

**BLIND – HELP**  
**A PROJECT REPORT**

*Submitted by*

**MALATHI V - 312317205077**

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**INFORMATION TECHNOLOGY**



**St. JOSEPH'S COLLEGE OF ENGINEERING**

**(An Autonomous Institution)**

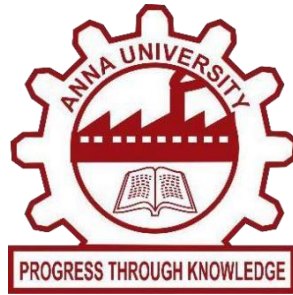
**St. Joseph's Groups of Institution**

**Jeppiaar Educational Trust**

**OMR, Chennai 600 119**

**ANNA UNIVERSITY: CHENNAI**

**July-2021**



**BONAFIDE CERTIFICATE**

Certified that this project report “ **BLIND HELP** ” is the bonafide work of **MALATHI V (312317205077)** who carried out the project under my supervision.

**SIGNATURE**

**Project Guide,  
Mr. N.Raja mohamed, M.E.,**

**Associate Professor,  
Department of IT,  
St. Joseph's College of  
Engineering, OMR, Chennai-  
600119.**

**SIGNATURE**

**Head of the department,  
Dr. V.Muthulakshmi, M.E., Ph.D,**

**Associate Professor,  
Department of IT,  
St. Joseph's College of  
Engineering, OMR, Chennai-  
600119.**

## CERTIFICATE OF EVALUATION

**COLLEGE NAME** : St. Joseph's College of Engineering.

**BRANCH** : **B.TECH. IT** (INFORMATION TECHNOLOGY)

**SEMESTER** VIII

| SL.NO | NAME OF THE STUDENT         | TITLE OF THE PROJECT | NAME OF THE SUPERVISOR WITH DESIGNATION                |
|-------|-----------------------------|----------------------|--|
| 1     | MALATHI V<br>(312317205077) | BLIND HELP           | MR.N. RAJA MOHAMAD,<br>M.E.,<br>ASSOCIATE<br>PROFESSOR |

The report of the project work submitted by the above student in partial fulfillment for the award of Bachelor of Technology Degree in Electronics and Instrumentation Engineering of Anna University was confirmed to be report of the work done by the above student and then evaluated.

Submitted to Project and Viva Examination held on\_\_\_\_\_.

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

At the outset we would like to express my sincere gratitude to our beloved **Chairman, Dr.Babu Manoharan, M.A.,M.B.A.,Ph.D.**, for his constant guidance and support.

We would like to express our heartfelt thanks to our respected **Managing Director Mrs. S. Jessie Priya, M.Com.**, for her kind encouragement and blessings.

We wish to express our sincere thanks to **Director Mr. B. Shashi Sekar, M.Sc.**, for providing ample facilities in the institution.

We express deepest gratitude and thanks to our beloved **Principal Dr.Vaddi Seshagiri Rao, M.E., M.B.A., Ph.D., F.I.E.**, for his inspirational ideas during the course of the project.

We wish to express our sincere thanks and gratitude to **Dr.V.Muthulakshmi, M.E., Ph.D.**, Head of the Department , Department of Information Technology, St.Joseph's College of Engineering for her guidance and assistance in solving the various intricacies involved in the project.

It is with deep sense of gratitude that we acknowledge our indebtedness to our supervisor **Mr.N.Raja Mohamad, M.E.**, for her expert guidance and connoisseur suggestion.

Finally we thank our department staff members who helped us in the successful completion of this project.

## **ABSTRACT**

Walking safely and confidently without any human assistance in urban or unknown environments is a difficult task for blind people. Blind people face several problems in their life, one of these issues that is the most vital one is identification the hindrances when they are walking. When moving from one place to another, they need help of other people around. Their independency in strolling is lost. Sticks can be usable but are not that reliable nor does everyone have it. A visually impaired person needs absolution to help him overcome problems in navigation due to his disability. The project is mainly focused on providing a type of visual aid to the visually impaired people. With the current advances in comprehensive innovation, it is conceivable to stretch out the help given to individuals with visual hindrance during their mobility. In this context we propose system in which an Android smartphone is used to help a blind user in obstacle detection and navigation. Today, smartphones are available to anyone. In fact, they have become the most common device available everywhere. Hence, this project uses an Android smartphone that uses its camera to identify objects in surroundings and gives an audio output. The hearing ability of the user tries to fulfil his seeing ability.

## TABLE OF CONTENTS

| CHAPTER  | TITLE                                  | PAGE NO. |
|----------|--|----------|
|          | ABSTRACT                               | v        |
|          | LIST OF FIGURES                        | vii      |
| <b>1</b> | <b>INTRODUCTION</b>                    |          |
|          | 1.1 OBJECT DETECTION                   | 1        |
|          | 1.2 SPEECH SYNTHESIS                   | 2        |
|          | 1.3 MACHINE LEARNING                   | 3        |
|          | 1.4 SYSTEM OVERVIEW                    | 4        |
|          | 1.5 SCOPE OF THE PROJECT               | 5        |
| <b>2</b> | <b>LITERATURE SURVEY</b>               | 6        |
| <b>3</b> | <b>SYSTEM ANALYSIS</b>                 |          |
|          | 3.1 EXISTING SYSTEM                    | 13       |
|          | 3.1.1 Disadvantages of existing system |          |
|          | 3.2 PROPOSED SYSTEM                    | 14       |
|          | 3.2.1 Advantages of proposed system    |          |
| <b>4</b> | <b>REQUIREMENT SPECIFICATION</b>       |          |
|          | 3.2.2 Hardware Requirements            | 15       |
|          | 3.2.3 Software Requirements            | 15       |
|          | 3.3 LANGUAGE SPECIFICATION             | 15       |
| <b>5</b> | <b>SYSTEM DESIGN</b>                   |          |
|          | 5.1 SYSTEM ARCHITECTURE                | 23       |
|          | 5.2 SEQUENCE DIAGRAM                   | 24       |

|          |  |    |
|----------|--|----|
|          | 5.3 USE CASE DIAGRAM                     | 25 |
|          | 5.4 STATE DIAGRAM                        | 26 |
| <b>6</b> | <b>MODULE DESCRIPTION</b>                |    |
|          | 6.1 MODULES                              | 27 |
|          | 6.1.1 READING INPUT IMAGE                | 27 |
|          | 6.1.2 OBJECT DETECTION                   | 28 |
|          | 6.1.3 OBJECT CLASSIFICATION              | 36 |
|          | 6.1.4 DISTANCE CALCULATION               | 41 |
|          | 6.1.5 OUTPUT VOICE                       | 43 |
| <b>7</b> | <b>CONCLUSION AND FUTURE ENHANCEMENT</b> |    |
|          | 7.1 CONCLUSION                           | 45 |
|          | 7.2 FUTURE ENHANCEMENT                   | 45 |
|          | <b>APPENDIX 1</b>                        | 46 |
|          | <b>APPENDIX 2</b>                        | 61 |
|          | <b>REFERENCES</b>                        | 63 |

## LIST OF FIGURES

| FIGURE NO | TITLE  | PAGE NO |
|-----------|--|---------|
| 5.1       | ARCHITECTURE OF PROPOSED SYSTEM                      | 23      |
| 5.2       | SEQUENCE DIAGRAM                                     | 24      |
| 5.3       | USE CASE DIAGRAM                                     | 25      |
| 5.4       | STATE DIAGRAM  | 26      |
| 6.1.1     | READING INPUT IMAGE                                  | 27      |
| 6.1.2     | OBJECT DETECTION INPUT IMAGE                         | 28      |
| 6.1.2.1   | DIVIDING IMAGES INTO REGIONS                         | 29      |
| 6.1.2.2   | DETECTED IMAGE                                       | 29      |
| 6.1.2.3   | CONVOLUTIONAL NEURAL NETWORK                         | 30      |
| 6.1.2.4   | CLASSIFICATION - FULLY CONNECTED LAYER               | 32      |
| 6.1.2.5   | YOLO OBJECT DETECTION WITH OPENCV                    | 35      |
| 6.1.3     | OBJECT CLASSIFICATION                                | 36      |
| 6.1.3.1   | DARKNET CONVOLUTIONAL NEURAL<br>NETWORK ARCHITECTURE | 38      |
| 6.1.3.2   | COCO PANOPTIC SEGMENTATION                           | 40      |
| 6.1.3.3   | COCO KEYPOINT DETECTION                              | 40      |
| 6.1.4     | OBJECT NEARER TO SCREEN                              | 42      |
| 6.1.4.1   | OBJECT AWAY FROM THE SCREEN                          | 42      |
| 6.1.5     | OUTPUT SPEECH  | 43      |



# **CHAPTER 1**

## **INTRODUCTION**

Visually impaired people heavily rely on their other senses such as touch and auditory signals for understanding the environment around them. The act of knowing what object is in front of the blind person without touching it. This project tries to transform the visual world into the audio world with the potential to inform blind people objects. Objects detected from the scene are represented by their names and converted to speech.

### **1.1 OBJECT DETECTION**

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin colour and distance between eyes can be found.

Methods for object detection generally fall into either neural network-based or non-neural approaches. For non-neural approaches, it becomes necessary to first define features using one of the methods below, then using a technique such

as support vector machine (SVM) to do the classification. On the other hand, neural techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

## **1.2 SPEECH SYNTHESIS**

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented in software or hardware products. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech. Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diphones provides the largest output range, but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high-quality output. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely "synthetic" voice output.

The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood clearly. An intelligible text-to-speech program allows people with visual impairments or reading disabilities to listen to written words on a home computer. Many computer operating systems have included speech synthesizers since the early 1990s. A text-to-speech system (or "engine") is composed of two parts: a front-end and a back-end. The front-end has two major tasks. First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization, pre-processing, or tokenization. The front-end

then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. The process of assigning phonetic transcriptions to words is called text-to-phoneme or grapheme-to-phoneme conversion. In certain systems, this part includes the computation of the target prosody (pitch contour, phoneme durations), which is then imposed on the output speech.

### **1.3 DEEP LEARNING**

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

- Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.
- Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled.
- Deep learning, a form of machine learning, can be used to help detect fraud or money laundering, among other functions.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

## **1.4 SYSTEM OVERVIEW**

Among the various technologies used for blind people, the majority is aids of mobility and obstacle detection. They are based on rules for converting images into data sensory substitution tactile or auditory stimuli. These systems are efficient for mobility and localization of objects which is sometimes with a lower precision. However, one of the greatest difficulties of blind people is the identification of their environment and its. Indeed, they can only be used to recognize simple patterns and cannot be used as tools of substitution in natural environments. Also, they don't identify object and they have in some cases a late detection of small objects. In addition, some of them seek additional auditory, others require a sufficiently long period for learning and testing. For this reason, we are interested in the evaluation of an algorithm for fast and robust computer vision application to recognize and locate objects in a video scene. Thus, it is important to design a system based on the recognition and detection of objects to meet the major challenges of the blind in three main categories of needs: displacement, orientation and object identification.

The project Machine learning-based blind aid is added with certain new features compared with the existing ones. Due to the poor performance of the previous models, people have started to use ML in object detection. The utilization of a machine learning method allows for significant accuracy. Machine learning has become an interesting topic in visual recognition since 2012 because it can classify the images with high accuracy. It has become the best method in solving

image classification problems in visual recognition and object detection. The COCO object detection model allows for a properly trained network to respond correctly when an image or pattern is presented for recognition. It is fast in object recognition. The recognized object is converted to speech so that it assists the blind people. Pyaudio is used for the conversion object detection to speech.

## **1.5 SCOPE OF THE PROJECT :**

Vision is one of the very essential human senses and it plays the most important role in human perception about surrounding environment. Hence, over thousands of papers have been published on these subjects that propose a variety of computer vision products and services by developing new electronic aids for the blind. This paper aims to introduce a proposed system that restores a central function of the visual system which is the identification of surrounding objects. This method is based on the local feature extraction concept. Good vision is a precious gift but unfortunately loss of vision is becoming common now a days. To help the blind people the visual world has to be transformed into the audio world with the potential to inform them about objects as well as the distance between the objects. Objects detected from the scene are represented by their names and converted to speech.

## **CHAPTER 2**

### **LITERATURE SURVEY**

**[1] M Afif, R Ayachi, Y Said, E Pissaloux, M Atri - Neural Processing Letters, “An Evaluation of RetinaNet on Indoor Object Detection for Blind and Visually Impaired Persons Assistance Navigation”, 2020**

Indoor object detection presents a computer vision task that deals with the detection of specific indoor classes. This task attracts a lot of attention, especially in the last few years. The strong interest related to this field can be explained by the big importance of this task for indoor assistance navigation for visually impaired people and also by the phenomenal development of the deep convolutional neural networks (Deep CNN). An effort is made to perform a new indoor object detector using the deep convolutional neural network-based framework. The framework is built based on the deep convolutional neural network “RetinaNet”. Evaluation is done by using various backbones as ResNet, DenseNet, and VGGNet in order to improve detection performances and processing time. They obtained very encouraging results coming up to 84.61% mAP as detection precision.

