

# **DAB - DIGITAL ASSISTANT FOR THE BLIND**

FINAL PROJECT REPORT

SUBMITTED BY

**ARJUN B DEV (MGP17CS024)**

**ANSEL BASIL SUNNY(MGP17CS021)**

**ANANTHU RAJENDRAN(MGP17CS019)**

**GABY S (MGP17CS043)**

IN PARTIAL FULLFILLMENT OF

THE REQUIREMENTS FOR

THE AWARD OF

**BACHELOR OF TECHNOLOGY**

**IN COMPUTER SCIENCE AND ENGINEERING**



**SAINTGITS**  
LEARN.GROW.EXCEL

**Department of Computer Science and**

**Engineering Saintgits College of Engineering,**

**Pathamuttom**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**

June 2021

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**SAINTGITS COLLEGE OF ENGINEERING**

**2020-2021**



**CERTIFICATE**

*Certified that this is the bonafide record of final project work entitled*

**DAB - DIGITAL ASSISTANT FOR THE BLIND**

*Submitted by*

**ARJUN B DEV (MGP17CS024)**

**ANSEL BASIL SUNNY(MGP17CS021)**

**ANANTHU RAJENDRAN(MGP17CS019)**

**GABY S (MGP17CS043)**

*Under the guidance of*

**ER. THOMAS JOSEPH**

(Assistant Professor)

*In partial fulfillment of the requirements for award of the degree of  
Bachelor of Technology in Computer Science and Engineering under the  
APJ Abdul Kalam Technological University during the year 2020-2021*

**HEAD OF DEPARTMENT**

**DR ANJU PRATAP**

Professor & HOD

**PROJECT COORDINATORS**

**(1) ER .VEENA A KUMAR**

Assistant Professor (Senior)

**(2) ER .JINU THOMAS**

Assistant Professor

**PROJECT GUIDE**

**ER.THOMAS JOESPH**

Assistant Professor

## ACKNOWLEDGEMENT

We express my gratitude to our principal, **Dr. Josephkunju Paul C**, Principal, Saintgits College of Engineering for providing us with excellent ambiance that laid a potentially strong foundation for our work.

We express my heartfelt thanks to **Dr. Anju Pratap**, Head of the Department of Computer Science and Engineering, Saintgits College of Engineering who has been a constant support in every step of my seminar and the source of strength in completing this seminar.

We express my sincere thanks to **Er. Veena A Kumar** and **Er. Jinu Thomas** of the Computer Science and Engineering Department providing us with all the facilities, valuable and timely suggestions and constant supervision for the successful completion of my seminar.

We also thank **Er. Thomas Joseph**, Assistant Professor, Computer Science and Engineering Department for his/her supervision, encouragement, help and support.

We am highly indebted to all the faculties of the department for their valuable guidance and instant help and for being with us. I extend my heartfelt thanks to my parents, friends and well wishers for their support and timely help.

Last but not the least we thank Almighty God for helping me in successfully completing this seminar

## **ABSTRACT**

Globally, around 3 million people are moderately or severely visually impaired. These blind and visually impaired are faced with a lot of challenges, when it comes to reading texts to improve their knowledge. The classical solution available for them turned out to be inefficient. So new and improved solutions were needed. Emergence of technology like image recognition, led to advanced systems like Optical Character Recognition and Text To Speech engines which could be configured to create digital smart assistants that could read out textual contents from real texts. Now there exists many OCR and TTS engine services either as open source projects or very economic for use. Speech synthesis technology will enable the machine learning systems to communicate with the blind efficiently taking advantage of the blind or visually impaired ability to hear. Other technology like object, face recognition can be interlinked with TTS engines to provide more services. Digital Assistant for the Blind is a smart assistant shades that enables the users to take images of texts and have that read out to them aloud. It also includes other features like object,face recognition and collision detection. Face detection feature enables the user to use the shades to recognize the person in front of them provided their images are trained for recognition. The prototype of the system is built on a Raspberry PI module with a Python based driver program controlling the entire operations based on the user's toggles. They could toggle either face,object or character recognition by pressing buttons and after recognition the result will be provided to them in an audio output format. These technology driven smart assistant systems empowers the B/VI users to be independent of others for reading and can reduce Blind/Visually-impaired depending on other people.

### **Keywords:**

optical character recognition, digital assistant, text to speech, object recognition

## TABLE OF CONTENTS

INDEX	PAGE NO
1 INTRODUCTION-----	07
2. PROBLEM STATEMENT-----	08
2.1 PROPOSED SOLUTION : DAB SHADES:-----	08
3. LITERATURE REVIEW :-----	09
4. TECHNOLOGY :-----	11
4.1 OBJECT RECOGNITION :-----	11
4.2 FACE RECOGNITION :-----	12
4.3 OPTICAL CHARACTER RECOGNITION :-----	13
4.4 TEXT TO SPEECH SYNTHESIS :-----	15
5. SYSTEM DESIGN :-----	17
5.1. HARDWARE REQUIREMENTS :-----	19
5.2. SOFTWARE REQUIREMENTS :-----	19
6. IMPLEMENTATION:-----	20
6.1 TESSERACT OCR:-----	20
6.2 GOOGLE TTS:-----	21
6.3 FACE DETECTION:-----	22
6.3.1. WEB APPLICATION:-----	25
6.4 OBJECT DETECTION:-----	26
6.5 IMAGE CAPTURE AND AUDIO OUTPUT:-----	27
6.6 PYTHON DRIVER PROGRAM:-----	27
7. CONCLUSION:-----	28
8. FUTURE SCOPE:-----	29
9. REFERENCES:-----	30
10. APPENDIX:-----	31

## LIST OF FIGURES

INDEX	PAGE NO
4.1.1 : Steps involved in an object detection system.-----	11
4.2.1: The work flow of a face recognition system.-----	12
4.3.1: The work flow of OCR engine.-----	13
4.3.2: Internal architecture of OCR Engine.-----	14
4.4.1 : Overview of a typical TTS system.-----	15
5.1: Block diagram of the DAB system.-----	17
5.2: Process flow in the DAB system.-----	18
6.1: The first image is denoised,filtered and converted to greyscale.-----	20
6.2:The text recognised from the iamge using tesseract ocr -----	21
6.3: face detection module detecting a face and recognizing it.-----	25
6.4:MongoDB collection with embeddings. -----	26
6.5: The web application-----	26
6.6: object detection module detecting objects from a captured image-----	27

## 1. INTRODUCTION

Since the development of technology, the application for latest technology has been growing exponentially. Major breakthroughs in the field of computer vision and speech synthesis has opened way to new possibilities. Navigating tools for blind is one among the many. Recent trends show various development in blind navigation-based system and wearables making an entry into the market. But a lot of research is still needed in these fields as these technologies are not as affordable or accessible to a majority of those affected by visual impairment.

Out of the total population of India of around 1.38 billion people, 31.6 million people are recorded to be blind or suffering from a major visual impairment. That consists of nearly 3 percent of the entire population of India. One major issue they face is their education. Traditional solutions like Braille requires training to be made use of, which is not very commonly accessible. Moreover books in Braille are not commonly accessible. Their disabilities naturally make them dependent on others, be it to walk around, to read a book, or even to do regular things enjoyed by people. Even if they find a way to be self dependent in the form of a guide dog, or using walking sticks, They suffer from a lot of limitations that they face by not being able to see. A ground breaking invention would be to create bionic eyes to help them see. Alas we are limited by our technology, The best possible alternative is to emulate an eye and allow the blind or visually impaired to take advantage of their other senses, say their ears. If technology could be used to capture photos to analyze them, and describe it to the user, that would be a helpful. A smart assistant system dedicated completely for the blind was needed.

