Blind's Eye : IoT based real-time surrounding identification and object detection

A project report submitted for the partial fulfilment of the

Bachelor of Technology Degree in

Computer Science & Engineering under

Maulana Abul Kalam Azad University of Technology by

Sujoy Seal

Roll No: 10400117056, Registration Number: 171040110167

&

Himanshu Shekhar

Roll No: 10400117160, Registration Number: 171040110063

Academic Session: 2017-2021

Under the Supervision of

Prof. Tufan Saha

Department of Computer Science and Engineering

Institute of Engineering & Management

Y-12, Salt Lake, Sector 5, Kolkata, Pin 700091, West Bengal, India

Affiliated To

Maulana Abul Kalam Azad University of Technology, West Bengal

formerly known as **West Bengal University of Technology**

In Pursuit of Knowledge and Excellence

Maulana Abul Kalam Azad University of Technology

BF 142, BF Block, Sector 1, Kolkata, West Bengal 700064

June 2021

INSTITUTE
OF ENGINEERING & MANAGEMENT
श्रद्धावान लभते ज्ञानम् Salt Lake Electronics Complex, Kolkata - 700091, WB, INDIA

Phone : (033) 2357 2969/2059/2995
(033) 2357 8189/8908/5389
Fax : 91 33 2357 8302
E-mail : director@iemcal.com
Website : www.iemcal.com

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that the project report titled **"Blind's Eye : IoT based real-time surrounding identification and object detection"**, submitted by **Sujoy Seal**, Roll No: **10400117056** Registration Number: **171040110167** and **Himanshu Shekhar**, Roll No: **10400117160**, Registration Number: **171040110063**, students of Institute of Engineering & Management in partial fulfilment of requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering, is a bonafide work carried out under the supervision of Prof. Tufan Saha during the final year of the academic session of 2017-2021. The content of this report has not been submitted to any other university or institute for the award of any other degree.

It is further certified that the work is entirely original and the performance has been found to be satisfactory.

Prof. Tufan Saha
Assistant Professor
Department of Computer Science & Engineering
Institute of Engineering & Management

Prof.(Dr.) Mohua Chakraborty
H.O.D.
Department of Computer Science & Engineering
Institute of Engineering & Management

Principal
Institute of Engineering & Management

INSTITUTE
OF ENGINEERING & MANAGEMENT
श्रद्धावान लभते ज्ञानम्
Salt Lake Electronics Complex, Kolkata - 700091, WB, INDIA

Phone : (033) 2357 2969/2059/2995
(033) 2357 8189/8908/5389
Fax : 91 33 2357 8302
E-mail : director@iemcal.com
Website : www.iemcal.com

DECLARATION
FOR NON-COMMITMENT OF PLAGIARISM

We, **Sujoy Seal** and **Himanshu Shekhar**, students of B.Tech in the Department of Computer Science & Engineering, Institute of Engineering & Management, Kolkata, have submitted the project report in partial fulfilment of the requirements to obtain the above noted degree. We declare that we have not committed plagiarism in any form or violated copyright while writing the report and have acknowledged the sources and/or the credit of other authors wherever applicable. If subsequently it is found that we have committed plagiarism or violated copyright, then the authority has full right to cancel/reject/revoke our degree.

Name of the Student: SUJOY SEAL

Full Signature: Sujoy Seal

Name of the Student: HIMANSHU SHEKHAR

Full Signature: Himanshu Shekhar

Date: 20/06/2021

Abstract

Vision is one of the most 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 report 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 features extraction concept using Convolutional Neural Network layers followed by fully connected layers. The simulation results using YOLO v3 algorithm and key-points matching showed good accuracy for detecting objects. Thus, our contribution is to present the idea of a visual substitution system based on features extractions and matching to recognize and locate objects in images. For, the purpose of user tracking we are sending the person's live environment and demographic data to his / her trusted contacts[1].

Acknowledgements

We must not forget to acknowledge everyone who has provided constant support to us during our B.Tech course. First and foremost, we would like to express sincere gratitude to our supervisor **Prof. Tufan Saha** for his continuous support and motivation in fuelling the pursuance of carrying out this project endeavour. Without his guidance and persistent encouragement, this project work would not have been possible. He has been a tremendous mentor for us throughout this academic journey. Many of his academic advises about our career growth have been priceless.

We would like to convey sincere gratitude to **Prof. Mohua Chakraborty** for providing us constant inspiration to stand firm against several setbacks throughout the course. Additionally, we would like to thank all the technical, non-technical and office staffs of our department for extending facilitating cooperation wherever required. We also express gratitude to all of our friends in the department for providing the friendly environment to work on the project work. We would also like to thank our Director **Prof. Satyajit Chakraborti** for providing us an outstanding platform in order to develop our academic career. In addition, we also preserve a very special thankful feeling about our Principal **Prof. Indranil Mukherjee** for being a constant source of inspiration.

A special thank is due to our family. Words cannot express how grateful we are to our parents for all the sacrifices that they have made while giving us necessary strength to stand on our own feet.

Finally, we would like to thank everybody who has provided assistance, in whatever little form, towards successful realization of this project.

Keywords

Keywords	Descriptions
YOLO	You Only Look Once, a deep neural network machine learning model for object detection with great accuracy even having large FPS videos.
Object Detection	Identifying objects like humans do
CNN	Convolutional Neural Network
Visually Impaired	A visually impaired person can not see objects properly due to some defects in their eye
TTS	Text-to-Speech which converts written text into speech format.
Cloud	An online platform where we can upload and from where we can retrieve datas. In this project we're using MongoDB cloud database.
Navigation	Moving from one place to another by taking help from someone/ something
NMS	Non-Max Suppression function selects a single entity out of many overlapping entities

List of Figures

Fig. 1. The VISION-800 smart glass

Fig. 2. The smart glasses for the blind

Fig. 3. A block diagram showing the process for providing surrounding information to the user

Fig. 4. People crossing road

Fig. 5. The 3 Binary Large Object (BLOB) images

Fig. 6. Output of Fig. 4

Fig. 7. A block diagram depicting security aspect embedded in the device

Fig. 8. Comparison of different weights over MS COCO Dataset

Fig. 9. YOLO versus other different object detection models in GeForce RTX 1080 Ti system

Fig. 11. Object identification in a traffic image

Fig. 12. Object identification in an indoor image

Fig. 13. Screenshot of real-time object identification in a video streaming from a laptop webcam

Fig. 14. JSON data received at command prompt from the cloud

Fig. 15. Real-time data stored in MongoDB Atlas

Fig. 16. Generating Maps from Cloud data using Charts.js to detect the location of the user on Google Maps

List of Tables

Table. 1. Performance of different models in GeForce RTX 2080 Ti GPU

Table. 2. Percentage comparison between YOLO v.3 and YOLO v.3-tiny

Contents

1. Abstract 4

2. Acknowledgement.....5

3. Keywords.....6

4. List of Figures 7

5. List of Tables 8

6. Introduction 11-12

7. Related Work.....13-16

8. Basic Underlying Concept 17-19

9. Proposed Approach 20-25

10. Source Code 26-32

11. Graph comparison of YOLO and other object identification models 33-34

12. Outputs 35

12.1. Object Identification with confidence score 35

12.1.1. Identification in still images 35

12.1.2. Identification in Videos 36

12.1.3. Sending Data to Trusted Contacts 36

12.2. Data Analytics and usage 37

12.2.1. Comparison of YOLO v.3 and YOLO v.3-tiny 37

12.2.2. Data at Cloud 37

13. Application of the data stored 38

14. Conclusion 39

15. References 40-42

Introduction

According to the World Health Organization (WHO), there are approximately 285 million people who are visual impairments, 39 million of them are blind[2] and 246 million have a decrease of Visual acuity. Almost 90% who are visually impaired are living in low-income countries[3]. In this context, Tunisia has identified 30,000 people with visual impairments; including 13.3% of them are blind. These Visual impairment present severe consequences on certain capabilities related to visual function: (a) The daily living activities (that require a vision at a medium distance), (b) Communication, reading, writing (which requires a vision closely and average distance), (c) Evaluation of space and the displacement (which require a vision far), (d) The pursuit of an activity requiring prolonged maintenance of visual attention[4]. In the computer vision community, developing visual aids for handicapped persons is one of the most active research projects. Mobility aids are intended to describe the environment close to the person with an appreciation of the surrounding objects. These aids are essential for fine navigation in an environment described in a coordinate system relative to the user. In this paper, we present an overview of vision substitution modalities and their functionalities. Then, we introduce our proposed system and the experiments tests[4].