**[2] M Anandan, M Manikandan, T Karthick, “ Advanced Indoor and Outdoor Navigation System for Blind People Using Raspberry-Pi ”, 2020**

M Anandan, M Manikandan, T Karthick focuses on providing the helping aid for the visually impaired person with an object detection and navigation system. The proposed module is divided into two, one is static object detection which uses Threshold value matching, SURF feature extraction for obstacle

matching and Bivariate gaussian mixture model for high dimensional bivariate features extracted and later to convert the obtained visual information into the audio information using a raspberry-pi setup. Secondly the system deals with the dynamic obstacles too for which employs Gaussian mixture distribution in modelling background of each pixel and an on-line approximation to update the model. The moving human region is detected by background subtraction. The shadow detection and elimination are implemented in the HSV space, and finally morphological operations are introduced to eliminate noise and reconstruct moving human regions. The results show that the method models stable background, eliminates shadow, and has a good detection result. This algorithm detects the shape of the object and searches from the database provided in the hardware and converts the textual information to an audio information. The audio information is given in-terms of a vibration which protects the eardrum and also helps as a hearing aid for the partial hearing loss people. This prototype provides the blind people to navigate in a free environment.

**[3] A Shinde, P Bhagat, S Dhobale, “Object detection and recognition for blind assistance”, 2019**

Vision is one of the essential human senses and it plays the most important role in human perception about surrounding environment. The blind people face difficulties in safe and independent mobility, issues of communication and access to information. They propose an object recognition algorithm, and an assistive system which is very useful for their safety, quality life and freedom from other person all the time. The purpose of this system is to make the visually challenged able to take decisions independently based on the audio output from this model which can be trained according to their needs. Convolutional Neural Network is designed to achieve better accuracy. It is implemented using multimedia processor equipped embedded board and OpenCV.

**[4] C Luo, L Jin, Z Sun , “ A multi-object rectified attention network for scene text recognition”, 2019**

Irregular text is widely used. However, it is considerably difficult to recognize because of its various shapes and distorted patterns. In the paper, they thus propose a multi-object rectified attention network (MORAN) for general scene text recognition. The MORAN consists of a multi-object rectification network and an attention-based sequence recognition network. The multi-object rectification network is designed for rectifying images that contain irregular text. It decreases the difficulty of recognition and enables the attention-based sequence recognition network to more easily read irregular text. It is trained in a weak supervision way, thus requiring only images and corresponding text labels. The attention-based sequence recognition network focuses on target characters and sequentially outputs the predictions. Moreover, to improve sensitivity of the attention-based sequence recognition network, a fractional pickup method is proposed for an attention-based decoder in the training phase. With the rectification mechanism, the MORAN can read both regular and irregular scene text. Extensive experiments on various benchmarks are conducted, which show that the MORAN achieves state-of-the-art performance. The source code is available.

**[5] Y Jia, Y Zhang, RJ Weiss, Q Wang, J Shen, “Transfer learning from speaker verification to multispeaker text-to-speech synthesis”, 2018**

They describe a neural network-based system for text-to-speech (TTS) synthesis that is able to generate speech audio in the voice of many different speakers, including those unseen during training. Our system consists of three independently trained components: (1) a speaker encoder network, trained on a speaker verification task using an independent dataset of noisy speech from thousands of speakers without transcripts, to generate a fixed-dimensional



embedding vector from seconds of reference speech from a target speaker; (2) a sequence-to-sequence synthesis network based on Tacotron 2, which generates a mel spectrogram from text, conditioned on the speaker embedding; (3) an autoregressive WaveNet-based vocoder that converts the mel spectrogram into a sequence of time domain waveform samples. They demonstrate that the proposed model is able to transfer the knowledge of speaker variability learned by the discriminatively-trained speaker encoder to the new task, and is able to synthesize natural speech from speakers that were not seen during training. They quantify the importance of training the speaker encoder on a large and diverse speaker set in order to obtain the best generalization performance. Finally, they show that randomly sampled speaker embeddings can be used to synthesize speech in the voice of novel speakers dissimilar from those used in training, indicating that the model has learned a high quality speaker representation.

**[6] SO Arik, M Chrzanowski, A Coates, “Deep voice: Real-time neural text-to-speech”, 2017**

They present Deep Voice, a production-quality text-to-speech system constructed entirely from deep neural networks. Deep Voice lays the groundwork for truly end-to-end neural speech synthesis. The system comprises five major building blocks: a segmentation model for locating phoneme boundaries, a grapheme-to-phoneme conversion model, a phoneme duration prediction model, a fundamental frequency prediction model, and an audio synthesis model. For the segmentation model, they propose a novel way of performing phoneme boundary detection with deep neural networks using connectionist temporal classification (CTC) loss. For the audio synthesis model, they implement a variant of WaveNet that requires fewer parameters and trains faster than the original. By using a neural network for each component, our system is simpler and more flexible than traditional text-to-speech systems, where each component requires laborious feature engineering and extensive domain expertise. Finally, they show that

inference with our system can be performed faster than real time and describe optimized WaveNet inference kernels on both CPU and GPU that achieve up to 400x speedups over existing implementations.

**[7] B Shi, X Bai, C Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition” 2016**

Image-based sequence recognition has been a long-standing research topic in computer vision. In the paper, they investigate the problem of scene text recognition, which is among the most important and challenging tasks in image-based sequence recognition. A novel neural network architecture, which integrates feature extraction, sequence modelling and transcription into a unified framework, is proposed. Compared with previous systems for scene text recognition, the proposed architecture possesses four distinctive properties: (1) It is end-to-end trainable, in contrast to most of the existing algorithms whose components are separately trained and tuned. (2) It naturally handles sequences in arbitrary lengths, involving no character segmentation or horizontal scale normalization. (3) It is not confined to any predefined lexicon and achieves remarkable performances in both lexicon-free and lexicon-based scene text recognition tasks. (4) It generates an effective yet much smaller model, which is more practical for real-world application scenarios. The experiments on standard benchmarks, including the IIIT-5K, Street View Text and ICDAR datasets, demonstrate the superiority of the proposed algorithm over the prior arts. Moreover, the proposed algorithm performs well in the task of image-based music score recognition, which evidently verifies the generality of it.

**[8] S Karaoglu, JC Van Gemert, T Gevers, “Object reading: text recognition for object recognition”, 2012**

They propose to use text recognition to aid in visual object class recognition. To this end they first propose a new algorithm for text detection in natural images. The proposed text detection is based on saliency cues and a context fusion step. The algorithm does not need any parameter tuning and can deal with varying imaging conditions. They evaluate three different tasks: 1. Scene text recognition, where we increase the state-of-the-art by 0.17 on the ICDAR 2003 dataset. 2. Saliency based object recognition, where they outperform other state-of-the-art saliency methods for object recognition on the PASCAL VOC 2011 dataset. 3. Object recognition with the aid of recognized text, where they are the first to report multi-modal results on the IMET set. Results show that text helps for object class recognition if the text is not uniquely coupled to individual object instances.

**[9] SM Murtoza, F Alam, R Sultana, S Absar, “Phonetically balanced Bangla speech corpus”, 2011**

SM Murtoza, F Alam, R Sultana, S Absar describes the development of a phonetically balanced Bangla speech corpus. Construction of speech applications such as text to speech and speech recognition requires a phonetically balanced speech database in order to obtain a natural output. Here they elicited text collection procedure, text normalization, G2P 1 conversion and optimal text selection using a greedy selection method and hand pruning.

**[10] L Dunai, GP Fajarnes, VS Praderas – IECON, “Real time assistance prototype - a new navigation aid for blind people”, 2010**

The paper presents a new prototype for being used as a travel aid for blind people. The system is developed to complement traditional navigation systems such as white cane and guide dogs. The system consists of two stereo cameras and a portable computer for processing the environmental information. The aim of the system is to detect the static and dynamic objects from the surrounding environment and transform them into acoustical signals. Through stereophonic headphones, the user perceives the acoustic image of the environment, the volume of the objects, moving object direction and trajectory, its distance relative to the user and the free paths in a range of 5m to 15m. The acoustic signals represent short train of delta sounds externalized with non-individual Head-Related Transfer Functions generated in an anechoic chamber. Experimental results show that users were able to control and navigate with the system safely both in familiar and unfamiliar environments.

## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

A vision based system to help visually impaired persons navigate through their surroundings. This system consists of a camera, a haptic feedback device to aware blind persons of an object, and an embedded computer. This system is wearable by the blind users thus providing mobility to users. Regional proposals are used to find empty spaces through the navigation path. The authors test their system by making a blind person walk through a maze wearing their system. The experiment results show that this system greatly helps blind user to navigate through a path without collisions. However, navigation using this system is slower as compared to using cane. If this system is combined with cane for blind user, the speed of navigation can be improved. [2] in their paper propose an assistive system to help blind persons read using the OCR algorithm. This system is in the form of a glove with camera embedded index that can be worn by the blind user. The blind user after wearing the glove will place their index finger over the first sentence from left to right. The camera underneath the finger will process the image of the text and give output in the form of audio. The experiment results show that the finger can successfully read the text under the camera, but the process is slow as the algorithm uses multiple images of the same frame to create a high resolution image. The feasibility of the project is also a matter consideration as the experiments were conducted on a webcam mounted on a table for reading the text. The actual result may vary if finger tip camera is used. An assistive system to help blind users. Their system makes use of Raspberry Pi, which is an embedded computer for processing text present in the images. While it uses the OCR algorithm, and the next uses YOLO v1. Text to Speech (TTS) library is used to read out the text to the blind user. The camera is placed on the

glasses of the user and is connected to Raspberry Pi. The experiment results show that the OCR algorithm is good in reading text but suffers from poor performance in face detection.

### **3.1.1 DISADVANTAGES OF THE EXISTING SYSTEM**

- The existing system uses Raspberry Pi and smart glasses as a navigation system which is an external hardware need to be brought.
- Many voice based systems such as voice based e-mail, voice based mobile features etc. which doesn't detect any objects.
- The other systems focus only on internal objects and not on external objects.

### **3.2 PROPOSED SYSTEM**

We have decided that implementing the system by using mobile application will provide blind persons mobility and ease of use. In our system, the blind person will hold the mobile phone in their hand while navigating through the streets while the application is running. The application will identify the objects present in the frame and announce the name of the object detected in the form of audio through headphones worn by the blind user. The audio announcement consists of the name of the object as well the distance. The blind user can then use this information to navigate.

#### **3.2.1 ADVANTAGES OF THE PROPOSED SYSTEM :**

- The system was completely a software project need not any hardware to be implemented.
- It focuses on both internal and external objects to be recognized.
- The distance between the object and the user was been calculated.

## **CHAPTER 4**

### **REQUIREMENTS SPECIFICATION**

#### **4.1 Hardware Requirements**

- Disk Space 32 GB or more,10 GB or more for Foundation Edition
- Processor 1.4 GHz 64 bit
- Memory 512 MB
- Display (800 × 600) Capable video adapter and monitor

#### **4.2 Software Requirements**

- Windows operating system XP and above
- Python IDLE
- OpenCV
- Torch
- Numpy

#### **4.3 LANGUAGE SPECIFICATION**

##### **4.3.1 About Python**

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features

dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has `filter`, `map`, and `reduce` functions; list comprehensions, dictionaries, sets, and generator expressions.<sup>[67]</sup> The standard library has two modules (`itertools` and `functools`) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy.

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer



marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

## **Numpy :**

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

## **OPENCV :**

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.<sup>[1]</sup> Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel<sup>[2]</sup>). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object detection
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation.<sup>[12]</sup> Wrappers in several programming languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms.

## **TORCH :**

Torch is an open-source machine learning library, a scientific computing framework, and a script language based on the Lua programming

language.<sup>[3]</sup> It provides a wide range of algorithms for deep learning, and uses the scripting language LuaJIT, and an underlying C implementation. As of 2018, Torch is no longer in active development.<sup>[4]</sup> However PyTorch, which is based on the Torch library, is actively developed as of December 2020. The core package of Torch is `torch`. It provides a flexible N-dimensional array or Tensor, which supports basic routines for indexing, slicing, transposing, type-casting, resizing, sharing storage and cloning. This object is used by most other packages and thus forms the core object of the library. The Tensor also supports mathematical operations like `max`, `min`, `sum`, statistical distributions like uniform, normal and multinomial, and BLAS operations like dot product, matrix-vector multiplication, matrix-matrix multiplication, matrix-vector product and matrix product.

