KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visveswaraya Technological University, Belagavi)

(APPROVED BY AICTE, NEW DELHI)

Course Activity Report on

"REAL-TIME HUMAN DETECTION AND COUNTING"

Submitted in the partial fulfilment for the academic requirement of

4TH *Semester B.E*

In

Information Science Engineering

Submitted by

Adarsh Kumar	**2GI20IS002**
Danesh Naik	**2GI20IS011**
Sahil Faniband	**2GI20IS032**
Vinayak Nikam	**2GI20IS050**

Under the Guidance Of
Dr.Kiran K Tangod
HOD,Dept Of ISE

Academic Year 2021-2022

CERTIFICATE

This is to certify that the Seminar entitled "REAL-TIME HUMAN DETECTION AND COUNTING " is a Bonafede record of the Seminar work done by **Adarsh Kumar, Danesh Naik, Sahil Faniband, Vinayak Nikam** having **USN 2GI20IS002,2GI20IS011, 2GI20IS032, 2GI20IS050** under my supervision and guidance, in partial fulfilment of the requirements for the Outcome Based Education Paradigm in ISE from Gogte Institute of Technology for the academic year 2021-2022.

Faculty In charge

Head of the Department

Rubrics for evaluation of Course Project

Sl. No	Batch No. :02					
1.	Project Title: REAL-TIME HUMAN DETECTION AND COUNTING	MARKS RANGE	USN			
			2GI29IS002	2GI20IS011	2GI20IS032	2GI20IS050
2.	Problem statement(PO2)	0-1				
3.	Objectives of Defined Problem(PO1,PO2)	0-2				
4.	Design/Algorithm/ Flowchart/Methodology (PO3)	0-3				
5.	Implementation details/Function/ Procedure/Classes and objects(Language/Tools)	0-4				
6.	Working model of the final solution (PO3,PO12)	0-5				
7.	Report and Oral presentation skill (PO9,PO10)	0-5				
	Total	20				

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. **Problem Analysis:** Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.

3. **Design/Development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental consideration. **Conduct investigation of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusion.
4. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

5. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

6. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

7. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

8. **Individual and team work:** Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

9. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as,being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

10. **Project management and finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments

11. **Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Real-Time Human Detection and Counting

Python Open CV Project for Human Detection and Counting System Utilizing a Camera or a Video or an Image

ABSTRACT

This project investigates and reports benchmarks for detecting and enumerating humans through real time images, videos and camera.

This is very useful in various image processing and performing computer vision tasks. This schemes have been implemented in Python programming language, and using various tech-stacks like OpenCv, etc.

Table of Contents:

List of Figures:

1. INTRODUCTION

1.1. Problem Statement:

Real-Time Counting People OpenCV Python – The people counting camera counts the number of people who arrive, leave, or pass through a defined region

1.2. Objectives:

Background subtraction technique has been widely used in foreground pixel detection where a fixed camera is usually used to observe dynamic scenes. Generating a background model from video sequences is a challenging task.

(1) A background estimation procedure is performed to separate motion from background. Many standard methods exist for background modeling and seg mentation of foreground objects. In these methods, a background scene model is statistically learned using the redundancy of the pixel intensities in a training stage. The foreground objects can be detected if their Gray value differs significantly from the back-ground model.

(2) Detecting humans in a video is a challenging task owing to their variable appearance and the wide range of poses that they can adopt. The first need is a robust feature set that allows the human form to be discriminated cleanly, even in cluttered backgrounds under difficult illumination. The detection of moving human bodies from live videos, especially from videos taken by a moving camera, is not a trivial task.

(3) The detection normally goes through two stages they are segmentation of moving target and classification of human body. In segmentation stage, optical flow method and difference method are often used. For classification, popular methods include template match, feature characterization, cluster analysis, and machine learning approaches using various techniques such as support vector machine, adaboost, neural network, etc.