Related works show that visual substitution devices accept input from the user's surroundings, decipher it to extract information about entities in the user's environment, and then transmit that information to the subject via auditory or tactile means or some combination of these two. Among the various technologies used for blind people, the majority is aids of mobility and obstacle detection[5,6] .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[7] .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 objects (e.g. whether it is a table or chair) 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. Among the problems in object identification, we note the redundancy of objects under different conditions: the change of viewpoint, the change of illumination and the change of size. We have the concept of intra-class variability (e.g. there are many types of chairs) and the inter-class similarity (e.g. television and computer). 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.

Related work

Here are some examples of how smart technology can be a game-changer, allowing everyone to interact with the world in new ways:

The eye in AI

Microsoft's Seeing AI is an app designed to help people with low vision or who are blind[7]. It enhances the world around the user with rich audio descriptions. It can read a handwritten note or scan a barcode and then tell the user what the product is. Point a camera at something and the app will describe how many people it can see and where they are in the image – center, top left and so on.

For a sighted person, walking along the street can mean taking in every detail that surrounds them. Microsoft Soundscape replicates that behaviour by building a detailed audio map[8] that relates what's taking place around a person with visual impairment. It creates layers of context and detail by drawing on location data, sound beacons and synthesized 3D stereo sound to build a constantly updating 3D sound map of the surrounding world.

Braille has been used for nearly 200 years as a tactile way of reading with fingertips. It has now jumped from the page to the screen with the updated version of Narrator[9], the screen-reader for Microsoft Windows, supporting digital Braille displays and keyboards. Outside of Microsoft's efforts, Braille touchscreens that work in the same way as tablets have already proved popular among students and teachers. At the Assistive Technology Industry Association's 2019 conference in Orlando, Florida, innovations on display included the BraiBook, a Braille e-reader that fits into the

palm of a hand, and even an electronic toy called the Braille Buzz, designed to teach Braille to preschoolers.

The vOICe

Fig. 1. The VISION-800 smart glass[19]

The vOICe for Android sensory substitution app for the blind nowadays runs on various types of smart glasses. Just like other smart glasses, the VISION-800 smart glasses form a standalone device that independently runs android apps inside the device. They are not a peripheral for a smartphone. The goal of the usage notes is to minimize the need for sighted assistance. The VISION-800 smart glasses come with the Google TalkBack screen reader (part of the Android Accessibility Suite), Google text-to-speech and Google Play pre-installed. If you are blind then you probably need only one-time sighted assistance to activate the Google TalkBack screen reader, and should be able to do everything else independently. The self-contained glasses include two mono earbuds, one for each leg, that together give stereo.[19]

OrCam MyEye

OrCam MyEye is a revolutionary voice activated device that attaches to virtually any glasses. It can instantly read to you text from a book, smartphone screen or any other surface, recognize faces, help you shop on your own, work more efficiently, and live a more independent life! OrCam MyEye conveys visual information audibly, in real-time and offline.[20]

Eyesynth

Eyesynth is an audiovisual system for the blind. It consists of a pair of glasses connected to a micro computer. The system records the surrounding environment in three dimensions. Then, the collected data is converted into understandable audio for the blind. The glasses send images to the micro-computer which processes them in real time, so the user receives instant feedback. The environment is captured and processed in full 3D. This means we have information about depth as well. Then, the computer converts all of that information into a series of sounds that interprets open spaces, shapes and obstacles.[21]

Fig. 2. The smart glasses for the blind[21]

Beacons of change

Bluetooth beacons, such as those being used by the company Foresight Augmented Reality[15], act like highly precise, personalized guides for people who are blind or partially sighted. While basic GPS technology can take users to a location, beacons mounted in a store, restaurant or public building can guide them to the entrance of the building in question. And when the user is inside, other beacons can direct them to the bathroom or other important facilities.

Smart Glasses

Researchers at Ajman University in the United Arab Emirates are working on the development of a set of smart glasses that can use AI to read[16], provide navigation information and potentially identify faces. Glasses are connected to a smartphone through a processing unit, allowing the system to function without an internet connection.

These smart glasses are still in the early stages of development but are said to work with a reading accuracy rate of 95%.

AI for Accessibility

Microsoft's AI for Accessibility program was launched last year, with a $25 million commitment to put Microsoft technology in the hands of start-ups, developers, researchers and non-profits in order to drive innovation and amplify human capability for people with disabilities. The program is continuously looking at new projects to support[10].

Basic underlying concepts

The first working step is to first capture an image or a video (in real time) through a sophisticated high quality camera. The choice of camera becomes a matter of paramount importance since better the quality of image, better is the input fed to the ML model. Our YOLO model uses CNN which relies on procedures such as flattening, Pooling which finds feature maps (or patterns) in the test inputs. So, a very good camera ensures that the output obtained has very high accuracy. Since, accuracy and speed are quite significant given the base case that this device is for disabled people, a highly sophisticated software in terms of power and user-friendliness, becomes indispensable. The immediate concept is to include this image in BlindsEye_img.py file and run it on any suited Python IDLE (Spyder3, Jupyter Notebook, Anaconda). Before running, ensure to have libraries and dependencies preinstalled.

Quick, exact calculations for object detection would permit computer to drive vehicles without particular sensors, empower assistive gadgets to pass on constant scene data to human clients, and open the potential for universally useful, responsive automated frameworks[11]. Object discovery includes identifying locale of interest of object from given class of picture[12]. There are basically two algorithms for object discovery and they can be arranged into two kinds:

1. Classification-dependent algorithms are performed in two steps. First, they define and select areas of significance for an image. Second, these regions are organized into convolutional neural networks. The above-mentioned arrangement is mild, since it is required to make estimates for all chosen regions. A commonly recognized case of this type of algorithm is the Regional Convolutional Neural Network (RCNN) and Medium RCNN, Faster RCNN, and the most recent: Mask RCNN[12].

2. Algorithms based on regression rather than selecting a field of interest for an image, they estimate groups and bounding boxes for the whole picture in one run of the algorithm. The two most common models in this set are the YOLO family algorithms which provides maximum speed and precision for multiple object detection in a single frame and the SSD this algorithms that are typically used to track objects in real-time.

