

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering
Degree in Computer Science and Engineering

By

NAME: YERRAGOPU VENKATA SAI
(Reg. No.38110669)

NAME: SURYA CHAITANYA JAKKA
(Reg. No.38110581)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SCHOOL
OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY JEPPIAAR
NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU
MAY 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119
www.sathyabamauniversity.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **YERRAGOPU VENKATA SAI (3811669), SURYA CHAITANYA JAKKA(38110581)** who carried out the project entitled “**BLIND ASSISTANCE SYSTEM USING TENSORFLOW**” under my supervision from August 2021 to November 2021.

Internal Guide

Name: Dr. A. JESUDOSS M.E., Ph.D.,

Head of the Department

Name: Dr. S. VIGNESHWARI M.E., Ph.D.,

Name: Dr. L. LAKSHMANAN M.E., Ph.D.,

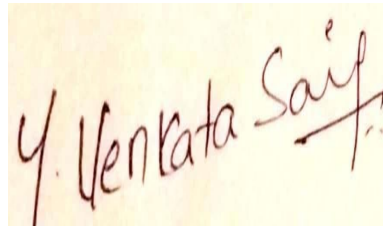
Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **YERRAGOPU VENKATA SAI** hereby declare that the Project Report entitled **BLIND ASSISTANCE SYSTEM USING TENSORFLOW** done by me under the guidance of **Dr. A. Jesudoss M.E., Ph.D.,** (Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

A handwritten signature in brown ink on a light yellow background. The signature reads "Y. Venkata Sai" with a stylized flourish at the end.

DATE: 10-04-2022

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

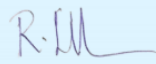
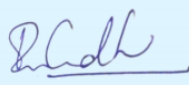
I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D., and Dr.L.Lakshmanan M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. A. Jesudoss M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE

	SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE UNIVERSITY) Accredited with Grade 'A' By NAAC 12B Status by UGC Approved By AICTE www.sathyabama.ac.in School of Computing Department of Information Technology	 
Certificate of Presentation This is to certify that YERRAGOPU VENKATA SAI of <u>Sathyabama Institute of Science and Technology</u> has presented a paper entitled <u>Blind Assistance System Using TensorFlow</u> in the AICTE sponsored International Conference on Artificial Intelligence and Machine Learning (IAIM-2022), held from 27th January 2022 - 29th January 2022.		
 Dr. R. SUBHASHINI HOD [Information Technology]	 Dr. T. SASIKALA Dean [School of Computing]	 Dr. T. SASIPRABA Vice Chancellor

ABSTRACT

There is a ton of classified and earnest data trade here today which is converted into a desired form with the convolutional neural network to make the blind feel things surrounding him. Efficient Implementation of this Model depends upon the compatibility with python and library installation hurdles. The proposed Blind Assistance ideas were developed to help those visually impaired people around the world. The proposed system has two levels. It works on the SSD algorithm and TensorFlow, in which it recognizes the object, not only the recognition but also localization of an object. It also provides the distance between the person and the object. The Laptop Based Server will be using a pre-trained SSD detection model trained on COCO DATASETS. It will then test and the output class will get detected with an accuracy metrics. In this manner, we have proposed a framework that utilizes AI, which makes the framework more straightforward to use for individuals with visual impedances. After testing with the help of voice modules the class of the object will be converted into a default voice notes which will then be sent to the blind victims for their assistance. Main key aspect of the proposed system is identifying or naming the object detected, prints accurate distance between the user and the objects and voice over using Audio Commands.

ABBREVIATIONS

SSD – Single Shot Detector
CNN – Convolutional Neural Network
CSS– Cascading Style Sheets

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	6
	LIST OF ABBREVIATIONS	6
1.	INTRODUCTION	8
1.1	PROJECT STATEMENT	8
1.2	PROJECT JUSTIFICATION	9
2.	LITERATURE SURVEY	9
3.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	21
4.	EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED	21
4.1	LANGUAGE USED	21
4.2	METHODS USED	21
4.3	PYTHON LIBRARIES USED	22
4.4	VOICE GENERATING LIBRARIES	22
4.5	ALGORITHMS USED	22
4.6	FLOW CHART	26
5.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	27
5.1	PERFORMANCE ANALYSIS	27
5.2	LIMITATIONS	29
6.	SUMMARY AND CONCLUSIONS	30
	REFERENCES	31
	APPENDIX	33
A.	SCREENSHOTS	33

CHAPTER 1

INTRODUCTION

Computer vision drew attention by swaying as a data- reliant stratified feature in extraction methods. It is a dream-based vision for the visually impaired. The framework is intended to permit blind individual admittance to programming, which likewise sends time spans to versatile PCs. Working on the capacity of individuals with visual disabilities to make independent savvy glasses can recognize things that are viable with the climate and to know them progressively. People suffering from such defective things, this project helps them to some extent. We need to acquire some updated technologies to finish this project with perfect output. Many projects just take input and display outcome but we need to make a chance of giving voice commands to the system. With the reference of “Object tracking and counting in a zone using YOLOv4, Deep SORT, and TensorFlow”. this paper helped me in many aspects of few steps in my project. In the above paper, they used technologies like YOLOv4, Deep SORT, and tensor flow. Tensor flow helped me to give a nice output to my project. The above-mentioned project mainly focused on detecting multiple peoples and objects like cars bicycles etc. Counts the objects detected, it also captures a video similar to our project. So, in this project only the detection takes place. In our project, we are going to implement new techniques using CNN, SSD, and tensor flow and the addition of voice also takes place. In this project, the user can experience the categorization of images. Detection of images, Localization of images, Voice given by the system single shot detector we come across many phases to detect the object like Anchor box, zoom level, depth level. Anchor box is nothing but a grid like box that surrounds the detected object. Zoom level helps the anchor box in size shape and few more aspects. Depth estimation is used to detect the distance between the two objects accurately

1.1 PROBLEM STATEMENT

It is observed that the performance of the object detection degrades drastically under the course of action. The results are also compared with existing blind assistance approaches to evaluate the accuracy drop. In use assistance systems for blind only tells the object name in front of the lens and nothing more. Main concern with the existential models is they don't tell the distance between the object and the person, objectifying the problem we have made it possible with neural networks to calculate the distance and will warn the user.

We propose a primary state-of-the-art performance using the solution in terms of using COCO datasets which will identify wide range of objects and residues the distance between object and the end user.

1.2 PROJECT JUSTIFICATION

The necessity of this protrude is extracting the distance between identified object and the users to residue the complexity of calculating distance. Integrated Machine Learning System and convolutional networks which allows the Blind Victims to identify and classify Real Time Based Common day-to-day Objects with the help of TensorFlow APIs and generate voice feedbacks using python modules. It also calculates distance which produces warnings whether the user is very close or far away from the object. The same system can be used for Obstacle Detection Mechanism.

CHAPTER 2

LITERATURE SURVEY

2.1 Object Detection System based on SSD Algorithm

Abstract:

This SSD (Single Shot Multi Box Detector) is an object detection algorithm based on deep learning. As one of the most mainstream detection algorithms, it can greatly improve the detection speed and ensure the detection accuracy. In this paper, the Batch Norm operation is added to the network in order to improve the generalization of the network and speed up network training. The object counting function is added to the image recognition. This paper uses SSD algorithm that incorporates Batch Norm algorithm. The object detection system was built by the Flask framework and the Layui framework. The system can select the data to be detected on the front-end page, the detection results and the number of each type of object were displayed on the front-end page in real time.

2.2 Object Detection Method Based on Improved One-Stage Detector

Abstract:

We use an efficient way to simulate the two-stage detector, and optimize the algorithm to improve efficiency of object detection as the accuracy of the one-stage detector often lags behind that of the two-stage detector. We have improved the one-stage detector to get higher accuracy on SSD. We mainly adjust the default boxes so that it can well capture the range of the target object in the image, and then the module given the adjusted boxes responds to the alignment of the receptive field of the convolution filter. Test results on the PASCAL VOC dataset show that the final model accuracy can reach that of the advanced detector.

2.3 A Deep Learning based Real Time Assistive Framework for Visually Impaired

Abstract:

Visually impaired people face great difficulties in interacting with unfamiliar surroundings and making use of public amenities. Commuting and using public services without any external help is very challenging to the visually impaired. Over time there have been numerous developments to aid visually impaired but mostly these have been limited the use of sensors to measure distances and send warning signals. The proposed work presents a deep learning technique to aid the visually impaired in real time by identifying nearby public amenities that are used by all in everyday life like Restrooms, ATMs, Metro stations and Pharmacies. This method used faster region-convolutional neural networks (Faster R-CNN) with resnet50 to identify the symbols of public amenities that are the same around the world. This algorithm was trained with a set of over 450 images and was tested on a varying database and achieved an accuracy of 92.13 percent. The experimental results to identify public amenities like

Restrooms, ATMs, Metro stations and Pharmacies are robust, efficient, low in cost and helps in providing sight to the visually impaired up to a large extent.