DAB - or digital assistant for the blind makes use of optical character recognition, object detection, face detection and speech synthesis engines to create a wearable smart assistant for the blind. The is a smart assistant shades that enables the users to take images of texts and have that read out to them aloud. It includes features like object recognition and collision detection which enables the user to recognize the objects in their path. Face detection feature enables the user to use the shades to recognize the person in front of them provided their images are preset for recognition. The shades will be modular and very easy to use. DAB shades will be beneficial to all visually impaired or blind as they can reduce their dependency on other people. This report will describe in detail the design methodology derived for implementing the product after doing an extensive literature review and designing the work flow for the smart assistant.

## **2. PROBLEM STATEMENT**

As the blind and visually impaired population is present worldwide, there are some challenges faced by them while reading or doing their regular life routines. Also there are a great many people around the world whose eye sight reduced drastically. Even though they might not fall into the category of visually impaired, their visual accuracy is very low. It is known that there are millions of people who are unable to read like an ordinary person. The reasons vary from :

- being born blind.

- accidents.

- old age.

- progressive sight loss.

There are a lot of problems faces by blind people. Some of them include navigating around places ,finding Reading Materials, arranging Clothes, overly helpful Individuals,getting dependent on others, taking medicine, identifying near by things, walking through unsafe areas etc. The blind and visually impaired has to depend on other people to read, or interact with their environment. With the emergence of technology like computer vision and speech synthesis, smart assistant systems can be built which could act as their personal assistant. Visual impairments disrupt the education of all those affected. Being unable to read, their options of further education falters and their options are limited. Traditional solutions like Braille are inefficient as not many books are printed in Braille as well as training is required for it. Current systems are neither accessible nor preferred as it has limits. Self dependent support solutions for the blind, using advanced technology is the next step.

### **2.1. PROPOSED SOLUTION : DAB SHADES**

DAB shades are wearable shades having camera modules for capturing snapshots of surroundings, which can be then processed to identify the objects in the captured image which is then fed to the wearer as audio signals through the sound output(an earpiece).A smart assistant system with a good OCR and TS engine helps them to have the text read out loud to them easily. It also has face detection, object detection system integrated with it, working in unison with the TTS engine to give output about the snapped image as per the user's request.



### 3. LITERATURE REVIEW

Nowadays, there are some technical systems to help the blind persons. To introduce these systems, the first one which comes in our mind is 'Braille'. [1] In this system, it uses 6 raised dots in 2X3 fashion per cell to represent a symbol. A character of any language, a punctuation, or any other indicator meaning is represented by this symbol. With the help of finger, the dots are sensed and from this pattern information is gained. Numbering of right column is from 1-3 and the rest is from 4-6. There will be 64 (26) possible different kinds of patterns per cell. A letter of any language including punctuation, symbols and so on are represented by each dot pattern. Multi-cell is used if one cell is not sufficient for representing a symbol of any language. To make multi-cell pattern recognizable, another special cell is used to allow a multi-cell pattern. Blind and visual impaired uses different kinds of Braille. Grade-1 type of Braille is a one-to one conversation. it doesn't permits abbreviations or words but grade-2 Braille is the advanced version that can permit abbreviations and words and thus saves spaces in printing. An unstandardized shorthand format is used in grade-3 type Braille which is not used in publications. Mostly used device which helps BVI is Braille Note-taker that replaces standard QWERTY keyboard to give input to computer. A text to speech synthesizer is also used and sometimes Braille displays used for output to BVI from the screen.

Another technology is Screen Reader that is used to help blind and also visually impaired people. It helps to read the text aloud that is displayed on the monitor of an image that is captured through image sensor which is connected to mobiles, computers, tablets or other devices. A text to speech synthesizer helps to produce speech of corresponding text, it gets information from the screen and the speech synthesizer recognizes. A text to speech engine software is embedded with the screen reader in this technology. [2]

Another technology which helps BVI is Finger Reader. The user wears this device to his finger. The plain printed text is accessed by BVI through finger reader. The user who put this device to their finger can be able to scan the text and get corresponding audio. It also has haptic sense of the layout. These senses alert the reader if he moves away from the line what he is scanning, it helps user to maintain scanning in straight line [3].

Braille Printer is a hardware that can print hard-copy of Braille, for that it uses Braille translation software program that converts electronic text from computer device or tablet or mobile to Braille. Heavy weight papers are used to print the Braille on each sides, but it is slower and extra high priced process [4].

Screen Magnifier is a graphical output of a computer which represents the textual content on the display in enlarged form. Thus, it improves the visibility which helps humans who have terrible

visibility. An E-Book Reader or Audio Book Reader which is accessible for laptop or mobile, that can examine the text louder from their screen to BVI, which helps to read the e-book that appeared on the display screen with a touch display gesture and it performs this job by using textual content to speech function.[5]

Blind Reader can be capable to study the electronic documents. Through this, a reader can clearly understand and go through the substances correctly by using the usage of feel of touch. Blind reader is an application. It is developed for android gadgets. The person has to cross his fingertip horizontally from left to right on the android cellular display screen and the phrases that are touched are study out through speaker which is built in with the cellular device. In this software the record is considered as the series of phrases so that the gadget can recognize the touched phrase accurately. For visually impaired humans the total facts on the display is now not clear will no longer be visible by them.[5]

A system, called FaceNet, that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors. This method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. To train, triplets of roughly aligned matching / non-matching face patches are generated using a novel online triplet mining method. The benefit of this approach is much greater representational efficiency and achieve state-of-the-art face recognition performance using only 128-bytes per face.[6]

The literature review conducted has led to the conclusion that although there exists various smart assistants, they lack certain features like ease of use, of serviceability, of affordability, or not takes into application certain cutting edge technology like computer vision or its subset optical character recognition. This helps to formulate the features needed for the DAB shades.

## 4. TECHNOLOGY

The DAB shades intended functionalities can be implemented using a combination of OCR engines, TTS engines, and object and face detection systems. These engines can be combined and configured to be controlled manually using toggles which communicates with the a driver program that runs the smart assistant system. Here each of these component technology will be discussed in detail.

### 4.1. OBJECT RECOGNITION :

Object recognition is a computer vision technique for identifying objects in images or videos. Object recognition is a key output of deep learning and machine learning algorithms. Identification and localization in object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them. The working principle of object detection is as follows. Every object class has its own special features that helps in classifying the class, for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point i.e. the center are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found. The steps involved in the working of object detection can be explained using the following :

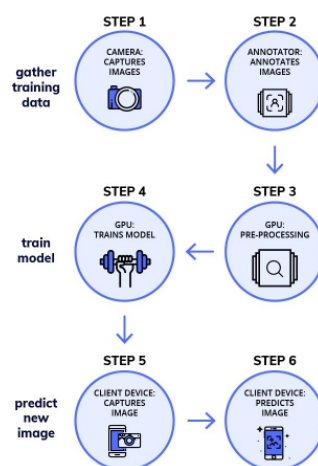


Figure 4.1.1 : steps involved in an object detection system.[image source : Google]

The first step is to capture images containing objects which are to be recognized. A collection

of photos in various distances or angles are taken. The next step is to annotate the images with the objects. Each region in the images containing the objects to be recognized are physically marked by the user for a few set of photos. The next step involves preprocessing, in which the images to be used , if needed by the object detection system is passed through a few digital image processing techniques to make it more suitable for detection. The next step is where the system trains the model with the image data-set provided earlier. After this step, the model is supposed to identify any images containing the trained objects given to it. This is done when a client system on which the trained model is working on, is provided with an image for recognition.

