CSE 499B

SENIOR DESIGN PROJECT

3rd Eye: Smart Eyeglass for Blind

GROUP: 2

Submitted By:

Md. Marjan---1621396042

Keya Barua---1731659042

Syed Ahmed---1521054642

Sumon Ray---1512162642

Faculty: Md. Mahfuzur Rahman

Declaration

This is to declare that no part of this report or the project has been previously submitted elsewhere for the fulfillment of any other degree or program. Proper acknowledgement has been provided for any material that has been taken from previously published sources in the bibliography section of this report.

…………………………..	…………………………..
Md. Marjan	Keya Barua
ECE Department	ECE Department
North South University	North South University
…………………………..	…………………………..
Syed Ahmed	Sumon Ray
ECE Department	ECE Department
North South University	North South University

Approval

The Senior Design Project entitled “3rd Eye: Smart Eyeglass for blind” by Md. Marjan (ID#1621396042), Keya Barua (ID#1731659042), Syed Ahmed (ID#1521054642) and Sumon Ray (ID#1512162642) has been accepted as satisfactory and approved for partial fulfillment of the requirement of BS in CSE degree program on October, 2020.

Supervisor’s Signature

…………………………………………………….

Md. Mahfuzur Rahman
Assistant Professor
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

Department Chair’s Signature

…………………………………………………

Dr. Rezaul Bari
Associate Professor
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

Acknowledgement

First of all, we would like to express our sincere gratitude to our honorable course instructor, Md. Mahfuzur Rahman, for his constant and meticulous supervision, valuable suggestions, his patience and encouragement to complete the project work.

We would also like to thank the ECE department of North South University for providing us with the opportunity to have an industrial level design experience as part of our curriculum for the undergraduate program.

Finally, we would like to thank our families and everybody who supported us and provided with guidance for the completion of this project.

Abstract

In this report we present a smart glass specifically built for the blind called the "3rd Eye". The smart eyeglass is capable of detecting objects in front by taking images of the front surrounding. Also, it is able to detect objects by processing images by taking screenshots from video. In our project, we have used Raspberry Pi and a Camera module to take pictures for object detection. We have also converted text to speech using ESpeak, so that the blind person can hear the names of the detected objects through an earphone attached to the glasses. We completed this project using Python and OpenCV.

Table of Content

Page

Chapter 1

Project Overview

1.1 Introduction

People born with disabilities are unfortunate as they cannot live the life of a normal person. For example, a blind person needs the support of someone else to do something or to go somewhere so that he is not hurt by any obstacle. The objective of this project is to build a Smart Glass for the blind so that the blind person will be informed of any obstacle beforehand and he can save himself from being hurt and also, he will not need the support of someone else. This project is important because Smart glasses available in the market are expensive which cannot be afforded by everyone. We tried to build a product which will be helpful for the blind and also affordable.

The 3rd Eye will have an attached camera to the glasses and a button. When the button is clicked, the camera will take a picture of the surrounding in front of the person. The glasses will then perform object detection on the picture and inform the person the names of the objects in the picture through a headphone attached to the glasses. We approached this project through several steps. First, we conducted research on the project, searched for previous related works. Then we collected the hardware components required. Then we started coding and building up the glasses.

This report talks about our planning, process and the end result of our project. It also discusses the challenges faced by us during the working of the project like budget challenges, time limitation etc.

1.2 Object Detection Basics

Object detection **[1]** is a computer vision technique that works to identify and locate objects within an image or video. Object detection draws bounding rectangles around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene. Most of the times, object detection is confused with Image Recognition.

Image recognition **[2]** assigns a label to an image. For example, in Image Recognition, a picture of a dog receives the label "dog". A picture of two dogs, still receives the label "dog". But object detection draws a rectangle around each dog and labels the rectangle "dog". Object detection predicts where each object is and what label should be applied. In this way, object detection provides more information about an image than recognition.

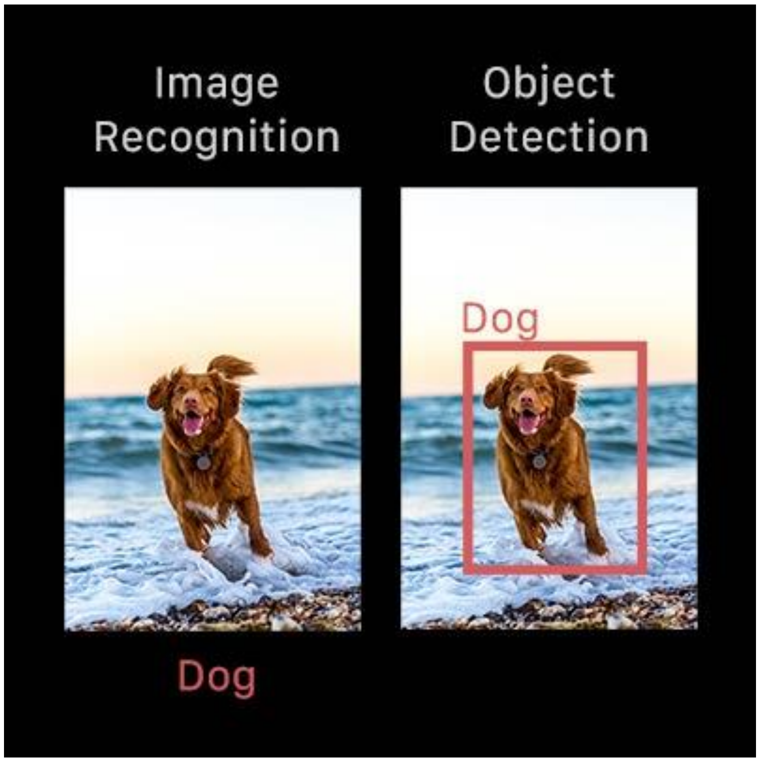


Figure 1: Difference between Image Recognition and Object Detection

Object Detection can be classified into two: **Machine Learning-based approach and Deep Learning-based approach.**
The Machine Learning-based approaches is a traditional computer vision technique that is used to look at various features of an image, like the color histogram or edges, to identify groups of pixels that may belong to an object. These features are then fed into the regression model that predicts the location of the object along with its label.
Whereas, the Deep Learning-based approaches use convolutional neural networks (CNNs) algorithms to perform end-to-end, unsupervised object detection, in which features do not need to be defined and extracted separately.

1.3 Project Goals

Technology has been developing rapidly and is being used in almost every fields like healthcare, farming, business and education. Object detection technique is used in many areas. Our objective was to build a system for the visually impaired which can detect objects in the surrounding helping the disabled to get to know his or her environment and move freely without being hurt or without the support of another person.

1.3.1 Project Description

The 3rd Eye: Smart glass for the blind is a technologically advanced eyeglass specifically for the blind, which can detect objects. Built using Python and OpenCV and Raspberry PI 3 Model B and a Camera module, it can detect objects from captured images or from screenshots of a video.

Capabilities of 3rd eye:

- Take image/video of the surrounding.
- Detect images from the images or screenshots from the video.
- Convert the texts of the detected objects into speech.
- The speech of the detected objects can be heard through the headphone.