2.4 Human-in-the-Loop Machine Learning to Increase Video Accessibility for Visually Impaired and Blind Users

Abstract:

Video accessibility is crucial for blind and visually impaired individuals for education, employment, and entertainment purposes. However, professional video descriptions are costly and time-consuming. Volunteer-created video descriptions could be a promising alternative, however, they can vary in quality and can be intimidating for novice describers. We developed a Human-in-the-Loop Machine Learning (HILML)

approach to video description by automating video text generation and scene segmentation and allowing humans to edit the output. The HILML approach facilitates human-machine collaboration to produce high quality video descriptions while keeping a low barrier to entry for volunteer describers. Our HILML system was significantly faster and easier to use for first-time video describers compared to a human-only control condition with no machine learning assistance. The quality of the video descriptions and understanding of the topic created by the HILML system compared to the human-only condition were rated as being significantly higher by blind and visually impaired users.

2.5 Smartphone-Based Assistance for Blind People to Stand in Lines

Abstract:

We present a system to allow blind people to stand in line in public spaces by using an off-the-shelf smartphone only. The technologies to navigate blind pedestrians in public spaces are rapidly improving, but tasks which require to understand surrounding people's behavior are still difficult to assist. Standing in line at shops, stations, and other crowded places is one of such tasks. Therefore, we developed a system to detect and notify the distance to a person in front continuously by using a smartphone with a RGB camera and an infrared depth sensor. The system alerts three levels of distance via vibration patterns to allow users to start/stop moving forward to the right position at the right timing. To evaluate the effectiveness of the system, we performed a study with six blind people. We observed that the system enables blind participants to stand in line successfully, while also gaining more confidence.

2.6 Tools and Technologies for Blind and Visually Impaired Navigation Support

Abstract:

The development of navigation tools for people who are visually impaired had become an important concern in the research area of assistive technologies. This paper gives a comprehensive review of different articles published in the area of navigation solutions for people who are visually impaired. Unlike other review papers, this review considers major solutions that work in both the indoor or/and outdoor environments which are based on different technology. From the review, it became clear that the navigation systems proposed for the target users lack some core features that are quite important for independent navigation. Also, there can be instances in which humanitarian conditions also have to be considered in the navigation system design. Based on these findings, a set of recommendations are also given which can be considered in the future design of navigation systems for

blind and visually impaired people.

2.7 Virtual navigation for blind people: Transferring route knowledge to the real-World

Abstract:

Independent navigation is challenging for blind people, particularly in unfamiliar environments. Navigation assistive technologies try to provide additional support by guiding users or increasing their knowledge of the surroundings, but accurate solutions are still not widely available. Based on this limitation and on the fact that spatial knowledge can also be acquired indirectly (prior to navigation), we developed an interactive virtual navigation app where users can learn unfamiliar routes before physically visiting the environment. Our main research goals are to understand the acquisition of route knowledge through smartphone based virtual navigation and how it evolves over time; its ability to support independent, unassisted real-world navigation of short routes; and its ability to improve user performance when using an accurate in-situ navigation tool (NavCog). With these goals in mind, we conducted a user study where 14 blind participants virtually learned routes at home for three consecutive days and then physically navigated them, both unassisted and with NavCog. In virtual navigation, we analyzed the evolution of route knowledge and we found that participants were able to quickly learn shorter routes and gradually increase their knowledge in both short and long routes. In the real-world, we found that users were able to take advantage of this knowledge, acquired completely through virtual navigation, to complete unassisted navigation tasks. When using NavCog, users tend to rely on the navigation system and less on their prior knowledge and therefore virtual navigation did not significantly improve users' performance.

2.8 Learning an Object Detector Using Zoomed Object Regions

Abstract:

The single shot multi-box detector (SSD) is one of the first real-time detectors, which uses a convolutional neural network (CNN) and achieves the state-of-the-art detection performance. However, owing to the semantic gap between each feature layer of CNN, the SSD has a room for improvement. In this paper, we propose a novel training scheme to enhance the performance of the SSD. In object detection, ground truth (GT) box is a bounding box enclosing an object boundary. To improve the semantic level

of the feature map, we generate additional GT boxes by zooming in to and out from the original GT boxes. Experimental results show that the SSD trained with our scheme outperforms the original one on public dataset.

2.9 Real Time Multi Object Detection for Blind Using Single Shot Multibox Detector

Abstract:

According to world health statistics 285 million out of 7.6 billion population suffers visual impairment; hence 4 out of 100 people are blind. Absence of vision restricts the mobility of a person to pronounced extent and hence there is a need to build an explicit device to conquer guiding aid to the prospect. This paper proposes to build a prototype that performs real time object detection using image segmentation and deep neural network. Further the object, its position with respect to the person and accuracy of detection is prompted through speech stimulus to the blind person. The accuracy of detection is also prompted to the device holder. This work uses a combination of single-shot multibox detection framework with mobileNet architecture to build rapid real time multi object detection for a compact, portable and minimal response time device construction.

2.10 A study on object detection method from manga images using CNN

Abstract:

Japanese comics (manga) are popular content worldwide. In order to acquire metadata from manga images, techniques automatic recognition of manga content have been

studied. Recently, Convolutional Neural Network (CNN) has been applied to object detection in manga images. R-CNN and Fast R-CNN generate region proposals by Selective Search. Faster R-CNN generates them using CNN layers called Region Proposal Network (RPN). Single Shot MultiBox Detector (SSD), the latest detection method, performs object classification and box adjustment for small regions in an image. These methods are effective to natural images. However, it is unclear whether such methods work properly to manga images or not, since those image features are different from natural images. In this paper, we examine the effectiveness of manga object detection by comparing Fast R-CNN, Faster R-CNN, and SSD. Here, manga objects are panel layout, speech balloon, character face, and text. Experimental results show that Fast R-CNN is effective for panel layout and speech balloon, whereas Faster R-CNN is effective for character face and text.

2.11 Speed-Up of Object Detection Neural Network with GPU

Abstract:

We realized a speed-up of an object detection neural network with GPU. We

improved the object detection speed of faster R-CNN [1], which is one of the most commonly used detection networks [2]. The speed of the original faster R-CNN (py-faster-rcnn [3]) was 72.4ms per image on our GPU server 1. We accelerated the detection speed by implementing our new algorithms that are suitable for GPUs. The speed-up is realized without sacrificing the object detection accuracy (mAP). Our GPUaccelerated faster R-CNN can detect objects with 55.8ms per image. This is nearly 30% speed-up.

In detection networks, the processes of building scored candidate regions, sorting and non-maximum-suppression (nms) are commonly used. In faster R-CNN, these processes are executed in proposal layer. We reduced the processing time of the proposal layer from 5.6ms to 2.2ms. This is 2.5 times as fast as the original one. We also evaluated the detection speed with larger batch sizes. By applying batch size 16, it is accelerated to 44.9ms per image. This is 1.6 times as fast as the original faster RCNN (py-faster-rcnn). Since we realized a speed-up of common basic methods for detection networks, our speed-up methods are also applicable to other detection networks such as R-FCN [4], YOLO [5] [6] and SSD [7].

2.12 Camera-Based Object Detection, Identification and Distance Estimation

Abstract:

A system combining deep learning and stereovision for detection, tagging and distance estimation of objects ahead, is presented in this study. A Convolutional Neural Network

(CNN) is used to detect and identify objects in the field of vision of the stereo camera. Once the objects are detected and identified, their distance from the camera is estimated by first constructing 3D point cloud using triangulation method. The system is implemented on Nvidia® Jetson TX1 with a Zed® stereo camera, tested and found to be capable of a detection accuracy of around 84% with the average error in distance estimation less than 4.7% up to a distance of five meters.

2.13 Object Detection Using Convolutional Neural Networks

Abstract:

Vision systems are essential in building a mobile robot that will complete a certain task like navigation, surveillance, and explosive ordnance disposal (EOD). This will make the robot controller or the operator aware what is in the environment and perform the next tasks. With the recent advancement in deep neural networks in image processing, classifying and detecting the object accurately is now possible. In this paper, Convolutional Neural Networks (CNN) is used to detect objects in the environment. Two state of the art models are compared for object detection, Single Shot Multi-Box Detector (SSD) with MobileNetV1 and a Faster Region-based

Convolutional Neural Network (Faster-RCNN) with InceptionV2. Result shows that one model is ideal for real-time application because of speed and the other can be used for more accurate object detection.

2.14 In-Door Assistant Mobile Application Using CNN and TensorFlow

Abstract:

Visually impaired people struggle to live without assistance or face any aspect of life alone especially with people that cannot afford extra assistance equipment. Usually impaired people receive assistance by either human or wearable devices. The first one bears the burden on the human, while the second adds financial burdens nevertheless the hassle of identifying an object is not decreased. Smartphones are almost accessible to everyone and equipped with accessibility features including sensors that can be utilized to help both visually impaired and sighted people. Thus, this paper proposes an approach using Convolutional Neural Network (CNN), speech recognition and smartphone camera calibration aiming at facilitating the process of indoor guidance for visually impaired people. A smartphone's camera acts as the user's eyes. A pre-trained CNN model is used for object detection and the distance to objects is calculated to guide the user toward the right directions and to warn them of obstacles. The speech recognition part is used as a communication channel between visually impaired people and the smartphone. Also, the proposed approach supports object personalizing that helps to distinguish user's item from other items found in the room. To evaluate the personalized object detection, a customized dataset is created for two objects. The experimental results indicate that the accuracy is 92% and 87% for both objects respectively. Also, we experiment the detect distance of two objects against their real distances. The results achieve 0.05 and 0.08 error ratio.

