A project report on

"NAVIGATION ASSISTANCE AND OCR FOR BLIND"

Submitted to

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

(VTU), BELAGAVI

in partial fulfilment of the requirements for the award of degree

BACHELORS OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

TEJASRI K	**1SB16CS043**
SANDHYA S	**1SB16CS083**
SACHIN KUMAR K S	**1SB16CS081**
DHANANJAYAN S	**1SB16CS031**

Under the valuable guidance of

Prof. SOWMYA AM

Assistant Professor, CSE

Department of CSE, SSCE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
SRI SAIRAM COLLEGE OF ENGINEERING,
Anekal, Bengaluru-562106

SRI SAIRAM COLLEGE OF ENGINEERING

Anekal, Bengaluru – 562106

Department of Computer Science and Engineering

CERTIFICATE

Certified that project work entitled "**NAVIGATION ASSISTANCE AND OCR FOR BLIND**" is a bonafide work carried out by

TEJASRI K	**1SB16CS043**
SANDHYA S	**1SB16CS083**
SACHIN KUMAR K S	**1SB16CS081**
DHANANJAYAN S	**1SB16CS031**

In partial fulfilment for the award of the Bachelor of Engineering in Computer Science and Engineering. Of the Visvesvaraya Technological University, Belgaum during the year 2018-19. It is certified that all corrections/suggestions indicated for the internal assessment have been incorporated in the report deposited in the department library. The project has been approved as it satisfies the academic requirements in respect of the project work prescribed for Bachelor of Engineering Degree.

Signature of the Guide Prof. Sowmya AM Asst. Prof., CSE Dept, SSCE.	Signature of the HOD Dr. G. Manjula HOD, CSE Dept, SSCE.	Signature of the Principal Dr. B. Shadaksharappa Principal, SSCE.

External Viva

Name of the Examiners: 1. _____

2._____

DECLARATION

We, the students of the eighth semester of Computer Science and Engineering, Sri Sairam College of Engineering, Anekal, declare that the work entitled **"NAVIGATION ASSISTANCE AND OCR FOR BLIND"** has been successfully completed under the guidance of Prof. Sowmya AM, Computer Science and Engineering Department, Sri Sairam College of Engineering, Anekal. This dissertation work is submitted to Visvesvaraya Technological University in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science during the academic year 2018 - 2019. Further, the matter embodied in the project report has not been submitted previously by anyone for the award of any degree or diploma to any university.

Place:

Date:

TEAM MEMBERS

1. **TEJASRI. K**
2. **SANDHYA. S**
3. **SACHIN KUMAR. K S**
4. **DHANANJAYAN. S**

ACKNOWLEDGEMENT

The successful completion of any task will be incomplete without complementing those who made possible and whose guidance and encouragement made my efforts successful. I offer my sincere thanks to Sri Sairam College of Engineering, Anekal, Bangalore for providing all kinds of facilities to carry out my project work.

I take immense pleasure in thanking our beloved chairman MJF.LN.LEO MUTHU and Dr. B. SHADAKSHARAPPA, Principal, Sri Sairam College of Engineering, Anekal, Bangalore for providing me all the facilities for successful completion of my project.

I am grateful to Dr. G Manjula, HOD, Department of Computer Science and Engineering, Sri Sairam College of Engineering, Anekal, Bangalore, for his constant motivation, encouragement and guidance to make this project a success.

I would like to express my humble thanks to my project guide Prof Sowmya A M, Assistant Professor, Department of Computer Science and Engineering, Sri Sairam College of Engineering, Anekal, Bangalore, for guiding me and having facilitated me to complete my project work successfully.

I would like to thank all the teaching and non-teaching staff of the Department of Computer Science and Engineering College, Bangalore, for their help and encouragement.

Last, but not the least, I would like to express my hearty gratitude to my family and friends, for having given me constant encouragement and moral support throughout the project work.

ABSTRACT

Visual impairment and illiterate, or have a learning disability is one of the biggest drawbacks for humanity, especially in this day and age when information and people is interconnected a lot by text messages (electronic and paper based) rather than talking. There is a need for a convenient text reader that is reasonable and readily available to the blind community. This work in this research these images are converted into audio output. It is mainly used in the field of research in Character recognition and product recognition, Artificial intelligence and computer vision. In this research, as the recognition process is done using Tensorflow Object Detection and OpenCV library. It recognizes character using convolution network algorithm and python programming, this paper describes the design, implementation and experimental results of the device. Here we are aiming to develop a offline application where cameras are utilised from mobile to detect the text and object. This device consists of three modules, image processing for text detection, object detection and to convert detected things to voice. The Photo OCR stands for "PHOTO OPTICAL CHARACTER RECOGNITION". Here Smart Recognition Application is designed to help blind people in their navigation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Blind people have to face a lot of difficulty in performing day-to-day tasks. But the advanced technology era has tried to make their lives a bit easier. This paper talks about one such idea which can be a step towards achieving the same goal. The paper describes the process of developing an android compatible application named **Lookout** which can capture image in front of them and can read out any text, if present in the captured image. The technology uses own developed OCR with an advancement in precision and accuracy. The conversion of text to voice is done using the already existing TTS framework. The work described in this paper strictly focuses on detection of text and street signs from the image. This can help blind people in reading documents and navigating by hearing the street signs. Hence, technology if used in a correct way can definitely be a boon for the visually impaired people.

Object Detection is the process of finding real-world object instances like car, bike, TV, flowers, and humans in still images or Videos. It allows for the recognition, localization, and detection of multiple objects within an image which provides us with a much better understanding of an image as a whole. It is commonly used in applications such as image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).

Image will be taken in two modes - continuous and triggered. In continuous mode the images will be taken one after the other as the images appear in front of the camera. This mode is generally used when the user is walking. Example - Reading of street signs while navigating. In triggered mode, the image will be taken only when the user activates a trigger. Otherwise no image and no processing will take place. In text detection phase the algorithm will decide whether the image taken contains text or not. It does so with the help of tons of examples - Images which contain text, which is called dataset in machine learning terminology. The algorithm outputs “yes” if the image contains text and “no” otherwise.

1.1 Problem definition

Near-sighted participants are facing difficulty in seeing the objects in front of them and illiterate people report several difficulties for understanding printed text or captured text using current technology, including problems with alignment, focus, precision.

1.2 Aim of the project

Our project mainly focuses on how to get applications to read the text and detect objects to the purest in images that we take. We have used tts technology which is mainly for text detection Both text and image which are recognised with the help of the algorithm are converted to speech. It can not only help machines to understand our images and texts better but also can help blind people in navigation.

1.3 Existing system

Stereo vision-based obstacle detection This method [1] describes a portable vision-based obstacle detection system, intended for use by blind people. The system combines an obstacle detection system designed for AGVs with recalibration of ground position and a Kalman Filter based model of the person's walking movement. Obstacle detection is achieved through comparison of the disparity seen with that expected from the position of the ground. Recalibration of ground position is made by plane fitting in the ground region. Motion estimation using two visual methods and the use of an inclinometer is described. The results show satisfactory success in all parts of the system. The system described in [1] provides part of an obstacle avoidance capability. It will form a major part of the mobility function of a larger project, Autonomous System for Mobility, Orientation, Navigation and Communication (ASMONC) which aims to provide a full navigation and mobility capability for blind and partially sighted people. Other sensors, such as sonar, are likely to be included to improve robustness. Sonar is reliable for detecting large and high obstacles, but it is not suitable for detecting small obstacles standing on the ground plane, as the angular resolution is insufficient to distinguish between the ground plane and the obstacle. Therefore, a major requirement for the vision system is to detect small obstacles, and this is investigated here. This system is aiming to be capable of detecting obstacles of around 10 cm in height, at a 3–5 m distance, with the vision system. The ASMONC project uses a backpack, worn by the user, to hold the system electronics and sensors. A Loughborough Sound Images colour image processing module, based around the TI C40 processor, is used to perform all of

the processing tasks required for vision. It is connected directly to two Sony NDP40BY/E cameras. The two cameras are mounted on two rigid arms, which extend over either shoulder of the user from the backpack. Camera mounts were designed which allow minor adjustment of the camera orientation for camera alignment. The sonar sensors are fixed on the chest and belt. The starting point of the vision system draws on the Ground Plane Obstacle Detection (GPOD) algorithm, which has been used successfully for obstacle avoidance in mobile robots. GPOD uses a pair of cameras to determine features which do not lie on the ground plane. It characterizes the ground plane by a parameterization based on measurement of disparity, rather than projection into the external world. This improves robustness to calibration errors. It includes an initial calibration stage in which the ground plane parameters are extracted. Features originally known to lie in the ground are tracked, and the ground plane re-parameterized at each iteration. This allows adaptation either to genuine changes in the plane's parameters or to changes in camera position. We show that the use of a Kalman Filter to track both the ground plane features and suspected obstacles provides sufficient stability for obstacle detection. Another major requirement is the measurement and prediction of camera motion to provide the parameters which guide ground plane recalibration. The image stabilization methods used in products such as camcorders are unacceptable, because this system need to identify all six degrees of movement of the cameras, and cannot simply use an image-based readjustment. The measurements are fed into a Kalman Filter gait model to improve the reliability of the estimates. And also, problem is facing in Near-sighted participants and illiterate people report several difficulties for understanding printed text or captured text using current technology, including problems with alignment.

1.3.1 Advantages

- It is portable vision-based obstacle detection system, intended for use by blind people.
- It aims to provide a full navigation and mobility capability for blind and partially sighted people.

1.3.2 Disadvantages

- Near-sighted participants and illiterate people report several difficulties for understanding printed text or captured text using current technology, including problems with alignment.
- Sonar is reliable for detecting large and high obstacles, but it is not suitable for detecting small obstacles.

1.4 Proposed system

This approach will solve major problems that faces by blind peoples. Because it is comprising some important units in to a single system. The proposed scheme can have the following capabilities. Near-sighted participants and illiterate people report several difficulties for understanding printed text or captured text using current technology, including problems with alignment, to overcome we are proposed this system. The proposed prototype works on the principles of image processing, machine learning and TTS (The Technology of Text to Speech) framework. We have eliminated the components like Raspberry pi, Display unit, cameras, speakers, switch and HDMI which are present in existing system. Instead we made use of all the available resources in a mobile. The functions of each components are described below.

Google Vision API which is used for image and text recognition and convert it to speech gtts framework is used. Running the applications that captures the real-world objects and texts like laptop, cell phone, Furniture etc. Cameras are used for capturing images brought in front of them. Here we have used mobiles integrated cameras. Speakers are used for voice output in which detected object and images are converted to speech. Timer or voice assistance is used to is used to capture the image through camera.