1.3.2 Difficulty

The project is of medium difficulty. As none of our members are experienced with working on object detection or machine learning, we had to learn machine learning through tutorials from the internet. We then went through object detection tutorials. This way, we made use of our timeline of 8 months into completing our project successfully.

1.3.3 Motivation

Technology has always been made available for the abled people. The disabled suffer the stress from their disabilities due to lack of technology or due to expensive available techs. This motivated us into building an eyeglass for the visually impaired which is able to detect objects through captured images or screenshots from video and convert the texts into voice.

The 3rd Eye can help the blind navigate around places easily by letting them know the objects present around. This way, the blind can move without the support of another and without hurting oneself.

1.4 Summary

This chapter specified the overview of our project, some basic information of object detection and project goals. This project was motivated by the lack of technology available for the disabled to help them navigate around easily without the support of another person. Detailed information about object detection are given in Chapter 3.

Chapter 2

Related Works

2.1 Introduction

In this section, we describe the previous works which have motivated us into building our smart glass for the blind. There are many projects and research papers on object detection which describes the technique of object detection in detail. Though there are similar projects related to eyeglass for the blind, these systems are slow in processing. The 3rd Eye is done using Python and OpenCV which are reliable techniques for object detection and processes rapidly.

2.2 Related works

2.2.1 Sight: For the Blind

The SIGHT: for the blind [3] is a smart glass for the blind built on TensorFlow and Android Things. But there are some limitations to the SIGHT. The glasses will not work with old Raspberry Pi Versions, like Raspberry Pi Zero, as Google Android Things is not supported by the old Raspberry Pi microprocessors. But the disadvantage in this is that since Android thing does not support the newer versions of Raspberry Pi, this project will not work with newer versions of Raspberry Pi.

Figure 2: Sight: For the blind

2.2.2 Traffic Light Detection using TensorFlow Object Detection Framework

This paper [4] presents a deep learning approach which detects traffic light by comparing two object detection models and by evaluating the flexibility of the TensorFlow Object Detection Framework to solve the real-time problems which include Single Shot Multibox Detector (SSD) Mobilenet V2 and Faster-RCNN. The experimental study shows that Faster-RCNN delivers 97.015%, which outperforms SSD by 38.806% for a model which has been trained using 441 images.

2.3 Summary

This section described the previous works and papers related to object detection. Sight: For the Blind is a smart glass built for the visually impaired powered by Android things and TensorFlow. The paper "Traffic Light Detection using TensorFlow object detection framework" detects traffic light using deep learning approach by comparing two object detection models and by evaluating the flexibility of the TensorFlow Object Detection Framework.

Chapter 3

Theory

3.1 Introduction

In this section, the theory behind object detection are described in detail. Object Detection is a Computer Vision technique which aims to identify objects within an image or a video. This technique draws boxes around the objects and labels the boxes with the names of the objects.

3.2 Working of object detection

Deep learning-based object detection models have two parts: Encode and Decoder. An **encoder** takes an input image and runs it through a series of blocks and layers that learn to extract statistical features that is used to locate and label objects. Outputs from the encoder are then passed to a **decoder**, which predicts bounding boxes and labels for each object.

Pure regressor is a simple regressor. It is connected to the output of the encoder and predicts the location and size of each bounding box directly. The output of the model is the X, Y coordinate pair for the object and its extent in the image. Though this model is simple, it is limited. The number of boxes must be specified ahead of time. For example, if your image has two dogs, but the model was designed to detect only a single object, one will go unlabeled. However, if the number of objects is known that is needed to predict in each image ahead of time, pure regressor-based models is a good option.

An extension of the regressor approach is a **region proposal network**. In this decoder, the model proposes regions of an image where it believes an object might be located. The pixels belonging to these regions are then fed into a classification subnetwork to determine a label (or reject the proposal). It then runs the pixels containing those regions through a classification network.

Object detectors output the location and label for each object. To know the accuracy of the location of the object, the most commonly-used metric is **intersection-over-union (IOU)**. Given two bounding boxes, we compute the area of the intersection divided by the area of the union. This value ranges from 0 (no interaction) to 1 (perfectly overlapping).

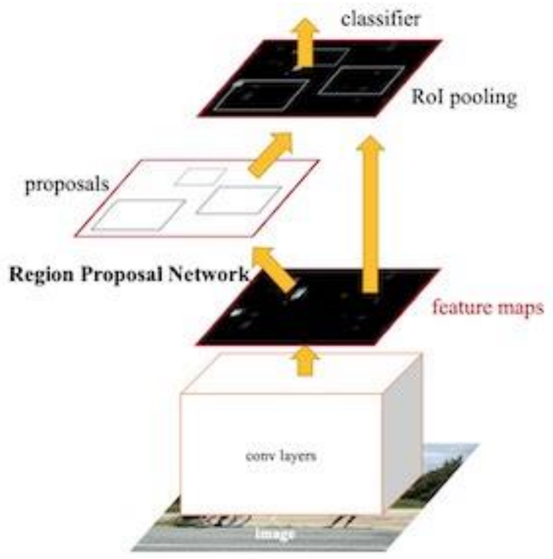


Figure 3: Region Proposal Network

3.3 Feature Extraction

Feature extraction reduces a variable sized image to a fixed set of visual features. Image classification models are typically constructed using strong visual feature extraction methods. Whether they are based on traditional computer vision approaches, like filter based approached methods or deep learning methods, they all have the exact same objective of extracting features from the input image and these features determine the class of the image. In object detection frameworks, people typically use pretrained image classification models to extract visual features.

3.4 Application of Object Detection

Object detection is used in many areas ranging from with use cases ranging from personal security to productivity in the workplace. Object detection and recognition is applied in many areas of computer vision, including image retrieval, security, surveillance, automated vehicle systems and machine inspection. Some of the present applications [5] of object detection are given below:

3.4.1 Optical Character Recognition

Optical character recognition is the mechanical or electronic 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 like the text on signs and billboards in a landscape photo or from subtitle text superimposed on an image.

3.4.2 Self-driving cars

Self-driving cars make extensive use of object detection to decide to whether accelerate, apply brakes or turn. It needs to know where all the objects are around the car and what those objects are which requires object detection and the car is trained to detect known set of objects such as cars, pedestrians, traffic lights, road signs, bicycles, motorcycles, etc.

3.4.3 Ball tracking in sports

In sports like cricket, football etc., audience requires more analyzing and more multidimensional information like location of the ball. Tracking of ball movement is of utmost importance for extracting any information from the video of this sports which is done with object detection.

3.4.4 Robotics

Automated robots have the capability to detect objects to avoid obstacles while moving. They are provided with the ability to process real-time visual data so that they can react quickly to adapt to changes in the environment. Reliable object detection and recognition is usually a necessary early step to achieve this goal.

3.5 Limitations of Object Detection

Object detection is being widely used in many fields. It has changed the technological world widely but there are some limitations to object detection. There are many objects in the world but the pretrained dataset that we have used in our project or any dataset used in any project has limited number of objects that can be identified. So, any object detection system cannot detect all the objects.