2.15 Android Based Object Detection System for Visually Impaired

Abstract:

Vision is one of the most significant human senses among all the human senses present, and it assumes a crucial job in the understanding of the surrounding environment. Visually impaired people find it difficult to move around outdoors without any supervision. Hence, this paper is an attempt at developing an object detection system for people with diminished sight. In order to do this, it requires a few segments such as a camera, an application, and an audio device. We have designed and implemented an Android application that will use the phone's camera to detect the objects around the visually impaired user. Moreover, the application will also inform the user about the direction of the object and the distance of the user from the object. The application will inform the visually impaired user about the name, direction, and distance of the object using an audio device such as headphones or

the phone's speaker. This system will assist the visually impaired people by informing them about the various objects around them and help them navigate around independently. Thus, our goal is to put forward a system of visual substitution which will help the visually impaired people in their day-to-day life by informing them about the various objects around them using an object-detection system.

2.16 Object recognition in images using convolutional neural network

Abstract:

Object detection from repository of images is challenging task in the area of computer vision and image processing in this work we present object classification and detection using cifar-10 data set with intended classification and detection of airplane images. So we used convolutional neural network on keras with TensorFlow support the experimental results shows the time required to train, test and create the model in limited computing system. We train the system with 60,000 images with 25 epochs each epoch is taking 722to760 seconds in training step on TensorFlow CPU system. At the end of 25 epochs the training accuracy is 96 percentage and the system can recognition input images based on train model and the output is respective label of images.

2.17 Intelligent Vision with TensorFlow using Neural Network Algorithms

Abstract:

Computer vision and video analytics are the torrid research area in Machine learning and their establishment process traditionally starts with object detection and eventually

tracking. In recent years, there is a tremendous growth in performing comprehensive study based on the field of object detection and Pattern Analysis. In our system we have improvised and experimented with detection method based on machine learning and deep learning approach in object recognition and pattern analysis. We assume object detection as a retrogression problem to spatially separated corresponding class

probabilities and bounding boxes. Many prominent algorithms have been designed for object detection, Pattern Analysis and tracking, which also includes edge tracking, color segmentation and pattern matching. A single neural network is capable of predicting class probabilities and bounding boxes directly from the full image per cycle. Therefore we have used various neural network algorithms such as YOLOv3, Single Shot Multiple detection algorithm to carry out video analysis using object detection and drowsiness detection using pattern or behavior analysis with the help of TensorFlow. The framework will recognize object continuously, from the input perceived through camera where it can apparently capture a required frames to

predict the object and also to match the pattern. It has been accomplished using real-time video processing and a single camera. The proposed system is versatile to operate in complex, real time, non-plain environment.

2.18 Real-Time Deep Learning-Based Object Detection Framework

Abstract:

Recently real-time detection, and recognition of an object of interest are becoming vital tasks in visual data processing and computer vision. Various models have been deployed to implement object detection and tracking in multiple fields. However, conventional classifiers are often faced with challenging tasks that visual frames come distorted due to overlapping, camera motion blur, changing subject appearances, and environmental variations. Models using OpenCV-based HAAR feature-based cascade classifiers, without integrating any error minimizing object detection algorithm, were unable to accurately detect an object and track it in a changing environment. Therefore, developing an embedded powerful framework for real time object detection and recognition becomes more of a vital need for future implementation in various fields. This study presents a powerful technique for a real-time detector that utilizes integrated Deep Learning Neural Networks (DNN) for optimal computational accuracy. Deploying such a framework will ensure the flexibility and reliability of the detector by eliminating the sources of distortion previously mentioned. The model relies on integrating the Image AI deep learning libraries and You Only Look Once (YOLO-v3) object detection method with a DarkNet 53 architecture. The algorithm was trained using the TensorFlow framework to ensure accurate data processing. This paper targets one vital component of our long-term project of developing a multi-agent system, as the proposed model is to be implemented in autonomous agents for the detection of landmines, ocean debris, and wildlife beside environmental scanning missions. In this study the performance of the model has been assessed through detecting and collecting tennis balls as a preliminary test for real-world applications. The model was able to approach the desirable result of surpassing the accuracy of conventional detectors.

2.19 Detection, identification and tracking of objects during the motion

Abstract:

This paper intends to introduce discovery of various objects, object classification, object tracking algorithms including analysis and a comparison of different techniques used for different stages of tracking. The purpose of tracking objects is segmenting a region of interest from a video scene and continuing tracking movement, positioning, and match. Detection of the object and classification are the steps preceding the object tracking in a sequence of images. Object detection is

performed to control the existence of objects in the video and find that object, and then the detected object can be classified into many categories such as people, vehicles, floating trees, trees and other moving objects. In this paper is elaborated the autonomous tracking system with

the aim of opposing movement in the event of a demonstration of human tracking, as such may enable the former to be distributed as it comprises more than one control center, key component parts of the system include: The camera and its 'interface, Arduino viewing boards., PC - Controlled Center.

2.20 Multiple Real-time object identification using Single shot Multi-Box detection

Abstract:

Real time object detection is one of the challenging tasks as it need faster computation power in identifying the object at that time. However, the data generated by any real time system are unlabeled data which often need large set of labeled data for effective training purpose. This paper proposed a faster detection method for real time object detection based on convolution neural network model called as Single Shot Multi-Box

Detection (SSD). This work eliminates the feature resampling stage and combined all calculated results as a single component. Still there is a need of a light weight network model for the places which lacks in computational power like mobile devices (eg: laptop, mobile phones, etc.). Thus, a light weight network model which use depth-wise separable convolution called MobileNet is used in this proposed work. Experimental result reveal that uses of MobileNet along with SSD model increase the accuracy level in identifying the real time household objects.

2.21 Objects Talk - Object detection and Pattern Tracking using TensorFlow

Abstract:

Objects in household that are frequently in use often follow certain patterns with respect to time and geographical movement. Analysing these patterns can help us keep better track of our objects and maximise efficiency by minimizing time wasted in forgetting or searching for them. In our project, we used TensorFlow, a relatively new library from Google, to model our neural network. The TensorFlow Object Detection API is used to detect multiple objects in real-time video streams. We then introduce an algorithm to detect patterns and alert the user if an anomaly is found. We consider the research presented by Laube et al., Finding REMO—detecting relative motion patterns in geospatial lifelines.

2.22 Object Recognition using TensorFlow

Abstract:

Computers can apply vision technologies using cameras and artificial intelligence software to achieve image recognition and identify objects, places, and people. The objective of this project is to capture the image of an automobile as it drives by, identify its model and color, and determine its location, travel direction, and speed. This system can be used to assist law enforcement with vehicle identification in an emergency such as an Amber alert or to detect traffic infractions. For this purpose, we constructed, trained, and applied an object detection model using TensorFlow. First, an image capturing system was built using camera lenses (Raspberry Pi Camera V2-8) and Raspberry Pi (Raspberry Pi 4) small computers. Next, the computers were set up with a software application called TensorFlow. The system was trained to recognize an automobile's model and color by processing a variety of car images. Pictures of different cars were uploaded from Google images and resized highlighting the features of the vehicle. Finally, code was developed in Python to create a universal clock for each camera that recorded the detection time. Five trials were conducted using 2 automobiles available for testing. The cars were recognized by the model with 87 percent certainty in each of the 5 trials. That information was recorded on a table together with the time of capture and the location of the camera. The information from the table was used to successfully identify a specific car's location and speed, with a few limitations. Because of budget restrictions only two cameras were built and two models were used for training. The information from the cameras was not transmitted in real time because wifi or LTE capability are not available at this time. An extension of this research will include multiple cameras, multiple models and real time data transmission.

2.23 Object Detection and Count of Objects in Image using Tensor Flow Object Detection API

Abstract:

Object Detection is widely utilized in several applications such as detecting vehicles, face detection, autonomous vehicles and pedestrians on streets. TensorFlow's Object Detection API is a powerful tool that can quickly enable anyone to build and deploy powerful image recognition software. Object detection not solely includes classifying and recognizing objects in an image however additionally localizes those objects and attracts bounding boxes around them. This paper mostly focuses on detecting harmful objects like threatening objects. To ease object detection for threatening objects, we have got Tensor flow Object Detection API to train model and we have used Faster R-CNN algorithm for implementation. The model is built on two

classes of threatening Objects. The model is evaluated on test data for the two classes of detecting threatening objects.