#### 4.2.FACE RECOGNITION :

A facial recognition system is a technology capable of matching a human face from a digital image against a database of faces. It is available on various machines, and there are various methods to implement face recognition. An example is to use OpenCV, dlib toolkit, and face\_recognition module with a Python VE too create a face recognition system. Face detection can be regarded as a specific case of object-class detection. Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process. The following flowchart can help understand the working of a trained basic face recognition system :

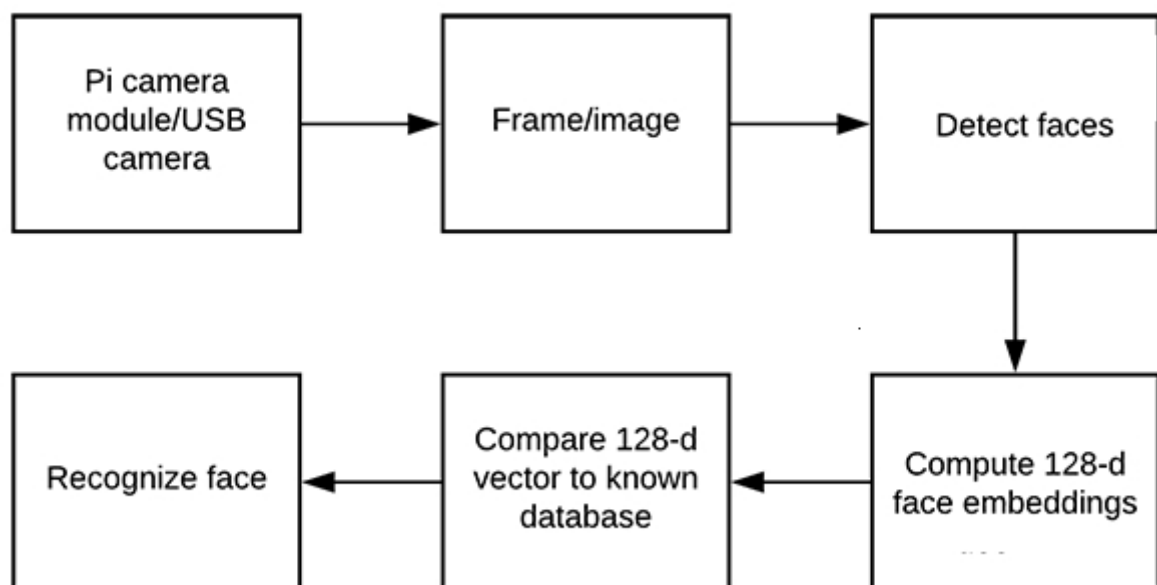


Figure 4.2.1: The work flow of a face recognition system.[10]

Initially the client system on which the model is running captures the image. This image is then

preprocessed to create frames in it around the possible faces in the image. During training, those images with faces that are to be recognized were used to create a 128-d embedding which was stored for later reference. The newly captured photos are also used to compute their corresponding 128-d face embedding. These are compared with the embedding values already stored earlier during training, and if a match is found, then the model is able to recognize the face.

### 4.3.OPTICAL CHARACTER RECOGNITION :

Optical character recognition is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text. Various OCR tools and services are already present in the market. Open source OCR tools are available on Raspberry PI. Combination of such OCR tools and OpenCV library to implement OCR is possible. Cloud based OCR services could be used by subscribing to their services, and availing them using API's in driver programs. Most of the OCR however requires their input images be preprocessed in a certain manner before they could be processed. These involves digital image processing techniques such as binarization, skew correction, image smoothing etc. Depending on the OCR system these might be internally available in the engine or may be need to be done externally using support softwares. The following figure helps understand how the OCR engine functions

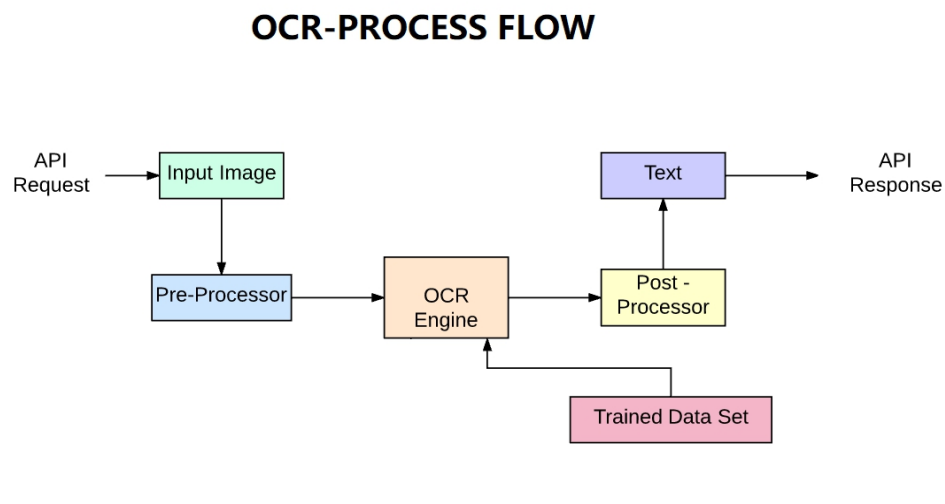


Figure 4.3.1: The work flow of OCR engine[8]

On request, the image to be processed is passed to a pre-processor. Depending on the OCR various digital image processing techniques are implemented on the image making it easier for the OCR for character recognition. This image is then passed onto the OCR Engine. The OCR engine

has a trained data set which it can use to process the input image. After analyzing the input image, various post-processing is done, and the text recognized from the image is received. How the OCR cross matches the data set with the input image depends on the model of OCR. One of the best open source OCR is TESSERACT OCR. The following figure shows the internal architecture of TESSERACT OCR which can be used to have an understanding of what is happening inside an OCR Engine.

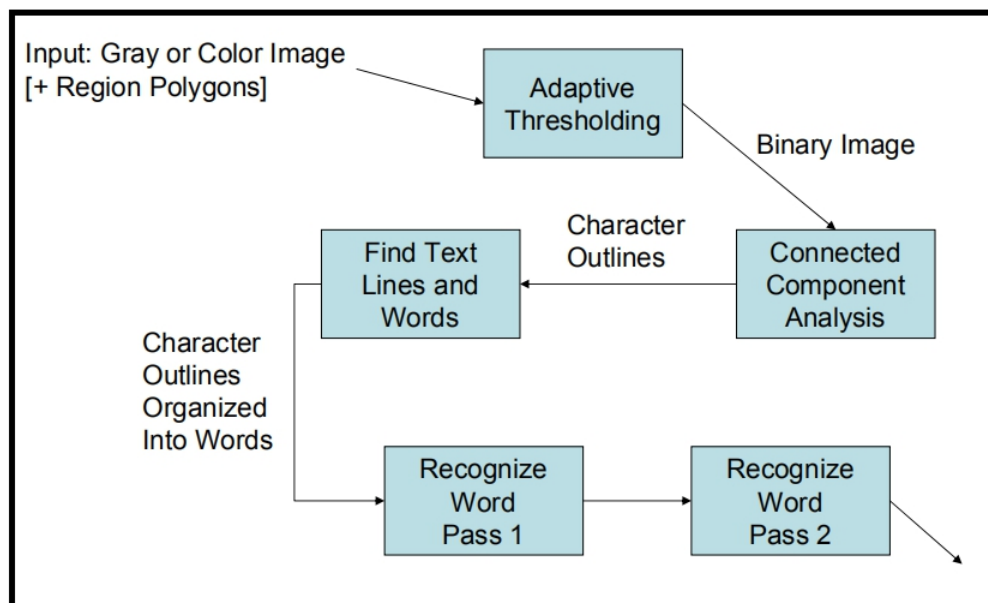


Figure 4.3.2: Internal architecture of OCR Engine.[6]

Initially the image taken is preprocessed and a binary image is received. There are 4 stages that follow it. They are connected component analysis, Finding text lines and words, Recognize words pass1, and Recognize words pass 2.

Stage 1: Connected component analysis - This is the step in which outlines of the components are stored. By inspecting the nesting of outlines, and the number of child and grandchild outlines, around the texts it is simple to detect them and recognize it as black-on-white text. Afterwards, the outlines are gathered together, purely by nesting, into Blobs (binary large objects) Think of it as a very large binary string.

Stage 2: Finding text lines and words - previous stage provided Blobs based on character outlines. These blobs are to be organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Depending on the font, the pitch maybe fixed or varying. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces

and fuzzy spaces.