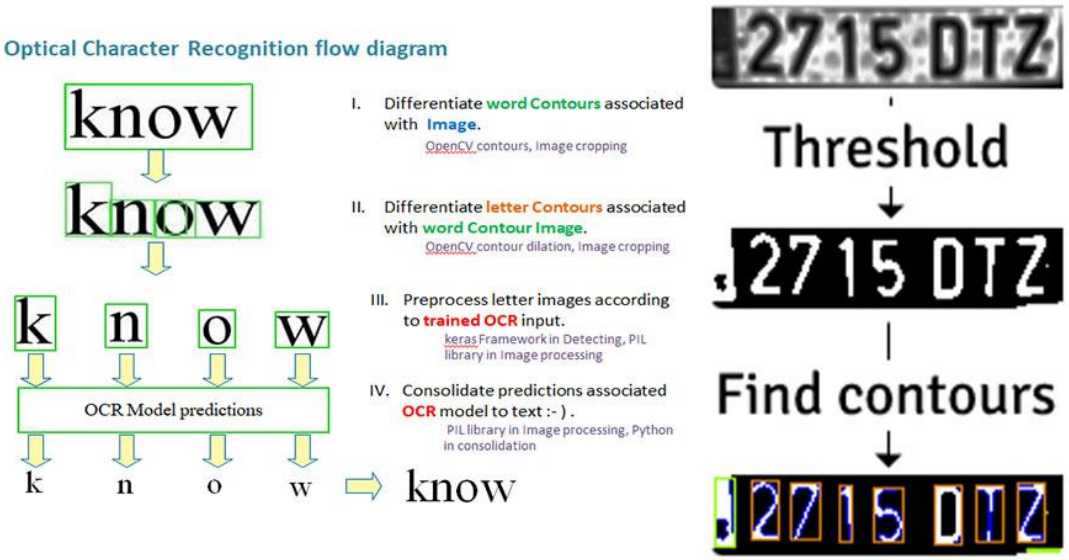


Fig 1.1 Classification steps

Image processing is used to scan the captured image and extract the portion of the image which contains text. Machine learning is used to process the extracted image portion (which contains text) to extract the text (character by character) and predict the text. The TTS framework takes input as the extracted text and converts that to speech, so that a blind person who is not deaf can listen and can get an idea of the scene in front of him/her. Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License. Tesseract has Unicode (UTF8) support, and can recognize more than 100 languages "out of the box". Tesseract supports various output formats: plain text, hOCR (HTML), PDF, invisible-text-only PDF, TSV. The master branch also has experimental support for ALTO (XML) output.

The field of computer vision has seen tremendous development in the recent past leading to a host of practical applications in a wide variety of industries and use cases. Optical Character Recognition (OCR) is one such application of computer vision with the potential to automate many tedious but necessary tasks. OCR technology can be used to process digital documents (PDFs, scanned documents, images of documents and the like), far more efficiently than humans can. In a nutshell, OCR can "read" a document and convert images of text into actual text. Current state-of-the-art algorithms are capable of near-flawless recognition of printed text, with handwriting recognition

When the application is started the camera of the mobile is turned on the starts to capture the image and text. The image is captured with the help of voice command or pressing shutter button and is recognised with the machine learning algorithms integrated into the models. The recognised image and text are detected through voice output. This approach will solve major problems that faced by blind peoples. Because it is comprising some important units in to a single system as an application into our mobile phone which intern reduces the use of carrying a separate device for this purpose. The proposed scheme can have the following capabilities. Near- sighted participants and illiterate people report several difficulties for understanding printed text or captured text using current technology, including problems with alignment and to overcome this we have proposed this system.

1.5 Organization of Report

- Chapter 1: **Introduction** tells about the issue explanation, existing and proposed frameworks.

- Chapter 2: **Literature survey** manages all the discoveries and perceptions which are led as attainability study before real improvement of the venture. The part additionally manages the clarification of existing methodologies.

- Chapter 3: **Software Requirement Specification** lists the equipment and programming determination for this task. It additionally portrays the general depiction of venture, item viewpoint, client characteristics and particular necessities. Here diverse outline requirements, interface and execution necessities clarified.

- Chapter 4: **System Design** manages the propelled programming building where the whole stream of the venture is spoken to by expert information stream charts and grouping graphs.

- Chapter 5: **Implementation** area clarifies coding rules and framework upkeep for the venture.

- Chapter 6: **Testing** manages the different sorts of experiments to demonstrate the legitimacy of the venture.

- Chapter 7: **Result Analysis** clarifies in insights about the result of the test and contrasts it and the outcome acquired in existing framework.

- Chapter 8: **Conclusion and Future work** this segment depicts the synopsis of the related work and future improvements of the proposed framework.

- **References**: This segment basically highlights all the diaries and contextual analysis papers being eluded for the advancement cycle of the venture.

- **Appendix**: It contains the **snapshot** that basically manages the graphical yield and client interface of the application, **publication** points of interest and **sample code** is given.

CHAPTER 2

LITERATURE SURVEY

The overwhelming volume of paper-based data in corporations and offices challenges their ability to manage documents and records. Computers, working faster and more efficiently than human operators, can be used to perform many of the tasks required for efficient document and content management. Computers understand alphanumeric characters as ASCII code typed on a keyboard where each character or letter represents a recognizable code. However, computers cannot distinguish characters and words from scanned images of paper documents. Therefore, where alphanumeric information must be retrieved from scanned images such as commercial or government documents, tax returns, passport applications and credit card applications, characters must first be converted to their ASCII equivalents before they can be recognized as readable text. Optical character recognition system (OCR) allows us to convert a document into electronic text, which we can edit and search etc. It is performed off-line after the writing or printing has been completed, as opposed to on-line recognition where the computer recognizes the characters as they are written. For these systems to effectively recognize hand-printed or machine printed forms, individual characters must be well separated. This is the reason why most typical administrative forms require people to enter data into neatly spaced boxes and force spaces between letters entered on a form. Without the use of these boxes, conventional technologies reject fields if people do not follow the structure when filling out forms, resulting in a significant overhead in the administration cost. Optical character recognition for English has become one of the most successful applications of technology in pattern recognition and artificial intelligence. OCR is the machine replication of human reading and has been the subject of intensive research for more than five decades. To understand the evolution of OCR systems from their challenges, and to appreciate the present state of the OCRs, a brief historical survey of OCRs is in order now. Depending on the versatility, robustness and efficiency, commercial OCR systems may be divided into the following four generations [Line, 1993; Pal & Chaudhuri, 2004]. It is to be noted that this categorization refers specifically to OCRs of English language.

First Generation:

First generation OCR systems Character recognition originated as early as 1870 when Carey invented the retina scanner, which is an image transmission system using photocells. It is used as an aid to the visually handicapped by the Russian scientist Tyurin in 1900. However, the first-generation machines appeared in the beginning of the 1960s with the development of the digital computers. It is the first time OCR was realized as a data processing application to the business world [Mantas, 1986]. The first-generation machines are characterized by the "constrained" letter shapes which the OCRs can read. These symbols were specially designed for machine reading, and they did not even look natural. The first commercialized OCR of this generation was IBM 1418, which was designed to read a special IBM font, 407. The recognition method was template matching, which compares the character image with a library of prototype images for each character of each font.

Second generation:

Next generation machines were able to recognize regular machine-printed and handprinted characters. The character set was limited to numerals and a few letters and symbols. Such machines appeared in the middle of 1960s to early 1970s. The first automatic letter sorting machine for postal code numbers from Toshiba was developed during this period. The methods were based on the structural analysis approach. Significant efforts for standardization were also made in this period. An American standard OCR character set: OCR-A font (Figure 2.1) was defined, which was designed to facilitate optical recognition, although still readable to humans. A European font OCR-B (Figure 2.2) was also designed.

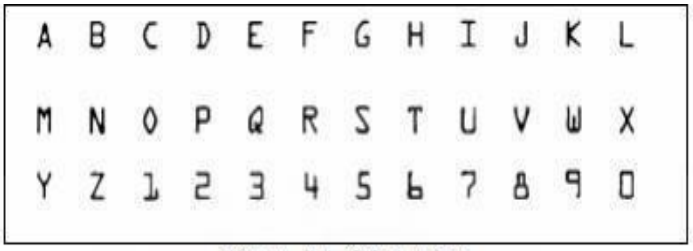


Figure 2.1 OCR-A font

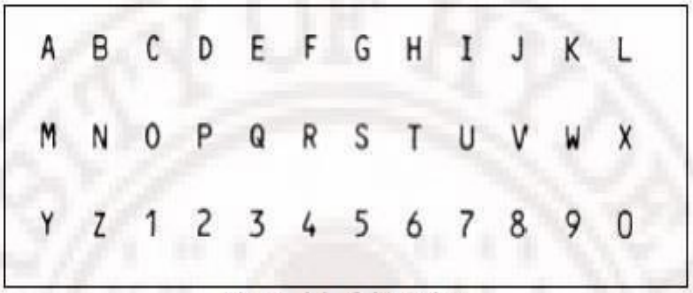


Figure 2.2 OCR-B font

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification (SRS) is a central report, which frames the establishment of the product advancement process. It records the necessities of a framework as well as has a depiction of its significant highlight. An SRS is essentially an association's seeing (in composing) of a client or potential customer's framework necessities and conditions at a specific point in time (generally) before any genuine configuration or improvement work. It's a two-way protection approach that guarantees that both the customer and the association comprehend alternate's necessities from that viewpoint at a given point in time.

The composition of programming necessity detail lessens advancement exertion, as watchful audit of the report can uncover oversights, mistaken assumptions, and irregularities ahead of schedule in the improvement cycle when these issues are less demanding to right. The SRS talks about the item however not the venture that created it, consequently the SRS serves as a premise for later improvement of the completed item. The SRS may need to be changed; however, it does give an establishment to proceeded with creation assessment.

In straightforward words, programming necessity determination is the beginning stage of the product improvement action. The SRS means deciphering the thoughts in the brains of the customers – the information, into a formal archive – the yield of the prerequisite stage. Subsequently the yield of the stage is a situated of formally determined necessities, which ideally are finished and steady, while the data has none of these properties.