The `torch` package also simplifies object oriented programming and serialization by providing various convenience functions which are used throughout its packages. The `torch.class(classname, parentclass)` function can be used to create object factories (classes). When the constructor is called, torch initializes and sets a Lua table with the user-defined metatable, which makes the table an object.

Objects created with the torch factory can also be serialized, as long as they do not contain references to objects that cannot be serialized, such as Lua coroutines, and Lua *userdata*. However, *userdata* can be serialized if it is wrapped by a table (or metatable) that provides `read()` and `write()` methods. The `nn` package is used for building neural networks. It is divided into modular objects that share a common `Module` interface. Modules have a `forward()` and `backward()` method that allow them to feedforward and backpropagate, respectively. Modules can be joined together

using module composites, like Sequential, Parallel and Concat to create complex task-tailored graphs. Simpler modules like Linear, Tanh and Max make up the basic component modules. This modular interface provides first-order automatic gradient differentiation.

#### **4.3.2. PYTHON IDLE :**

IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

- coded in 100% pure Python, using the tkinter GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and macOS
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

#### **Automatic indentation**

After a block-opening statement, the next line is indented by 4 spaces (in the Python Shell window by one tab). After certain keywords (break, return etc.) the next line is dedented. In leading indentation, Backspace deletes up to 4 spaces if they are there. Tab inserts spaces (in the Python Shell window one tab), number depends on Indent width. Currently, tabs are restricted to four spaces due to Tcl/Tk limitations. Also the indent/dedent region commands on the Format menu.

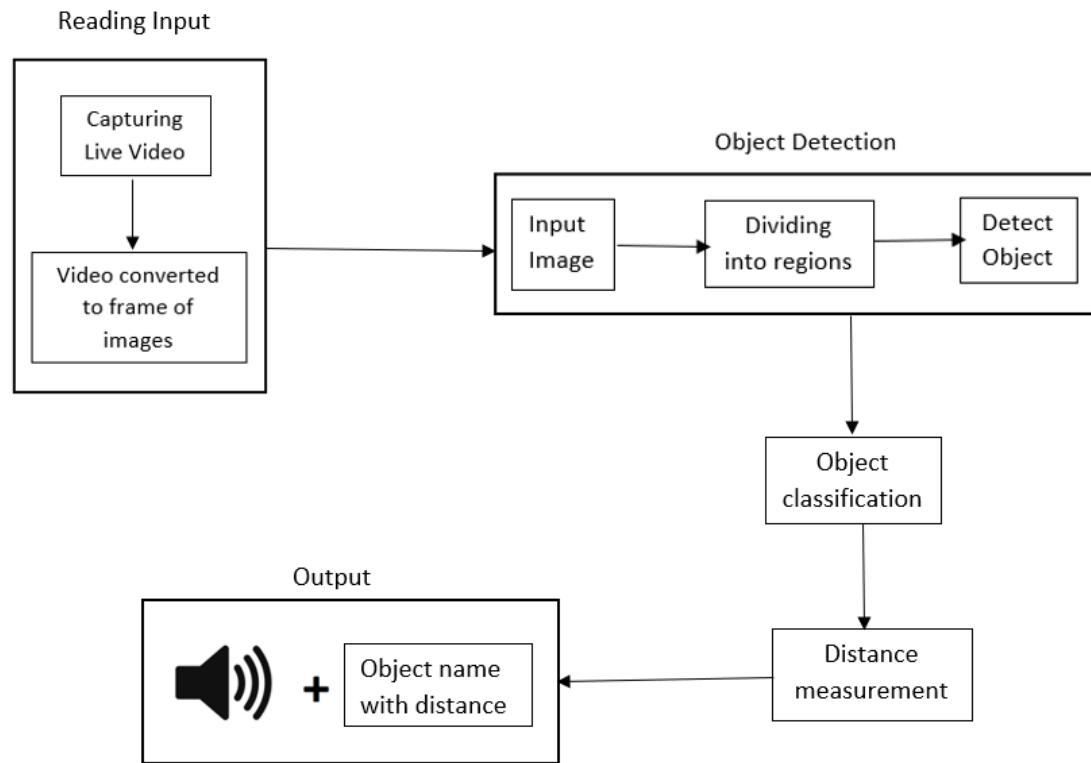
## Completions

Completions are supplied, when requested and available, for module names, attributes of classes or functions, or filenames. Each request method displays a completion box with existing names. (See tab completions below for an exception.) For any box, change the name being completed and the item highlighted in the box by typing and deleting characters; by hitting Up, Down, PageUp, PageDown, Home, and End keys; and by a single click within the box. Close the box with Escape, Enter, and double Tab keys or clicks outside the box. A double click within the box selects and closes.

## CHAPTER 5

### SYSTEM DESIGN

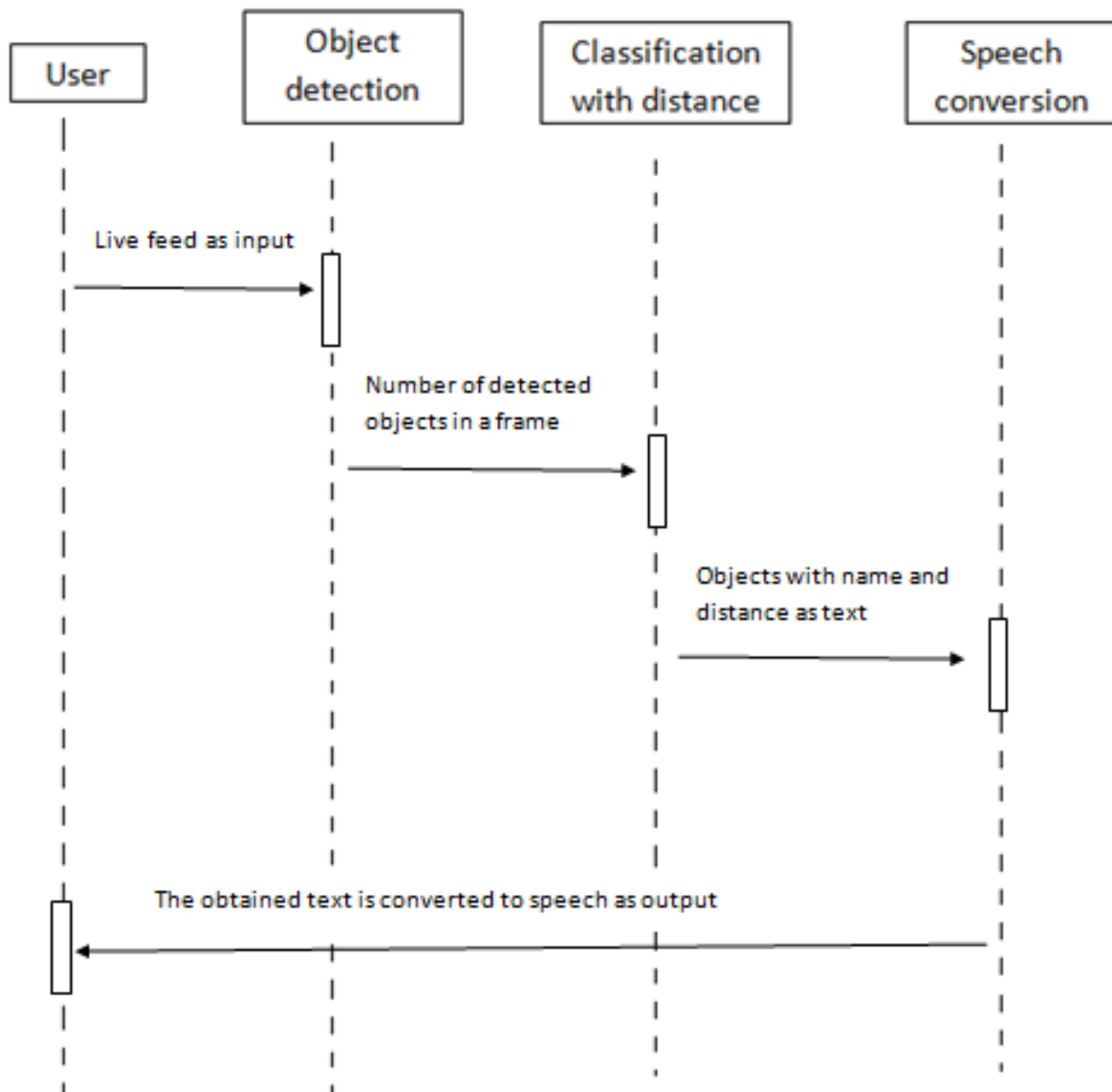
#### 5.1 SYSTEM ARCHITECTURE



**Figure 5.1 Architecture of proposed system**

The figure 5.1 represents the Architecture of proposed system. A application will provide blind persons mobility and ease of use. In our system, the blind person will hold the mobile phone in their hand while navigating through the streets while the application is running. The application will identify the objects present in the frame and announce the name of the object detected in the form of audio through headphones worn by the blind user. The audio announcement consists of the name of the object as well the distance. The blind user can then use this information to navigate.

## 5.2 SEQUENCE DIAGRAM

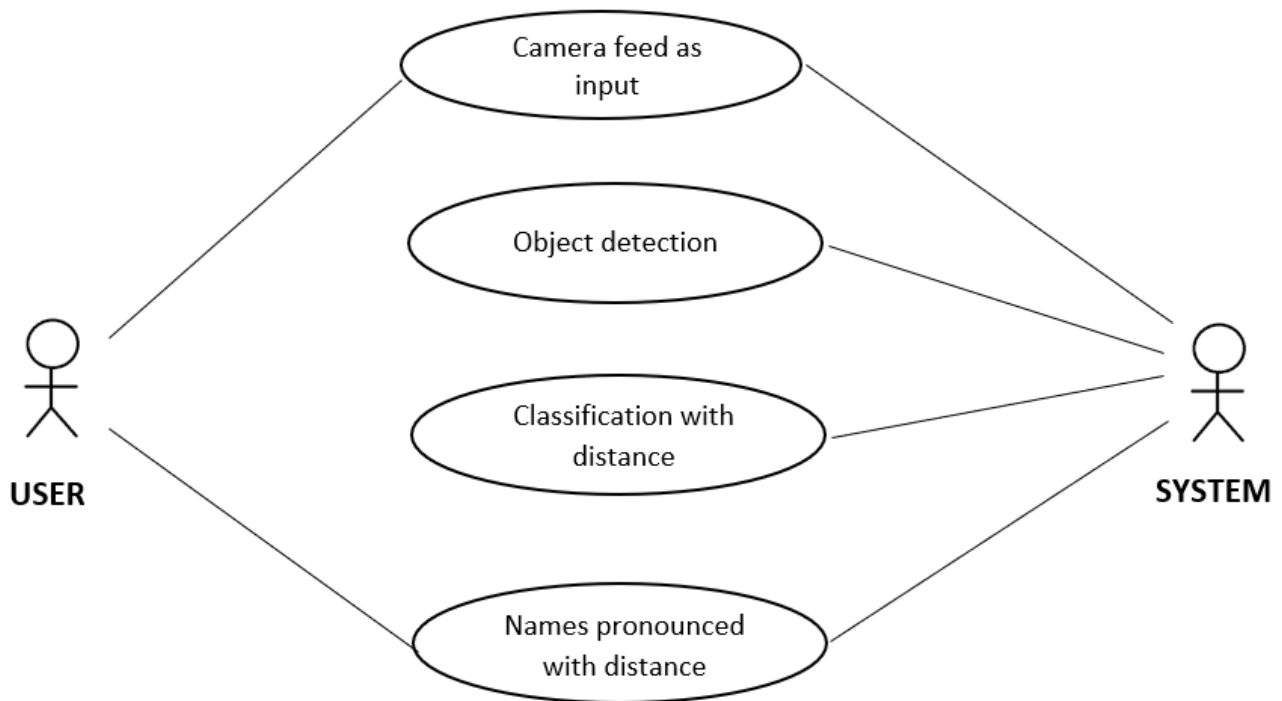


**FIGURE 4.2 SEQUENCE DIAGRAM**

The figure 4.2 represents the sequence diagram. Live camera feed is given as input from the user to the system. The process called object detection is executed to find the n number of objects present in the video frame. Then the detected objects get separated under various classes of COCO dataset of MobileNet. The distance between the objects and the person holding the mobile camera will be calculated and finally the object name and the distance will be announced.



