REAL TIME HUMAN DETECTION

By

ISHA VERMA (1816413022)

NEELESH ASTHANA (1816413028)

YASH NIGAM (1816413063)

Submitted to the

Department of Information Technology

Pranveer Singh Institute of Technology, Kanpur

(Dr. A.P.J. Abdul Kalam Technical University, Lucknow)

May, 2022

REAL TIME HUMAN DETECTION

By

ISHA VERMA (1816413022)

NEELESH ASTHANA (1816413028)

YASH NIGAM (1816413063)

Submitted to the
Department of Information Technology

Under the supervision of
Mr. Piyush Bhushan Singh

In Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY
(INFORMATION TECHNOLOGY)

Dr. APJ Abdul Kalam Technical University, Lucknow

May, 2022

TABLE OF CONTENT

DECLARATION

We hereby declare that the project entitled –**Real Time Human Detection** submitted for the B.Tech. (IT) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles. The best of our knowledge and belief, it contains no material previously published or written by any other person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:

Name: Isha Verma

Roll No: 1816413022

Date:

Signature:

Name: Neelesh Asthana

Roll No: 1816413028

Date:

Signature:

Name: Yash Nigam

Roll No: 1816413063

Date:

CERTIFICATE

This is to certify that the project titled **"Real Time Human Detection"** is the bonafide work carried out by,

Isha Verma (1816413022)

Neelesh Asthana (1816413028)

Yash Nigam (1816413063)

In partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Information Technology) student of B.Tech (IT) of Pranveer Singh Institute of Technology, Kanpur affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh (India) during the academic year 2021-22, and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Signature of the Guide

Place:

Date:

ACKNOWLEDGEMENT

We would like to express our gratitude towards our project mentor, **Mr. Piyush Bhushan Singh, Head of Department, Information Technology,** without his guidance and support this work would not have been possible.

We would also like to thank **Rakesh Ranjan, Assistant Professor, Department of Information Technology.** For always motivating us to take up challenging and interesting projects that develop our knowledge in new domains.

Lastly, we would like to thank our parents, family and friends for keeping us motivated in all our life‘s Endeavour‘s.

Signature:

Name: Isha Verma

Roll No: 1816413022

Date:

Signature:

Name: Neelesh Asthana

Roll No: 1816413028

Date:

Signature:

Name: Yash Nigam

Roll No: 1816413063

Date:

ABSTRACT

In recent years, human detection from images and videos has been one of the active research topics in computer vision and machine learning due to many potential applications including image and video content management, video surveillance and driving assistance systems. We have been working with **Human Detection Systems** for the final year research project of our undergraduate studies. Our team was focused on Human Detection from **Live CCTV Camera Feeds**, **images** and **videos** from the given path. Human Detection is a branch of Object Detection. Object Detection is the task of identifying the presence of predefined types of objects in an image. This task involves both **identification of the presence of the objects** and **identification of the rectangular boundary surrounding each object** (i.e. Object Localisation). An object detection system which can detect the class "Human" can work as a Human Detection System. In this python project, we are going to build the Human Detection and Counting System through Webcam or you can give your own video or images. This is a deep learning project on computer vision.

CHAPTER-I

INTRODUCTION

1. INTRODUCTION

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. Detecting humans in images 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.

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.

Some common methods are : -

- **Using Haar Cascade Classifier:**
    - Here we make use of .xml file for human detection, and using that we detect the humans in real time videos and images
- **Using HOG(Histogram of Oriented Gradients) :**
    - Here we make used of predefined functions and with that we detect, and this case gives some how better accuracy as compared to Harr Cascade Classifier.
- **Using Tensorflow:**
    - TensorFlow is an open-source API from Google, which is widely used for solving machine learning tasks that involve Deep Neural Networks. And again this method gives even better accuracy than above two methods.

After analyzing or evaluating these 3 approaches on different scenarios like camera orientation, density of the people, lightning conditions, presence of occlusion. We found that Tensorflow approach has the highest accuracy followed by HOG (Histogram of Oriented Gradients) Approach. HOG Approach is robust in different scenarios, but has poor response time due to the classification requirement. We also observe some drawbacks of these approaches like – Missed Detections, Unreliable Detection Boundary and Flickers in Detection.

We have implemented the application using the Tensorflow approach and got almost the better accuracy.

Prior work on Human Detection: Detecting and counting people and objects has a long history, including a number of projects that focused on performance evaluation. Ogale provides a survey of video-based human detection. The PETS (Performance Evaluation of Tracking and Surveillance) workshops, Ferryman, Crowley, focused on algorithm development and performance evaluation of tasks such as multiple object detection, event detection, and recognition. The National Institute of Standards and Technology (NIST) has helped to develop performance metrics for object and human detection in a number of different applications, ranging from videoconferences through surveillance to counting and tracking people in stores and commercial establishments. NIST has worked with the United States Department of Homeland Security, with the British Home Office, and with the European CHIL program (Computers in the Human Interaction Loop), and the CLEAR evaluations, NIST has also worked with the US Army Collaborative Technology Alliance (CTA) on Robotics to evaluate systems that locate and track human pedestrians from a moving vehicle.

Concurrent work on Human Detection: Concurrently in our project we have developed similar Human Detection system, in which we detect the Human in real time from given images, videos or Live video from CCTV webcam and return the total count of the humans that are present in given image, video or live video from CCTV webcam. To fulfill the objective of our project we have selected Tensorflow Approach among the various algorithms like CHT (Circular Hough Transform), Frame differencing, Haar Cascades, HOG (Histogram of Oriented Gradients) etc. available for detecting objects / Humans in a real time scenerios. TensorFlow is an open-source API from Google, which is widely used for solving machine learning tasks that involve Deep Neural Networks. And again this method gives even better accuracy than all other methods available for detecting objects / Humans in a real time scenerios.

1.1 Problem Definition

Suppose we are given the task of counting the number of humans in a scene (see Figure 1.1 as an example). A computer vision approach to solving the problem might start with detecting and identifying objects in the scene that can be classified as human. Once those objects are detected, the counting process is straightforward. In general, the problem of detecting humans occurs in many applications of computer vision, including image and video content management, video surveillance, assisted automotive, etc. Human detection is an active research topic in computer vision and the problem can be stated simply as: given an image or video sequence, localise all objects that are humans. This problem corresponds to determining the regions within an image or video sequence containing humans. The usual representation of such regions is a rectangular bounding box. Figure 1.1 shows some examples of human detection results.