Stage 3: Recognize word pass 1 - In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data, which helps in recognizing the text lower down the page.

Stage 4: Recognize word pass 2 - The reason why there is a second pass is because the adaptive classifier may have learned something useful too late to make a contribution near the top of the page. So a second pass is run over the page, in which words that were not recognized well enough earlier are recognized again.

The text will be recognized from the image. Tesseract OCR has command line Support and allows for the recognized text to be exported in the form of a text file. This concludes the internal working of an OCR.

#### 4.4.TEXT TO SPEECH SYNTHESIS :

Speech synthesis is the artificial production of human speech. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech. Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. There are a variety of TTS engines available, be it as cloud services or as open source local engines. The internal working of TTS engines can be described using the following figure :

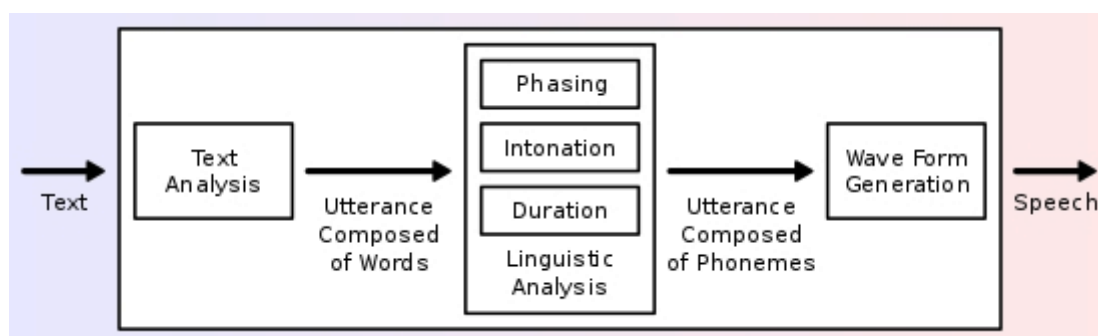


Figure 4.4.1 : Overview of a typical TTS system.[7]

A text-to-speech system is composed of two parts: a front-end and a back-end. The front-end has two major tasks. First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization, pre-processing, or tokenization. The front-end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. The process of assigning phonetic transcriptions to words is

called text-to-phoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end. The back-end then converts the symbolic linguistic representation into sound. In certain systems, this part includes the computation of the target prosody which is then imposed on the output speech. This concludes the chapter about technology.



## 5.SYSTEM DESIGN

The DAB system needs to have a hardware specification and a system flow specified. Based on the technology, the following hardware structure was devised :

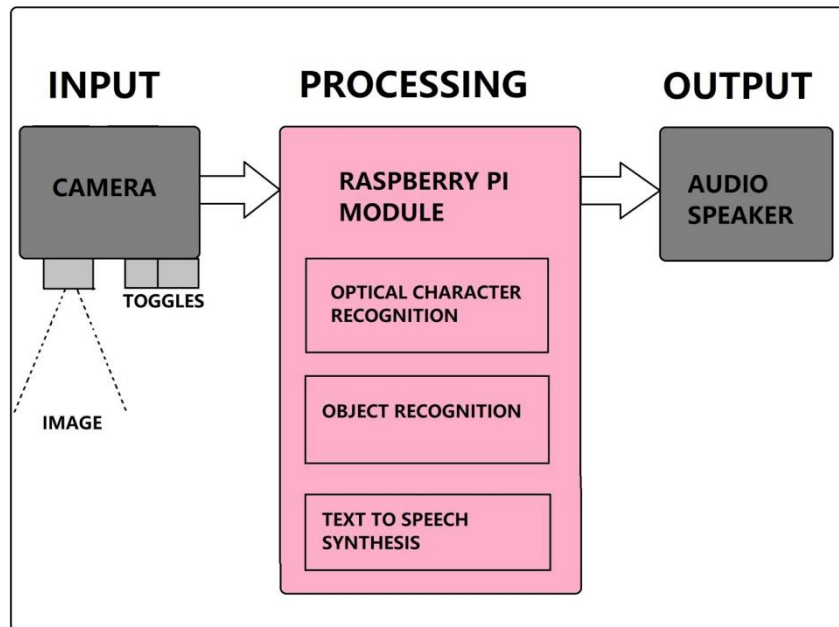


Figure 5.1: Block diagram of the DAB system

The hardware is basically divided into three parts. The input system, the processing unit and the output unit. The input units include a camera and toggles. The processing unit chosen is a Raspberry PI module. The output device will be an earpiece or speaker. All these are to be included as a system over a wearable shades. There will also be a power supply unit for the processing unit. The user, on toggling corresponding button captures an image using the camera. Based on the user's toggle, the corresponding engine, be it object recognition, face detection, or character recognition is run on the processing module. The image is processed and then the TTS engine in the processing unit outputs the sound through the speaker which can be heard by the user. The following figure can be used to understand the flow process in the DAB system.

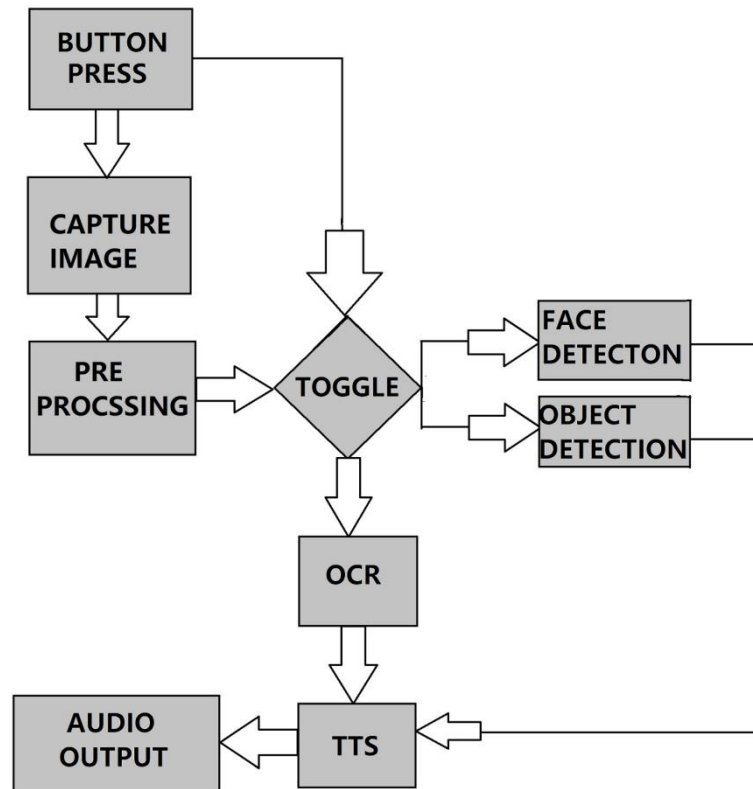


Figure 5.2: Process flow in the DAB system.

Note that the process flow begins when a button press occurs. The image is captured and preprocessing is done on it. On second toggle, the corresponding engine is given the image. The engine processes the image and passes its output to the TTS engine. It generates the corresponding audio output that is to be provided to the speaker.

There are quite a few software requirements for implementing the DAB system. For development of driver program, configuring the processing system, and configuring the engines to be used by the driver program, a few softwares are required. Open source OCR Engines can be installed in systems and can be used from python based driver programs. Python and libraries are required to create driver programs, for image capture and to perform various preprocessing techniques. Certain toolkits like dlib and face recognition modules are needed to be installed with the python virtual environment. TensorFlow Lite has pre-trained models or customizable models for object recognition. Keras could be used to train ML models. TTS engines or services are available to be accessed using API's from driver programs.