3.6 Summary

Object detection has changed our life by bringing in more technology like self-driving cars or giving us more detailed information about our favorite sports. This section describes the object detection technique, the working of object detection and how the features are extracted from the pretrained images to train the model.

Chapter 4

Structure of the system

4.1 Introduction

The 3rd Eye, a smart eyeglass for the blind is able to detect objects in the surrounding. The structure of our system is divided into two parts:

1. Object Detection
2. Conversion of text-to-speech

4.2 Structure of our System

The structure of the 3rd Eye is given below:

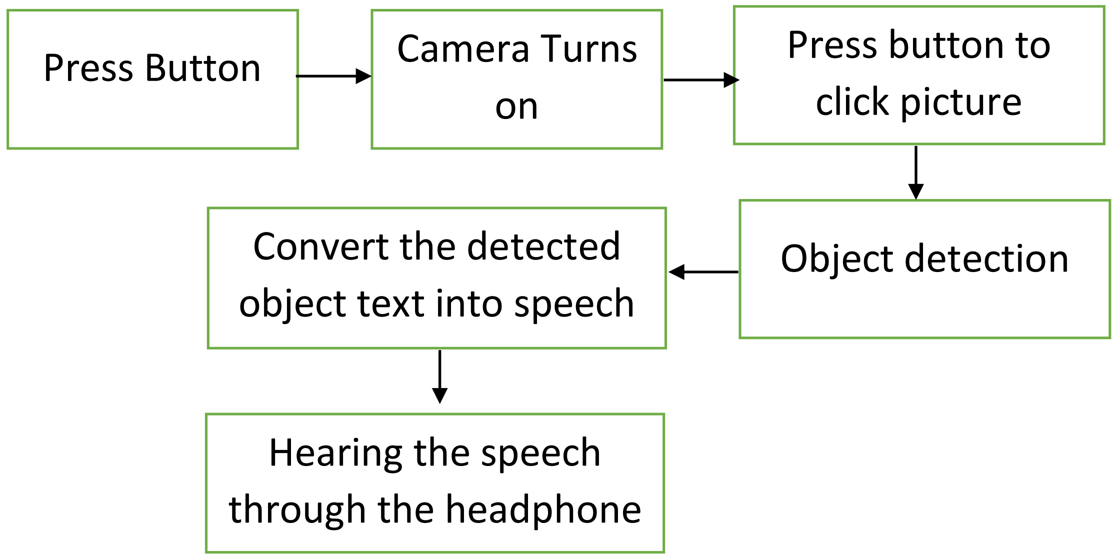

Press Button
Camera Turns on
Press button to click picture
Object detection
Convert the detected object text into speech
Hearing the speech through the headphone

Figure 4: Structure of image capture system by 3rd Eye

First, the wearer of the eyeglass presses the push button which turns on the camera. Pressing the push button once again triggers the camera to take a picture of the surrounding. The taken image is then processed for object detection. After the objects are detected, the texts are converted into speech by a speech synthesizer which can be heard through the headphone.

The structure of detecting objects from video is given below: -

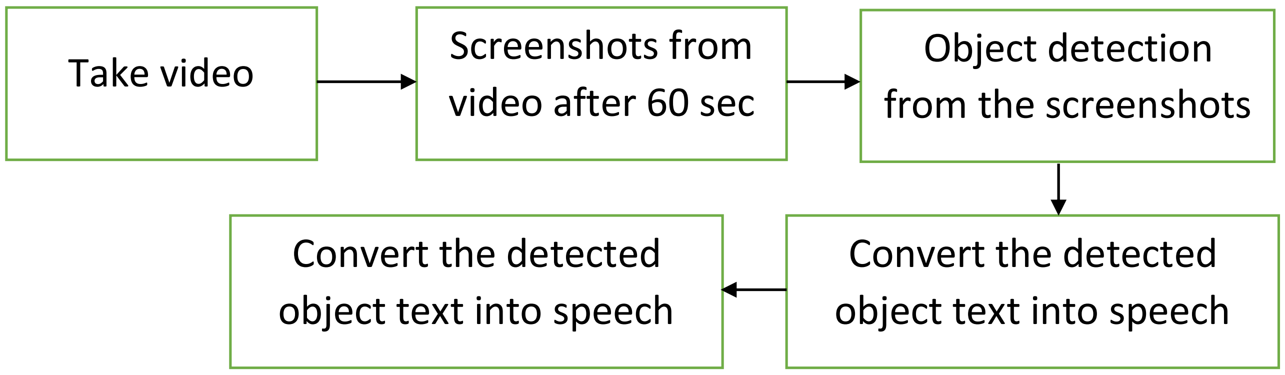

Take video
Screenshots from video after 60 sec
Object detection from the screenshots
Convert the detected object text into speech
Convert the detected object text into speech

Figure 5: Structure of object detection from video

The smart glass can record a video and take a screenshot after every 60 seconds and detect objects from the screenshot images. The detected objects' texts are then converted into speech which can be heard through the headphone.

4.3 Workflow

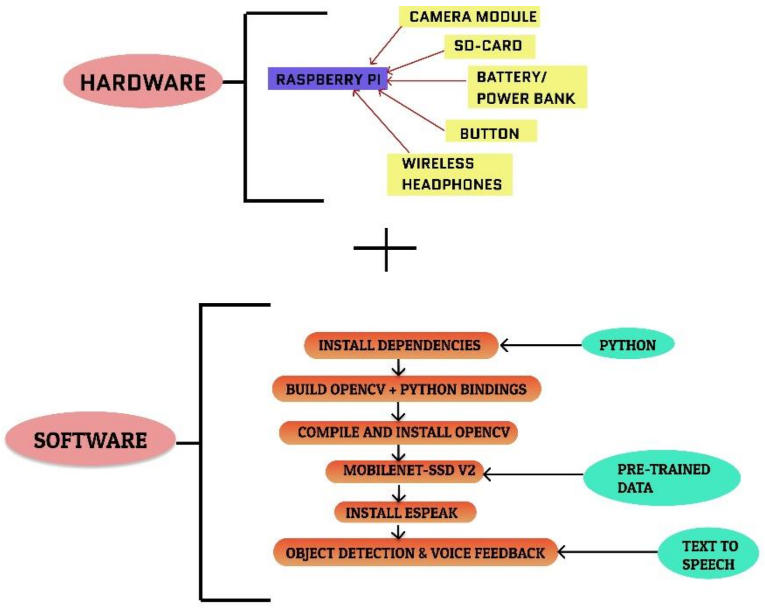


Figure 6: Workflow of 3rd Eye

4.4 Summary

In this section, we have described the structure of our smart glass for the blind. 3rd Eye can capture image or record video and take screenshots and detect objects from those images. The texts from detected objects are then converted into speech by speech synthesizer.

Chapter 5

Components used in the system

5.1 Introduction

In this section, the components used to build the system are described in detail. We have used Raspberry Pi, which is a mini computer with a single board. We have also used camera module to capture images of the surrounding, a push button and a wireless headphone. Raspberry Pi is being used widely now in several projects and is very common in creating mini projects, though it is not very widely used commercially.