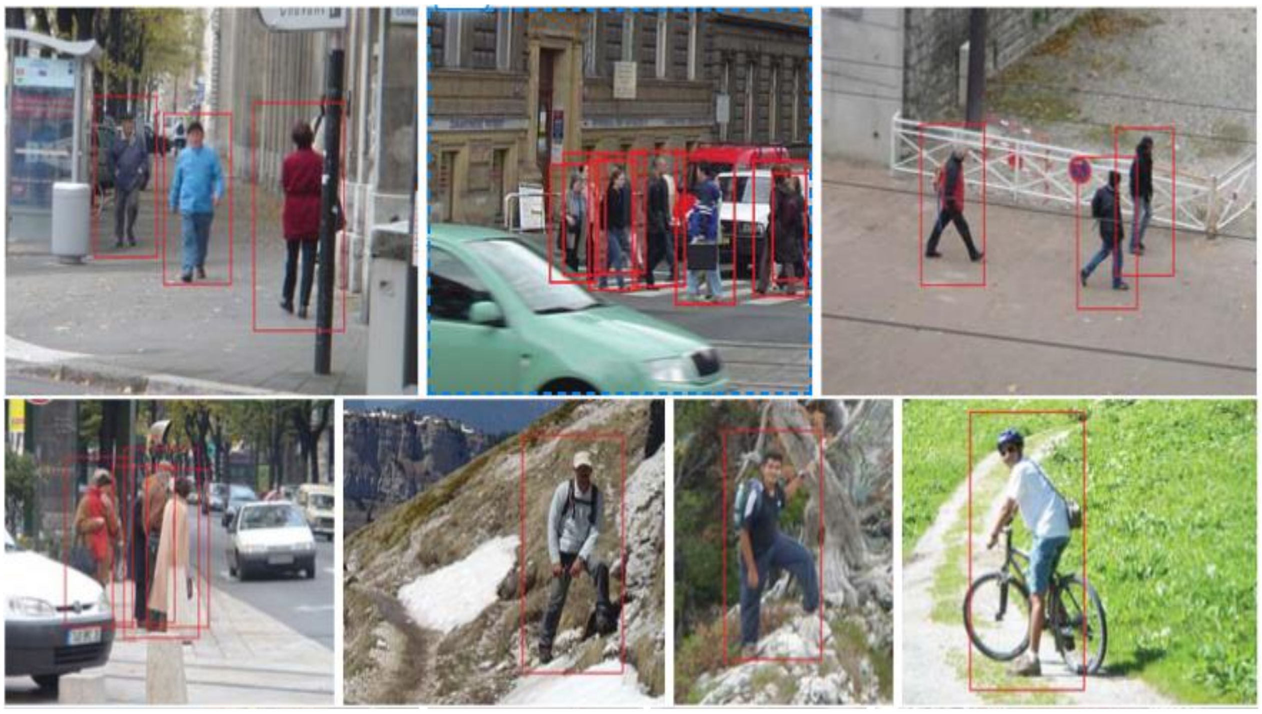
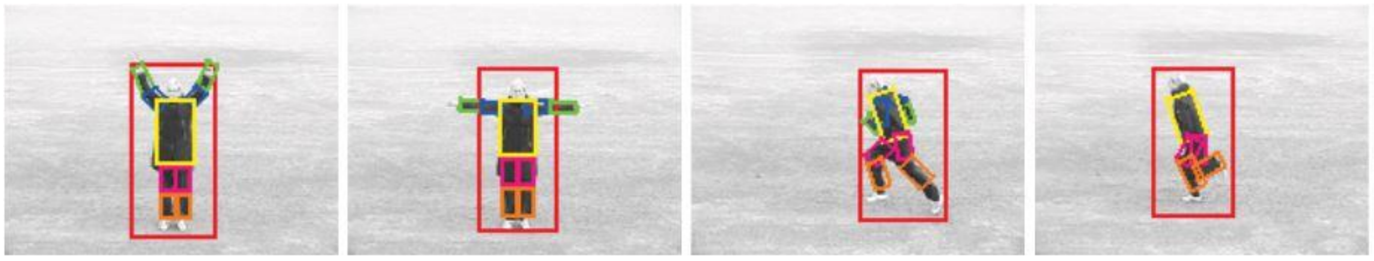
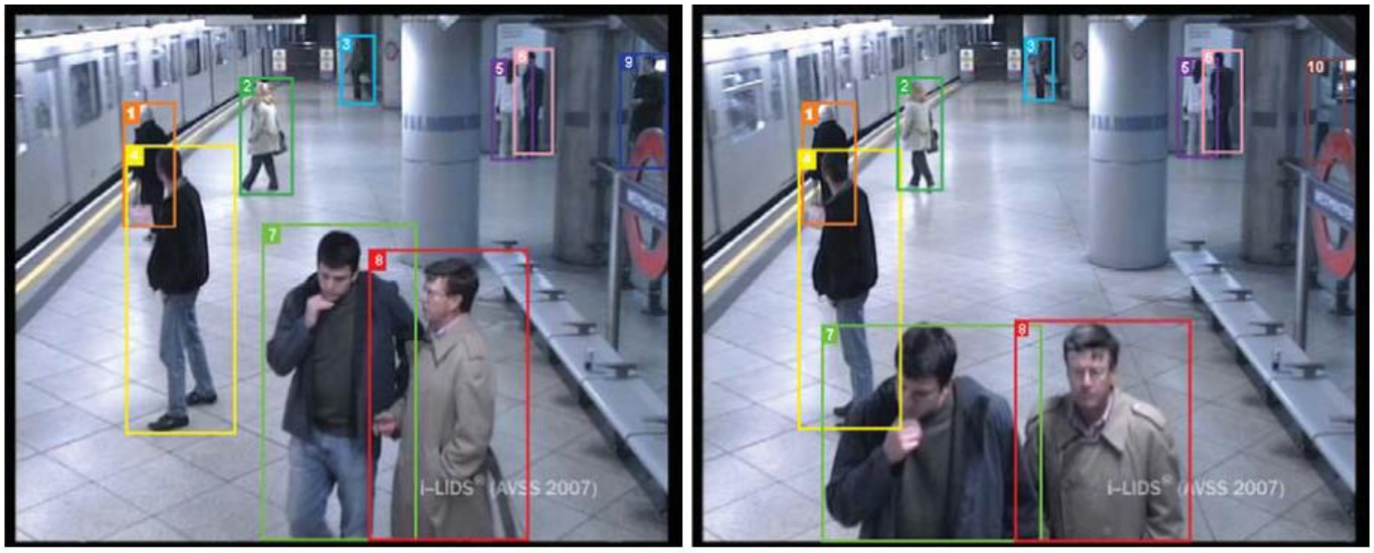


Figure 1.1

During the last decade or so, human detection has attracted considerable attention from the computer vision research community largely due to the variety of applications in which it forms an integral part. In video-based surveillance systems, human detection is a crucial step. The aim of video-based surveillance is usually to identify and monitor humans, for security purposes, in crowded scenes such as airports, train stations, supermarkets, etc. The video footage captured by the installed cameras are processed to detect and track the full human body or body parts in the scene. Based on the location and movements of the body parts, the poses can be estimated and actions are recognised. Often, such detection is combined with face detection and recognition so as to enhance the performance of the whole system. Figure 1.2(a) and (b) show examples of surveillance systems.

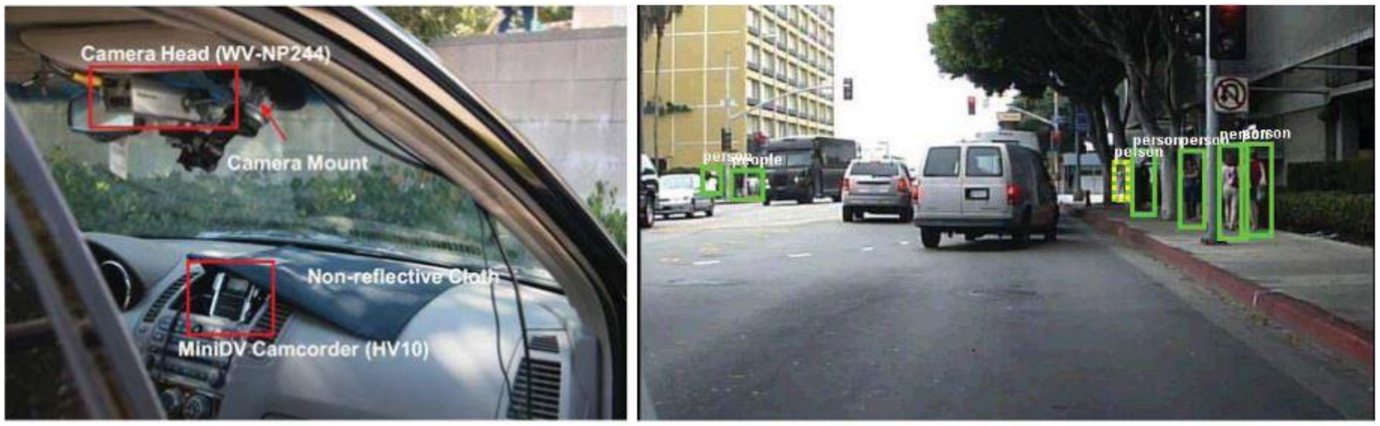

(a)



(b)

Human detection has also found an application in driving assistance systems that are incorporated into automobiles. These systems alert the driver to dangerous situations involving the presence of pedestrians on streets. Since the late 1990s, driving assistance systems have been studied intensively. Examples include the ARGO vehicle developed by the University of Parma, Italy and the Chamfer system released by the University of Amsterdam and Daimler Chrysler. Recently, Mobileye has launched the first vision-based collision warning system with full auto brake and pedestrian detection for use in the Volvo S60 cars.

This system consists of a camera to capture the image scenes, a dual-mode radar unit to measure the distance from the car to pedestrians, and a central processing unit integrated with a human detection algorithm to locate pedestrians. When pedestrians are detected at a nearby distance, the driver is first alerted by a flash and audible warning. The recommended reactions such as speed reduction and braking are then indicated to the driver. If the driver does not respond to the warning appropriately and the system realises that a collision is imminent, a full brake will be performed automatically. Figure 1.2(c) represents pedestrian detection in autonomous vehicles. It is clear to see that the overall performance of the system would be improved by employing robust human detection algorithms.



(c)

1.2 Project Overview/Specifications

Our target is to develop a robust human detection Model which is able to detect the humans in various postures, viewpoints, under partial occlusion and realistic environments. Basically in our project we have developed similar Human Detection system, in which we detect the human in real time from given images, videos or Live video from CCTV webcam and return the total count of the humans that are present in given image, video or live video from CCTV webcam. To fulfill the objective of our project we have selected Tensorflow approach among the various algorithms like CHT (Circular Hough Transform), Frame differencing, Haar Cascades, HOG (Histogram of Oriented Gradients) etc. available for detecting objects / Humans in a real time scenerios.

1.3 Hardware Specifications

1.3.1 RAM and Storage

Deep learning is a very CPU intensive program-esque thing to be running, so be prepared to shell out a lot of money for a good enough system. Here are some system requirements to adhere to (parenthesis are what you can maybe get away with)

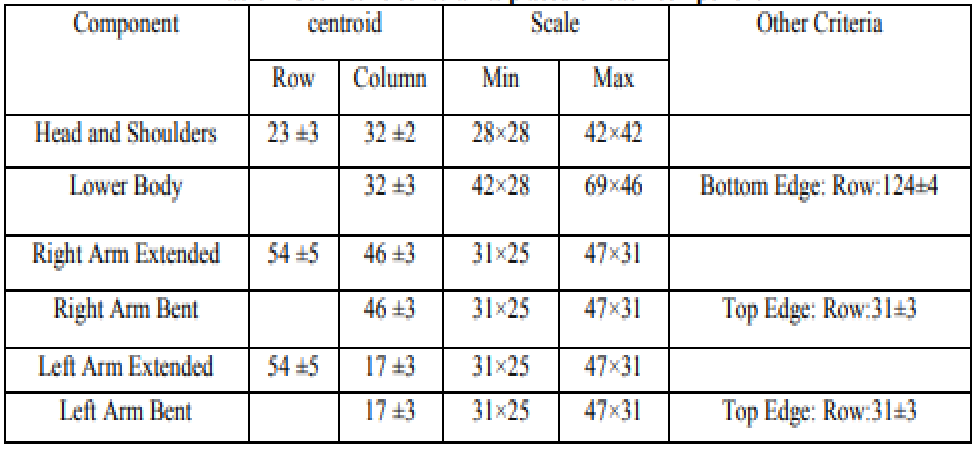
- Intel® Pentium® CPU G3250 @ 3.20GHz 3.20 GHz
- 16GB of RAM (8GB is okay but not for the performance you may want and or expect)
- Windows 64-bit operating system.