### **5.1. HARDWARE REQUIREMENTS :**

1. Raspberry pi 3b model
2. USB webcam
3. Audio speaker

### **5.2. SOFTWARE REQUIREMENTS :**

1. Tesseract OCR
2. Espeak
3. IDLE 3
4. Thonny IDE
5. Facenet
6. VSCode

## 6. IMPLEMENTATION

The Raspberry pi system is used for implementing the driver program for the DAB shades. Raspberry pi OS is installed onto the Raspberry pi. Raspberry pi OS already has Python3 and its associated IDE installed in it. The first step is to decide how each of the modules needed for the DAB system is to be implemented. The optical character recognition can be implemented using TESSERACT OCR. The face recognition module can be implemented using a trained deep neural network built by an open-source project named Openface. The object recognition module is implemented using Tensorflow lite based pre-trained object detection models. Each modules and its associated technology will be described in the following sections.

### 6.1 TESSERACT OCR :

The first step is to install the Tesseract OCR and its associated python library named pytesseract. The python wrappers in the pytesseract module could be used to configure and use the Tesseract OCR to be used from python driver programme. The tesseract version installed is 4.0.0. After installing the tesseract they can be used for character recognition.

The image to be input to the tesseract need to be preprocessed properly to improve the efficiency of the character recognition. Opencv library is used to preprocess the image. The image is thresholded, denoised, blurred and converted to greyscale before they are input to the tesseract. The following figures show the iamge going through different preprocessing states.

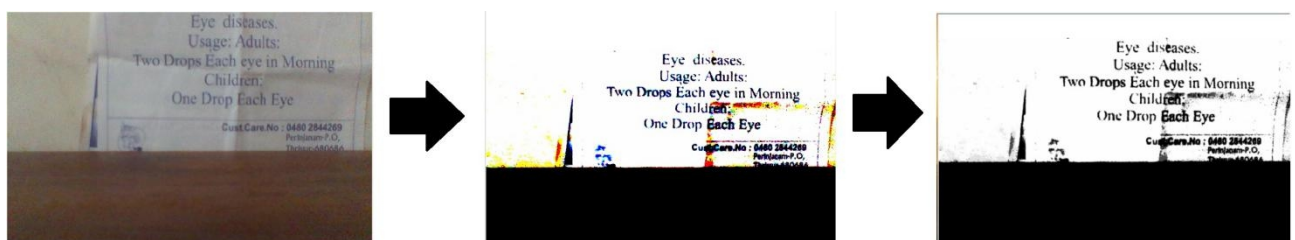
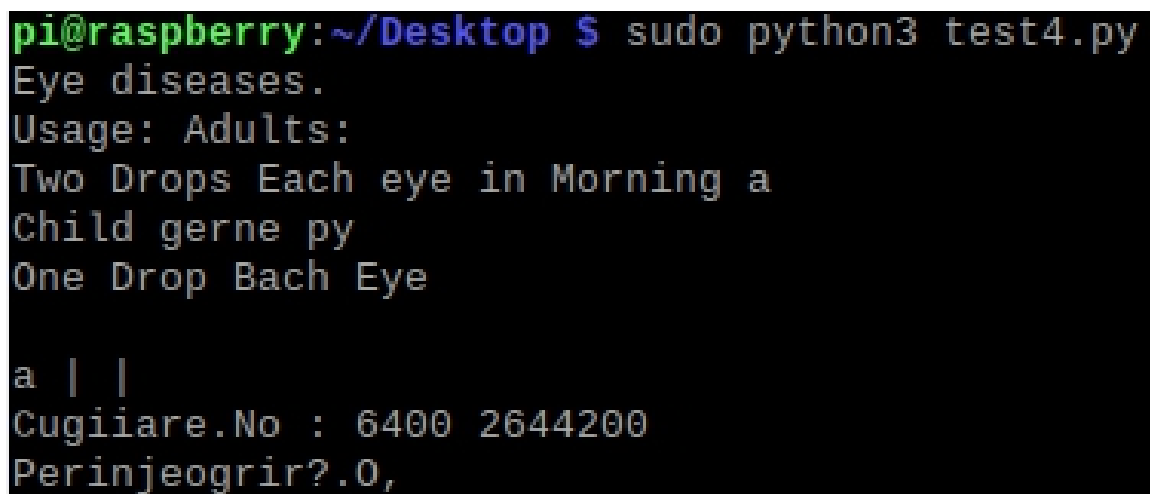


Figure 6.1: The first image is denoised, filtered and converted to greyscale.

The final greyscale image is then input to the tesseract ocr for character recognition. Tesseract assumes that its input is a binary image with optional polygonal text regions defined. Processing follows a traditional step-by-step pipeline. The first step is a connected component analysis in which outlines of the components are stored. This was a computationally expensive design decision at the time, but had a significant advantage: by inspection of the nesting of outlines, and the number of child and grandchild outlines, it is simple to detect inverse text and recognize it as easily as

black-on-white text. Tesseract was probably the first OCR engine able to handle white-on-black text so trivially. At this stage, outlines are gathered together, purely by nesting, into Blobs. Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces. Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page. Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, a second pass is run over the page, in which words that were not recognized well enough are recognized again. A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small-cap text. The character recognized could then be used as an input to the TTS engine for converting it into speech for output. The following figure shows the text recognized using python programme.



```
pi@raspberrypi:~/Desktop $ sudo python3 test4.py
Eye diseases.
Usage: Adults:
Two Drops Each eye in Morning a
Child gerne py
One Drop Bach Eye

a | |
Cugiare.No : 6400 2644200
Perinjeogrir?.0,
```

Figure 6.2: The text recognised from the image using tesseract ocr

## 6.2 GOOGLE TEXT-TO-SPEECH :

The Google Text-to-Speech API enables developers to generate human-like speech. The API converts text into audio formats such as WAV, MP3, or Ogg Opus. It also supports Speech Synthesis Markup Language (SSML) inputs to specify pauses, numbers, date and time formatting, and other pronunciation instructions. Speech Synthesis Markup Language (SSML) is used in the Text-to-Speech request to allow for more customization in the audio response by providing details

on pauses, and audio formatting for acronyms, dates, times, abbreviations, or text that should be censored. Depending on the implementation, it is needed to escape quotation marks or quotes in the SSML payload that is sent to the cloud Text-to-Speech. SSML is part of a larger set of markup specifications for voice browsers developed through the open processes of the W3C. It is designed to provide a rich, XML-based markup language for assisting the generation of synthetic speech in Web and other applications. The essential role of the markup language is to give authors of synthesizable content a standard way to control aspects of speech output such as pronunciation, volume, pitch, rate, etc. across different synthesis-capable platforms. A related initiative to establish a standard system for marking up text input is SABLE [SABLE], which tried to integrate many different XML-based markups for speech synthesis into a new one. The activity carried out in SABLE was also used as the main starting point for defining the Speech Synthesis Markup Requirements for Voice Markup Languages. Since then, SABLE itself has not undergone any further development. The intended use of SSML is to improve the quality of synthesized content. Different markup elements impact different stages of the synthesis process. The markup may be produced either automatically, for instance via XSLT or CSS3 from an XHTML document, or by human authoring. Markup may be present within a complete SSML document or as part of a fragment embedded in another language, although no interactions with other languages are specified as part of SSML itself. Most of the markup included in SSML is suitable for use by the majority of content developers. The common format SSML input is included is within a JSON object. The google text to speech is used as the TTS module for the DAB system which inputs the JSON object. Google TTS provides python API that could be configured to pass text as input to the cloud based TTS engine. The name of the google TTS module is gTTS. The output format of the audio file generated was selected to be mp3 format. The language chosen for generating the audio is english. The audio file generated can be played using any audio player application installed in raspberry pi or using os module along with aplay command from python programme.