3.1 Software and Hardware Requirements

3.1.1. Software Requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. Software that are required for this project can be listed as follows:

- 2 GB RAM minimum and 8 GB RAM recommended.
- 2 GB of available disk space minimum.
- 4 GB recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
- 1200x800 minimum screen resolution.
- Java Development Kit (JDK) 8
- For accelerated emulator: 64-bit operating system and Intel® processor with support for Intel® VTx, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality
- Android Studio 3.0
- Wamp Server Version 2.5
- MySQL Database version 5.6

3.1.2 Hardware Requirements

Hardware requirements is most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements. Hardware that is required for this project can be listed as follows:

- Laptop/PC for Android Application Development.
- Server (Windows 7/8/10 (32-bit or 64-bit)).
- Android Mobile with minimum android version 4.0

3.2 System Requirements

3.2.1 Functional Requirements

The Functional Requirements Definition reports and tracks the fundamental data needed to successfully characterize business and practical necessities. The Functional Requirements Definition report is made amid the Planning Phase of the undertaking. Its target group is the undertaking supervisor, task group, venture support, customer/client, and any partner whose data/regard into the necessity's definitions procedure is required. The practical prerequisites incorporate the accompanying: We have classified these functional requirements as follow:

1. Taking/ choosing the desired text or object image.
2. Recognition of the text or object.
3. Copying the text and objects for different uses.

1.Taking/ choosing the desired text or object image:

For the mobile application: The most important thing here is the use of an Android mobile phone and its camera. The user can take a picture of a text image or choose one from the mobile's directory. The user must use a camera of typical resolution and take a picture of a text image or choose one from existing ones in his phone.

- 1: Android Mobile Phone.
- 2: Text Image or Object Image.
- 3: Images Dataset.

2. Recognition of the text or object:

The text and objects will be recognized from the image taken by the mobile's camera or from any chosen image from the phone directory.

The text will be recognized and ready to be used.

- 1: Recognition of the text and object from the image.
- 2: Ready to be used.

3. Copying the text and object for different uses:

Once the text is recognized and ready to be used, the user will be able to copy, edit, and modify it. He/she may also be able to retrieve the data from the image and store it directly on the phone such as the contact information taken from a Business Card. The recognized text may be retrieved to make it editable or store it directly on the phone.

- 1: Copy the text from the text from the image and modify it.
- 2: Retrieve data from the text image and store it on the phone.

3.2.2 Non-Functional Requirements:

- **Reliability**

The framework ought to be dependable and solid in giving the functionalities. When a client has rolled out a few improvements, the progressions must be made unmistakable by the framework. The progressions made by the Programmer ought to be unmistakable both to the Project pioneer and in addition the Test designer.

- **Security**

Aside from bug following the framework must give important security and must secure the entire procedure from smashing. As innovation started to develop in quick rate the security turned into the significant concern of an association. A great many dollars are put resources into giving security. Bug following conveys the greatest security accessible at the most noteworthy execution rate conceivable, guaranteeing that unapproved clients can't get to imperative issue data without consent. Bug following framework issues diverse validated clients their mystery passwords so there are limited functionalities for all the clients.

- **Maintainability**

The framework observing and upkeep ought to be basic and target in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.

- **Performance**

The framework will be utilized by numerous representatives all the while. Since the framework will be facilitated on a solitary web server with a solitary database server out of sight, execution turns into a noteworthy concern. The framework ought not succumb when numerous clients would be utilizing it all the while. It ought to permit quick availability to every last bit of its clients. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not be any irregularity at the same time.

- **Portability**

The framework should to be effectively versatile to another framework. This is obliged when the web server, which s facilitating the framework gets adhered because of a few issues, which requires the framework to be taken to another framework.

- **Scalability**

The framework should be sufficiently adaptable to include new functionalities at a later stage. There ought to be a typical channel, which can oblige the new functionalities.

- **Flexibility**

Flexibility is the capacity of a framework to adjust to changing situations and circumstances, and to adapt to changes to business approaches and rules. An adaptable framework is one that is anything but difficult to reconfigure or adjust because of diverse client and framework prerequisites. The deliberate division of concerns between the trough and motor parts helps adaptability as just a little bit of the framework is influenced when strategies or principles change.

CHAPTER 4:

SYSTEM DESIGN

The system design process builds up general framework building design. Programming outline includes speaking to the product framework works in a shape that may be changed into one or more projects. The prerequisite indicated by the end client must be put in a systematically manner. Outline is an inventive procedure; a great configuration is the way to viable framework. The framework "Outline" is characterized as "The procedure of applying different systems and standards with the end goal of characterizing a procedure or a framework in adequate point of interest to allow its physical acknowledgment". Different configuration components are taken after to add to the framework. The configuration detail portrays the components of the framework, the segments or components of the framework and their appearance to end-clients.

4.1 Design Consideration

The reason for the design is to arrange the arrangement of the issue determined by the necessities report. This stage is the initial phase in moving from issue to the arrangement space. As such, beginning with what is obliged; outline takes us to work towards how to fulfil those needs. The configuration of the framework is maybe the most basic component influencing the nature of the product and has a noteworthy effect on the later stages, especially testing and upkeep. Framework outline depicts all the significant information structure, document arrangement, yield and real modules in the framework and their Specification is chosen.

4.2 System Architecture

The architectural configuration procedure is concerned with building up a fundamental basic system for a framework. It includes recognizing the real parts of the framework and interchanges between these segments. The beginning configuration procedure of recognizing these subsystems and building up a structure for subsystem control and correspondence is called construction modelling outline and the yield of this outline procedure is a portrayal of the product structural planning.

The proposed architecture for this system is given below. It shows the way this system is designed and brief working of the system.

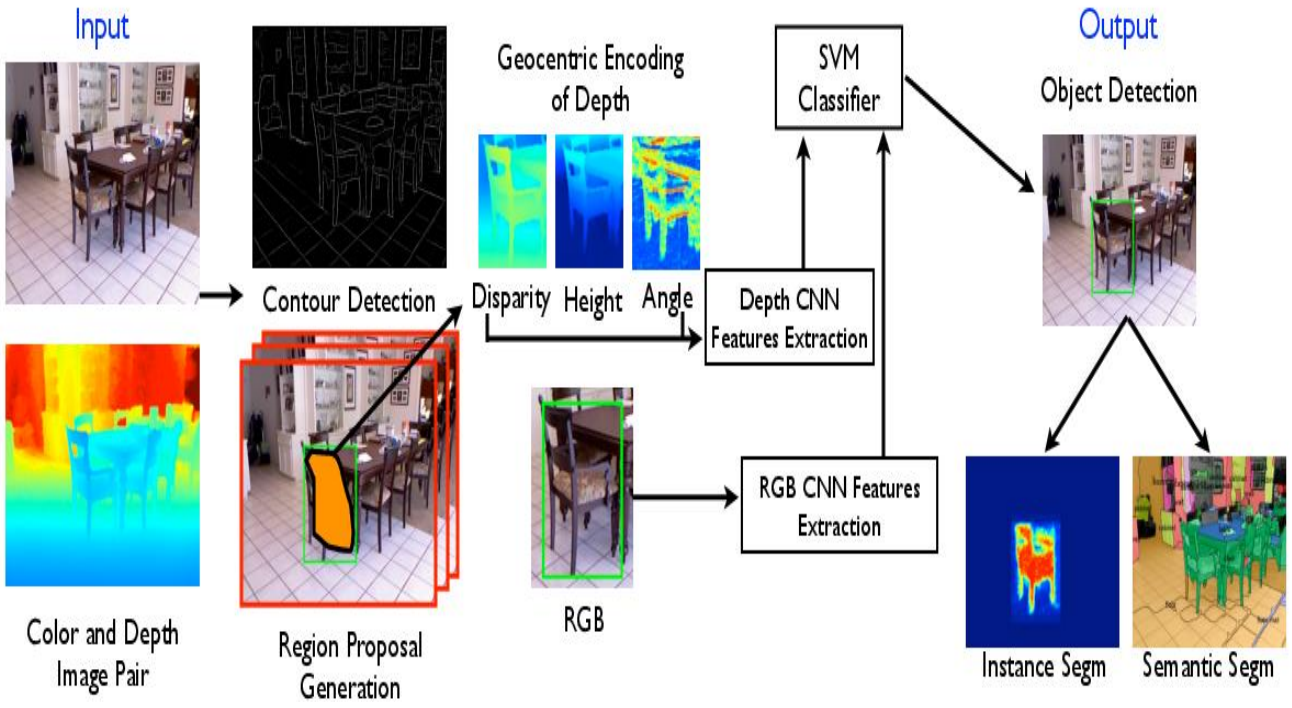


Fig 4.1 System Architecture

BLOCK DIAGRAM:

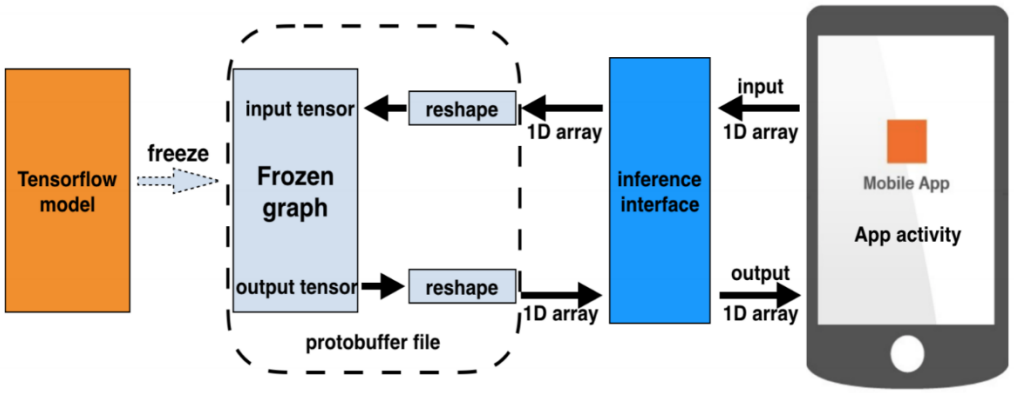


Fig 4.2 Block Diagram of Prototype

For the implementation of CNN model in Android device, we used the interface provided by Tensorflow First, the CNN model parameters need to be trained and saved into a protobuffer file. Basically, the way to save the CNN graph is to freeze all variables into constants with well-trained values and save them by their names. Then with Android interface tool (called “Interface”), Android app can load tensor with values, run the graph and read tensor output values. However, current interface only support loading values and reading outputs in the format of 1-D array. So, the input node/output node in the graph should be designed to be 1D array to accommodate that. The app is designed with a streaming video from the camera, and each image frame is passed to the CNN model for object detection. And then the detected results are marked with boxes in real time. To accommodate the 8 fps of the default frame rate in Android device, we need the total processing time to be less than 125 ms. To train a robust classifier, we need a lot of pictures which should differ a lot from each other. So, they should have different backgrounds, random object, and varying lighting conditions. In order to label our data, we need some kind of image labelling software. LabelImg is a great tool for labelling images. With the images labelled, we need to create TFRecords that can be served as input data for training of the object detector. Namely, the xml_to_csv.py and generate_tfrecord.py files. To train the model we use faster_rcnn_inception, which just like a lot of other models will start with sample config. trained model need to generate an inference graph, which can be used to run the model.

