Crowd Detection System Using Raspberry Pi

Garapati.Dinesh, Tunga Yogeswara, Priyanka Grover Arora, Pasivedala Lakshmi Srinivas

Electronics and Communication Engineering, Lovely professional University, Punjab

Abstract- In this COVID19 scenario many do not have permission to go out freely and buy, they have been provided a particular time to go out and buy things. So in these particular hours they come out in huge crowds and do not maintain social distancing. Managing this crowd to maintain social distance is huge task for government, so they require a system which will help them to control the people to maintain social distance. In our study, we develop a device using Raspberry Pi-4 board in the region which helps in detecting the people who are in the frame of camera and if they are closer than the distance which we are allocating in the device then an alarm will blow and inform them that they are violating the rules. In this way we can control the crowd going out of their homes for work in COVID situation.

Introduction:

People safety has become a major problem in many areas like malls, railway stations and streets during this pandemic situation where people gather in crowds. An efficient automated system to manage the crowd count is essential. Managing a crowd of varying densities involves detection of the individual humans in the crowd. In a high density crowd, because of inter-object occlusion, detection of humans in the crowd will be a challenge in computer vision. Existing tracking algorithms have many challenges in solving this problem. Researchers have proposed and studied the tracking methods which are more efficient.

This study focuses on capturing the video in which it is placed and then divide into frames and then detects the people in that particular frame and if they are closer then the assigned distance then it will automatically blow an alarm which indicates that people are standing or moving in that area are closer to each other. It will be a kind of warning.

The rest of the paper is organized as follows: 1. Literature survey; 2. Proposed work; 3. Tools used; 4. Implementation procedure; 5. Stimulation results; 6. Future scope of project; 7. Conclusion.

Literature Survey:

Detection of objects was mainly introduced by researchers for face detection. The Haar feature was extracted from internal image detection and they elaborated their work. Some researchers explained a framework that can detect by dividing frames. Basic functions from set of wavelet, an object class is derived and it is used as input for support vector machine. Many studies have been carried out to detect humans in a scene by using part-based human detector. Parts cannot be detected clearly, so it depend on factors such as inter- object occlusion, illumination, etc. Some other researchers have done their work on presence of human based on foreground segmentation. The background is deleted from the frame and using some types of filtering, for example Gaussian filtering, is done to extract human motion blobs out of the frame. This process is usually done in a grayscale image. The motion blobs are then done to determine its similarities with that of humans. Illumination variation in different regions has rise of different motion blobs and hence it becomes challenging to formulate a shape for human detection.

Density based crowd detection has been demonstrated by some researchers. The density of crowd is initially detected by taking the video of the area and then dividing that into frame by frame and detects the human's presence in that area. In this detection there will be assigned a particular value for maintain distance between the people by which if any violation is detected then an alarm will blow indicating that they are not maintaining social distancing.

Proposed Work:

The proposed work is to develop a system for detection of crowd and detection of crowd movement using Raspberry pi. Initially, video is collected from the area to detect the presence of human. Then the image is divided into frames and then detection is done. The proposed system uses a camera to record the crowded scene. Raspberry pi 4 uses a Tensorflow which processes the video frame by frame to detect the presence of humans. This process gives the count of humans present in the scene. To manage the crowd, the count of the persons is kept in track and distance between person to person, prevention human from gathering in huge amount in particular area. The detected humans are tracked using the algorithm and if they violate to maintain distance, an alarm will be blown which is connected to the system.