To understand the YOLO algorithm, it is important to determine what is currently expected. It varies from the majority of the neural network models because it uses a single convolutional network that predicts bounding boxes and the resulting probabilities. The bounding boxes are weighted by the probabilities and the model makes their detection dependent on the final weights. Thus, end-to-end output of the model can be directly maximized and, as a result, images can be produced and processed at a rapid pace[13]. Every bounding box can be represented using four descriptors:

1. Centre of a bounding box
2. Width
3. Height
4. Value c refers to an object class

The pc value also needs to be predicted, that indicates the likelihood that there is an object in the bounding box.[14]

After the predictions are made, we have to convert this obtained text input to sound. Imagine how can a blind person read from a display! Sounds ridiculous, right? Therefore, we decided to have some audio output to the earpiece which becomes like a guiding torch for a blind person. We therefore converted text to speech. For implementing this, we used Google Text-to-Speech API . Once this audio is received, we can have an attenuator (amplifier) which boosts up the amplitude of the signal. Remember that along with sound audibility, we need to ensure that the sound power (in decibels) are safe for practical use because people are expected to wear them for long hours. We also retrieved the IP addresses, the geographic location, and real time data and decided to keep them . For efficient

storage we used MongoDB Atlas Database and created our collection for document storage. We used pymongo library in order to transfer our identified data to this cloud storage in real time. We also incorporated an efficient retrieval system through which any number of trusted contacts of this blind person shall be able to track and receive the storage data in real time . This blind person may fall into a danger zone or may be kidnapped. In any such scenarios, the tracked data shall help investigating officers to rescue the person. This is because the tracked data will contain real time address locations, pin code, surrounding description. This data can also be used to generate real time hotspots and maps as to where the person may go next. This tracked data is stored securely and safety of the blind person is very much guaranteed. Also, we have made him a lot more empowered, a lot more stable, secure, work-thirsty and someone who can travel safely amidst any climatic or political conditions irrespective of age, disability or any distinction. This makes our product marketable and contributes to the welfare of the society and the nation.

Proposed Approach

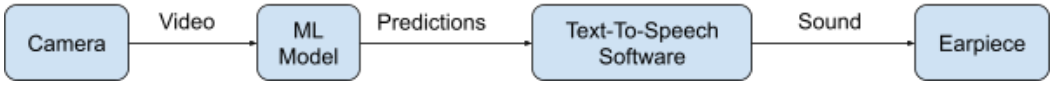

Fig. 3. A block diagram showing the process for providing surrounding information to the user

We need 3 main files

1. yolov3.cfg (Download from here) — configuration file
2. yolov3.weights (Download from here) — pre-trained weights
3. coco.names (Download from here) — 80 classes names

Next, we will define output layers because that's where we will be defining what object is detected by using **net.getUnconnectedOutLayersNames()** and **net.forward().**

Since we'll be using a pre-trained model, we'd have to download certain files. The "weights" file , the "configuration" file, and the "coco-names" file. The weights and the configuration file can be found in this https://pjreddie.com/darknet/yolo/ and the coco-names file can be downloaded/copied from https://github.com/pjreddie/darknet/blob/master/data/coco.names. There are several pre-trained models available and we would be using the "YOLOv3–416" model. The models are trained on the MS COCO dataset which has 80 classes of objects present in it. After downloading all the files, it's time to create and load our model. The names of the objects that our model has been trained to identify is given in the "coco.names" file, which we store in a list called **classes**. We also retrieve the names of the output layers with the help of the **getLayerNames()** and **getUnconnectedOutLayersNames()** function and store them too in output_layers list.

Fig. 4. People crossing road

This is our original image from which we want to detect as many objects as possible. But we cannot give this image directly to algorithm. So we need to do some conversion from this image. This is called blob conversion which is basically extracting features from image. We will detect objects in blob by using **cv2.dnn.blobFromImage** and passing few variables: **img** is file name, scale-factor of 1/255, size of image to be used in blob be (416,416), no mean subtraction from layers as (0,0,0), setting **True** flag means we will be inverting blue with red since OpenCV uses BGR but we have channels in image as RGB.

```
blob=cv2.dnn.blobFromImage(img,1/255,(416,416),
(0,0,0),swapRB=True,crop=False)
                        #Inverting blue with red, i.e., bgr->rgb
```

Now let's see how the 3 different blobs looks like by using following code.

```
for b in blob:
   for n, img_blob in enumerate(b):
         cv2.imshow(str(n), img_blob)
```

Fig. 5. The 3 Binary Large Object (BLOB) images

We don't observe much difference but this is what we will input to YOLO algorithm. We now pass this blob to network using **net.setInput(blob)** and then forward this to the **outputlayers.**

```
#We need to pass the img_blob to the algorithm
net.setInput(blob)

output_layers_names=net.getUnconnectedOutLayersNames()
layerOutputs=net.forward(output_layers_names)
```

Here all objects have been detected and **outs** contains all the information we need to instruct to extract the position of the object like top, left, right, bottom positions, name of class. Now let's evaluate **outs** by showing information on screen. Mainly we will be trying to predict the confidence meaning how confident algorithm is when it predict some object. For this, we will loop through **outs,** first get all the **scores** for each **out** in **outs**. Then get the **class_id** that has highest score amongst them and then assign **confidence** to value of **scores** by passing **class_id**. Now we will assign the confidence level threshold as 0.5. Anything above 0.5 should mean object detected. Let also have the **center_x, center_y, w** as width, **h** as height of the object detected. Here we have **height, width** variables we saved previously of original image.

Further lets draw rectangle around detected object by using **center_x, center_y, w, h**. And append some information to that like **class, confidence**[17].

There might be cases that multiple times the same object might be detected. To eliminate this, we will use Non-Max Suppression(NMS) functionality. What this will do is eliminate the boxes by using some threshold value (any box having value less than 0.4 will be removed).

```
#Removes redundant boxes and keeping only having high scores
indexes=cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
```

Now using loop over all found boxes, push all the parameters and create bounding boxes with label and confidence scores.

```
x,y,w,h=boxes[i]
label=str(classes[class_ids[i]])
confidence=str(round(confidences[i],2))
color=colors[i]
#Creating bounding boxes
cv2.rectangle(img,(x, y),(x+w, y+h),color,2)
#Adding label & confidence over bounding boxes
cv2.putText(img, label + " " + confidence, (x, y+20), font, 2,
 (255,255,255), 2)
```

Below is the output we get:

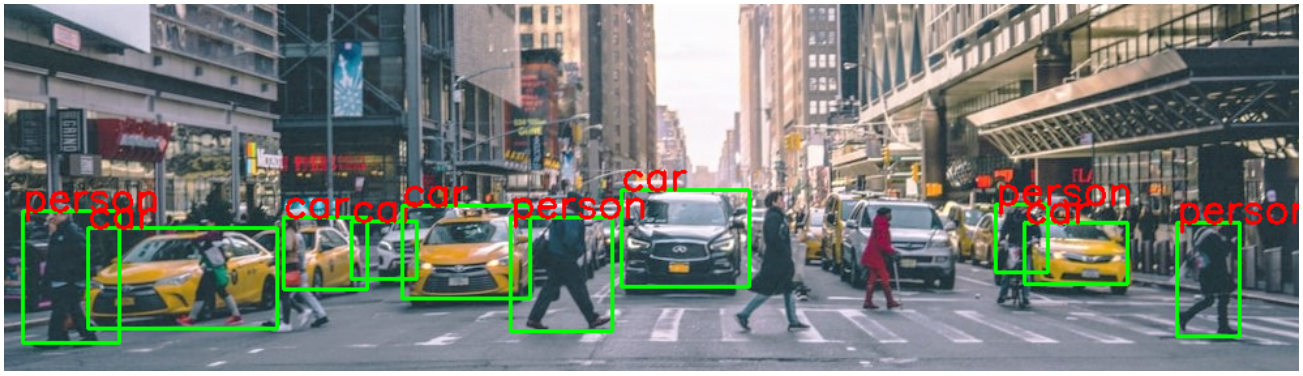

Fig. 6. Output of Fig. 4

For security purpose of the user, we decided to send the current location of the user, IP address of the device, and current timestamp to saved trusted contacts.