The software is installed using command lines. The first setup is to download the installation script, second command is to convert it to executable form and the last command starts the script which does the rest of the installation work. The basic framework is this implemented system that captures an image, extracts only the region of interest (i.e. region of the image that contains text) and converts that text to audio. It is developed using a Google Vision API and TensorFlow Object Detection API.

This project presents a sample system for recognition of text present in the image using mobile application. The system agenda consists of five well-designed components: Image acquisition, Image pre- processing, Text extraction, Text to speech conversion and Speech output. This paper proposed a system using Android Application Model for perusing the images from stereotypical forms – such as street signs, hospital signs, and bus numbers as well as more variable forms such as shop signs, house numbers, and billboards. Here we are using Ada Boost Algorithm for treating the visual information and converting into audio speech. The proposed is small, lightweight, efficient, cost effective and of course user-pleasant the accuracy of the mobile in the conversion efforts is better, mainly due to the high-resolution camera built in the device.

Developing technology and in future expansions of this project, the system can be provided with a good and high-resolution camera contrasted with the one used in this project, and we anticipate, this will improve its inevitability. We predict more work will be produced in this critical area of assistive technology, and project that future transportable gadgets will have easy to use and built in mechanism as reading assistance for the blind, similar, to the mobile based solution presented here. Users should capture image and then system read out the text from image. It will be more applicable for persons those are going through visual surgery. It can be suitable for road side text recognition so that visually impaired person can travel alone. this paper proposed a system using Model for scanning the images from stereotypical forms – such as street signs, hospital signs, and bus numbers –as well as more variable forms such as shop signs, house numbers, and billboards. Here they are using Ada Boost Algorithm for processing the visual information and converting into audio with default sound. image. The input text image is converted from RGB to GRAY form. In case of coloured text image, initially it is converted to Gray form and then to binary in order to segregate 0`s and 1`s, whereas 0`s represents non-text part and 1`s represents the textual part The converted GRAY image will be then converted into binary form in order to match with the machine. The input image is then segmented in the form of binary. The conversion of binary codes is done by assuming white spaces as 1`s and black spaces as 0‟s.

FLOW CHART DIAGRAM:

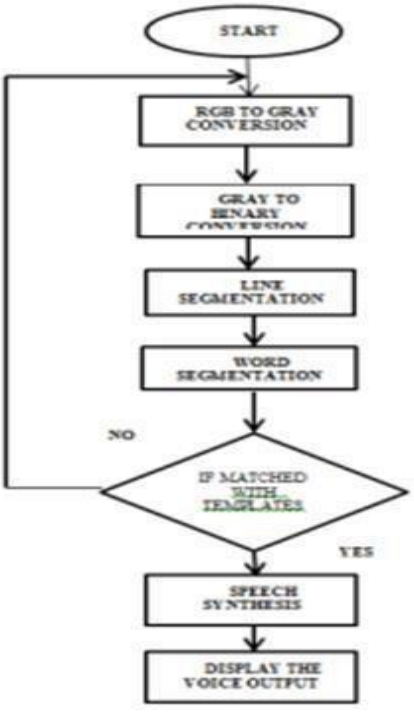


Fig 4.3 Representation of flowchart for text detection

The general form of text will be in Matrix form where f1 indicates the first line f2 indicates the second line and so on .The words of the first line can be represented as a11, a12…..a1n where the number "1" indicates the first line and "n" indicates the number of words in the corresponding line. The general form of representing any word from any line can be given as a where, m represents the Corresponding line and "n" represents the corresponding word. Likewise, the same form will be followed for representing each line from any part of the paragraph. For example, if second word of the third line to read it will be given as a 32 of the third line. The word segmentation is done with the pixel of each letter with the suitable machine language for example, in case, if first word from the first line has to read, it can be segmented as where a11 represents first word from first line and f1- first line. Similarly, any word from any line can be called for the conversion of text to speech. The segmented word from line will be then matched with a template which is pre-loaded already.

SEQUENCE DIAGRAM:

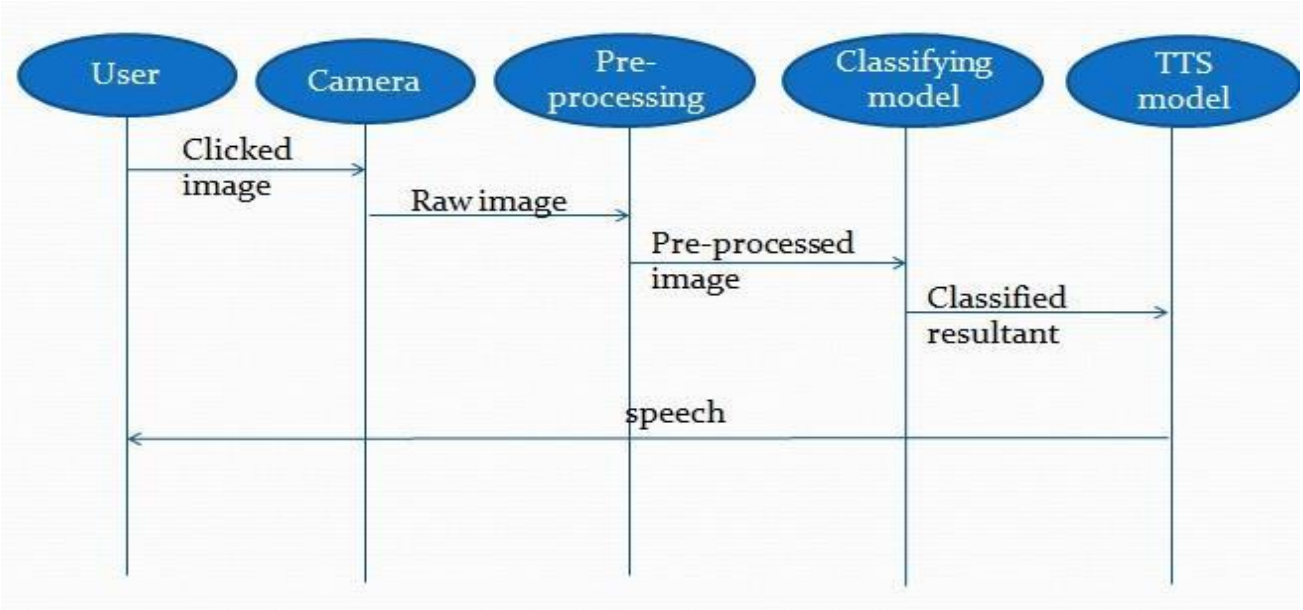


Fig 4.4 Sequence representation

The system agenda consists of five well-designed components: Image acquisition, Image pre-processing, Text extraction, Text to speech conversion and Speech output. This paper proposed a system using Model for perusing the images from stereotypical forms – such as street signs, hospital signs, and bus numbers –as well as more variable forms such as shop signs, house numbers, and billboards. Here we are using Ada Boost Algorithm for treating the visual information and converting

into audio speech. The proposed system helps visually impaired, illiterate, or have a learning disability to read product the project aims to implement a reading aid that is small, lightweight, efficient, cost effective and of course user. However, the accuracy of the mobile in the conversion efforts is better, mainly due to the high-resolution camera built in the device.Users should capture image and then system read out the text from image. It will be more applicable for persons those are going through visual surgery. It can be suitable for road side text recognition so that visually impaired person can travel alone. this paper proposed a system Mobile Application Model for scanning the images from stereotypical forms – such as street signs, hospital signs, and bus numbers –as well as more variable forms such as shop signs, house numbers, and billboards. Here they are using Ada Boost Algorithm for processing the visual information and converting into audio with default sound. image.

DATAFLOW DIAGRAM:

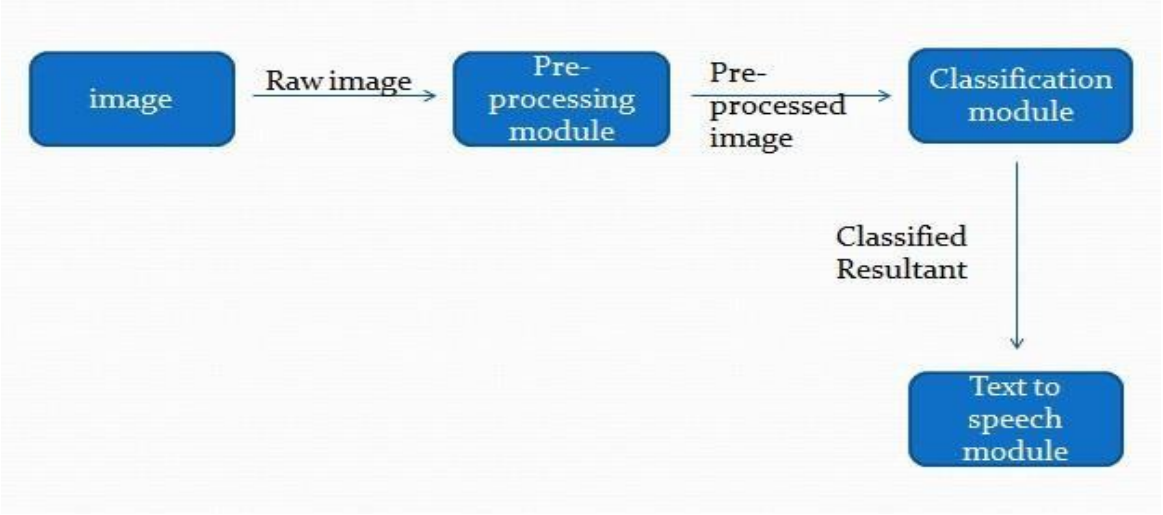


Fig 4.5 Dataflow diagram

The method comprises of three sections - capturing of image, extracting of text from image and finally converting text to voice. An image can be taken with the help of a camera, here we use a Mobile camera. The image taken is sent for further processing where text is extracted from the image and predicted. From there, the predicted text is sent to TTS framework for converting text to speech. An image can be taken with the help of a camera, here we use integrated mobile camera. The image taken is sent for further processing where text is extracted from the image and predicted. From there, the predicted text is sent to TTS framework for converting text to speech.