### **6.3 FACE DETECTION :**

Face detection module is implemented using a pipeline of algorithm for detecting faces in an image, to detect incline of a face in an image and to detect the image. The method used to find the faces in an image is named HOG an acronym for Histogram of Oriented Gradients. The algorithm first makes the image in greyscale because the image is not needed to be in colour for image recognition. The pixels are grouped into 16x16sets. Each pixel group is taken and the pixels are iterated over. In each pixel, its associated neighbor pixels are considered and is compared with the itself regarding its darkness. An arrow is drawn towards the most darker neighbouring pixel. These arrows are

called gradients and they show the flow of light from dark in the entire image. After each pixel has gradients found, these gradients are counted and grouped. The gradient with the most count is taken to be the gradient of the entire pixel group. This is done to find the general change over the direction of region brightness and is unaffected even if the image is darker or brighter. The c++ cross platform software library named dlib has HOG algorithm that could be imported and used. The following clause describes the internal working of HOG algorithm in detail.

The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions called cells, for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram “energy” over somewhat larger spatial regions and using the results to normalize all of the cells in the block. We will refer to the normalized descriptor blocks as Histogram of Oriented Gradient (HOG) descriptors. Tiling the detection window with a dense grid of HOG descriptors and using the combined feature vector in a conventional SVM based window classifier gives the human detection chain. The use of orientation histograms has many precursors, but it only reached maturity when combined with local spatial histogramming and normalization in Lowe’s Scale Invariant Feature Transformation (SIFT) approach to wide baseline image matching, in which it provides an image patch descriptor for matching scale invariant keypoints. SIFT-style approaches perform remarkably well in face detection application. The Shape Context work studied alternative cell and block shapes, albeit initially using only edge pixel counts without the orientation histogramming that makes the representation so effective. The success of these sparse feature based representations has somewhat overshadowed the power and simplicity of HOG’s as dense image descriptors however the HOG/SIFT representation has several advantages. It captures edge or gradient structure that is very characteristic of local shape, and it does so in a local representation with an easily controllable degree of invariance to local geometric and photometric transformations: translations or rotations make little difference if they are much smaller than the local spatial or orientation bin size. For human detection, rather coarse spatial sampling, fine orientation sampling and strong local photometric normalization turns out to be the best strategy, presumably because it permits limbs and body segments to change appearance and move from side to side quite a lot provided that they maintain a roughly upright orientation. An overview of the feature extraction and object detection chain can be described as follows. The detector window is

tilled with a grid of overlapping blocks in which Histogram of Oriented Gradient feature vectors are extracted. The combined vectors are fed to a linear SVM for object/non-object classification. The detection window is scanned across the image at all positions and scales, and conventional non-maximum suppression is run on the output pyramid to detect object instances i.e. faces in this case.

Faces in an image could be tilted. They could be turned in an angle. So face detection from just front angle is not enough. An algorithm for face landmark estimation is used for this scenario. This machine learning algorithm finds 68 known points of a face say like chin, nosetip etc. in an image. These points are called as the landmarks. These landmarks are rotated, scaled or sheared so as to make the face in the image as centered as possible. This makes detecting faces from images a lot more easier.

The third step in the pipeline for face recognition is the face detection algorithm. An open-source project named Openface released a deep neural network based unified embedding for face detection. The model could intake an image and create 128 bit embedding that could be stored for later comparison. The technology for face detection is called FaceNet, that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.

The method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. To train, a triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method is used. The benefit of this approach is much greater representational efficiency: It achieved a state-of-the-art face recognition performance using only 128-bytes per face. On the widely used Labeled Faces in the Wild (LFW) dataset, our system achieves a new record accuracy of 99.63%. For use an image of the person to be recognized their name and a photo is to be provided to the system first. This could be used later to recognize them. When a new image is input, the faces in it are recognized and their 128 bit embedding is calculated. This newly calculated embedding is then saved along with the corresponding name into a database.

Later on, when a new image is taken and input for recognition, the faces in this image is taken and their embeddings are calculated. This newly calculated embeddings are then compared with the saved embeddings from the database. If a match is found, the corresponding name associated with that embedding is given as the recognized output string. This output string is given to the TTS module for it to be provided as audio output.



The following figures shows the face detection module.

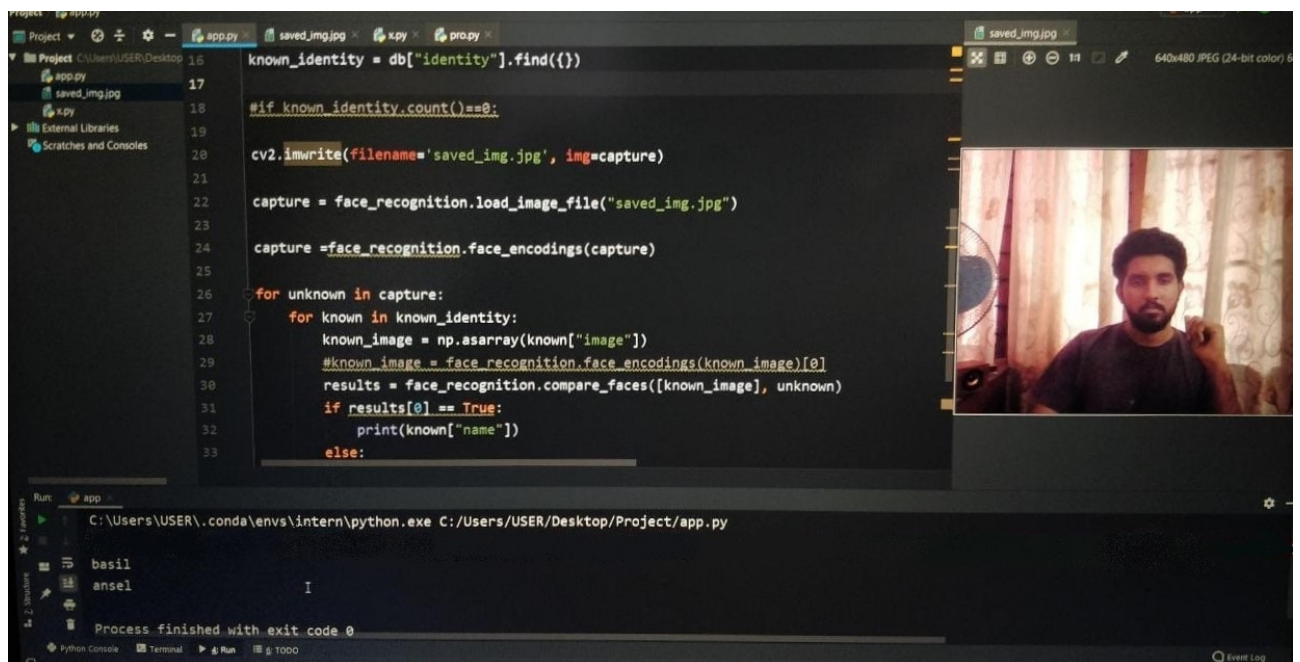


Figure 6.3: face detection module detecting a face and recognizing it.

### 6.3.1 WEB APPLICATION:

A web application is built to upload image of the face to be recognized and their name. The web application is built using a python based Flask API driver program and standard HTML and BootstrapCSS. A form is designed into the web application that receives the image in any common image file format. It also has a input section to give the face name available in that image. The image is then processed to find the 128 bit encoding of the face in that image. The Facenet module is called from the python based server program for creating the encoding before it is pushed into the database This is then connected to a Mongodb collection to store the 128 bit encoding of images provided. This embedding is also stored in a local database system. The encoding is stored in the form of a an unordered list in the database. These encoding are for later accessed for recognition. Mongodb Atlas was configured as a cloud database. The encodings when needed are converted into a 2 dimensional array with first iteration in an internal array object having the encoding and the latter holding the name of the encoding if a match is found.

The following figures shows the web application for uploading the Images.

