Lip Reading to Text

Mogammat Waleed Deaney

Project presented in partial fulfillment
of the requirements for the degree of
Bachelor of Science (Honours)
at the University of the Western Cape

Supervisor: Prof. IM Venter
Co-supervisor: Mr M Ghaziasgar
Mentor: Kenzo Abrahams
Co-mentor: Nathan de la Cruz

28 March 2014

Declaration

I, Mogammat Waleed Deaney, declare that this project "*Lip Reading to Text*" is my own work, that it has not been submitted before for any degree or assessment at any other university, and that all the sources I have used or quoted have been indicated and acknowledged by means of complete references.

Signature: ..... Date: .....

Mogammat Waleed Deaney.

Abstract

Lipreading also known as speechreading, is a visual way of "listening" to someone. This is done by looking at the speakers face to follow their speech patterns in order to recognise what is being said. This project will attempt to develop a computerized lip-reader. The system will observe the user's lip-movement through a sequence of frames and convert the recognised sound/word to text which will be displayed on a screen. The project will regard the mouth as the region of interest for lip reading. All other features such as hand and face gestures is considered to be beyond the scope of the project. The subject will be asked to articulate certain sounds or letters which need to be correctly identified by the system.

Key words

Visual Speech Recognition

Lip Reading

Local Binary Patterns

Speech Reading

Support Vector Machine

Haar-Like Features

Lip Segmentation

Acknowledgments

I would like to thank my supervisors Prof. IM Venter and Mr M Ghaziasgar for their support and encouragement during my Honours year. Without our weekly meetings, this work would not have been possible. Without their help I would certainly not be where I am today.

Contents

List of Tables

List of Figures

Glossary

GUI	Graphical User Interface
LBP	Local Binary Patterns
ROI	Region of Interest
SVM	Support Vector Machine

Chapter 1

Background

Speech recognition is not purely auditory. A speaker produces visual information which can be used by the listener to interpret what the speaker is saying. This is considered to be part of the speech recognition process. The first reports of visual information being used for speech recognition were contributed by McGurck and MacDonald [McGurk and MacDonald, 1976; Barnard et al., 2010].

This project is an attempt to recognise lip movement as speech and display the result as text. It entails developing an application that uses a web camera, which will be pointed at a person's face, to lip read that person. This will involve recognizing simple vowels or sounds at first. It can later be extended to words and sentences, time permitting. The application will then take the recognized vowel, word or sentence and convert it to text to be displayed on the screen.

Chapter 2

User Requirements

2.1 Introduction

This section describes the problem from the user's point of view. It states the problem domain and the functionality of the program.

2.2 Users view of problem

Lip reading is the process of understanding speech by visually interpreting movement of the mouth. The user needs a system that will computerize the process of lip reading.

2.3 Description of the problem

A system is required that can automatically lip read letters, sounds or words without input from the user. The objective is to create an automated lip reader that will determine what the subject has said and display it on the screen.

Speech recognition translates spoken words into text. It is difficult for the software to identify a particular speaker when another person is speaking at the same time. Another disadvantage is that it does not work well in noisy environments. The sound of particular words can sometimes be similar for e.g "one" and "won". The words can be mistaken for one another as they sound the same but have distinctly different meaning and lip movements.

2.4 Expectations of the software solution

The software should be able to detect the subject's mouth-movements using a web camera and be able to recognise simple sounds or letters articulated by the subject. The program should display the recognised sound or letter as

text to the user. It should also allow the user to stop and start the program at any given time.

2.5 Not expected of the software solution

The software will not track multiple subjects and is not expected to monitor subjects who are not directly facing the camera. The subject being lip read is required to be as still as possible. The camera should have a clear uninhibited view of the user's face. The lip reader application will not do real-time lip reading.

2.6 Conclusion

The above section describes the user requirements of the system that will allow speech recognition visually. It also describes the scope of the project. In Chapter 3 the designers interpretation of the problem will identify the most beneficial means for the system's implementation.

Chapter 3

Requirements analysis design

3.1 Introduction

In the previous chapter, the user specified the need for a visual speech recognition system. This chapter will explain how the designer will deal with the user's requirements. This is done by breaking down the problem into high level constituent parts and identifying all the relevant details.

3.2 Interpretation and breakdown of the problem

The input of the application will be a live video stream of the subject speaking. The subject will have to face the camera directly and not move his/her head around while the program is running. When the subject speaks he/she will have to articulate well. The web-cam will capture the change of mouth movements when the subject speaks.

In order to accurately identify the change in mouth movements, preprocessing of the the video snippet will have to be done. Some of the preprocessing will be done with OpenCV. OpenCV is an open source computer vision and machine learning software library which allows for a simple-to-use computer vision infrastructure which has over 500 different functions available [Bradski and Kaehler, 2008].

3.3 Complete analysis of the problem

3.3.1 Face and mouth detection

The application will accept video input from a webcam frame by frame. For this project, the mouth is considered to be the region of interest. In order to detect the mouth, the subjects face needs to be located. This is done by implementing OpenCV's face detection library. It uses the Haar-classifier, a tree-

based technique used to detect rigid objects [Bradski and Kaehler, 2008]. The core basis for Haar-classifier object detection is the Haar-like features [Wilson and Fernandez, 2006]. Haar-like features are a contrast in values across a rectangular area of pixels showing both light and dark areas. Both the face and mouth have Haar-like features and therefore can be detected using this algorithm.(see Figure 3.1)

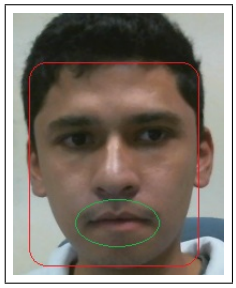


Figure 3.1: Locating the face and mouth

3.3.2 Pre-processing frames

Once the mouth is detected it is segmented from the rest of the frame. The image of the mouth is then grey-scaled in preparation for the next operation. Next the Local Binary Pattern (LBP) operator will be applied on the image. LBP as an image operator is used to extract LBP histogram (LBPH) features for texture description [Yang and Ai, 2007]. This will show the valleys and ridges of the image and could be useful in extracting features from the mouth.

3.3.3 Categorizing frames

A Support Vector Machine (SVM) will be used to categorize the frames. The SVM will be fed images to train and test the system. The frames will be classified as a mouth movement which can be matched to speech. A basic SVM takes a set of input data and predicts for each given input which of the two possible classes forms the output, making it a non-probabilistic binary linear classifier. The decision is made by solving a linearly constrained quadratic programming problem [Osuna et al., 1997].