2.24 Real-time Object Detection for 360-degree Panoramic Image using CNN

Abstract:

The object detection for 360-degree panoramic images are widely applied in many areas such as automatic driving, navigation of drones and driving assistance. Most of state-of-the-art approaches for detecting objects in ordinary images cannot work well on the object detection for 360-degree panoramic images. As a 360-degree panoramic image can be considered as a 2D image which is the result of a 360-degree panoramic sphere being expanded along the longitude line, objects in it will be twisted or divided apart and the detection will be more difficult. In this paper, we present a real-time object detection system for 360-degree panoramic images using convolutional neural network (CNN). We adopt a CNN-based detection framework for object detection with a post-processing stage to fine-tune the result. Additionally, we propose a novel method to reuse those existing datasets of ordinary images, e.g., the ImageNet and PASCAL VOC, in the object detection for 360-degree panoramic images. We will demonstrate with several examples that our method yields higher accuracy and recall rate than traditional methods in object detection for 360-degree panoramic images.

2.25 Transfer learning-based Object Detection by using Convolutional Neural Networks

Abstract:

Object detection has become an important task for various purposes in our daily lives. Machine learning techniques have been used for this task from earlier but they are used for the classification of imagebased species to extract the feature set. This task of deciding the feature set helps to decide the desired object detection. To overcome the object classification problem, this paper proposes a transfer learning-based deep learning method. The different convolutional neural networks (CNN) are studied in this work. Here for the improvement in the result, the majority voting scheme is used. The overall work is carried out on the CUB 200-2011 dataset. The results obtained have shown incredible improvement in the accuracy of the proposed work when compared to the different CNN models.

2.26 Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Abstract:

State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with ‘attention’ mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

2.27 You Only Look Once: Unified, Real-Time Object Detection

Abstract:

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

The main agenda to initiate this setup is to make the hinderance and myth of calculating distance between object and the user using auto generated voice with great accuracy. Efficient Implementation of this Model depends upon the compatibility with python and library installation hurdles. We have implemented it by using TensorFlow APIs. The advantage one has by using APIs is it provides us with a set of common operations. TensorFlow object detection API is basically a structure build for creating a deep learning network that solves the problems for object detection. There are trained models in their framework and they refer it as Model Zoo. This includes a collection of COCO dataset, the KITTI dataset, and the Open Images Dataset. Here, we are primarily focused on COCO DATASETS. The above lines tell us the importance foe identification of wide range of objects. This AI framework makes the model think and enacts accordingly after the calculation of distance and scope of this project can be used in Obstacle Detection Mechanism.

CHAPTER 4

EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

4.1 LANGUAGE USED:

Python 3.9

4.2 METHODS USED:

TENSORFLOW API:

The TensorFlow article location Programming interface is essentially a design work for making a profound learning network that tackles the issues for object recognition.

TensorFlow API depends on following Python libraries:

4.3 PYTHON LIBRARIES USED:

- PROTOBUF : Creates barrier between read and write messages.
- Python-Tk : It creates Graphical User Interface.
- Pillow 1.0 : An Imaging library that helps to embed an image.
- LXML : Establishes co-relates between XML and HTML with Python.

- TF-slim : Assists evaluating complex models in TensorFlow.
- Jupiter notebook : Web lines the Python IDE.
- Matplotlib : Initiates the graph for Machine Learning algorithms
- TensorFlow : A comprehensive platform for Machine Learning.
- Contextlib2 : A backport that connects a rear Python versions.
- Coco API : It provides an API for Object Detection.

4.4 VOICE GENERATING LIBRARIES:

Audio commands are achieved by certain libraries like:

- Pytorch : A Deep Learning Tensor that operates on arrays.
- Pyttsx3 : Python interpretation library that changes over text into speech .
- Pytesseract : Meticulous technology that tries to identify in-written characters.
- Engine.io : Basic foundation of generating artificial voice from the system.

4.5 ALGORITHMS USED:

CNN (CONVOLUTIONAL NEURAL NETWORK):

First phase - In this phase, it analyses the image or video captured with the aid of using the digital digicam the use of numerous layers: Convolutional layer, Pooling layer, Classifier Layer.

SSD (SINGLE SHOT DETECTOR):

This is the second segment of the Blind Assisting project. In this segment, the detection and identity of the item are achieved in step with colour, and size. Post detection, the item is rounded via way of means of the Anchor container line. For an input picture ,the spine brings about a 256 7x7 component maps in ResNet34 . SSD partitions the picture utilizing a matrix and have every network cell be liable for identifying objects around there of the picture. Identifying objects essentially implies foreseeing the class and area of an item inside that locale.

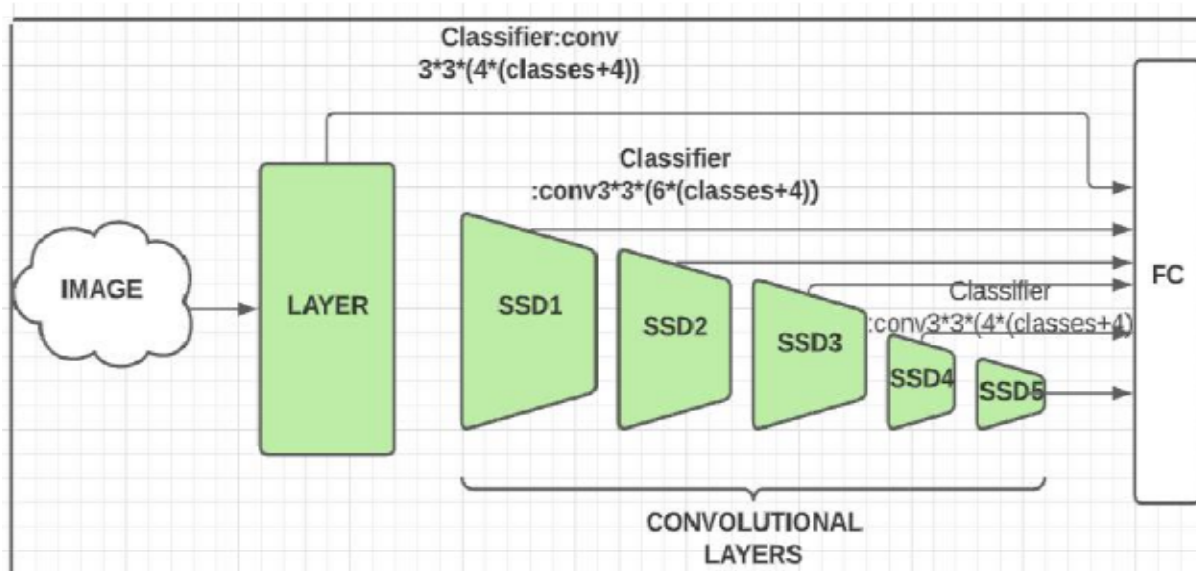


Fig.1 : SSD (Single Shot Detector) Algorithm

ANCHOR BOX:

Multiple anchor/earlier boxes can be relegated to every network cell in SSD. These relegated anchor boxes are pre-characterized and everyone is liable for a size and shape inside a matrix cell. Matching stage is utilized by SSD while preparing, so that there's a suitable coordinate to moor box with the jumping boxes of each ground truth object inside a picture. For foreseeing that article's class and its area the anchor box with the most significant level of cross-over with an item is responsible. Once the organization has been trained, this property is utilized for preparing the organization and for anticipating the recognized articles and their areas. For all intents and purposes, each anchor box is determined with a viewpoint proportion and a zoom level. Well, we realize that all items are not square in shape. Some are more limited, some are longer and some are more extensive, by differing degrees. The SSD design permits pre-characterized perspective proportions of the anchor boxes to represent this. The different angle proportions can be indicated utilizing proportions boundary of the anchor boxes related with every lattice cell at each zoom/scale level.

ZOOM LEVEL:

It isn't compulsory for the anchor boxes to have the very size as that of the lattice cell. The client may be keen on tracking down both more modest or bigger articles inside a network cell. To indicate how much the anchor boxes should be increased or down regarding every matrix cell, the zooms boundary is utilized.

MOBILENET:

This model depends on the philosophy of THE MobileNet model in view of depth wise detachable convolutions and it shapes a factorized Convolutions. These believers an essential standard convolutions into a depth wise convolutions. This 1×1 convolutions are additionally called as pointwise convolutions. For MobileNets to work, these depth wise convolutions applies an overall single channel based idea to every one of the info channels. These pointwise convolutions applies a 1×1 convolutions to converge with the results of the depth wise convolutions. As a standard convolution the two channels consolidates the contributions to another arrangement of results in a single step. The depth wise recognizable convolutions parts this into two layers - a different layer for the sifting reason and the other separate layer for the consolidating reason. This factorization technique has the impact of radically lessening the calculation and that of the model size.