1.3 Methodology:

Human detection and tracking is a task common to many applications, such as video surveillance and security, intelligent vehicles, safety driving, public security, etc. Tracking of humans in dynamic scenes has been an important topic of research. Most techniques, however, are limited to situations where humans appear isolated and occlusion is small. Typical methods rely on appearance models that must be acquired when the humans enter the scene and are not occluded

The user interface is developed using Tkinter library of Python programming language.

General Flow Diagram Of Human Detection And Counting System is as shown in Figure 1.1

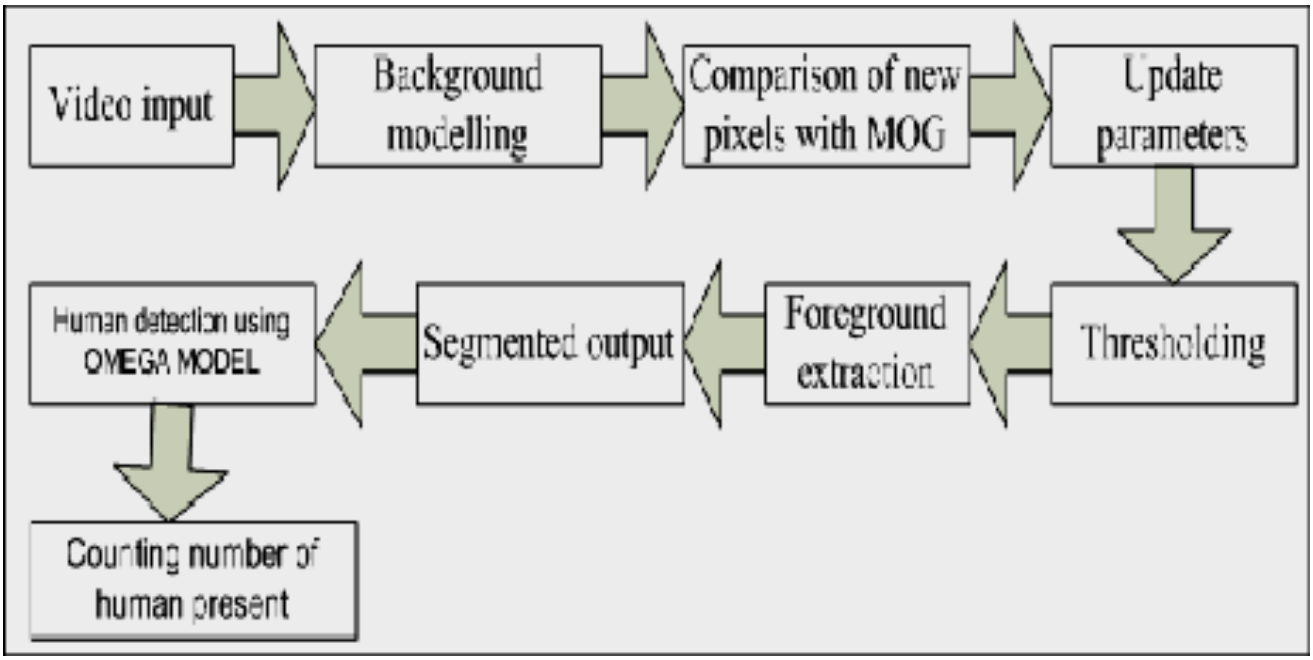

Figure 1.1 Flow of Human Detection and Counting System

1.3.1 Project Details and Technology:

Project Name:	Real-Time Human Detection And Counting
Language/S Used:	Python Deep Learning
Python Version (Recommended):	3.8/3.9
Database :	None
Type:	Desktop Application
Updates:	0

Table 1.0.2 Project Details

1.3.2 Python Libraries Used:

OpenCV:

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

Numpy:

NumPy in Python is a library that is used to work with arrays and was created in 2005 by Travis Oliphant. NumPy library in Python has functions for working in domain of fourier transform, linear algebra, and matrices. Python NumPy is an open-source project that can be used freely. NumPy stands for Numerical Python.