3.3.4 Displaying the speech as text

The system will allow the user to see what the subject has said on the screen. The result will be displayed as soon as the user finishes the sound/letter.

3.4 Other similar systems

People inherit the process of lip reading subconsciously by interacting with others from a young age, although a skilled lip reader cannot be a 100% accurate as lip reading is not a precise art.

H. Mehrotra, G. Agrawal and M.C. Srivastava created a lip reading system which proposed a solution based on automatic lip contour tracking. Their aim was to recognize characters of the English language after a lip contour is accurately detected on each frame. The system made use of the k-nearest neighbour machine learning technique and was trained on five different speakers with five characters each A,E,I,O and U. Results varied between a 33-73% accuracy proportional to the value of k. [Mehrotra et al., 2009]

3.5 Suggested solution

The solution will detect mouth movement and should be able to recognise simple letters. The solution could later be extended to capture words and sentences leading to a real-time lip reader. The tools needed for the implementation are a web camera and a computer with OpenCV installed.

3.6 Conclusion

This section states how the designer interpreted the problem. A generic framework for a solution was given in order to solve how the “lip reader” would proceed in visually recognising speech. The next chapter will describe how the results will be displayed to the user.

Chapter 4

User interface specification

4.1 Introduction

This chapter will describe the user interface and will explain how the user interacts with the program. This includes displaying how the user can manipulate the system and in turn display the effects of the manipulation.

4.2 Description of the user interface

The application will have a rudimentary Graphical User Interface (GUI), since the application is a prototype. The interface will be minimal, yet simple and intuitive to use. A window containing the video stream of the subject will be created and displayed to the user.

4.2.1 Input video window

The system will start with a video feed of the subject which is displayed inside a window. The window will open only when the system detects the subject's mouth (See Figure 4.1).

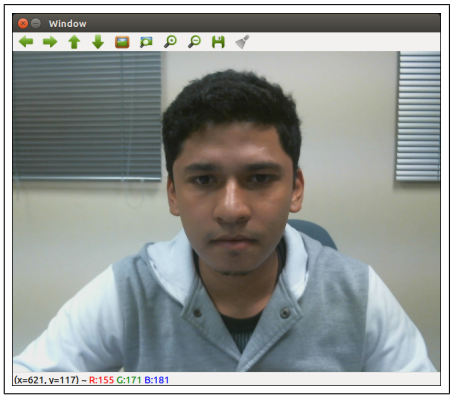


Figure 4.1: Window displaying subject

4.3 How the user interface behaves

The user will interact with the system with mouse clicks, each click will create a new window displaying the next stage of the application. The user will click on the input video window once the subject is ready to speak starting the process as shown in Figure 4.2. Two new windows will subsequently open, one displaying the subjects mouth and the other an image of the local binary patterns. Once the computations are complete a new window will open displaying the result to the user.

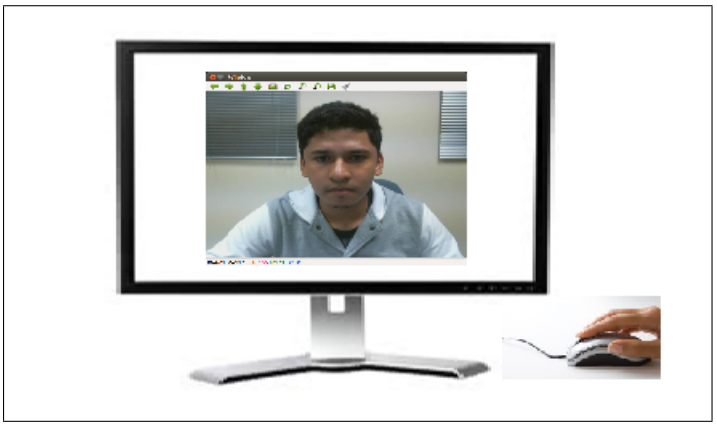


Figure 4.2: User interaction

4.4 Conclusion

This chapter discusses how the user interacts with the system. The user interface is minimal and uses windows to display the output. The aim of the project is to provide results for research purposes, thus less focus is placed on the interface. The next chapter will discuss the high level design of the software.

Chapter 5

High level design

5.1 Introduction

This chapter describes the High Level Design (HLD) of the software solution. This perspective shows a high level abstraction of the automated lip reader to analyse the methodology in the construction of the system. The programming language of choice is C/C++.

5.2 Breakdown of technical solution in subsystems

The technical solution consists of four phases which include input from the video camera, pre-processing, classification and the recognition of the letter/-sound (output). These phases form the technical solution of the automated lip reader system. A breakdown of these phases are necessary to form high level constituent parts of the system. Figure 5.1 illustrates the breakdown of the technical solution into subsystems namely, video frames, segmentation of the lip region (Region of Interest), feature extraction, support vector machine, recognition of the letter/sound.

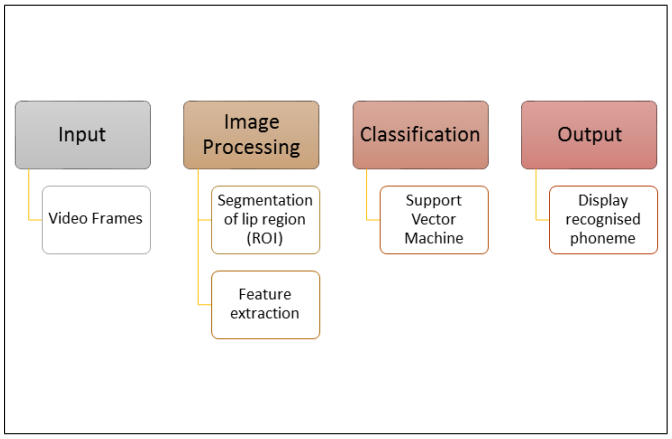


Figure 5.1: Breakdown of technical solution to HLD

5.3 Description of subsystems and operations

Table 5.1 represents important components including and relating to the subsystems and their operations.

