AN INTRAORAL CAMERA FOR SUPPORTING ASSISTIVE DEVICES

by

Muhammad Amin Tily

A Thesis presented to the Faculty of the
American University of Sharjah
College of Engineering
In Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Mechatronics Engineering

Sharjah, United Arab Emirates

December 2018

Approval Signatures

We, the undersigned, approve the Master's Thesis of Muhammad Amin Tily

Thesis Title: An Intraoral Camera for Supporting Assistive Devices

Signature	**Date of Signature** (dd/mm/yyyy)
___________________________ Dr. Hasan Al-Nashash Professor, Department of Electrical Engineering Thesis Advisor	_______________
___________________________ Dr. Hasan Mir Professor, Department of Electrical Engineering Thesis Co-Advisor	_______________
___________________________ Dr. Abdulrahman Al-Ali Professor, Department of Computer Science and Engineering Thesis Committee Member	_______________
___________________________ Dr. Usman Tariq Assistant Professor, Department of Electrical Engineering Thesis Committee Member	_______________
___________________________ Dr. Lotfi Romdhane Director, Mechatronics Engineering Graduate Program	_______________
___________________________ Dr. Ghaleb Husseini Associate Dean for Graduate Affairs and Research College of Engineering	_______________
___________________________ Dr. Richard Schoephoerster Dean, College of Engineering	_______________
___________________________ Dr. Mohamed El-Tarhuni Vice Provost for Graduate Studies	_______________

Acknowledgement

All praise and thanks belong to Allah alone, The All-Knowing, The Most Wise. I start by thanking Allah who is the source of all of the uncountable blessings that surround us, and whose guidance I long for in each and every step I take on this earth.

I would like to thank my parents who have sacrificed a great deal for me and have supported me in all walks of life. I will always remain indebted to them for all that they have done for me.

I am also extremely grateful to both my sisters and brothers-in-law who keep supporting me in unique ways, and to all my nephews (Bilal, Muhammad and Zaid) and nieces (Zahra, Maryam and Juwairiyah) for spreading joy in my life.

I wish to express my sincere thanks to my advisers, Dr. Hasan Al-Nashash and Dr. Hasan Mir for guiding me throughout my coursework and making it possible for me to successfully complete my thesis. I would also like to thank my university for generously granting me Graduate Teaching Assistantships for 4 semesters which helped me conduct my research at AUS and also enriched me with the unique experience of teaching undergraduate students at AUS.

Last but not the least, a special thanks to all my colleagues in the Mechatronics lab who have made my stay at AUS memorable, especially Ali Qahtan, Daniyal Al-Waleed, Faris Al-khawaja, Ghaith Rousan, Hassan Umari, Hussein Ali, Koshish Koirala, Mirghani Daffalla, Mohamad Kassem, Muhammad Usman, Omer AbuBakr, Wasim Almasri and Zain Ali.

Dedication

Dedicated to my family…

Abstract

Thousands of patients around the globe are affected by paralysis which hinders the fulfilment of their basic needs such as mobility and speech. Several research topics have been dedicated to improve the livelihood of paralytic patients and a small subset of the topics has focused on capturing inputs from the tongue. The tongue is a muscular organ directly connected to the brain through a cranial nerve known as the hypoglossal nerve which is responsible for the motor functions of the tongue. Hence, tongue movements are not affected during spinal cord injuries, which is one of the major causes of paralysis. Given the importance of capturing inputs from the tongue, this research proposes a novel method of using an intraoral camera for this purpose. It discusses the methods used for capturing the images with the help of an Endoscope camera. It explains how the features were extracted in real-time using image processing techniques on each captured frame and how the orientation and position of the tongue was then accurately classified into one of the 11 possible categories to produce specific outputs which could be used by paralytic patients as inputs to any external system. After testing the system with a data entry application, an average of 19.34 correct entries per minute was calculated from 5 different experiments, and an average error rate of 3.96% was obtained, which outperforms systems such as the Resistopalatography and the MouthPad in terms of accuracy.

Keywords: ***Assistive Technology; disability; paralysis; tongue; intraoral camera; image processing.***

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AT	Assistive technology
BCI	Brain-Computer Interface
EEG	Electroencephalogram
EMG	Electromyography
EOG	Electrooculography
HMIs	Human Machine Interfaces
HSV	Hue, Saturation, Value
SBC	Single-Board Computer

Chapter 1. Introduction

1.1. Overview

1.1.1. Motivation

The total number of adults in the United States facing at least some form of physical disability adds up to 39.6 million [1], and one of the major causes of concern is the spread of paralysis amongst the masses. Paralysis is a loss of muscle function which restricts voluntary muscle movements. It is approximated that around 5.4 million people across the United States are living with paralysis [2]. According to the same study, it was revealed that the primary cause of paralysis is stroke (33.7%), followed by spinal cord injuries (27.3%) and then multiple sclerosis (18.6 %). Several other causes of paralysis exist such as the Parkinson's disease, cerebral palsy and Amyotrophic lateral sclerosis (ALS), etc. The ALS is a disease that gradually leads to complete paralysis, and it is estimated that around 15 people each day are diagnosed with ALS, and around 30,000 Americans are currently suffering from this disease [3].

Paralysis may affect one’s face, hands, one arm or leg (monoplegia), one side of the body (hemiplegia), both legs (paraplegia), or both arms and legs (tetraplegia or quadriplegia) [4]. Moreover, even the severity of paralysis may differ ranging from partial to complete paralysis which would cause the patient to be bed-ridden. Thus, patients’ needs vary from the type of paralysis they are affected with, which may restrict their mobility, speech or even their ability to grasp objects. These restrictions make it challenging for the patients to fulfill their very basic needs.

1.1.2. Assistive technologies

Latest advancements which aim to assist patients affected by paralysis either involve the activation of paralyzed muscles or the restoration of their voluntary movements; the latter being the subject of this research which comes under the section of assistive technologies. Assistive technology (AT) can be defined as “any item, piece of equipment, software program, or product system that is used to increase, maintain, or improve the functional capabilities of persons with disabilities” [5]. Several solutions in AT have been presented which aim to empower the users by providing alternative methods of capturing their inputs to a system. The implications of this is huge for people of special needs, as their education, nutrition, communication and mobility may otherwise all be completely dependent on those around them. ATs assist by creating more independency for the users. Thus, in order to improve the well-being of physically challenged people, it is vital to improve the ATs in order to cater to their specific needs.

The current available solutions in AT for paralytic patients include the tracking of limb movements (such as eye, head or tongue movements, etc.), obtaining Electroencephalogram (EEG), Electrooculography (EOG) and Electromyography (EMG) signals, capturing inhalation and exhalation of the user, etc. Each solution has its pros and cons and the most suitable solution varies from user to user, based on the type and degree of paralysis or disability.

1.1.3. Utilizing the tongue

One of the proposed solutions involves using the tongue as an input to a system. There are several advantages with this approach, as the tongue's movement does not cause exhaustion for the user, it is capable of making complex movements while remaining hidden from others, and its movement capabilities remain independent form the position of the body (i.e. lying, sitting down or standing up does not affect tongue movements) [6].

1.1.4. Improving the existing tongue-driven systems

After an extensive literature review dealing with tongue driven systems, it was noted that the present solutions had a lot of room for improvement. They can be improved by reducing the area occupied by the system, making the system less invasive (by avoiding the piercing of any piece through the tongue) and completely hidden within the intraoral environment. Moreover, most of the solutions utilized contact sensors, which are not very comfortable for the users and thus may not be practical for long term usage. Therefore, in order to enhance the existing solutions by avoiding the usage of contact sensors, the use of a camera was proposed in this research. Furthermore, with the trend of cameras being available in smaller and smaller sizes, it is expected that the intraoral camera would also be very conveniently hidden completely inside the oral cavity, along with fully functional Wi-Fi capabilities to transmit data, in the near future.

1.2. Thesis Objectives

This thesis aims to introduce an innovative alternative in assistive technologies for aiding the people affected by disabilities to fulfil their basic needs. The thesis was broken down into the following main parts:

1.2.1. Sensor technology

In this part, an extensive study was carried out of different technologies aiding people affected by paralysis to fulfill their basic needs, such as empowering them to commute freely, grasp objects, etc. We propose a novel method of utilizing an intraoral camera for using the tongue as an input to any external system. In order to form a working prototype, the most suitable camera for the intraoral environment also had to be selected.

1.2.2. Making a prototype

Once a camera was selected, a wearable prototype had to be set up which would hold the camera in place, ensure a proper lighting system in the intraoral region and would be comfortable for the patients to use. This would also ensure that the results are replicable since the camera and the lighting system would remain the same during the entire testing phase.

1.2.3. Image processing

Real time image processing was the key to form a working prototype. Each frame captured from the camera had to be processed in order to find the exact location and orientation of the tongue which would be used as an input to control an external system.

1.2.4. Connecting an application

Once the information of the location of the tongue would be received, it was also connected to a system in order to test its performance. The application chosen for this system was a robotic arm which is of particular benefit to paralytic patients as it is able to fulfil similar functions as that of the human arm.

1.3. Research Contribution

The implementation of an intraoral camera would improve the current ATs focusing on the tongue by:

- Preventing the system from being intrusive.

- Making the system more comfortable for the users as the tongue would not require any contact to a sensor.
- Allowing the system to be completely hidden in the intraoral environment with the latest developments of small-sized cameras.

1.4. Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 provides a literature review on key research papers and existing solutions with regards to the different sensors assisting paralytic patients to fulfil particular tasks. Chapter 3 lays down the problem statement and discusses the proposed method of introducing an intraoral camera along with the image processing techniques used. The image processing results and the identification of the exact location of the tongue is presented in Chapter 4. Finally, Chapter 5 concludes the thesis and outlines the future work.

Chapter 2. Background and Literature Review

In order to restore mobility and independence for patients suffering from paralysis, a number of human machine interfaces (HMIs) have been researched and developed, the most common of which are listed below.

2.1. Eye Movements

Eye-blinks are used as inputs for a variety of applications such as detecting the level of drowsiness of drivers [7] and giving simple yes/no responses. A lot of methods have been devised to capture the eye-blinks such as using infrared sensors, image processing [7], electrooculography (EOG) signals [8] and Doppler sensors [9].

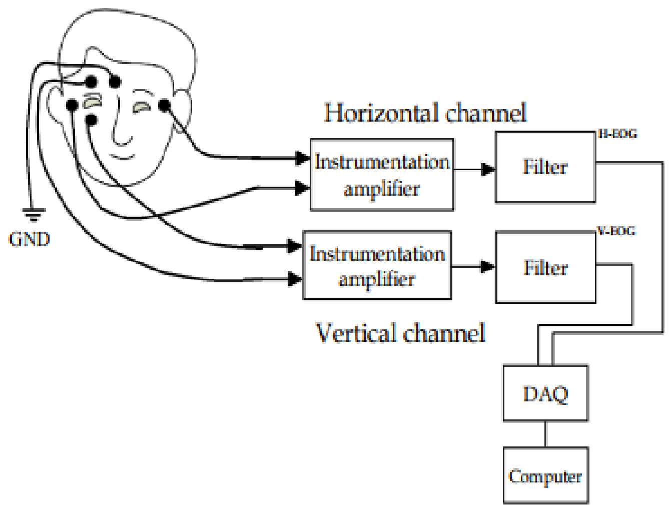


Figure 2.1: An example illustrating the position of electrodes for acquiring EOG measurements [8].

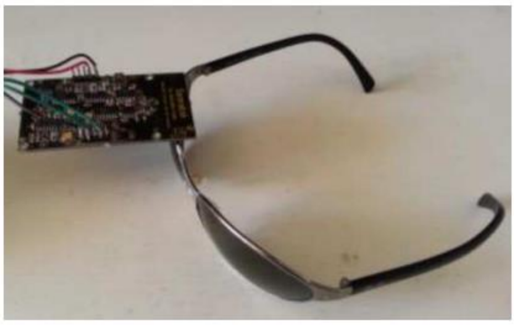


Figure 2.2: A setup using a Doppler sensor to detect eye-blinks [9].

A lot more flexibility is added to the inputs by taking into account the direction of the eye gaze. This may be achieved by using imaging techniques ranging from simple cameras [10] or even utilizing infrared imaging techniques.

Eye-blinks and the direction of eye gaze may also be estimated using Electromyography (EMG) and Electrooculography (EOG) signals [11]. EOG involves the placement of electrodes near the eye (e.g. above and below the eye as shown in figure 1) and the potential difference of the signals are able to map out the direction of the eye movements. EMG signals are capable of recording muscle activities which can be used to detect eye-blinks. Depending on the seriousness of paralysis of the patient, EMG signals can be used to detect a number of different movements, as described in the next section.

One of the disadvantages of using eye-blinking or eye tracking methods, such as those using remote cameras, is that it restricts the natural eye or head movements thus restricting the user's field of vision, and it also demands high levels of concentration. Equipment such as head-mounted cameras allow for greater head movements but are more obtrusive.

2.2. Muscle Activities

EMG signals are able to record muscle activities, and thus form an additional source of input for paralytic patients. Apart from eye-blinks, they are also able to detect a number of different movements such as eyebrow raises [12], and neck, shoulder and cheek movements [13]. Infrared switches have also been used to detect cheek movements [14] which work well for non-contact sensing applications.

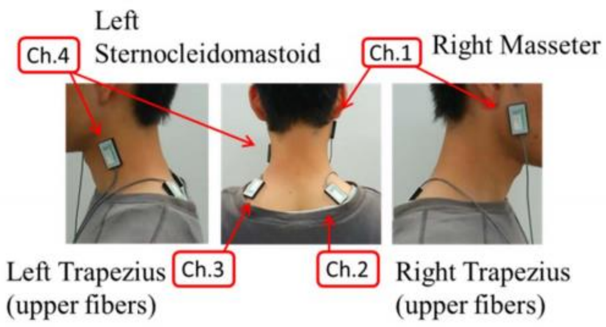


Figure 2.3: A setup using EMG measurements to detect neck, shoulder and cheek movements [13].

The clenching of teeth has also been used as an input by detecting muscle movements. Apart from detecting the movements via EMG signals [15], the contact of the maxillary and mandibular teeth has also been detected with the help of an accelerometer placed against the user's ear [16]. The sensor was used to identify the vibrations resulting from the tooth-clicks. However, the drawback of using such devices is that hardware adjustments of such equipment are challenging, if not impossible, without external assistance for people suffering from paralysis.

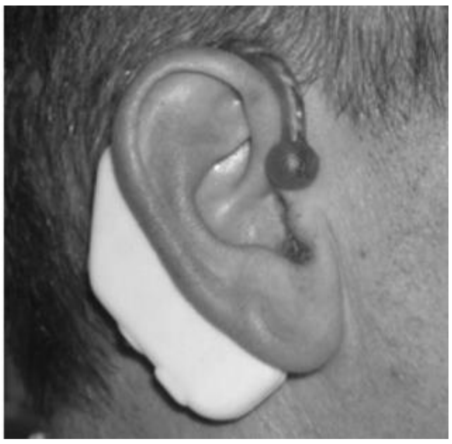


Figure 2.4: A Tooth-click detector device worn behind the ear.

The drawback of using EOG and EMG signals as inputs is the inconvenience of using electrodes which cannot give accurate results if the contact with the skin is loose. Disposable electrodes even come with adhesives but cause discomfort for the users. Moreover, the long wires associated with the electrodes make it less appealing for the users.

2.3. Inhalation and Exhalation

The sip-and-puff technology has been popular amongst paralytic patients as it is an easy to use technology. It includes pressure sensors to gauge the air pressure and thus distinguish between the inhalation and exhalation of the air by the user. The device also takes into account the intensity of the sips or puffs (hard or soft) and therefore typically allows for four inputs by the users. Of the disadvantages is that regular cleaning and sterilization of the tubes is necessary, long usage of the device may be tiring for users and the available inputs are limited. However, recently researches have sought to incorporate more inputs by using pattern recognition software to study breath patterns of the users [17].

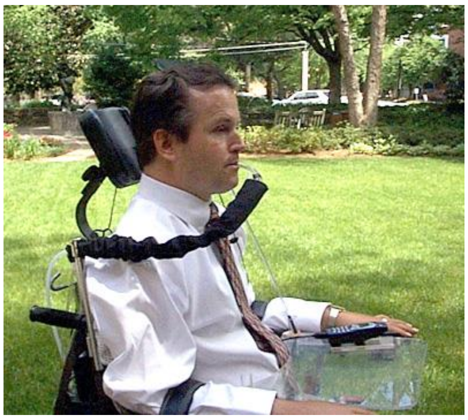


Figure 2.5: Sip-and-Puff Assistive Technology for controlling a wheelchair [18].

2.4. Electroencephalogram (EEG) Signals

Brain-controlled techniques have attracted a lot of research which give an opportunity to communicate via thoughts and are extremely valuable especially for people affected by complete paralysis. Acquiring and processing EEG signals has been the most commonly used approach in brain-computer interfaces (BCIs), but EEG controlled devices typically tend to have high uncertainty in the commands provided by the user and is very time consuming for the users as well [19].

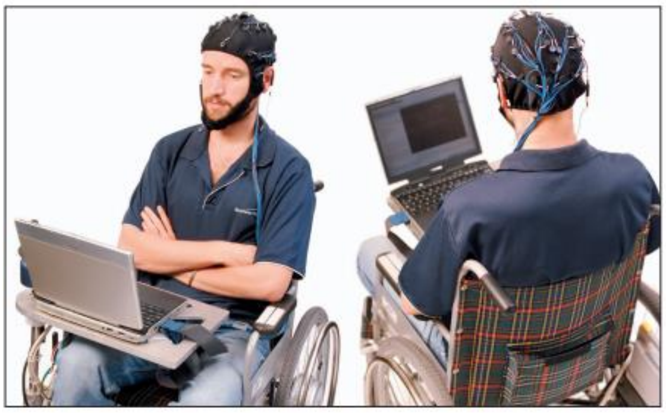


Figure 2.6: A brain-controlled wheelchair acquiring EEG signals via an EEG cap [19].