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Matplotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

Dlib:

Dlib is a general purpose cross-platform software library written in the programming language C++. Its design is heavily influenced by ideas from design by contract and component-based software engineering. Thus it is, first and foremost, a set of independent software components. It is open-source software released under a Boost Software License.

Since development began in 2002, Dlib has grown to include a wide variety of tools. As of 2016, it contains software components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many other tasks. In recent years, much of the development has been focused on creating a broad set of statistical machine learning tools and in 2009 Dlib was published in the Journal of Machine Learning Research Since then it has been used in a wide range of domains

1.3.3 Human Detection :

Human detection is the task of locating all instances of human beings present in an image, and it has been most widely accomplished by searching all locations in the image, at all possible scales, and comparing a small area at each location with known templates or patterns of people.

In this we can use various predefined methods and can detect the human in any image, video and can even get various factors like accuracy, each detections counting, etc.

We have implemented the project for three different cases:

- Image.
- Video.
- Camera.

1. Detection & Counting through Image:

This section works with real time images. Here will allow user to select any real time image from the local system and then user can detect the humans in it. And along with that it also gives the count of humans detected.

2. Detection & Counting through Video:

This section works with real time videos. Here will allow user to select any real time video from the local system and then user can detect the humans in it.

Now in case of video, since it is running, while the detection process is going on user will be able to see the detected peoples and their count for each frames per second of the video.

3. Detection & Counting through Camera:

This section works somehow similar to case of video. Here user will be asked to first open the webcam, and it will detect humans that will comes in that webcam during the detection process.

1.3.4 Accuracy:

Now here we have discussed about the main key point of all computer vision project i.e. Accuracy. During the detection process of human, we along with process also kept track of the accuracy with each human is getting detected in image, video and camera.

In our method, we have set the threshold accuracy for the detection process as 70%, so the object detected with accuracy more than the threshold accuracy, we declared it as the well detected human, and display detection indicator around that human during process. We have set this threshold in order to prevent false detection to det displayed while detection process.

Now whenever term accuracy comes, there is always a general question, “What is the maximum accuracy of the detection?” and that we have discussed in the next topic.