Table 5.1: Description of Components

COMPONENT	DESCRIPTION
OpenCV	OpenCV is an open source computer vision library. It provides functions and data structures that allow for the processing of images[Bradski and Kaehler, 2008].
Haar-like Features	Features used in object detection. The idea was extended by Viola and Jones. It is capable of processing images extremely rapidly and achieving high detection rates[Viola and Jones, 2001].
Region of Interest (ROI)	An area which is segmented from the original image to form the ROI. The ROI is isolated from the original image to be further processed upon. The system regards the mouth area of the subject as the ROI.
Feature Extraction	A manipulation of an image to represent interesting parts as a compact feature vector. An example of feature extraction is the Local Binary Pattern operator.
Local Binary Pattern (LBP)	The local binary pattern is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number[Matti Pietikinen, 2011].
Histogram	A representation of the colour distribution within an image. This representation is sent to the SVM for classification.
Classification	A process in which data is grouped according to specific parameters. The process is completed with the use of a SVM.
Support Vector Machine (SVM)	A SVM is used to recognize patterns associated with the intensity of pixels. Data is sent to the SVM and is used to classify which class the image pixels belong. The extracted histogram data is sent to the SVM for training and testing the system.

5.4 Interaction between subsystems

Figure 5.2 represents the relationship between subsystems. Each subsequent process can begin only if its predecessor is complete. The interaction begins

once the user sits in front of the camera. This will provide input frames for the system. Once the subject has been detected and is ready to speak, the system will start. The lips are segmented from the frames providing a ROI and the features are extracted. The extracted features are sent to the SVM for classification. The final step involves displaying the result to the user as a sound/letter.

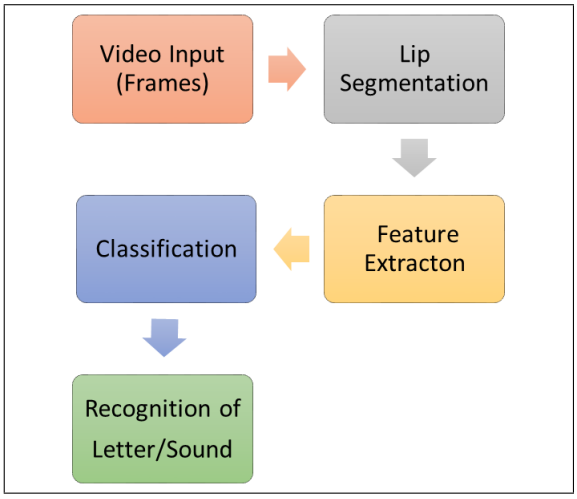


Figure 5.2: Relationship between subsystems

5.5 Conclusion

This chapter illustrates the high level design of the visual speech recognition system. It describes the concepts and provides a visual representation of the system giving a general idea of how the system should behave. The subsystems will be discussed in further detail in the next chapter which will provide the low level design.

Chapter 6

Low level design

6.1 Introduction

In the previous chapter the high level design was discussed resulting in a broad understanding of how the system would be implemented. In this chapter, the Low Level Design (LLD) of the software is described, it will breakdown the HLD into smaller constituent parts allowing for more detail when describing the system.

6.2 Breakdown of high level design

Figure 6.1 illustrates the breakdown of the HLD into the necessary elements required to complete the subsystem.

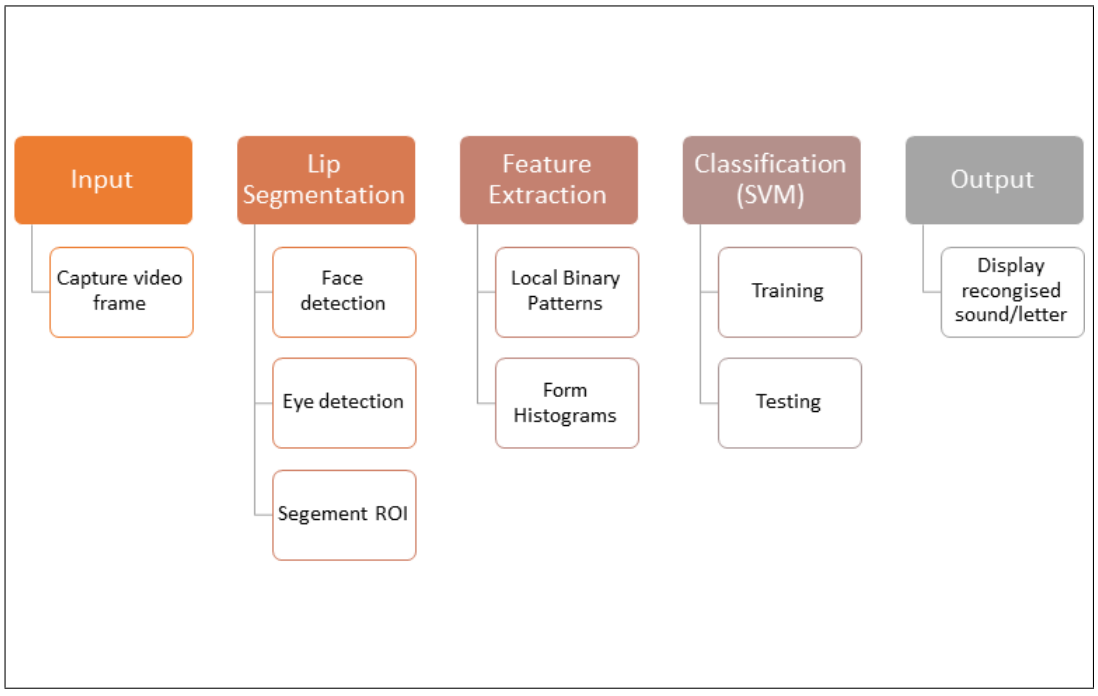


Figure 6.1: Breakdown of HLD to LLD

6.2.1 Capture video frames

Figure 4.1 depicts how video frames are displayed to the user. The method VideoCapture() provides the computer access to the web camera and displays the image frame by frame forming the video.

6.2.2 Face and eye detection

For rapid object detection the system will implement the Viola and Jones object classifier. This component requires a greyscaled frame as input and returns the vector of the object specified. The classifier uses haar-like features to detect the face. Haar-like features represent different intensities of greyscale between two or more adjacent rectangles. The face is detected using face_ cascade.detectMultiScale(). As a result the face is isolated from the background (see Figure 6.2).

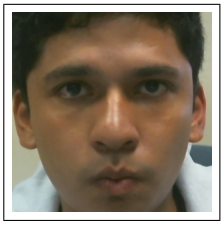


Figure 6.2: Isolated face

The eyes are detected using the eye_ cascade.detectMultiScale(). This is used on the isolated image of the face in order to decrease computing and accurately detect the eyes (see Figure 6.3).