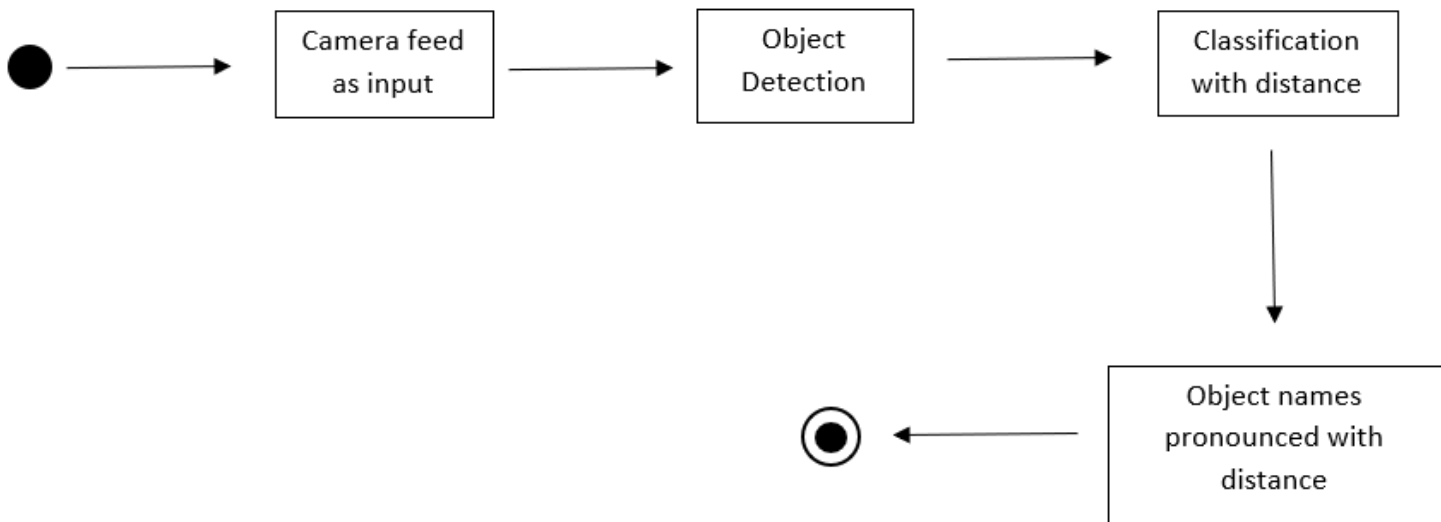
### 5.3 USE CASE DIAGRAM :



**FIGURE 5.3 USE CASE DIAGRAM**

The figure 5.3 represents the use case diagram. User will hold the mobile in which the live video will be captured by the camera. The captured video will be given as the input to the system. The process called object detection will be carried out by the system to find the numerous object present in the video frame. Then the detected objects will be classified under various classes. The distance between the objects and user will calculated. And finally the output will as a speech pronounced with the object name and its distance.

## 5.4 STATE DIAGRAM



**FIGURE 5.4 STATE DIAGRAM**

The figure 5.4 represents the state diagram. The first basic step of this project is to feed the live camera input to the system. The system will then convert the live video into numerous video frames. The process called object detection is executed to find the n number of objects present in the video frame. Then the detected objects get separated under various classes of COCO dataset of MobileNet. The distance between the objects and the person holding the mobile camera will be calculated and finally the object name and the distance will be announced.

## **CHAPTER 6**

### **MODULE DESCRIPTION**

#### **6.1 MODULES**

The modules are

- Reading input image
- Object detection
- Object classification
- Distance calculation
- Output speech

##### **6.1.1 READING INPUT IMAGE :**

This module reads the input from the device camera. The user will be holding the device and the live video will be given as input to the system. The figure 6.1.1 represents Reading input image.



**Figure 6.1.1 Reading input image**

This module basically performs two major steps,

- Capturing live video
- Converting video to frames

The live video will be captured from the camera hold by the user. Video is defined as number of frames per second. Frame rate, the number of still pictures per unit of time of video, ranges from six or eight frames per second. The capturing live video will be converted to numerous numbers of frames. Those frames will be given as input sequence. This leads to the next process called object detection.

### **6.1.2 OBJECT DETECTION :**

With the obtained input from the above module, we proceed with the next module called object detection. The steps involved in object detection are,

1. First, we take an image as input, Figure 6.1.2 represents Object detection input image.



**Figure 6.1.2 Object detection input image**

2. Then we divide the image into various regions, figure 6.1.2.1 represents Diving images into regions.



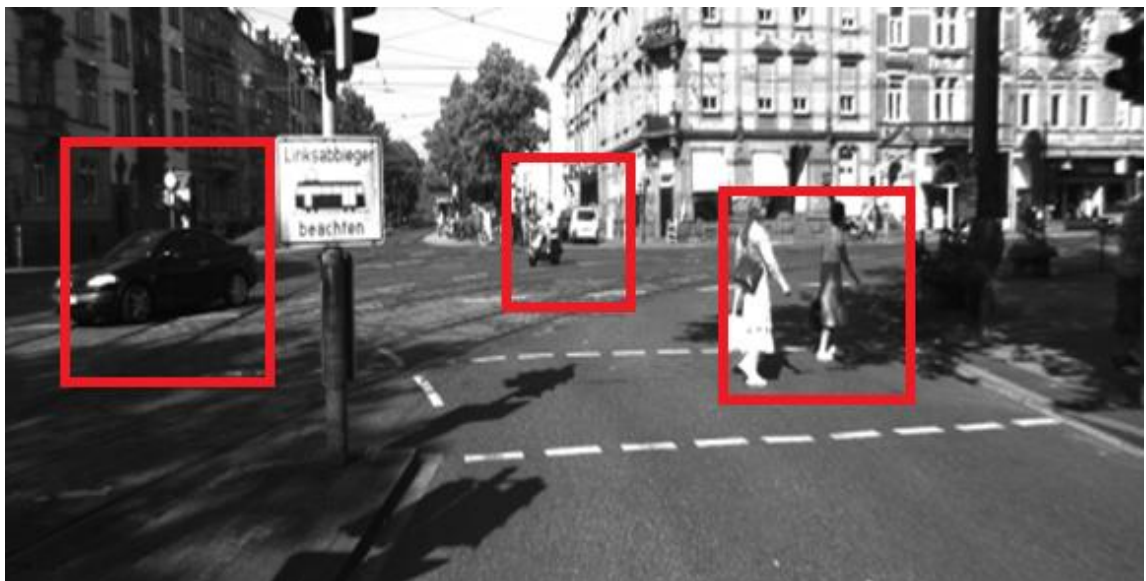
**Figure 6.1.2.1 Diving image into regions**

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

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

5. Combine all these regions to get the original image with the detected objects.

Figure 6.1.2.2 represents the Detected image.



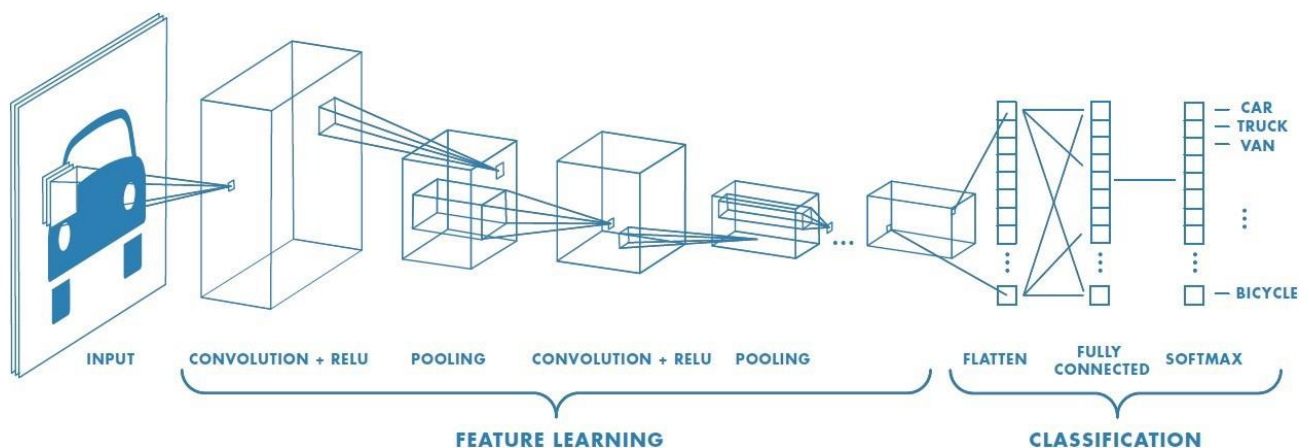
**Figure 6.1.2.2 Detected image**



## CNN :

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modelling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.



**Figure 6.1.2.3 convolutional neural network**

The above figure 6.1.2.3 represents the CNN. The convolutional neural network, or CNN for short, is a specialized type of neural network model designed for working with two-dimensional image data, although they can be used with one-dimensional and three-dimensional data.

Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a “convolution”.

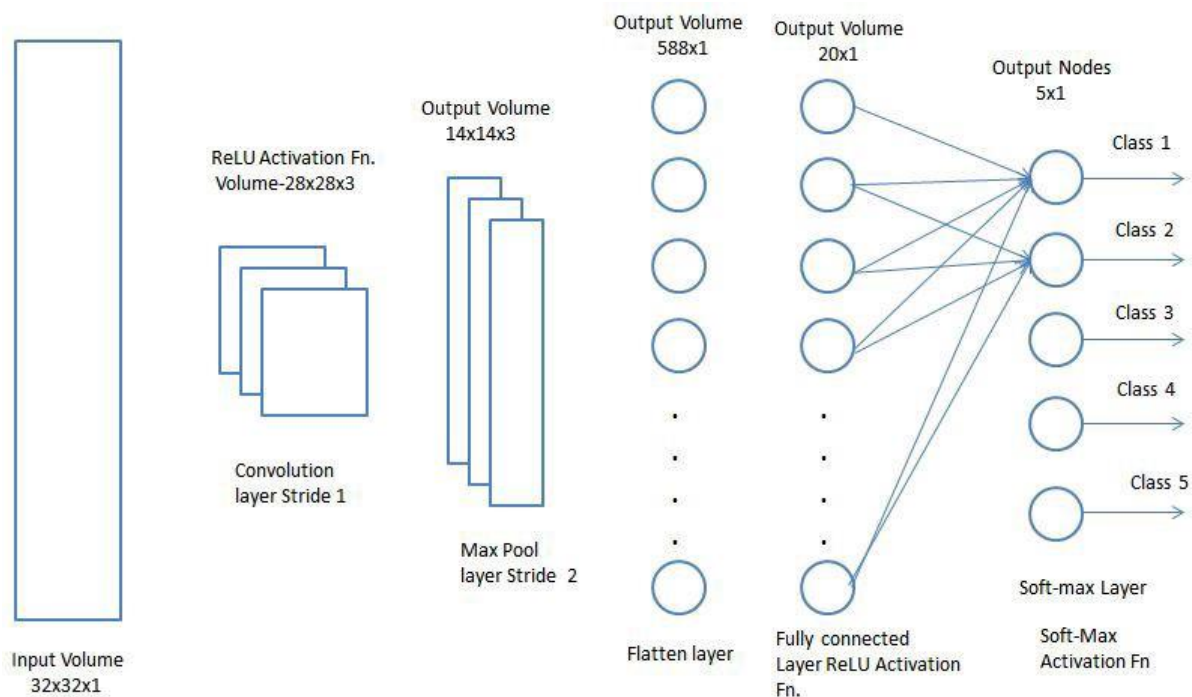
In the context of a convolutional neural network, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the “scalar product”.

Using a filter smaller than the input is intentional as it allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input. Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom.

This systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image. This capability is commonly referred to as translation invariance, e.g. the general interest in whether the feature is present rather than where it was present.

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space



**Figure 6.1.2.4 Classification — Fully Connected Layer (FC Layer)**

The above figure 6.1.2.4 represents the classification – fully connected layer (FC Layer) Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

## YOLO ALGORITHM :

Let's review the YOLO (You Only Look Once) real-time object detection algorithm, which is one of the most effective object detection algorithms that also encompasses many of the most innovative ideas coming out of the computer vision research community. Object detection is a critical capability of autonomous vehicle technology. It's an area of computer vision that's exploding and working



so much better than just a few years ago. At the end of this article, we'll see a couple of recent updates to YOLO by the original researchers of this important technique.

YOLO came on the computer vision scene with the seminal 2015 paper by Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection," and immediately got a lot of attention by fellow computer vision researchers. Here is a TED talk by University of Washington researcher Redmon in 2017 highlighting the state of the art in computer vision.

Object detection is one of the classical problems in computer vision where you work to recognize what and where — specifically what objects are inside a given image and also where they are in the image. The problem of object detection is more complex than classification, which also can recognize objects but doesn't indicate where the object is located in the image. In addition, classification doesn't work on images containing more than one object.