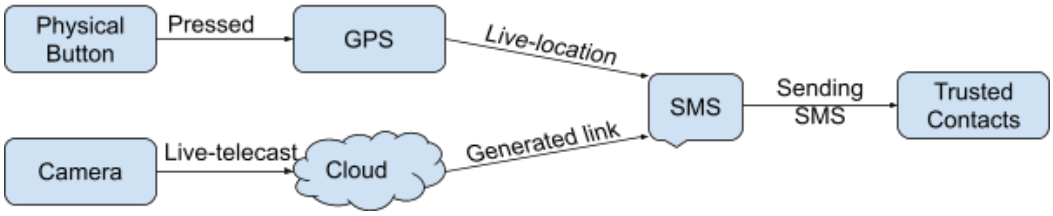

Fig. 7. A block diagram depicting security aspect embedded in the device

To get IP address of the device we are using Socket library.

```
#Getting the hostname
hostname = socket.gethostname()
#Getting the IP address
ip_address = socket.gethostbyname(hostname)
```

And to get current timestamp we use datetime library.

```
#Get current time
now = datetime.now()
timestamp = now.strftime("%Y/%m/%d %H:%M:%S")
return timestamp
```

Getting current location is a bit trickier. At first we need to find current location in co-ordinates, i.e., latitude & longitude using geocoder library. Then using geopy library we get the location in "city, district, state, pin code and country" format. Let's see how we did.

```
#Get your coordinates
g=geocoder.ip('me')
latitude=g.lat
longitude=g.lng
#Build coordinates string to pass to reverse() function
coordinates = f"{latitude}, {longitude}”
try:
    return app.reverse(coordinates, language=en).raw
except:
    return get_address_by_location(latitude, longitude)
```

We need to send required datas to trusted contacts. For this we need to first upload it to the cloud then retrieve from there into the trusted contacts device. Here we are using MongoDB cloud for storing the datas. Below code snippet shows our implementation.

```
collection = connect_to_cloud()

results = collection.find({}, {"_id" : 0})
for x in results:

    print(x)
```

While extracting from cloud we are excluding “_id” because it is redundant.

As we know that our users are blind, hence we need to provide output in audio format. Hence we are using Google Text-to-Speech API for converting output into audio format.

```
#Converting text to speech
tts = gTTS(text=label, lang='en', slow=False)
#Saving the converted audio file into the same directory
tts("Outputs/output.mp3")
#Play the converted file
AudioPlayer(“Outputs/output.mp3").play(block=True)
```

Source Code

Find full code at : https://github.com/Pandit98himanshu/BlindsEye_IoT/tree/master/src

1. BlindsEye_vid.py

```
import cv2
from geopy.geocoders import Nominatim
from get_address import *
from upload_to_cloud import *
from get_datetime import *
from get_ipaddress import *
from create_bounding_boxes import *
from text_to_speech import *
from extract_info import *
from connect_to_cloud import *
from load_all_object_names import *

#Connecting to Mongodb cloud
collection=connect_to_cloud()

#Load YOLO Algorithm
net=cv2.dnn.readNet("yolov3.weights","yolov3.cfg")

#To load all objects that have to be detected
classes=load_all_object_names()

#Loading the Video
cap=cv2.VideoCapture(0)   #openCV considers "0" as webcam

#Colors of bounding boxes
colors=np.random.uniform(0, 255, size=(len(classes), 3))

while True:
    #Extract info from raw image frame
```

```
    img, height, width, layerOutputs = extract_info_from_image
(cap, net)

    boxes=[]            #bounding boxes
    confidences=[]      #confidences
    class_ids=[]        #predicted classes

    #Extract all informations from identified objects
    extract_info_from_layers(layerOutputs, boxes, confidences,
 class_ids, height, width)

    #Removes redundant boxes and keeping only boxes with high
scores
    indexes=cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    #Pass all parameters to show on the output video
    if len(indexes)>0:
        for i in indexes.flatten():
            #Creating bounding boxes
             label, confidence = create_bounding_boxes(i, img,
boxes, classes, class_ids, confidences, colors)

            #Convert identified objects into speech format and
 play
            text_to_speech(label)

            #Get timestamp
            timestamp=get_datetime()

            #Get the address info
            address=get_address()

            #Get IP Address
            ip_address=get_ipaddress()

            #Uploading identified objects to the cloud
            upload_to_cloud(collection, ip_address, timestamp,
 label, confidence, address['display_name'])


    cv2.imshow("Output",img)
    key=cv2.waitKey(1)
    #Press esc key to stop the program
```

```
    if key==27:
        break

#Release camera
cap.release()
#Close all output windows
cv2.destroyAllWindows()
```

2. connect_to_cloud.py

```
import pymongo
from pymongo import MongoClient

def connect_to_cloud():
    #Connecting to Mongodb cloud
    cluster=MongoClient("mongodb+srv://
SujoySeal:bKWb09V1uBbMMhJb@post-boxcluster-
sujoy.i9ean.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority")
    #Assigning database
    db=cluster["FYP-Database"]
    #Assigning collection
    collection=db["Identified-Objects"]
    return collection
```

3. create_boundingboxes.py

```
import cv2

#Font of parameters
font=cv2.FONT_HERSHEY_PLAIN

def create_bounding_boxes(i, img, boxes, classes, class_ids, c
onfidences, colors):
    x,y,w,h=boxes[i]
    label=str(classes[class_ids[i]])
    confidence=str(round(confidences[i],2))
```

```
    color=colors[i]
    #Creating bounding boxes
    cv2.rectangle(img,(x,y),(x+w, y+h),color,2)
    #Adding label & confidence over bounding boxes
    cv2.putText(img, label + " " + confidence, (x, y+20), font
, 2, (255,255,255), 2)
    return label, confidence
```

4. extract_info.py

```
import cv2
import numpy as np

def extract_info_from_image(cap, net):
    #Capture each frames of the video file
    _, img=cap.read()
    #Capturing its height and width used to scale back to orig
inal file
    height,width,_=img.shape

    #Extracting features to detect objects
          blob=cv2.dnn.blobFromImage(img,1/255,(416,416),
(0,0,0),swapRB=True,crop=False)
                                    #Inverting blue with red
                                    #bgr->rgb

    #We need to pass the img_blob to the algorithm
    net.setInput(blob)

    output_layers_names=net.getUnconnectedOutLayersNames()
    layerOutputs=net.forward(output_layers_names)

    return img, height, width, layerOutputs


def extract_info_from_layers(layerOutputs, boxes, confidences,
 class_ids, height, width):
    #Extract all informations form the layers output
    for output in layerOutputs:
        #Extract information from each of the identified objec
ts
```

```
        for detection in output:
            #Should contain 4 bounding boxes, or 85 parameters
            scores=detection[5:]          #Get score after 5th
            #Get index having maximum scores
            class_id=np.argmax(scores)
            confidence=scores[class_id]
             #If confidence is strong enough, get locations of
those bounding boxes
            if confidence>0.5:
                center_x=int(detection[0]*width)
                center_y=int(detection[1]*height)
                w=int(detection[2]*width)
                h=int(detection[3]*height)

                x=int(center_x-w/2)
                y=int(center_y-h/2)

                boxes.append([x, y, w, h])
                confidences.append((float(confidence)))
                class_ids.append(class_id)
```

5. get_address.py

```
from geopy.geocoders import Nominatim
import time
import geocoder

#Instantiate a new Nominatim client
app = Nominatim(user_agent="BlindsEye")

def get_address(language="en"):
    #Get your coordinates
    g=geocoder.ip('me')
    latitude=g.lat
    longitude=g.lng
    #Build coordinates string to pass to reverse() function
    coordinates = f"{latitude}, {longitude}"
    try:
        return app.reverse(coordinates, language=language).raw
    except:
```

```
        return get_address_by_location(latitude, longitude)
```