The P300 evoked potentials have been shown to possess the capability of implementing a successful brain-computer interface (BCI) and have also been used to form entire sentences by the users character by character [20]. The method of operation of brain-to-text systems includes a screen which outputs a matrix of characters. While the user focuses on his/her character of interest, one row or column of the matrix is randomly highlighted at a time (until all of the rows and columns have been highlighted once) which forms the oddball paradigm in the experiment. Every time the character of interest in highlighted and spotted by the user, the transient activity in recorded via EEG signals and the target character is thus eventually outputted via the interface.

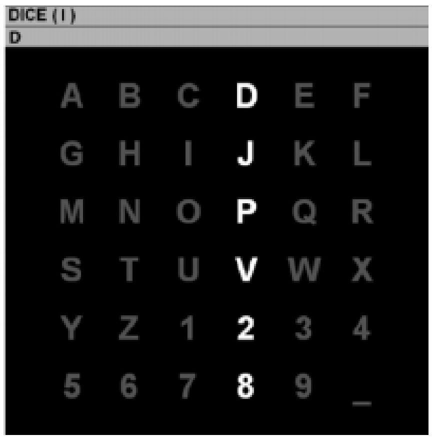


Figure 2.7: A 6 × 6 matrix for the P300 Event-Related Potential. The rows and columns are highlighted at regular intervals [20].

However, as discussed earlier, the use of electrodes (which are typically attached to a cap for EEG recordings) make this option less popular amongst the users. The electrodes require careful connections with the patient's scalp and in some cases (if special dry electrodes are not used) it may even require the use of electro-gels which would increase discomfort for the patients. In addition, the data rate of such systems is relatively low allowing for only few words per minute.

2.5. Tongue Movements

One of the most efficient ways for paralytic patients to use assistive devices is through tongue movements. The tongue is a muscular organ which is directly connected to the brain through a cranial nerve known as the hypoglossal nerve which is responsible for the motor function of the tongue. Hence, tongue movements are not affected during spinal cord injuries, which is one of the major causes of paralysis in people. Moreover, the input device can remain hidden and tongue movements for an extended time will not be burdensome for patients.

Several methods have been researched and implemented with regards to using tongue movements as inputs to control assistive devices. The following list provides a brief overview of the main contributions in this field.

2.5.1. Resistopalatography

The Resistopalatography sensor [21] works by detecting the change in pressure (exerted by the tongue) which alters the resistance. Thus, a change in resistance indicates the tongue movements which form the desired input. In order to avoid interference from the moisture of the mouth, the sensor was also laminated, and a complete 360 degrees detection with a resolution of 45 degrees was made possible.

The data from the sensor is sent to a microcontroller after passing through an analog to digital converter, and x and y values are read as outputs. The sensor in the center is used for mouse clicks so it provides an output if the input goes beyond a predefined threshold value.

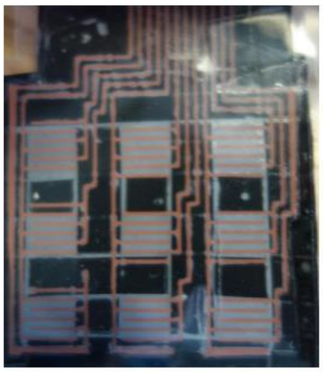


Figure 2.8: Construction of a Resistopalatography Sensor [21].

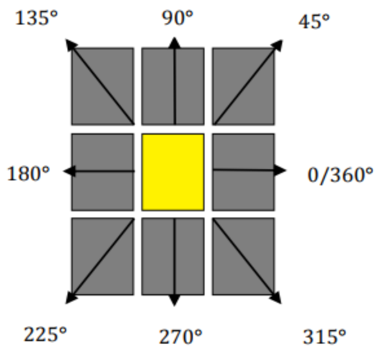


Figure 2.9: Electrodes in grey are used for moving the cursor, while the yellow electrode enables the mouse click [21].

2.5.2. The MouthPad

The MouthPad [22] is a tongue-computer interface placed on the palate that measures the contact impedance between an electrode array and the tip of the tongue and thus identifies the tongue position and its movements. The electrode array receives an AC signal and the ground is attached to the fingers or the wrist. Thus, the circuit is closed only when the tongue touches the electrode array.

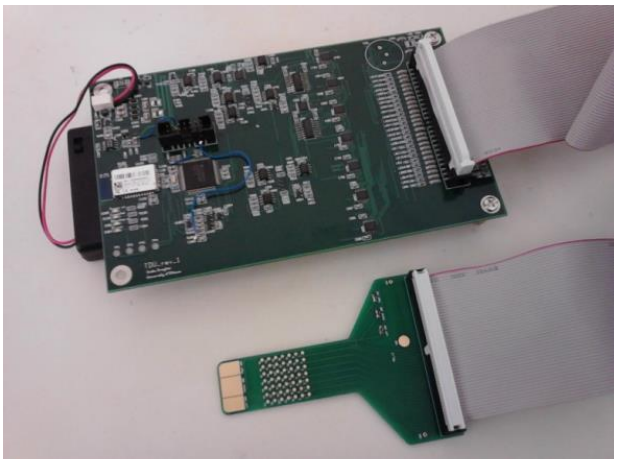


Figure 2.10: The MouthPad with 49 (7×7) gold plated electrodes connected to a Microcontroller [22].

The MouthPad was tested for the application of controlling a mouse on the screen using only 8 pairs of electrodes (thus using 16 of the 49 available electrodes). The direction of movement of the cursor was chosen according to the pair of electrodes that had been touched by the tongue (which was detected by the impedance). If more than one pair of electrodes were activated, then the average of the directions was taken as the output, and if several electrodes were activated at once, this was detected as an involuntary tongue movement (such as swallowing) and thus no movement was outputted.

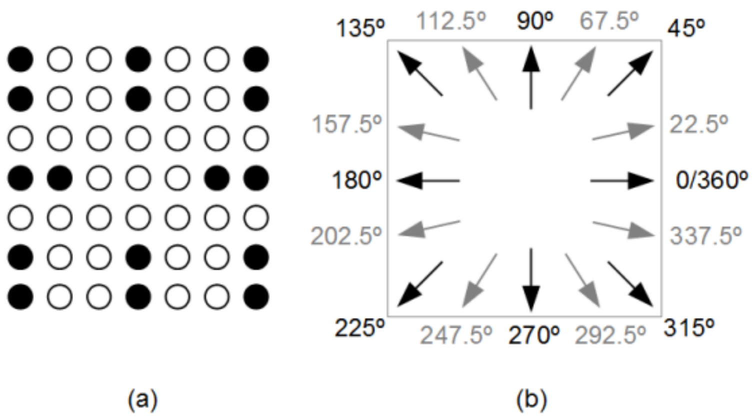


Figure 2.11: a) The electrodes used in the joystick mode used 8 pairs of electrodes, b) Corresponding directions of the cursor movements [22].

While testing the MouthPad on subjects, one of the causes of unwanted fluctuations in the cursor movements that were reported by users was due to the saliva interfering with the contact impedance of the electrode array.

2.5.3. Magnetic sensors

The tongue-drive system [23] detects tongue movements using hall-effect magnetic sensors placed on a dental retainer. The sensors detect the magnetic field produced by a permanent magnet which was placed in the middle of the tongue (for long term use the magnet is pierced).

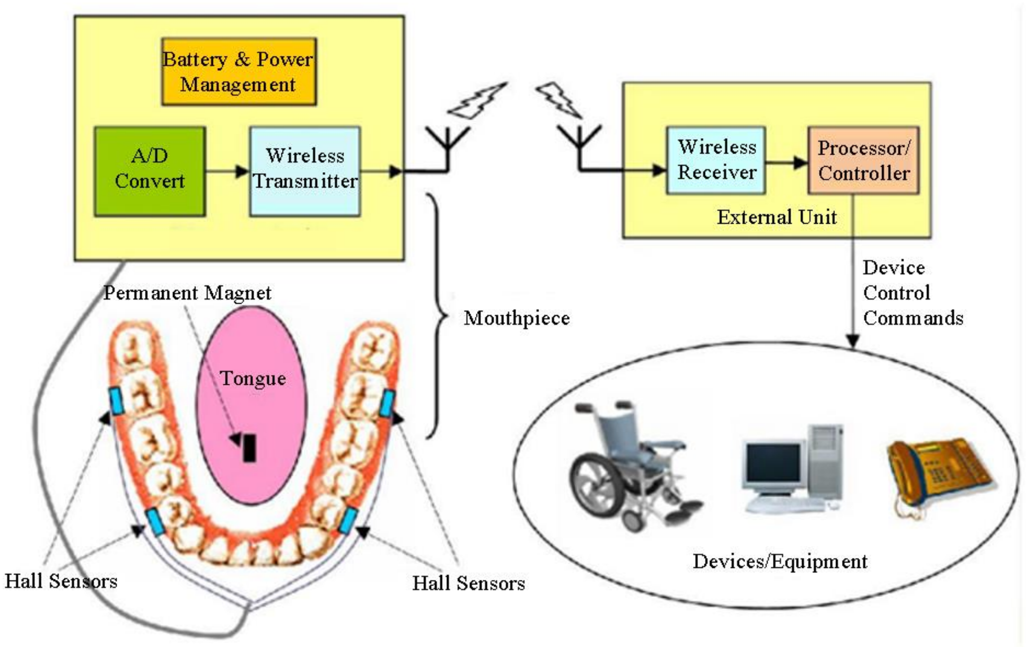


Figure 2.12: Block diagram of the tongue-drive system [23].

The mouthpiece is powered by small batteries and the sensors are scanned one at a time to read analog values which are digitized and wirelessly sent to a controller unit where the data is processed and the motion of the tongue is estimated.

The downside of using this method is the invasive nature of the device as the permanent magnet has to be fixed by piercing the tongue, which can even cause discomfort while performing regular activities such as eating and speaking.

There are several research papers that deal with the tongue drive system which has been used for several different applications such as controlling wheelchairs [24], [25] and performing computer related tasks [26], [27].

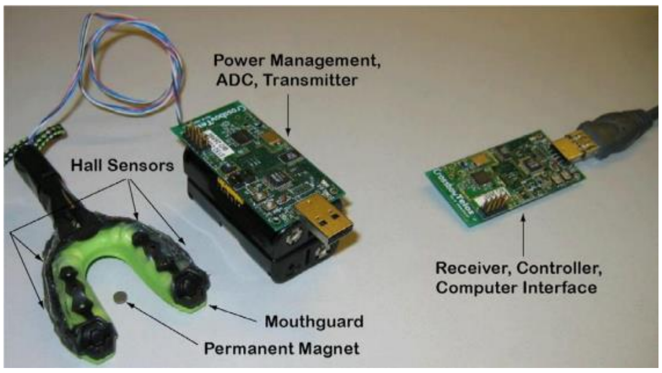


Figure 2.13: A prototype of the tongue-drive system [23].

2.5.4. Tongue-operated joystick

A prototype of the tongue-operated joystick [28] using a slider crank mechanism is demonstrated in the figure below. However, such devices are obtrusive and may not be suitable for long term usage as natural movements of the tongue, such as during swallowing, are prevented with the holding part in the mouth.

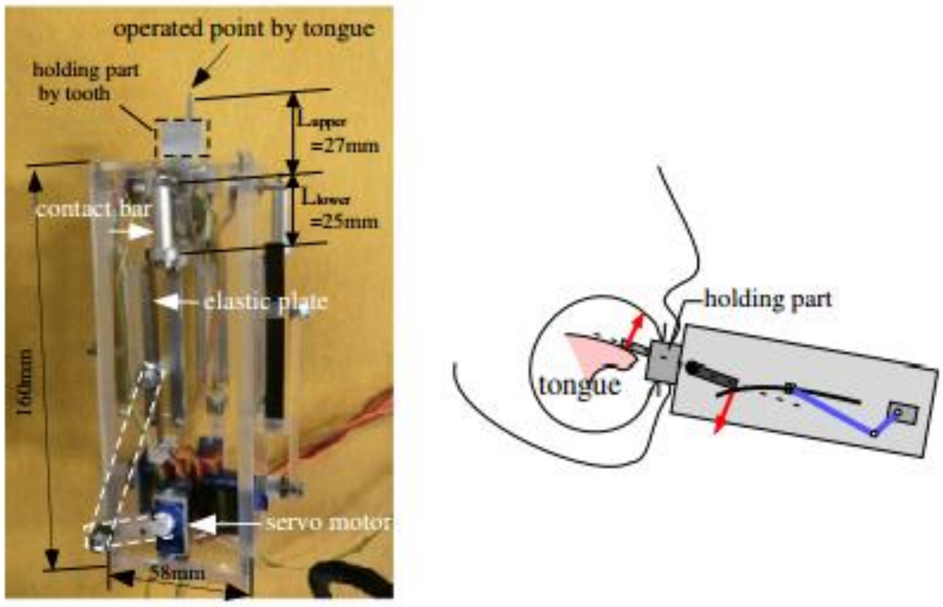


Figure 2.14: A prototype of the tongue-operated joystick.

Chapter 3. Methodology

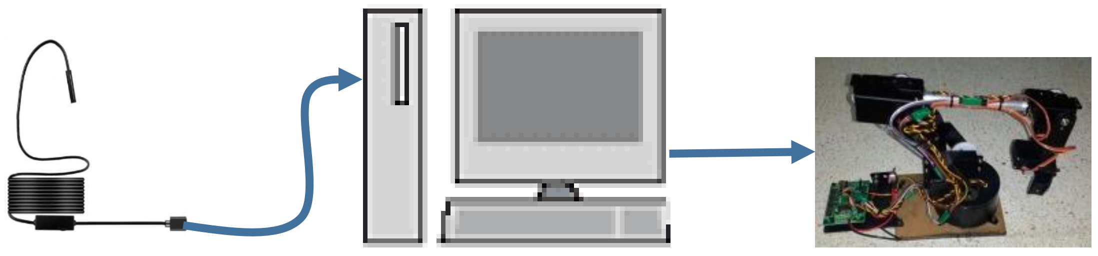
3.1. Problem Formulation

As discussed in the literature review, utilizing the tongue as an input has been an efficient way for people of special needs to accomplish their basic needs. However, after studying the existing solutions, the following problems are addressed in this research:

- Capture inputs from the tongue without the use of contact sensors which would cause discomfort for the patients.
- Use non-invasive methods to make it convenient for the users.
- Use a system which occupies less space within the oral cavity, and is also less obtrusive.

3.2. Proposed Solution

In this research, we propose a novel technique of using an intraoral camera to detect tongue movements. Using a camera would ensure that the tongue would not have to continuously establish contact with any object (thus making it comfortable for the users). Furthermore, the system would not require any piercing of the tongue, and with the latest advancements in technology, mini-cameras are also growing in popularity which would ensure that minimal space is consumed within the oral cavity. Thus, a novel solution of using an intraoral camera is proposed. Figure 3.1 shows the overview of the complete system.



1. An endoscope camera (fit inside the user's mouth) takes images of the tongue which are sent to the PC in real-time.

2. The PC processes each frame received from the camera and evaluates the position and orientation of the tongue.

3. According to the location of the tongue, a control signal is sent to a device (the robotic arm in this case) allowing the user to control it with his/her tongue.

Figure 3.1: An overview of the system.

Each of the steps presented in Figure 3.1 are explained in greater details in the subsequent sections.

3.3. Capturing the Images

Initially the Raspberry Pi's NOIR camera was selected which has the infrared (IR) cut filter removed. The IR cut filters allow visible light to pass but block mid-infrared wavelengths (3–8µm). Thus, the filter's removal means that the camera would be suitable for capturing images in the dark which would be crucial for the intraoral environment. However, the camera was not found to be suitable for capturing close-range images which rendered this camera unfeasible for the prototype.

Figure 3.2: Raspberry Pi's NOIR camera along with the board.

The AN97 endoscope camera was considered to be the most feasible option. The resolution of this low-cost camera is 640×480 and being waterproof makes it appropriate for the intraoral environment. The diameter of the camera head is only 7mm and the length of the camera head is 0.043m and thus it can easily fit on the roof of one's mouth. A mirror angled at 30° was added at the camera head in order to reflect the tongue's image on the camera lens as shown in Figure 3.4.

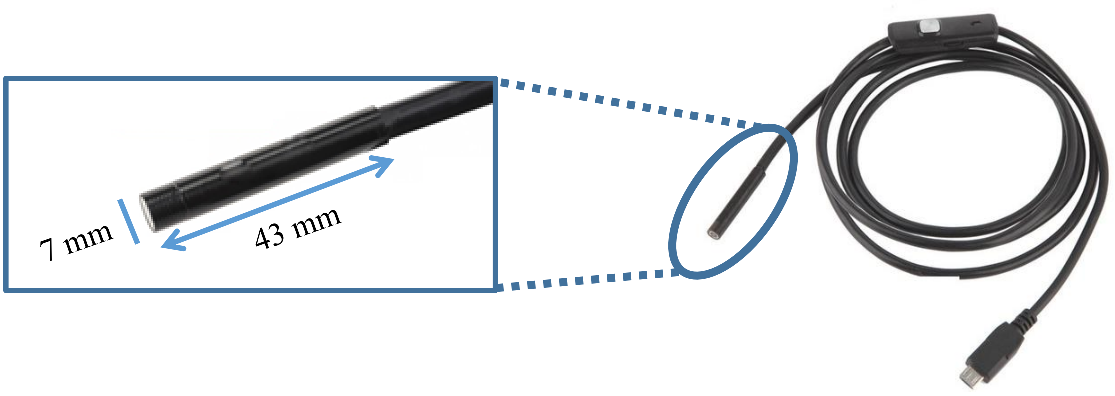


Figure 3.3: The AN97 Endoscope camera along with its dimensions.

