 2018)

```
//Include libraries
#include<SPI.h>
//Default baud speed for communication
#define BAUD 9600
//Pins
#define CS_ADC 10
#define CS_POT 8
#define led 13
//macro for on/off
#define on (digitalWrite(led, HIGH))
#define off (digitalWrite(led, LOW))

byte channel0 = 0xE0;
byte channel1 = 0xE8;
byte POT_FB = 0x12; //00010001
byte POT_LR = 0x11; //00010010
byte POT_ALL = 0x13; //00010011
int jstickval;

void setup() {
    pinMode(CS_POT, OUTPUT);
    pinMode(CS_ADC, OUTPUT);
    pinMode(led, OUTPUT);

    digitalWrite(CS_ADC, LOW);
    digitalWrite(CS_ADC, HIGH);
    digitalWrite(CS_POT, HIGH);

    Serial.begin(BAUD);
    SPI.begin();

    SPI.setBitOrder(MSBFIRST);
    SPI.setDataMode(SPI_MODE3);
    SPI.setClockDivider(SPI_CLOCK_DIV64);

    POTcontrol(POT_ALL, 0x80);
    delay(2000);
}

void loop() {
    if (!check_joystick())
        face_keyboard_control();

    else
        joystick();
}

void POTcontrol(byte address, byte value) //send
{
    digitalWrite(CS_POT, LOW);
    SPI.transfer(address);
    SPI.transfer(value);
    digitalWrite(CS_POT, HIGH);
}

void face_keyboard_control()
{
    String input;
```



```

//If any input is detected in arduino
if (Serial.available() > 0) {
    //read the whole string until '\n' delimiter is read
    input = Serial.readStringUntil('\n');

    //If input == "ON" then turn on the led
    //and send a reply
    if (input.equals("ON")) {
        digitalWrite(led, HIGH);
        Serial.println("Led is on");
    }
    //If input == "OFF" then turn off the led
    //and send a reply
    else if (input.equals("OFF")) {
        digitalWrite(led, LOW);
        Serial.println("Led is off");
    }

    if (input == "w") //w Forward
    {
        POTcontrol(POT_FB, 0xFB);
        POTcontrol(POT_LR, 0x80);
        Serial.println("Forward");
    }

    else if (input == "a") //a Left
    {
        POTcontrol(POT_FB, 0x80);
        POTcontrol(POT_LR, 0x0A);
        Serial.println("Left");
    }

    else if (input == "s") //s Reverse
    {
        POTcontrol(POT_FB, 0x08);
        POTcontrol(POT_LR, 0x80);
        Serial.println("Reverse");
    }

    else if (input == "d") //d Right
    {
        POTcontrol(POT_FB, 0x08);
        POTcontrol(POT_LR, 0xFB);
        Serial.println("Right");
    }

    else if (input == "q") //stop q
    {
        POTcontrol(POT_FB, 0x80);
        POTcontrol(POT_LR, 0x80);
        Serial.println("Stop");
    }
}

}

int ADC_read(byte channel)
{
    int com;
    int volt;
    byte sbit;
    byte fbit;
    com = channel << 8 | 0x00;
}

```

```

        digitalWrite(CS_ADC, LOW);
        volt = SPI.transfer16(com) & 0x3FF;
        digitalWrite(CS_ADC, HIGH);
        return volt;
    }

    void joystick()
    {
        int turnjval;
        int fbjval;
        int turn;
        int drive;
        float mult = 0.85;

        turnjval = ADC_read(channel0);
        fbjval = ADC_read(channel1);
        turn = (turnjval - 100)*mult;
        drive = (fbjval - 100)*mult;
        POTcontrol(POT_FB, drive);
        POTcontrol(POT_LR, turn);
    }

    byte check_joystick()
    {
        int jval;

        jval = ADC_read(channel1);

        if (jval > 266 || jval < 240)
            return 1;
        else
            return 0;
    }

```

APPENDIX E – STANDARD TESTING PROTOCOL

A total of 9 tests will be conducted for a minute each. Gestures 2-5 will be performed twice to observe both short and continuous actions. You will be played a metronome at a certain tempo which will help to signify when you will perform the gestures. The first round will involve performing the action every 4 beats and the second will involve holding the action for 4 beats, relax for 4 beats and holding the action for 4 beats. These are the tests:

1. Neutral i.e. relaxed face
2. Raise Eyebrows every 4 beats
3. Wrinkle Nose every 4 beats
4. Open Mouth every 4 beats
5. Lip Suck every 4 beats
6. Repeat 2-5 and instead of performing the action every 4 beats - hold the action for 4 beats, relax for 4 beats and hold again for 4 beats, etc.

If you have any questions or require a demonstration of this procedure, feel free to ask the researcher.