CHAPTER 5

IMPLEMENTATION

Implementation is the phase of the undertaking when the hypothetical configuration is transformed out into a working framework. Subsequently it can be thought to be the most discriminating stage in accomplishing a fruitful new framework and in giving the client, certainty that the new framework will work and be viable. The usage stage includes watchful arranging, examination of the current framework and its limitations on execution, planning of systems to accomplish changeover and assessment of changeover strategies. In this stage, the design or design changes are introduced and made operational in a specific situation. The stage is introduced after the framework has been tried and acknowledged by the client and framework administrator. Exercises in this stage incorporate notice of usage to end clients, execution of the already characterized preparing arrangement, information passage or discussion, and post usage survey. Implementation is an essential stage in the improvement of the task where the product configuration is acknowledged as a situated of system units. The items that are recognized in the outline stage are actualized and capacities, which control these articles, are figured it out. Subsequent to performing framework investigation took after by outlining and coding, execution is done to check whether the result of the venture is of course.

The implementation has many stages and the stages are

- Methodical planning.
- Examination of constraints in system.
- Assessment of methods and changes in the system.
- Platform selection.

Implementation of any product is constantly continued by critical choices with respect to choice of the stage, the dialect utilized, and so forth these choices are frequently affected by a few elements, for example, genuine environment in which the framework meets expectations, the velocity that is needed, the security concerns, and other execution choices that have been made before the usage of this venture. They are as per the following:

- Choosing of programming language.
- Guidelines for coding.

5.1 Selection of Coding Language

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions- **Python 2 and Python 3**. Both are quite different.

i. **Python3**

Python was developed by Guido van Rossum in early 1990‟s and its latest version is 3.7.1, we can simply call it as Python3. Python 3.0 was released in 2008. and is interpreted language i.e it's not compiled and the interpreter will check the code line by line.

Reason for increasing popularity

1. Emphasis on **code readability, shorter codes**, ease of writing
2. Programmers can express logical concepts in **fewer lines** of code in comparison to languages such as C++ or Java.
3. Python supports **multiple** programming paradigms, like object-oriented, imperative and functional programming or procedural.
4. There exist inbuilt functions for almost all of the frequently used concepts.
5. Philosophy is "Simplicity is the best"

Language features

- **Interpreted**
- There are no separate compilation and execution steps like C and C++.
- Directly run the program from the source code.
- Internally, Python converts the source code into an intermediate form called bytecodes which is then translated into native language of specific computer to run it.
- No need to worry about linking and loading with libraries, etc.
- **Platform Independent**
- Python programs can be developed and executed on multiple operating system platforms. Python can be used on Linux, Windows, Macintosh, Solaris and many more.
- **Free and Open Source;** Redistributable **High-level Language**

- In Python, no need to take care about low-level details such as managing the memory used by the program.
- **Simple**
- Closer to English language; Easy to Learn
- More emphasis on the solution to the problem rather than the syntax
- **Embeddable**
- Python can be used within C/C++ program to give scripting capabilities for the program's users.
- **Robust**:
- Exceptional handling features
- Memory management techniques in built **Rich Library Support**
- The Python Standard Library is varying vast.
- Known as the **"batteries included"** philosophy of Python; It can help do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, email, XML, HTML, WAV files, cryptography, GUI and many more.
- Besides the standard library, there are various other high-quality libraries such as the Python Imaging Library which is an amazingly simple image manipulation library.

ii. Java

Java is a predominantly static typed programming language (though it also supports dynamic typing for some OOPs concepts like polymorphism) developed by James Gosling, Mike Sheridan, and Patrick Naughton in the year June 1991 at Sun Microsystems [K-3]. Many of the constructs were influenced from its counterparts in C and C++ programming languages [K-3]. Though it was heavily influenced by C and C++, it was different in its own ways. It eliminated pointers as its designers thought that developers are using pointers the way it was not meant to be.

Language features

- In Java, everything is an Object. Java can be easily extended since it is based on the **Object model**.
- Java is designed to be **easy** to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- With Java's **secure** feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

- Java compiler generates an **architecture-neutral** object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean **portability** boundary, which is a POSIX subset.
- Java being **robust** makes an effort to eliminate error-prone situations by emphasizing mainly on compile time error checking and runtime checking.
- With Java's **multithreaded** feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- With the use of Just-In-Time compilers, Java enables **high performance.**
- Java is designed for the **distributed** environment of the internet.
- Java is considered to be more **dynamic** than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time.

5.2 Input Design

The input is provided to the system in the form of image taken from the two cameras. These cameras are fitted on a helmet. One of the cameras is used for the purpose of detecting text. Another camera is used for detecting objects in front of it. In order to take a snap, a switch provided in the system must be pressed.

The camera used has image sensor quality CMOS sensor. Image resolution interpolated to 12 mega pixels with 6 light sensors. Image control colour saturation, brightness, sharpness and brightness is adjustable. Anti-flicker 50Hz, 60Hz or outdoor. Resolution hardware is 500K pixels. Image quality is RGB24 or I420. Exposure modes are Auto or manual and angle of view is 58 Degree. Interface used is USB2.0. Frame rate is 30 fps (max).

5.3 Output Design

The output of the system is in the form of speech. This speech contains information about the detected text or object in front of it. In case of object detection, the object detected is marked with a green rectangular box, to indicate that it has been detected. The detected object is then converted to speech. Conversion of text to speech is done using gTTS implemented in python.

Google Text-to-Speech is a screen reader application developed by Google for its Android operating system. It powers applications to read aloud (speak) the text on the screen which support many languages. Text-to-Speech may be used by apps such as Google Play Books for reading books aloud, by Google Translate for reading aloud translations providing useful insight to the pronunciation of words, by Google Talkback and other spoken feedback accessibility-based applications, as well as by third-party apps. Users must install voice data for each language.

5.4 Development Environment

Tensorflow:

Tensorflow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning application such as neural networks, etc. It is used for both research and production by Google. Tensorflow is developed by the Google Brain team for internal Google use. It is released under the Apache License 2.0 on November 9,2015. Tensorflow is Google Brain's second-generation system.1st Version of Tensorflow was released on February 11, 2017.While the reference implementation runs on single devices, Tensorflow can run on multiple CPU's and GPU (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on various platforms such as64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. The architecture of Tensorflow allows the easy deployment of computation across a variety of platforms (CPU's, GPU's, TPU's), and from desktops - clusters of servers to mobile and edge devices. 22 Tensorflow computations are expressed as stateful dataflow graphs. The name Tensorflow derives from operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

5.5 Modules

Entire work is divided into three modules. First module deals with the capturing of image and pre-processing. When the switch is pressed, static image is taken. The captured image is pre-processed. The captured RGB image is converted to grayscale. The resultant is in the form of a binary matrix. Second module deals with the classification of captured image – to identify text or objects. For identifying text, tesseract software is used. For identifying various objects, a pre-trained model is loaded. Input in the form of captured image is provided to the model. Consequently, the input image is classified into different objects. The third module deals with the conversion of text into speech using gTTS. The resultant of the second module is in the form of string (containing the classified text or object), which is fed to gTTS to be converted to speech.

The modules are:

- image capturing and pre-processing
- classification of image
- conversion of text to speech

5.5.1 Module I: Image capturing and pre-processing

The image is captured when the switch is pressed. The switch is connected to the cameras and is used to take still images, when pressed. The captured image is in the form of RGB. This is converted to grayscale. The result is a binary matrix.

5.5.2 Module II: Text recognition

For text detection, Google's tesseract is used. **Tesseract** is an optical character recognition engine for various operating systems. It is a free software, released under the Apache License, Version 2.0,and development has been sponsored by Google since 2006. In Python, we use the pytesseract module. It is simply a wrapper around the command line tool with the command line options specified using the config argument. The basic usage requires us to first read the image using OpenCV and pass the image for **image to string** method of the pytesseract class along with the language.

5.5.3 Module III: Object recognition

For object detection, Tensorflow library is used. Tensorflow is Google's Open Source Machine Learning Framework for dataflow programming across a range of tasks. Nodes in the graph represent mathematical operations, while the graph edges represent the multi-dimensional data arrays (**tensors**) communicated between them. The model is trained on the COCO dataset. COCO stands for **Common Objects in Context;** this dataset contains around 330K labelled images.

5.5.4 Model IV: Conversion of text to speech

The result of the above module is in the form of string containing information about the identified text or object. This string is fed to gTTS api for conversion to speech. Python implementation of gTTS api is used. This output can be listened with help of earphones.

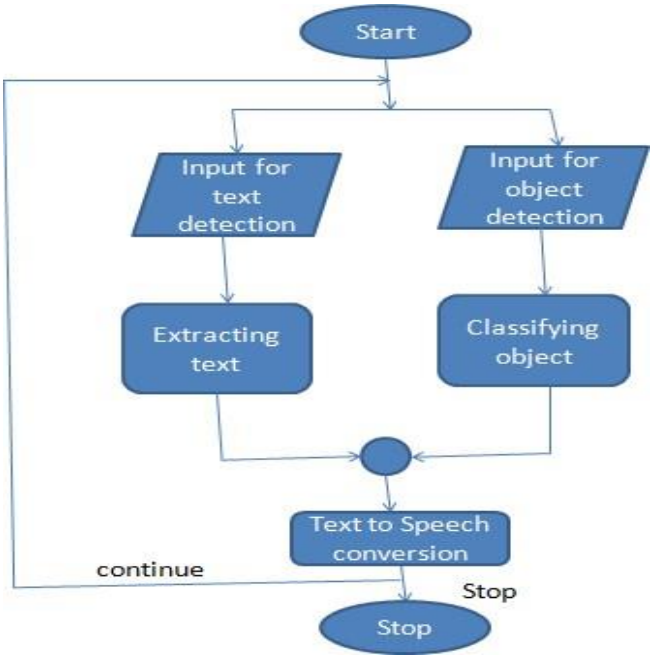


Fig 5.1 Representational flowchart

5.6 System Maintenance

The system can be placed on any convenient prop as per the requirement. The type of camera can be maintained according to the requirement. The switch can be placed on any convenient place as required.

CHAPTER 6

TESTING

Testing is an essential stage in the advancement life cycle of the item. This is the stage, where the remaining lapses, if any, from all the stages are identified. Thus, testing performs an extremely discriminating part for quality certification and guaranteeing the dependability of the product.

Amid the testing, the system to be tried was executed with a situated of experiments and the yield of the project for the experiments was assessed to figure out if the project was executing of course. Slips were discovered and adjusted by utilizing the beneath expressed testing steps and remedy was recorded for future references. Consequently, a progression of testing was performed on the framework, before it was prepared for usage.