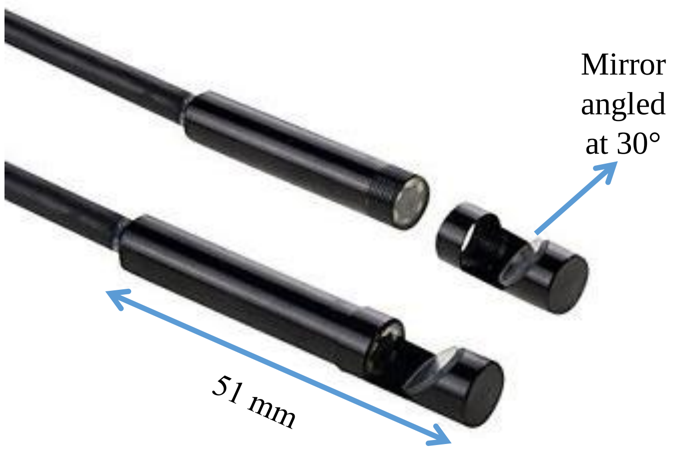


Figure 3.4: The AN97 Endoscope camera along with a 30° mirror added at the camera head.

Although the AN97 camera had 6 built-in white LEDs, their usage was restricted with the addition of the mirror, since the reflection of the light from the mirror considerably lowered the image quality. Thus, in order to lighten up the intraoral environment, 2 white LilyPad LEDs were used whose dimensions are 5.5mm×12.5mm as shown in Figure 3.5. The schematic representation is depicted in Figure 3.6 which shows a 150 Ω resistor connected in series with the LED.

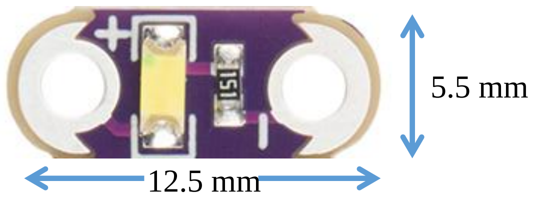


Figure 3.5: A White LilyPad LED with its dimensions.

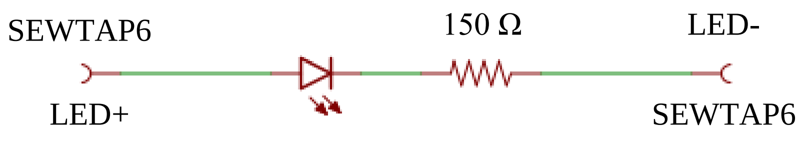


Figure 3.6: Schematic representation of the LilyPad LED.

3.4. Image Processing

In order to locate the position and orientation of the tongue in real-time, a number of different image processing techniques are implemented on each frame captured from the camera. The code is written using the Python programming language which also allows us to make use of the Open Source Computer Vision Library (OpenCV) which simplifies the implementation of numerous image processing techniques. The techniques implemented to extract information about the location and orientation of the tongue are explained below.

3.4.1. Smoothing the image

The colored image (stored in the BGR format in python) is first blurred in order to reduce noise. This is achieved by adding a Gaussian filter to the image, which means that a Gaussian kernel (array of pixels corresponding to a 2D Gaussian curve) is convolved with each image pixel and the summation of which produces the output image. The default values of sigmaX and sigmaY are used for forming a 5×5 Gaussian kernel. A 2D Gaussian function is given by the following formula:

$$\mathrm{G}(\mathrm{x},\mathrm{y}) = \mathrm{Ae}^{\mathrm{z}} \tag{3.1}$$

$$\text{Where } z = \frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2} \tag{3.2}$$

Where A=coefficient of the amplitude, μ = mean, σ_x and σ_y = standard deviation of x and y respectively.

3.4.2. Conversion to HSV (Hue, Saturation, Value)

The colored image is converted to HSV (Hue, Saturation, Value) format in which the points from the RGB model are represented in the form of a cylindrical coordinate system as shown in Figure 3.7. The 'hue' includes the spectrum of colors, 'saturation' gives an indication of the number of white pixels mixed with the color, and 'value' defines the lightness of the color. This is extremely beneficial in order to: (1) separate the color of the tongue from the set of teeth using the hue, and (2) to use the 'saturation' and 'value' in order to separate the tongue from the floor of the mouth which is approximately the same color as the tongue. In order to use the 'saturation' and 'value' effectively, the lighting within the mouth has to be manually adjusted in such a way that the brightness of the tongue is much greater than the brightness of the floor of the mouth. This is a critical step which would ensure a proper isolation of the tongue from its surroundings.

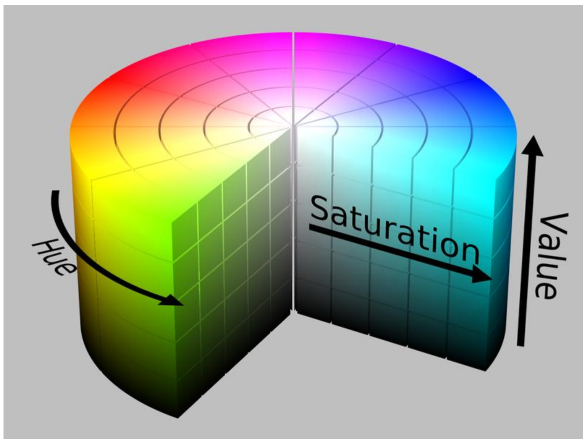


Figure 3.7: The HSV color model.

3.4.3. Binarization of the image

Once the tongue is isolated from its surroundings, the image is then converted to a grayscale image which is denoised using a Gaussian blur (similar to the procedure described in section 3.4.1). The grayscale image is then converted to a binary image using Otsu's binarization technique [29] which avoids the manual selection of the threshold based on trial and error. Using the histogram of the image, the Otsu's binarization technique automatically calculates the threshold. This is done by minimizing the weighted within-class variance (denoted by the following formula 3.3) and finding the corresponding threshold value 't' [29].

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{3.3}$$

Where t= threshold, $q_1(t) = \sum_{i=1}^{t} P(i)$ and $q_2(t) = \sum_{i=t+1}^{I} P(i)$ (3.4)

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \tag{3.5}$$

$$\mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)} \tag{3.6}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \tag{3.7}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)} \tag{3.8}$$

Thus this method assumes that two classes are present within the image and it then computes the optimal threshold that distinguishes between the two classes. From 3.3 and 3.4, q_1 is the probability of the first class while q_2 is the probability of the second class. Formula 3.4 shows how the probabilities are computed from the histogram which is done for each intensity value. From each possible threshold (from t=0 to t=255 for our case since we are implementing the binarization on a grayscale image), the probability and mean is computed from which we obtain the inter-class variance which needs to be maximized. The inter-class variance ($\sigma_b^2(t)$) is derived by subtracting the within-class variance from the total variance of the image, as shown in formula 3.9. The threshold (t) corresponding to the maximum inter-class variance is the selected threshold value that will be used to binarize the grayscale image [29].

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = q_1(\mu_1 - \mu_T)^2 + q_2(\mu_2 - \mu_T)^2 \tag{3.9}$$

3.4.4. Applying a morphological transformation on the binary image

After obtaining the binarized image, a morphological transformation known as 'closing' is applied to the image. This is to close tiny holes on the image and remove any unnecessary black points on the tongue (represented by the white pixels). The 'closing' morphological transformation is simply carried out by first applying the dilation and then the erosion morphological transformations [30]. It can be represented by the formula shown in 3.10.

$$A \cdot B = (A \oplus B) \ominus B \tag{3.10}$$

Where $\oplus$ represents dilation and $\ominus$ represents erosion.

3.4.5. Obtaining the contour of the image

Once the binary image has been obtained, the next step is to obtain the contour of the image which would basically highlight the boundaries of the object. There are different contour approximation methods to avoid storing all of the points in the contour. In order to compress the contour and remove the redundant points (and thus save memory), the 'CHAIN_APPROX_SIMPLE' option from OpenCV is used. This option compresses the vertical, horizontal and diagonal segments and only utilizes their end points to form the contour. Thus, instead of using hundreds of points to form a contour of an upright rectangle, this option would only store 4 points for the contour and save a lot of processing time and memory. Once the contour is obtained, the features of the contour can easily be obtained by using the inbuilt OpenCV commands. In particular, two features are extracted:

- **Area of the Contour:** In case the noise is not completely removed from the image, after obtaining all of the contours, an 'if' statement is added such that the unwanted contours can simply be removed based on the area of the contour.
- **Centroid:** The centroid of the contour gives valuable information about the location of the object. The x and y values of the centroid change based on the horizontal and vertical movements of the tongue respectively.

3.4.6. Segmentation of the image

The image is then segmented into 12 different rectangles, each of which would individually gauge the presence of the tongue within its region of interest. This is done by determining the number of white pixels in each region. If it is above a particular threshold, it would indicate the presence of the tongue in that particular region of interest. Utilizing this information along with the centroid of the contour (from 3.4.5), the orientation and location of the tongue could finally be accurately evaluated.

3.4.7. Detecting the tongue location and forming the categories

The information obtained regarding the centroid of the contour along with the presence of the tongue in the pre-defined segments is used in order to form definite outputs (which would be inputs to an assistive device). The image is segmented into 12 regions as shown in Figure 3.8. The role of the top 3 segments is to detect whether the tongue was present or not (i.e. whether the tongue has been pulled back by the user). This is then used as an indication to begin looking for contours when the tongue is in the picture. The reason this condition is set is due to the fact that when the tongue is completely pulled back by the user, certain contours are detected in the middle of the image as background noise, and implementing a check into the presence of the tongue in the top 3 segments would eliminate that.

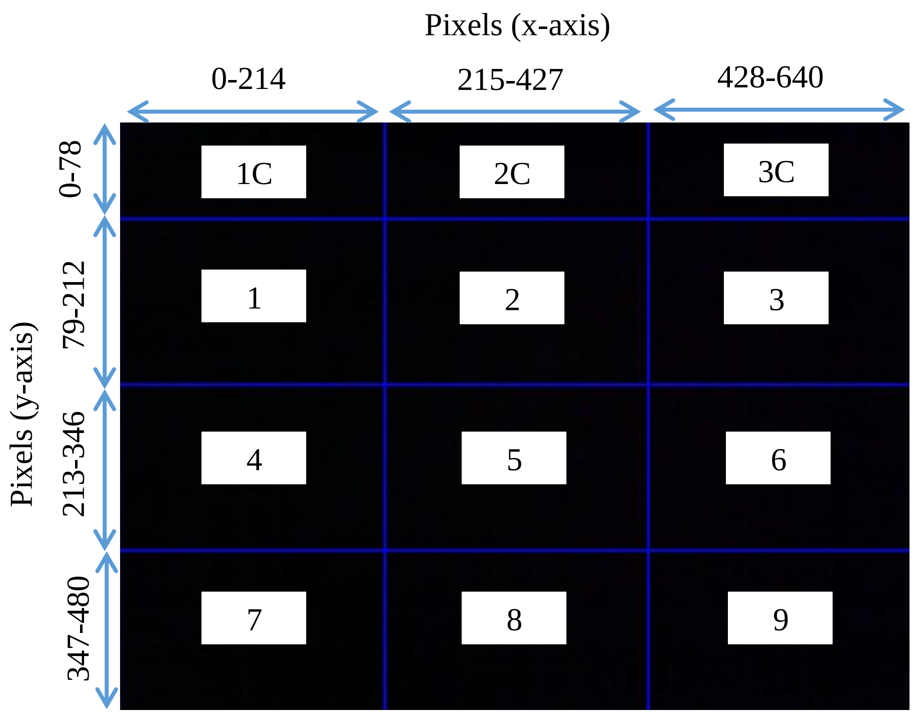


Figure 3.8: Segmentation of the 640×480 image into 12 regions.

Once the contour is formed, the centroid of the contour is first analyzed, which contains the C_x and C_y values (which is the value of the centroid on the x and y axis respectively). The values of C_x are gauged when the tongue is in the right, middle and left position, and two threshold values for C_x had to be selected which would help to give an indication which region the tongue is in (right, middle or center) and thus narrow down the 'if' statements in the code. From trial and error it was found that the threshold values would change slightly according to the positioning of the LEDs (especially when packing and unpacking the mouthpiece). Thus, a calibration system was set up such that when the code runs, it first asks the user to position his/her tongue in the middle during which the C_x value is recorded. The same is done after prompting the user to position his/her tongue to the right and the left. Once the tongue has been categorized into these 3 positions (right, middle or center), the numbers of white pixels in the regions of interest are then evaluated in order to determine the sub-categories. The categories are depicted in Figure 3.9. Table 3.1 summarizes the logic behind the assertion of the categories.

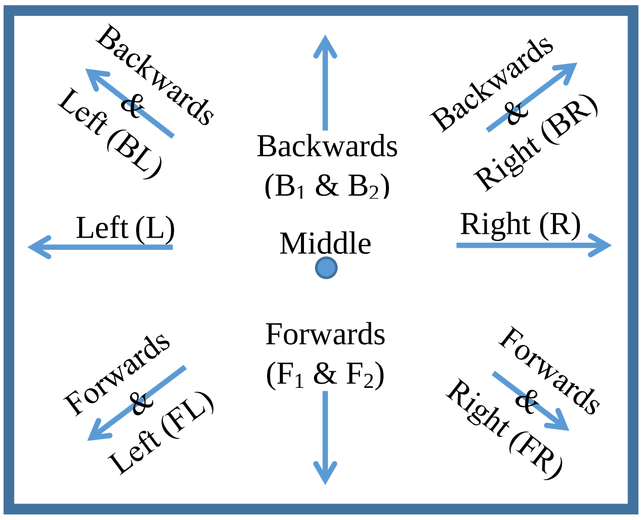


Figure 3.9: Depiction of the 11 categories.

Table 3.1 shows how a total of 11 different possibilities that are extracted from the movements of the tongue. The category of ‘Forwards’ is divided into 2 categories of F_1 and F_2, and similarly, ‘Backwards’ is also divided into B_1 and B_2. This is made possible by the assertion of the presence of teeth. Thus, when the users opt for the ‘Forwards’ option with their jaw open to the extent of their teeth being visible, this would constitute the F_1 option, and when the users close their jaw with their teeth being hidden, this would constitute the F_2 option. Similarly, the ‘Backwards’ option is also divided into two parts: B_1 (with their teeth visible) and B_2 (without their teeth visible). This helps to extract two new inputs from the user and thus enhancing the range of inputs.

The way that the presence of the teeth is asserted is by adjusting the range of the HSV values of a cloned image such that the teeth are isolated from the surroundings. Given the significant contrast between the color of the teeth and its surroundings, this step is considerably easier than the isolation of the tongue. Once the HSV range is selected, the image is then binarized and then the ‘closing’ morphological operation is then applied (similar to the procedure explained in 3.4.4). In order to gauge the presence of the teeth, a different set of regions of interest has to be set up owing to the location of the teeth.

Table 3.1: Evaluation of the categories based on the centroid of the contour and the white pixels in the regions of interest (1=tongue is present in that region, 0=tongue is not present in that region, ✗=don't care).

Regions of Interest	**C_x = Right**			**C_x = Left**			**Middle**	**C_y = Front**	**C_y = Back**	**C_y = Front**	**C_y = Back**
								With Teeth		**Without Teeth**	
	FR	**R**	**BR**	**FL**	**L**	**BL**	**M**	**F_1**	**B_1**	**F_2**	**B_2**
1C	✗	✗	✗	✗	✗	1	✗	✗	✗	✗	✗
2C	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
3C	✗	✗	1	✗	✗	✗	✗	✗	✗	✗	✗
1	✗	✗	0	✗	✗	1	1	✗	1	✗	1
2	✗	✗	✗	✗	✗	✗	1	✗	1	✗	1
3	✗	✗	1	✗	✗	0	1	✗	1	✗	1
4	✗	✗	✗	✗	1	0	1	1	0	✗	0
5	✗	✗	✗	✗	✗	✗	1	1	✗	✗	✗
6	✗	1	0	✗	✗	✗	1	1	0	✗	0
7	0	0	✗	1	0	0	0	✗	✗	1	✗
8	✗	✗	✗	✗	✗	✗	0	✗	✗	1	✗
9	1	0	0	0	0	✗	0	✗	✗	1	✗

The selected regions (for checking the presence of teeth) are shown in Figure 3.10 and the numbers of white pixels in that region have to be independently evaluated. A threshold for the number of white pixels is manually selected which can be used to compare the existing white pixels to its value. If the white pixels within the regions of interest are greater than the threshold, a variable is assigned a value of 1, otherwise it is assigned a 0. This variable is then used to determine whether the 'Forwards' option selected by the user is F_1 or F_2, and whether the 'Backwards' option selected by the user is B_1 or B_2.

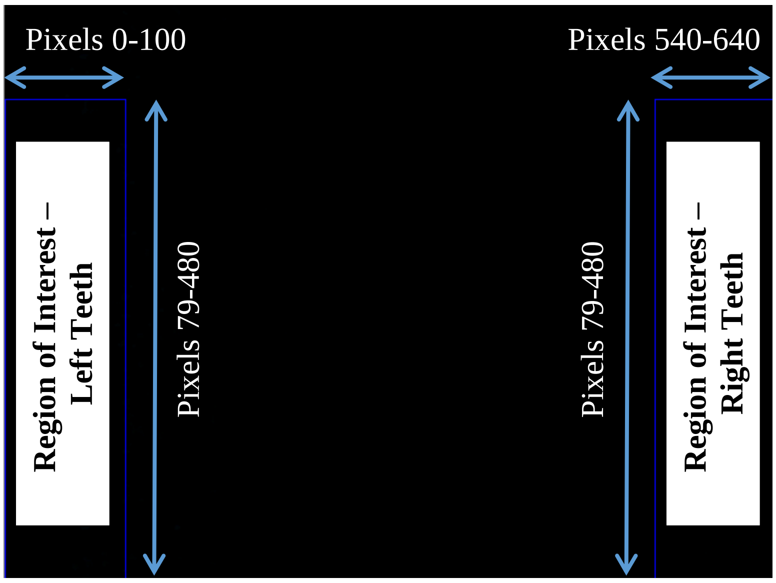

Pixels 0-100
Pixels 540-640
Region of Interest – Left Teeth
Pixels 79-480
Pixels 79-480
Region of Interest – Right Teeth

Figure 3.10: Segmentation of the 640×480 image into 2 regions for gauging the presence of teeth.