6. get_data_from_cloud.py

```
import pymongo
from pymongo import MongoClient
from connect_to_cloud import *

collection = connect_to_cloud()

results = collection.find({}, {"_id" : 0})
for x in results:

    print(x)
```

7. get_datetime.py

```
from datetime import datetime

def get_datetime():
   #Get current time
   now = datetime.now()
   timestamp = now.strftime("%Y/%m/%d %H:%M:%S”)
   return timestamp
```

8. get_ipaddress.py

```
import socket

def get_ipaddress():
    #Getting the hostname
    hostname = socket.gethostname()
    #Getting the IP address
    ip_address = socket.gethostbyname(hostname)
    return ip_address
```

10. load_all_objects.py

```
#Load all object names which we have to be detected
def load_all_object_names():
    classes=[]
    with open("coco.names","r") as f:
        classes=f.read().splitlines()
        return classes
```

11. text_to_speech.py

```
from gtts import gTTS
from audioplayer import AudioPlayer

#Text to speech using Google Text-to-speech API
def text_to_speech(label):
    #Converting text to speech
    tts = gTTS(text=label, lang='en', slow=False)
    #Saving the converted audio file into the same directory
    tts("Outputs/output.mp3")
    #Play the converted file
    AudioPlayer("Outputs/output.mp3").play(block=True)
```

12. upload_to_cloud.py

```
def upload_to_cloud(collection, ip_address, timestamp, label,
confidence, address):
    #Uploading parameters to the Mongodb cloud
        post = { "ip_address" :   ip_address, "timestamp" :
timestamp, "label" :   label, "confidence" :   confidence,
"address" :  address }
    collection.insert_one(post)
```

Graph comparison of YOLO and other object identification models

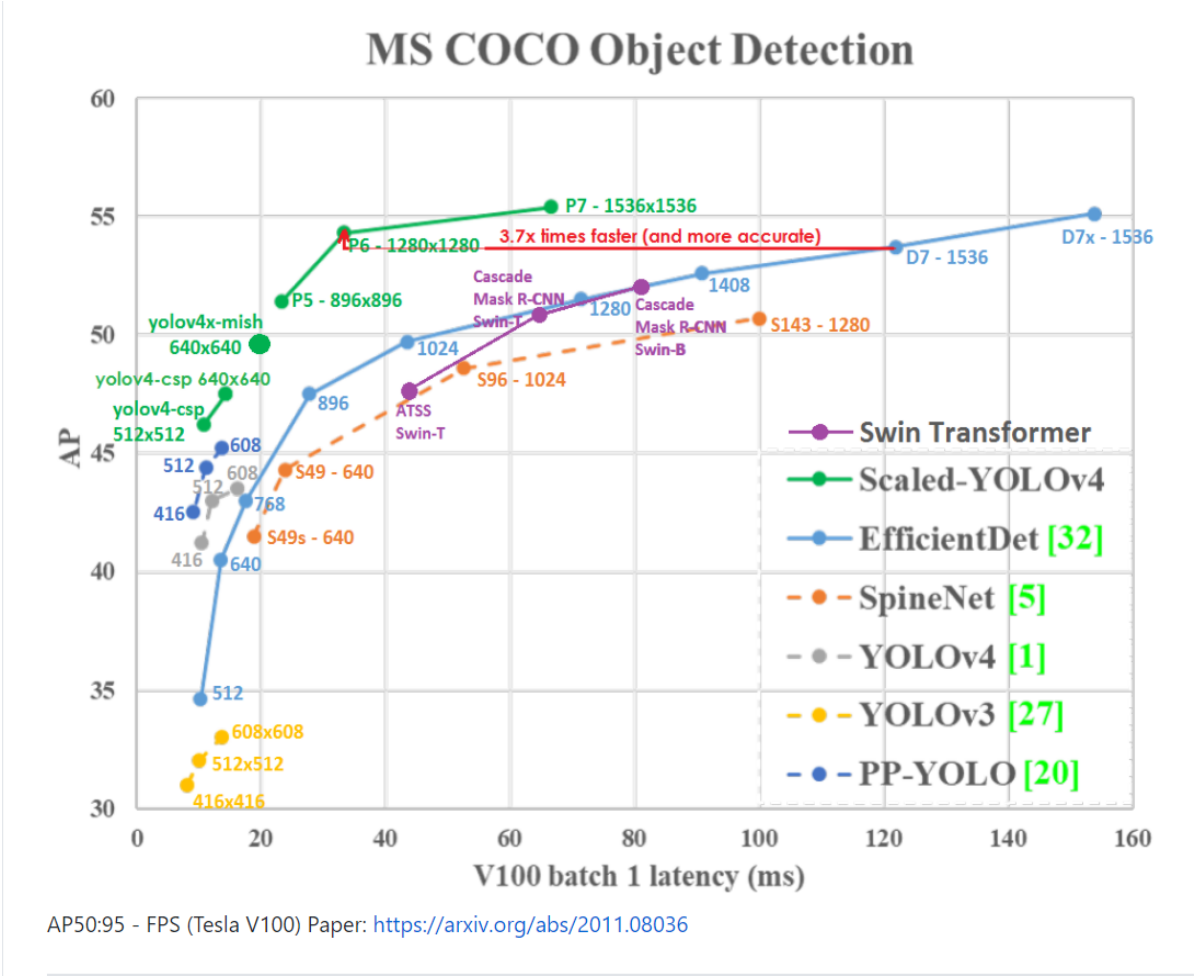

Fig. 8. Comparison of different weights over MS COCO Dataset[18]

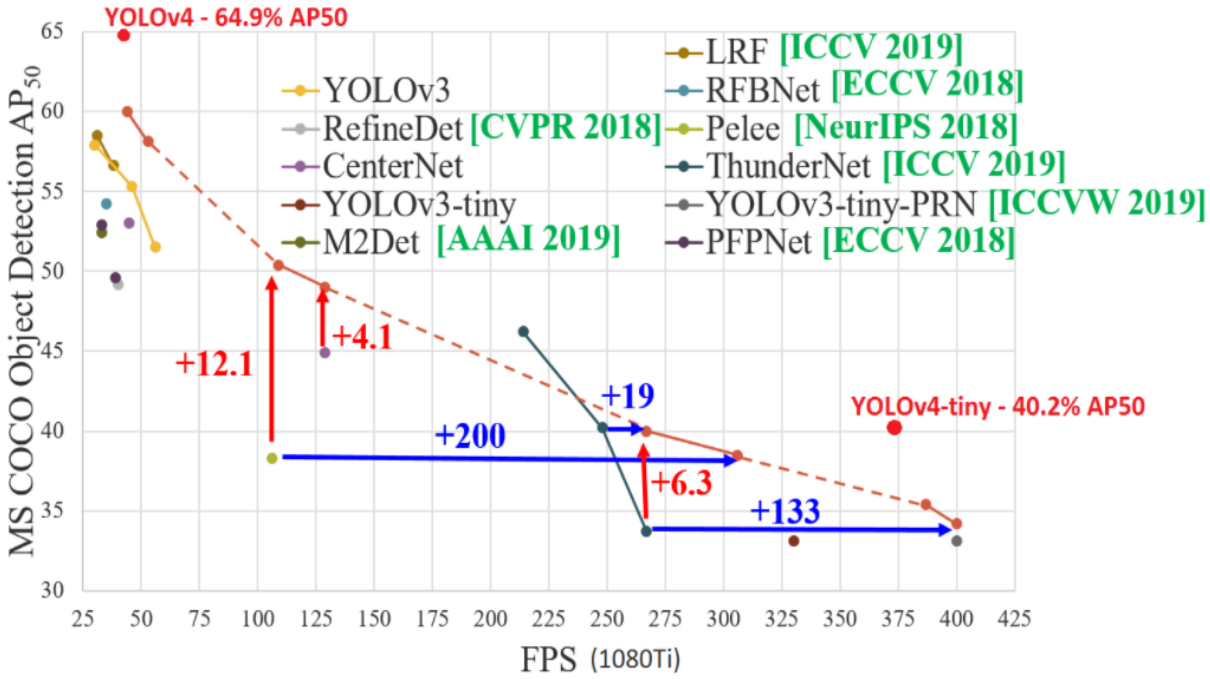

Fig. 9. YOLO versus other different object detection models in GeForce RTX 1080 Ti system[18]