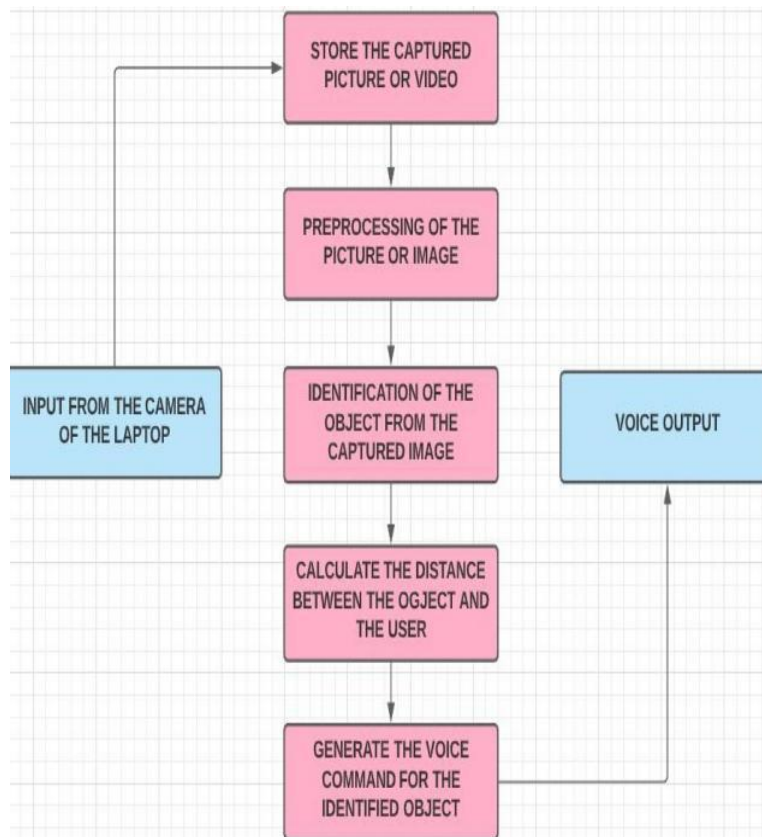
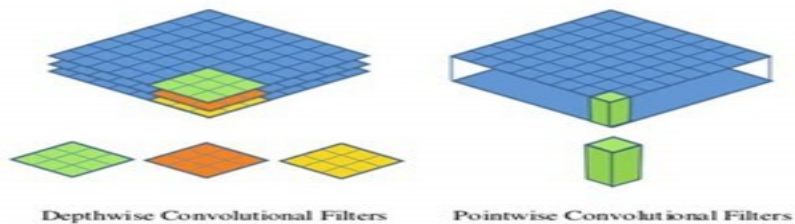


Fig.2: Architecture Diagram

DEPTH ESTIMATION:

Profundity assessment or extraction highlight is only the procedures and calculations which expects to get a portrayal of the spatial construction of a scene. In less complex words, ascertaining the distance between two objects is utilized. Our model is utilized to help the visually impaired individuals which intends to give advance notice to the visually impaired individuals about the obstacles coming on their way. To do this, we really want to observe that at how much distance the snag and individual are situated experiencing the same thing. After the item is recognized rectangular box is produced around that article.

DISTANCE APPROXIMATIONS:

If the thing includes an immense piece of the scope, a couple of constraints examine the division from a particular person to the smaller than expected PC. The going with code is used to recognize and respond to region information.

```
(boxes, scores, classes, num_detections) = sess.run([boxes,
    scores, classes, num_detections], feed_dict={image_tensor:
    image_np_expanded})
```

Here we present the TensorFlow module, which includes key parts of the thought. Thusly, for specific young people, as they get more prepared, he will adjust to it. The case is in progression, before the line. Yet again this suggests that you really want to spend on these cycles.

VOICE GENERATING MODULE:

After the identification of an article, it is most extreme vital to recognize the individual about the presence of that item on his/her way. For the voice age module PYTTSX3 assumes a significant part. Pyttsx3 is a transformation library in Python which changes over text into discourse. This library functions admirably with both Python 2 and 3. To get reference to a pyttsx. Motor example, a manufacturing plant work called as pyttsx.init() is conjured by an application. Pyttsx3 is an apparatus which changes text over to discourse without any problem.

This calculation fills in as at whatever point an item is being identified, surmised distance is being calculated, with the assistance of cv2 library and cv2.putText()

work, the texts are getting showed on to the screen. To distinguish the secret text in an image, we use Python-tesseract for character acknowledgment. OCR identifies the text content on pictures and encodes it in the structure which is effortlessly perceived by the PC. This text discovery is finished by checking and investigation of the picture. Along these lines, the text installed in pictures are perceived and "read" utilizing Python-tesseract. Further these texts are highlighted a pyttsx.Engine occasion, an industrial facility work called as pyttsx.init() is conjured by an application. During development, a pyttsx.driver. DriverProxy object is introduced by motor which is answerable for stacking a discourse motor driver from the pyttsx.drivers module. After development, an item made by a motor is utilized by the application to enlist and unregister occasion call-backs; produce and stop discourse; get and set discourse motor properties; and begin and stop occasion circles.

4.6 FLOW CHART

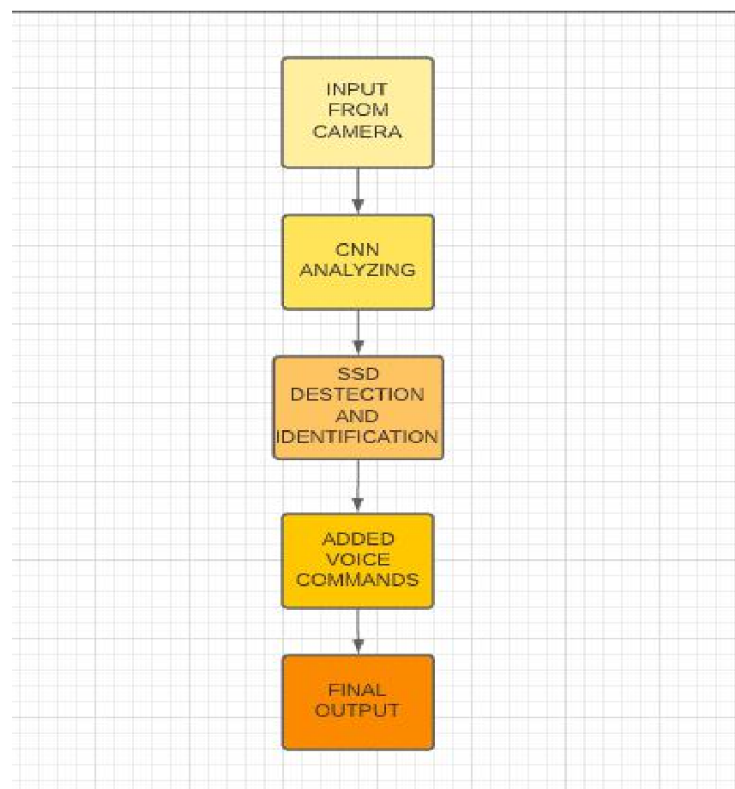


Fig.3: OVERVIEW

CHAPTER 5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

Knowing things. Counting the distance to an article. Discover more. Voice Assistance: This alarms the patron assuming that something is excessively close. Moreover, it distinguishes the object proven withinside the API streaming tensor. Coming up next is a rundown of our most well-known pages, including the accompanying: matplotlib libraries followed by the precise results as depicted below. The toast message of accuracy is displayed with the assistance of TensorFlow API.

The last accuracy of controller is 91%. The last distance is 0.8 parts from the line, it is a protected distance, so there is no reasonable class sign and no sign demonstrating the distance given, taking everything into account. (Fig.7)

The last accuracy of bed is 94%. The last distance is 0.9 units from the level, it is a protected distance, so there is no development in light of the level, yet the level declaration sound will be set and the name of the thing will seem like a bed. (Fig.8)

The last distance is 0.7 units from the line, it is a protected distance, and the sound is clear, and the school doesn't give a distance in light of the data since it is a seat. It can distinguish different areas. (Fig.9)

The highest accuracy to this capture is 96% and the last distance is 0.8 units from the level. Our project can detect multiple objects as shown in the above figure. (Fig.10)

5.1 PERFORMANCE ANALYSIS

X- axis: Time (in sec).

Y- axis: Accuracy values.

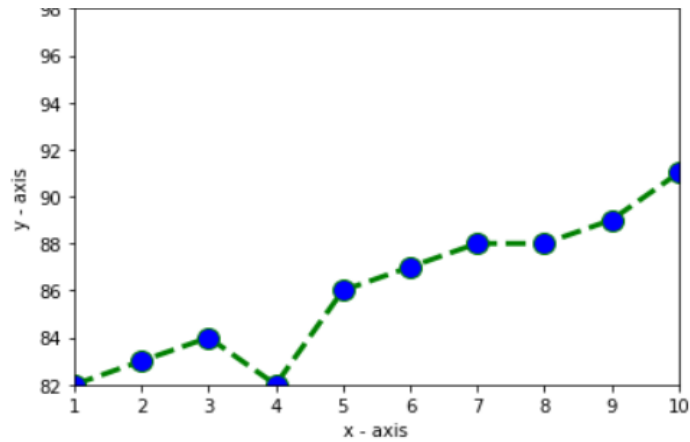


Fig.4: Performance analysis of remote

From the (Fig.7) and above graph, we have recorded the accuracy values of the remote in the time period of 10sec. With the highest accuracy of 91 percent.