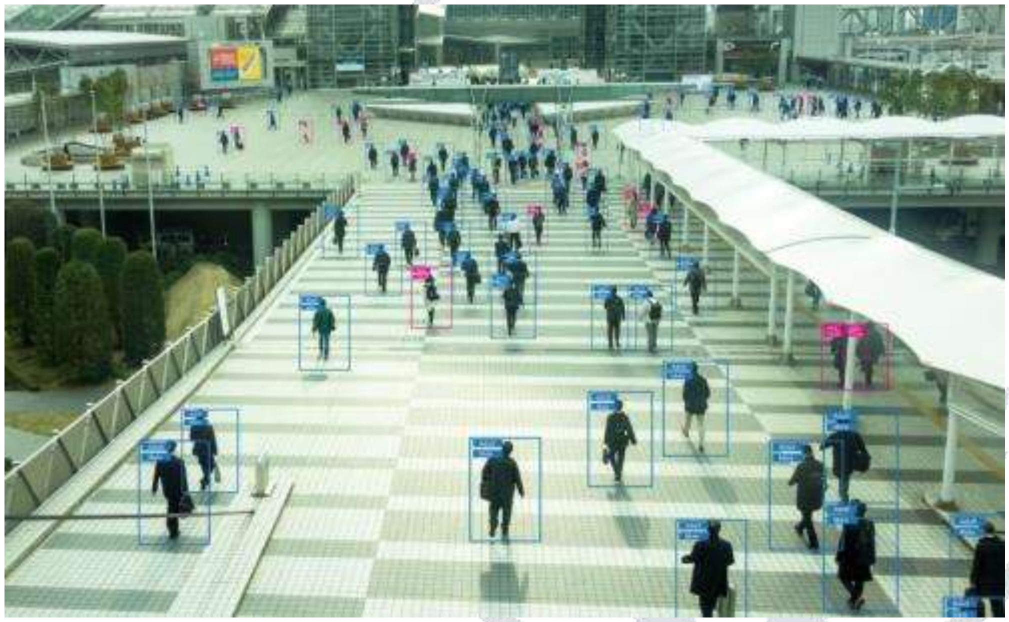


Fig1. Crowd detection in frame

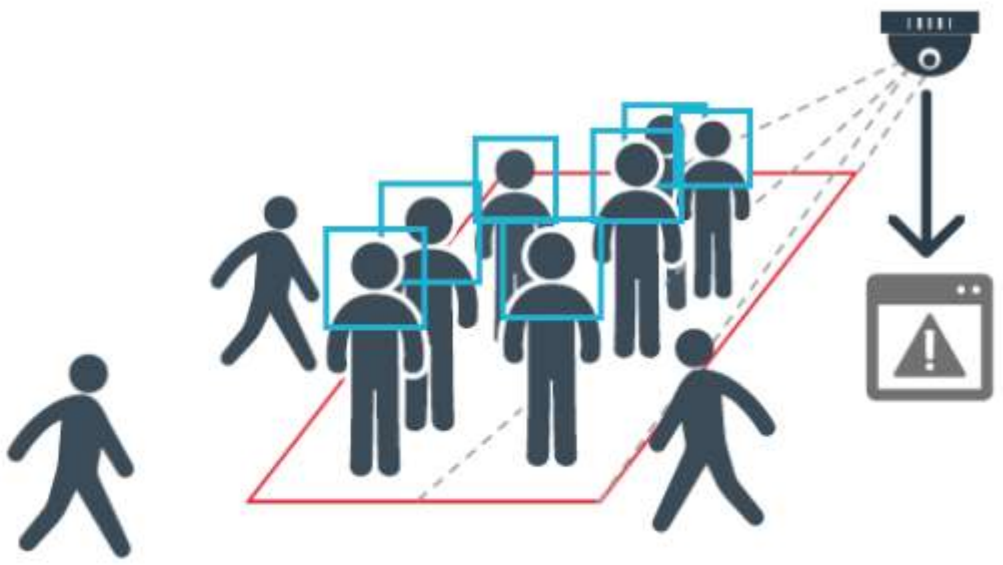


Fig2. Crowd detection in graphic

In the fig 1 there are people walking around which is captured in video and the shown fig is of one frame divided by the system where people are detected and showing the distance between them if they cross the distance between each other there will be blow of alarm. Fig 2 shows the graphic image in which video is captured from the camera and then it send to the system and from there video is dived into frames and if any violation is detected an alarm will be blow.

Flow Chart: RPI camera takes video—> video is sliced into frames—–>frame is resized For processing—>Tensor flow process the frames and detect people

The flow chat of proposed work is above. It mainly consists of two processes: First capturing the video by RPI camera and then dividing that into frames and then using tensor flow sensor people will be detected on the frame and then an alarm will be blown.