5.2 Hardware Components

5.2.1 Raspberry Pi

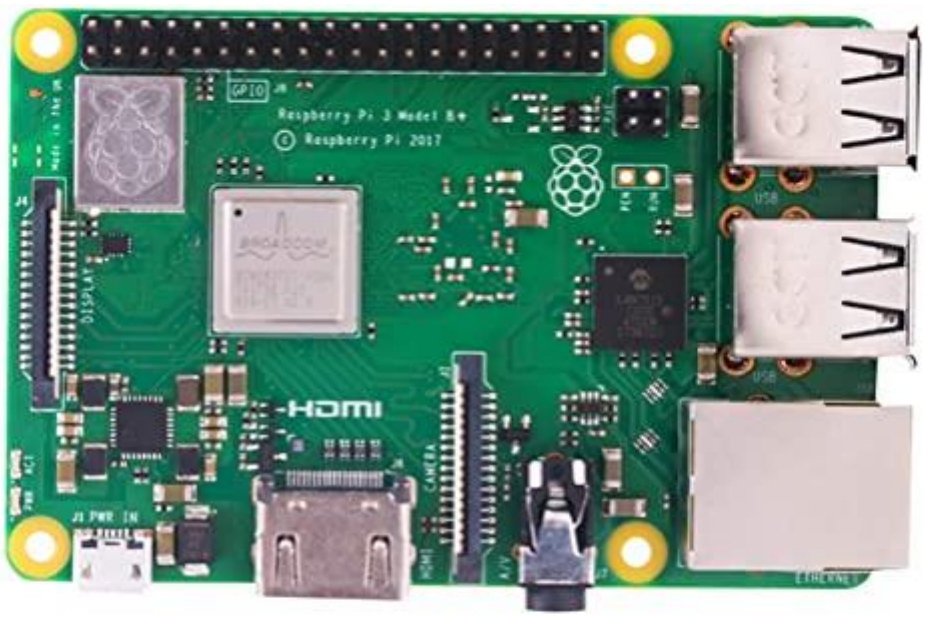


Figure 7: Raspberry Pi

Raspberry Pi 3**[6],[7]** is the processor of our system. It is an ARM based credit card sized SBC (Single Board Computer) created by Raspberry

Pi Foundation developed in the United Kingdom. Raspberry Pi runs Debian based GNU/Linux operating system Raspbian and ports of many other OSes exist for this SBC.

There are various benefits of Raspberry Pi:

1. Low cost
2. Huge processing power in a compact board
3. Several modules can be connected such as HDMI, multiple USB, Ethernet, onboard Wi-Fi and Bluetooth etc.
4. Support Linux, Python

There are several versions of Raspberry Pi:

1. The first generation **"Raspberry Pi Model B"** was released in February 2012, followed by the simpler and cheaper Model A.
2. In 2014, the Foundation released a board with an improved design, **"Raspberry Pi Model B+"**. These boards are approximately credit-card sized and represent the standard mainline form-factor.
3. In February 2015, the **"Raspberry Pi 2"** was released featuring a 900 MHz quad-core ARM Cortex-A7 processor and 1 GiB RAM.
4. A smaller size **"Raspberry Pi Zero"** with reduced input/output (I/O) and general-purpose input/output (GPIO) capabilities was released in November 2015 for US$5.
5. In February 2016, **"Raspberry Pi 3 Model B"** was released with a 1.2 GHz 64-bit quad core processor, on-board 802.11n Wi-Fi, Bluetooth and USB boot capabilities.
6. **"Raspberry Pi 4 Model B"** was released in June 2019 with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor, on-board 802.11ac Wi-

Fi, Bluetooth 5, full gigabit Ethernet, two USB 2.0 ports, two USB 3.0 ports, and dual-monitor support via a pair of micro HDMI ports for up to 4K resolution.

5.2.2 Camera Module

The Raspberry Pi Camera Module **[8]** are official products from the Raspberry Pi foundation. The Pi camera module is a portable light weight camera that supports Raspberry Pi. It communicates with Pi using the MIPI camera serial interface protocol. All current models of Raspberry Pi have a port for connecting the Camera Module. The original 5-megapixel model was released in 2013 and an 8-megapixel Camera Module v2 was released in 2016. For both the models, there are visible light and infrared versions. A 12-megapixel high quality camera was released in 2020.

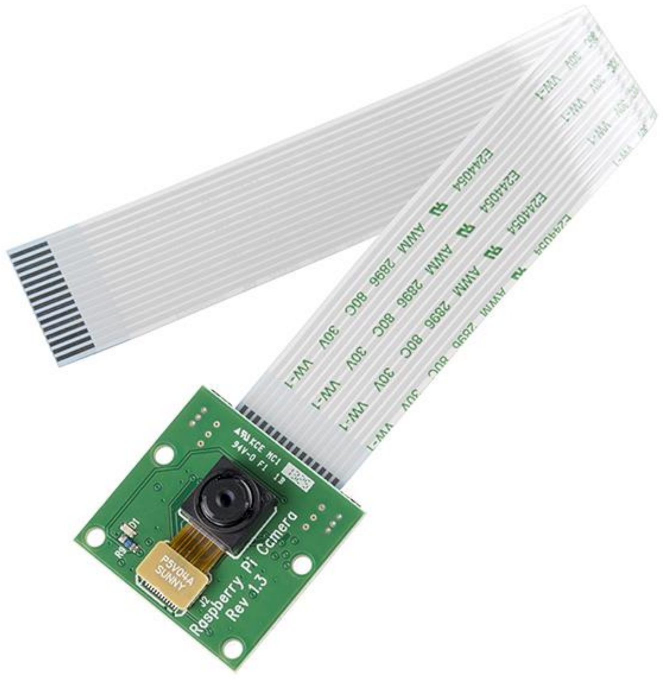


Figure 8: Raspberry Pi Camera Module

5.2.3 Push Buttons

The push button acts as a trigger for the camera to click a picture. When the push button is pressed first, it starts the camera module and pressed once more, the camera takes a picture.

5.2.4 SD card

The SD card is a key part of the Raspberry Pi; it provides the initial storage for the Operating System and files. Storage can be extended through many types of USB connected peripherals.

5.2.5 Power Bank/Battery

To power up the Raspberry pi, we use a power bank and connect it via a USB port. We can also use batteries.

5.2.6 Wireless Headphone

A wireless headphone is attached through which the person will be able to hear the names of the detected objects.

Figure 9: Push Button

5.3 Implementation Methods

5.3.1 Python 3

Python **[9][10]** is an interpreted, high-level, general-purpose programming language. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and largescale projects.

Python was created by Guido van Rossum and first released in 1991. Python's design emphasizes on code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented, and functional programming. Python 3.0 was released in 2008 which was a major revision of the language that is not completely backward-compatible and more features are supported in Python 3 than in Python 2.

5.3.2 OpenCV

We used the OpenCV **[11],[12]** (Open source computer vision) with Python to build our project.
OpenCV is a library of programming functions mainly aimed at real-time computer vision. The library is open source and cross-platform under the BSD license. OpenCV is a highly optimized library with real time applications. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, Py-Torch.