CHAPTER-II
LITERATURE REVIEW

2.1. Component based people detection:

Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio [4] 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. In particular, the system detects the components of a person’s body in an image, i.e., the head, the left and right arms, and the legs, instead of the full body by using four distinct example based detectors. The system then checks to ensure that the detected components are in the proper geometric configuration (i.e. shown inTable 1).

Table 1 Geometric constraints placed on each component



Component	centroid		Scale		Other Criteria
	Row	Column	Min	Max	
Head and Shoulders	23 ±3	32 ±2	28×28	42×42	
Lower Body		32 ±3	42×28	69×46	Bottom Edge: Row:124±4
Right Arm Extended	54 ±5	46 ±3	31×25	47×31	
Right Arm Bent		46 ±3	31×25	47×31	Top Edge: Row:31±3
Left Arm Extended	54 ±5	17 ±3	31×25	47×31	
Left Arm Bent		17 ±3	31×25	47×31	Top Edge: Row:31±3

We calculated the geometric constraints for each component from a sample of the training images, tabulated in Table 1, by taking means of the centroid and top and bottom boundary edges of each component over positive detections in the training set. There are two sets of constraints for the arms, one intended for extended arms and the other for bent arms. Haar wavelet functions are used to represent the components in the images and Support Vector Machines (SVM) to classify the patterns. Four component-based detectors are combined at the next level by another SVM. The results of the component detectors are used to classify a pattern as either a “person” or a “nonperson”. For this purpose uses one classifier, named as Adaptive Combination of Classifiers (ACC) that improves accuracy of people detection. This system performs significantly better than a similar full-body person detector. This suggests that the improvement in performance is due to the componentbased approach and the ACC data classification architecture. While this paper establishes that, this system can detect people who are slightly rotated in depth, it does not determine, quantitatively, the extent of this capability. This is the main drawback of the method and also more time consuming task.

2.2 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 (in Fig 1).

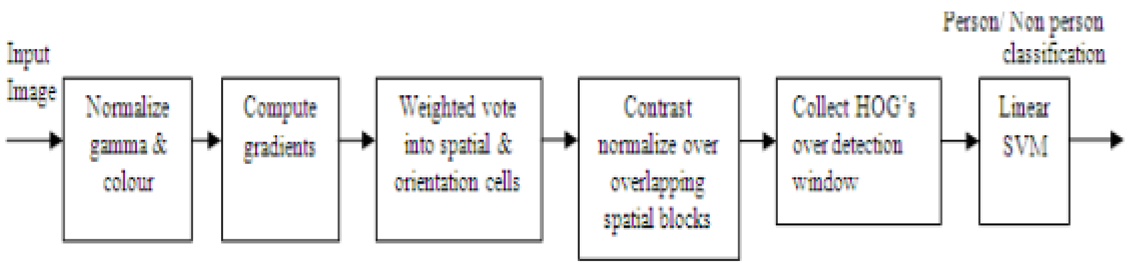

Input Image
Normalize gamma & colour
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
Collect HOG's over detection window
Linear SVM
Person/ Non person classification

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

We have shown that using locally normalized histogram of gradient orientations features similar to SIFT descriptors in a dense overlapping grid gives very good results for person detection, reducing false positive rates by more than an order of magnitude relative to the best Haar wavelet based detector. Histograms of oriented Gradients may achieve more accurate counting and detection results when the crowd is small. Disadvantage of this process is that time consuming task and shows only results for a small crowd with few occlusions, it also required high resolution images.

2.3 DDMCMC approach:

Tao Zhao and Bo Wu [2] 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

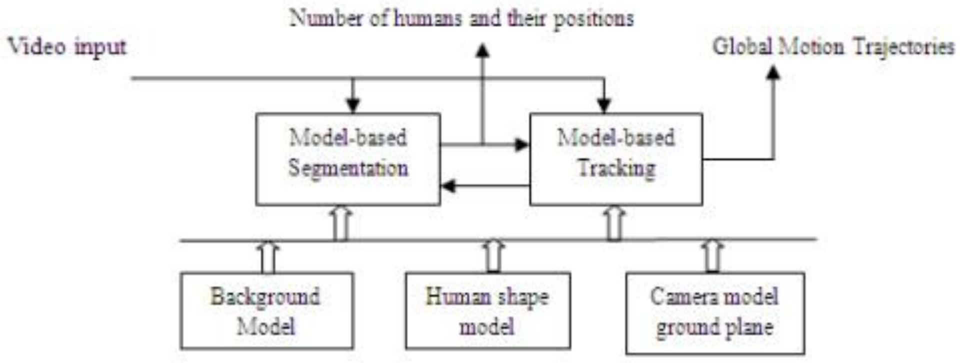


Figure 2. An overview diagram of Segmentation and tracking

In fig 2.based on a background model, the foreground blobs are extracted as the basic observation. By using the camera model and the assumption that objects move on a known ground plane, multiple 3D human hypotheses are projected onto the image plane and matched with the foreground blobs. Since the hypotheses are in 3D, occlusion reasoning is straightforward. In one frame, we segment the foreground blobs into multiple humans and associate the segmented humans with the existing trajectories. Then, the tracks are used to propose human hypotheses in the next frame. The segmentation and tracking are integrated in a unified framework and interoperate along time. We formulate the problem of segmentation and tracking as one of Bayesian inference to find the best interpretation given the image observations, the prior models, and the estimates from the previous frame analysis that is, the maximum a posteriori (MAP) estimation.

The optimal solution is obtained by using an efficient sampling method, data-driven Markov chain Monte Carlo (DDMCMC), which uses image observations for proposal probabilities. Knowledge of various aspects, including human shape, camera model, and image cues, are integrated in one theoretically sound framework. To improve the computational efficiency, we use direct image features from a bottom-up image analysis as importance proposal probabilities to guide the moves of the markov chain. This method is able to successfully detect and track humans in the scenes of complexity with high detection and low false alarm rates. Here a more accurate 3-D model composed of three ellipsoids was used. To deal with the occlusion problem, a joint probability for multiple humans has been considered. Finally, the human detection and tracking problem was formulated as a Maximum A Posteriori (MAP) problem simultaneously. A sophisticated sampling algorithm, Data Driven Markov Chain Monte Carlo, is used to find the best configuration for the MAP problem. Some positive results for a crowd of a dozen people were obtained. To reduce the dependence on an accurate foreground contour, this may be easily corrupted by noise. This is a time consuming task.

CHAPTER-III
BACKGROUND OF THE PROJECT

3.1 Computer Vision

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects.

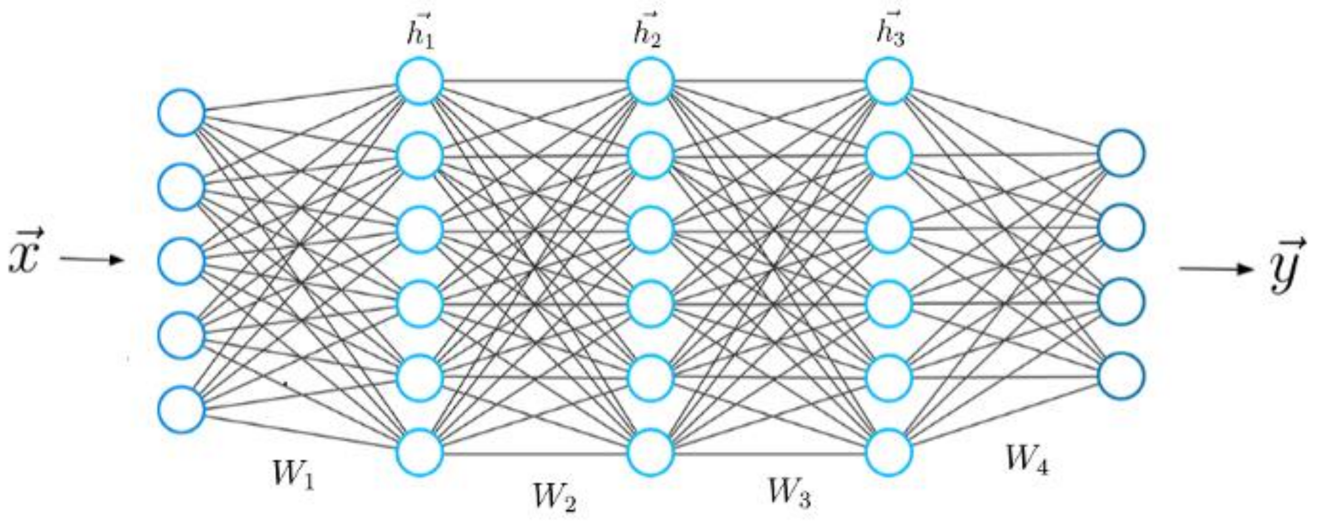
Two essential technologies are used to accomplish this: a type of machine learning called :-

- deep learning
- convolutional neural network (CNN).

3.1.1 Deep Learning

Deep Learning is a subset of Machine Learning, which on the other hand is a subset of Artificial Intelligence. Artificial Intelligence is a general term that refers to techniques that enable computers to mimic human behavior. Machine Learning represents a set of algorithms trained on data that make all of this possible.

Basically Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier. At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.