It is the procedure used to help recognize the accuracy, fulfilment, security, and nature of created PC programming. Testing is a procedure of specialized examination, performed for the benefit of partners, i.e. proposed to uncover the quality-related data about the item as for connection in which it is planned to work. This incorporates, however is not restricted to, the procedure of executing a project or application with the goal of discovering lapses.

The quality is not an outright; it is worth to some individual. In light of that, testing can never totally build up the rightness of discretionary PC programming; Testing outfits a "feedback" or examination that looks at the state and conduct of the item against detail. An essential point is that product testing ought to be recognized from the different control of Software Quality Assurance (SQA), which incorporates all business process zones, not simply testing.

There are numerous ways to deal with programming testing, yet viable testing of complex items is basically a procedure of examination not only a matter of making and taking after routine method. Albeit a large portion of the scholarly procedures of testing are almost indistinguishable to that of audit or investigation, the word testing is indicated to mean the dynamic examination of the item putting the item through its paces. A portion of the normal quality traits incorporate capacity, unwavering quality, productivity, versatility, viability, similarity and ease of use. A decent test is now and then depicted as one, which uncovers a slip; nonetheless, later thinking recommend that a decent test is one which uncovers data of enthusiasm to somebody who matters inside of the undertaking group.

6.1 Testing Types

A methodology for framework testing coordinates framework experiments and outline systems into very much arranged arrangement of steps that outcomes in the fruitful development of programming. The testing method must co-work test arranging, experiment configuration, test execution, and the resultant information accumulation and assessment.

A procedure for programming testing must suit low-level tests that are important to confirm that a little source code fragment has been effectively actualized and additionally abnormal state tests that are accept significant framework capacities against client prerequisites.

6.1.1 Unit Testing

Unit testing centres check exertion on the unit of programming configuration (module). Utilizing the unit test arrangements, arranged in the configuration period of the framework improvement as an aide, essential control ways are tried to uncover lapses inside of the limit of the modules.

The interfaces of each of the module were tried to guarantee legitimate stream of the data into and out of the modules under thought. Limit conditions were checked. Every single autonomous way was practiced to guarantee that all announcements in the module are executed in any event once and all lapse taking care of ways were tried. Every unit was completely tried to check in the event that it may fall in any conceivable circumstance. This testing was done amid the programming itself. Toward the end of this testing stage, every unit was discovered to be working palatably, as respect to the normal yield from the module.

Unit Test cases are given below:

Table 6.1 Unit Test Case 1

S l # Test Case	UTC-1
Name of Test	Taking static image with the help of voice command
Item being tested	Voice interaction with the cameras
Sample input	Voice Command
Expected output	Static image of the scene taken in front of the camera
Actual output	Same as expected
Remarks	Successful

Table 6.2 Unit Test Case 2

S1 # Test Case	UTC-2
Name of Test	Image pre-processing test
Item being tested	Image taken from cameras
Sample input	Static image
Expected output	RGB image converted to grayscale
Actual output	Same as expected
Remarks	Successful

Table 6.3 Unit Test Case 3

S1 # Test Case	UTC-3
Name of Test	Character recognition
Item being tested	Text recognition module
Sample input	Image containing a single character
Expected output	Correct classification of the character in the image
Actual output	Same as expected
Remarks	Successful

Table 6.4 Unit Test Case 4

S1 # Test Case	UTC-4
Name of Test	Word recognition
Item being tested	Text recognition module
Sample input	Image containing a single word
Expected output	Correct extraction of the word in the image
Actual output	Same as expected
Remarks	Successful

Table 6.5 Unit Test Case 5

S1 # Test Case	UTC-5
Name of Test	Sentence recognition
Item being tested	Text recognition module
Sample input	Image containing a single sentence
Expected output	Correct extraction of the sentence in the image
Actual output	Same as expected
Remarks	Successful

Table 6.6 Unit Test Case 6

S1 # Test Case	UTC-6
Name of Test	Object recognition
Item being tested	Object recognition module
Sample input	Sample images containing pre-defined objects
Expected output	Correct classification of the objects
Actual output	Same as expected
Remarks	Successful

Table 6.7 Unit Test Case 7

S1 # Test Case	TC-7
Name of Test	Txt to Speech conversion
Item being tested	GTTS module
Sample input	String
Expected output	Input string converted to speech
Actual output	Same as expected
Remarks	Successful

6.1.2 Integration Testing

Information can be lost over an interface: one module can have an unfriendly impact on another's sub capacities, when joined may not deliver the fancied real capacity; worldwide information structures can exhibit issues. Combination testing was a symmetric method for the building the project structure while in the meantime directing tests to uncover mistakes connected with the interface. All modules are consolidated in this testing step. At that point the whole program was tried in general. Integration testing is another part of testing that is by and large done with a specific end goal to reveal mistakes connected with stream of information crosswise over interfaces. The unit-tried modules are gathered together and tried in little portion, which make it less demanding to confine and right blunders. This methodology is proceeded with unit I have incorporated all modules to frame the framework overall.

Table 6.8 Integration Test Case 8

S1 # Test Case	C-8
Name of Test	Real time object recognition
Item being tested	Object recognition module
Sample input	Sampling real time images containing pre-defined objects
Expected output	Correct classification of the objects
Actual output	Same as expected
Remarks	Successful

Table 6.9 Integration Test Case 9

S1 # Test Case	C-9
Name of Test	Real time text recognition
Item being tested	Text detection module
Sample input	Real time images containing text
Expected output	Correct extraction of text and conversion of resultant to speech
Actual output	Same as expected
Remarks	Successful

Table 6.10 Integration Test Case 10

S1 # Test Case	C-10
Name of Test	Real time object recognition
Item being tested	Object detection module
Sample input	Real time images containing text
Expected output	Correct classification of the objects in image
Actual output	Same as expected
Remarks	Successful

6.1.3 System Testing

After the Integration testing, the product was totally collected as a bundle; interfacing slips have been uncovered and rectified the last arrangement of programming tests, approval tests start. Approval test succeeds when the product capacities in a way that can be sensibly expected by the client. Here the framework was tried against framework necessity determination. Framework testing was really a progression of diverse tests whose basic role was to completely practice the PC based framework. Albeit every test has an alternate reason all work to confirm that all framework components have been legitimately incorporated and perform distributed capacities.

Table 6.11 System Test Case 11

S1 # Test Case	STC-11
Name of Test	System testing for text detection
Item being tested	Text detection module
Sample input	Real time images containing text
Expected output	Correct extraction of text and conversion of same to speech
Actual output	Same as expected
Remarks	Successful

Table 6.12 System Test Case 12

S1 # Test Case	STC-12
Name of Test	System testing for object detection
Item being tested	Object detection module
Sample input	Real time images containing pre-defined objects
Expected output	Correct classification of objects and conversion of same to speech
Actual output	Same as expected
Remarks	Successful

6.1.4 Performance Testing

The performance testing guarantee that the yield being created inside of as far as possible and time taken for the framework accumulating, offering reaction to the clients and solicitation being send to the framework keeping in mind the end goal to recover the outcomes.

6.1.5 Validation Testing

The validation testing can be characterized from multiple points of view, yet a basic definition is that. Acceptance succeeds when the product capacities in a way that can be sensibly expected by the end client.

Black Box testing

In this testing by knowing the inside operation of an item, tests can be led to guarantee that "all apparatuses network", that is the inner operations performs as per detail and all interior segments have been sufficiently worked out. It on a very basic level spotlights on the practical necessities of the product.

Black Box testing is done to locate the accompanying

- Incorrect or missing capacities
- Interface mistakes
- Errors in outside database access
- Performance mistake
- Initialization and cessation error

White Box Testing

This testing is additionally called as glass box testing normally done by code designers. In this testing, by knowing the predefined capacity that an item has been intended to perform test can be directed that shows every capacity is completely operation in the meantime scanning for lapses in every capacity. It is an experiment plan strategy that uses the control structure of the procedural configuration to determine experiments. Premise way testing is a white box testing.

This permits the tests to

- Check whether every free way inside of a module have been practiced in any event once.
- Exercise every coherent choice on their false sides.
- Execute all circles and their limits and inside of their limits.
- Exercise the interior information structure to guarantee their legitimacy.
- Ensure whether all conceivable legitimacy checks and legitimacy lookups have been given to accept information.

6.1.6 Acceptance Testing

This is the last phase of testing process before the framework is acknowledged for operational utilization. The framework is tried inside of the information supplied from the framework procurer instead of recreated information. Client Acceptance testing is a basic period of any venture and requires huge cooperation by the end client. It additionally guarantees that the framework meets the utilitarian prerequisites.

CHAPTER 7

RESULTS

The paper presents a description of making an application which can detect text from image and convert it to voice. This can make the life of blind people a bit easier in performing daily tasks like reading a document, moving from one place to another. The application makes use of android mobile with a camera. Also, it makes use of machine learning concepts in predicting the text present in the image and in increasing the accuracy of this prediction. There is also a usage of image processing in detecting text portion in the captured image. The proposed application makes use of TTS framework to convert text to speech. Combining all the technologies in a systematic way gives this application as a result which can help blind people.