5.3.3 Dataset

The dataset we have used in our project is **Mobilenet SSD V2**.
The Mobilenet SSD V2 **[13]** model is a Single-Shot multibox Detection (SSD) network intended to perform object detection. The model has been trained from the Common Objects in Context (COCO) image dataset. The model input is a blob that consists of a single image of 1x3x300x300 in RGB order and the model output is a typical vector containing the tracked object data.

5.3.4 ESpeak

To convert the texts to voice, we have used a speech synthesizer called ESpeak. Python supports ESpeak **[14]** to convert the text from objects detected and convert it into speech so that the blind can hear through the earphone.

5.4 Summary

To implement our system, we have used Raspberry Pi 3, a camera module, SD card, power bank and a push button as hardware components. For coding, we have used Python 3, OpenCV which is a library in Python, Mobilenet SSD V2 for training dataset and ESpeak as speech synthesizer to convert texts to speech.

Chapter 6

Results and Discussions

6.1 Introduction

This section discusses the solutions and results we have obtained from our project. The 3rd eye is able to perform object detection from images accurately and specify the names of the objects through the earphone. It is also able to detect objects from a video by taking screenshot images after every 30 seconds.

6.2 The Completed System

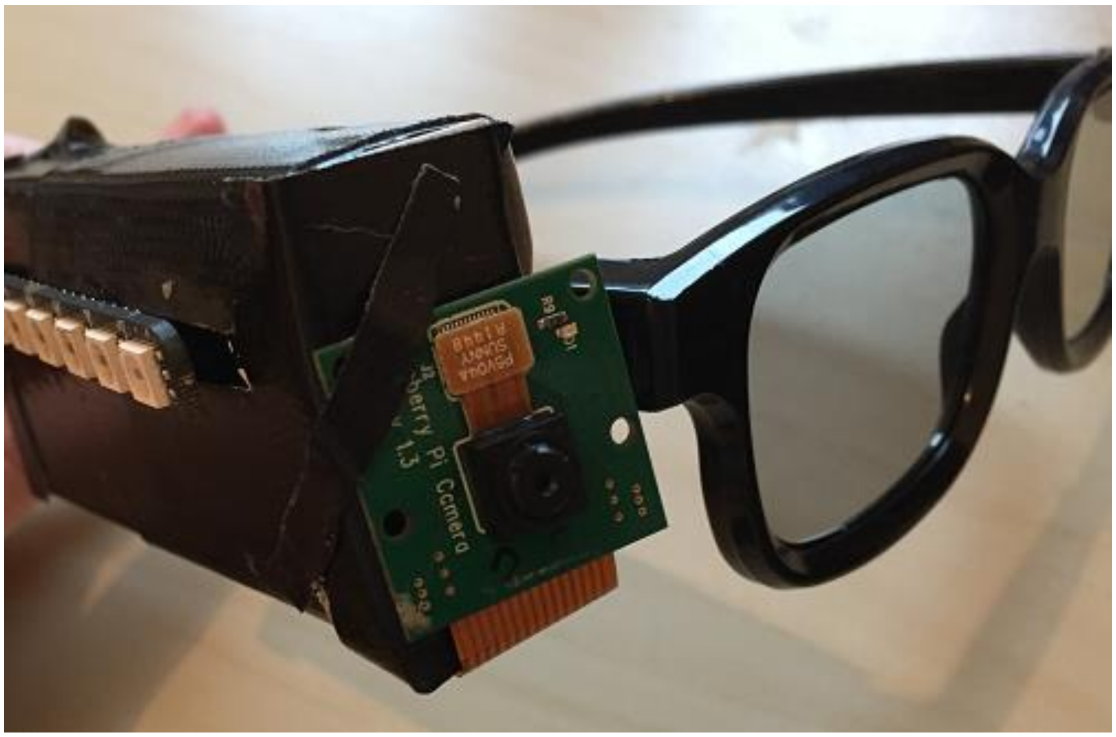
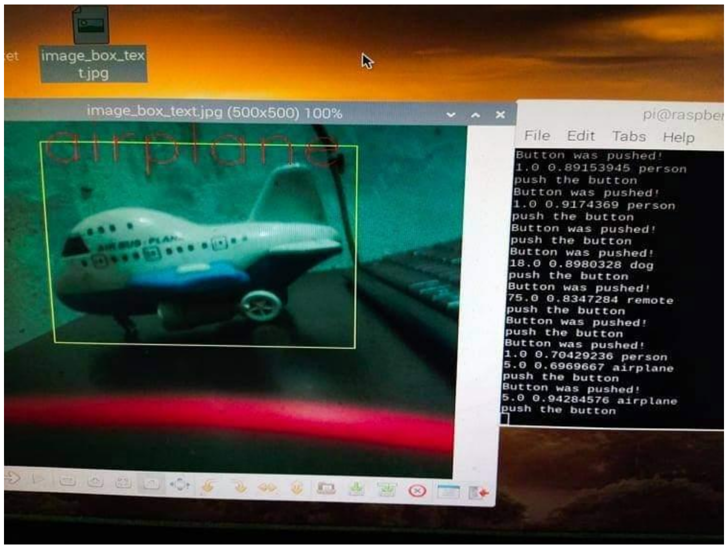


Figure 10: 3rd Eye-Smart glass for the blind

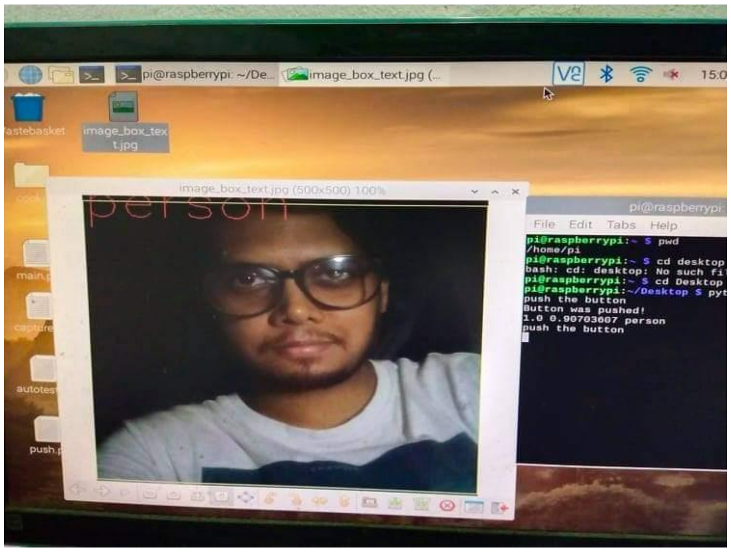
6.3 Object Detection Results

The eyeglass can detect several objects like bus, airplane, person, motorcycle, traffic light etc. The eyeglass can also take a video and capture screenshots from the video after every 30 seconds and detect objects.