To understand deep learning, imagine a toddler whose first word is dog. The toddler learns what a dog is -- and is not -- by pointing to objects and saying the word dog. The parent says, "Yes, that is a dog," or, "No, that is not a dog." As the toddler continues to point to objects, he becomes more aware of the features that all dogs possess. What the toddler does, without knowing it, is clarify a complex abstraction -- the concept of dog -- by building a hierarchy in which each level of abstraction is created with knowledge that was gained from the preceding layer of the hierarchy.

Deep Learning, on the other hand, is just a type of Machine Learning, inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a given logical structure. To achieve this, deep learning uses a multi-layered structure of algorithms called neural networks.

3.1.2 Convolutional Neural Network (CNN)

CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on a underdone image and do not need any preprocessing.

A convolutional neural network is a feed forward neural network, seldom with up to 20. The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces. The agenda for this sphere is to activate machines to view the world as humans do, perceive it in a alike fashion and even use the knowledge for a multitude of duty such as image and video recognition, image inspection and classification, media recreation, recommendation systems, natural language processing, etc.

Convolutional Neural Network Design : The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order. It is the sequential design that give permission to CNN to learn hierarchical attributes. In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers. The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will “look” at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

A CNN helps a machine learning or deep learning model “look” by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is “seeing.” The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

Much like a human making out an image at a distance, a CNN first discerns hard edges and simple shapes, then fills in information as it runs iterations of its predictions. A CNN is used to understand single images. A recurrent neural network (RNN) is used in a similar way for video applications to help computers understand how pictures in a series of frames are related to one another.

3.2 Computer vision applications

There is a lot of research being done in the computer vision field, but it’s not just research. Real-world applications demonstrate how important computer vision is to endeavors in business, entertainment, transportation, healthcare and everyday life. A key driver for the growth of these applications is the flood of visual information flowing from smartphones, security systems, traffic cameras and other visually instrumented devices. This data could play a major role in operations across industries, but today goes unused. The information creates a test bed to train computer vision applications and a launchpad for them to become part of a range of human activities.

It has various different application that too in various fields. Some of them are listed below:

- Object Detection
- Screen Reader
- Intruder Detection
- Code and Character Reader
- Robotics
- Motion Analysis
- Image Restoration

3.2 Computer vision examples

There Here are a few examples of established computer vision tasks:

- **Image classification** sees an image and can classify it (a dog, an apple, a person’s face). More precisely, it is able to accurately predict that a given image belongs to a certain class. For example, a social media company might want to use it to automatically identify and segregate objectionable images uploaded by users.

- **Object detection** can use image classification to identify a certain class of image and then detect and tabulate their appearance in an image or video. Examples include detecting damages on an assembly line or identifying machinery that requires maintenance.

- **Object tracking** follows or tracks an object once it is detected. This task is often executed with images captured in sequence or real-time video feeds. Autonomous vehicles, for example, need to not only classify and detect objects such as pedestrians, other cars and road infrastructure, they need to track them in motion to avoid collisions and obey traffic laws.

- **Content-based image retrieval** uses computer vision to browse, search and retrieve images from large data stores, based on the content of the images rather than metadata tags associated with them. This task can incorporate automatic image annotation that replaces manual image tagging. These tasks can be used for digital asset management systems and can increase the accuracy of search and retrieval.

3.3 Detection and Enumeration in Computer Vision

Detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.

For Detection process in computer vision, there are various methods and each one have different level of accuracy according to their advancement level, like is some methods that is invented in very early stage, they give more cases of false detection as compared to the advanced methods that had been discovered after that.

And here we have used the human as an entity which we are detecting our project and along with that, we are also counting humans through image, video and camera.

CHAPTER -IV

PROJECT PREREQUISITE

The project in Python requires you to have basic knowledge of python programming and the Tensorflow library. We will be needing following libraries:

- OpenCV
- Fpdf
- Numpy
- Tkinter
- PIL
- matplotlib
- Tensorflow

4.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

OpenCV plays a major role in real-time operation which is very important in today’s systems.

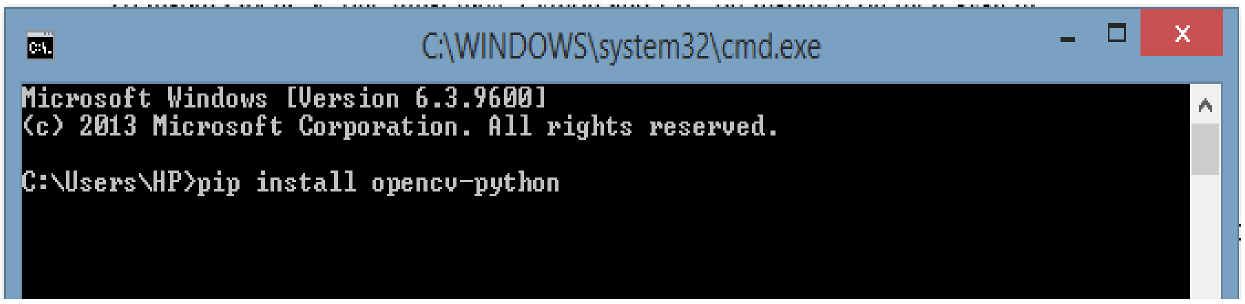
By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To Identify image patterns and its various features we use vector space and perform mathematical operations on these features

To install OpenCV, one must have Python and PIP, preinstalled on their system.

Downloading and Installing OpenCV:

OpenCV can be directly downloaded and installed with the use of pip (package manager). To install OpenCV, just go to the command-line and type the following command:

pip install opencv-python



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\HP>pip install opencv-python
```

4.2 Fpdf

PyFPDF is a library for PDF document generation under Python, ported from PHP (see FPDF “Free”-PDF, a well-known PDFlib-extension replacement with many examples, scripts and derivatives).

Compared with other PDF libraries, PyFPDF is simple, small and versatile, with advanced capabilities and easy to learn, extend and maintain.

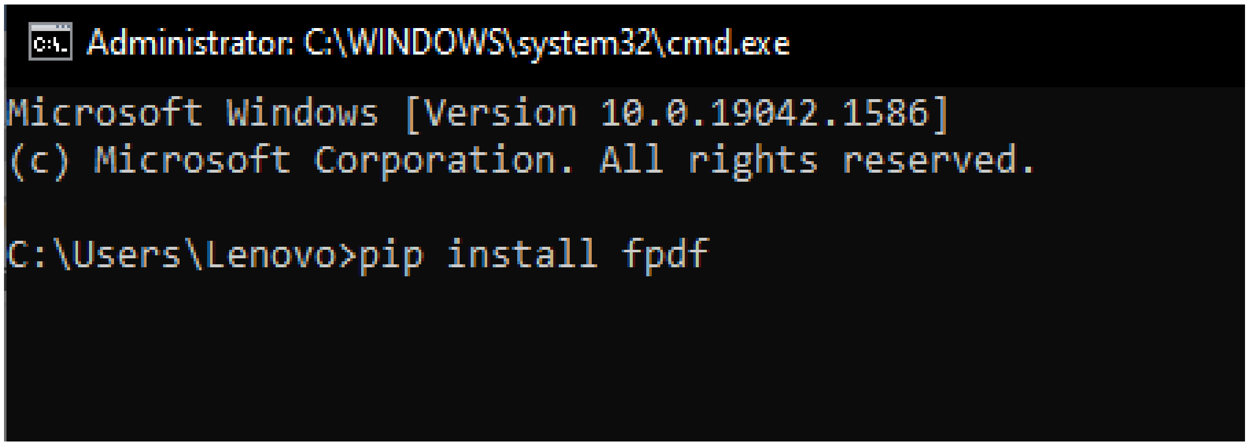
Features:

- Python 2.5 to 2.7 support (with experimental Python3 support)
- Unicode (UTF-8) TrueType font subset embedding
- Barcode I2of5 and code39, QR code coming soon …
- PNG, GIF and JPG support (including transparency and alpha channel)
- Templates with a visual designer & basic html2pdf
- Exceptions support, other minor fixes, improvements and PEP8 code cleanups

Downloading and Installing Fpdf:

Fpdf can also be directly downloaded and installed with the use of pip (package manager). To install Fpdf, just go to the command-line and type the following command:

pip install fpdf



Administrator: C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>pip install fpdf
```

4.3 Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.

NumPy uses much less memory to store data: The NumPy arrays takes significantly less amount of memory as compared to python lists. It also provides a mechanism of specifying the data types of the contents, which allows further optimisation of the code.

As an example, we can create a simple array of six elements using a python list as well as by using numpy.array, The difference in amount of memory occupied by it is quite outstanding.

Using NumPy for creating n-dimension arrays: An n-dimension array is generally used for creating a matrix or tensors, again mainly for the mathematical calculation purpose. Compare to python list base n-dimension arrays, NumPy not only saves the memory usage, it provide a significant number of additional benefits which makes it easy to mathematical calculations

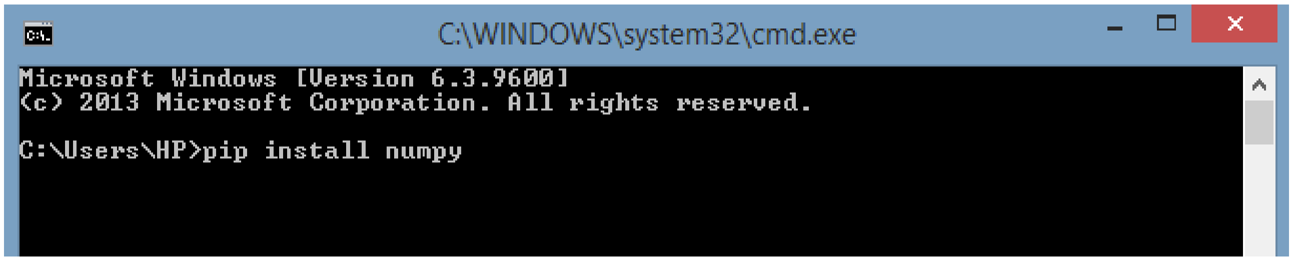
Here is a list of things we can do with NumPy n-dimensional arrays which is otherwise difficult to do.