Average Accuracy Graph as shown in the figure 1.1

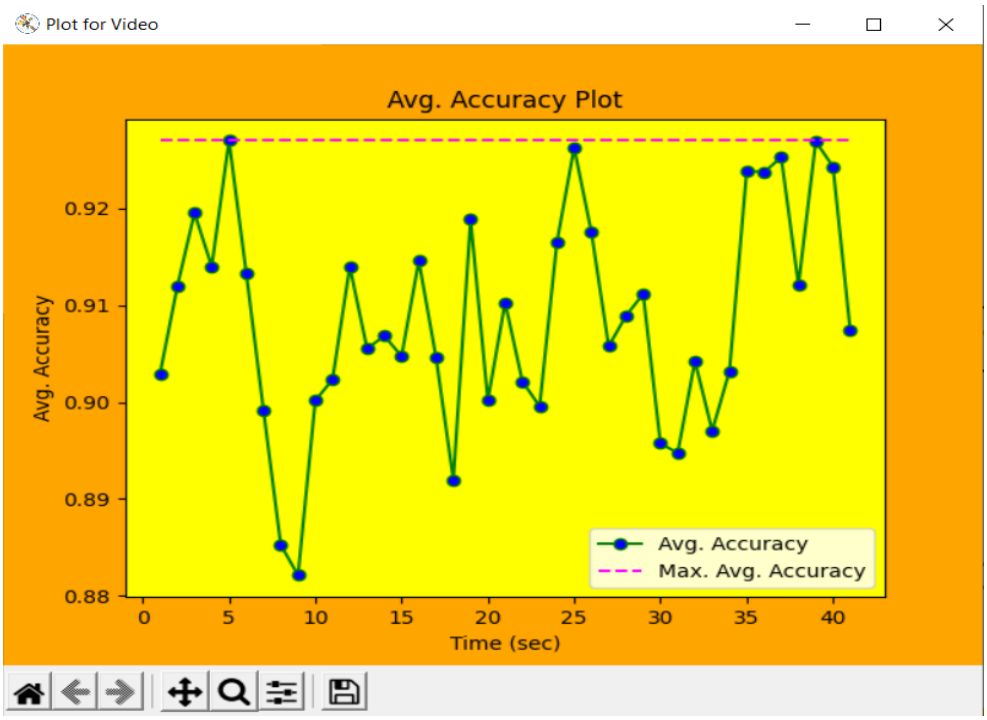

Figure 1.2 Avg. Accuracy plot.

2. LITERATURE SURVEY

Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio proposed a general example-based framework for detecting objects in static images by components. The technique is demonstrated by developing a system that locates people in cluttered scenes. Especially, the system detects the components of a person's body in a picture, i.e., the head, the left and right arms, and therefore the legs, rather than the complete body by using four distinct example based detectors. The system then checks to make sure that the detected components are within the proper geometric configuration.

Histograms of Oriented Gradients Approach for Human detection.

Navneet Dalal and Bill Triggs proposed that the grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. We study the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high- quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so we introduce a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds.

An Overview of our feature extraction and object detection chain is as shown in fig 1. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small spatial regions, for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram "energy "over somewhat larger spatial regions and using the results to normalize all of the cells in the block. We will refer to the normalized descriptor blocks as Histogram of Oriented Gradient (HOG) descriptors. Tiling the detection window with a dense grid of HOG descriptors and using the combined feature vector in a conventional SVM based window classifier gives our human detection chain as shown in Figure 1.3.

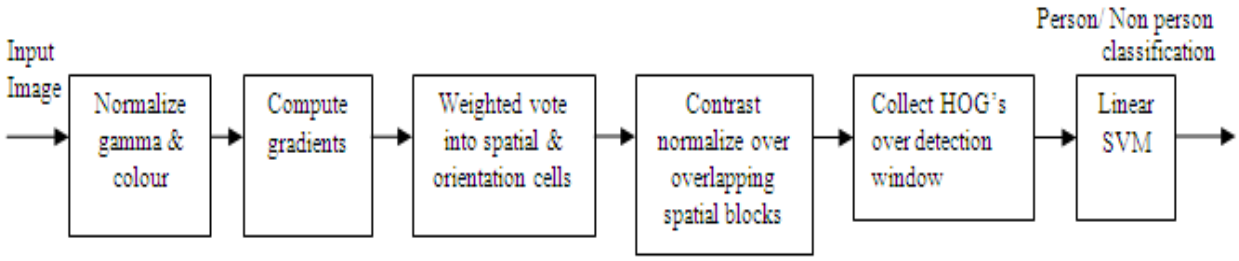

Figure 1. An overview of our feature extraction and object detection chain.

Figure 1.3 An Overview of our feature extraction and object detection chain

Neural Network and EM based people counting and individual detection.

Although there are many human detection methods, most of them are not producing high accuracy and the image resolution must be high. Ya-Li Hou and Grantham K.H. Pang is developed neural network based people counting and EM based individual detection in a low resolution image with complicated scenes that is shown in Figure 3.