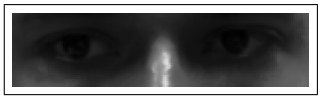


Figure 6.3: Isolated eyes

6.2.3 Segmentation of ROI

The distance between the eyes are used to segment the mouth. It provides the appropriate width and an accurate position along the x-axis for the lips to be segmented. An approximation of the distance is calculated by dividing the eyes into four equal quadrants and using the bounds to locate the lips. The

left bound of quadrant two and the right bound of quadrant three provide the distance and x-positions for the lips. The bottom third of the face provides the appropriate y-position of the face. The bounded regions are illustrated in Figure 6.4. The mouth is then resized to a specific resolution which is to be

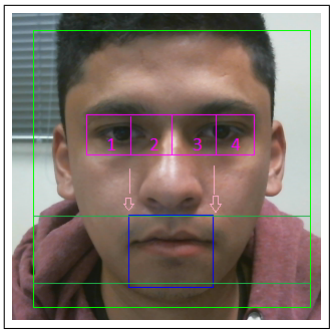


Figure 6.4: Boundries used to isolate the mouth

optimised once the training is complete.

6.2.4 Local binary patterns

The local binary pattern is an image operator which transforms an image into an array or image of integer labels describing small-scale appearance of the image. The basic LBP operator works in a 3x3 pixel block of an image. The pixels in the block are thresholded by its center pixel value. The thresholding values are computed by comparing the center pixel with its 8 neighbours -in a circle around the center. If the centre value is greater than its neighbours value write "1" else write "0". The decimal value is computed and becomes the new pixel of the LBP image. This process is followed for each pixel of the image undergoing the LBP operation[Matti Pietikinen, 2011]. Figure 6.5 displays the LBP procedure pictorially.

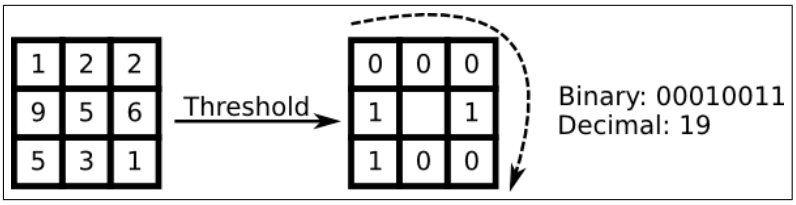


Figure 6.5: LBP calculation from binary to decimal

The features can be extended by changing the neighbourhood values of the LBP where P is the number of sampling points and R the radius of a circle

around the center pixel. Figure 6.6 illustrates the different $LBP_{P,R}$ neighbourhoods. The leftmost image represents a $LBP_{8,1}$ neighbourhood , the center

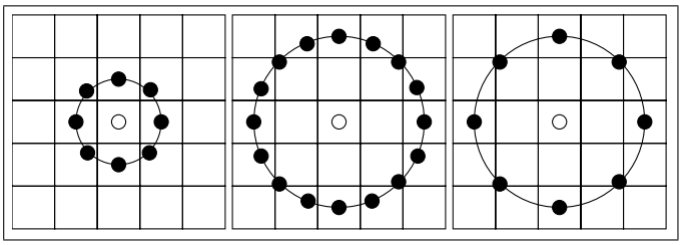


Figure 6.6: Representations of $LBP_{P,R}$

image a $LBP_{16,2}$ neighbourhood and the rightmost image a $LBP_{8,2}$ neighbourhood. Research shows that the $LBP_{8,2}$ neighbourhood has consistently proven to give higher accuracies when extracting facial features [Ahonen et al., 2006; Shan et al., 2009; Hadid et al., 2004; Mushfieldt, 2014]. The $LBP_{8,2}$ will be applied to the greyscaled image of the mouth resulting in the grey texture image which will be used to form histograms (See Figure 6.7).

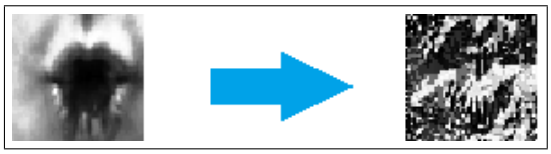


Figure 6.7: Transformation of segmented mouth from Greyscale to LBP

Another extension of the local binary patterns operator uses so called uniform patterns [Ojala et al., 2002]. A local binary pattern is called uniform if its uniformity measure is at most 2. Uniformity is measured by counting the amount of bit transitions within the local binary pattern. For example the patterns 11111111 (0 transitions), 01110000 (2 transitions) and 11001111 (2 transitions) are uniform whereas the patterns 11001001 (4 transitions) and 01010011 (6 transitions) are not.

6.2.5 Form histograms

The LBP texture features produced by the mouth are used to form histograms. It produces a representation of the different intensities of the greyscale values associated with the the LBP image. The bin size is relative to the value of

the neighbours in the LBP operator. The $LBP_{8,1}$ consists of 8 neighbourhood pixels which result in a total of $2^8 = 256$ different labels (See Figure 6.8).

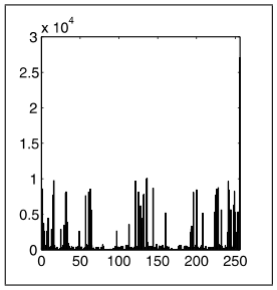


Figure 6.8: Histogram representation of the LBP image

In uniform LBP histogram mapping a label for each uniform pattern and all the non-uniform patterns. This speeds up computation as the histogram will consist of 58 uniform patterns and one extra label (59) containing all the non-uniform patterns.

The histogram is calculated by dividing the LBP image in smaller quadrants called region sizes. A histogram is computed for region. The histograms are then concatenated to form a feature vector which is sent to the SVM for further processing.

6.2.6 Training

This phase includes training the system to recognise specific sounds using the feature vectors (histogram) produced from the LBP image. Each sound is given a unique label representing a sound which was articulated. Different subjects will be used to train the system on visually distinct sounds namely the "Neutral", "OOO" , "EEE", "LLL", "MMM", "FFF" and "TH" sounds.

6.2.6.1 Data set

For this system the training set consisted of 13 South Africans of diverse skin tones. The training subjects were asked to articulate 7 visually distinct sounds including the neutral shape. The videos were taken at 24 frames per second (FPS) to minimize interference. Each subject articulated each specified sound for 3-5 seconds of which 5 frames was extracted. This resulted in a database of 455 images of subjects articulating the various sounds. This data set will be referred to as the "training data".

6.2.6.2 Modelling of data set

The training data was used to build a model with LIBSVM (A Library for Support Vector Machines). The LIBSVM library is open source tool used for support vector classification [Chang and Lin, 2011]. Decision boundaries are computed for the data which form a "model" to be used for classification.