The dimensions of the array can be changed at runtime as long as the multiplicity factor produces the same number of elements. For example, a 2 * 5 matrix can be converted into 5 * 2 and a 1 * 4 into 2 * 2. This can be done by calling the NumPy .reshape(...) function on the arrays.

Downloading and Installing Numpy:

Numpy can also be directly downloaded and installed with the use of pip (package manager). To install Numpy, just go to the command-line and type the following command:

```
pip install numpy
```



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\HP>pip install numpy
```

4.4Tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

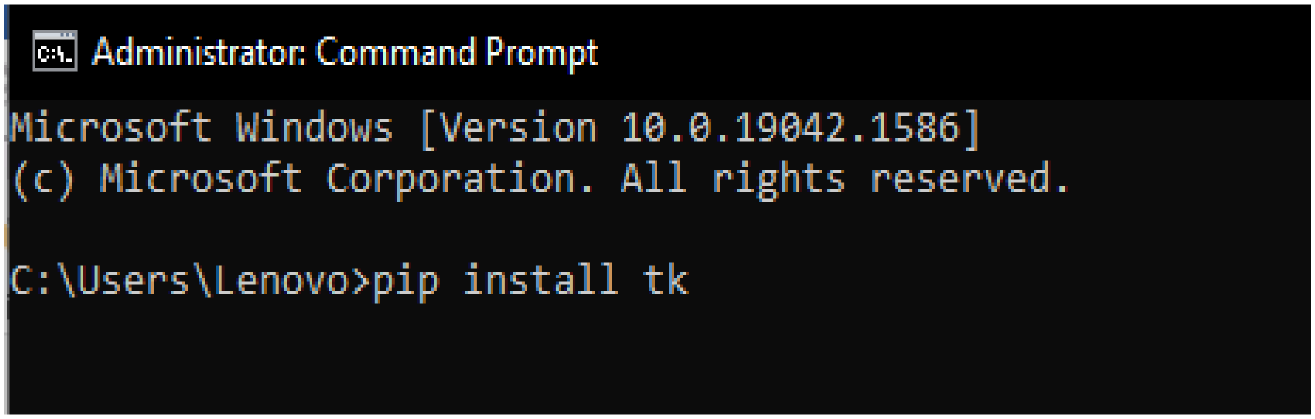
tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **Pack( ) method:** It organizes the widgets in blocks before placing in the parent widget.
2. **grid( ) method:** It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place( ) method:** It organizes the widgets by placing them on specific positions directed by the programmer.

Downloading and Installing imutils:

tkinter can be directly downloaded and installed with the use of pip (package manager). To install tkinter, just go to the command-line and type the following command:

pip install tk



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>pip install tk
```

4.5 PIL (Python Imaging Library)

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

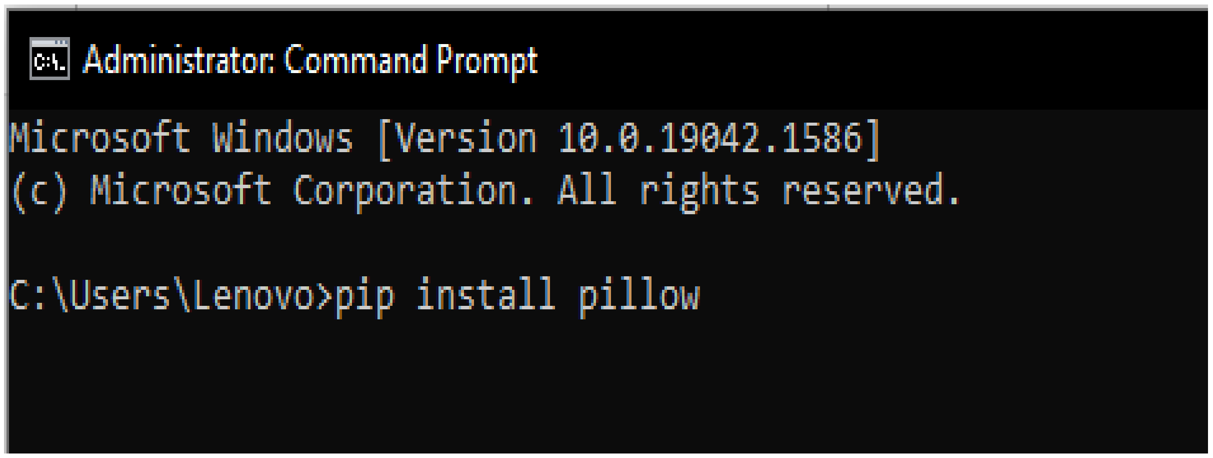
Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and added Python3.x support to it.

Pillow was announced as a replacement for PIL for future usage. Pillow supports a large number of image file formats including BMP, PNG, JPEG, and TIFF. The library encourages adding support for newer formats in the library by creating new file decoders.

Downloading and Installing PIL:

PIL can be directly downloaded and installed with the use of pip (package manager). To install PIL, just go to the command-line and type the following command:

pip install pillow



```
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>pip install pillow
```

4.6 matplotlib

A matplotlib is an open-source Python library which used to plot the graphs. It represents the data through the graphical form. The graphical form can be a Scatter Plot, Bar Graph, Histogram, Area Plot, Pie Plot, etc. The matplotlib library is generally used to data visualization. Data visualization allows us to make a effective decision for organization.

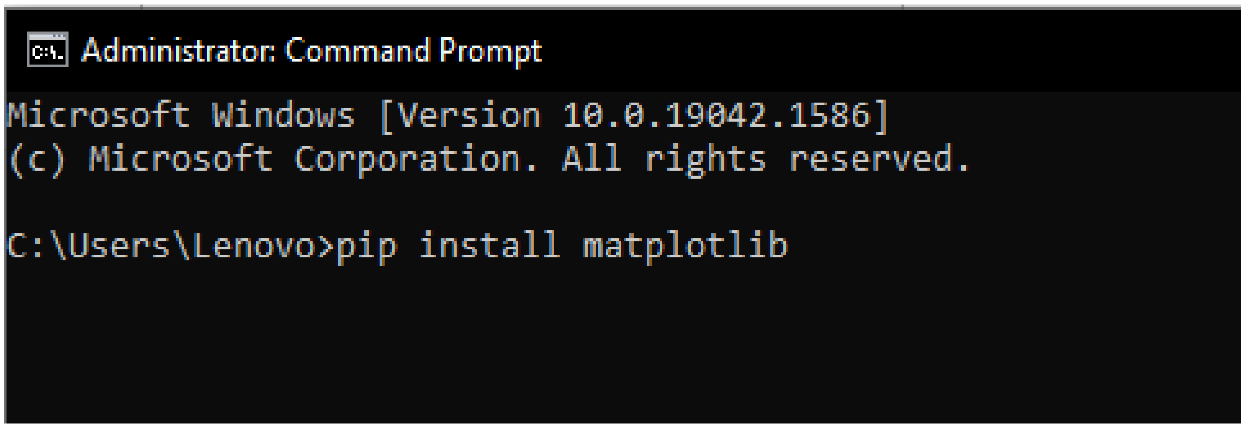
Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications. A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

- The pyplot API is a hierarchy of Python code objects topped by *matplotlib.pyplot*
- An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib's backend layers.

Downloading and Installing matplotlib:

matplotlib can be directly downloaded and installed with the use of pip (package manager). To install matplotlib, just go to the command-line and type the following command:

pip install matplotlib



```
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>pip install matplotlib
```

4.7 tensorflow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

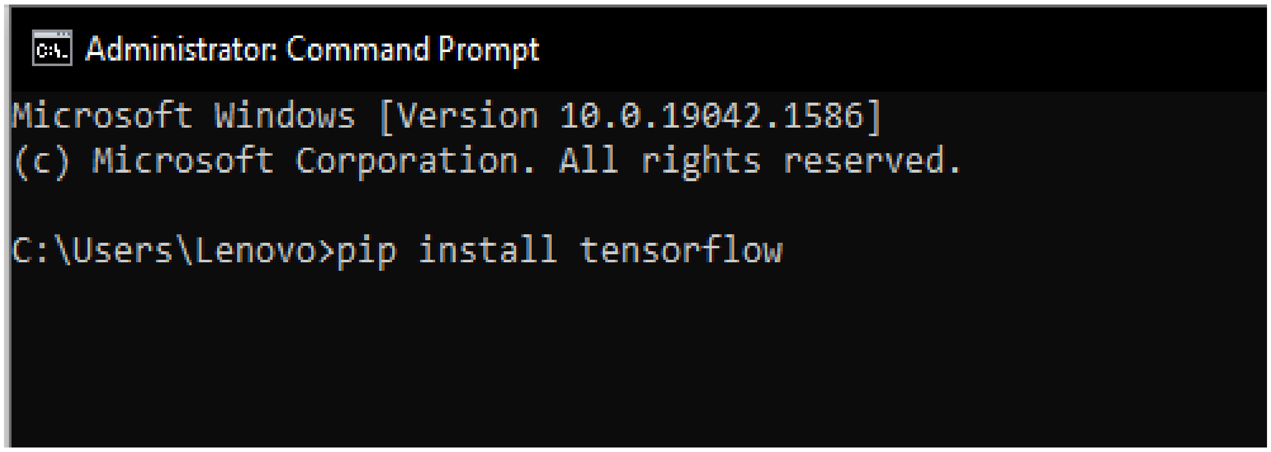
TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

Downloading and Installing tensorflow:

tensorflow can be directly downloaded and installed with the use of pip (package manager). To install tensorflow, just go to the command-line and type the following command:

pip install tensorflow



```
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>pip install tensorflow
```

CHAPTER-V
SOFTWARE SPECIFICATION

5.1 Visual Studio Code

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. A preview build was released shortly thereafter. On November 18, 2015, the source of Visual Studio Code was released under the MIT License, and made available on GitHub. Extension support was also announced. On April 14, 2016, Visual Studio Code graduated from the public preview stage and was released to the Web.[12] Microsoft has released most of Visual Studio Code's source code on GitHub under the permissive MIT License, while the releases by Microsoft are proprietary freeware.