Chapter 4. Experimental Setup

In this chapter, the complete prototype of the intraoral camera along with a personalized mouthguard is presented. Having a prototype set in place also allows the system to be reproducible since the location of the camera and LEDs would be constant. The application on which the prototype was tested is also explained towards the end of this chapter.

4.1. The Personalized Mouthguard

The Endoscope camera had to be fixed on the roof of the mouth, and thus a mouldable mouthguard for the upper jaw was used in order to manufacture a personalized mouthguard. The mouthguard (shown in Figure 4.1) is made of silicone and can alter its shape when kept in hot water.

Figure 4.1: The Adidas ADIBP09 Single Mouthguard.

The Mouthguard is moulded such that it could get fixed on the upper jaw without much effort from the user. Furthermore, a hollow cylinder is also formed using the same material, in the middle of the mouthgaurd in order to fix the Endoscope camera. Lastly, 2 LilyPad LEDs are fixed on either side of the camera to light up the intraoral cavity for better image quality and also to make it possible to implement the image processing techniques.

The angle of the LEDs along with their brightness is of significant importance in order to make the image processing successful. This is due to the fact that details of the 'saturation' and 'value' from the HSV format were being used in order to isolate the tongue from the floor of the mouth. In order to achieve this, the LEDs have to be pointing at such an angle that would brighten up the tongue considerably more than the floor of the mouth. Furthermore, the brightness of the LEDs is also manually altered by using different valued resistors in series with the LEDs. A resistor of 201.2Ω was finally selected which gave the best image processing results. The mouthpiece is shown in Figures 4.2 and 4.3, while the overall system is further clarified in Figure 4.4.

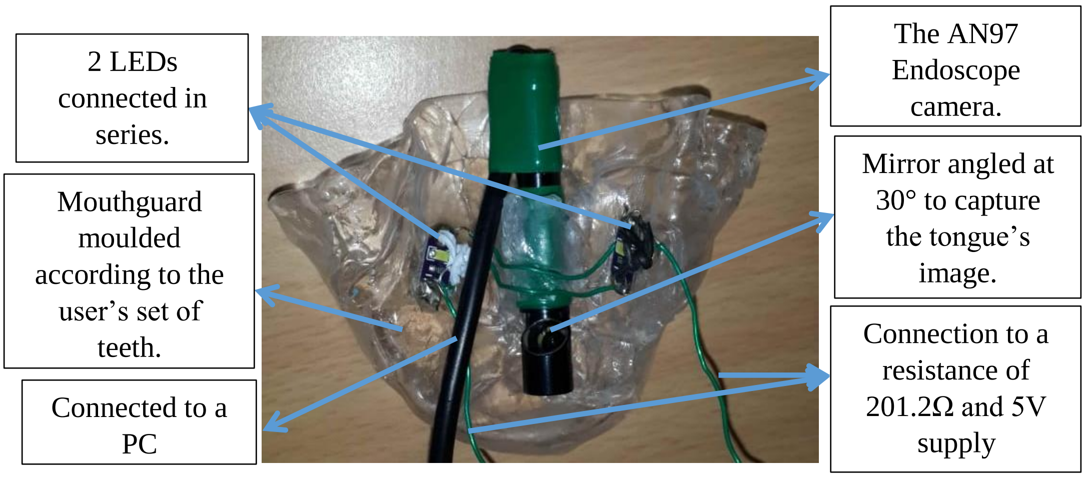

2 LEDs connected in series.
Mouthguard moulded according to the user's set of teeth.
Connected to a PC
The AN97 Endoscope camera.
Mirror angled at 30° to capture the tongue's image.
Connection to a resistance of 201.2Ω and 5V supply

Figure 4.2: The personalized mouthguard along with the LEDs and the Endoscope camera.

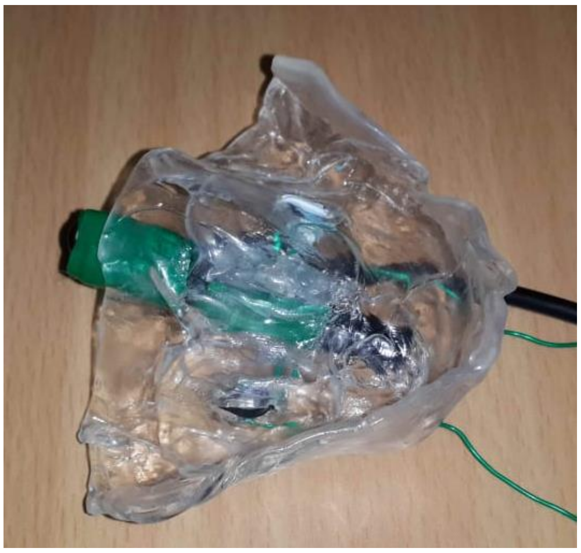


Figure 4.3: The bottom view of the personalized mouthguard.

As seen from the bottom view of the mouthguard, it has been moulded by taking the impression of the user to ensure suction when the user wears the mouthpiece and thus easily fits inside the mouth. This is important to make the system comfortable for the user and that usage for long hours doesn't cause any physical pain.

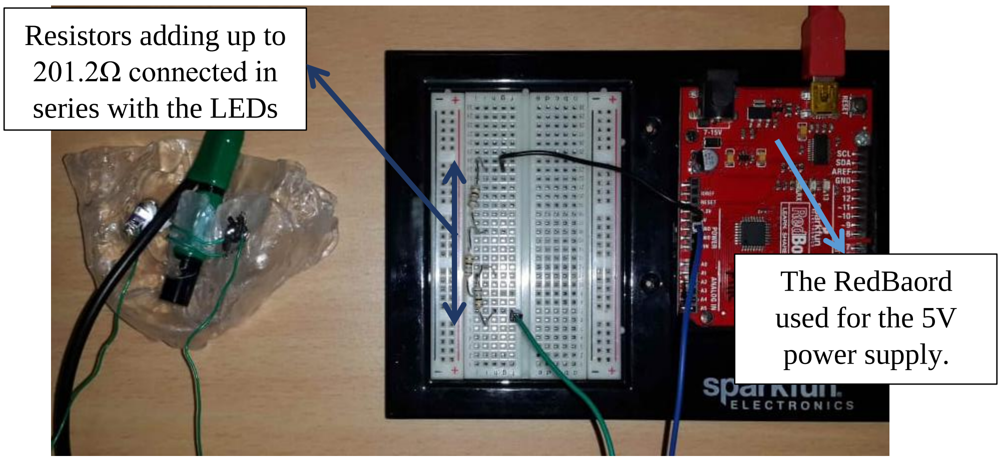

Resistors adding up to 201.2Ω connected in series with the LEDs
The RedBaord used for the 5V power supply.

Figure 4.4: The overall prototype.

4.2. The Interface

The Endoscope camera is connected to the PC using a Micro USB (female) to USB (male) adapter. The Python programming language is used in order to complete the image processing tasks which make extensive use of the OpenCV library. By using Python, it is very easy to also run the same code on different platforms such as the Raspberry Pi, which is an inexpensive credit card sized computer. The small-size and image processing capabilities would make it an ideal choice for connecting our system especially to mobile applications (such as an automatic wheelchair, etc.).

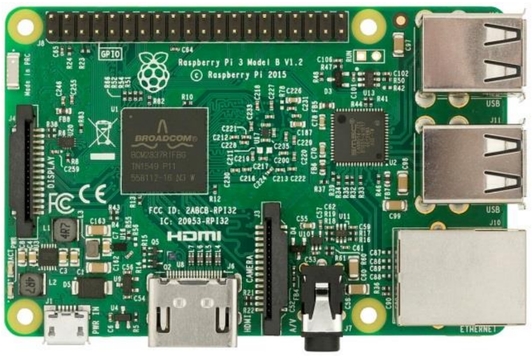


Figure 4.5: Raspberry Pi 3 Model B.

4.3. Application 1: Controlling a Robotic Arm

The information obtained about the position and orientation of the tongue can then applied to any assistive device. The application selected in this case is a 5 degrees of freedom robotic arm (as shown in Figure 4.6) which is useful for pick and place operations and especially valuable for paralytic patients. The information of the location of the tongue is utilized and processed in order to make the robotic arm move as desired by the user. The robot is connected to the PC using a USB, and the connection via Python is established by writing a code to access the COM port using serial communication. The servo motors of the robotic arm are controlled by either incrementing or decrementing the angles based on the orientation of the tongue.

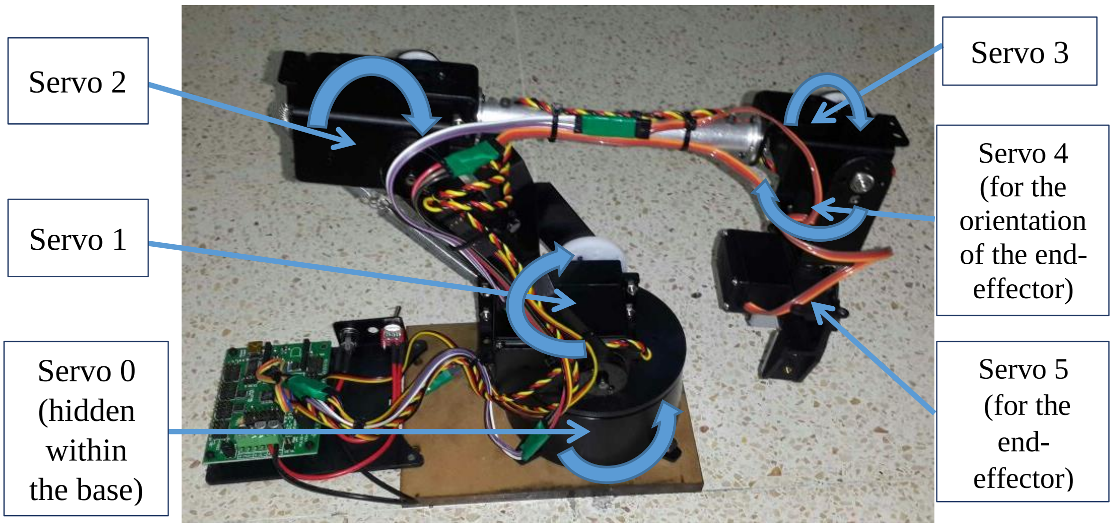

Servo 2
Servo 1
Servo 0 (hidden within the base)
Servo 3
Servo 4 (for the orientation of the end-effector)
Servo 5 (for the end-effector)

Figure 4.6: A five degrees of freedom robotic arm

Before the implementation of a code, the logic behind the increments and decrements of the angles of the servo motors was decided. The commands are summarized in Table 4.1.

Table 4.1: Determining the commands sent to the robot according to each acquired position of the tongue.

Category	**Command to the Robot**
Forwards + Right (FR)	Servo 0: Fast Increment for clockwise rotation.
Right (R)	Servo 0: Slow Increment for clockwise rotation.
Backwards + Right (BR)	Servo 4: Increment for clockwise rotation.
Forwards + Left (FL)	Servo 0: Fast Increment for anti-clockwise rotation.
Left (L)	Servo 0: Slow Increment for anti-clockwise rotation.
Backwards + Left (BL)	Servo 4: Increment for anti-clockwise rotation.

Forwards With Teeth (F_1)	Servo 2: Increment for forward movement.
Backwards With Teeth (B_1)	Servo 2: Increment for backward movement.
Forwards Without Teeth (F_2)	Servo 1: Increment for forward movement.
Backwards Without Teeth (B_2)	Servo 1: Increment for backward movement.

As seen from Table 4.1, the servo 3 is unused since the calculation for its angle will be such that it is exactly 90° to the horizontal axis, which is what is generally required for the pick and place tasks. This can be achieved by calculating the angles from servo 1 and servo 2 as explained in Figure 4.7.

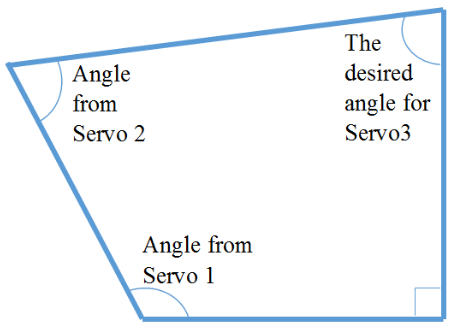


Figure 4.7: Calculation of the angle for Servo 3.

As the sum of the interior angles within quadrilaterals adds up to 360°, the formula for finding the angle for Servo 3 is quite straightforward. Servo 3 will consequently position the end-effector to be perpendicular to the horizontal surface. In order to move the end-effector for picking up or releasing an object, a button was used as in input (which could be replaced by any customized input based on the demands of the user.

$$Servo\ 3\ Angle\ = 360 - Servo\ 2\ Angle - Servo\ 1\ Angle \qquad (4.1)$$

4.4. Application 2: Typing

The aim of this application was to allow the users to type using their tongues. In order to achieve this, the outputs from the system would have to allow for the selection of the characters that appear on the screen along with an option to scroll through the different characters. A total of 36 different options were selected which included the 26 alphabets, 7 special characters, a ‘backspace’, ‘space’ and ‘enter’. A scrolling option had to be created such that the user would be able to view 9 options at a time. If the user selects a particular option for a certain amount of time (which is selected by trial and error), the option is highlighted, indicating that the selection is confirmed, and the text eventually appears on the screen. In order to achieve this, the F_1 (Forwards with Teeth) and B_1 (Backwards with Teeth) categories were selected for the scrolling options, and the rest of the 9 categories were for the selection of the options. The different options appearing on screen for the user are shown in Figures 4.8-4.12.

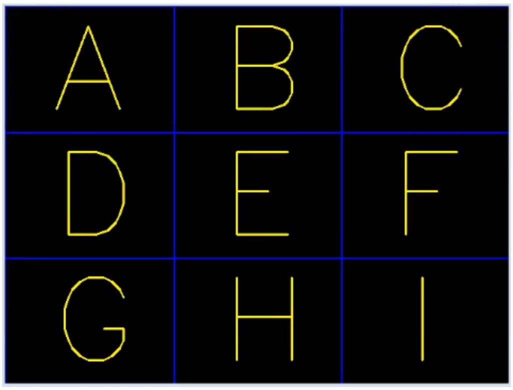


Figure 4.8: Options 1-9 for the user.

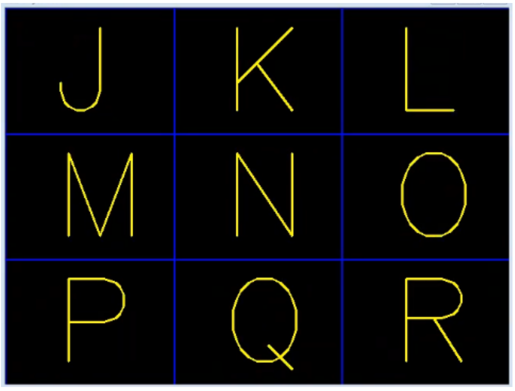


Figure 4.9: Options 10-18 for the user.

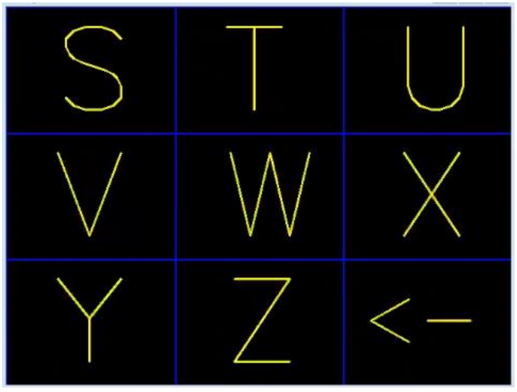


Figure 4.10: Options 19-27 for the user.

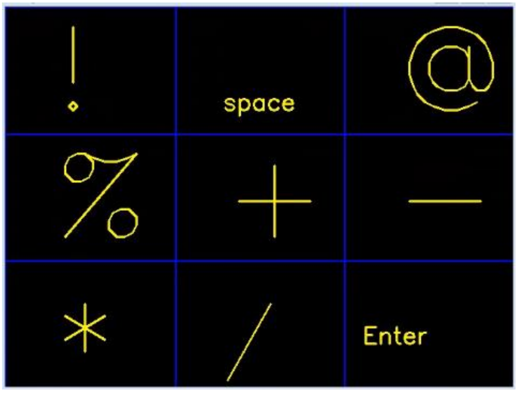


Figure 4.11: Options 28-36 for the user.

In order to achieve the above, a counter was first initiated in the code which would increment or decrement from 1-4 based on the F_1 and B_1 selections. Each number of the counter corresponded to a database of the options which would be connected to the category that was selected. For example, when the counter is 1, the 9 categories, namely, BL, B_2, BR, L, M, R, FR, F_2 and FL, would correspond to the characters, ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’ and ‘I’ respectively. The same would apply for the different counters with the only difference being the characters that are selected. Once the characters are selected, they are stored in a string and displayed on the screen. The backspace option deletes the last character in the string by reducing the size of the string by one each time it is selected.

4.5. Cost Analysis

A breakdown of the calculation of the cost is provided in Table 4.2 which makes it evident that the complete system is quite low in cost.

Table 4.2: Cost Analysis.