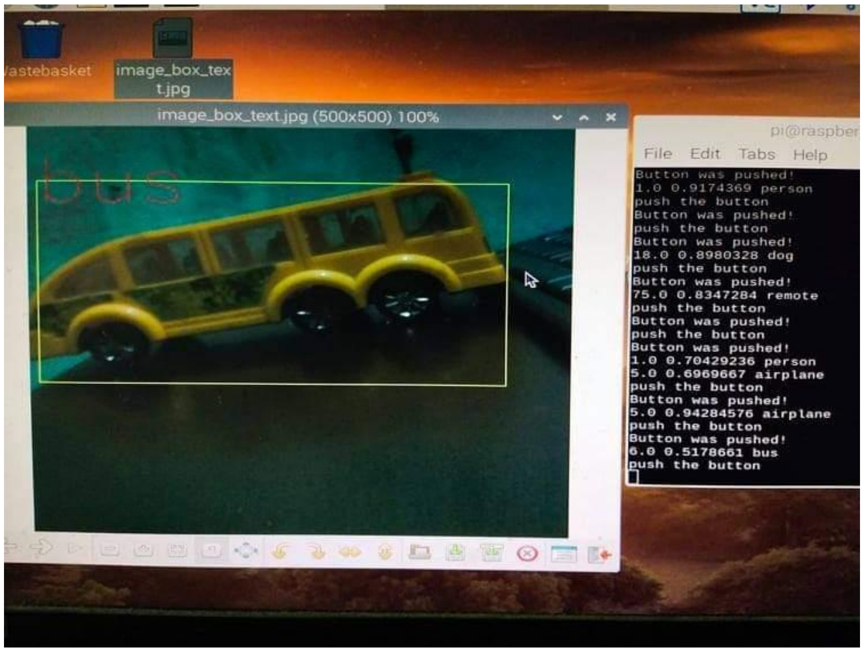
6.3.1 Object Detection from captured images



```
Button was pushed!
1.0 0.89153945 person
push the button
Button was pushed!
1.0 0.9174369 person
push the button
Button was pushed!
push the button
Button was pushed!
18.0 0.8980328 dog
push the button
Button was pushed!
75.0 0.8347284 remote
push the button
Button was pushed!
push the button
Button was pushed!
1.0 0.70429236 person
5.0 0.6969667 airplane
push the button
Button was pushed!
5.0 0.94284576 airplane
push the button
```



```
pi@raspberrypi:~ $ pwd
/home/pi
pi@raspberrypi:~ $ cd desktop
bash: cd: desktop: No such fi
pi@raspberrypi:~ $ cd Desktop
pi@raspberrypi:~/Desktop $ pyt
push the button
Button was pushed!
1.0 0.90703607 person
push the button
```



```
Button was pushed!
1.0 0.9174369 person
push the button
Button was pushed!
push the button
Button was pushed!
18.0 0.8980328 dog
push the button
Button was pushed!
75.0 0.8347284 remote
push the button
Button was pushed!
push the button
Button was pushed!
1.0 0.70429236 person
5.0 0.6969667 airplane
push the button
Button was pushed!
5.0 0.94284576 airplane
push the button
Button was pushed!
6.0 0.5178661 bus
push the button
```