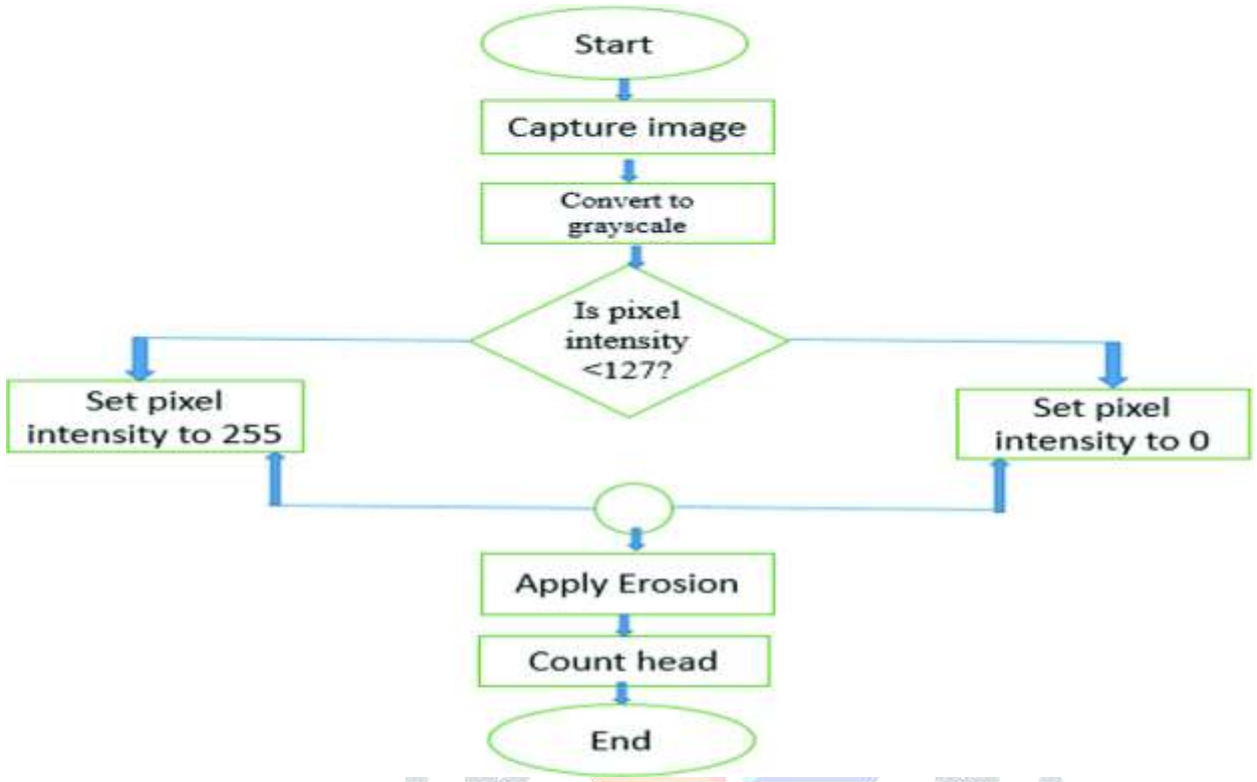
Hardware and Software Tools:

For hardware implementation of this study, a Raspberry Pi 4 board is used. It has a 1.5 GHz processor, Wireless LAN. It also has a 1 GB RAM, 4 USB ports, 40 GPIO pins, Full HDMI port, Ethernet port, Camera interface, Display interface and a Micro SD card slot.

The software used here is OpenCV-Python. OpenCV is a library of programming functions. It is mainly aimed at real-time computer vision. OpenCV supports a wide variety of programming languages like C++, Python, Java, etc. OpenCV-Python is the Python API of OpenCV. To support various applications, OpenCV includes a statistical machine learning library.