GeForce RTX 2080 Ti:

Network Size	Darknet, FPS (avg)	tkDNN TensorRT FP32, FPS	tkDNN TensorRT FP16, FPS	OpenCV FP16, FPS	tkDNN TensorRT FP16 batch=4, FPS	OpenCV FP16 batch=4, FPS	tkDNN Speedup
320	100	116	**202**	183	423	**430**	**4.3x**
416	82	103	**162**	159	284	**294**	**3.6x**
512	69	91	134	**138**	206	**216**	**3.1x**
608	53	62	103	**115**	150	**150**	**2.8x**
Tiny 416	443	609	**790**	773	**1774**	1353	**3.5x**
Tiny 416 CPU Core i7 7700HQ	3.4	-	-	42	-	39	**12x**

Table. 1. Performance of different models in GeForce RTX 2080 Ti GPU[18]

Outputs

1. Object Identification with confidence score

1.1. Identification in still images

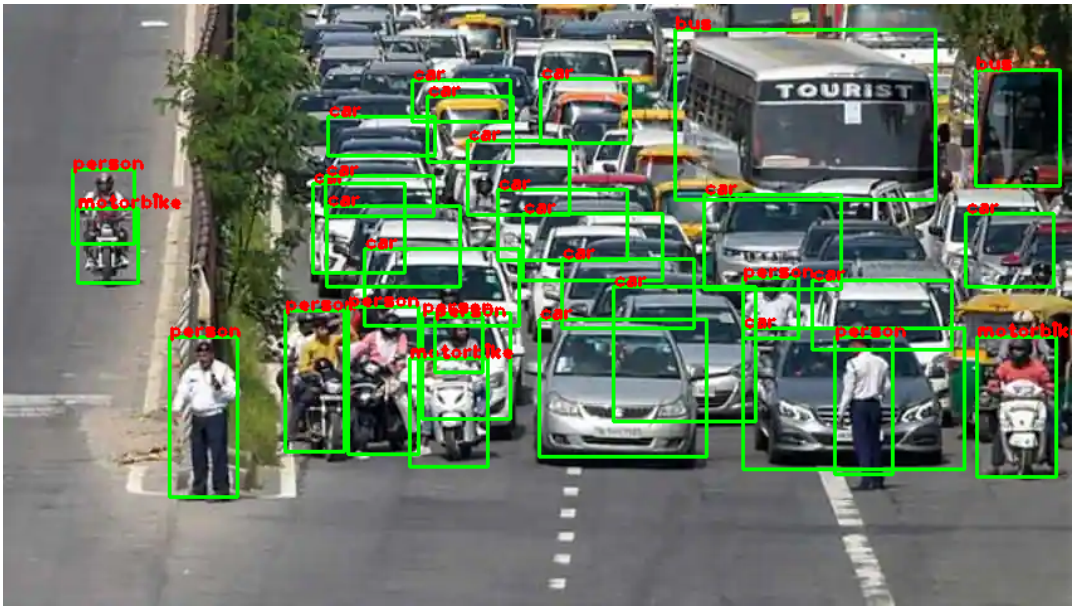

Fig. 11. Object identification in a traffic image

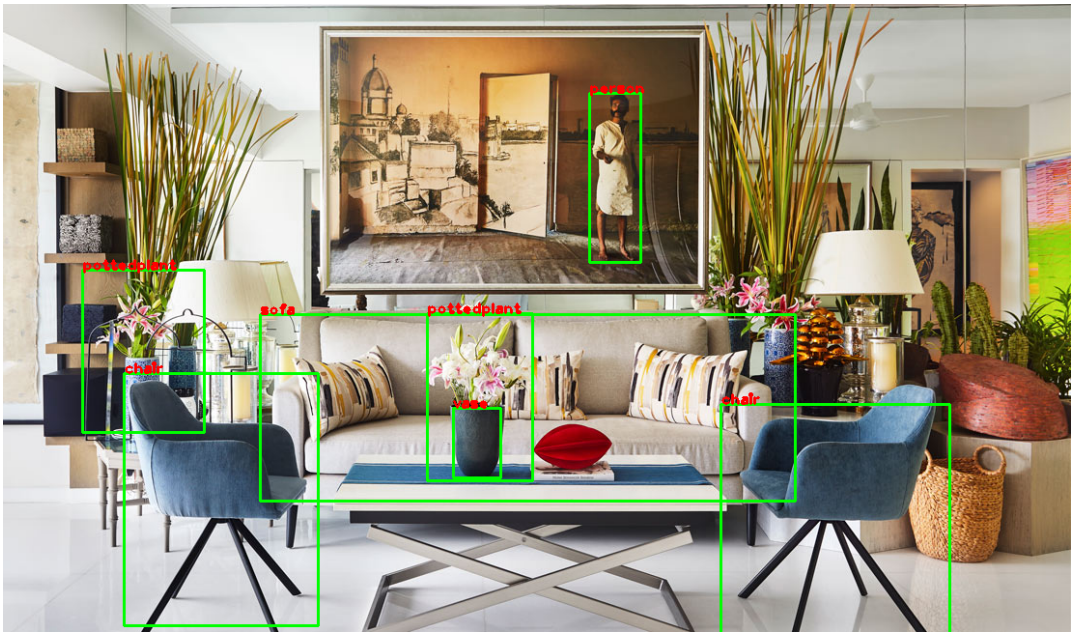

Fig. 12. Object identification in an indoor image

1.2. Identification in Videos

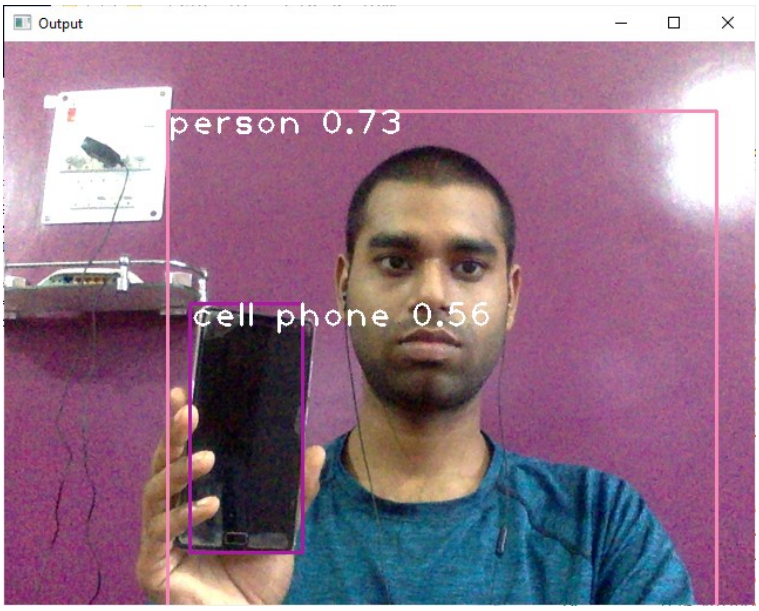

Fig. 13. Screenshot of real-time object identification in a video streaming from a laptop webcam