YOLO uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

- YOLO is extremely fast
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

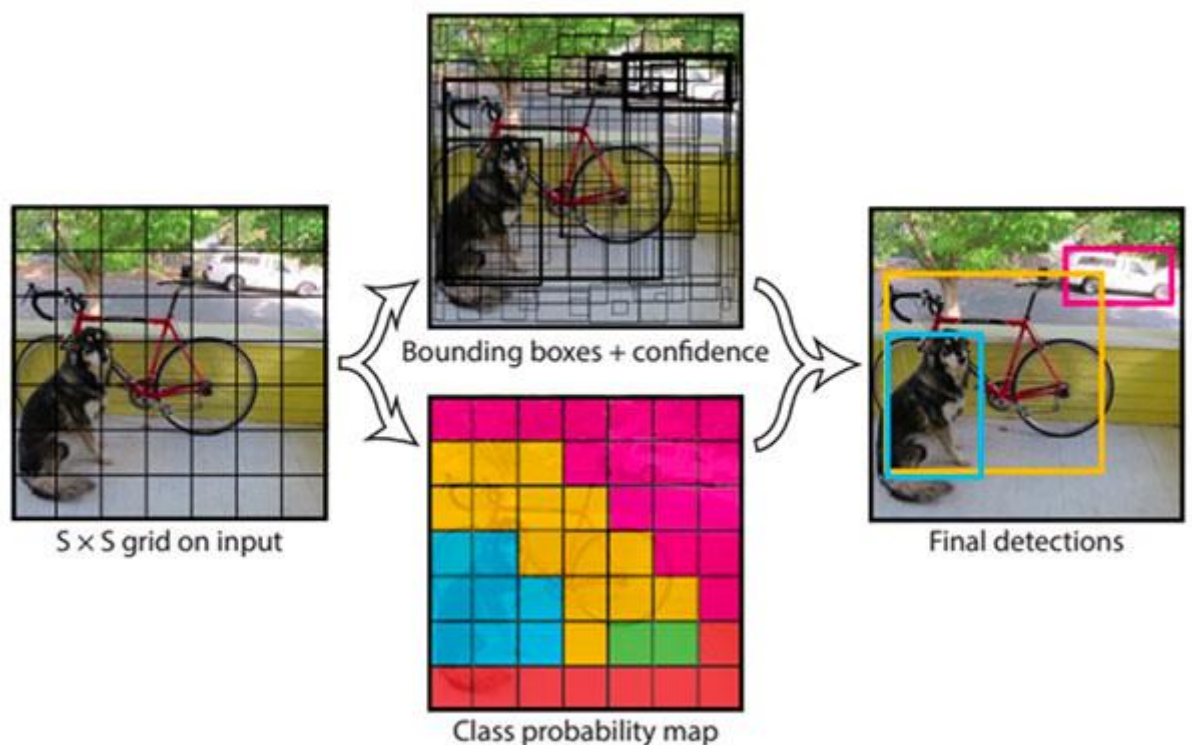
Here is an impressive video demonstration that shows YOLO's success in object detection:

Further research was conducted resulting in the December 2016 paper “YOLO9000: Better, Faster, Stronger,” by Redmon and Farhadi, both from the University of Washington, that provided a number of improvements to the YOLO detection method including the detection of over 9,000 object categories by jointly optimizing detection and classification.

Even more recently, the same researchers wrote another paper in April 2018 on their progress with evolving YOLO even further, “YOLOv3: An Incremental Improvement” with code available on a GitHub repo.

As with a lot of research work in deep learning, much of the effort is trial and error. In pursuit of YOLOv3, this effect was in force as the team tried a lot of

different ideas, but many of them didn't work out. A few of the things that stuck include: a new network for performing feature extraction consisting of 53 convolutional layers, a new detection metric, predicting an "objectness" score for each bounding box using logistic regression, and using binary cross-entropy loss for the class predictions during training. The end result is that YOLOv3 runs significantly faster than other detection methods with comparable performance. In addition, YOLO no longer struggles with small objects.

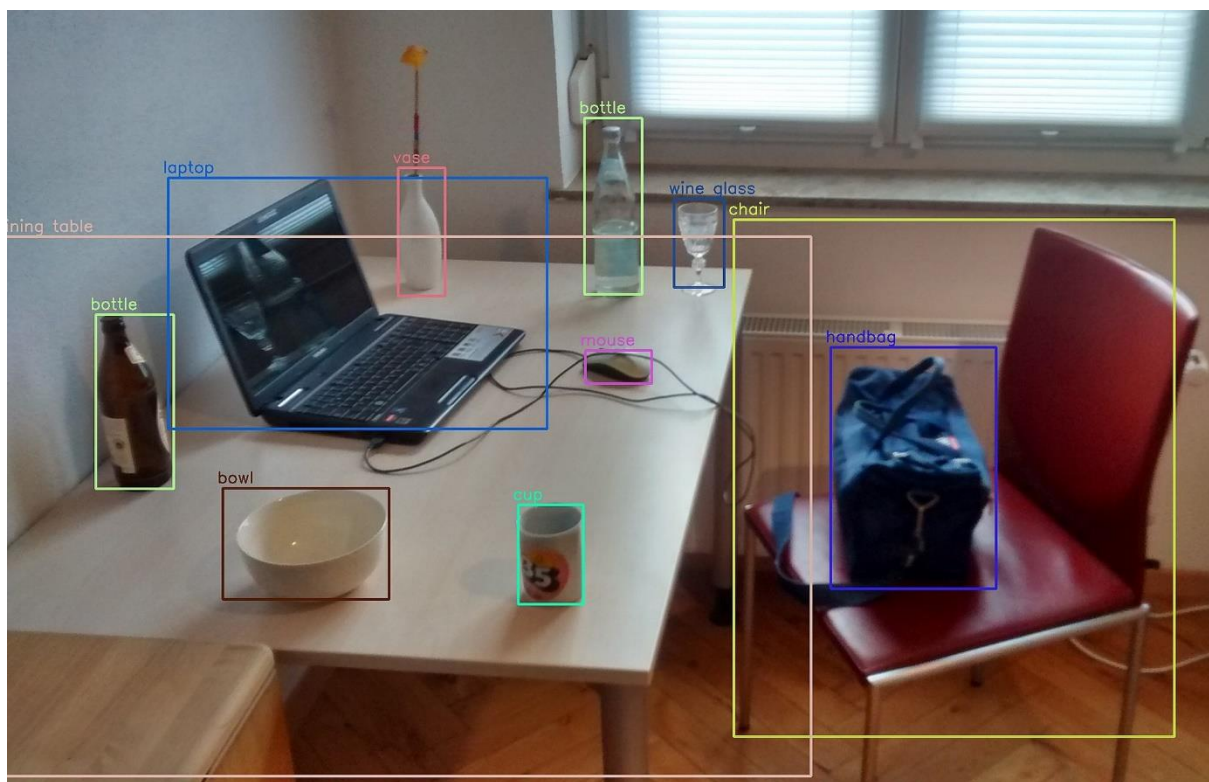


**Figure 6.1.2.5 YOLO object detection with openCV**

The above figure 6.1.2.5 represents YOLO object detection with openCV. I found it interesting that at the end of the YOLOv3 paper the authors, who are top-notch researchers actively pushing the envelope of this technology, reflect on how object detection is destined to be used.

### 6.1.3 OBJECT CLASSIFICATION:

Object Classification is a classification problem which tends to classify different objects which could be flowers, faces, fruits or any object we could imagine. Figure 6.1.3 represents the Object classification. All the detected objects from the above module will be classified under COCO dataset. COCO dataset, meaning “Common Objects In Context”, is a set of challenging, high quality datasets for computer vision, mostly state-of-the-art neural networks.



**Figure 6.1.3 Object classification**

### Darknet :

A dark net or darknet is an overlay network within the Internet that can only be accessed with specific software, configurations, or authorization,<sup>[1]</sup> and often uses a unique customized communication protocol. Two typical darknet types are social networks (usually used for file hosting with a peer-to-

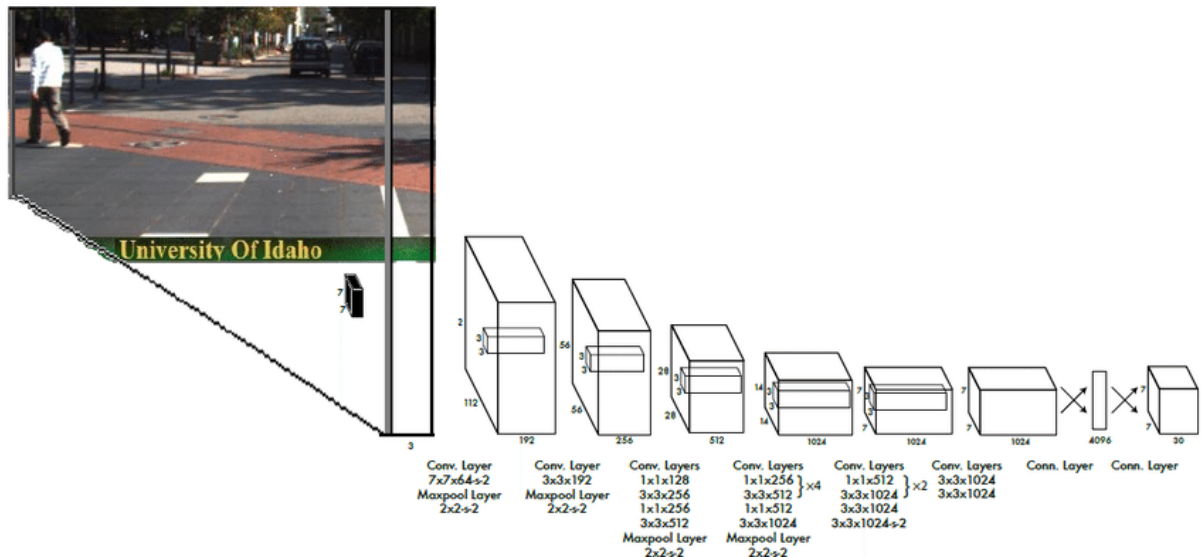
peer connection), and anonymity proxy networks such as Tor via an anonymized series of connections.

The term 'darknet' was popularised by major news outlets to associate with Tor Onion services, when the infamous drug bazaar Silk Road used it,<sup>[4]</sup> despite the terminology being unofficial. Technology such as Tor, I2P, and Freenet was intended to defend digital rights by providing security, anonymity, or censorship resistance and is used for both illegal and legitimate reasons. Anonymous communication between whistle-blowers, activists, journalists and news organisations is also facilitated by darknets through use of applications such as SecureDrop.

The term originally described computers on ARPANET that were hidden, programmed to receive messages but not respond to or acknowledge anything, thus remaining invisible, in the dark.<sup>[6]</sup> An account detailed how the first online transaction related to drugs transpired in 1971 when students of the Massachusetts Institute of Technology and Stanford University traded marijuana using ARPANET accounts in the former's Artificial Intelligence Laboratory.

Since ARPANET, the usage of dark net has expanded to include friend-to-friend networks (usually used for file sharing with a peer-to-peer connection) and privacy networks such as Tor.<sup>[8][9]</sup> The reciprocal term for a darknet is a clearnet or the surface web when referring to content indexable by search engines.

The term "darknet" is often used interchangeably with "dark web" due to the quantity of hidden services on Tor's darknet. The term is often inaccurately used interchangeably with the deep web due to Tor's history as a platform that could not be search-indexed. Mixing uses of both these terms has been described as inaccurate, with some commentators recommending the terms be used.



### Figure 6.1.3.1 Darknet convolutional neural network architecture

The above figure represents the darknet convolutional neural network architecture. All darknets require specific software installed or network configurations made to access them, such as Tor, which can be accessed via a customised browser from Vidalia (aka the Tor browser bundle), or alternatively via a proxy configured to perform the same function.