Figure 11: Detection of objects from captured images

6.3.2 Object Detection from video

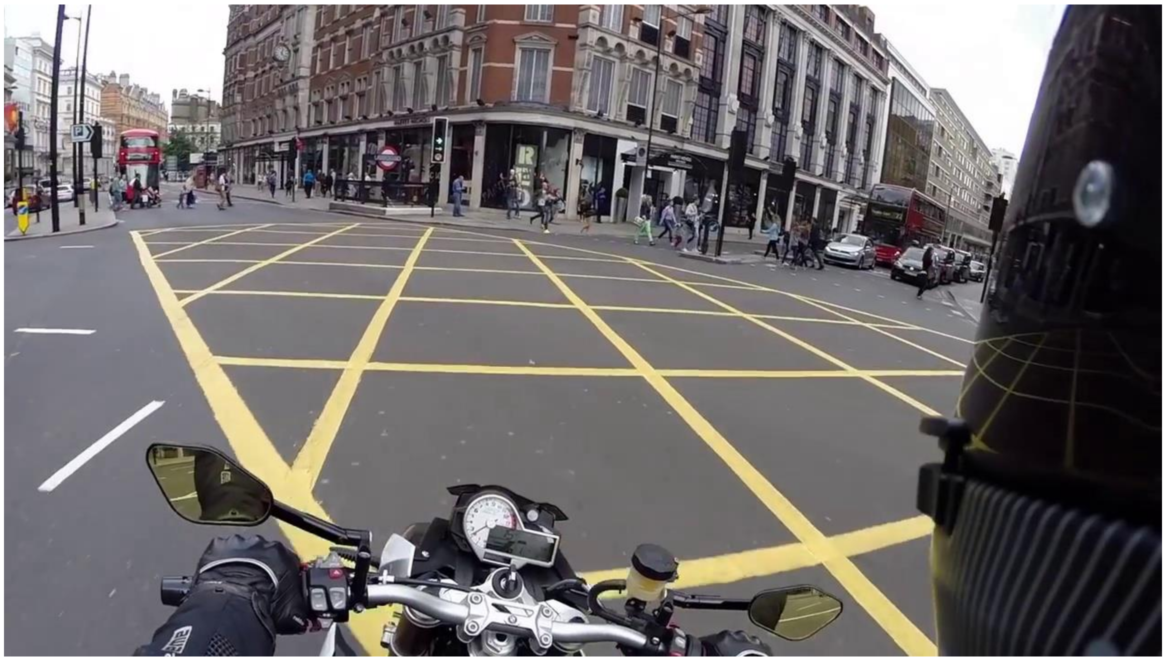


Figure 12: Image before detection

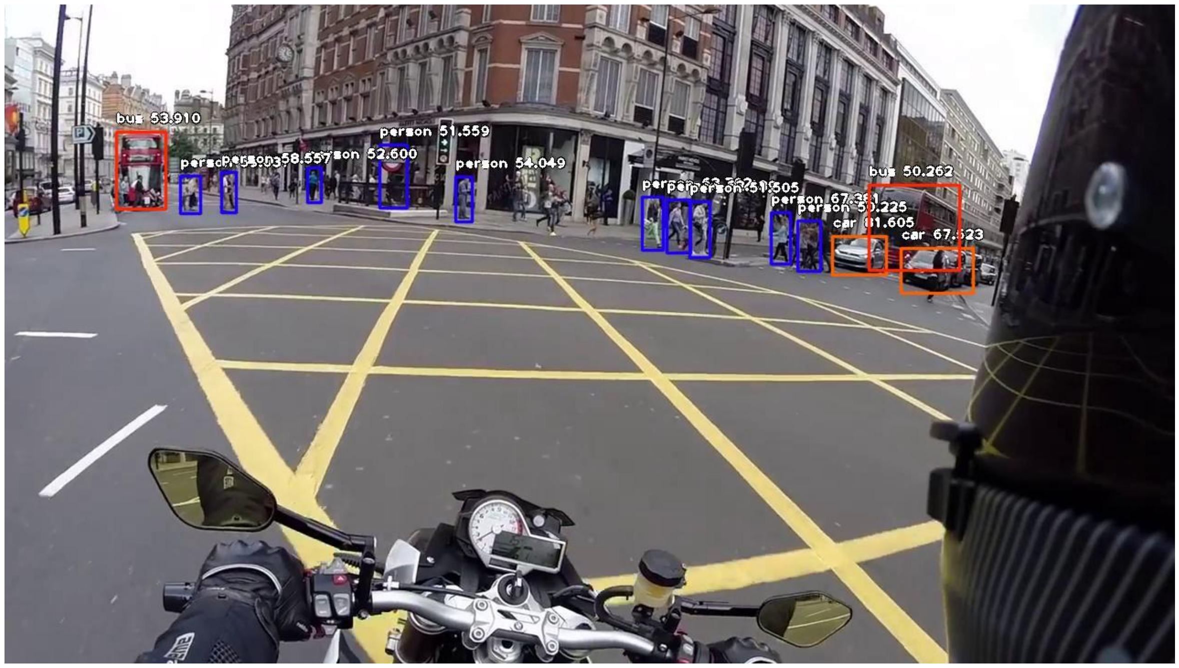


Figure 13: Image after detection

6.4 Summary

The 3rd Eye: Smart glass for the blind can detect several objects. The images before and after detection are given above. The glass can also take screenshot images from video and perform object detection on those images.

Chapter 7

Limitations and Challenges

7.1 Introduction

We had to face several challenges while working on this project. For example, we had to face challenges like budget, time etc. Since we had limited time and budget, we had to opt for cheap products and complete the project within the given time.

7.2 Limitations

7.2.1 Budget

Price List		
Item	**Quantity**	**Price**
Raspberry Pi Model B	1	Tk 4,500
Transcend 32 GB MicroSD card	1	Tk 400
Raspberry Pi Camera Module V2 – 8	1	Tk 1,200
HDMI cable	1	Tk 200
Push button, wires	3, 1 set	Tk 100
OTG Cable	1	Tk 200

Table 1: Budget

7.2.2 Timeline

Table 14: Task Distribution

Project Timeline			
SL	Date	Task(s)	Responsible Person(s)
1	Week 1	Push Button Coding	Marjan
2	Week 2	Object Detection Coding	Marjan
3	Week 3	Object Detection Coding	Marjan
4	Week 4	Multiple object detection	Marjan
5	Week 5	Image Detection from the video	Keya,Sumon,Syed
6	Week 6	Tried to implement detection from video in Raspberry Pi	Marjan, Keya
7	Week 7	Solved object detection error	Marjan,Keya,Sumon,Syed
8	Week 8	Image to Speech Coding	Marjan
	Week 8	Final report Writing	Keya,Sumon,Syed

7.2.3 Overheads

We had to face many challenges during the working of the project. Our Raspberry Pi 3 was damaged, so we had to buy another one. Also, due to the Covid-19 Pandemic, group members could not gather together and work on the project. But we tried to communicate virtually and tried finishing the project on time.

7.3 Summary

In this section, the limitations and challenges we had to face and the implementation overheads are described.

Chapter 8

Compliance with standards

8.1 Introduction

There are several international standards a project should meet. Some of the important ones are IEEE **[15]** standard. The compliance of 3rd Eye with these standards are described in this section.

8.2 Compliance with IEEE standard

There are a few distinct guidelines put forward by IEEE Standards affiliation. Our project meets the IEEE standards for Raspberry Pi. Our system is safe, beneficial for the disabled, and uses equipment that do not pose any threat to the environment.

8.3 Summary

In this section, the compliance of our project with the IEEE standard are described.

Chapter 9

Design Impact

9.1 Introduction

The impacts that our project have on the environment, health and economy are described in this section. Also, the manufacturability and sustainability are described in detail.

9.2 Impact on Economy

There are many smart glasses available in the market like the Google Glasses. But, due to its very high price, many are not capable of affording it. So, this eyeglass is economically profitable for the industry as this will be available at a cheap price and many people will be able to but it. Nowadays, people are interested in smart devices. There are many devices available in the market like smart phones, smart watches etc. So, the 3rd Eye will be industrially profitable as this can be used not only by the blind but also by normal people.

9.3 Impact on Environment

Though our project does not aid in the development of the environment, it in no way harms it. Our project is free from all sorts of materials and chemicals that may harm the wild life or the environment. So, it is safe to say that our project does not have any negative environmental impact.

9.4 Ethical Impact

Our project can be put to a wide range of uses. It can be used by the visually impaired to aid them in moving around freely without the support of another person or without hurting themselves. It is intended to make the life of the disabled easier.

Thus, we can say that it is expected to have a positive ethical impact.

9.5 Political impact

Our project does not have any direct impact on the political aspect.

9.6 Health and safety impact

Our aim for the project was to positively impact the health and safety of its user. The 3rd Eye is meant to help visually impaired live a better life with the confidence that normal people live with. Another point worth mentioning is that our project does not use any harmful chemicals or substances, or emit any harmful radiation that could negatively affect the manufacturers, users or the environment. Thus, we can say that it has an overall positive health and
safety impact.

9.7 Manufacturability

The 3rd Eye requires components which are widely available at a low price. The mechanical design is simple and has been precisely documented. Thus, it would not be a complex system to manufacture.

9.8 Sustainability

Our smart glass for the blind has been made rigid and all parts are fixed strongly to each other so there is very little chance of the components breaking down, loosening, or falling apart. The design is expected to be durable and resistant to

pressure or stress of any kind. The Raspberry PI 3 is a good model which has a very high processing power and has very low chance of damage.
Thus, it is expected to sustain for a long period of time.

9.9 Summary

The different aspects of this project’s impact and its manufacturability and sustainability have been discussed in this chapter.

Chapter 10

Conclusion

In this project, we focused on developing a smart glass called the 3rd Eye which is capable of detecting objects from image or from a video. We completed this project using Raspberry Pi 3 Model B, a camera module and implemented it using Python and OpenCV. The 3rd Eye can detect several objects such as airplane, bus, person, traffic lights etc.

The eyeglass can detect objects by capturing images through the camera or by capturing screenshots after every 60 seconds from video. The eye-glass is a user-friendly glass for the disabled, especially the blind. When a person pushes the

button, the camera attached to the glasses takes image and detects objects from the image and the person wearing it will be able to hear the name of the object through the wireless headphone. We worked hard and tried making the project
a successful one. In the future, we will try integrating facial recognition
feature to the glasses so that the glasses will be able to detect faces of
persons and tell the name of the person, if known.

We hope that our smart eyeglass will help the disabled, especially the blind to live an easier life and navigate around easily without harming oneself. We also hope to that our smart eyeglass will have a great impact on people’s lives.

Bibliography

[1] [Online] https://www.fritz.ai/object-detection/#:~:text=Object%20detection%20is%20a%20computer%20vision%20technique%20that%20works%20to,move%20through)%20a%20given%20scene.

[2] [Online]https://searchenterpriseai.techtarget.com/definition/image-recognition

[3] hackster.io. Sight: for the blind.

[4] T. V. Janahiraman and M. S. M. Subuhan, "Traffic Light Detection Using Tensorflow Object Detection Framework," *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, Shah Alam, Malaysia, 2019, pp. 108-113, doi: 10.1109/ICSEngT.2019.8906486.