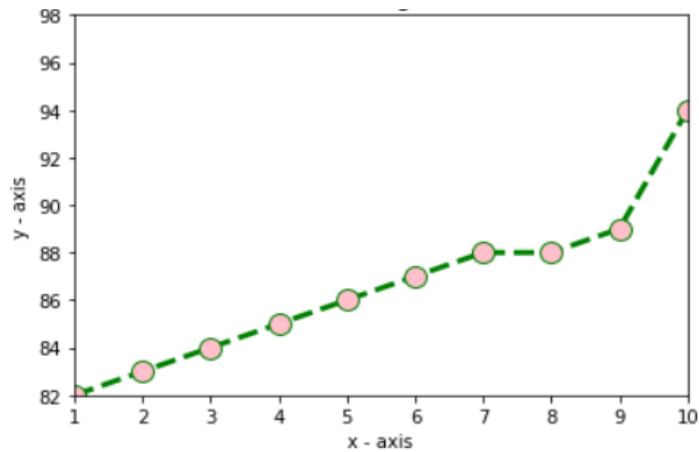


Fig.5: Performance analysis of Bed

From the (Fig.8) and above graph, we have recorded the accuracy values of the bed in the time period of 10sec. With the highest accuracy of 94 percent.

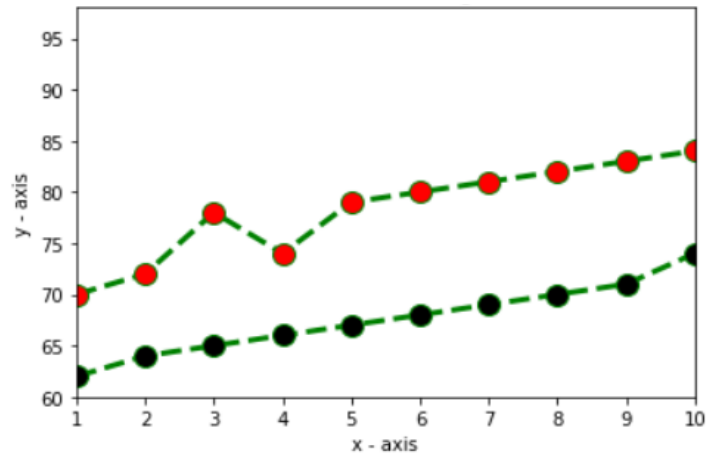


Fig.6: Performance analysis of Multiple Objects

From the (Fig.9) and above graph, we have recorded the accuracy values of the chair and bed in the time period of 10sec. With the highest accuracy of 84 and 74 percent respectively.

5.2 LIMITATIONS:

- In SSD, small objects can only be detected in higher resolution layers.
- When it comes to video, detectors need to be trained to identify the objects that are in motion.
- View point variation, Object viewed from different angles may look completely different.
- Sometimes objects can be disturbed by other things, makes it difficult to identify the objects.
- Finally, a good detector should detect any type of objects assigned.

CHAPTER 6

SUMMARY AND CONCLUSIONS

Binary digitalization of items encompassing us is the preeminent thought about this undertaking with voice associate orders to count the item to clear a way for outwardly tested individuals. Essentially The model processes the distance between the thing and camera to let us know the exact distance with fake voice produced by the framework. We have involved TensorFlow Programming interface for location of different articles. Followed by the above line in the tertiary arrangement of voice orders, pytsx3 python module have done the needful to incorporate voice module to the underlying construction to make it more available. The fundamental philosophy can be solidified with an application to connect outwardly impeded individuals.

REFERENCES

- [1] Q. Shuai and X. Wu, "Object detection system based on SSD algorithm," 2020 International Conference on Culture-oriented Science & Technology (ICCST), 2020, pp. 141-144, doi: 10.1109/ICCST50977.2020.00033.
- [2] W. Hongtao and Y. Xi, "Object Detection Method Based On Improved One-Stage Detector," 2020 5th International Conference on Smart Grid and Electrical Automation (ICSGEA), 2020, pp. 209-212, doi: 10.1109/ICSGEA51094.2020.00051.
- [3] D. Dahiya, H. Gupta and M. K. Dutta, "A Deep Learning based Real Time Assistive Framework for Visually Impaired," 2020 International Conference on Contemporary Computing and Applications (IC3A), 2020, pp. 106-109, doi:10.1109/IC3A48958.2020.233280.
- [4] Yuksel, Beste F. and Fazli, Pooyan and Mathur, Umang and Bisht, Vaishali and Kim, Soo Jung and Lee, Joshua Junhee and Jin, Seung Jung and Siu, Yue-Ting and Miele, Joshua A. and Yoon, Ilmi, Human-in-the-Loop Machine Learning to Increase Video Accessibility for Visually Impaired and Blind Users, 2020, isbn 9781450369749 doi: [10.1145/3357236.3395433](https://doi.org/10.1145/3357236.3395433)
- [5] Kayukawa, Seita and Takagi, Hironobu and Guerreiro, João and Morishima, Shigeo and Asakawa, Chieko, Smartphone-Based Assistance for Blind People to Stand in Lines, 2020, isbn 9781450368193, DOI = [10.1145/3334480.3382954](https://doi.org/10.1145/3334480.3382954)
- [6] Kuriakose, Bineeth, Shrestha, Raju Sandnes, Frode Eika, Tools and Technologies for Blind and Visually Impaired Navigation Support: A Review ,2020/09/27, 2020 ,DOI :10.1080/02564602.2020.1819893
- [7] João Guerreiro, Daisuke Sato, Dragan Ahmetovic, Eshed Ohn-Bar, Kris M. Kitani, Chieko Asakawa, Virtual navigation for blind people: Transferring route knowledge to the real-World, Volume 135, 2020, 102369, ISSN 1071-5819, DOI :10.1016/j.ijhcs.2019.102369.
- [8] S. Cho, S. Kim, K. Uhm, H. Kook and S. Ko, "Learning an Object Detector Using Zoomed Object Regions," 2019 International Conference on Electronics, Information, and Communication (ICEIC), 2019, pp. 1-3, doi:10.23919/ELINFOCOM.2019.8706381.
- [9] Arora, A., Grover, A., Chugh, R. et al. Real Time Multi Object Detection for Blind Using Single Shot Multibox Detector. Wireless Pers Commun 107, 651–661 (2019). DOI :10.1007/s11277-019-06294-1.
- [10] H. Yanagisawa, T. Yamashita and H. Watanabe, "A study on object detection method from manga images using CNN," 2018 International Workshop on Advanced Image Technology (IWAIT), 2018, pp. 1-4, doi: 10.1109/IWAIT.2018.8369633.
- [11] T. Fukagai et al., "Speed-Up of Object Detection Neural Network with GPU," 2018 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 301-305, doi: 10.1109/ICIP.2018.8451814.
- [12] Rahul and B. B. Nair, "Camera-Based Object Detection, Identification and Distance Estimation," 2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), 2018, pp. 203-205, doi: 10.1109/ICMETE.2018.00052.
- [13] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra and J. M. Z. Maningo, "Object Detection Using Convolutional Neural Networks," TENCON 2018 - 2018 IEEE Region 10 Conference, 2018, pp. 2023-2027, doi: 10.1109/TENCON.2018.8650517.

- [14] N. Khaled, S. Mohsen, K. E. El-Din, S. Akram, H. Metawie and A. Mohamed, "In-Door Assistant Mobile Application Using CNN and TensorFlow," 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), 2020, pp. 1-6, doi: 10.1109/ICECCE49384.2020.9179386.
- [15] A. Badave, R. Jagtap, R. Kaovasia, S. Rahatwad and S. Kulkarni, "Android Based Object Detection System for Visually Impaired," 2020 International Conference on Industry 4.0 Technology (I4Tech), 2020, pp. 34-38, doi: 10.1109/I4Tech48345.2020.9102694.
- [16] D. P. Sudharshan and S. Raj, "Object recognition in images using convolutional neural network," 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 718-722, doi: 10.1109/ICISC.2018.8398893.
- [17] A. R. Visalatchi, T. Navasri, P. Ranjanipriya and R. Yogamathi, "Intelligent Vision with TensorFlow using Neural Network Algorithms," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), 2020, pp. 944-948, doi: 10.1109/ICCMC48092.2020.ICCMC-000175.
- [18] W. Tarimo, M. M. Sabra and S. Hendre, "Real-Time Deep Learning-Based Object Detection Framework," 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2020, pp. 1829-1836, doi: 10.1109/SSCI47803.2020.9308493.
- [19] L. Koraqi and F. Idrizi, "Detection, identification and tracking of objects during the motion," 2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2019, pp. 1-6, doi: 10.1109/ISMSIT.2019.8932833.
- [20] S. Kanimozhi, G. Gayathri and T. Mala, "Multiple Real-time object identification using Single shot Multi-Box detection," 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), 2019, pp. 1-5, doi: 10.1109/ICCIDS.2019.8862041.