Darknets in general may be used for various reasons, such as:

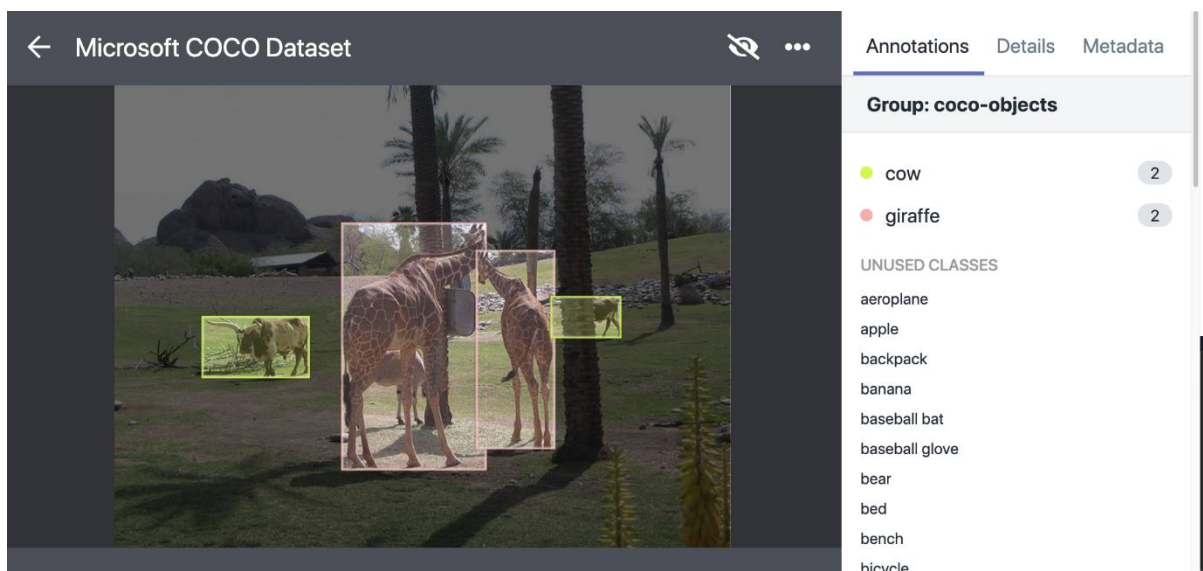
- To better protect the privacy rights of citizens from targeted and mass surveillance
- Computer crime (cracking, file corruption, etc.)
- Protecting dissidents from political reprisal
- File sharing (warez, personal files, pornography, confidential files, illegal or counterfeit software, etc.)
- Sale of restricted goods on darknet markets
- Whistleblowing and news leaks
- Purchase or sale of illicit or illegal goods or services



## COCO DATASET :

The MS COCO (Microsoft Common Objects in Context) dataset is a large-scale object detection, segmentation, key-point detection, and captioning dataset. The dataset consists of 328K images.

**Splits:** The first version of MS COCO dataset was released in 2014. It contains 164K images split into training (83K), validation (41K) and test (41K) sets. In 2015 additional test set of 81K images was released, including all the previous test images and 40K new images.



**Figure 6.1.3.2 Cow and Giraffe objects in the COCO object detection dataset**

The above figure 6.1.3.2 represents Cow and Giraffe objects in the COCO object detection dataset. Based on community feedback, in 2017 the training/validation split was changed from 83K/41K to 118K/5K. The new split uses the same images and annotations. The 2017 test set is a subset of 41K images of the 2015 test set. Additionally, the 2017 release contains a new unannotated dataset of 123K images.



**Figure 6.1.3.3 COCO Panoptic Segmentation**

- The above figure 6.1.3.3 represents the COCO Panoptic Segmentation. Semantic Segmentation - The boundary of objects are labeled with a mask and object classes are labeled with a class label



**Figure 6.1.3.4 COCO Keypoint Detection**

- The above figure 6.1.3.4 represents COCO Keypoint Detection. Keypoint Detection - Humans are labeled with key points of interest (elbow, knee, etc.)



Annotations: The dataset has annotations for

- object detection: bounding boxes and per-instance segmentation masks with 80 object categories,
- captioning: natural language descriptions of the images (see MS COCO Captions),
- keypoints detection: containing more than 200,000 images and 250,000 person instances labeled with keypoints (17 possible keypoints, such as left eye, nose, right hip, right ankle),
- stuff image segmentation – per-pixel segmentation masks with 91 stuff categories, such as grass, wall, sky (see MS COCO Stuff),
- panoptic: full scene segmentation, with 80 thing categories (such as person, bicycle, elephant) and a subset of 91 stuff categories (grass, sky, road),
- dense pose: more than 39,000 images and 56,000 person instances labeled with DensePose annotations – each labeled person is annotated with an instance id and a mapping between image pixels that belong to that person body and a template 3D model. The annotations are publicly available only for training and validation images.

#### **6.1.4 DISTANCE CALCULATION:**

The user will be holding the device through which the objects will be identified. The exact distance between the user and the object cannot be determined without the help of hardware such as sensor etc. So here we find the approximate distance between the user and the object through bounding boxes.

- If the object or the person is close to the user then the object will be displayed large in the screen where the bounding boxes also will be large.

Figure 6.1.4 represents the Object nearer to screen.

- If the object or the person is far away from the user then the object will be displayed small in the screen where the bounding box will also be small.

Figure 6.1.4.1 represents the Object away from the screen.

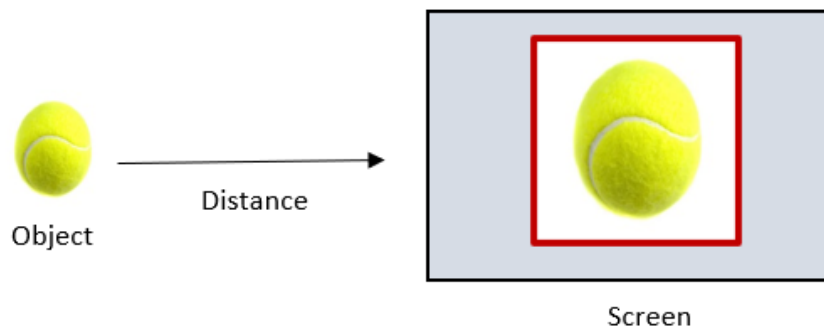
Now the distance between the object and the user will be calculated in this module. Here the distance is calculated in inches. The distance is calculated as follows,

$x = \text{boundingboxes}[0]$     $w = \text{boundingboxes}[2]$

$y = \text{boundingboxes}[1]$     $h = \text{boundingboxes}[3]$

**$\text{distance} = ((2 * 3.14 * 180) / (w.\text{item}() + h.\text{item}() * 360) * 1000 + 3) * 20$**

By this above formula the distance between the objects and the user is calculated.



**Figure 6.1.4 Object nearer to screen**



**Figure 6.1.4.1 Object away from the screen**

### 6.1.5 OUTPUT SPEECH:

In this module the speech will be the output. The obtained distance and classified objects will be converted to speech. The distance calculated as well the object classified under classes will be in the form of text. Now the obtained text will be converted to voice. Pyaudio is used for the conversion of text to speech. Figure 6.1.5 represents the output speech. To use PyAudio, first instantiate PyAudio using `pyaudio. PyAudio()` , which sets up the portaudio system. To record or play audio, open a stream on the desired device with the desired audio parameters using `pyaudio`.



**Figure 6.1.5 Output Speech**

#### **Pyaudio :**

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms. PyAudio is inspired by:

- `pyPortAudio/fastaudio`: Python bindings for PortAudio v18 API.
- `tkSnack`: cross-platform sound toolkit for Tcl/Tk and Python.
- To use PyAudio, first instantiate PyAudio using `pyaudio.PyAudio()` , which sets up the portaudio system.

This sets up a **pyaudio.Stream** to play or record audio.

- Play audio by writing audio data to the stream using **pyaudio.Stream.write()**, or read audio data from the stream using **pyaudio.Stream.read()**.
- Note that in “blocking mode”, each **pyaudio.Stream.write()** or **pyaudio.Stream.read()** blocks until all the given/requested frames have been played/recorded. Alternatively, to generate audio data on the fly or immediately process recorded audio data, use the “callback mode” outlined below.
- Use **pyaudio.Stream.stop\_stream()** to pause playing/recording, and **pyaudio.Stream.close()** to terminate the stream.
- Finally, terminate the portaudio session using **pyaudio.PyAudio.terminate()**

In callback mode, PyAudio will call a specified callback function whenever it needs new audio data (to play) and/or when there is new (recorded) audio data available. Note that PyAudio calls the callback function in a separate thread.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **7.1 CONCLUSION**

This paper proposes a computer vision-based system that assists blind persons in object detection as well as tries to answers the questions raised in the Introduction section. The proposed system is tested for different conditions. It is evident from the observations that the application performs fairly well under different circumstances. The application is also fairly accessible to blind users as it allows the blind user to open the application by inserting earphone jack into the mobile phones which is missing in many other applications. In the current state of application, this application can be used for navigation but has a few limitations. There are still many areas that are a concern for improvement.

#### **7.2 FUTURE ENHANCEMENT**

- As of now this application can detect around 100 objects and about 20 classes of objects, this could be improved to encompass more classes of objects
- The application can be improved by merging it with Internet of Devices technology devices to account for better object detection
- With enhancements in deep learning and YOLO algorithm, the application will see better performance in the future
- The application is limited to android phones and needs redesigning be able to run on cross platform in the future.

## APPENDIX 1

### SAMPLE CODING

#### Preprocess

```
import torch

import torch.nn as nn

import torch.nn.functional as F
from torch.autograd import Variable
import numpy as np
import cv2
import matplotlib.pyplot as plt
from util import count_parameters as count
from util import convert2cpu as cpu
from PIL import Image, ImageDraw

def letterbox_image(img, inp_dim):

    '''resize image with unchanged aspect ratio using padding'''
    img_w, img_h = img.shape[1], img.shape[0]
    w, h = inp_dim
    new_w = int(img_w * min(w/img_w, h/img_h))
    new_h = int(img_h * min(w/img_w, h/img_h))
    resized_image = cv2.resize(img, (new_w,new_h), interpolation =
cv2.INTER_CUBIC)

    canvas = np.full((inp_dim[1], inp_dim[0], 3), 128)

    canvas[(h-new_h)//2:(h-new_h)//2 + new_h,(w-new_w)//2:(w-new_w)//2
+ new_w, :] = resized_image

    return canvas
```

```

def prep_image(img, inp_dim):
    """
    Prepare image for inputting to the neural network.

    Returns a Variable
    """

    orig_im = cv2.imread(img)
    dim = orig_im.shape[1], orig_im.shape[0]
    img = (letterbox_image(orig_im, (inp_dim, inp_dim)))
    img_ = img[:, :, :-1].transpose((2, 0, 1)).copy()
    img_ = torch.from_numpy(img_).float().div(255.0).unsqueeze(0)
    return img_, orig_im, dim

def prep_image_pil(img, network_dim):
    orig_im = Image.open(img)
    img = orig_im.convert('RGB')
    dim = img.size
    img = img.resize(network_dim)
    img =
    torch.ByteTensor(torch.ByteStorage.from_buffer(img.tobytes()))
    img = img.view(*network_dim,
3).transpose(0, 1).transpose(0, 2).contiguous()
    img = img.view(1, 3, *network_dim)
    img = img.float().div(255.0)
    return (img, orig_im, dim)

def inp_to_image(inp):
    inp = inp.cpu().squeeze()
    inp = inp*255
    try:
        inp = inp.data.numpy()
    except RuntimeError:
        inp = inp.numpy()
    inp = inp.transpose(1, 2, 0)

    inp = inp[:, :, :-1]

```

```
return inp
```

### **Darknet :**

```
import torch

import torch.nn as nn

import torch.nn.functional as F

from torch.autograd import Variable

import numpy as np

import cv2

import matplotlib.pyplot as plt

from util import count_parameters as count

from util import convert2cpu as cpu

from util import predict_transform

class test_net(nn.Module):

    def __init__(self, num_layers, input_size):

        super(test_net, self).__init__()

        self.num_layers= num_layers

        self.linear_1 = nn.Linear(input_size, 5)

        self.middle = nn.ModuleList([nn.Linear(5,5) for x in range(num_layers)])

        self.output = nn.Linear(5,2)

        def forward(self, x):

            x = x.view(-1)

            fwd = nn.Sequential(self.linear_1, *self.middle, self.output)

            return fwd(x)

    def get_test_input():

        img = cv2.imread("dog-cycle-car.png")
```



```

img = cv2.resize(img, (416,416))
img_ = img[:,::-1].transpose((2,0,1))
img_ = img_[np.newaxis,:,:]/255.0
img_ = torch.from_numpy(img_).float()
img_ = Variable(img_)
return img_

```

```

def parse_cfg(cfgfile):
    file = open(cfgfile, 'r')
    lines = file.read().split('\n')    #store the lines in a list
    lines = [x for x in lines if len(x) > 0] #get rid of the empty lines
    lines = [x for x in lines if x[0] != '#']
    lines = [x.rstrip().lstrip() for x in lines]
    block = {}
    blocks = []

    for line in lines:
        if line[0] == "[":            #This marks the start of a new block
            if len(block) != 0:
                blocks.append(block)
                block = {}
            block["type"] = line[1:-1].rstrip()
        else:
            key,value = line.split("=")
            block[key.rstrip()] = value.lstrip()
    blocks.append(block)