Implementation:

The platform chosen for the implementation of the study is OpenCV-Python which is compatible with the Raspberry Pi 4. The proposed work is implemented using the tiny computer, Raspberry Pi 4. It's 1.5 GHz CPU provides results in a short time. The video frames are recorded by the camera interfaced with the Raspberry pi.



First RPI camera captures video in the area placed. The video is then divided into frame by frame. These frames are separated and then processed frame-by-frame. Then by using image processing with the help of TensorFlow, people and vehicles in the video are identified. When all the process is completed, it will check whether the people and vehicles are far from each other in the frame maintain the distance. When that happens, the device gives an alert by speaking or by triggering the light or bulb if there are any disturbances in the frames.

Hardware:

After getting all required components like RPI with Raspbian Image and Python environment setup, we can install the required libraries and modules.

Firstly, install the following libraries in Python.

- Espeak
- Numpy
- Scipy
- Opencv
- Dlib
- Keras
- Tensorflow

To install the above libraries, type the following commands in LX terminal

Sudo apt-get update

Sudo apt-get upgrade

Sudo nano/etc/dphys-swapefile

Then change the line CONF_SWAPSIZE=100 to SWAPSIZE=1024

Sudo /etc/dphys-swapfile swapoff

Sudo /etc/dphys-swapfile swapon

Sudo apt-get install opencv

Sudo apt-get install numpy

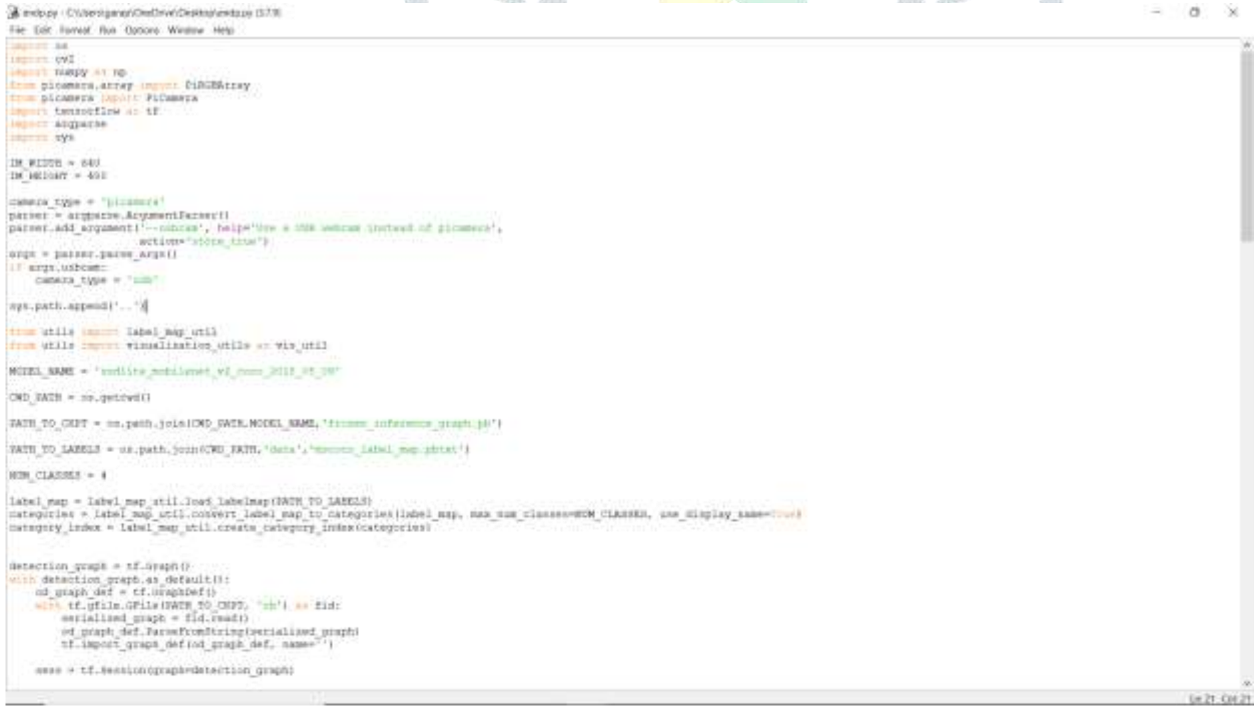
wget https://bootstrap.pypa.io/get-pip.py