In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 70% of 82,000 respondents reporting that they use it.

Features

- Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

- Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

- Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.
- Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.
- Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

CHAPTER-VI

CODE AND IMPLEMENTATION

Main.py

1. Import Libraries

```
from tkinter import *

import tkinter as tk

import tkinter.messagebox as mbox

from tkinter import filedialog

from PIL import ImageTk, Image

import cv2

import argparse

from persondetection import DetectorAPI

import matplotlib.pyplot as plt

from fpdf import FPDF
```

2. Main Window & Configuration

```
window = tk.Tk()

window.title("Real Time Human Detection & Counting")

window.iconbitmap('Images/icon.ico')

window.geometry('1000x700')
```

3. Top label

```
start1 = tk.Label(text = "REAL-TIME-HUMAN\nDETECTION  &  COUNTING",
font=("Arial", 50,"underline"), fg="magenta")

start1.place(x = 70, y = 10)
```

4. Function defined to start the main application

```
def start_fun():
        window.destroy()
```

5. **Creation of Start Button:**

```
    Button(window, text="▶ START",command=start_fun,font=("Arial", 25), bg =
    "orange", fg = "blue", cursor="hand2", borderwidth=3,
    relief="raised").place(x =130 , y =570 )
```

6. **Images on main window:**

```
    path1 = "Images/front2.png"

    img2 = ImageTk.PhotoImage(Image.open(path1))

    panel1 = tk.Label(window, image = img2)

    panel1.place(x = 90, y = 250)


    path = "Images/front1.png"

    img1 = ImageTk.PhotoImage(Image.open(path))

    panel = tk.Label(window, image = img1)

    panel.place(x = 380, y = 180)
```

7. **Function created for exiting from window:**

```
    exit1 = False

    def exit_win():

        global exit1

        if mbox.askokcancel("Exit", "Do you want to exit?"):

            exit1 = True

            window.destroy()
```

8. **Creation of Exit button:**

```
Button(window, text="❌ EXIT",command=exit_win,font=("Arial", 25), bg =
"red", fg = "blue", cursor="hand2", borderwidth=3,
relief="raised").place(x =680 , y = 570 )

window.protocol("WM_DELETE_WINDOW", exit_win)

window.mainloop()

if exit1==False:

    # Main Window & Configuration of window1

    window1 = tk.Tk()

    window1.title("Real Time Human Detection & Counting")

    window1.iconbitmap('Images/icon.ico')

    window1.geometry('1000x700')


    filename=""

    filename1=""

    filename2=""
```

9. Argparse ( ) Method:

```
    def argsParser():

        arg_parse = argparse.ArgumentParser()

        arg_parse.add_argument("-v", "--video", default=None, help="path
to Video File ")

        arg_parse.add_argument("-i", "--image", default=None, help="path
to Image File ")

        arg_parse.add_argument("-c", "--camera", default=False, help="Set
true if you want to use the camera.")

        arg_parse.add_argument("-o", "--output", type=str, help="path to
optional output video file")

        args = vars(arg_parse.parse_args())
```

```
            return args
```

10. Image section

```
        def image_option():
            # new windowi created for image section
            windowi = tk.Tk()
            windowi.title("Human Detection from Image")
            windowi.iconbitmap('Images/icon.ico')
            windowi.geometry('1000x700')


            max_count1 = 0
            framex1 = []
            county1 = []
            max1 = []
            avg_acc1_list = []
            max_avg_acc1_list = []
            max_acc1 = 0
            max_avg_acc1 = 0
```

11. function defined to open the image:

```
            def open_img():
                global filename1, max_count1, framex1, county1, max1,
    avg_acc1_list, max_avg_acc1_list, max_acc1, max_avg_acc1
                max_count1 = 0
                framex1 = []
                county1 = []
                max1 = []
                avg_acc1_list = []

                max_avg_acc1_list = []

                max_acc1 = 0

                max_avg_acc1 = 0


                filename1 = filedialog.askopenfilename(title="Select Image
    file", parent = windowi)

                path_text1.delete("1.0", "end")

                path_text1.insert(END, filename1)
```

12. function defined to detect the image:

```
        def det_img():

            global filename1, max_count1, framex1, county1, max1,
avg_acc1_list, max_avg_acc1_list, max_acc1, max_avg_acc1

            max_count1 = 0

            framex1 = []

            county1 = []

            max1 = []

            avg_acc1_list = []

            max_avg_acc1_list = []

            max_acc1 = 0

            max_avg_acc1 = 0


            image_path = filename1

            if(image_path==""):

                mbox.showerror("Error", "No Image File Selected!", parent
= windowi)
```

```
                return

            info1.config(text="Status : Detecting...")

            #info2.config(text="
        ")

            mbox.showinfo("Status", "Detecting, Please Wait...", parent =
windowi)

            # time.sleep(1)

            detectByPathImage(image_path)
```

13. main detection process start from here:

```
        def detectByPathImage(path):

            global filename1, max_count1, framex1, county1, max1,
avg_acc1_list, max_avg_acc1_list, max_acc1, max_avg_acc1

            max_count1 = 0

            framex1 = []

            county1 = []

            max1 = []

            avg_acc1_list = []

            max_avg_acc1_list = []

            max_acc1 = 0

            max_avg_acc1 = 0
```

14. function defined to plot the enumeration fo people detected:

```
            def img_enumeration_plot():

                plt.figure(facecolor='orange', )

                ax = plt.axes()

                ax.set_facecolor("yellow")
                plt.plot(framex1, county1, label="Human Count",
color="green", marker='o', markerfacecolor='blue')

                plt.plot(framex1, max1, label="Max. Human Count",
linestyle='dashed', color='fuchsia')

                plt.xlabel('Time (sec)')

                plt.ylabel('Human Count')

                plt.legend()

                plt.title("Enumeration Plot")

                plt.get_current_fig_manager().canvas.set_window_title("Pl
ot for Image")

                plt.show()


            def img_accuracy_plot():

                plt.figure(facecolor='orange', )

                ax = plt.axes()

                ax.set_facecolor("yellow")

                plt.plot(framex1, avg_acc1_list, label="Avg. Accuracy",
color="green", marker='o', markerfacecolor='blue')

                plt.plot(framex1, max_avg_acc1_list, label="Max. Avg.
Accuracy", linestyle='dashed', color='fuchsia')

                plt.xlabel('Time (sec)')

                plt.ylabel('Avg. Accuracy')

                plt.title('Avg. Accuracy Plot')

                plt.legend()

                plt.get_current_fig_manager().canvas.set_window_title("Pl
ot for Image")

                plt.show()
```

15. PDF generation:

```
            def img_gen_report():
                pdf = FPDF(orientation='P', unit='mm', format='A4')
                pdf.add_page()
                pdf.set_font("Arial", "", 20)
                pdf.set_text_color(128, 0, 0)
                pdf.image('Images/Crowd_Report.png', x=0, y=0, w=210,
h=297)

                pdf.text(125, 150, str(max_count1))
                pdf.text(105, 163, str(max_acc1))
                pdf.text(125, 175, str(max_avg_acc1))
                if (max_count1 > 25):
                    pdf.text(26, 220, "Max. Human Detected is greater
than MAX LIMIT.")
                    pdf.text(70, 235, "Region is Crowded.")
                else:
                    pdf.text(26, 220, "Max. Human Detected is in range of
MAX LIMIT.")
                    pdf.text(65, 235, "Region is not Crowded.")

                pdf.output('Crowd_Report.pdf')
                mbox.showinfo("Status", "Report Generated and Saved
Successfully.", parent = windowi)

        odapi = DetectorAPI()
        threshold = 0.7
        image = cv2.imread(path)
```

```
            img = cv2.resize(image, (image.shape[1], image.shape[0]))

            boxes, scores, classes, num = odapi.processFrame(img)

            person = 0

            acc=0

            for i in range(len(boxes)):


                if classes[i] == 1 and scores[i] > threshold:

                    box = boxes[i]

                    person += 1

                    cv2.rectangle(img, (box[1], box[0]), (box[3], box[2]),
(255,0,0), 2)  # cv2.FILLED #BGR

                    cv2.putText(img, f'P{person, round(scores[i], 2)}', (box[1]
- 30, box[0] - 8), cv2.FONT_HERSHEY_COMPLEX,0.5, (0, 0, 255), 1)  # (75,0,130),

                    acc += scores[i]

                    if (scores[i] > max_acc1):

                        max_acc1 = scores[i]


            if (person > max_count1):

                max_count1 = person

            if(person>=1):

                if((acc / person) > max_avg_acc1):

                    max_avg_acc1 = (acc / person)


            cv2.imshow("Human Detection from Image", img)

            info1.config(text="
 ")

            info1.config(text="Status : Detection & Counting Completed")
            #
info2.config(text="                                                  ")

            # info2.config(text="Max. Human Count : " + str(max_count1))

            cv2.waitKey(0)

            cv2.destroyAllWindows()



            for i in range(20):

                framex1.append(i)

                county1.append(max_count1)

                max1.append(max_count1)

                avg_acc1_list.append(max_avg_acc1)

                max_avg_acc1_list.append(max_avg_acc1)
```