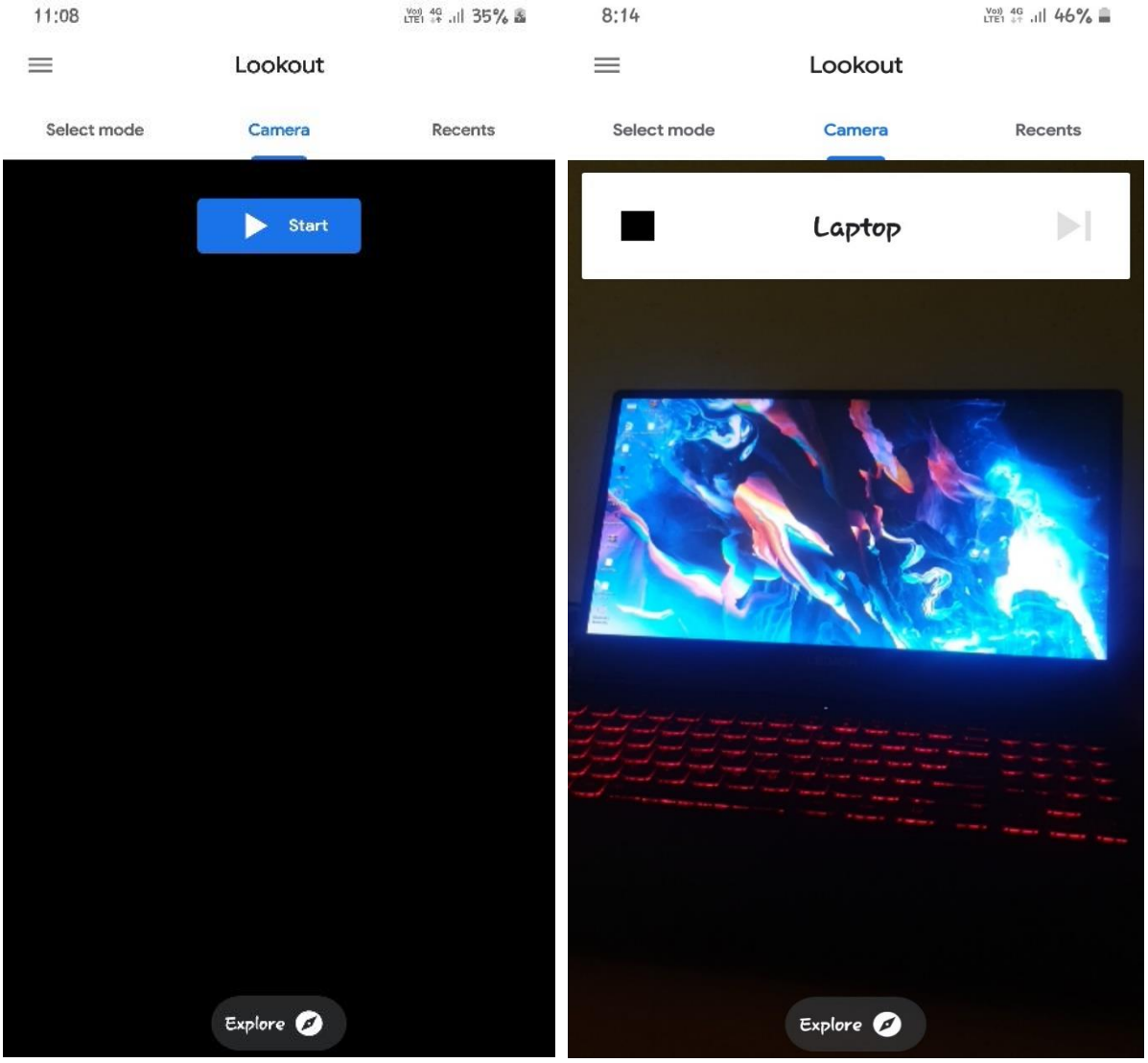


Fig 7.1 Application Outlook

CHAPTER 8

CONCLUSION

The research, implementation and optimization developed allowed the design of a free application allowing the reading of texts, provided that the light conditions are the ideal for image recording and the equipment is properly directed to the text required to listen to so that recognition and reading are as satisfactory as possible. With the growth of digital photography, we now have tons of visual pictures that we take all over the place. One of the things that have interested many developers is how to get our computers to understand the content of these pictures a little bit better. Visual impairment is one of the biggest drawbacks of today's era. With advancement in technology, this problem can be tackled. The proposed implementation successfully identifies text and objects. The identified information is converted to speech. This may improve the life of blind people.

REFERENCES

[1] Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Mach. Intell. 26,1475–1490. doi:10.1109/TPAMI.2004.108

[2] Asha G. Hagargund, SharshaVanria Thota, MitadruBera, Eram Fatima Shaik "Image to speech conversion for visually impaired" Volume 03 - Issue 06 June 2017

[3] Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). "Fast classification using sparsedecision dags," in Proceedings of the 29th International Conference on MachineLearning (ICML-12), ICML '12, eds J. Langford and J. Pineau (New York, NY:Omnipress), 951–958.

[4] A. SUBBIAH*, T. ARIVUKKARASU, M. S. SARAVANAN, V. balaji "camera based label reader for blind people"Int. J. Chem. Sci.: 14(S3), 2016, 840-844 ISSN 0972-768X

[5] Mrs.Shilpa Reddy K, Mounika S.K,Pooja K , Sahana N "Text to Speech for the Visually Impaired" International Research Journal of Computer Science (IRJCS) ISSN: 2393- 9842Issue 05, Volume 4 , May 2017

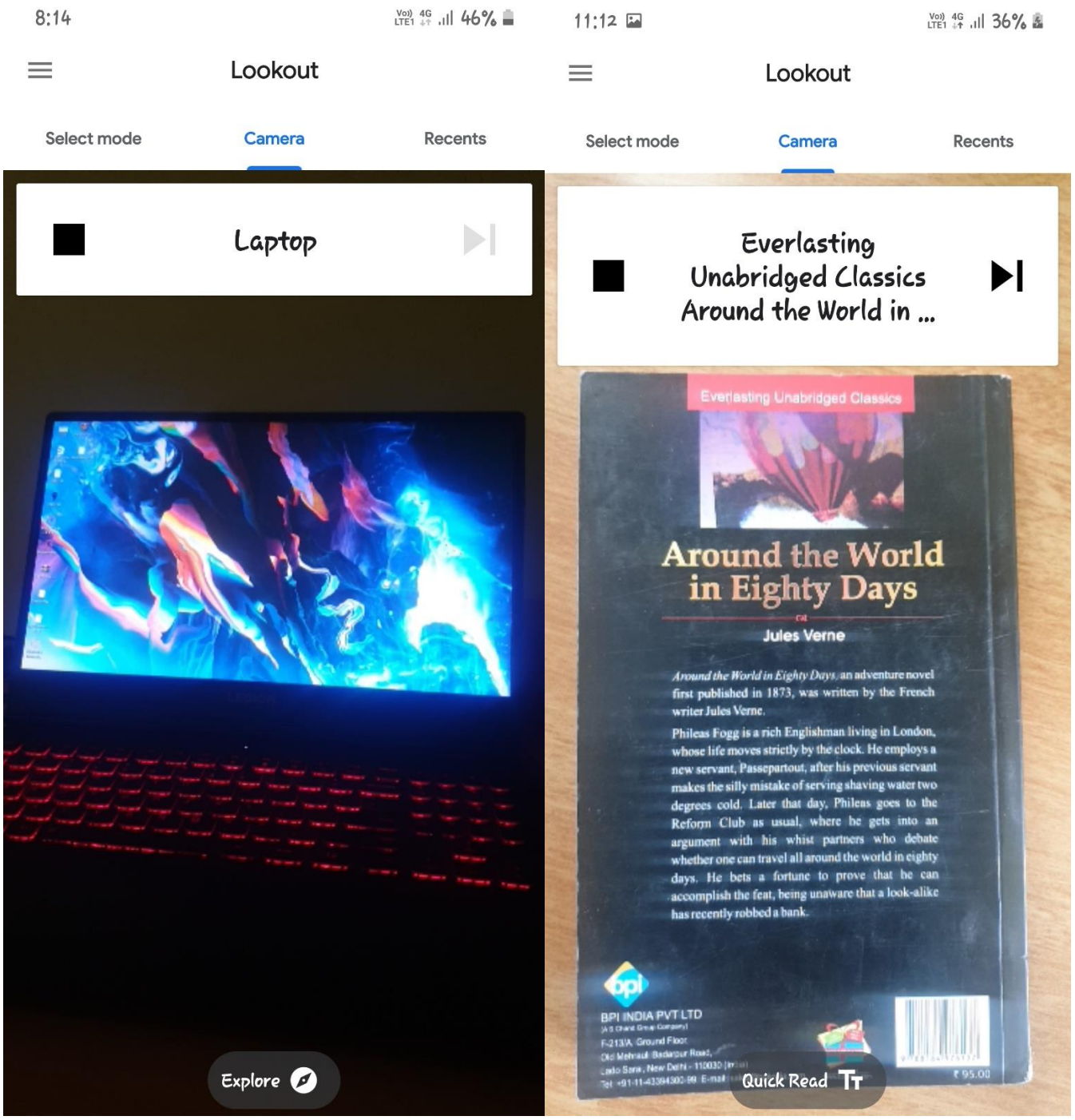
[6] https://www.researchgate.net/publication/275231020_Camera_Reading_for_Blind_People