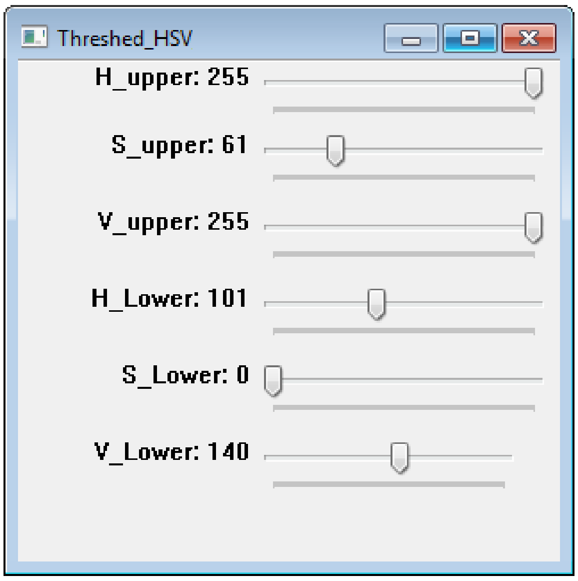
No.	Product Name	Cost
1	Adidas ADIBP09 Single Mouth Guard	32.0 AED
2	Waterproof Endoscope Snake Inspection Borescope Camera For Android Phones	36.0 AED
3	LilyPad LED White (5pcs)	12.0 AED
Total		**80.0 AED**

Chapter 5. Results and Analysis

In this chapter, we present the image processing results which are obtained by implementing a Python code on a PC with extensive use of the OpenCV library. The results of the steps mentioned in Chapters 3 and 4 are presented in this chapter.

5.1. Capturing the HSV Range

The first step was to determine the best HSV range which will be used to isolate the tongue from its surroundings. In order to do that systematically, a trackbar was created in Python which would allow to dynamically change the upper and lower bounds for the HSV range in real time. After going through the different possible ranges, the best range was found as shown in Figure 5.1.


Threshed_HSV

H_upper: 255

S_upper: 61

V_upper: 255

H_Lower: 101

S_Lower: 0

V_Lower: 140

Figure 5.1: A trackbar created in Python to tune the range in real-time (with the numbers representing the best solution found to isolate the tongue from its surroundings).

5.2. Summary of the Complete Procedure

The procedure explained in depth in Chapter 3 is summarized in the following steps. The original colored image (as seen in Figure 5.2) is first smoothed and then converted into the HSV format. A mask is then applied in order to just view the area of the image which falls within the HSV range (as seen in Figure 5.3). After further smoothing of the image, it is then converted to grayscale (as seen in Figure 5.4). The Otsu's binarization technique is then applied to the grayscale image along with the 'closing' Morphological operation (as seen in Figure 5.5). Next, the contours are detected making sure that the contours of the background noise are eliminated based on the presence of the tongue in the image (to avoid capturing any contours when the tongue is completely pulled back by the user) and the size and location of the contour. The binary image with the background noise being eliminated is shown in Figure 5.6. Finally, Figure 5.7 shows the original image along with the centroid of the contour (represented by the green dot), an ellipse which fits the contour and the regions of interest which are highlighted red only if the tongue is detected in that region.

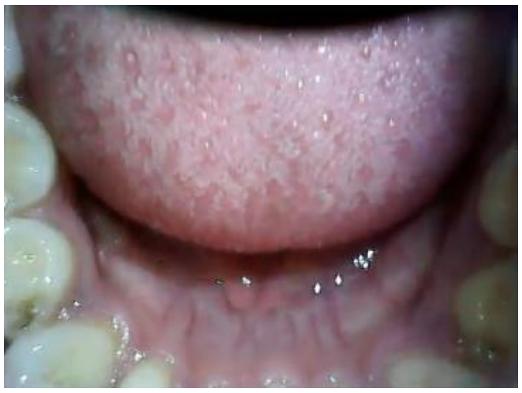


Figure 5.2: Original Image.

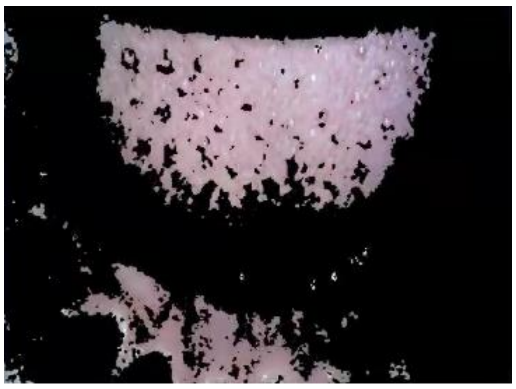


Figure 5.3: After applying a mask on the original image to only view the colors within the HSV range.

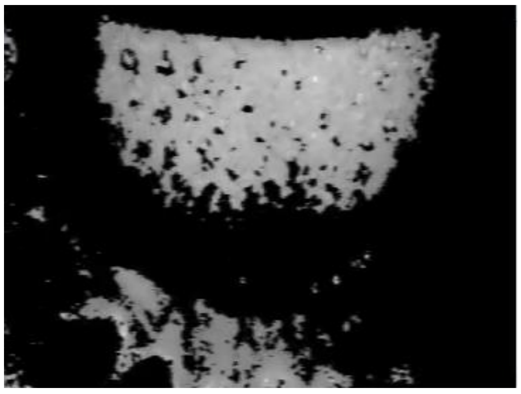


Figure 5.4: After conversion to grayscale.

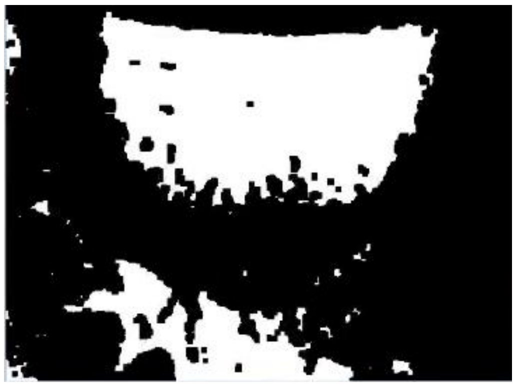


Figure 5.5: After applying the Otsu's binarization technique along with the application of the 'closing' operation.

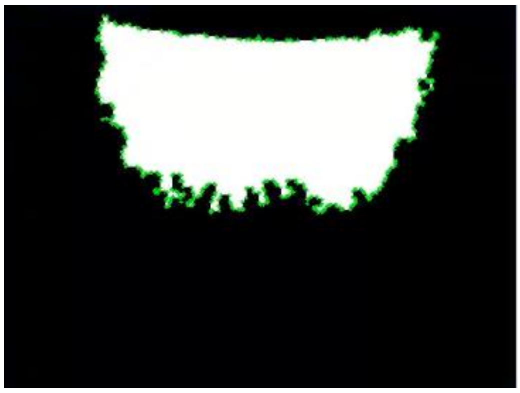


Figure 5.6: After the removal of noise and the addition of the contour.

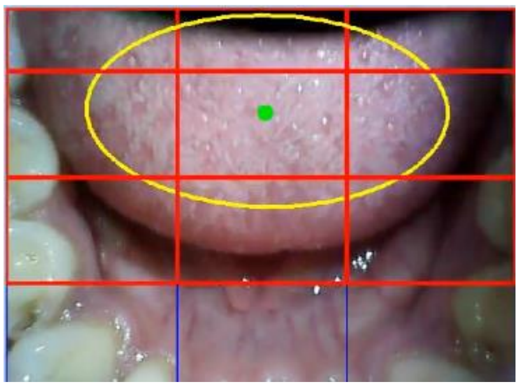


Figure 5.7: The final image with information about the centroid of the contour, the fitted ellipse and the regions of interest covered by the tongue.

5.3. Results within Different Categories

In order to prove the accuracy of the system, the results for each of the different categories (which were summarized in Table 3.1) are also presented below with the conclusion of the control signal also specified on the bottom left in each colored frame containing the segments.

Category 1: Backwards (No Teeth) – B_2

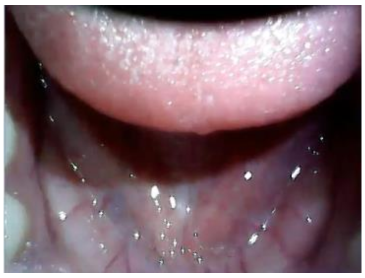


Figure 5.8: Original Image (Backwards).

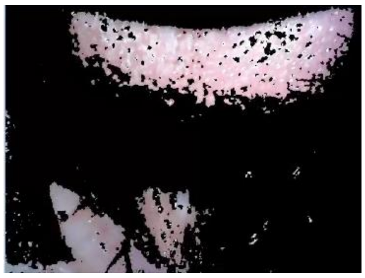


Figure 5.9: After applying the HSV mask (Backwards).

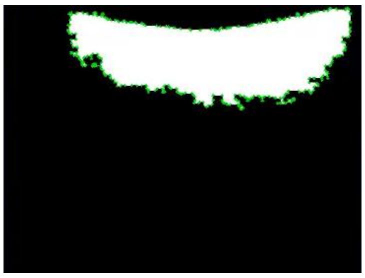


Figure 5.10: Binary image after removing the noise (Backwards).

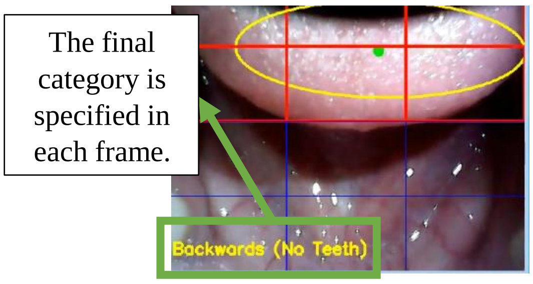


Figure 5.11: Image with segmentation, fitted ellipse and centroid of the contour (Backwards).

Category 2: Forwards (No Teeth) – F_2

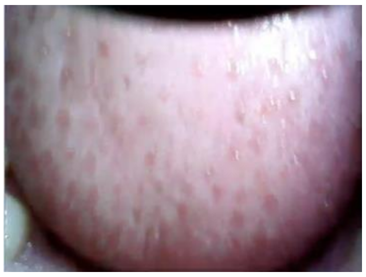


Figure 5.12: Original Image (Forwards).

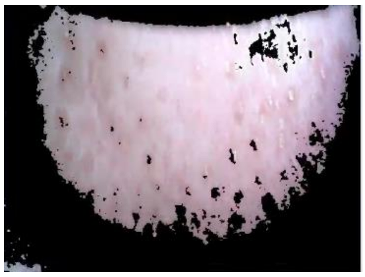


Figure 5.13: After applying the HSV mask (Forwards).

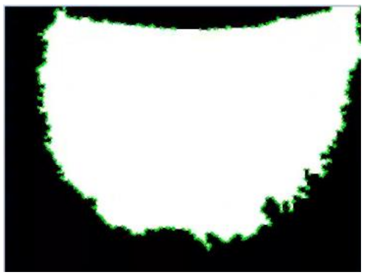


Figure 5.14: Binary image after removing the noise (Forwards).

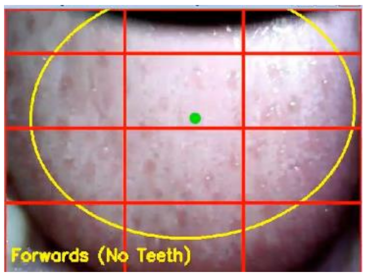


Figure 5.15: Image with segmentation, fitted ellipse and centroid of the contour (Forwards).

Category 3: Backwards (With Teeth) – B_1

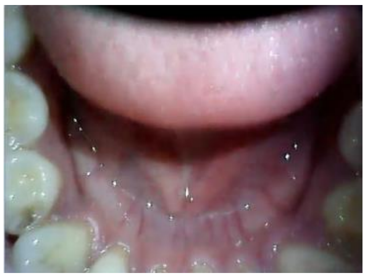


Figure 5.16: Original Image (Backwards with Teeth).

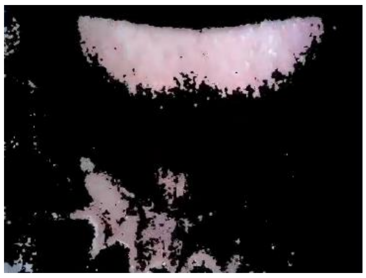


Figure 5.17: After applying the HSV mask (Backwards with Teeth).

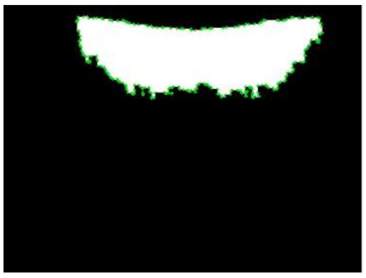


Figure 5.18: Binary image after removing noise (Backwards with Teeth).

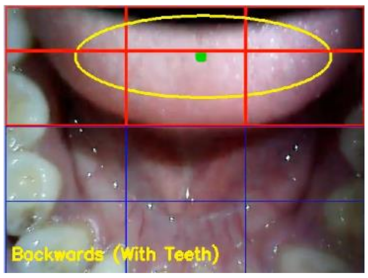


Figure 5.19: Image with segmentation, fitted ellipse and centroid of the contour (Backwards with Teeth).

Category 4: Forwards (With Teeth) – F_1

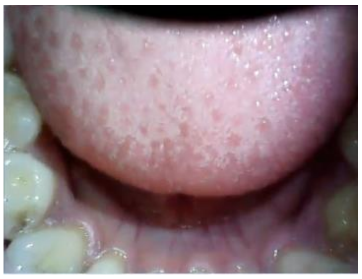


Figure 5.20: Original Image (Forwards with Teeth).

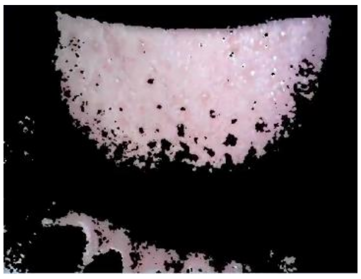


Figure 5.21: After applying the HSV mask (Forwards with Teeth).

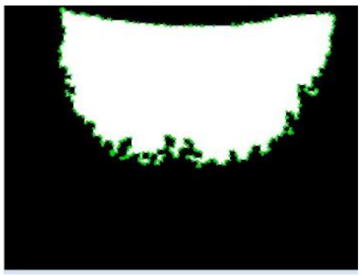


Figure 5.22: Binary image after removing noise (Forwards with Teeth).

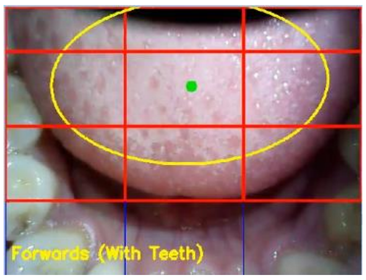


Figure 5.23: Image with segmentation, fitted ellipse and centroid of the contour (Forwards with Teeth).

Category 5: Forwards + Right (FR)

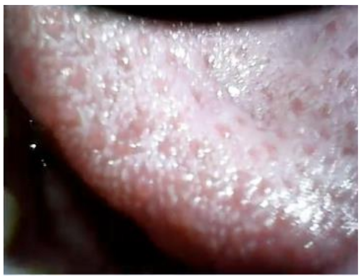


Figure 5.24: Original Image (Forwards + Right).

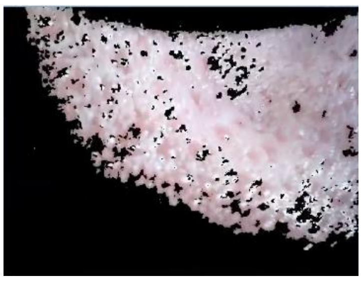


Figure 5.25: After applying the HSV mask (Forwards + Right).

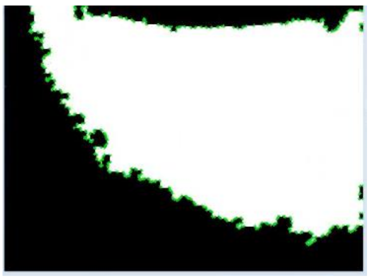


Figure 5.26: Binary image after removing noise (Forwards + Right).

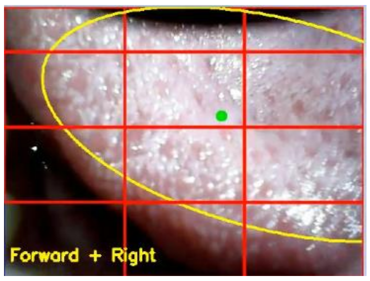


Figure 5.27: Image with segmentation, fitted ellipse and centroid of the contour (Forwards + Right).

Category 6: Right (R)

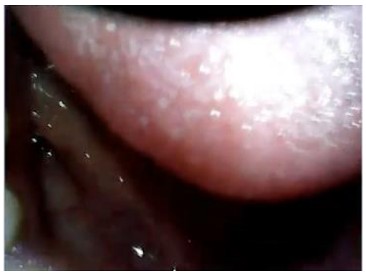


Figure 5.28: Original Image (Right).

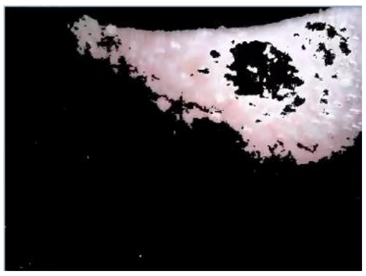


Figure 5.29: After applying the HSV mask (Right).

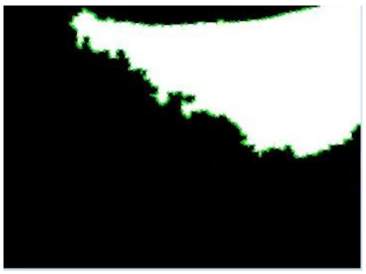


Figure 5.30: Binary image after removing noise (Right).

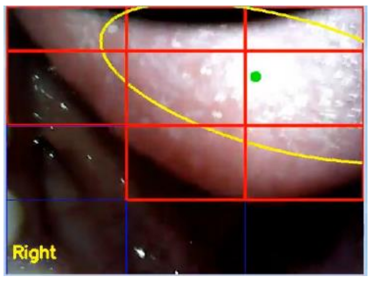


Figure 5.31: Image with segmentation, fitted ellipse and centroid of the contour (Right).

Category 7: Backwards + Right (BR)

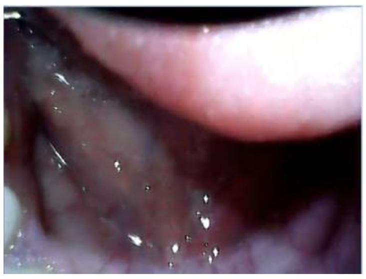


Figure 5.32: Original Image (Backwards + Right).