16. Enumeration Plot Button:

```
Button(windowi, text="Enumeration\nPlot", command=img_enumeration_plot,
cursor="hand2", font=("Arial", 20),bg="orange", fg="blue").place(x=100,
y=530)
```

17. Avg. Accuracy Plot Button:

```
Button(windowi, text="Avg. Accuracy\nPlot", command=img_accuracy_plot,
cursor="hand2", font=("Arial", 20),bg="orange", fg="blue").place(x=700,
y=530)
```

18. Generate Crowd Report Button:

```
Button(windowi, text="Generate  Crowd  Report", command=img_gen_report,
cursor="hand2", font=("Arial", 20),bg="light gray",
fg="blue").place(x=325, y=550)
```

19. Preview Image:

```
        def prev_img():
            global filename1
            img = cv2.imread(filename1, 1)
            cv2.imshow("Selected Image Preview", img)
```

20. For images:

```
        lbl1 = tk.Label(windowi,text="DETECT  FROM\nIMAGE",
font=("Arial", 50, "underline"),fg="brown")
        lbl1.place(x=230, y=20)
        lbl2 = tk.Label(windowi,text="Selected Image", font=("Arial",
30),fg="green")
        lbl2.place(x=80, y=200)
        path_text1 = tk.Text(windowi, height=1, width=37, font=("Arial",
30), bg="light yellow", fg="orange",borderwidth=2, relief="solid")
        path_text1.place(x=80, y = 260)
```

#BUTTONS FOR IMAGES WINDOW

21. Select button:

```
Button(windowi, text="SELECT", command=open_img, cursor="hand2",
font=("Arial", 20), bg="light green", fg="blue").place(x=220, y=350)
```

22. Preview button:

```
Button(windowi, text="PREVIEW",command=prev_img, cursor="hand2",
font=("Arial", 20), bg = "yellow", fg = "blue").place(x = 410, y = 350)
```

23. Detect button:

```
Button(windowi, text="DETECT",command=det_img, cursor="hand2",
font=("Arial", 20), bg = "orange", fg = "blue").place(x = 620, y = 350)
```

```
        info1 = tk.Label(windowi,font=( "Arial", 30),fg="gray")

        info1.place(x=100, y=445)

        # info2 = tk.Label(windowi,font=("Arial", 30), fg="gray")

        # info2.place(x=100, y=500)
```

24. Exit window:

```
        def exit_wini():

            if mbox.askokcancel("Exit", "Do you want to exit?", parent =
windowi):

                windowi.destroy()

        windowi.protocol("WM_DELETE_WINDOW", exit_wini)
```

25. Video Section:

```
         def video_option():

             # new windowv created for video section

             windowv = tk.Tk()

             windowv.title("Human Detection from Video")

             windowv.iconbitmap('Images/icon.ico')

             windowv.geometry('1000x700')


             max_count2 = 0

             framex2 = []

             county2 = []

             max2 = []

             avg_acc2_list = []

             max_avg_acc2_list = []

             max_acc2 = 0
```

```
        max_avg_acc2 = 0
```

26. function defined to open the video:

```
    def open_vid():
        global filename2, max_count2, framex2, county2, max2,
avg_acc2_list, max_avg_acc2_list, max_acc2, max_avg_acc2
        max_count2 = 0
        framex2 = []
        county2 = []
        max2=[]
        avg_acc2_list = []
        max_avg_acc2_list = []
        max_acc2 = 0
        max_avg_acc2 = 0

        filename2 = filedialog.askopenfilename(title="Select Video
file", parent=windowv)
        path_text2.delete("1.0", "end")
        path_text2.insert(END, filename2)
```

27.function defined to detect the video:

```
    def det_vid():
        global filename2, max_count2, framex2, county2, max2,
avg_acc2_list, max_avg_acc2_list, max_acc2, max_avg_acc2
        max_count2 = 0
        framex2 = []
        county2 = []
```

```
            max2 = []

            avg_acc2_list = []

            max_avg_acc2_list = []

            max_acc2 = 0

            max_avg_acc2 = 0


            video_path = filename2

            if (video_path == ""):

                mbox.showerror("Error", "No Video File Selected!", parent
= windowv)

                return

            info1.config(text="Status : Detecting...")

            #
info2.config(text="                                                  ")

            mbox.showinfo("Status", "Detecting, Please Wait...",
parent=windowv)

            # time.sleep(1)


            args = argsParser()

            writer = None

            if args['output'] is not None:

                writer = cv2.VideoWriter(args['output'],
cv2.VideoWriter_fourcc(*'MJPG'), 10, (600, 600))


            detectByPathVideo(video_path, writer)
```

28. main detection process in video start from here:

```
        def detectByPathVideo(path, writer):

            global filename2, max_count2, framex2, county2, max2,
avg_acc2_list, max_avg_acc2_list, max_acc2, max_avg_acc2

            max_count2 = 0

            framex2 = []

            county2 = []

            max2 = []

            avg_acc2_list = []

            max_avg_acc2_list = []

            max_acc2 = 0

            max_avg_acc2 = 0
```

29. Enumeration Graph ( ):

```
            def vid_enumeration_plot():

                plt.figure(facecolor='orange', )

                ax = plt.axes()

                ax.set_facecolor("yellow")

                plt.plot(framex2, county2, label = "Human Count", color =
"green", marker='o', markerfacecolor='blue')

                plt.plot(framex2, max2, label="Max. Human Count",
linestyle='dashed', color='fuchsia')

                plt.xlabel('Time (sec)')

                plt.ylabel('Human Count')

                plt.title('Enumeration Plot')

                plt.legend()

                plt.get_current_fig_manager().canvas.set_window_title("Pl
ot for Video")

                plt.show()
```

30. Accuracy Graph ( ):

```
def vid_accuracy_plot():
    plt.figure(facecolor='orange', )
    ax = plt.axes()
    ax.set_facecolor("yellow")
    plt.plot(framex2, avg_acc2_list, label="Avg. Accuracy", color="green", marker='o', markerfacecolor='blue')
    plt.plot(framex2, max_avg_acc2_list, label="Max. Avg. Accuracy", linestyle='dashed', color='fuchsia')
    plt.xlabel('Time (sec)')
    plt.ylabel('Avg. Accuracy')
    plt.title('Avg. Accuracy Plot')
    plt.legend()
    plt.get_current_fig_manager().canvas.set_window_title("Plot for Video")
    plt.show()
```

31. PDF generation:

```
def vid_gen_report():
    pdf = FPDF(orientation='P', unit='mm', format='A4')
    pdf.add_page()
    pdf.set_font("Arial", "", 20)
    pdf.set_text_color(128, 0, 0)
    pdf.image('Images/Crowd_Report.png', x=0, y=0, w=210, h=297)


    pdf.text(125, 150, str(max_count2))
```

```
                pdf.text(105, 163, str(max_acc2))

                pdf.text(125, 175, str(max_avg_acc2))

                if(max_count2>25):

                    pdf.text(26, 220, "Max. Human Detected is greater
than MAX LIMIT.")

                    pdf.text(70, 235, "Region is Crowded.")

                else:

                    pdf.text(26, 220, "Max. Human Detected is in range of
MAX LIMIT.")

                    pdf.text(65, 235, "Region is not Crowded.")



                pdf.output('Crowd_Report.pdf')

                mbox.showinfo("Status", "Report Generated and Saved
Successfully.", parent = windowv)


video = cv2.VideoCapture(path)

            odapi = DetectorAPI()

            threshold = 0.7

            check, frame = video.read()

            if check == False:

                print('Video Not Found. Please Enter a Valid Path (Full
path of Video Should be Provided).')

                return



            x2 = 0

            while video.isOpened():

                # check is True if reading was successful
```

```
                    check, frame = video.read()

                    if(check==True):

                        img = cv2.resize(frame, (800, 500))

                        boxes, scores, classes, num = odapi.processFrame(img)

                        person = 0

                        acc = 0

                        for i in range(len(boxes)):

                            # print(boxes)

                            # print(scores)

                            # print(classes)

                            # print(num)

                            # print()

                            if classes[i] == 1 and scores[i] > threshold:

                                box = boxes[i]

                                person += 1

                                cv2.rectangle(img, (box[1], box[0]), (box[3],
box[2]), (255, 0, 0), 2)  # cv2.FILLED

                                cv2.putText(img, f'P{person,
round(scores[i],2)}', (box[1]-30, box[0]-8), cv2.FONT_HERSHEY_COMPLEX,
0.5, (0,0,255), 1 )#(75,0,130),

                                acc+=scores[i]

                                if(scores[i]>max_acc2):

                                    max_acc2 = scores[i]


                        if(person>max_count2):

                            max_count2 = person

                        county2.append(person)
                        x2+=1
                        framex2.append(x2)
                        if(person>=1):
                            avg_acc2_list.append(acc/person)
                            if((acc/person)>max_avg_acc2):
                                max_avg_acc2 = (acc/person)
                        else:
                            avg_acc2_list.append(acc)

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

                        cv2.imshow("Human Detection from Video", img)
                        key = cv2.waitKey(1)
                        if key & 0xFF == ord('q'):
                            break
                    else:
                        break

                video.release()
                info1.config(text="
            ")
                #info2.config(text="
            ")
                info1.config(text="Status : Detection & Counting Completed")
                # info2.config(text="Max. Human Count : " + str(max_count2))
```

```
                cv2.destroyAllWindows()

                for i in range(len(framex2)):
                    max2.append(max_count2)
                    max_avg_acc2_list.append(max_avg_acc2)
```