[5] Abdul Vahab, Maruti S Naik, Prasanna G Raikar and Prasad S R, "Applications of Object Detection System", vol. 06, no. 4[Online]. Available: https://www.irjet.net/archives/V6/i4/IRJET-V6I4920.pdf

[6] [online] https://www.raspberrypi.org/

[7] [online] https://opensource.com/resources/raspberry-pi

[8] [online] https://www.raspberrypi.org/documentation/hardware/camera/

[9] [online] https://www.python.org/

[10] David Ascher, “Learning Python”

[11] [online] https://opencv.org/

[12] Alberto Fernández Villán, “Mastering OpenCV 4 with Python”, 2019

[13] [online]

https://docs.openvinotoolkit.org/latest/omz_models_public_mobilenet_ssd_mobile

net_ssd.html

[14] espeak, https://pypi.org/project/py-espeak-ng/

[15] ieee, http://standards.ieee.org/

APPENDIX

Capture

```
from picamera import PiCamera

from time import sleep


camera=PiCamera()

camera.resolution = (500,500)


camera.start_preview()

sleep(5)

camera.capture('image.jpg')

camera.stop_preview()
```

Main

```
import cv2

from picamera import PiCamera

from time import sleep

from espeak import espeak

import os

import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library




def looping():

    espeak.synth("push the button ")

    print('push the button')

    testnew=0

    def button_callback(channel):


        print("Button was pushed!")
```

```
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(8, 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)


while True: # Run forever
  if GPIO.input(8) == GPIO.HIGH:
    print("Button was pushed!")
    break

camera=PiCamera()
camera.rotation=90
camera.resolution = (500,500)

camera.start_preview()
sleep(5)
camera.capture('image.jpg')
camera.stop_preview()
camera.close()
```

```
espeak.synth("please wait for the result")




# Pretrained classes in the model

classNames = {0: 'background',

        1: 'person', 2: 'bicycle', 3: 'car', 4: 'motorcycle', 5: 'airplane', 6: 'bus',

        7: 'train', 8: 'truck', 9: 'boat', 10: 'traffic light', 11: 'fire hydrant',

        13: 'stop sign', 14: 'parking meter', 15: 'bench', 16: 'bird', 17: 'cat',

        18: 'dog', 19: 'horse', 20: 'sheep', 21: 'cow', 22: 'elephant', 23: 'bear',

        24: 'zebra', 25: 'giraffe', 27: 'backpack', 28: 'umbrella', 31: 'handbag',

        32: 'tie', 33: 'suitcase', 34: 'frisbee', 35: 'skis', 36: 'snowboard',

        37: 'sports ball', 38: 'kite', 39: 'baseball bat', 40: 'baseball glove',

        41: 'skateboard', 42: 'surfboard', 43: 'tennis racket', 44: 'bottle',

        46: 'wine glass', 47: 'cup', 48: 'fork', 49: 'knife', 50: 'spoon',

        51: 'bowl', 52: 'banana', 53: 'apple', 54: 'sandwich', 55: 'orange',

        56: 'broccoli', 57: 'carrot', 58: 'hot dog', 59: 'pizza', 60: 'donut',

        61: 'cake', 62: 'chair', 63: 'couch', 64: 'potted plant', 65: 'bed',

        67: 'dining table', 70: 'toilet', 72: 'tv', 73: 'laptop', 74: 'mouse',

        75: 'remote', 76: 'keyboard', 77: 'cell phone', 78: 'microwave', 79: 'oven',
          80: 'toaster', 81: 'sink', 82: 'refrigerator', 84: 'book', 85: 'clock',

          86: 'vase', 87: 'scissors', 88: 'teddy bear', 89: 'hair drier', 90: 'toothbrush'}




def id_class_name(class_id, classes):

  for key, value in classes.items():

    if class_id == key:

        return value


#Loading model

model = cv2.dnn.readNetFromTensorflow('models/frozen_inference_graph.pb',

                                      'models/ssd_mobilenet_v2_coco_2018_03_29.pbtxt')



image = cv2.imread("image.jpg")


image_height, image_width, _ = image.shape
```

```
    model.setInput(cv2.dnn.blobFromImage(image, size=(300, 300),
swapRB=True))
    output = model.forward()
    # print(output[0,0,:,:].shape)



    for detection in output[0, 0, :, :]:
        confidence = detection[2]
        if confidence > .5:
            class_id = detection[1]
            class_name=id_class_name(class_id,classNames)
            print(str(str(class_id) + " " + str(detection[2])  + " " + class_name))
            espeak.synth("there is")
            espeak.synth(class_name)
            box_x = detection[3] * image_width
            box_y = detection[4] * image_height
            box_width = detection[5] * image_width
            box_height = detection[6] * image_height
            cv2.rectangle(image, (int(box_x), int(box_y)), (int(box_width),
int(box_height)), (23, 230, 210), thickness=1)
```

```
            cv2.putText(image,class_name ,(int(box_x),
int(box_y+.05*image_height)),cv2.FONT_HERSHEY_SIMPLEX,(.005*image_w
idth),(0, 0, 255))







    cv2.imshow('image', image)
    cv2.imwrite("image_box_text.jpg",image)

    cv2.waitKey(10)
    cv2.destroyWindow('image')
    os.remove("image.jpg")





    looping()
```

looping()

Push

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
def button_callback(channel):
    print("Button was pushed!")
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(8, 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)
while True: # Run forever
    if GPIO.input(8) == GPIO.HIGH:
        print("Button was pushed!")
        break
```

Real Time Object Detection

```
# USAGE
# python real_time_object_detection.py --prototxt
MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2
import picamera


# construct the argument parse and parse the arguments
```

```
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
    help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
```

```
# initialize the video stream, allow the cammera sensor to warmup,
# and initialize the FPS counter
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
fps = FPS().start()

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # grab the frame dimensions and convert it to a blob
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
        0.007843, (300, 300), 127.5)
```

```
# pass the blob through the network and obtain the detections and
# predictions
net.setInput(blob)
detections = net.forward()

# loop over the detections
for i in np.arange(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with
    # the prediction
    confidence = detections[0, 0, i, 2]

    # filter out weak detections by ensuring the `confidence` is
    # greater than the minimum confidence
    if confidence > args["confidence"]:
        # extract the index of the class label from the
        # `detections`, then compute the (x, y)-coordinates of
        # the bounding box for the object
        idx = int(detections[0, 0, i, 1])
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")
```

```
			# draw the prediction on the frame
			label = "{}: {:.2f}%".format(CLASSES[idx],
				confidence * 100)
			cv2.rectangle(frame, (startX, startY), (endX, endY),
				COLORS[idx], 2)
			y = startY - 15 if startY - 15 > 15 else startY + 15
			cv2.putText(frame, label, (startX, y),
				cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

	# show the output frame
	cv2.imshow("Frame", frame)
	key = cv2.waitKey(1) & 0xFF

	# if the `q` key was pressed, break from the loop
	if key == ord("q"):
		break

	# update the FPS counter
	fps.update()
```

```
# stop the timer and display FPS information

fps.stop()

print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))

print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))


# do a bit of cleanup

cv2.destroyAllWindows()

vs.stop()
```