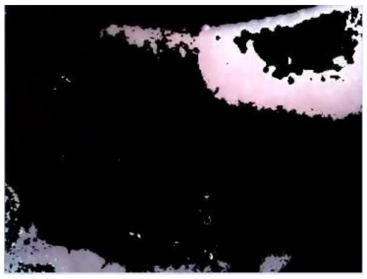


Figure 5.33: After applying the HSV mask (Backwards + Right).

Figure 5.34: Binary image after removing noise (Backwards + Right).

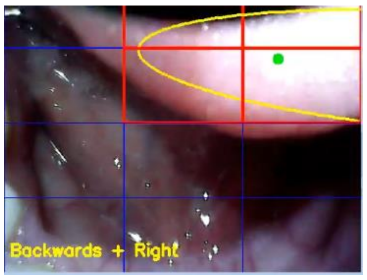


Figure 5.35: Image with segmentation, fitted ellipse and centroid of the contour (Backwards + Right).

Category 8: Forwards + Left (FL)

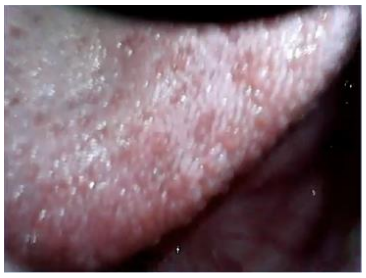


Figure 5.36: Original Image (Forwards + Left).

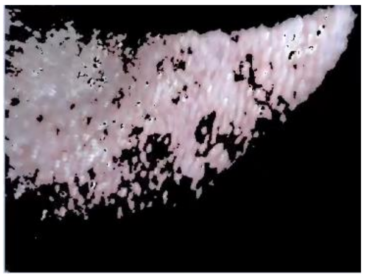


Figure 5.37: After applying the HSV mask (Forwards + Left).

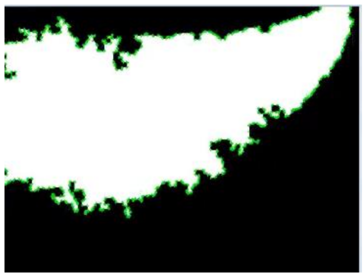


Figure 5.38: Binary image after removing noise (Forwards + Left).

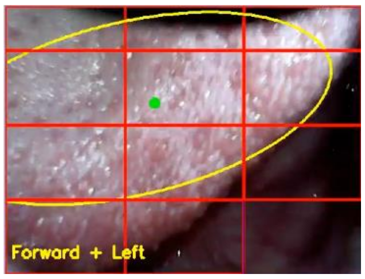


Figure 5.39: Image with segmentation, fitted ellipse and centroid of the contour (Forwards + Left).

Category 9: Left (L)

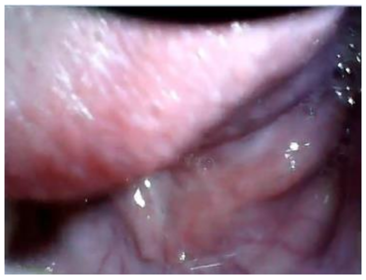


Figure 5.40: Original Image (Left).

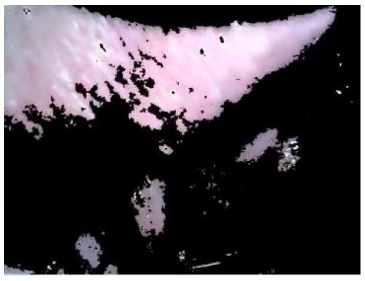


Figure 5.41: After applying the HSV mask (Left).

Figure 5.42: Binary image after removing noise (Left).

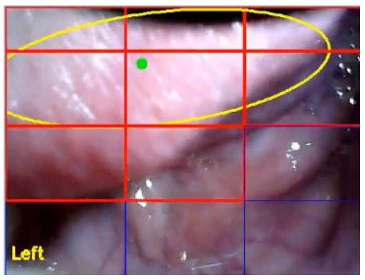


Figure 5.43: Image with segmentation, fitted ellipse and centroid of the contour (Left).

Category 10: Backwards + Left (BL)

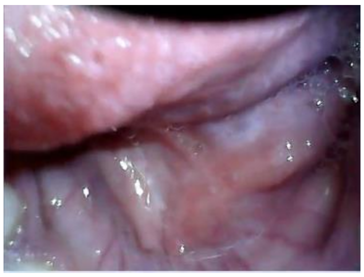


Figure 5.44: Original Image (Backwards + Left).

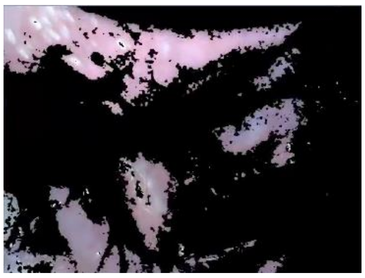


Figure 5.45: After applying the HSV mask (Backwards + Left).

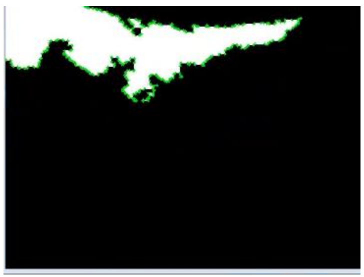


Figure 5.46: Binary image after removing noise (Backwards + Left).

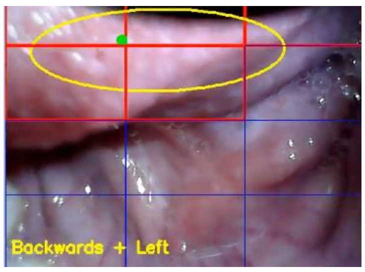


Figure 5.47: Image with segmentation, fitted ellipse and centroid of the contour (Backwards + Left).

Category 11: Middle

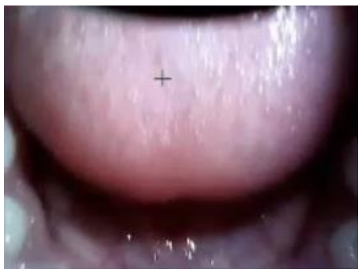


Figure 5.48: Original Image (Middle).

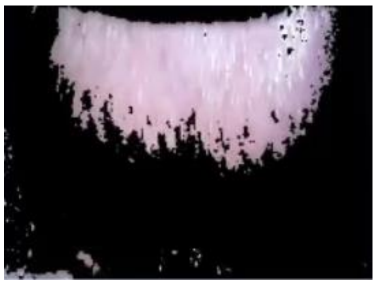


Figure 5.49: After applying the HSV mask (Middle).

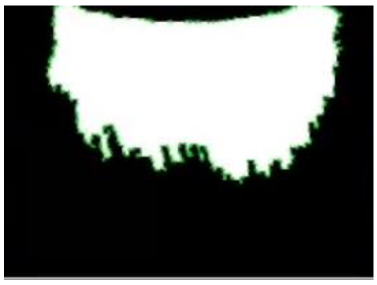


Figure 5.50: Binary image after removing noise (Middle).

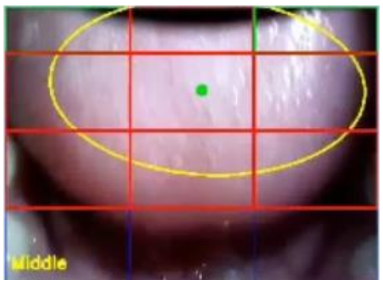


Figure 5.51: Image with segmentation, fitted ellipse and centroid of the contour (Middle).

The above results conclude that with the proposed image processing techniques, the position and orientation of the tongue were accurately determined for all of the 11 categories. This would form the control signal which could be used as an input to any assistive device. Based on Table 4.1, the control signal was then used in order to increment or decrement the angles of the servo motors of a robot arm.

5.4. Testing the Performance of the System Using the Robotic Arm

In order to test the performance of the system, a set of experiments were given to a user to complete. The user was asked to output all of the different categories in a particular order by moving his tongue. Each experiment consisted of the user going through all of the categories 5 times. The performance evaluation was done in 2 ways:

- **Criteria 1:** If the user was able to reach the desired output with his tongue effortlessly, a '1' was awarded for that particular part of the trial. On the other hand, if the user struggled to reach the output (which included flickering of the output, taking some time to adjust his tongue to the exact position and a wrong output being displayed on the screen), a '0' was marked for that part. It is worth noting that in this first criterion the '0' does not necessarily indicate a complete failure of the user to reach the desired output, but it indicates the inefficiency in reaching his goal.
- **Criteria 2:** In this criterion it was noted whether or not the user was eventually able to reach the desired output, without taking into account the adjustments of the tongue made by the user.

5.4.1. Experiment 1

Each experiment consisted of 5 trials and the performance of the user and the system were evaluated for each case. Tables 5.1 and 5.2 summarize the performance for the 1st experiment.

The percentage of accuracy of the user from the 1st experiment can be calculated from the total as follows:

$$Accuracy\ (Experiment\ 1, Criteria\ 1) = \frac{30}{50} \times 100 = \mathbf{60}\% \qquad (5.1)$$

Table 5.1: Performance evaluation (1st experiment, criteria 1).

	Categories										Sum for Each Trial (Max. 10)
	FR	R	BR	FL	L	BL	F_1	B_1	F_2	B_2	
Trial 1	1	1	1	1	1	1	1	0	0	0	**7**
Trial 2	1	1	0	0	1	0	1	0	1	1	**6**
Trial 3	1	1	1	0	0	0	1	1	0	1	**6**
Trial 4	1	1	0	1	1	0	0	0	0	1	**5**
Trial 5	1	1	1	1	0	1	1	0	0	0	**6**
Total	**5**	**5**	**3**	**3**	**3**	**2**	**4**	**1**	**1**	**3**	**30**

Moreover, the user was also allowed to adjust his tongue in order to ensure that he eventually reaches the desired output. Table 5.2 summarizes how many times the user was eventually able to reach the desired output in the 1st experiment.

Table 5.2: Evaluation of whether or not the user was eventually able to reach his desired output for all cases (1st experiment, criteria 2).

	Categories										Sum for Each Trial (Max. 10)
	FR	R	BR	FL	L	BL	F_1	B_1	F_2	B_2	
Trial 1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 2	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	**9**
Trial 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 4	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	**8**
Trial 5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Total	**5**	**5**	**5**	**5**	**5**	**3**	**4**	**5**	**5**	**5**	**47**
Total Time Taken = 358 seconds											

$$Accuracy\ (Experiment\ 1, Criteria\ 2) = \frac{47}{50} \times 100 = \mathbf{94\%} \qquad (5.2)$$

5.4.2. Experiment 2

The experiment conducted was similar to the 1st experiment, with the only difference being that the user was now more accustomed to the system. Tables 5.3 and 5.4 summarize the performance for the 2nd experiment.

Table 5.3: Performance evaluation (2nd experiment, criteria 1).

	Categories										**Sum for Each Trial (Max. 10)**
	FR	**R**	**BR**	**FL**	**L**	**BL**	**F_1**	**B_1**	**F_2**	**B_2**	
Trial 1	1	1	1	1	1	0	0	1	1	1	**8**
Trial 2	1	1	1	1	0	0	0	1	0	0	**5**
Trial 3	1	1	1	0	1	1	1	0	0	1	**7**
Trial 4	1	1	1	1	1	1	1	1	0	1	**9**
Trial 5	1	1	0	1	0	1	1	1	0	1	**7**
Total	**5**	**5**	**4**	**4**	**3**	**3**	**3**	**4**	**1**	**4**	**36**

$$Accuracy\ (Experiment\ 2, Criteria\ 1) = \frac{36}{50} \times 100 = \mathbf{72\%} \qquad (5.3)$$

Table 5.4: Evaluation of whether or not the user was eventually able to reach his desired output for all cases (2nd experiment).

	Categories										**Sum for Each Trial (Max. 10)**
	FR	**R**	**BR**	**FL**	**L**	**BL**	**F_1**	**B_1**	**F_2**	**B_2**	
Trial 1	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	**9**
Trial 2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Total	5	5	5	5	5	3	5	5	5	5	**49**
Total Time Taken = 246 seconds											

$$Accuracy\ (Experiment\ 2, Criteria\ 2) = \frac{49}{50} \times 100 = \mathbf{98\%} \qquad (5.4)$$

5.4.3. Experiment 3

Tables 5.5 and 5.6 summarize the performance for the 3rd experiment.

Table 5.5: Performance evaluation (3rd experiment, criteria 1).

	Categories										**Sum for Each Trial (Max. 10)**
	FR	**R**	**BR**	**FL**	**L**	**BL**	**F_1**	**B_1**	**F_2**	**B_2**	
Trial 1	1	1	0	1	0	1	1	1	1	1	**8**
Trial 2	1	1	1	1	0	0	1	1	1	1	**8**
Trial 3	1	1	0	1	1	1	1	0	1	0	**7**
Trial 4	1	1	0	1	1	1	1	1	1	1	**9**
Trial 5	1	1	1	1	1	1	1	1	1	1	**10**
Total	**5**	**5**	**2**	**5**	**3**	**4**	**5**	**4**	**5**	**4**	**42**

$$Accuracy\ (Experiment\ 3, Criteria\ 1) = \frac{42}{50} \times 100 = \mathbf{84\%} \qquad (5.5)$$

Table 5.6: Evaluation of whether or not the user was eventually able to reach his desired output for all cases (3rd experiment, criteria 2).

	Categories										**Sum for Each Trial (Max. 10)**
	FR	**R**	**BR**	**FL**	**L**	**BL**	**F_1**	**B_1**	**F_2**	**B_2**	
Trial 1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Trial 5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**10**
Total	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**50**
Total Time Taken = 226 seconds											

$$Accuracy\ (Experiment\ 3, Criteria\ 2) = \frac{50}{50} \times 100 = \mathbf{100\%} \qquad (5.6)$$

The above experiments and the percentages of the accuracies found in formulas 5.1-5.6 make it clear that the effectiveness of the system is also dependant on the amount of training the user undergoes. This is also evident by analysing the time it takes for the user to complete each experiment (as also shown in Figure 5.48).

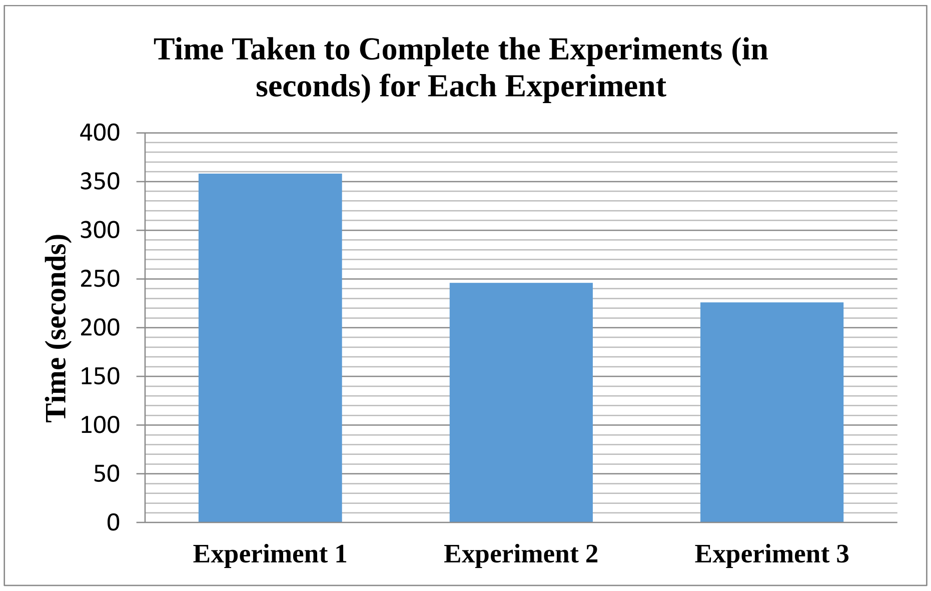


Figure 5.52: Analysis of the time taken by the user to complete each experiment.

As the user gets used to system, he/she is likely to make less mistakes. On most of the occasions that the user achieved a ‘0’ mark (for ‘criteria 1’ in the 3 experiments), he was still able to adjust his tongue further to reach the desired objective.

5.5. Testing the Performance of the System Using the Typing System

The typing system, explained in section 4.4, was also tested which allows the user to select from 36 different options. Figures 5.49-5.51 show a user selecting an option while attempting to write, ‘THANK YOU.’ The message is stored in a string inside the code and also appears on the screen for confirmation.

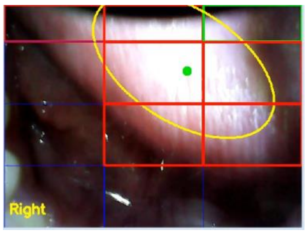


Figure 5.53: The user aligns his tongue to the right in order to select the letter ‘O’.

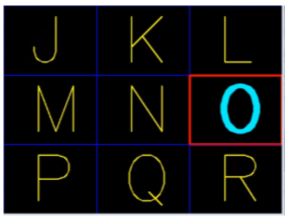


Figure 5.54: The letter ‘O’ is highlighted confirming that the letter has been selected successfully.

Figure 5.55: The letter ‘O’ also appears on the screen.

Table 5.7 shows a breakdown of the analysis of the different trials of the user which would provide an idea of the system performance.

Table 5.7: Analysis of the typing tests.

Trial No.	**Text Entered by the User**	**No. of Mistakes Made by the User**	**Explanation of the Mistake**	**Time Taken to Complete the Task**	**No. of Correct Characters/ Min**
1	All the letters of the alphabets without any space	1	User intended to select option F_1 but entered F_2 instead.	95 seconds	16.42
2	'This is Wonderful!'	2	1: Forgot to add space between words. 2: Unintentional selection of a character.	203 seconds	5.32
3	'An Intraoral Camera!'	7	1: User intended to select F_1 but selected FL instead. 2-6: Unintentional entry of category B_2 thrice when the user wanted to select B_1 (for scrolling). 7: User accidentally selected category L instead of FL.	234 seconds	5.13
4	'Muhammad Amin Tily'	3	1-2: Unintentional selection of category B_2 instead of B_1. 3: Wrong entry of a character.	187 seconds	5.78
5	'Thank You!'	0	-	74 seconds	8.11

Most of the errors of the user were during the scrolling phase as the F_1 and B_1 categories would instead be inputted as F_2 and B_2 respectively. This would easily be solved with further training as the user would get used to the amount required to open or close his mouth in order to distinguish between the categories. Table 5.8 shows the evaluation of the error rate taking into consideration even the scrolling and the backspace since those options were selected using the tongue as well.