```

```

    return blocks

# print('\n\n'.join([repr(x) for x in blocks]))

import pickle as pkl

class MaxPoolStride1(nn.Module):

    def __init__(self, kernel_size):

        super(MaxPoolStride1, self).__init__()

        self.kernel_size = kernel_size

        self.pad = kernel_size - 1

    def forward(self, x):

        padded_x = F.pad(x, (0,self.pad,0,self.pad), mode="replicate")

        pooled_x = nn.MaxPool2d(self.kernel_size, self.pad)(padded_x)

        return pooled_x

class EmptyLayer(nn.Module):

    def __init__(self):

        super(EmptyLayer, self).__init__()

class DetectionLayer(nn.Module):

    def __init__(self, anchors):

        super(DetectionLayer, self).__init__()

        self.anchors = anchors

    def forward(self, x, inp_dim, num_classes, confidence):

        x = x.data

        global CUDA

        prediction = x

```

```
prediction = predict_transform(prediction, inp_dim, self.anchors,
num_classes, confidence, CUDA)
```

```
return prediction
```

```
class Upsample(nn.Module):
```

```
def __init__(self, stride=2):
```

```
    super(Upsample, self).__init__()
```

```
    self.stride = stride
```

```
def forward(self, x):
```

```
    stride = self.stride
```

```
    assert(x.data.dim() == 4)
```

```
    B = x.data.size(0)
```

```
    C = x.data.size(1)
```

```
    H = x.data.size(2)
```

```
    W = x.data.size(3)
```

```
    ws = stride
```

```
    hs = stride
```

```
    x = x.view(B, C, H, 1, W, 1).expand(B, C, H, stride, W,
stride).contiguous().view(B, C, H*stride, W*stride)
```

```
    return x
```

### **Bounding Box :**

```
import torch
```

```
import random
```

```
import numpy as np
```

```
import cv2
```

```
def confidence_filter(result, confidence):
```

```
    conf_mask = (result[:, :, 4] > confidence).float().unsqueeze(2)
```

```

    result = result*conf_mask
    return result

def confidence_filter_cls(result, confidence):
    max_scores = torch.max(result[:, :, 5:25], 2)[0]
    res = torch.cat((result, max_scores), 2)
    print(res.shape)
    cond_1 = (res[:, :, 4] > confidence).float()
    cond_2 = (res[:, :, 25] > 0.995).float()
    conf = cond_1 + cond_2
    conf = torch.clamp(conf, 0.0, 1.0)
    conf = conf.unsqueeze(2)
    result = result*conf
    return result

def get_abs_coord(box):
    box[2], box[3] = abs(box[2]), abs(box[3])
    x1 = (box[0] - box[2]/2) - 1
    y1 = (box[1] - box[3]/2) - 1
    x2 = (box[0] + box[2]/2) - 1
    y2 = (box[1] + box[3]/2) - 1
    return x1, y1, x2, y2

def sanity_fix(box):
    if (box[0] > box[2]):
        box[0], box[2] = box[2], box[0]
    if (box[1] > box[3]):
        box[1], box[3] = box[3], box[1]
    return box

```

```

def bbox_iou(box1, box2):
    """
    Returns the IoU of two bounding boxes
    """
    #Get the coordinates of bounding boxes
    b1_x1, b1_y1, b1_x2, b1_y2 = box1[:,0], box1[:,1], box1[:,2], box1[:,3]
    b2_x1, b2_y1, b2_x2, b2_y2 = box2[:,0], box2[:,1], box2[:,2], box2[:,3]
    #get the corrdinates of the intersection rectangle
    inter_rect_x1 = torch.max(b1_x1, b2_x1)
    inter_rect_y1 = torch.max(b1_y1, b2_y1)
    inter_rect_x2 = torch.min(b1_x2, b2_x2)
    inter_rect_y2 = torch.min(b1_y2, b2_y2)

    #Intersection area
    if torch.cuda.is_available():
        inter_area = torch.max(inter_rect_x2 - inter_rect_x1 +
1,torch.zeros(inter_rect_x2.shape).cuda())*torch.max(inter_rect_y2 -
inter_rect_y1 + 1, torch.zeros(inter_rect_x2.shape).cuda())
    else:
        inter_area = torch.max(inter_rect_x2 - inter_rect_x1 +
1,torch.zeros(inter_rect_x2.shape))*torch.max(inter_rect_y2 - inter_rect_y1 + 1,
torch.zeros(inter_rect_x2.shape))

    #Union Area
    b1_area = (b1_x2 - b1_x1 + 1)*(b1_y2 - b1_y1 + 1)
    b2_area = (b2_x2 - b2_x1 + 1)*(b2_y2 - b2_y1 + 1)
    iou = inter_area / (b1_area + b2_area - inter_area)

```

```

    return iou

def pred_corner_coord(prediction):
    #Get indices of non-zero confidence bboxes
    ind_nz = torch.nonzero(prediction[:, :, 4]).transpose(0, 1).contiguous()

    box = prediction[ind_nz[0], ind_nz[1]]
    box_a = box.new(box.shape)
    box_a[:, 0] = (box[:, 0] - box[:, 2]) / 2
    box_a[:, 1] = (box[:, 1] - box[:, 3]) / 2
    box_a[:, 2] = (box[:, 0] + box[:, 2]) / 2
    box_a[:, 3] = (box[:, 1] + box[:, 3]) / 2
    box[:, :4] = box_a[:, :4]
    prediction[ind_nz[0], ind_nz[1]] = box
    return prediction

def write(x, batches, results, colors, classes):
    c1 = tuple(x[1:3].int())
    c2 = tuple(x[3:5].int())
    img = results[int(x[0])]
    cls = int(x[-1])
    label = "{0}".format(classes[cls])
    color = random.choice(colors)
    cv2.rectangle(img, c1, c2, color, 1)
    t_size = cv2.getTextSize(label, cv2.FONT_HERSHEY_PLAIN, 1, 1)[0]
    c2 = c1[0] + t_size[0] + 3, c1[1] + t_size[1] + 4
    cv2.rectangle(img, c1, c2, color, -1)

    cv2.putText(img, label, (c1[0], c1[1] + t_size[1] + 4),
cv2.FONT_HERSHEY_PLAIN, 1, [225, 255, 255], 1);

```

```
    return img
```

### **Object detection :**

```
import torch,cv2,random,os,time
import torch.nn as nn
from torch.autograd import Variable
import numpy as np
import pickle as pkl
import argparse
import threading, queue
from torch.multiprocessing import Pool, Process, set_start_method
from util import write_results, load_classes
from preprocess import letterbox_image
from darknet import Darknet
from imutils.video import WebcamVideoStream,FPS
from gtts import gTTS
import os
import time
# from camera import write
import win32com.client as wincl    ##### Python's Text-to-speech (tts) engine
for windows, multiprocessing
speak = wincl.Dispatch("SAPI.SpVoice")  ##### This initiates the tts engine
torch.multiprocessing.set_start_method('spawn', force=True)
## Setting up torch for gpu utilization
if torch.cuda.is_available():
    torch.backends.cudnn.enabled = True
    torch.backends.cudnn.benchmark = True
    torch.backends.cudnn.deterministic = True
    torch.set_default_tensor_type('torch.cuda.FloatTensor')
```

```

def prep_image(img, inp_dim)
    orig_im = img
    dim = orig_im.shape[1], orig_im.shape[0]
    img = (letterbox_image(orig_im, (inp_dim, inp_dim)))
    img_ = img[:, :, ::-1].transpose((2, 0, 1)).copy()
    img_ = torch.from_numpy(img_).float().div(255.0).unsqueeze(0)
    return img_, orig_im, dim

labels = {}
b_boxes = {}
def write(bboxes, img, classes, colors):
    class_idx = bboxes
    bboxes = bboxes[1:5]
    bboxes = bboxes.cpu().data.numpy()
    bboxes = bboxes.astype(int)
    b_boxes.update({"bbox":bboxes.tolist()})
    # bboxes = bboxes + [150,100,200,200] # personal choice you can modify
    this to get distance as accurate as possible
    bboxes = torch.from_numpy(bboxes)
    cls = int(class_idx[-1])
    label = "{0}".format(classes[cls])
    labels.update({"Current Object":label})
    color = random.choice(colors)

    ## Put text configuration on frame
    text_str = '%s' % (label)
    font_face = cv2.FONT_HERSHEY_DUPLEX
    font_scale = 0.6
    font_thickness = 1

```



```

    text_w, text_h = cv2.getTextSize(text_str, font_face, font_scale,
font_thickness)[0]

    text_pt = (bboxes[0], bboxes[1] - 3)

    text_color = [255, 255, 255]

## Distance Meaasurement for each bounding box

    x, y, w, h = bboxes[0], bboxes[1], bboxes[2], bboxes[3]

    ## item() is used to retrieve the value from the tensor

    distance = ((2 * 3.14 * 180) / (w.item()+ h.item() * 360) * 1000 + 3) * 20 ###
Distance measuring in Inch

    feedback = ("{} ".format(labels["Current Object"])+ " " +"is"+" at { }
".format(round(distance))+"Inches")

    # # speak.Speak(feedback)    # If you are running this on linux based OS
kindly use espeak. Using this speaking library in winodws will add unnecessary
latency

    print(feedback)

    mytext = feedback

    language = 'en'

    myobj = gTTS(text=mytext, lang=language, slow=True)

    time.sleep(5)

    myobj.save("text.mp3")

    os.system("start text.mp3")

    cv2.putText(img, str("{:.2f} Inches".format(distance)), (text_w+x,y),
cv2.FONT_HERSHEY_DUPLEX, font_scale, (0,255,0), font_thickness,
cv2.LINE_AA)

    cv2.rectangle(img, (bboxes[0],bboxes[1]),(bboxes[2] + text_w -
30,bboxes[3]), color, 2)

    cv2.putText(img, text_str, text_pt, font_face, font_scale, text_color,
font_thickness, cv2.LINE_AA)

return img

class ObjectDetection:

    def __init__(self, id):

```

```

# self.cap = cv2.VideoCapture(id)
self.cap = WebcamVideoStream(src = id).start()
self.cfgfile = "cfg/yolov3.cfg"
# self.cfgfile = 'cfg/yolov3-tiny.cfg'
self.weightsfile = "yolov3.weights"
# self.weightsfile = 'yolov3-tiny.weights'
self.confidence = float(0.6)
self.nms_thesh = float(0.8)
self.num_classes = 80
self.classes = load_classes('data/coco.names')
self.colors = pkl.load(open("pallette", "rb"))
self.model = Darknet(self.cfgfile)
self.CUDA = torch.cuda.is_available()
self.model.load_weights(self.weightsfile)
self.model.net_info["height"] = 160
self.inp_dim = int(self.model.net_info["height"])
self.width = 1280 #640#1280
self.height = 720 #360#720
print("Loading network.....")
if self.CUDA:
    self.model.cuda()
print("Network successfully loaded")
assert self.inp_dim % 32 == 0
assert self.inp_dim > 32
self.model.eval()

def main(self):
    q = queue.Queue()
    while True:

```

```

def frame_render(queue_from_cam):
    frame = self.cap.read() # If you capture stream using opencv
(cv2.VideoCapture()) the use the following line
    # ret, frame = self.cap.read()
    frame = cv2.resize(frame,(self.width, self.height))
    queue_from_cam.put(frame)
cam = threading.Thread(target=frame_render, args=(q,))
cam.start()
cam.join()
frame = q.get()
q.task_done()
fps = FPS().start()
try:
    img, orig_im, dim = prep_image(frame, self.inp_dim)
    im_dim = torch.FloatTensor(dim).repeat(1,2)
    if self.CUDA: ##### If you have a gpu properly
installed then it will run on the gpu
        im_dim = im_dim.cuda()
        img = img.cuda()
    # with torch.no_grad(): ##### Set the model in the evaluation
mode
    output = self.model(Variable(img), self.CUDA)
    output = write_results(output, self.confidence, self.num_classes, nms
= True, nms_conf = self.nms_thesh) ##### Localize the objects in a frame
    output = output.type(torch.half)

    if list(output.size()) == [1,86]:
        print(output.size())
        pass
    else:

```

```

        output[:,1:5] = torch.clamp(output[:,1:5], 0.0,
float(self.inp_dim))/self.inp_dim

        #         im_dim = im_dim.repeat(output.size(0), 1)
        output[:,[1,3]] *= frame.shape[1]
        output[:,[2,4]] *= frame.shape[0]

        list(map(lambda boxes: write(boxes, frame, self.classes,
self.colors),output))

    except:
        pass

    fps.update()
    fps.stop()
    print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
    print("[INFO] approx. FPS: {:.1f}".format(fps.fps()))
    cv2.imshow("Object Detection Window", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    continue

    torch.cuda.empty_cache()

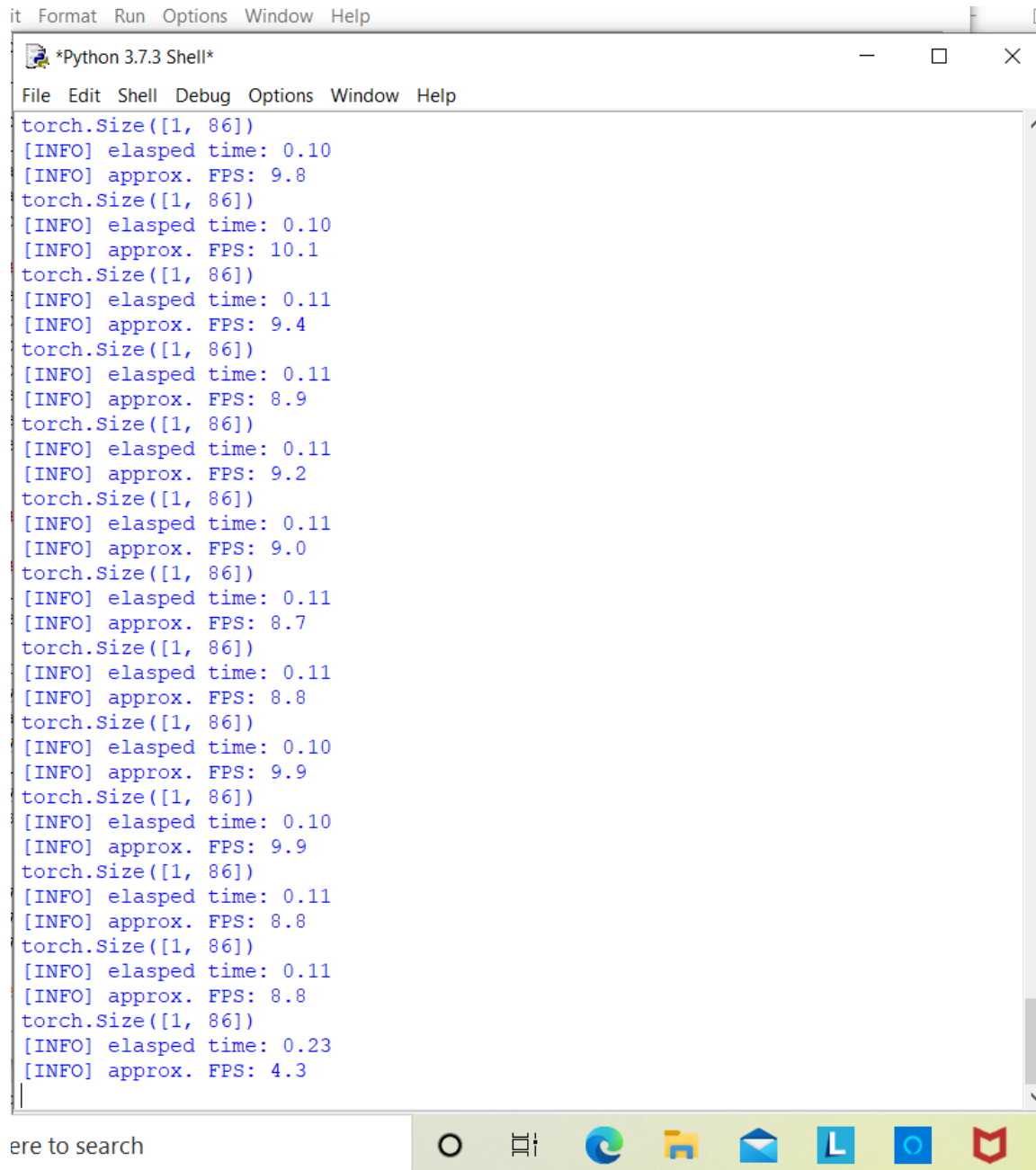
if __name__ == "__main__":
    id = 0
    ObjectDetection(id).main()

```

## APPENDIX 2

### SCREENSHOTS

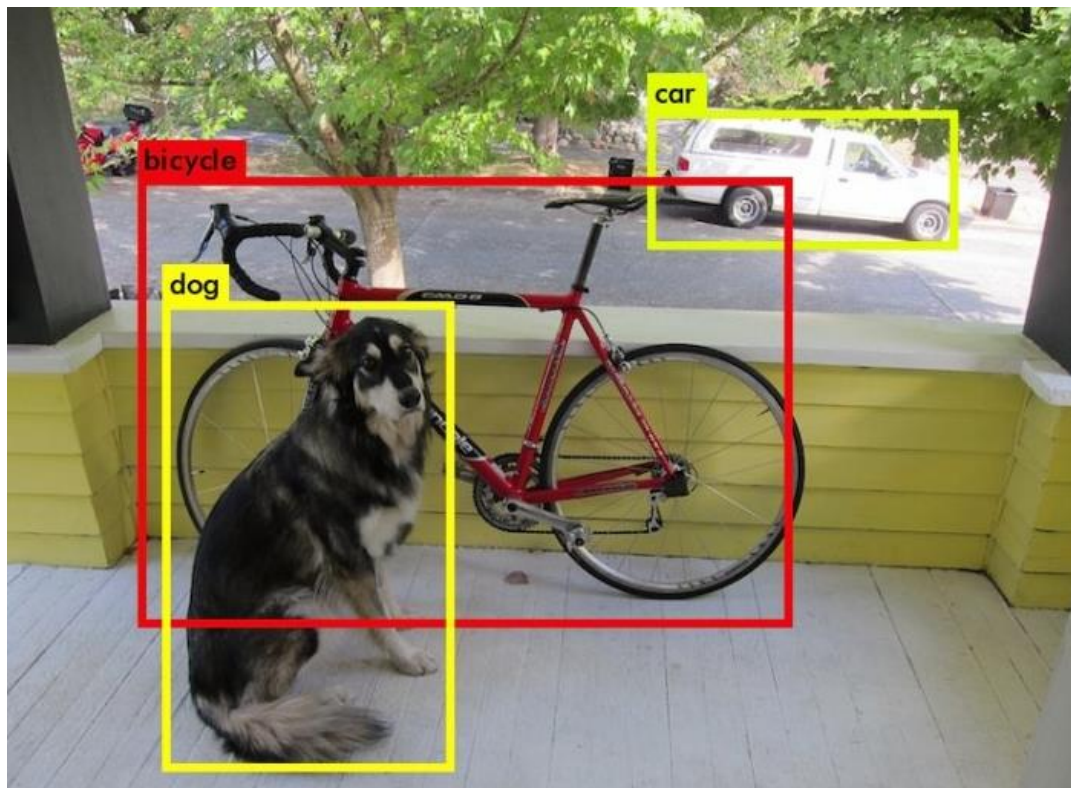
#### OBJECT DETECTION TERMINAL :



```
it Format Run Options Window Help
*Python 3.7.3 Shell*
File Edit Shell Debug Options Window Help
torch.Size([1, 86])
[INFO] elapsed time: 0.10
[INFO] approx. FPS: 9.8
torch.Size([1, 86])
[INFO] elapsed time: 0.10
[INFO] approx. FPS: 10.1
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 9.4
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 8.9
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 9.2
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 9.0
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 8.7
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 8.8
torch.Size([1, 86])
[INFO] elapsed time: 0.10
[INFO] approx. FPS: 9.9
torch.Size([1, 86])
[INFO] elapsed time: 0.10
[INFO] approx. FPS: 9.9
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 8.8
torch.Size([1, 86])
[INFO] elapsed time: 0.11
[INFO] approx. FPS: 8.8
torch.Size([1, 86])
[INFO] elapsed time: 0.23
[INFO] approx. FPS: 4.3
```

```
*Python 3.7.3 Shell*
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\MALATHI\Desktop\Object-Detection-and-Distance-Measurement-master\Object-Detection-and-Distance-Measurement-master\object_detection.py
Loading network.....
Network successfully loaded
person is at 22 Inches
[INFO] elapsed time: 7.31
[INFO] approx. FPS: 0.1
person is at 21 Inches
person is at 21 Inches
[INFO] elapsed time: 11.11
[INFO] approx. FPS: 0.1
person is at 21 Inches
[INFO] elapsed time: 5.70
[INFO] approx. FPS: 0.2
person is at 22 Inches
[INFO] elapsed time: 5.83
[INFO] approx. FPS: 0.2
person is at 22 Inches
[INFO] elapsed time: 5.78
[INFO] approx. FPS: 0.2
person is at 22 Inches
|
```

## OBJECT DETECTION OUTPUT :



## REFERENCES

- [1] “Wearable technology,” Nov 2019. [Online]. Available: [https://en.wikipedia.org/wiki/Wearable\\_technology](https://en.wikipedia.org/wiki/Wearable_technology)
- [2] J. Hui, “What do we learn from single shot object detectors (ssd, yolov3), fpn & focal loss (retinanet)?” Medium [Online]. Available: [Accessed: vol. 28, September 2019. [Online]. Available: [https://medium.com/@jonathan\\_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d](https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d)
- [3] C.-Y. Cao, J.-C. Zheng, Y.-Q. Huang, J. Liu, and C.-F. Yang, “Investigation of a Promoted You Only Look Once Algorithm and Its Application in Traffic Flow Monitoring,” Applied Sciences, vol. 9, no. 17, p. 3619, 2019.
- [4] Z. Zhang, T. He, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of Freebies for Training Object Detection Neural Networks,” arXiv:1902.04103 [cs], 2019.

- [5] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3." IEEE, 2019, pp. 1–6.
- [6] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garc'ia, and D. Scaramuzza, "Eventbased vision meets deep learning on steering prediction for self-driving cars," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2018, pp. 5419–5427.
- [7] A. Wong, M. J. Shafiee, F. Li, and B. Chwyl, "Tiny SSD: A Tiny Single-shot Detection Deep Convolutional Neural Network for Real-time Embedded Object Detection," arXiv:1802.06488 [cs], 2018.
- [8] K.-J. Kim, P.-K. Kim, Y.-S. Chung, and D.-H. Choi, "Performance Enhancement of YOLOv3 by Adding Prediction Layers with Spatial Pyramid Pooling for Vehicle Detection," 2018, pp. 1–6.
- [9] P. Maolanon and K. Sukvichai, "Development of a Wearable Household Objects Finder and Localizer Device Using CNNs on Raspberry Pi 3," 2018, pp. 25–28.
- [10] M. B. Jensen, K. Nasrollahi, and T. B. Moeslund, "Evaluating State-of-the-Art Object Detector on Challenging Traffic Light Data," 2017, pp. 882–888.
- [11] M. Rajesh, B. K. Rajan, A. Roy, K. A. Thomas, A. Thomas, T. B. Tharakan, and C. Dinesh, "Text recognition and face detection aid for visually impaired person using Raspberry PI," in 2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT). Kollam, India: IEEE, Apr. 2017, pp. 1–5. [Online]. Available: <http://ieeexplore.ieee.org/document/8074355/>
- [12] H.-C. Wang, R. K. Katzschnmann, S. Teng, B. Araki, L. Giarre, and D. Rus, "Enabling independent navigation for visually impaired people through a wearable vision-based feedback system," in 2017 IEEE International Conference on Robotics and Automation (ICRA). Singapore, Singapore: IEEE, May 2017, pp.
- [13] Afp, "The number of blind people in the world is set to triple by 2050." [Online]. Available: <https://www.thejournal.ie/blindness-triple-2050-3527437-Aug2017/> 15
- [14] Z. Liu, Y. Luo, J. Cordero, N. Zhao, and Y. Shen, "Finger-eye: A wearable text reading assistive system for the blind and visually impaired," in 2016 IEEE



International Conference on Real-time Computing and Robotics (RCAR). Angkor Wat, Cambodia: IEEE, Jun. 2016, pp. 123–128. [Online]. Available: <http://ieeexplore.ieee.org/document/7784012/>

[15] S. Deshpande and R. Shriram, “Real time text detection and recognition on hand held objects to assist blind people,” in 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT). Pune, India: IEEE, Sep. 2016, pp. 1020–1024. [Online]. Available: <http://ieeexplore.ieee.org/document/7877741/>

[16] G. Feng and R. Buyya, “Maximum revenue-oriented resource allocation in cloud,” IJGUC, vol. 7, no. 1, pp. 12–21, 2016.

[17] R. Kune, P. Konugurthi, A. Agarwal, C. R. Rao, and R. Buyya, “The anatomy of big data computing,” *Softw., Pract. Exper.*, vol. 46, no. 1, pp. 79–105, 2016. 16

[18] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” *arXiv:1612.08242 [cs]*, 2016.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *arXiv:1506.02640 [cs]*, 2015.

[20] D. G. Gomes, R. N. Calheiros, and R. Tolosana-Calasan, “Introduction to the special issue on cloud computing: Recent developments and challenging issues,” *Computers & Electrical Engineering*, vol. 42, pp. 31–32, 2015.