1.3. Sending Data to Trusted Contacts

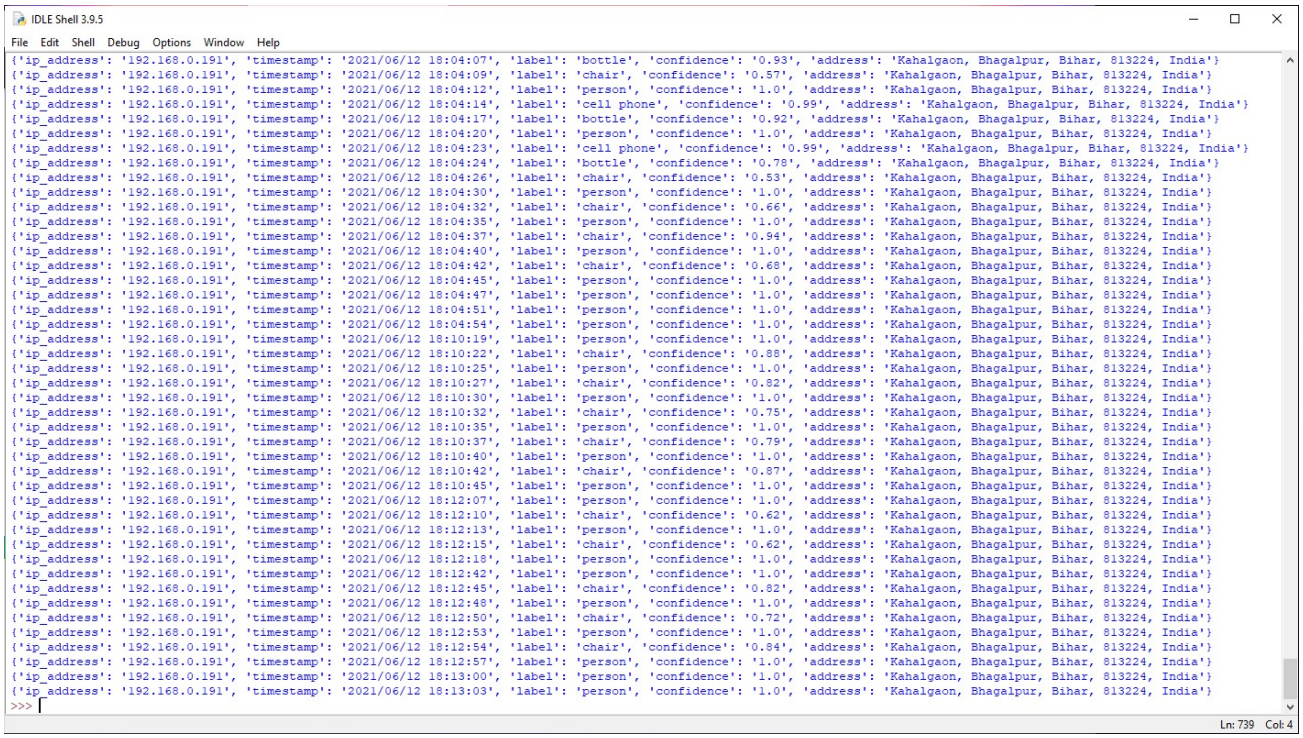

Fig. 14. JSON data received at command prompt from the cloud

2. Data Analytics and Usage

2.1. Comparison of YOLO v.3 and YOLO v.3-tiny

TABLE III. YOLO V.3 AND TINY YOLO V.3 PERFORMANCE

Size	mAP	IoU	FPS
YOLO v.3			
608 × 608	90.91%	85.44%	13
416 × 416	90.73%	84.56%	23
320 × 320	90.64%	81.83%	34
224 × 224	81.72%	78.32%	51
Tiny YOLO v.3			
608 × 608	89.41%	80.18%	41
416 × 416	90.30%	79.75%	68
320 × 320	81.62%	78.83%	92
224 × 224	71.94%	78.46%	105

Table. 2. Percentage comparison between YOLO v.3 and YOLO v.3-tiny[18]