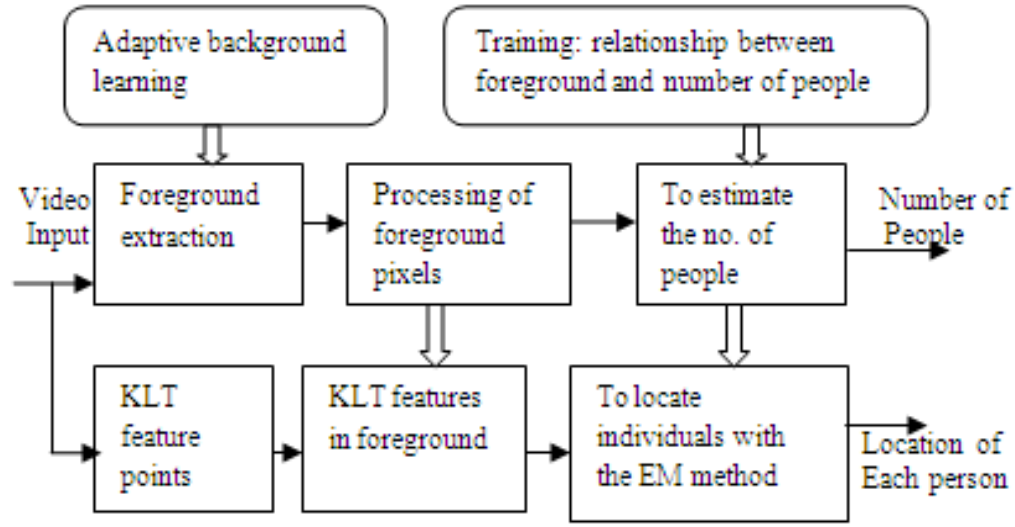

Figure 3. Block diagram of people counting and human detection

Figure 1.4 Block Diagram of people counting and human detection

DDMCMC approach

Tao Zhao and Bo Wu proposed a model based approach to interpret the image observations by multiple partially occluded human hypotheses in a Bayesian framework. This approach to segmenting and tracking multiple humans emphasizes the use of shape models. An overview diagram is given in figure.2.