Table 5.8: Evaluation of the Error Rate.

Trial No.	**No. of Charac-ters of the Text**	**No. of Mistakes Made (corres-ponding to extra charac-ters)**	**No. of Times the Back-space was Used**	**No. of Times the F_1 or B_1 Category was Selected (for Scrolling)**	**Total No. of Options Selected (including characters, backspace and scrolling options)**	**Error Rate (%)**	**No. of Correct Entries/Min**
1	26	1	1	3	31	**3.2%**	**18.95**
2	18	2	2	32	54	**3.7%**	**15.37**
3	20	7	7	47	81	**8.6%**	**18.97**
4	18	3	3	46	70	**4.3%**	**21.5**
5	10	0	0	17	27	**0%**	**21.89**

On average, the error rate for the user was 3.96% but as seen with the individual trials, they are dependant on how comfortable the user has become to the system. The error rate even reached 0% for one of the trials. On average, the number of correct entries/minute was 19.34. The comparison of the performance of this system with different techniques is shown in Table 5.9.

Table 5.9: Comparison with different techniques.

Technique	**Number of Inputs**	**Average Error Rate (%)**	**Number of Characters/Min**
Resistopalatography [21]	9	5.63%	-
MouthPad [22]	8	2.3%	-
Tongue Drive System (TDS) [31]	6	-	8.8 char/min
The Intraoral Camera	11	3.96%	8.15 char/min

The intraoral camera introduced in this work outperforms the Resistopalatography and the MouthPad in terms of the average error rate. Furthermore, the intraoral camera is able to allow for more inputs compared to the other systems operated using the tongue.

5.6. Limitations and Inaccuracies of the System

The system is very sensitive to the lighting of the intraoral environment. Since the system attempts to isolate the tongue from its background by using the information of the 'hue', 'saturation' and 'value' of the image, it is essential that the LEDs are positioned in such a manner that it lights the tongue significantly more than the floor of the mouth. If the lighting is not accurate, it may lead to the detection of false contours arising from the reflection of the floor of the mouth. The way this error was minimized through the code was by ensuring that only one contour would be drawn in each iteration of the loop based on 3 factors:

1. Whether or not the tongue is present in the image (which is detected by the regions defined as 1C, 2C and 3C which was explained in section 3.4.7), and
2. The contour with the largest area is always selected.
3. The contour would only be displayed if the 'y' value of its centroid would be less than a certain threshold.

This minimizes the detection of false contours as the floor of the mouth doesn't generate any contours when the tongue isn't present in the image, and even when the tongue is present, the small contours that are occasionally found are also eliminated. Furthermore, if the intraoral lighting is adjusted in such a way that even the regions 1C, 2C and 3C are lit up even when the tongue is absent from the image, it may also give rise to false detection of contours. This was also minimized by placing a condition based on the centroid of the contour such that the contour would only be displayed if the 'y' value of the centroid was above a certain value (which could be found by extending the tongue to the maximum in the vertical direction and finding out the corresponding 'y' value of the centroid). Nevertheless, if the LEDs are not placed correctly, there is still a possibility that a contour with larger area than the area of the tongue's contour (especially when it is placed in the backward categories (FR & BR)) would be detected. This would occasionally give false outputs.

Moreover, the HSV range must be selected carefully which would also take the lighting into consideration. Thus, any change in the position of the LEDs would also result in parts of the tongue not being included in the contour. Therefore, the system would have to be calibrated again by tuning the HSV range. Due to this error, even though the tongue would be present within a region of interest, the region of interest would occasionally be unable to detect its presence since the contour would not extend until the region (which is being analysed by each of the regions in every iteration of the code).

Furthermore, one of the limitations, especially during the users' training phase, is that the users have to constantly look at the screen in order to judge whether or not they are positioning their tongues correctly. This would be a disadvantage when the user is controlling hardware such as the robotic arm which would be away from the field of view of the user. Nevertheless, the users' dependency on the screen would eventually be minimized once they get used to the system.

5.6. Running the Code on Raspberry Pi

Since the code was written in Python, the advantage was that it could also be easily executed on a Raspberry Pi which has the added advantage of being small in size. This would be quite useful especially for mobile applications (such as controlling a wheelchair, etc.) in which case using a desktop PC would be impractical. Although the code was able to be executed on Raspberry Pi, a time delay of 3 seconds was obtained for each loop which would be too slow for practical usage. The system is summarized in the block diagram depicted in Figure 5.56.

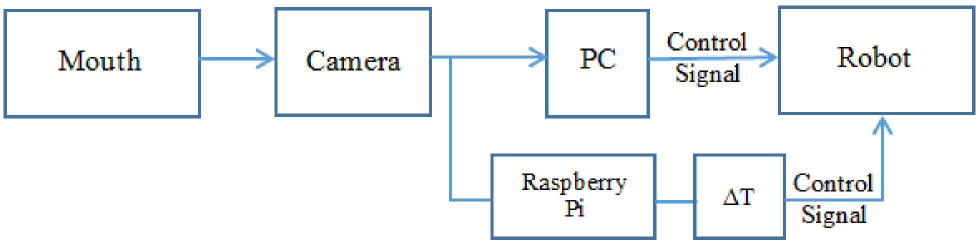


Figure 5.56: Block diagram summarizing the complete system.

Chapter 6. Conclusion and Future Work

In this thesis, a thorough research was conducted of the available methods used to assist people afflicted with disabilities to make use of alternative inputs. A novel method was proposed of using an intraoral camera to utilize the tongue as an input. Moreover, a prototype was built which included a hand-moulded mouthpiece, LEDs for lighting up the oral cavity and a cylindrical slot for the intraoral camera. An endoscope camera was selected for this prototype and image processing techniques were applied using the Python programming language with extensive use of the OpenCV library. Through real-time image processing, the orientation and position of the tongue were successfully evaluated based on which an output from the 10 different categories was then decided. Each category was successfully tested to ensure that the desired control signal was achieved. The control signal was then also connected to an application, namely a robotic arm. Each control signal was then used to control a particular servo motor of the robotic arm which would be extremely helpful for paralytic patients to fulfil their pick-and-place tasks.

A time delay of 3 seconds was received when the Raspberry Pi was used instead of the PC. However, it is worth noting that, in this thesis, the detection of the position and orientation of the tongue was successfully carried out with the help of pure image processing techniques, and not the popular object detection techniques such as YOLO (‘You Only Look Once’ real-time object detection technique) and R-CNN (Region-based Convolutional Neural Networks). The advantage of doing so is to save a lot of the processing time which would otherwise create a much greater lag in producing the outputs.

This thesis was successful in implementing a system for supporting assistive devices, and it was able to reduce or eliminate some of the drawbacks that are present in the current ATs that focus on using the tongue as an input. The systems known as Resistopalatography and the MouthPad possess the disadvantage of having a size constraint. However, with the emergence of miniature cameras, the actual implementation of inserting a camera within the intraoral environment would not be expected to take up much space. Furthermore, the insertion of magnetic sensors within the tongue has the disadvantage of being quite invasive. Contrary to that, the proposed system offers a non-invasive and thus more comfortable and feasible approach to capture inputs from the tongue. Lastly, the tongue-operated joystick is quite obtrusive and thus may not be favoured by most of the users. Although the proposed system doesn’t eliminate this drawback (as the wires are exposed to the outside environment which are connected to the PC), it is still possible to do so in the future by implementing a wireless system which could help establish the communication between the camera and the PC.

For future work, alternatives to the Raspberry Pi should be researched which could ensure that the entire image processing takes place on a single-board computer (SBC) along with minimal delay time. Moreover, the code can also be modified to minimize the processing time. Once the processing is done on SBCs, different applications (especially in mobile robots) can be connected to the system and tested. This would broaden the scope of this prototype. Furthermore, in order to control the robot arm, the most efficient technique would be to implement inverse kinematics, in which the user would simply move the end-effector point (consisting of the x, y and z value of the point) and the calculation of the angles for all of the motors would be computed automatically.

References

[1] (2017, May) Disability and functioning (noninstitutionalized adults aged 18 and over). [Online]. Available: https://www.cdc.gov/nchs/fastats/disability.htm [Accessed: Dec 17, 2018].

[2] Stats about paralysis. Christopher and Dana Reeve Foundation. [Online]. Available: https://www.christopherreeve.org/living-with-paralysis/stats-about-paralysis [Accessed: Dec 17, 2018].

[3] Quick facts about als and the als association. ALS Association. [Online]. Available: https://www.christopherreeve.org/living-with-paralysis/stats-about-paralysis [Accessed: Dec 17, 2018].

[4] (2017, Nov.) Paralysis. ALS Association. [Online]. Available: https://www.nhs.uk/conditions/paralysis/ [Accessed: Dec 17, 2018].

[5] What is at? [Online]. Available: https://www.atia.org/at-resources/what-is-at/ [Accessed: Dec 17, 2018].

[6] B. Yousefi *et al.*, "Quantitative and comparative assessment of learning in a tongue-operated computer input device—part ii: Navigation tasks," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 4, pp. 633–643, 2012.

[7] M. Suzuki, N. Yamamoto, O. Yamamoto, T. Nakano and S. Yamamoto, "Measurement of Driver's Consciousness by Image Processing -A Method for Presuming Driver's Drowsiness by Eye-Blinks coping with Individual Differences -," *2006 IEEE International Conference on Systems, Man and Cybernetics*, Taipei, 2006, pp. 2891-2896.

[8] M. S. Reddy, A. Sammaiah, B. Narsimha and K. S. Rao, "Analysis of EOG Signals Using Empirical Mode Decomposition for Eye Blink Detection," *2011 International Conference on Multimedia and Signal Processing*, Guilin, Guangxi, 2011, pp. 293-297.

[9] Y. Kim, "Detection of Eye Blinking Using Doppler Sensor With Principal Component Analysis," in *IEEE Antennas and Wireless Propagation Letters*, vol. 14, pp. 123-126, 2015.

[10] Y. Cheung and Q. Peng, "Eye Gaze Tracking With a Web Camera in a Desktop Environment," in *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 4, pp. 419-430, Aug. 2015.

[11] H. S. Dhillon, R. Singla, N. S. Rekhi and R. Jha, "EOG and EMG based virtual keyboard: A brain-computer interface," *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, 2009, pp. 259-262.

[12] A. Castillo *et al*., "Hands free mouse," *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN),* San Francisco, CA, 2016, pp. 109-114.

[13] C. Ishii and R. Konishi, "A Control of Electric Wheelchair Using an EMG Based on Degree of Muscular Activity," *2016 Euromicro Conference on Digital System Design (DSD),* Limassol, 2016, pp. 567-574.

[14] S. Hawking. My computer. [Online]. Available: http://www.hawking.org.uk/the-computer.html [Accessed: Dec 17, 2018].

[15] Hyuk Jeong, Jong-Sung Kim and Wook-Ho Son, "An EMG-based Mouse Controller for A Tetraplegic," *2005 IEEE International Conference on Systems, Man and Cybernetics*, Waikoloa, HI, 2005, pp. 1229-1234.

[16] T. Simpson, C. Broughton, M. J. A. Gauthier and A. Prochazka, "Tooth-Click Control of a Hands-Free Computer Interface," in *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 8, pp. 2050-2056, Aug. 2008.

[17] (2015, Aug.) Turning breath into words new device unveiled to give paralysis sufferers a voice. [Online]. Available: https://www.lboro.ac.uk/news-events/news/2015/august/turning-breath-into-words.html [Accessed: Dec 17, 2018].

[18] (2012, Dec.). [Online]. Available: http://atwiki.assistivetech.net/images/3/32/Shepherd-center-tube.jpg [Accessed: Dec 17, 2018].

[19] B. Rebsamen *et al*., "Controlling a Wheelchair Indoors Using Thought," in *IEEE Intelligent Systems*, vol. 22, no. 2, pp. 18-24, March-April 2007.

[20] E.W. Sellers, D. J. Krusienski, D. J. McFarland, T. M. Vaughan and J. R. Wolpaw, “A p300 event-related potential brain–computer interface (bci): the effects of matrix size and inter stimulus interval on performance,” *Biological psychology*, vol. 73, no. 3, pp. 242–252, 2006.

[21] R. Horne, S. Kelly and P. Sharp, "Resistopalatography as an assistive technology for users with spinal cord injuries," *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Milan, 2015, pp. 4367-4370.

[22] O. Draghici, I. Batkin, M. Bolic and I. Chapman, "Performance evaluation of the MouthPad," *2014 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, Lisboa, 2014, pp. 1-5.

[23] G. Krishnamurthy and M. Ghovanloo, "Tongue drive: a tongue operated magnetic sensor based wireless assistive technology for people with severe disabilities," *2006 IEEE International Symposium on Circuits and Systems*, Island of Kos, 2006, pp. 5551–5554.

[24] X. Huo and M. Ghovanloo*, "Using Unconstrained Tongue Motion as an Alternative Control Mechanism for Wheeled Mobility," in *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 6, pp. 1719-1726, June 2009.

[25] B. Yousefi, X. Huo, E. Veledar and M. Ghovanloo, "Quantitative and Comparative Assessment of Learning in a Tongue-Operated Computer Input Device," in *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 5, pp. 747-757, Sept. 2011.

[26] M. N. Sahadat, A. Alreja, N. Mikail and M. Ghovanloo, "Comparing the Use of Single Versus Multiple Combined Abilities in Conducting Complex Computer Tasks Hands-Free," in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 9, pp. 1868-1877, Sept. 2018.

[27] M. N. Sahadat, S. Dighe, F. Islam and M. Ghovanloo, "An Independent Tongue-Operated Assistive System for Both Access and Mobility," in *IEEE Sensors Journal*, vol. 18, no. 22, pp. 9401-9409, 15 Nov.15, 2018.

[28] T. Ohba and S. Kajikawa, "Tongue-operated joystick device with reaction force feedback mechanism," *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Munich, 2017, pp. 207-212.

[29] Image thresholding. [Online]. Available: https://docs.opencv.org/3.0-beta/doc/py tutorials/py imgproc/py thresholding/py thresholding.html [Accessed: Dec 17, 2018].

[30] Morphological transformations. [Online]. Available: https://docs.opencv.org/3.4/ d9/d61/tutorial py morphological ops.html [Accessed: Dec 17, 2018].

[31] X. Huo, H. Park, J. Kim and M. Ghovanloo, "A Dual-Mode Human Computer Interface Combining Speech and Tongue Motion for People with Severe Disabilities," in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 21, no. 6, pp. 979-991, Nov. 2013.