pip3 install dlib

pip3 install tensorflow

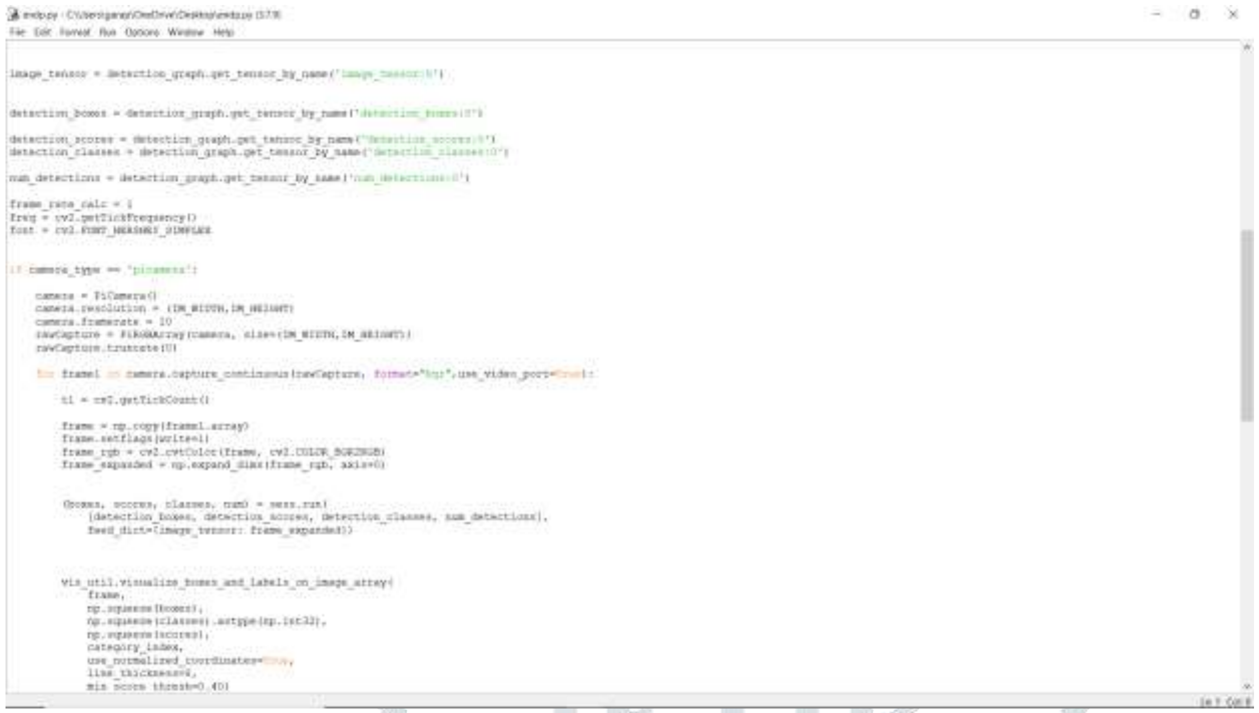
CODE:

After running the below attached code you a window will be open showing detecting the persons present in that area.