2.2. Data at Cloud

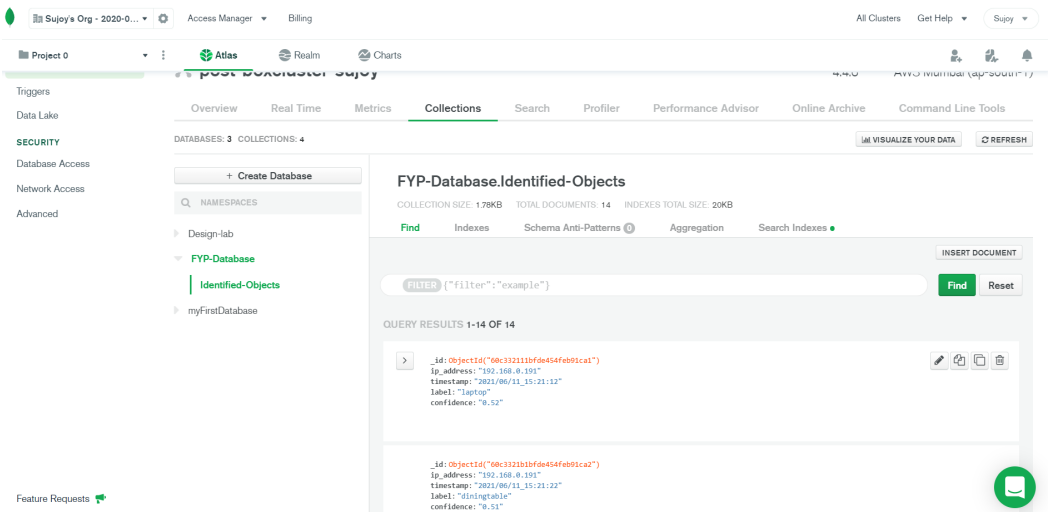

Fig. 15. Real-time data stored in MongoDB Atlas

2.3. Application of the data stored

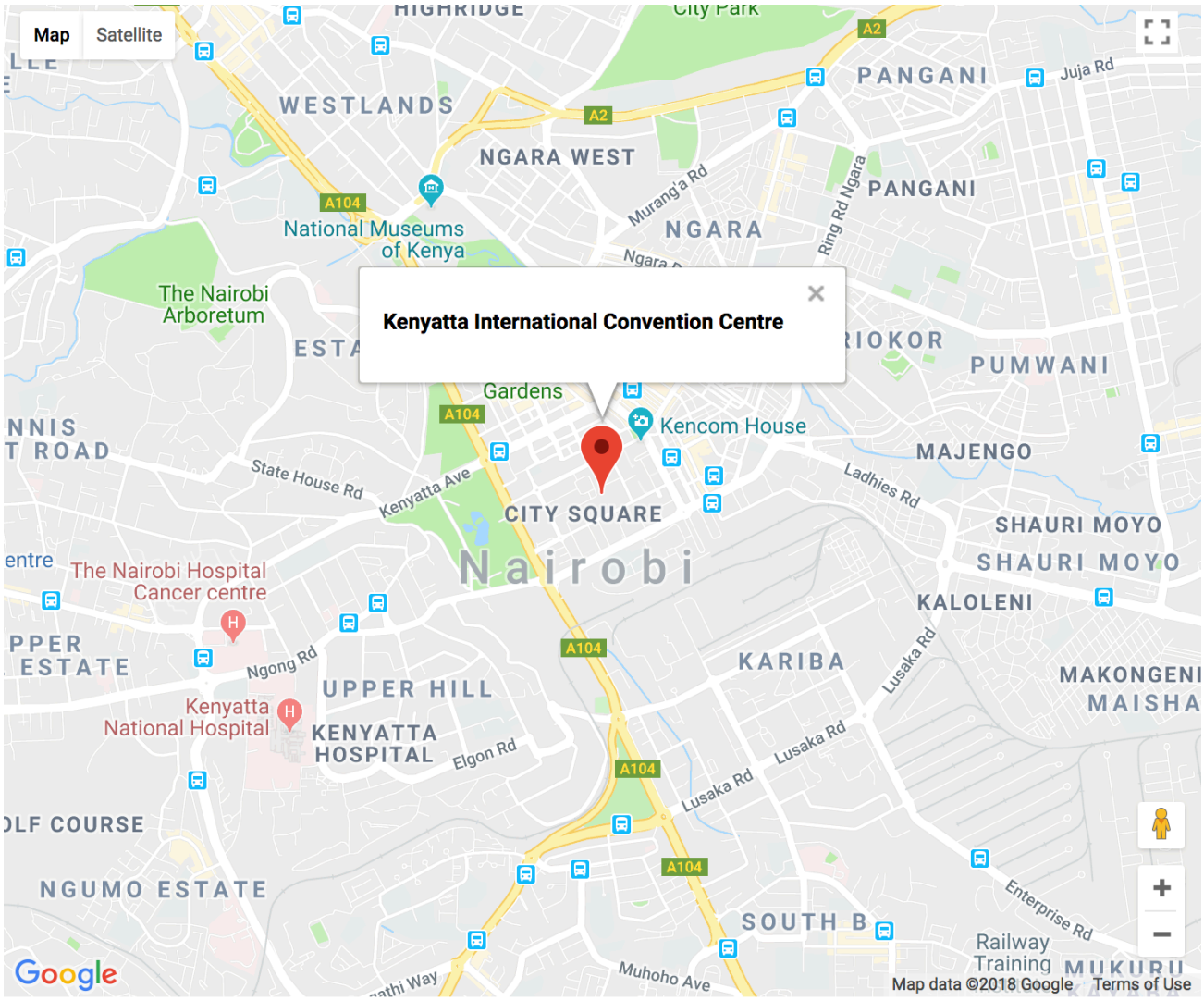

Fig. 16. Generating Maps from Cloud data using Charts.js to detect the location of the user on Google Maps[21]

Conclusion

We get the required output, that is, our model successfully identifies surrounding objects quickly, which is being captured by camera embedded into the device (here we are using laptop webcam), and also uploading necessary information like IP address of the device, current local time, identified objects with confidence score of identification and current address of the user to the cloud. We can successfully fetch those datas from the cloud and displays it on the screen on the trusted contacts. Our idea is not unique but it is completely ours. We realized this after doing some research work on this topic, as you can see our research in Related works section. Here we are using YOLO v3 pre-trained model for object detection because it is a powerful model for object detection which is known for fast detection and most accurate prediction. The complete code can be found on GitHub. Additionally, I would like to give a big shout out to Huynh Ngoc Anh and Jason Brownlee throughout my journey to dive deeply into understanding how YOLO v3 works and implement this into python. There are lots of future scope in this project, like face detection and storing it in database for future use, etc., but for now due to time constraint we have had to end this project here.

We enjoyed working in this project and we get a lot to learn. Hope we will continue this project in near future and move this to the next level with more user-friendly, advanced features and more reliable.

References

1. Pascolini D, Mariotti SPM. Global estimates of visual impairment: 2010. British Journal Ophthalmology Online First published December 1, 2011 as 10.1136/bjophthalmol-2011-300539.
2. “Two Billion Eyes” TWOBILLIONEYES Foundation, https://www.twobillioneyes.org/the-problem [Last Accessed 28 June 2021]
3. Jabnoun, Hanen, Faouzi Benzarti, and Hamid Amiri. "Object detection and identification for blind people in video scene." *2015 15th International Conference on Intelligent Systems Design and Applications (ISDA)*. IEEE, 2015.
4. Kammoun, Slim, et al. "Navigation and space perception assistance for the visually impaired: The NAVIG project." *Irbm* 33.2 (2012): 182-189.
5. Martnez, B.D.C., Vergara-Villegas, O.O., Snchez, V.G.C., De Jess Ochoa Domnguez, H., and Maynez, L.O., 2011. Visual Perception Substitution by the Auditory Sense. In Beniamino Murgante, Osvaldo Gervasi, Andrs Iglesias, David Taniar, and Bernady O. Apduhan, editors, ICCSA (2), volume 6783 of Lecture Notes in Computer Science, pp. 522–533. Springer.
6. Brian Katz, F.G. et al, 2012. NAVIG: Guidance system for the visually impaired using virtual augmented reality. In Technology and Disability, pp. 163–178.
7. “Seeing AI in new languages” Microsoft, https://www.microsoft.com/en-us/ai/seeing-ai [Last Accessed 28 June 2021]
8. “Microsoft Soundscape” Microsoft, A map delivered in 3D sound, https://www.microsoft.com/en-us/research/product/soundscape/ [Last Accessed 28 June 2021]
9. “Chapter 8: Using Narrator with braille” Microsoft, https://support.microsoft.com/en-us/windows/chapter-8-using-narrator-with-braille-3e5f065b-1c9d-6eb2-ec6d-1d07c9e94b20 [Last Accessed 28 June 2021]

10. Fleming, Seán. “7 smart tech developments for people who are blind or have low vision” On the Issues, Microsoft, 8 Aug 2019, https://news.microsoft.com/on-the-issues/2019/08/08/smart-tech-blind-low-vision/ [Last Accessed 28 June 2021]
11. Redmon, J., and S. Divvala. "Girshic), R., & Farhadi, A.(2016). You only loo) once: Unified, realYtime object detection." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779Y788)*.
12. Chandan, G., Ayush Jain, and Harsh Jain. "Real time object detection and tracking using Deep Learning and OpenCV." *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE, 2018.
13. Kumar, Chethan, and R. Punitha. "YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications." *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020.
14. Hassan, Nurul Iman, et al. "People Detection System Using YOLOv3 Algorithm." 2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE). IEEE, 2020.
15. “Building Smart Cities For the Visually Impaired and Blind” Foresight Augmented Reality, https://www.foresightar.com/ [Last Accessed 28 June 2021]
16. “Ajman University Develops Smart Glasses for people who are blind and Partially Sighted” Cool Blind Tech, 20 June 2019, https://coolblindtech.com/ajman-university-develops-smart-glasses-for-people-who-are-blind-and-partially-sighted/ [Last Accessed 28 June 2021]
17. Adakane, Darshan. “Object Detection with OpenCV-Python using YOLOv3” Analytics Vidhya, Medium, 19 Oct 2019, https://medium.com/analytics-vidhya/object-detection-with-opencv-python-using-yolov3-481f02c6aa35 [Last Accessed 28 June 2021]
18. Redmond, Joseph[pjreddie]. “Darknet”, Github, https://github.com/pjreddie/darknet/ [Last Accessed 28 June 2021]
19. “Usage Notes for The vOICe for Android on Smart Glasses,

functional vision for the blind now available through affordable smart glasses" The vOICe, Seeingwithsound, https://www.seeingwithsound.com/android-glasses.htm [Last Accessed 28 June 2021]

20. "What is Eyesynth?" Eyesynth, https://eyesynth.com/what-is-eyesynth/?lang=en [Last Accessed 28 June 2021]
21. Njeri, Rachael. "How to Integrate the Google Maps API into React Applications", DigitalOcean Community, DigitalOcean, Last Validated on 11 September 2020, https://www.digitalocean.com/community/tutorials/how-to-integrate-the-google-maps-api-into-react-applications [Last Accessed 28 June 2021]