[7] https://www.coursera.org/learn/machine-learning

[8] https://www.orcam.com/en/

APPENDIX-1

SNAPSHOTS



APPENDIX-2

SAMPLE CODE

```
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.graphics.RectF;
import android.graphics.Typeface;
import android.media.ImageReader.OnImageAvailableListener;
import android.os.SystemClock;
import android.util.Size;
import android.util.TypedValue;
import android.widget.Toast;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;
import org.tensorflow.lite.examples.detection.customview.OverlayView;
import org.tensorflow.lite.examples.detection.customview.OverlayView.DrawCallback;
import org.tensorflow.lite.examples.detection.env.BorderedText;
import org.tensorflow.lite.examples.detection.env.ImageUtils;
import org.tensorflow.lite.examples.detection.env.Logger;
import org.tensorflow.lite.examples.detection.tflite.Classifier;
import org.tensorflow.lite.examples.detection.tflite.TFLiteObjectDetectionAPIModel;
import org.tensorflow.lite.examples.detection.tracking.MultiBoxTracker;

/**
 * An activity that uses a TensorFlowMultiBoxDetector and ObjectTracker to detect and then track
 * objects.
 */
public class DetectorActivity extends CameraActivity implements OnImageAvailableListener {
  private static final Logger LOGGER = new Logger();

  // Configuration values for the prepackaged SSD model.
  private static final int TF_OD_API_INPUT_SIZE = 300;
  private static final boolean TF_OD_API_IS_QUANTIZED = true;
  private static final String TF_OD_API_MODEL_FILE = "detect.tflite";
  private static final String TF_OD_API_LABELS_FILE = "file:///android_asset/labelmap.txt";
```

```
  private static final DetectorMode MODE = DetectorMode.TF_OD_API;
  // Minimum detection confidence to track a detection.
  private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.5f;
  private static final boolean MAINTAIN_ASPECT = false;
  private static final Size DESIRED_PREVIEW_SIZE = new Size(640, 480);
  private static final boolean SAVE_PREVIEW_BITMAP = false;
  private static final float TEXT_SIZE_DIP = 10;
  OverlayView trackingOverlay;
  private Integer sensorOrientation;

  private Classifier detector;

  private long lastProcessingTimeMs;
  private Bitmap rgbFrameBitmap = null;
  private Bitmap croppedBitmap = null;
  private Bitmap cropCopyBitmap = null;

  private boolean computingDetection = false;

  private long timestamp = 0;

  private Matrix frameToCropTransform;
  private Matrix cropToFrameTransform;

  private MultiBoxTracker tracker;

  private BorderedText borderedText;

  @Override
  public void onPreviewSizeChosen(final Size size, final int rotation) {
    final float textSizePx =
        TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP,
getResources().getDisplayMetrics());
    borderedText = new BorderedText(textSizePx);
    borderedText.setTypeface(Typeface.MONOSPACE);

    tracker = new MultiBoxTracker(this);

    int cropSize = TF_OD_API_INPUT_SIZE;

    try {
      detector =
          TFLiteObjectDetectionAPIModel.create(
              getAssets(),
              TF_OD_API_MODEL_FILE,
              TF_OD_API_LABELS_FILE,
```

```
            TF_OD_API_INPUT_SIZE,
            TF_OD_API_IS_QUANTIZED);
    cropSize = TF_OD_API_INPUT_SIZE;
  } catch (final IOException e) {
    e.printStackTrace();
    LOGGER.e(e, "Exception initializing classifier!");
    Toast toast =
        Toast.makeText(
            getApplicationContext(), "Classifier could not be initialized", Toast.LENGTH_SHORT);
    toast.show();
    finish();
  }

  previewWidth = size.getWidth();
  previewHeight = size.getHeight();

  sensorOrientation = rotation - getScreenOrientation();
  LOGGER.i("Camera orientation relative to screen canvas: %d", sensorOrientation);

  LOGGER.i("Initializing at size %dx%d", previewWidth, previewHeight);
  rgbFrameBitmap = Bitmap.createBitmap(previewWidth, previewHeight, Config.ARGB_8888);
  croppedBitmap = Bitmap.createBitmap(cropSize, cropSize, Config.ARGB_8888);

  frameToCropTransform =
      ImageUtils.getTransformationMatrix(
          previewWidth, previewHeight,
          cropSize, cropSize,
          sensorOrientation, MAINTAIN_ASPECT);

  cropToFrameTransform = new Matrix();
  frameToCropTransform.invert(cropToFrameTransform);

  trackingOverlay = (OverlayView) findViewById(R.id.tracking_overlay);
  trackingOverlay.addCallback(
      new DrawCallback() {
        @Override
        public void drawCallback(final Canvas canvas) {
          tracker.draw(canvas);
          if (isDebug()) {
            tracker.drawDebug(canvas);
          }
        }
      });

  tracker.setFrameConfiguration(previewWidth, previewHeight, sensorOrientation);
}
```

```
@Override
protected void processImage() {
  ++timestamp;
  final long currTimestamp = timestamp;
  trackingOverlay.postInvalidate();

  // No mutex needed as this method is not reentrant.
  if (computingDetection) {
    readyForNextImage();
    return;
  }
  computingDetection = true;
  LOGGER.i("Preparing image " + currTimestamp + " for detection in bg thread.");

  rgbFrameBitmap.setPixels(getRgbBytes(), 0, previewWidth, 0, 0, previewWidth, previewHeight);

  readyForNextImage();

  final Canvas canvas = new Canvas(croppedBitmap);
  canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);
  // For examining the actual TF input.
  if (SAVE_PREVIEW_BITMAP) {
    ImageUtils.saveBitmap(croppedBitmap);
  }

  runInBackground(
      new Runnable() {
        @Override
        public void run() {
          LOGGER.i("Running detection on image " + currTimestamp);
          final long startTime = SystemClock.uptimeMillis();
          final List<Classifier.Recognition> results = detector.recognizeImage(croppedBitmap);
          lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

          cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
          final Canvas canvas = new Canvas(cropCopyBitmap);
          final Paint paint = new Paint();
          paint.setColor(Color.RED);
          paint.setStyle(Style.STROKE);
          paint.setStrokeWidth(2.0f);


float minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API;
          switch (MODE) {
            case TF_OD_API:
              minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API;
              break;
```

```
        }

        final List<Classifier.Recognition> mappedRecognitions =
            new LinkedList<Classifier.Recognition>();

        for (final Classifier.Recognition result : results) {
          final RectF location = result.getLocation();
          if (location != null && result.getConfidence() >= minimumConfidence) {
            canvas.drawRect(location, paint);

            cropToFrameTransform.mapRect(location);

            result.setLocation(location);
            mappedRecognitions.add(result);
          }
        }

        tracker.trackResults(mappedRecognitions, currTimestamp);
        trackingOverlay.postInvalidate();

        computingDetection = false;

        runOnUiThread(
            new Runnable() {
              @Override
              public void run() {
                showFrameInfo(previewWidth + "x" + previewHeight);
                showCropInfo(cropCopyBitmap.getWidth() + "x" + cropCopyBitmap.getHeight());
                showInference(lastProcessingTimeMs + "ms");
              }
            });
      }
    });
  }

  @Override
  protected int getLayoutId() {
    return R.layout.tfe_od_camera_connection_fragment_tracking;
  }

  @Override

protected Size getDesiredPreviewFrameSize() {
    return DESIRED_PREVIEW_SIZE;
  }

  // Which detection model to use: by default uses Tensorflow Object Detection API frozen
```

```
  // checkpoints.
  private enum DetectorMode {
    TF_OD_API;
  }

  @Override
  protected void setUseNNAPI(final boolean isChecked) {
    runInBackground(() -> detector.setUseNNAPI(isChecked));
  }

  @Override
  protected void setNumThreads(final int numThreads) {
    runInBackground(() -> detector.setNumThreads(numThreads));
  }
}
```

APPENDIX-3

ISSN:0971-1260
Vol-22- Issue-14-December-2019

Application for OCR & Navigation Assistance for Blind

Sowmya A.M[1], Sachin KumarS[2], Sandhya.S[3], Tejasri.K[4], Dhananjayan.S[5]

[1]Assistant Professor, Department of Computer Science & Engineering, Sri Sairam College of Engineering, Bengaluru.

[2345]UG Scholars, Department of Computer Science & Engineering, Sri Sairam College of Engineering, Bengaluru.

ABSTRACT---Visually impairment and illiterate, or have a learning disability is one of the biggest drawbacks for humanity, especially in this day and age when information and people is interconnected a lot by text messages (electronic and paper based) rather than talking. There is a need for convenient text reader that is reasonable and readily available to the Blind Community. The work in this research is images are converted into audio output. It is mainly used in the field of research in Character Recognition and Product Recognition, AI and Computer Vision. This device consists of three modules, image processing for text detection, object detection and convert detected things to voice. And then assisting the blind people to navigate by using the object detection API recognizing the objects and spelling it to the user. Button Camera/ Smart Phone Camera are designed to help Visually impaired in Navigation.

KEYWORDS--OCR, Tensor Flow, Tesser act, pyttsx.

1. INTRODUCTION

A common use of machine learning is to identify what an image represents. For example, we might want to know what type of animal appears in the following photograph. The task of predicting what an image represents is called image classification. An image classification model is trained to recognize various classes of images. For example, a model might be trained to recognize photos representing three different types of animals: rabbits, hamsters, and dogs. When we subsequently provide a new image as input to the model, it will output the probabilities of the image representing each of the types of animal it was trained on. Tensor flow is an open-source deep learning framework created by Google Brain. Tensor flow's Object Detection API is a powerful tool which enables everyone to create their own powerful Image Classifiers'

The Google Cloud Vision API enables developers to create vision-based machine learning applications based on object detection, OCR, etc. without having any actual background in machine learning. The Google Cloud Vision API takes incredibly complex machine learning models centred on image recognition and formats it in a simple REST API interface. It encompasses a broad selection of tools to extract contextual data on your images with a single API request. It uses a model which trained on a large dataset of images, similar to the models used to power Google Photos, so there is no need to develop and train your own custom model. Using the TensorFlow object Detection API and the Google Vision API can be deployed on a portable mobile device generally an android OS supported mobile phone. The visually challenged user can be guided through his navigation and OCR with speech output.

2.PROPOSED ALGORITHM

The below image (2.1) is a popular example of illustrating how an object detection algorithm works. Each object in the image, from a person to a kite, have been located and identified with a certain level of precision.

1.First, we take an image as input

2.Then we divide the image into various regions

3.We will then consider each region as a separate image.

4.Pass all these regions (images) to the CNN and classify them into various classes.

THINK INDIA JOURNAL

ISSN:0971-1260
Vol-22- Issue-14-December-2019

5.Once we have divided each region into its corresponding class, we can combine all these regions to get the original image with the detectedobjects.

Fig (2.1)

3.OCR and Object detection

a. Google Vision

Google Cloud's Vision API offers powerful pre-trained machine learning models through REST and RPC APIs. Assign labels to images and quickly classify them into millions of predefined categories. Detect objects and faces, read printed and handwritten text, and build valuable metadata into your image catalog.

b. TensorFlow Object Detection API

For assisting the user, the system has to first identify the objects through the camera and spell it as a speech output. When the user is walking the obstacle is called that object which comes near and in your way. The video frame will be containing so many objects but we will only look at some objects that are in our way. For that ROI is defined. This function will black the other part in the frame except the coordinates which we will give. Hence, extra objects will not be detected. For that the logic is to create two boundaries in ROI named left and right boundary. Hence it will split out the ROI into 3 parts. If the obstacle is in the right part, we will tell the user to move left and if it is into right then we will tell the user to move right.

c. Text recogniser

All of these systems take in data – often an image – from an unknown source, analyse the data in that input, and attempt to match them to existing entries in a database of known images. Object recognition does this inthree steps: detection, Text print creation, and verification or identification. When an image is captured, computer software analyses it to identify where the text are in, say, a crowd of words. In a photo, for example, cameras will feed into a software to identify text in the video feed.

d. Speech Output

To convert speech to on screen text or a computer command, a computer has to go through several complex steps in this implementat6ion we useText to Speech (TTS) library for Python 2 and 3.Works without internet connection or delay. Supports multiple TTS engines, including Sapi5, nsss, and espeak. pyttsx is a cross-platform text to speech library which is platform independent. The major advantage of using this library for text-to-speech conversion is that it works offline. TTS includes two different model implementations which are based on Tacotron and Tacotron2. Tacotron is smaller, efficient and easier to train but Tacotron2 provides better results, especially when it is combined with a Neural vocoder.

e. Android Application

An Android app is a software application running on the Android platform. Because the Android platform is built for mobile devices, a typical Android app is designed for a smartphone or a tablet PC running on the Android OS. Both the API are integrated into the application the user can use both OCR and navigation assistance system there is an option to switch between the modes.

4.Implementation of Algorithm OCR

OCR is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image. Through the scanning process a digital