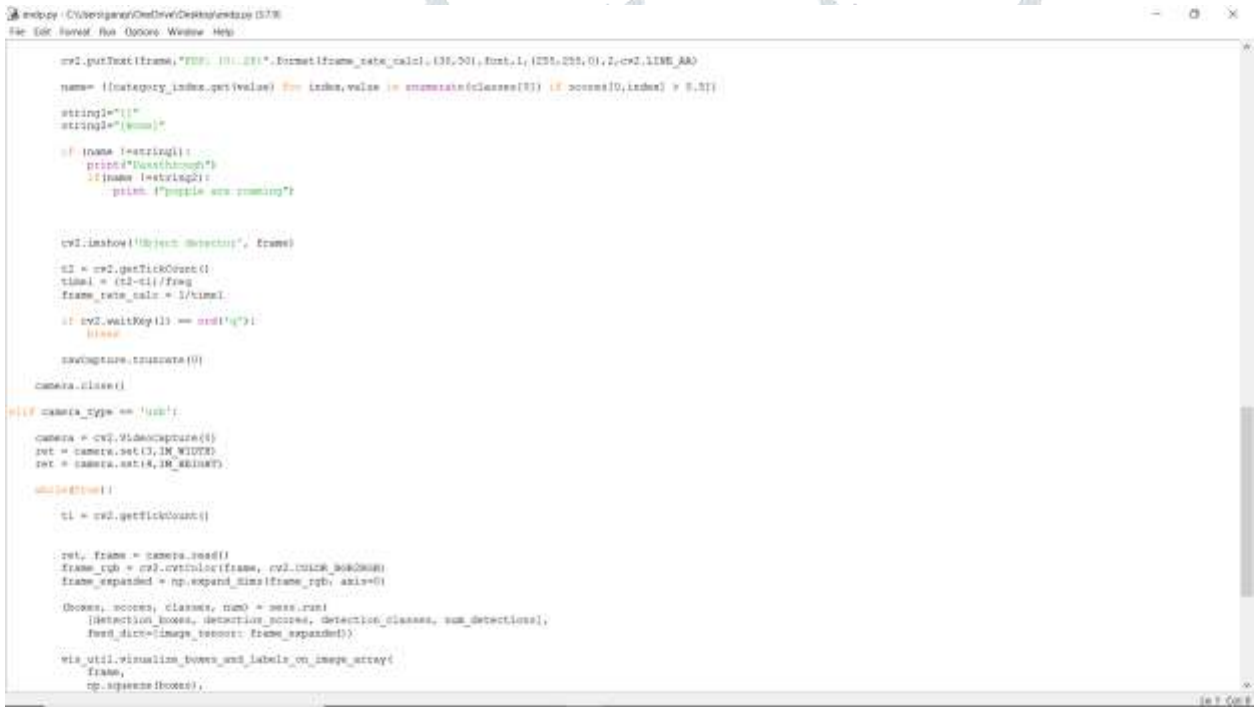
```
import os
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import tensorflow as tf
import argparse
import sys

IM_WIDTH = 640
IM_HEIGHT = 480

camera_type = 'picamera'
parser = argparse.ArgumentParser()
parser.add_argument('--usbcam', help='Use a USB webcam instead of picamera',
                    action='store_true')
args = parser.parse_args()
if args.usbcam:
    camera_type = 'usb'

sys.path.append('..')

from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09'

CWD_PATH = os.getcwd()

PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

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

NUM_CLASSES = 4

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)


detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.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='')

    sess = tf.Session(graph=detection_graph)
```

Fig 1.



```
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')


detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

num_detections = detection_graph.get_tensor_by_name('num_detections:0')

frame_rate_calc = 1
freq = cv2.getTickFrequency()
font = cv2.FONT_HERSHEY_SIMPLEX


if camera_type == 'picamera':

    camera = PiCamera()
    camera.resolution = (IM_WIDTH,IM_HEIGHT)
    camera.framerate = 10
    rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
    rawCapture.truncate(0)

    for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):

        t1 = cv2.getTickCount()

        frame = np.copy(frame1.array)
        frame.setflags(write=1)
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_expanded = np.expand_dims(frame_rgb, axis=0)


        (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores, detection_classes, num_detections],
            feed_dict={image_tensor: frame_expanded})


        vis_util.visualize_boxes_and_labels_on_image_array(
            frame,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=8,
            min_score_thresh=0.40)
```