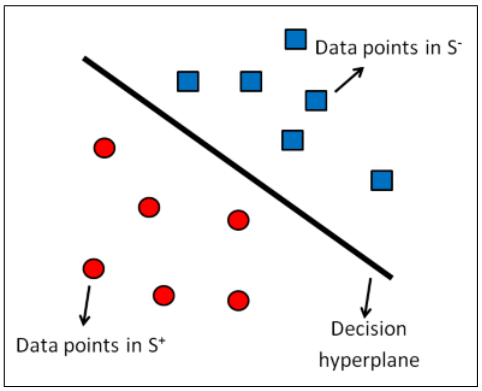


Figure 6.9: Example of a model of data

Figure 6.9 represents a model with the decision boundaries used for SVM classification. Data points S+ and S- could represent the different sounds. The boundaries are computed to separates the data. This idea of data by computing boundaries to fit the given data will be used to classify the different sounds.

6.2.7 Testing

Testing was done after the training phase was completed. The system was tested on unseen subjects. The resultant feature vectors (LBP histogram) formed from these images of the unseen subjects is sent to the SVM to be classified. The classification process returns a label indicting to which sound the image belongs. Chapter 8 will discuss the details of the test set and the accuracy of the system.

6.2.8 Displaying recognised sound/letter

The label is converted back to the recognised sound and displayed to the user as text. The system will display its response to the users input. The success of the system is determined by how accurately it recognises the sound articulated by the user.

6.3 Conclusion

This chapter describes and details all the relevant information needed to complete the system. It describes the system in an algorithmic fashion by detailing each step required to complete each constituent part of the system.

Chapter 7

Code documentation

7.1 Introduction

The previous chapter discuses the low level design of the system. This chapter illustrates snippets of code. The language of choice was C++.

7.2 Function documentation

The lip reading system consist of multiple functions. Figure 7.1 depicts some of the functions used and the comments above each function describes it's parameters and what they return.

```
/*
    Function : detectMouth
        Detect the mouth and provide a placeholder for it
    Parameters:
        frame - The input frame from the video
        mouth - Placeholder for the mouth image
        x - Size of x axis of the mouth
        y - Size of y axis of the mouth
    Returns:
        True if mouth detetced and False if not
*/
 bool detectMouth( Mat frame,  Mat mouth &,int x ,int y);

 /*
    Function : LBP
        Converts the mouth image into an local binray patterns image
    Parameters:
        greyMouth - The input frame refrencing the greyscaled mouth
        lbpImage - Placeholder for the lbp mouth image for display
        uniform - Palceholder for the uniform patterns in lbp image
    Returns:
        Null
*/
 void LBP(const Mat greyMouth & , Mat lbpImage & , Mat uniform &);

/*
    Function : buildHistogram
        Builds histogram from uniform lbp placeholder.
    Parameters:
        unifromLBP - The input unifrom lbp to be coverted to a histogram
        histogram - palceholder for the uniform patterns histogram of the uniformlbp
    Returns:
        Null
*/
 void buildHistogram(Mat uniformLBP ,Mat histogram & ,int , int );

 /*
    Function : printHistogram
        Formats histogram for later processing via Support vector machine.
    Parameters:
        hist - Historgram to be sent to svm
    Returns:
        Null
*/
 void printHistogram(Mat hist);
```

Figure 7.1: Documentation of functions

7.3 Source code

Figure 7.2 illustrates the source code for the function named detectMouth. The function pinpoints the eyes and face of the subject and returns true only if both the face and eyes are detected on the input frame. If true it provides a place holder for the mouth image. The parameters x and y represent the resolution or size of the mouth image.

```
/*
    Function : detectMouth
    Detect the mouth and provide a placeholder for it.
    Parameteres:

    frame - The input frame from the video
    mouth - Placeholder for the mouth image
    dimx - size of x axis of the mouth
    dimy - size of y axis of the mouth

    Returns:
    True if mouth detetced and False if not
*/
bool detectMouth( Mat frame, Mat & mouth, int dimx, int dimy)
{

Mat frame_gray;
cvtColor( frame, frame_gray, CV_BGR2GRAY );
equalizeHist( frame_gray, frame_gray );
vector<Rect> face;

face_cascade.detectMultiScale( frame_gray, face, 1.1, 3, 1|CV_HAAR_FIND_BIGGEST_OBJECT, Size(70, 70) );//-- Detect face

if (face.size())////-- find face
{
    Rect face_i = face[0];//store points on face
    int x_face = face_i.x;
    int y_face = face_i.y;
    int width_face = face_i.width;
    int height_face = face_i.height;

    Point face_ptr1( x_face , y_face);
    Point face_ptr2((x_face + width_face) , (y_face +  height_face) );
    rectangle(frame , face_ptr1 , face_ptr2 , CV_RGB(0, 255,0), 1);

    vector<Rect> eye;
    eye_cascade.detectMultiScale( frame_gray, eye, 1.1,4,0|CV_HAAR_SCALE_IMAGE,Size(30,30) );//detects eye region

    if (eye.size()) // find eyes
    {
        Rect eye_i = eye[0];
        rectangle(frame , eye_i , CV_RGB(255, 0,255), 1);
        //Use eyes and face to get mouth
        int x_mouth =  eye_i.x + eye_i.width/4.5;
        int y_mouth =  face_i.y + (face_i.height/1.38);
        int width_mouth =  eye_i.width/1.8;
        int height_mouth =  face_i.height/4.5;

        Point mouth_ptr1( x_mouth , y_mouth);
        Point mouth_ptr2( (x_mouth + width_mouth ) , ( y_mouth + height_mouth) );
        rectangle(frame , mouth_ptr1, mouth_ptr2 , CV_RGB(0, 0,255), 1);
        Rect mouthROI = Rect(mouth_ptr1 , mouth_ptr2);

        mouth = frame_gray(mouthROI);
        equalizeHist( mouth, mouth);
        resize(mouth,mouth,Size(dimx,dimy),1,1,INTER_CUBIC);
    }

    else
        {return false;}
}

else
    {return false;}
}
```

Figure 7.2: Source Code of mouth detection

7.4 Conclusion

This chapter describes how the code was documented. Screen shots were taken of the code to illustrate the code documentation from the programmers point of view. The comments above each function are used to provide an understanding of what the function is used for. In-line comments were also used to provide further understanding (see Figure 7.2).

Chapter 8

Testing and Evaluation

8.1 Introduction

The previous chapter discuses the code and how it was documented. This chapter presents the optimization and the testing and evaluation of the program based on the requirements described in Chapter 2. From chapter 2 the software is expected to segment the mouth and recognise sounds or letters articulated by the user. This chapter is organized as follows: Section 8.2 discusses different testing methodologies and how they were applied to the application; Section 8.3 discusses the optimization of the training set model;Section 8.4 The lip reading accuracy testing.