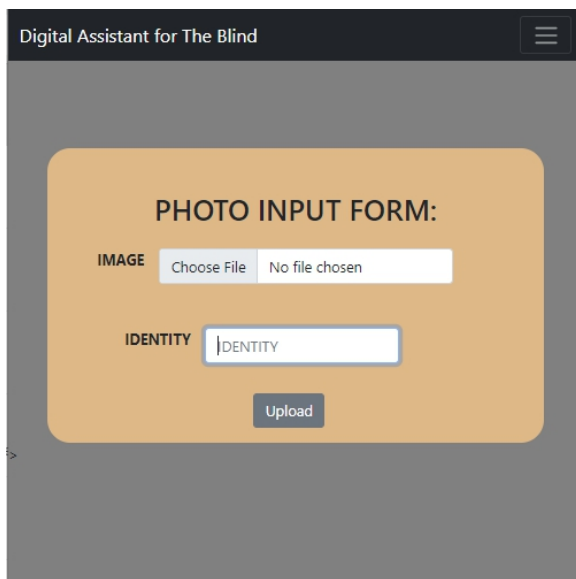


Figure6.4: The web application

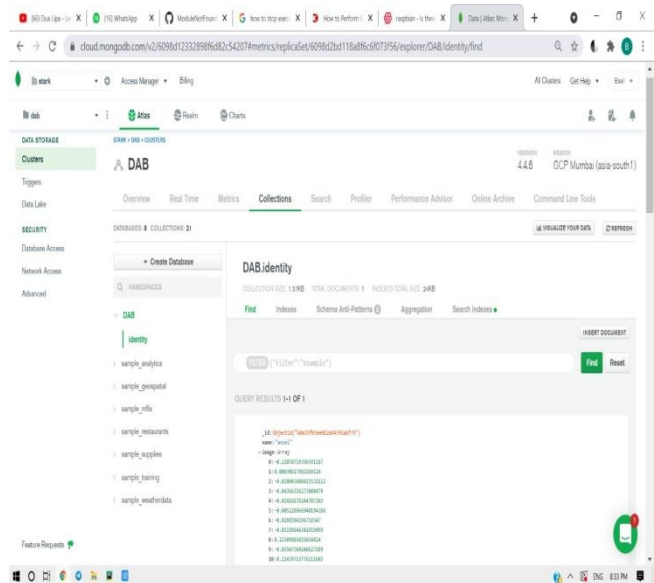


Figure6.5:MongoDB collection with embeddings.

## 6.4 OBJECT DETECTION:

Object detection is performed using Tensorflow Lite on raspberry pi using a pre-trained single shot multibox model. The object detection model used is a MobileNet V1 model that is trained on the COCO dataset. Mobilenet is a class of efficient models for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks. It introduced two simple global hyperparameters that efficiently trade off between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. It shows strong performance compared to other popular models on ImageNet classification. MobileNets is effective across a wide range of applications and use cases including object detection, finegrain classification, face attributes and large scale geo-localization. COCO is a large collection of images that have been tagged and labeled and contains over 200,000 images with around 90 object categories. The image captured is passed to the object detection model using python driver program. After recognition, a list of detected objects in a frame and a confidence score of each object is returned. This array could be used to output only the name of the detected objects and provide them as input to the TTS system.

The following figure shows the object detection in action.

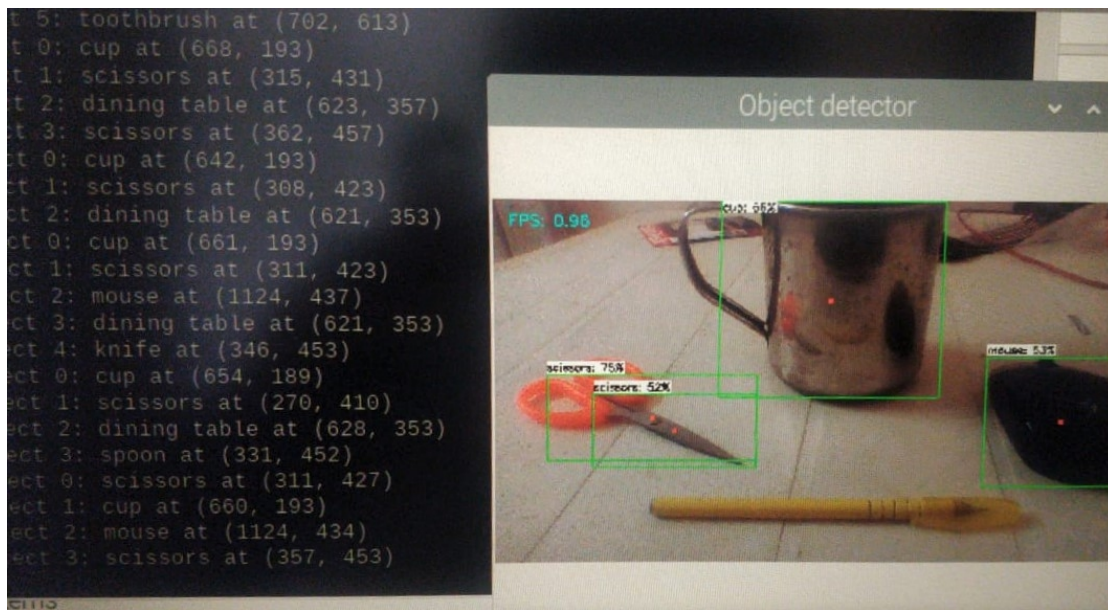


Figure6.6: object detection module detecting objects from a captured image

## 6.5 IMAGE CAPTURE AND AUDIO OUTPUT :

OpenCV is used in Python programme to capture image using a USB webcam. This image is passed to the corresponding module based on the users triggers i.e. either to OCR, object detection or face detection module. This image is then preprocessed using openCV functions. In the end, after the audio file is generated by the TTS module, it is played using ‘aplay’ call in an os.system() call. The audio could be played out through a speaker setup or a bluetooth ear phones.

## 6.6 PYTHON DRIVER PROGRAM :

All the modules are implemented but there is a need for a driver program that could be configured to switch between different modules based on toggles from the user. Python 3 based programme is used and functions are configured so that when a specific button press occurs the corresponding module is enabled for use. Various python modules were imported in installed and imported for using the python wrappers to create all the needed functions to implement the required features on Raspberry pi.

## **7.CONCLUSION**

The Project Function is to create a wearable shades having camera modules for capturing snapshots of surroundings, which can be then processed to identify the objects in the captured image which is then fed to the wearer as audio signals through the sound output.

In this, the design and development of a working model gives a smart reading help for the Blind and Visually Impaired people. The system not only helps Blind and visually impaired people, even a individual who wishes luxury and comfort, or an aged man or woman can come up with the money for this. Once mounted and configured can act as a ideal nonpublic machine for the user. The machine even finds small scale purposes in Schools, Libraries etc. This will encourages the reader and brings in them an inspiration to work on the advancements or a similar undertaking that helps the society. The computer vision and TTS technology is already used for finding technological solutions benefiting the world. So these speech synthesis and Character recognition technology can be used to support and help those people suffering from disabilities related to their speech and vision. Various derivatives of these technologies are already existing in the market, that could help them. There are a lot of time and expertise already put into the research work related to speech synthesis and computer vision technologies that every year the world is receiving better technologies in these domains.

## **8.FUTURE SCOPE**

After creation of the DAB proof of concept has led to the creation of a baseline as to how the digital smart assistant system could be implemented. Further study and research could be put into creating better and efficient modules, driver program and better hardware modules. Better and portable power supply system could be implemented. Faster and modular processing module is to be made to provide better response time after processing. The system could be improved by making the processing a hybrid system incorporating both offline as well as online support. The DAB system could have audio based control system where the users could interact with the device using voice commands.