Fig 2.



```
        cv2.putText(frame,"FPS: {0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LINE_AA)

        name= [(category_index.get(value)) for index,value in enumerate(classes[0]) if scores[0,index] > 0.5]]

        string1="[]"
        string2="[None]"

        if (name !=string1):
            print("Passthrough")
            if(name !=string2):
                print ("people are coming")



        cv2.imshow('Object detector', frame)

        t2 = cv2.getTickCount()
        time1 = (t2-t1)/freq
        frame_rate_calc = 1/time1

        if cv2.waitKey(1) == ord('q'):
            break

        rawCapture.truncate(0)

    camera.close()

elif camera_type == 'usb':

    camera = cv2.VideoCapture(0)
    ret = camera.set(3,IM_WIDTH)
    ret = camera.set(4,IM_HEIGHT)

    while(True):

        t1 = cv2.getTickCount()


        ret, frame = camera.read()
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_expanded = np.expand_dims(frame_rgb, axis=0)

        (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores, detection_classes, num_detections],
            feed_dict={image_tensor: frame_expanded})

        vis_util.visualize_boxes_and_labels_on_image_array(
            frame,
            np.squeeze(boxes),
```

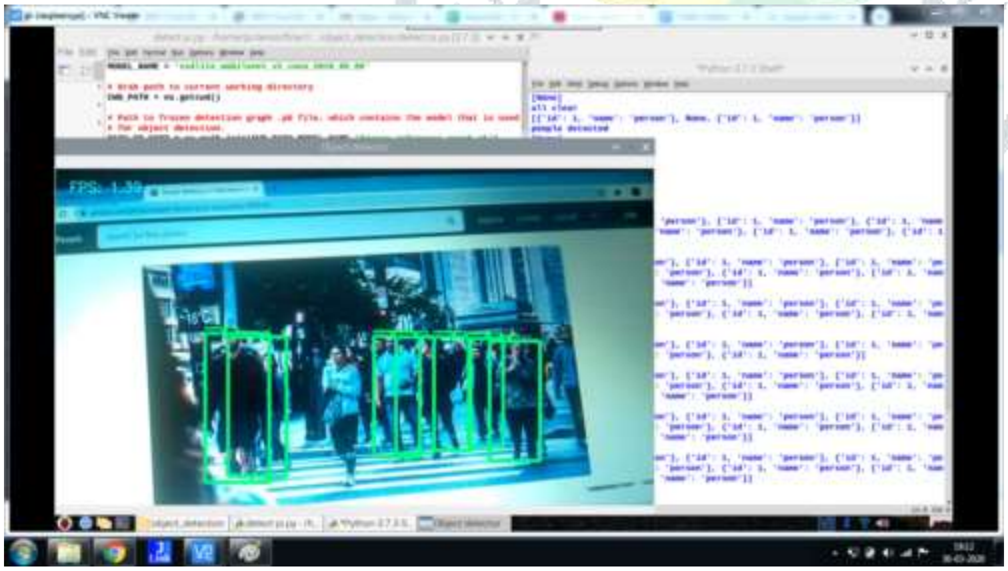
Fig 3.

Source code

endp.py

Results:

After all the process above and complete connections open the folder run the code. Wait for few seconds, a new window opens up and shows the result of Rpi camera. The detection of people is shown in the window and alert is given about their presence in the area place.



```
MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
```

```
[None]
all clear
[{'id': 1, 'name': 'person'}, None, {'id': 1, 'name': 'person'}]
people detected
```

Future Scope:

- This will be able to detect the crowd in public places for avoiding covid19.

- In further we will be able to maintain the Stamped across the country.
- It can be used in Army to detect the enemy movements even in dark by some modification in the visuals of camera.
- It can be further modified into robot surveying for detection of booms and other things which can be watched from far distances.

References:

- Crowd analysis and Its application by Nilam Nur Amir Sjarif, Siti Mariyam Shamsuddin-Johar.
- A Study on Crowd Detection and Density Analysis for Safety Control by Archana Ghotkar- Pune institude of computer technology
- Crowd Pedestrian Counting Considering Network Flow Constraints in Videos- J.Wang.
- Crowd safety: A Real Time System For Counting People – Dr. Arthanariee
- Realization of Multiple Human Head Detection and Direction Movement Using Raspberry Pi- S.Syed Ameer Abbas, M. Anitha.