APPENDIX:

A: Screenshots:-

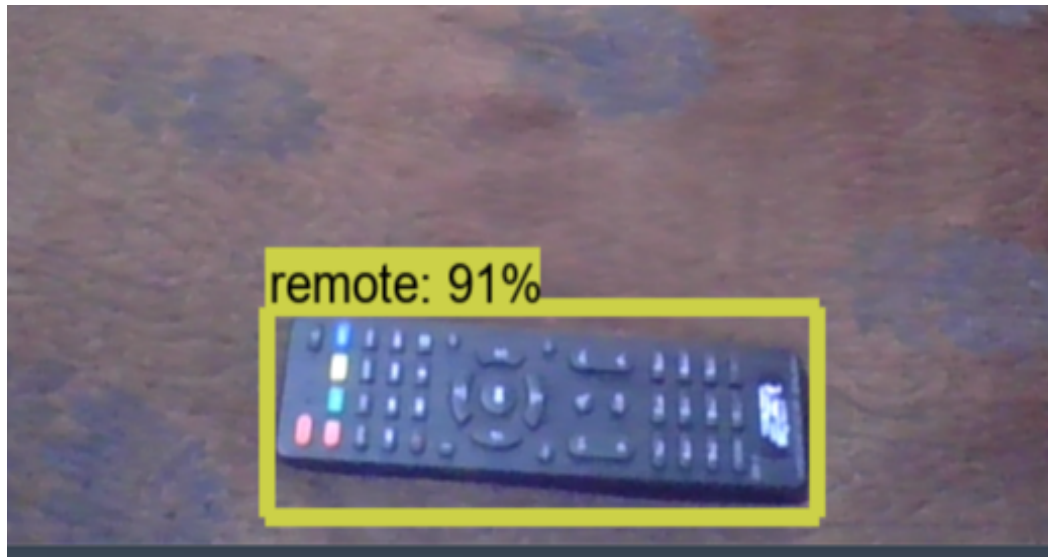


Fig.7: Identification of Remote



Fig.8: Identification of Bed



Fig.9: Identification of Two Objects

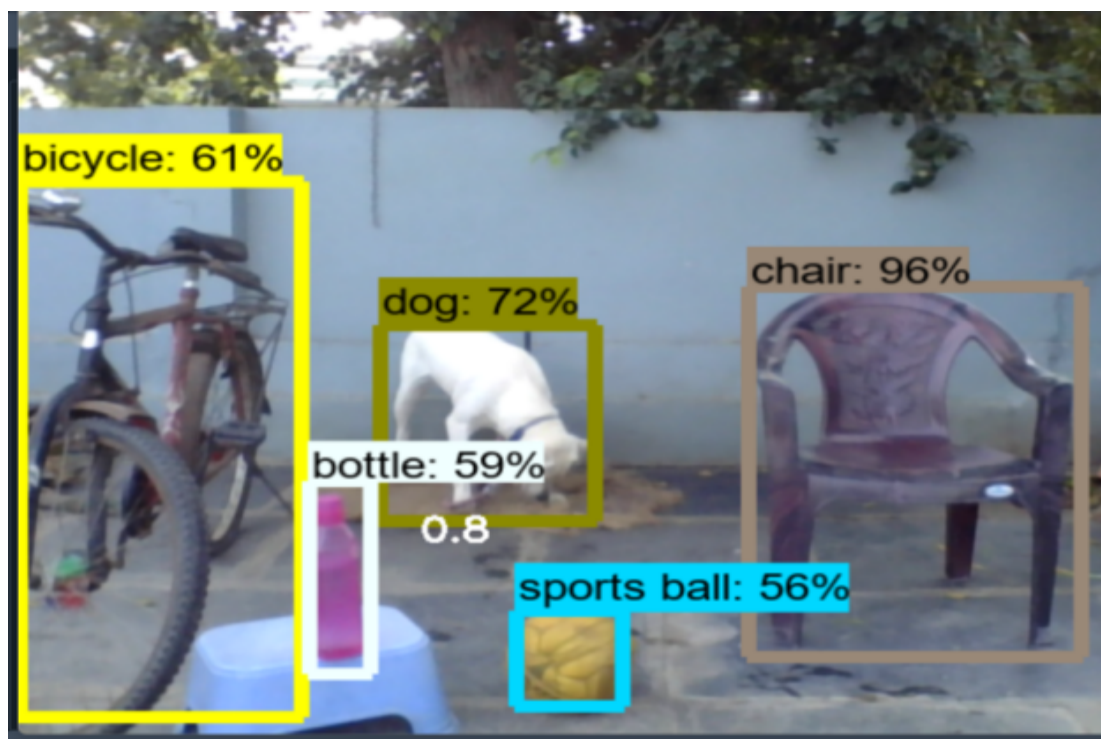


Fig.10: Identification of Multiple Objects

B: Source code:

```
import numpy as np
import os
import six.moves.urllib as urllib
import urllib.request as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import time
import pytesseract
import engineio

import torch
from torch.autograd import Variable as V
import models as models
from torchvision import transforms as trn
from torch.nn import functional as F

import pyttsx3
#from .engine import Engine
engine = pyttsx3.init()

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

arch = 'resnet18'

model_file = 'whole_%s_places365_python36.pth.tar' % arch
if not os.access(model_file, os.W_OK):
    weight_url = 'http://places2.csail.mit.edu/models_places365/' +
model_file
    os.system('wget ' + weight_url)

# = label_map_util.create_category_index(categories)

pytesseract.pytesseract.tesseract_cmd = 'C:\\Program Files
(x86)\\Tesseract-OCR\\tesseract'

from utils import label_map_util
#/object_detection/' m2

from utils import visualization_utils as vis_util
MODEL_NAME = 'ssd_inception_v2_coco_2017_11_17'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90
```

```

if not os.path.exists(MODEL_NAME + '/frozen_inference_graph.pb'):
    print ('Downloading the model')
    opener = urllib.request.URLopener()
    opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
    tar_file = tarfile.open(MODEL_FILE)
    for file in tar_file.getmembers():
        file_name = os.path.basename(file.name)
        if 'frozen_inference_graph.pb' in file_name:
            tar_file.extract(file, os.getcwd())
    print ('Download complete')
else:
    print ('Model already exists')

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
#
url='http://10.67.208.240:8080//shot.jpg'

import cv2
cap = cv2.VideoCapture(0)

with detection_graph.as_default():
    with tf.compat.v1.Session(graph=detection_graph) as sess:
        ret = True
        while (ret):
            ret,image_np = cap.read()

            if cv2.waitKey(20) & 0xFF == ord('b'):

                cv2.imwrite('opencv'+'.jpg', image_np)

        model_file = 'whole_%s_places365_python36.pth.tar' % arch
        if not os.access(model_file, os.W_OK):
            weight_url = 'http://places2.csail.mit.edu/models_places365/' +
model_file
            os.system('wget ' + weight_url)

        useGPU = 1
        if useGPU == 1:
            model = torch.load(model_file)
        else:
            model = torch.load(model_file, map_location=lambda storage,
loc: storage) # model trained in GPU could be deployed in CPU machine like
this!

```

```

model.eval()

centre_crop = trn.Compose([
    trn.Resize((256,256)),
    trn.CenterCrop(224),
    trn.ToTensor(),
    trn.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

file_name = 'categories_places365.txt'
if not os.access(file_name, os.W_OK):
    synset_url =
'https://raw.githubusercontent.com/csailvision/places365/master/categories_places365.txt'
    os.system('wget ' + synset_url)
classes = list()
with open(file_name) as class_file:
    for line in class_file:
        classes.append(line.strip().split(' ')[0][3:])
classes = tuple(classes)

img_name = 'opencv.jpg'
if not os.access(img_name, os.W_OK):
    img_url = 'http://places.csail.mit.edu/demo/' + img_name
    os.system('wget ' + img_url)

img = Image.open(img_name)
input_img = V(centre_crop(img).unsqueeze(0), volatile=True)

logit = model.forward(input_img)
h_x = F.softmax(logit, 1).data.squeeze()
probs, idx = h_x.sort(0, True)

print('POSSIBLE SCENES ARE: ' + img_name)
engine.say("Possible Scene may be")
engine.say(img_name)

for i in range(0, 5):
    engine.say(classes[idx[i]])
    print('{}'.format(classes[idx[i]]))

# Expand dimensions since the model expects images to have shape: [1,
None, None, 3]
image_np_expanded = np.expand_dims(image_np, axis=0)
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
# Each box represents a part of the image where a particular object was
detected.
boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represent how level of confidence for each of the objects.
# Score is shown on the result image, together with the class label.
scores = detection_graph.get_tensor_by_name('detection_scores:0')