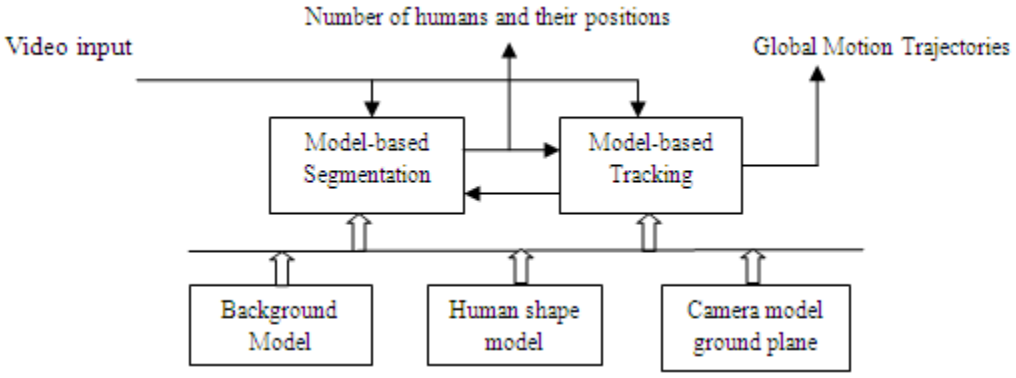

Figure 1.5 Human Detect by DDMCMC Approach

3. IMPLEMENTATION

3.1. Source Code:

```
import cv2 # library of Python bindings designed to solve computer vision problems. cv2
import imutils #basic image processing
import numpy as np
import argparse #helps create a program in a command-line-environment
#frame
def detect(frame):
    bounding_box_cordinates, weights = HOGCV.detectMultiScale(frame, winStride=(4, 4), padding=(8, 8), scale=0.5)
    #detectMultiScale function is used to detect the faces.
# This function will return a rectangle with coordinates(x,y,w,h) around the detected face.
    person = 1
    for x, y, w, h in bounding_box_cordinates:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2) # used to draw a rectangle on image.
        cv2.putText(frame, f'person {person}', (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)# used to draw a text string on detected image
        person += 1


    cv2.putText(frame, 'Status : Detecting ', (40, 40), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255, 0, 0), 2)
    cv2.putText(frame, f'Total Persons : {person-1}', (40, 70), cv2.FONT_HERSHEY_DUPLEX, 0.8, (255, 0, 0), 2)
    cv2.imshow('output', frame)# display an image in a window


    return frame


def detectByPathVideo(path, writer):
    video = cv2.VideoCapture(path) # This will return video from the video to which path is funtion parameter  on your computer.
    check, frame = video.read()
    if check == False:
        print('Video Not Found. Please Enter a Valid Path (Full path of Video Should be Provided).')
        return


    print('Detecting people...')
    while video.isOpened():
        #check is True if reading was successful
        check, frame = video.read()
```

```
        if check:
            frame = imutils.resize(frame, width=min(800, frame.shape[1]))
            frame = detect(frame)

            if writer is not None:
                writer.write(frame)

            key = cv2.waitKey(1) #allows display a window for given milliseconds
            if key == ord('q'):
                break
        else:
            break
    video.release()
    cv2.destroyAllWindows()

def detectByCamera(writer):
    video = cv2.VideoCapture(0) # This will return video from the first webcam on your
computer.
    print('Detecting people...')
    while True:
        check, frame = video.read()

        frame = detect(frame)
        if writer is not None:
            writer.write(frame)

        key = cv2.waitKey(1) #allows display a window for given milliseconds
        if key == ord('q'):
            break

    video.release()
    cv2.destroyAllWindows()

def detectByPathImage(path, output_path):
    image = cv2.imread(path)
    image = imutils.resize(image, width=min(800, image.shape[1]))
    result_image = detect(image)
    if output_path is not None:
```

```
    cv2.imwrite(output_path, result_image) #This will save the image according to the specified format in current working directory.
  cv2.waitKey(0) #It takes time in milliseconds as a parameter and waits for the given time to destroy the window,
            # if 0 is passed in the argument it waits till any key is pressed
  cv2.destroyAllWindows()


def humanDetector(args):
  image_path = args["image"]
  video_path = args['video']
  if str(args["camera"]) == 'true': camera = True
  else: camera = False


  writer = None
  if args['output'] is not None and image_path is None:
    writer = cv2.VideoWriter(args['output'], cv2.VideoWriter_fourcc(*'MJPG'), 10, (600, 600))


  if camera:
    print('[INFO] Opening Web Cam.')
    detectByCamera(writer)
  elif video_path is not None:
    print('[INFO] Opening Video from path.')
    detectByPathVideo(video_path, writer)
  elif image_path is not None:
    print('[INFO] Opening Image from path.')
    detectByPathImage(image_path, args['output'])


def argsParser():
  arg_parse = argparse.ArgumentParser() # intialize the parser start to add custm arguments
  arg_parse.add_argument("-v", "--video", default=None, help="path to Video File ")# command
  arg_parse.add_argument("-i", "--image", default=None, help="path to Image File ")# command
  arg_parse.add_argument("-c", "--camera", default=False, help="Set true if you want to use the camera.")# command
  arg_parse.add_argument("-o", "--output", type=str, help="path to optional output video file")# command
  args = vars(arg_parse.parse_args()) # get an object's _dict_ attribute


  return args
```

```
if __name__ == "__main__":
    HOGCV= cv2.HOGDescriptor() #( Histograms of Oriented Gradients)object detector.
    HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())#set the trainer classifiers
    args = argsParser()#recommended command-line parsing module
    humanDetector(args)


#main module of the system
```

4. OUTPUTS

The screenshots of Outputs are as shown in the figure 1.6,Figure1.7 and Figure 1.8.