Appendix: Python Code

The following code was written using the Python programming language. However, the presented code just detects 8 categories and is without the calibration system for automatically detecting the thresholds for Cx.

```
import cv2
import numpy as np
import imutils
from matplotlib.widgets import RectangleSelector
import matplotlib.pyplot as plt
import time
import serial
import keyboard #Using module keyboard

def nothing(x):
   pass

cap = cv2.VideoCapture(0)

# Create window for Trackbars
cv2.namedWindow('Threshed_HSV')
cv2.namedWindow('Threshed_HSV_Teeth')

# Create trackbars for threshold change - to detect the tongue
cv2.createTrackbar('H_upper','Threshed_HSV',255,255,nothing)    #190
cv2.createTrackbar('S_upper','Threshed_HSV',61,255,nothing)     #52
cv2.createTrackbar('V_upper','Threshed_HSV',255,255,nothing)    #255

cv2.createTrackbar('H_Lower','Threshed_HSV',101,255,nothing)    #128
cv2.createTrackbar('S_Lower','Threshed_HSV',0,255,nothing)      #0
cv2.createTrackbar('V_Lower','Threshed_HSV',140,255,nothing)    #0

# create trackbars for threshold change - to detect the teeth
cv2.createTrackbar('H_upper_Teeth','Threshed_HSV_Teeth',112,255,nothing)    #190
cv2.createTrackbar('S_upper_Teeth','Threshed_HSV_Teeth',255,255,nothing)     #52
cv2.createTrackbar('V_upper_Teeth','Threshed_HSV_Teeth',255,255,nothing)    #255

cv2.createTrackbar('H_Lower_Teeth','Threshed_HSV_Teeth',13,255,nothing)    #128
cv2.createTrackbar('S_Lower_Teeth','Threshed_HSV_Teeth',0,255,nothing)      #0
cv2.createTrackbar('V_Lower_Teeth','Threshed_HSV_Teeth',0,255,nothing)    #0

#Initializing variables which will be the commands sent to the Servo motors of the Robot
motor1 = 1500
motor2 = 1500
motor3 = 1500
motor4 = 1500
motor5 = 1500

#Establishing Serial communication for communication with the robot
ser = serial.Serial(
   port='COM7',
    baudrate=9600,
    bytesize=serial.EIGHTBITS,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE)

if(ser.isOpen() == False):
    ser.open()

ser.write(b'#0P1500\r')
ser.write(b'#1P1500\r')
ser.write(b'#2P1500\r')
ser.write(b'#3P1500\r')
ser.write(b'#4P1500\r')

#The following loop will run continuously
while True:
    _, img = cap.read()

    Verdict = 'No Input' #The text that is seen on the final image defining the category

    clone = img.copy()
    clone_black = img.copy()
    clone_black[:,:,:] = [0,0,0]

    # Blur and HSV Conversion
    blurred_frame = cv2.GaussianBlur(clone, (5,5),0)
    hsv = cv2.cvtColor(blurred_frame, cv2.COLOR_BGR2HSV)

    # Get current positions of six trackbars (for the tongue)
    H_Up = cv2.getTrackbarPos('H_upper','Threshed_HSV')
    H_Down = cv2.getTrackbarPos('H_Lower','Threshed_HSV')

    S_Up = cv2.getTrackbarPos('S_upper','Threshed_HSV')
    S_Down = cv2.getTrackbarPos('S_Lower','Threshed_HSV')

    V_Up = cv2.getTrackbarPos('V_upper','Threshed_HSV')
    V_Down = cv2.getTrackbarPos('V_Lower','Threshed_HSV')

    upper = np.array([H_Up,S_Up,V_Up])
    lower = np.array([H_Down,S_Down,V_Down])

    Threshold = 127 #for the binarization - however sine OTSU was used, this variable was not
useful

    #TEETH: get current positions of six trackbars
    H_Up_Teeth = cv2.getTrackbarPos('H_upper_Teeth','Threshed_HSV_Teeth')
    H_Down_Teeth = cv2.getTrackbarPos('H_Lower_Teeth','Threshed_HSV_Teeth')

    S_Up_Teeth = cv2.getTrackbarPos('S_upper_Teeth','Threshed_HSV_Teeth')
    S_Down_Teeth = cv2.getTrackbarPos('S_Lower_Teeth','Threshed_HSV_Teeth')

    V_Up_Teeth = cv2.getTrackbarPos('V_upper_Teeth','Threshed_HSV_Teeth')
    V_Down_Teeth = cv2.getTrackbarPos('V_Lower_Teeth','Threshed_HSV_Teeth')
    upper_for_teeth = np.array([H_Up_Teeth,S_Up_Teeth,V_Up_Teeth])
    lower_for_teeth = np.array([H_Down_Teeth,S_Down_Teeth,V_Down_Teeth])

    # Isolation of the image of the tongue through the HSV range
    mask = cv2.inRange(hsv, lower, upper)
    res = cv2.bitwise_and(clone,clone, mask= mask)

    # Isolation of the teeth through the HSV range
    mask_teeth = cv2.inRange(hsv, lower_for_teeth, upper_for_teeth)
    res_teeth = cv2.bitwise_and(clone,clone, mask= mask_teeth)

    # Conversion to Grayscale
    frame_gray = cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)

    # Blurring
    frame_gray = cv2.GaussianBlur(frame_gray,(11,1),0)

    # Binarization
    ret2,th2                                                                =
cv2.threshold(frame_gray,Threshold,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    # 'Closing' Operation
    kernel = np.ones((7,7),np.uint8)
    th2 = cv2.morphologyEx(th2, cv2.MORPH_CLOSE, kernel)
    th2_teeth = cv2.morphologyEx(mask_teeth, cv2.MORPH_CLOSE, kernel)

    # Initialization of the variables for the regions of Interest
    roi1c_val = 0
    roi2c_val = 0
    roi3c_val = 0
    roic_total = 0
    roiRight_val_teeth = 0
    roiLeft_val_teeth = 0

    # TEETH: Defining the regions of interest and Calculating the white pixels in them
    # a)Left
    roi_left_teeth = th2_teeth[79:480, 0:100]
    n_white_pix1_teeth = np.sum(roi_left_teeth == 255)
    print('Teeth - Number of white pixels - Left:', n_white_pix1_teeth)
    if (n_white_pix1_teeth<15000):
        cv2.rectangle(res_teeth, (0, 79), (100, 480), (255,0,0), 1)
        roiLeft_val_teeth = 0
    elif (n_white_pix1_teeth>15000):
        cv2.rectangle(res_teeth, (0, 79), (100, 480), (0,0,255), 3)
        roiLeft_val_teeth = 1

    # b)Right
    roi_right_teeth = th2_teeth[79:480, 540:640]
    n_white_pix2_teeth = np.sum(roi_right_teeth == 255)
    print('Teeth - Number of white pixels - Right:', n_white_pix2_teeth)
    if (n_white_pix2_teeth<15000):
        cv2.rectangle(res_teeth, (540, 79), (640, 480), (255,0,0), 1)
        roiRight_val_teeth = 0
    elif (n_white_pix2_teeth>15000):
        cv2.rectangle(res_teeth, (540, 79), (640, 480), (0,0,255), 3)
        roiRight_val_teeth = 1

    ## TONGUE: Defining the regions of interest and Calculating the white pixels in them
    # a) Left
    roi1c = th2[0:78, 0:214]
    n_white_pix1c = np.sum(roi1c == 255)
    print('Number of white pixels 1c:', n_white_pix1c)
    if (n_white_pix1c>10000):
        cv2.rectangle(clone, (0, 0), (214, 78), (0,0,255), 3)
        roi1c_val = 2 #Assign a higher value if more pixels are present (when the tongue is
turned)
    elif (n_white_pix1c>3000 and n_white_pix1c<10000):
        cv2.rectangle(clone, (0, 0), (214, 78), (0,0,255), 3)
        roi1c_val = 1 #The tongue is simply present
    elif (n_white_pix1c<2000):
        cv2.rectangle(clone, (0, 0), (214, 78), (255,0,0), 1)
        roi1c_val = 0

    # b) Middle
    roi2c = th2[0:78, 215:427]
    n_white_pix2c = np.sum(roi2c == 255)
    print('Number of white pixels 2c:', n_white_pix2c)
    if (n_white_pix2c<2000):
        cv2.rectangle(clone, (215, 0), (427, 78), (255,0,0), 1)
        roi2c_val = 0
    elif (n_white_pix2c>2000):
        cv2.rectangle(clone, (215, 0), (427, 78), (0,0,255), 3)
        roi2c_val = 1

    # c) Right
    roi3c = th2[0:78, 428:640]
    n_white_pix3c = np.sum(roi3c == 255)
    print('Number of white pixels 3c:', n_white_pix3c)
    if (n_white_pix3c>4000):
        cv2.rectangle(clone, (428, 0), (640, 78), (0,0,255), 3)
        roi3c_val = 2
    elif (n_white_pix3c>2000 and n_white_pix3c<4000):
        cv2.rectangle(clone, (428, 0), (640, 78), (0,0,255), 3)
        roi3c_val = 1
    elif (n_white_pix3c<2000):
        cv2.rectangle(clone, (428, 0), (640, 78), (255,0,0), 1)
        roi3c_val = 0

    # Calculation of the total value
    roic_total = roi1c_val + roi2c_val + roi3c_val

    # Finding the contours
    _,          contours,          _=          cv2.findContours(th2,          cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    contour_list = []

    # Initialization for the centroid variables
```

```
cx=250
cy=250

for contour in contours:
    area = cv2.contourArea(contour)
    if (roic_total < 3):
        break
    if ((len(contour) > 8) and (area > 10000)):
        M = cv2.moments(contour)
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        if(cy<250):
            print('cx=',cx)
            print('cy=',cy)
            cv2.circle(clone,(cx,cy), 10, (0,255,0), -1)
            ellipse = cv2.fitEllipse(contour)


## Checking for Rest of the 9 Regions of Interest
#ROI = 1
roi1 = clone_black[79:212, 0:214]
n_white_pix1 = np.sum(roi1 == 255)
print('Number of white pixels 1:', n_white_pix1)
if (n_white_pix1<500):
    cv2.rectangle(clone, (0, 79), (214, 212), (255,0,0), 1)
    roi1_val = 0
elif (n_white_pix1>500):
    cv2.rectangle(clone, (0, 79), (214, 212), (0,0,255), 3)
    roi1_val = 1

#ROI = 2
roi2 = clone_black[79:212, 215:427]
n_white_pix2 = np.sum(roi2 == 255)
print('Number of white pixels 2:', n_white_pix2)
if (n_white_pix2<500):
    cv2.rectangle(clone, (215, 79), (427, 212), (255,0,0), 1)
    roi2_val = 0
elif (n_white_pix2>500):
    cv2.rectangle(clone, (215, 79), (427, 212), (0,0,255), 3)
    roi2_val = 1

#ROI = 3
roi3 = clone_black[79:212, 428:640]
n_white_pix3 = np.sum(roi3 == 255)
print('Number of white pixels 3:', n_white_pix3)
if (n_white_pix3<500):
    cv2.rectangle(clone, (428, 79), (640, 212), (255,0,0), 1)
    roi3_val = 0
elif (n_white_pix3>500):
    cv2.rectangle(clone, (428, 79), (640, 212), (0,0,255), 3)
    roi3_val = 1

#ROI = 4
roi4 = clone_black[213:346, 0:214]
```

```
n_white_pix4 = np.sum(roi4 == 255)
print('Number of white pixels 4:', n_white_pix4)
if (n_white_pix4<500):
    cv2.rectangle(clone, (0, 213), (214, 346), (255,0,0), 1)
    roi4_val = 0
elif (n_white_pix4>500):
    cv2.rectangle(clone, (0, 213), (214, 346), (0,0,255), 3)
    roi4_val = 1

#ROI = 5
roi5 = clone_black[213:346, 215:427]
n_white_pix5 = np.sum(roi5 == 255)
print('Number of white pixels 5:', n_white_pix5)
if (n_white_pix5<500):
    cv2.rectangle(clone, (215, 213), (427, 346), (255,0,0), 1)
    roi5_val = 0
elif (n_white_pix5>500):
    cv2.rectangle(clone, (215, 213), (427, 346), (0,0,255), 3)
    roi5_val = 1

#ROI = 6
roi6 = clone_black[213:346, 428:640]
n_white_pix6 = np.sum(roi6 == 255)
print('Number of white pixels 6:', n_white_pix6)
if (n_white_pix6<500):
    cv2.rectangle(clone, (428, 213), (640, 346), (255,0,0), 1)
    roi6_val = 0
elif (n_white_pix6>500):
    cv2.rectangle(clone, (428, 213), (640, 346), (0,0,255), 3)
    roi6_val = 1

#ROI = 7
roi7 = clone_black[347:480, 0:214]
n_white_pix7 = np.sum(roi7 == 255)
print('Number of white pixels 7:', n_white_pix7)
if (n_white_pix7<500):
    cv2.rectangle(clone, (0, 347), (214, 480), (255,0,0), 1)
    roi7_val = 0
elif (n_white_pix7>500):
    cv2.rectangle(clone, (0, 347), (214, 480), (0,0,255), 3)
    roi7_val = 1

#ROI = 8
roi8 = clone_black[347:480, 215:427]
n_white_pix8 = np.sum(roi8 == 255)
print('Number of white pixels 8:', n_white_pix8)
if (n_white_pix8<500):
    cv2.rectangle(clone, (215, 347), (427, 480), (255,0,0), 1)
    roi8_val = 0
elif (n_white_pix8>500):
    cv2.rectangle(clone, (215, 347), (427, 480), (0,0,255), 3)
    roi8_val = 1

#ROI = 9
```

```
    roi9 = clone_black[347:480, 428:640]
    n_white_pix9 = np.sum(roi9 == 255)
    print('Number of white pixels 9:', n_white_pix9)
    if (n_white_pix9<500):
        cv2.rectangle(clone, (428, 347), (640, 480), (255,0,0), 1)
        roi9_val = 0
    elif (n_white_pix9>500):
        cv2.rectangle(clone, (428, 347), (640, 480), (0,0,255), 3)
        roi9_val = 1

    # Displaying the values of the centroid of the contour
    print('Global cx=',cx)
    print('Global cy=',cy)

    ## Checking for the Categories and Moving the Robot Accordingly
    if(cx>379):
        #FR
        if((roi9_val==1)and(roi7_val==0)):
            motor1 = motor1 + 10
            string1 = "#0P" + str(motor1) + "\r"
            ser.write(string1.encode("utf-8"))
            Verdict = 'Forward + Right'
        #R
        elif((roi9_val==0)and(roi6_val==1)and(roi7_val==0)):
            motor1 = motor1 + 20
            string1 = "#0P" + str(motor1) + "\r"
            ser.write(string1.encode("utf-8"))
            Verdict = 'Right'
        #BR

elif((roi3c_val>0)and(roi1_val==0)and(roi3_val==1)and(roi6_val==0)and(roi9_val==0)):
            Verdict = 'Backwards + Right'

    if(cx<280):
        #FL
        if((roi7_val==1)and(roi9_val==0)):
            motor1 = motor1 - 10
            string1 = "#0P" + str(motor1) + "\r"
            ser.write(string1.encode("utf-8"))
            Verdict = 'Forward + Left'
        #L
        elif((roi7_val==0)and(roi4_val==1)and(roi9_val==0)):
            motor1 = motor1 - 20
            string1 = "#0P" + str(motor1) + "\r"
            ser.write(string1.encode("utf-8"))
            Verdict = 'Left'
        #BL

elif((roi1c_val>0)and(roi1_val==1)and(roi3_val==0)and(roi4_val==0)and(roi7_val==0)):
            Verdict = 'Backwards + Left'

    ## Forwards & Backwards if Teeth are NOT Visible
    if( (roiRight_val_teeth==0) and (roiLeft_val_teeth == 0)):
        #F2
        if (roi7_val==roi8_val==roi9_val==1):
            motor2 = motor2 - 10
            string2 = "#1P" + str(motor2) + "\r"
            ser.write(string2.encode("utf-8"))
            Verdict = 'Forwards (No Teeth)'
        #B2
        if ((roi1_val==roi2_val==roi3_val==1) and (roi4_val==roi6_val==0) ):# and
(roi9_val==0):
            motor2 = motor2 + 10
            string2 = "#1P" + str(motor2) + "\r"
            ser.write(string2.encode("utf-8"))
            Verdict = 'Backwards (No Teeth)'

    ## Forwards & Backwards if Teeth are Visible
    if( (roiRight_val_teeth==1) or (roiLeft_val_teeth == 1)):
        #F1
        if ((roi1_val==roi2_val==roi3_val==1)and((roi4_val==roi6_val==1))):
            motor3 = motor3 + 10
            string3 = "#2P" + str(motor3) + "\r"
            ser.write(string3.encode("utf-8"))
            Verdict = 'Forwards (With Teeth)'
        #B1
        if ((roi1_val==roi2_val==roi3_val==1) and (roi4_val==roi6_val==0) ):# and
(roi9_val==0):
            motor3 = motor3 - 10
            string3 = "#2P" + str(motor3) + "\r"
            ser.write(string3.encode("utf-8"))
            Verdict = 'Backwards (With Teeth)'

    ## Motor 4 - Pick and Drop
    if keyboard.is_pressed('p'):#if key 'p' is pressed
        string4 = "#4P" + "2000" + "\r"
        ser.write(string4.encode("utf-8"))
    if keyboard.is_pressed('d'):#if key 'd' is pressed
        string4 = "#4P" + "700" + "\r"
        ser.write(string4.encode("utf-8"))


    #Preparing the Text to be Inserted on the Final Image
    font                   = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (10,450)
    fontScale              = 1
    fontColor              = (0,255,255)
    lineType               = 3

    cv2.putText(clone,Verdict,
    bottomLeftCornerOfText,
    font,
    fontScale,
    fontColor,
    lineType)

    ## Displaying the results
    cv2.namedWindow("1. Original Image",cv2.WINDOW_NORMAL)
    cv2.resizeWindow('1. Original Image', 512,384)
    cv2.moveWindow("1. Original Image", 0,0)
    cv2.imshow('1. Original Image', img)

    cv2.namedWindow("2. Isolating the Tongue Based on the HSV
Range",cv2.WINDOW_NORMAL)
    cv2.resizeWindow('2. Isolating the Tongue Based on the HSV Range', 512,384)
    cv2.moveWindow("2. Isolating the Tongue Based on the HSV Range", 530,0)
    cv2.imshow("2. Isolating the Tongue Based on the HSV Range", res)

##        cv2.namedWindow("3. Isolating the Teeth Based on the HSV
Range",cv2.WINDOW_NORMAL)
##    cv2.resizeWindow('3. Isolating the Teeth Based on the HSV Range', 512,384)
##    cv2.imshow("3. Isolating the Teeth Based on the HSV Range", mask_teeth)

    cv2.namedWindow("3. Grayscale Image After Smoothing",cv2.WINDOW_NORMAL)
    cv2.resizeWindow('3. Grayscale Image After Smoothing', 512,384)
    cv2.moveWindow("3. Grayscale Image After Smoothing", 1060,0)
    cv2.imshow("3. Grayscale Image After Smoothing", frame_gray)

    cv2.namedWindow("4. Binarization of the Image After Applying the (Closing)
Morphological Transform",cv2.WINDOW_NORMAL)
    cv2.resizeWindow('4. Binarization of the Image After Applying the (Closing)
Morphological Transform', 512,384)
    cv2.moveWindow("4. Binarization of the Image After Applying the (Closing)
Morphological Transform", 0,425)
    cv2.imshow("4. Binarization of the Image After Applying the (Closing) Morphological
Transform", th2)

    cv2.namedWindow("5. Final Binarized Image With Contour",cv2.WINDOW_NORMAL)
    cv2.resizeWindow('5. Final Binarized Image With Contour', 512,384)
    cv2.moveWindow("5. Final Binarized Image With Contour", 530,425)
    cv2.imshow("5. Final Binarized Image With Contour", clone_black)

    cv2.namedWindow("6. Obtaining the Contour & Location of the
Tongue",cv2.WINDOW_NORMAL)
    cv2.resizeWindow('6. Obtaining the Contour & Location of the Tongue', 512,384)
    cv2.moveWindow("6. Obtaining the Contour & Location of the Tongue", 1060,425)
    cv2.imshow("6. Obtaining the Contour & Location of the Tongue", clone)

    # ESC to break
    k = cv2.waitKey(1) & 0xFF
    if k == 27:
        break

# close all open windows
cap.release()
cv2.destroyAllWindows().
```

Vita

Muhammad Amin Tily was born in 1991, in Abu Dhabi, United Arab Emirates. He received his primary and secondary education in Sharjah, UAE. He received his B.Sc. degree in Electrical Engineering from the American University of Sharjah in 2013. In September 2016, he joined the Mechatronics Engineering master's program in the American University of Sharjah where he spent two years as a graduate teaching assistant. His research interests are in image processing, machine learning and Biomedical Engineering.