8.2 Testing methodology

This section discuses the testing methodology used to assess the application and determine whether the requirements were met. There are many types of testing methodologies which can be implemented in order to evaluate the properties of interest within the system. For this software, regression, destructive and quantitative testing was used.

8.2.1 Destructive testing

This form of testing attempts to make the software fail or crash by giving the software unexpected inputs. This method was used to find the points of failure in the program. In terms of the lip reading application the users were asked to be a specific distance (approximately 70cm) from the camera in order for the system to detect the user's mouth accurately. If the system is unsuccessful in detecting the user's mouth the program waits until the user is the appropriate distance away from the camera.

8.2.2 Regression testing

This form of testing is used to discover bugs within the system after changes such as repairs or improvements are made [Myers et al., 2011]. In terms of the development of the lip reading system this approach was used to test to the correctness as well as the quality of the program while it was being developed. The lip reading system consists of multiple functions. Each function was tested to see whether it gave the desired output.

8.3 Feature Vector and SVM Optimization

This section discusses the optimization of the resolution and region size of mouth images. Before formal testing can commence optimization is done to produce the ideal "model" based on the training data set. Optimization of SVM was done using LibSVMs grid search function [Chang and Lin, 2011]. The search enquires a number of possible C and γ values used as parameters for the Radial Bias Function (RBF) kernel. The RBF kernel can handle the case of non-linear relation between class and attributes [Hsu et al., 2003]. Grid search uses v fold cross-validation set by partitioning the training data into v subsets of equal size. Sequentially one subset is tested using the the classier it is trained on the remaining $v-1$ subsets [Hsu et al., 2003]. It returns the best pair of C and γ values with determined by the best cross-validation accuracy.

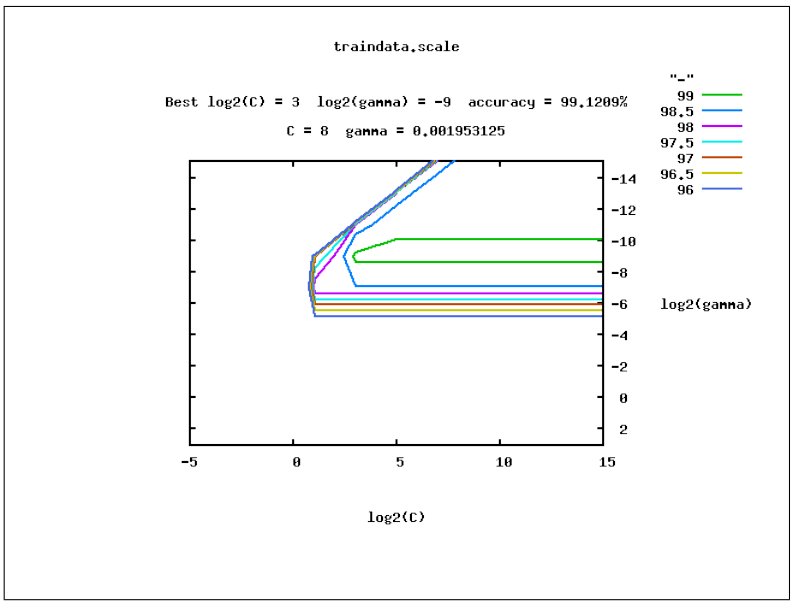


Figure 8.1: Example of LIBSVM grid search output

Figure 8.1 represents a grid search model. An accuracy of 99.12% was achieved when testing on the cross validation sets. $C = 8$ and $\gamma = 1.953125 \times 10^{-3}$ were the best pair of parameters for the SVM for this specific set of data. The data used was the training data at a resolution of (50 x 50) and a region size of (10 x 10).

Determining an optimal resolution and region size is a process of trial and error. Width and height combinations of 40, 50 and 60 pixels were used for the resolution alongside region sizes of (5 x 5) and (10 x 10) respectively. The final accuracy was determined by testing the model against four of the six test subjects described in Section 8.4.2.

8.3.1 Results and analysis

Table 8.1: Optimized resolution and region sizes (RS) for mouth images

Resolution	RS=(5 x 5)	RS=(10 x 10)
(40 x 40)	72.1429	84.29
(40 x 50)	67.8571	82.87
(40 x 60)	73.5714	78.57
(50 x 40)	71.4286	77.86
(50 x 50)	68.5714	85.71
(60 x 40)	71.4286	81.43
(60 x 50)	69.2857	81.43

Table 8.1 illustrates the results and the optimal resolution was found to be (50 x 50) with a region size (10 x 10) with an accuracy of 85.71%. This was obtained with a $C = 8$ and $\gamma = 1.953125 \times 10^{-3}$ see Figure 8.1 . With these parameter values a new SVM was trained resulting in the use of the optimal model of the system.

8.4 Lip reading accuracy testing

This section describes the experiment carried out in order to test whether the lip reading software recognises the desired sounds. The system aims to accurately recognise each of seven distinct sounds with regards to the mouth area. In each case the system response was compared to the ground truth. The lip reader classifies each input frame of the seven sounds. If the output of

the system matches that of the label of the sound articulated it is considered a correct classification, otherwise it is an incorrect classification.

8.4.1 Experimental Set-up

All experiments were carried out on a PC using the following specification; Intel i5 2400 3.2 GHz quad core CPU, and 8 GB RAM, running Ubuntu 14.04 (LTS) x64 operating system. The videos were taken with the Logitech-HD Webcam C615 at a resolution of 640 x 480 at a frame rate of 25 frames per second (FPS).