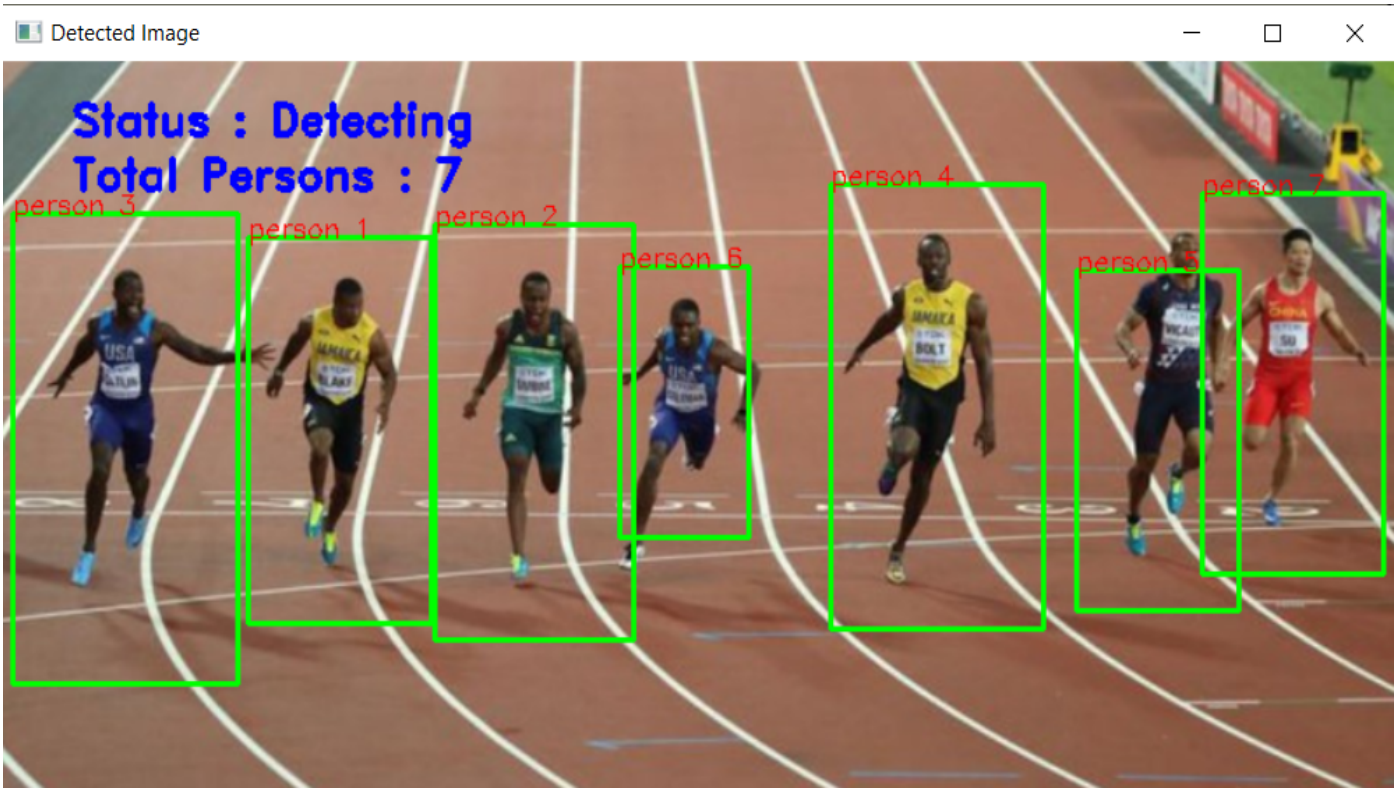

Figure 1.6 Output of Image Input

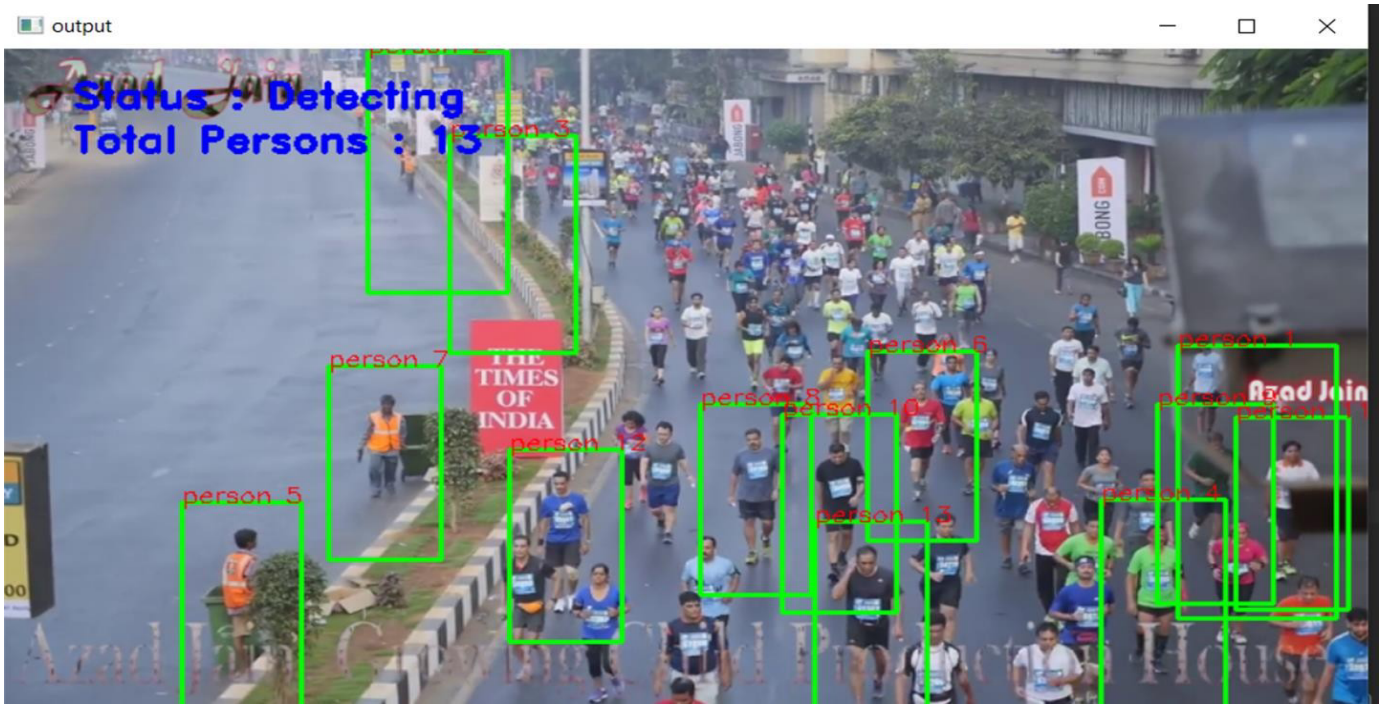

Figure 1.7 Output of Camera Input

Figure 1.8 Output of Video Input

5. CONCLUSION

In the last section of the project, we generate Crowd Report, which will give some message on the basis of the results we got from the detection process. For this we took some threshold human count and we gave different message for different results of human count we got form detection process.

Now coming to the future scope of this project or application, since in this we are taking any image, video or with camera we are detecting humans and getting count of it, along with accuracy.

So some of the future scope can be :

- This can be used in various malls and other areas, to analyse the maximum people count, and then providing some restrictions on number of people to have at a time at that place.
- This can replace various mental jobs, and this can be done more efficiently with machines.
- This will ultimately leads to some kind of crowd-ness control in some places or areas when implemented in that area.

6. REFERENCE

[1] Programming Computer Vision with Python, 1st Edition, Jan Eric Solem, 2012, O' Reily

[2] Learning OpenCv, Adrian Kaehler and Gary Rost Bradski, 2008, O' Reily

[3] Python GUI Programming with Tkinter, Alan D. Moore, 2018