```

```

classes = detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = detection_graph.get_tensor_by_name('num_detections:0')
# Actual detection.
(boxes, scores, classes, num_detections) = sess.run(
    [boxes, scores, classes, num_detections],
    feed_dict={image_tensor: image_np_expanded})

# Visualization of the results of a detection.
if cv2.waitKey(2) & 0xFF == ord('a'):
    vis_util.vislize_boxes_and_labels_on_image_array(
        image_np,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8)
else:
    vis_util.visualize_boxes_and_labels_on_image_array(
        image_np,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8)
if cv2.waitKey(2) & 0xFF == ord('r'):
    text=pytesseract.image_to_string(image_np)
    print(text)
    engine.say(text)
    engine.runAndWait()

for i,b in enumerate(boxes[0]):

    #                car                bus                truck
    if classes[0][i] == 3 or classes[0][i] == 6 or classes[0][i] == 8:
        if scores[0][i] >= 0.5:
            mid_x = (boxes[0][i][1]+boxes[0][i][3])/2
            mid_y = (boxes[0][i][0]+boxes[0][i][2])/2
            apx_distance = round(((1 - (boxes[0][i][3] -
boxes[0][i][1]))**4),1)
            cv2.putText(image_np, '{}'.format(apx_distance),
(int(mid_x*800),int(mid_y*450)), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(255,255,255), 2)

            if apx_distance <=0.5:
                if mid_x > 0.3 and mid_x < 0.7:
                    cv2.putText(image_np, 'WARNING!!!', (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255), 3)
                    print("Warning -Vehicles Approaching")
                    engine.say("Warning -Vehicles Approaching")
                    engine.runAndWait()

```

```

        if classes[0][i] ==44:
            if scores[0][i] >= 0.5:
                mid_x = (boxes[0][i][1]+boxes[0][i][3])/2
                mid_y = (boxes[0][i][0]+boxes[0][i][2])/2
                apx_distance = round(((1 - (boxes[0][i][3] -
boxes[0][i][1]))**4),1)
                cv2.putText(image_np, '{}'.format(apx_distance),
(int(mid_x*800),int(mid_y*450)), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(255,255,255), 2)
                print(apx_distance)
                engine.say(apx_distance)
                engine.say("units")
                engine.say("BOTTLE IS AT A SAFER DISTANCE")

            if apx_distance <=0.5:
                if mid_x > 0.3 and mid_x < 0.7:
                    cv2.putText(image_np, 'WARNING!!!', (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255), 3)
                    print("Warning -BOTTLE very close to the frame")
                    engine.say("Warning -BOTTLE very close to the frame")
                    engine.runAndWait()
        if classes[0][i] ==44:
            if scores[0][i] >= 0.5:
                mid_x = (boxes[0][i][1]+boxes[0][i][3])/2
                mid_y = (boxes[0][i][0]+boxes[0][i][2])/2
                apx_distance = round(((1 - (boxes[0][i][3] -
boxes[0][i][1]))**4),1)
                cv2.putText(image_np, '{}'.format(apx_distance),
(int(mid_x*800),int(mid_y*450)), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(255,255,255), 2)
                print(apx_distance)
                engine.say(apx_distance)
                engine.say("units")
                engine.say("cell phone IS AT A SAFER DISTANCE")

            if apx_distance <=0.5:
                if mid_x > 0.3 and mid_x < 0.7:
                    cv2.putText(image_np, 'WARNING!!!', (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255), 3)
                    print("Warning -cell phone very close to the
frame")
                    engine.say("Warning -cell phone very close to
the frame")
                    engine.runAndWait()
        if classes[0][i] ==1:
            if scores[0][i] >= 0.5:
                mid_x = (boxes[0][i][1]+boxes[0][i][3])/2
                mid_y = (boxes[0][i][0]+boxes[0][i][2])/2
                apx_distance = round(((1 - (boxes[0][i][3] -
boxes[0][i][1]))**4),1)
                cv2.putText(image_np, '{}'.format(apx_distance),
(int(mid_x*800),int(mid_y*450)), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(255,255,255), 2)
                print(apx_distance)
                engine.say(apx_distance)
                engine.say("units")

```

```

engine.say("Person is AT A SAFER DISTANCE")

if apx_distance <=0.5:
    if mid_x > 0.3 and mid_x < 0.7:
        cv2.putText(image_np, 'WARNING!!!', (50,50),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0,0,255), 3)
        print("Warning -Person very close to the frame")
        engine.say("Warning -Person very close to the frame")
        engine.runAndWait()

# plt.figure(figsize=IMAGE_SIZE)
# plt.imshow(image_np)
#cv2.imshow('IPWebcam',image_np)
cv2.imshow('image',cv2.resize(image_np,(1024,768)))
if cv2.waitKey(2) & 0xFF == ord('t'):
    cv2.destroyAllWindows()
    cap.release()
    break

#{1: {'id': 1, 'name': 'person'}, 2: {'id': 2, 'name': 'bicycle'}, 3: {'id':
3, 'name': 'car'}, 4: {'id': 4, 'name': 'motorcycle'}, 5: {'id': 5, 'name':
'airplane'}, 6: {'id': 6, 'name': 'bus'}, 7: {'id': 7, 'name': 'train'}, 8:
{'id': 8, 'name': 'truck'}, 9: {'id': 9, 'name': 'boat'}, 10: {'id': 10,
'name': 'traffic light'}, 11: {'id': 11, 'name': 'fire hydrant'}, 13: {'id':
13, 'name': 'stop sign'}, 14: {'id': 14, 'name': 'parking meter'}, 15: {'id':
15, 'name': 'bench'}, 16: {'id': 16, 'name': 'bird'}, 17: {'id': 17, 'name':
'cat'}, 18: {'id': 18, 'name': 'dog'}, 19: {'id': 19, 'name': 'horse'}, 20:
{'id': 20, 'name': 'sheep'}, 21: {'id': 21, 'name': 'cow'}, 22: {'id': 22,
'name': 'elephant'}, 23: {'id': 23, 'name': 'bear'}, 24: {'id': 24, 'name':
'zebra'}, 25: {'id': 25, 'name': 'giraffe'}, 27: {'id': 27, 'name':
'backpack'}, 28: {'id': 28, 'name': 'umbrella'}, 31: {'id': 31, 'name':
'handbag'}, 32: {'id': 32, 'name': 'tie'}, 33: {'id': 33, 'name':
'suitcase'}, 34: {'id': 34, 'name': 'frisbee'}, 35: {'id': 35, 'name':
'skis'}, 36: {'id': 36, 'name': 'snowboard'}, 37: {'id': 37, 'name': 'sports
ball'}, 38: {'id': 38, 'name': 'kite'}, 39: {'id': 39, 'name': 'baseball
bat'}, 40: {'id': 40, 'name': 'baseball glove'}, 41: {'id': 41, 'name':
'skateboard'}, 42: {'id': 42, 'name': 'surfboard'}, 43: {'id': 43, 'name':
'tennis racket'}, 44: {'id': 44, 'name': 'bottle'}, 46: {'id': 46, 'name':
'wine glass'}, 47: {'id': 47, 'name': 'cup'}, 48: {'id': 48, 'name': 'fork'},
49: {'id': 49, 'name': 'knife'}, 50: {'id': 50, 'name': 'spoon'}, 51: {'id':
51, 'name': 'bowl'}, 52: {'id': 52, 'name': 'banana'}, 53: {'id': 53, 'name':
'apple'}, 54: {'id': 54, 'name': 'sandwich'}, 55: {'id': 55, 'name':
'orange'}, 56: {'id': 56, 'name': 'broccoli'}, 57: {'id': 57, 'name':
'carrot'}, 58: {'id': 58, 'name': 'hot dog'}, 59: {'id': 59, 'name':
'pizza'}, 60: {'id': 60, 'name': 'donut'}, 61: {'id': 61, 'name': 'cake'},
62: {'id': 62, 'name': 'chair'}, 63: {'id': 63, 'name': 'couch'}, 64: {'id':
64, 'name': 'potted plant'}, 65: {'id': 65, 'name': 'bed'}, 67: {'id': 67,
'name': 'dining table'}, 70: {'id': 70, 'name': 'toilet'}, 72: {'id': 72,
'name': 'tv'}, 73: {'id': 73, 'name': 'laptop'}, 74: {'id': 74, 'name':
'mouse'}, 75: {'id': 75, 'name': 'remote'}, 76: {'id': 76, 'name':
'keyboard'}, 77: {'id': 77, 'name': 'cell phone'}, 78: {'id': 78, 'name':
'microwave'}, 79: {'id': 79, 'name': 'oven'}, 80: {'id': 80, 'name':
'toaster'}, 81: {'id': 81, 'name': 'sink'}, 82: {'id': 82, 'name':

```

```
'refrigerator'}, 84: {'id': 84, 'name': 'book'}, 85: {'id': 85, 'name':  
'clock'}, 86: {'id': 86, 'name': 'vase'}, 87: {'id': 87, 'name': 'scissors'},  
88: {'id': 88, 'name': 'teddy bear'}, 89: {'id': 89, 'name': 'hair drier'},  
90: {'id': 90, 'name': 'toothbrush'}}
```

```
#open("yolo-coco/coco.names").read().strip().split("\n")
```