8.4.2 Data Set

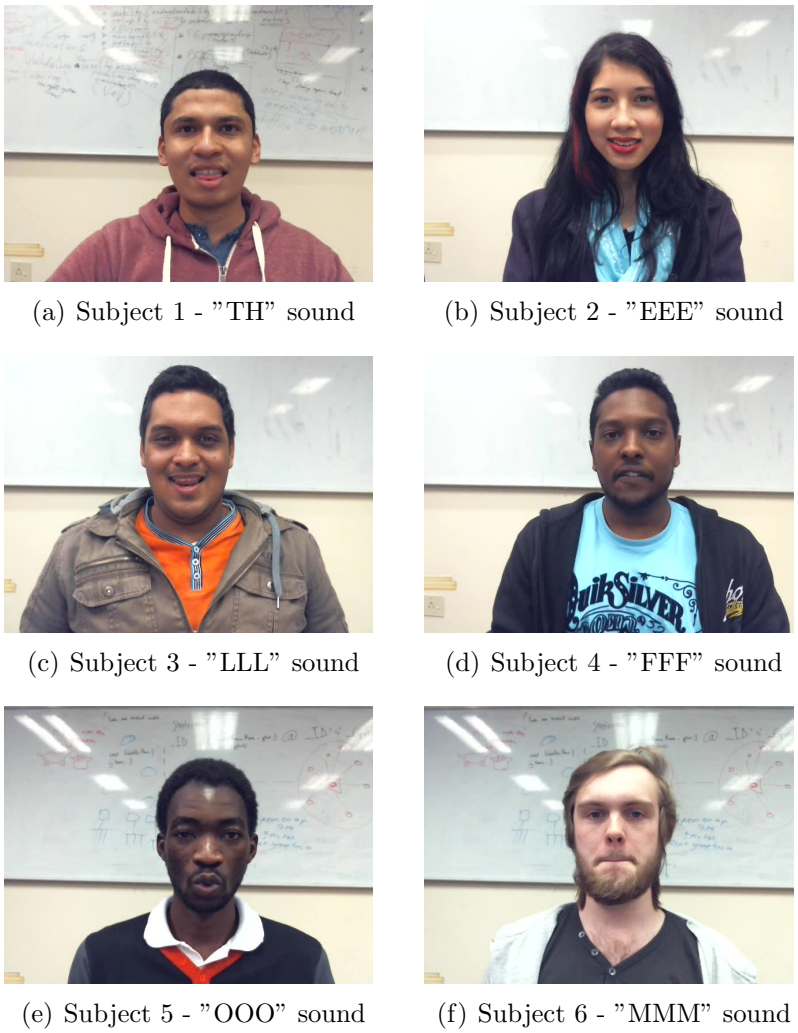
For this experiment, 6 South Africans with diverse skin tones were used. Each subject was required to sit on a chair facing the web camera (approximately 70cm from the screen) and instructed to articulate each of the seven visually distinct sounds for a total of three to five seconds. Each sound's mouth image was shown to each subject prior to video capturing. Using a web camera at a frame rate of 25 FPS, 5 frames were randomly extracted from each video. This resulted in a database containing a total of 210 images of all subjects. Each subject had a total of 35 images and each sound a total of 30 images. This data set is henceforth referred to as the test data set. Figure 8.2 illustrates the six subjects articulating different sounds each having different skin tones.

8.4.3 Experimental procedure

Six subjects were used outside the training set as input to the system. The images contain the subjects articulating the following sounds; "Neutral", "OOO" , "EEE", "LLL", "MMM", "FFF" and "TH". In each case, the output of the system was analysed for accurately recognizing the specific sound articulated by the subject.

8.4.4 Results and analysis

The table provides the system response for each image test data set. Table 8.2 summarizes the overall results as a percentage per sound with regards to each test subject. Table 8.3 is a confusion matrix summarizing the results of the system response to the actual sound articulated by the test subject's.



(a) Subject 1 - "TH" sound (b) Subject 2 - "EEE" sound

(c) Subject 3 - "LLL" sound (d) Subject 4 - "FFF" sound

(e) Subject 5 - "OOO" sound (f) Subject 6 - "MMM" sound

Figure 8.2: An example of six test subjects each articulating a sound

The system achieved an overall average recognition accuracy of 78.57%. The averages range from 46% to 96.66% for each sound. It is noted that the system registered a high average recognition accuracy of 89.33% for the "Neutral", "OOO", "MMM", "LLL" and "FFF" sounds. These results are highly encouraging considering the amount of sounds being classified. The lowest accuracies were registered for the following two cases: 46.66% for "EEE" and 56.66% for "TH". Both of these accuracies are considerably lower than the average of the system. In the case of the "EEE" sound the low accuracy could be attribute to how it was articulated as it is considered to be an open mouth sound with the teeth showing. The difference in each subjects teeth and position of the tongue when articulating these sounds may have resulted in the misclassifications. Table 8.3 shows that "EEE" was misclassified with "LLL"

Table 8.2: Results for lip reading system

	Neutral	OOO	MMM	EEE	LLL	FFF	TH	Total
Test Subject 1	5	5	5	3	5	5	5	33
Test Subject 2	5	4	4	5	5	5	5	33
Test Subject 3	5	5	0	5	0	5	0	20
Test Subject 4	5	5	5	0	5	5	5	30
Test Subject 5	4	5	5	0	5	5	0	24
Test Subject 6	5	5	5	1	5	2	2	25
Total	29	29	24	14	25	27	17	165
Total (%)	**96.66**	**96.66**	**80**	**46.66**	**83.33**	**90**	**56.66**	**78.57**

Table 8.3: Confusion matrix for lip reading system

	Neutral	OOO	MMM	EEE	LLL	FFF	TH
Neutral	**29**	0	0	0	0	0	1
OOO	0	**29**	0	1	0	0	0
MMM	4	0	**24**	0	0	2	0
EEE	0	0	0	**14**	9	0	7
LLL	0	0	0	5	**25**	0	0
FFF	0	0	3	0	0	**27**	0
TH	3	0	0	5	3	2	**17**

on nine images and "TH" on seven. In this case the SVM tried a best-effort incorrect classification. It is also noted that the variable misclassification of the "TH" sound could be due to specific test subjects not articulating the sounds correctly. The the "TH" sound should be pronounced with the tongue outside the mouth and the top set of teeth slightly exposed. On inspection of the vidoes, test subject 3 and 5 were articulation of the sounds were not pronounced enough.

Figure 8.2 is a visual summary of Table 8.1. The number of classifications done for each sound amounts to 30. It is clear that five of the seven sounds have exceptionally high accuracies as compared to the other two. It is also noted that the test subject 3 and 5 are have 0% value on the "TH" sound. The test data set in general appears to be more affected by the "TH" and "EEE" sounds.