## 9.REFERENCES

- [1]Ravi, A., et al. "Raspberry pi based Smart Reader for Blind People." *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 2020.
- [2] G. Anuradha, "A Refreshable Braille Display for the Interaction with Deafblind People" in *Middle-East Journal of Scientific Research*, IDOSI Publications,2016.
- [3] Roy Shilkrot, Jochen Huber, Meng Ee Wong, Pattie Maes, Suranga Chandima Nanayakkara, "Finger Reader" in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*,2015.
- [4] Ruman Sarkar, Smita Das, Sharmistha Roy, "SPARSHA: A low Cost Refreshable Braille for Deaf-Blind People for Communication with Deaf-Blind and Non-disabled Persons" in *International Conference on Distributed Computing and Internet Technology*,2013.
- [5] Velmurugan.D, Srilakshmi, Umamaheswari.S, Parthasarathy.S, Arun.K.R , "Hardware Implementation of Smart Reader for Visually Impaired People Using Raspberry PI" , *IJAREEIE* , Vol. 5, Issue 3, March 2016.
- [6] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
- [7] Smith, Ray. "Tesseract ocr engine." *Lecture. Google Code. Google Inc* (2007).
- [8] Wouters, Johan, et al. "Text to speech synthesis." U.S. Patent No. 7,979,280. 12 Jul. 2011.
- [9] [medium.com/@Johanni/introduction-on-ocr-242d5dbd3d01](https://medium.com/@Johanni/introduction-on-ocr-242d5dbd3d01)
- [10] [openhab.org/addons/voice/googletts/](https://openhab.org/addons/voice/googletts/)
- [11] [en.wikipedia.org/wiki/Face\\_recognition\\_\(software\)](https://en.wikipedia.org/wiki/Face_recognition_(software))

## 10. APPENDIX

```
# driver package
import RPi.GPIO as GPIO
import time

# face_detection package
import pymongo
import ssl
import face_recognition
import cv2
import os
from gtts import gTTS
import vlc

#object_detection package
import argparse
import numpy as np
import sys
import time
from threading import Thread
import importlib.util

#ocr package
from PIL import Image
import pytesseract

def face_detection():
    print("Starting Face Detection")
    language = 'en'

    camera = cv2.VideoCapture(0)
    for c in range(30):
        return_value, capture = camera.read()
    del (camera)
    client = pymongo.MongoClient(
        "mongodb+srv://basil:project%40123@dab.izlar.mongodb.net/DAB?retryWrites=
true&w=majority",
        ssl_cert_reqs=ssl.CERT_NONE)
    # client = pymongo.MongoClient("mongodb://localhost:27017/")

    db = client["DAB"]
    known_identity = db["identity"].find({})

    # if known_identity.count()==0:

    cv2.imwrite(filename='saved_img.jpg', img=capture)

    capture = face_recognition.load_image_file("saved_img.jpg")

    capture = face_recognition.face_encodings(capture)
    if capture == []:
        print("not found")
        myobj = gTTS(text="person not found", lang=language, slow=False)
        myobj.save("sound.mp3")
        media = vlc.MediaPlayer("sound.mp3")
```

```

        media.play()
    for unknown in capture:
        for known in known_identity:
            known_image = np.asarray(known["image"])
            # known_image = face_recognition.face_encodings(known_image)[0]
            results = face_recognition.compare_faces([known_image], unknown)
            if results[0] == True:
                myobj = gTTS(text=known["name"], lang=language, slow=False)
                myobj.save("sound.mp3")
                media = vlc.MediaPlayer("sound.mp3")
                media.play()
                print(known["name"])
            else:
                print("not found")
                myobj = gTTS(text="person not found", lang=language,
slow=False)
                myobj.save("sound.mp3")
                media = vlc.MediaPlayer("sound.mp3")
                media.play()
#face module

def object_detection():
    print("Starting object Detection")

    class VideoStream:

        def __init__(self, resolution=(640, 480), framerate=30):
            # Initialize the PiCamera and the camera image stream
            self.stream = cv2.VideoCapture(0)
            ret = self.stream.set(cv2.CAP_PROP_FOURCC,
cv2.VideoWriter_fourcc(*'MJPG'))
            ret = self.stream.set(3, resolution[0])
            ret = self.stream.set(4, resolution[1])

            # Read first frame from the stream
            (self.grabbed, self.frame) = self.stream.read()

            # Variable to control when the camera is stopped
            self.stopped = False

        def start(self):
            # Start the thread that reads frames from the video stream
            Thread(target=self.update, args=()).start()
            return self

        def update(self):
            # Keep looping indefinitely until the thread is stopped
            while True:
                # If the camera is stopped, stop the thread
                if self.stopped:
                    # Close camera resources
                    self.stream.release()
                    return

                # Otherwise, grab the next frame from the stream
                (self.grabbed, self.frame) = self.stream.read()

```



```

def read(self):
    # Return the most recent frame
    return self.frame

def stop(self):
    # Indicate that the camera and thread should be stopped
    self.stopped = True

MODEL_NAME = "object_detection"
GRAPH_NAME = "detect.tflite"
LABELMAP_NAME = "labelmap.txt"
min_conf_threshold = float(0.5)

imW, imH = 1280, 720

from tf.lite_runtime.interpreter import Interpreter

# Get path to current working directory
CWD_PATH = os.getcwd()

# Path to .tflite file, which contains the model that is used for object
detection
PATH_TO_CKPT = os.path.join(CWD_PATH, MODEL_NAME, GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH, MODEL_NAME, LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]

if labels[0] == '???':
    del (labels[0])

# Load the Tensorflow Lite model.
interpreter = Interpreter(model_path=PATH_TO_CKPT)

interpreter.allocate_tensors()

# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5

# Initialize video stream
videostream = VideoStream(resolution=(imW, imH), framerate=30).start()
time.sleep(1)

final = []
for q in range(3):

```

```

# Grab frame from video stream
frame1 = videostream.read()

# Acquire frame and resize to expected shape [1xHxWx3]
frame = frame1.copy()
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
frame_resized = cv2.resize(frame_rgb, (width, height))
input_data = np.expand_dims(frame_resized, axis=0)

# Normalize pixel values if using a floating model (i.e. if model is
non-quantized)
if floating_model:
    input_data = (np.float32(input_data) - input_mean) / input_std

# Perform the actual detection by running the model with the image as
input
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()

# Retrieve detection results
classes = interpreter.get_tensor(output_details[1]['index'])[0] #
Class index of detected objects
scores = interpreter.get_tensor(output_details[2]['index'])[0] #
Confidence of detected objects

# Loop over all detections and draw detection box if confidence is above
minimum threshold
for i in range(len(scores)):
    if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

        object_name = labels[int(classes[i])] # Look up object name
from "labels" array using class index
        if object_name not in final:
            final.append(object_name)

    if final==[]:
        final.append("no object")

# Clean up
cv2.destroyAllWindows()
videostream.stop()
for o in final:
    myobj = gTTS(text=o, lang="en", slow=False)
    myobj.save("sound.mp3")
    media = vlc.MediaPlayer("sound.mp3")
    media.play()
    time.sleep(1)
#object module

def character_detection():

    print("Starting Optical Character Recognition")

    camera = cv2.VideoCapture(0)
    for c in range(20):
        return_value, img = camera.read()

```

```

del (camera)
cv2.imwrite(filename='test.jpg', img=img)
#
img=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)
ret, thresh1 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
denoise = cv2.fastNlMeansDenoisingColored(thresh1, None, 10, 10, 7, 21)
cv2.imwrite('gt.png', denoise)
blur = cv2.GaussianBlur(denoise, (5, 5), 0)
grey = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
cv2.imwrite('hulk.png', grey)
original = pytesseract.image_to_string(grey, config=' ')
original = original.replace("\n", " ")
print(original)
try:
    sound = gTTS(original, lang="en")
    sound.save("sound.mp3")
except:
    original="no text"
    sound = gTTS(original, lang="en")
    sound.save("sound.mp3")

media = vlc.MediaPlayer("sound.mp3")
media.play()
#ocr module

GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # Set pin 10 to be an
input pin and set initial value to be pulled low(off)
GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(12, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

while True: # Run forever

    if GPIO.input(10) == GPIO.HIGH:
        face_detection()
        time.sleep(1)

    elif GPIO.input(11) == GPIO.HIGH:
        object_detection()
        time.sleep(1)

    elif GPIO.input(12) == GPIO.HIGH:
        character_detection()
        time.sleep(1)

GPIO.cleanup() # Clean up

```