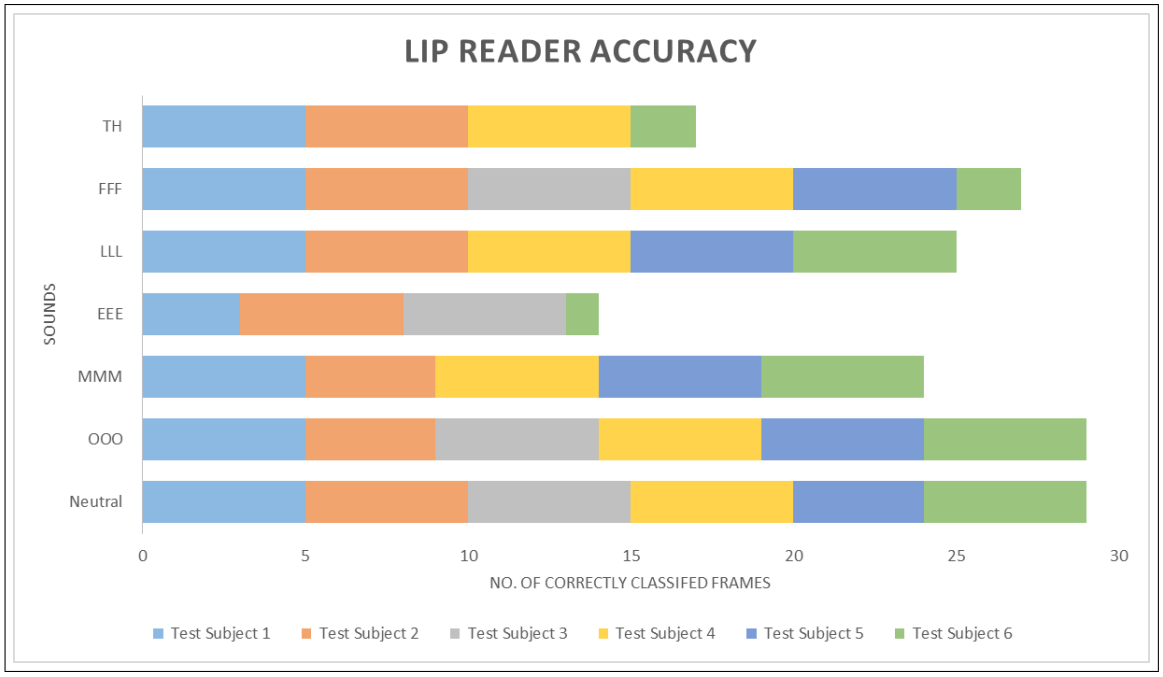


Figure 8.3: Accuracy of Lip reading system

8.5 Conclusion

The objective of the system as described in Chapter 2 was to accurately recognise specific sounds. The results have shown that the system does so with an average accuracy of 78.57%. The mouth was accurately segmented from the test data which played a role in the accuracy. The high accuracy, accurate mouth segmentation and fast computation of feature vectors lead to the software responding to live video feed. The system was then stabilized by classifying four consecutive frames. If the output label was the same for each of the four frames the system displayed the recognised label as text to the user. Tis completed a successful real time lip reader. The next chapter discusses the user guide for the system.

Chapter 9

User Guide

A user guide is intended to assist people with using a particular system.The chapter will discuss what is needed to run the software and how to run it. The interface of the system is rudimentary. Emphasis was placed on the accuracy of the system and displaying the results.

9.1 Pre-requisites

The following are the prerequisites that must be installed to be able to run the application:

- Ubuntu or any variant of Ubuntu x64 or x86
- GCC compiler
- Web camera preferably the Logitech-HD Webcam C615
- V4L2 camera driver
- OpenCV version 2.4.2 or higher

9.2 Configuration and execution

This section details the steps to be followed by user in order to configure the system.

- The web-cam should on top of the monitor with a clear unobstructed view of the user.
- Open V4L2 to adjust the camera brightness and switch off camera auto-focus.
- The user should be seated facing the camera at a distance of approximately 70cm.

- Open terminal and change directories to the folder containing code. The folder is named Lip_ Reader.
- Type "make" to compile the program and produce an output file.
- If errors during compilation occur make sure the GCC compiler and OpenCV libraries are correctly installed.
- Run by typing the command "./lip_ reader"

```
P = main

all:
    g++ $(P).cpp -o lip_reader `pkg-config --cflags --libs opencv`

clean:
    rm lip_reader
```

Figure 9.1: Makefile

Figure 9.1 represents the makefile. This file contains the command for compiling the code and creating an output file. The "make" command will compile the C++ code and create the output file lip_ reader.

Once the system starts windows will appear containing the live feed of the user and their mouth. The windows can be moved to a suitable position if they do not start up next to one another. The system will pause if it does not detect the users mouth. If this is the case the user would have to adjust themselves by either moving closer or further away from the camera until the software resumes. The system will classify the frames as soon as a user is detected. The output is displayed in the top left hand corner of the screen. Should the user want to exit the program, the button "c" will do so.

Bibliography

Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041.

Barnard, M., Holden, E.-J., and Owens, R. (2010). Lip tracking using pattern matching snakes. In *Asian Conference on Computer Vision - ACCV*.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV*. O'Reilly Media, first edition.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27:1–27:27.

Hadid, A., Pietikainen, M., and Ahonen, T. (2004). A discriminative feature space for detecting and recognizing faces. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–797. IEEE.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification.

Matti Pietikinen, Guoying Zhao, A. H. T. A. (2011). *Computer Vison Using Local Binary Patterns*, volume 40. Springer.

McGurk, H. and MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, 264:746–748.

Mehrotra, H., Agrawal, G., and Srivastava, M. (2009). Automatic lip contour tracking and visual character recognition for computerized lip reading. *International Journal of Computer Science*, 4:62–72.

Mushfieldt, D. (2014). Robust facial expression recognition in the presence of rotation and partial occlusion. Masters thesis, University of the Western Cape.

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The art of software testing.* John Wiley & Sons.

Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987.

Osuna, E., Freund, R., and Girosi, F. (1997). Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition, Proceedings., IEEE Computer Society Conference*, pages 130–136.

Shan, C., Gong, S., and McOwan, P. W. (2009). Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 27(6):803–816.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference*, volume 1, pages 511–518. IEEE.

Wilson, P. I. and Fernandez, D. J. (2006). Facial feature detection using haar classifiers. *Journal of Computing Sciences in Colleges*, 21(4):127–133.

Yang, Z. and Ai, H. (2007). Demographic classification with local binary patterns. In *ICB'07 Proceedings of the 2007 international conference on Advances in Biometrics*, pages 464–473.

Appendix A

Term Plan

GOAL	Due Date
Research • Learn how to use OpenCV	End of Term 1
• Accurately locate mouth and extract features	End of Term 2
Implementation • Train the system to recognize a sounds or letters • Optimize image for better recognition	End of Term 3
Test and Evaluate • Add more training and testing data	End of Term4

Figure A.1: Term Plan