32. Enumeration Plot Button:

```
Button(windowv, text="Enumeration\nPlot", command=vid_enumeration_plot, cursor="hand2", font=("Arial", 20),bg="orange", fg="blue").place(x=100, y=530)
```

33. Avg. Accuracy Plot Button:

```
Button(windowv, text="Avg. Accuracy\nPlot", command=vid_accuracy_plot, cursor="hand2", font=("Arial", 20),bg="orange", fg="blue").place(x=700, y=530)
```

34. Generate Crowd Report Button:

```
Button(windowv, text="Generate  Crowd  Report", command=vid_gen_report, cursor="hand2", font=("Arial", 20),bg="gray", fg="blue").place(x=325, y=550)
```

35. Preview Image:

```
def prev_vid():
    global filename2
    cap = cv2.VideoCapture(filename2)
    while (cap.isOpened()):
        ret, frame = cap.read()
        if ret == True:
            img = cv2.resize(frame, (800, 500))
```

```
                    cv2.imshow('Selected Video Preview', img)

                    if cv2.waitKey(25) & 0xFF == ord('q'):

                        break

                else:

                    break

            cap.release()

            cv2.destroyAllWindows()
```

36. For videos:

```
        lbl1 = tk.Label(windowv, text="DETECT  FROM\nVIDEO", font=("Arial",
   50, "underline"), fg="brown")

        lbl1.place(x=230, y=20)

        lbl2 = tk.Label(windowv, text="Selected Video", font=("Arial", 30),
   fg="green")

        lbl2.place(x=80, y=200)

        path_text2 = tk.Text(windowv, height=1, width=37, font=("Arial",
   30), bg="light yellow", fg="orange", borderwidth=2,relief="solid")

        path_text2.place(x=80, y=260)
```

#BUTTONS FOR VIDEOS WINDOW

37. Select button:

```
    Button(windowv, text="SELECT", command=open_vid, cursor="hand2",
    font=("Arial", 20), bg="light green", fg="blue").place(x=220, y=350)
```

38. Preview button:

```
    Button(windowv, text="PREVIEW", command=prev_vid, cursor="hand2",
    font=("Arial", 20), bg="yellow", fg="blue").place(x=410, y=350)
```

39. Detect button:

```
Button(windowv, text="DETECT", command=det_vid, cursor="hand2",
font=("Arial", 20), bg="orange", fg="blue").place(x=620, y=350)


 info1 = tk.Label(windowv, font=("Arial", 30), fg="gray")  # same way bg

 info1.place(x=100, y=440)

 # info2 = tk.Label(windowv, font=("Arial", 30), fg="gray")  # same way bg

 # info2.place(x=100, y=500)
```

40. Exit window:

```
        def exit_winv():

            if mbox.askokcancel("Exit", "Do you want to exit?", parent =
windowv):

                windowv.destroy()

        windowv.protocol("WM_DELETE_WINDOW", exit_winv)
```

41. Camera Section:

```
        def camera_option():

            # new windowc created for camera section

            windowc = tk.Tk()

            windowc.title("Human Detection from Camera")

            windowc.iconbitmap('Images/icon.ico')

            windowc.geometry('1000x700')


            max_count3 = 0

            framex3 = []

            county3 = []
            max3 = []

            avg_acc3_list = []

            max_avg_acc3_list = []

            max_acc3 = 0

            max_avg_acc3 = 0
```

42. function defined to open the Camera:

```
        def open_cam():

            global max_count3, framex3, county3, max3, avg_acc3_list,
    max_avg_acc3_list, max_acc3, max_avg_acc3

            max_count3 = 0

            framex3 = []

            county3 = []

            max3 = []

            avg_acc3_list = []

            max_avg_acc3_list = []

            max_acc3 = 0

            max_avg_acc3 = 0


            args = argsParser()


            info1.config(text="Status : Opening Camera...")

            #
    info2.config(text="                                                  ")

            mbox.showinfo("Status", "Opening Camera...Please Wait...",
    parent=windowc)

            # time.sleep(1)
```

```
                writer = None

                if args['output'] is not None:

                    writer = cv2.VideoWriter(args['output'],
    cv2.VideoWriter_fourcc(*'MJPG'), 10, (600, 600))

                if True:

                    detectByCamera(writer)
```

43.function defined to detect the camera:

```
            def detectByCamera(writer):

                global max_count3, framex3, county3, max3, avg_acc3_list,
    max_avg_acc3_list, max_acc3, max_avg_acc3

                max_count3 = 0

                framex3 = []

                county3 = []

                max3 = []

                avg_acc3_list = []

                max_avg_acc3_list = []

                max_acc3 = 0

                max_avg_acc3 = 0
```

44. Enumeration Graph ( ):

```
            def cam_enumeration_plot():

                plt.figure(facecolor='orange', )

                ax = plt.axes()

                ax.set_facecolor("yellow")
                plt.plot(framex3, county3, label="Human Count",
color="green", marker='o', markerfacecolor='blue')

                plt.plot(framex3, max3, label="Max. Human Count",
linestyle='dashed', color='fuchsia')

                plt.xlabel('Time (sec)')

                plt.ylabel('Human Count')

                plt.legend()

                plt.title("Enumeration Plot")

                plt.get_current_fig_manager().canvas.set_window_title("Plot
for Camera")

                plt.show()
```

45. Accuracy Graph ( ):

```
            def cam_accuracy_plot():

                plt.figure(facecolor='orange', )

                ax = plt.axes()

                ax.set_facecolor("yellow")

                plt.plot(framex3, avg_acc3_list, label="Avg. Accuracy",
color="green", marker='o', markerfacecolor='blue')

                plt.plot(framex3, max_avg_acc3_list, label="Max. Avg.
Accuracy", linestyle='dashed', color='fuchsia')

                plt.xlabel('Time (sec)')

                plt.ylabel('Avg. Accuracy')

                plt.title('Avg. Accuracy Plot')

                plt.legend()

                plt.get_current_fig_manager().canvas.set_window_title("Plot
for Camera")

                plt.show()
```

46. PDF generation:

```
              def cam_gen_report():
                  pdf = FPDF(orientation='P', unit='mm', format='A4')
                  pdf.add_page()
                  pdf.set_font("Arial", "", 20)
                  pdf.set_text_color(128, 0, 0)
                  pdf.image('Images/Crowd_Report.png', x=0, y=0, w=210, h=297)

                  pdf.text(125, 150, str(max_count3))
                  pdf.text(105, 163, str(max_acc3))
                  pdf.text(125, 175, str(max_avg_acc3))
                  if (max_count3 > 25):
                      pdf.text(26, 220, "Max. Human Detected is greater than
  MAX LIMIT.")
                      pdf.text(70, 235, "Region is Crowded.")
                  else:
                      pdf.text(26, 220, "Max. Human Detected is in range of
  MAX LIMIT.")
                      pdf.text(65, 235, "Region is not Crowded.")

                  pdf.output('Crowd_Report.pdf')
                  mbox.showinfo("Status", "Report Generated and Saved
  Successfully.", parent = windowc)

   video = cv2.VideoCapture(0)
              odapi = DetectorAPI()
              threshold = 0.7
```

```
            x3 = 0
            while True:
                check, frame = video.read()
                img = cv2.resize(frame, (800, 600))
                boxes, scores, classes, num = odapi.processFrame(img)
                person = 0
                acc = 0
                for i in range(len(boxes)):

                    if classes[i] == 1 and scores[i] > threshold:
                        box = boxes[i]
                        person += 1
                        cv2.rectangle(img, (box[1], box[0]), (box[3],
box[2]), (255, 0, 0), 2)  # cv2.FILLED
                        cv2.putText(img, f'P{person, round(scores[i], 2)}',
(box[1] - 30, box[0] - 8),cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 255), 1)  #
(75,0,130),
                        acc += scores[i]
                        if (scores[i] > max_acc3):
                            max_acc3 = scores[i]

                if (person > max_count3):
                    max_count3 = person

                if writer is not None:
                    writer.write(img)
```

```
    cv2.imshow("Human Detection from Camera", img)
    key = cv2.waitKey(1)
    if key & 0xFF == ord('q'):
        break

    county3.append(person)
    x3 += 1
    framex3.append(x3)
    if(person>=1):
        avg_acc3_list.append(acc / person)
        if ((acc / person) > max_avg_acc3):
            max_avg_acc3 = (acc / person)
    else:
        avg_acc3_list.append(acc)

video.release()
info1.config(text="                                       ")
#info2.config(text="                                      ")
info1.config(text="Status : Detection & Counting Completed")
# info2.config(text="Max. Human Count : " + str(max_count3))
cv2.destroyAllWindows()
for i in range(len(framex3)):
    max3.append(max_count3)
    max_avg_acc3_list.append(max_avg_acc3)
```

47. Enumeration Plot Button:

```
Button(windowc, text="Enumeration\nPlot", command=cam_enumeration_plot, cursor="hand2", font=("Arial", 20),bg="orange", fg="blue").place(x=100, y=530)
```

48. Avg. Accuracy Plot Button:

```
Button(windowc, text="Avg. Accuracy\nPlot", command=cam_accuracy_plot, cursor="hand2", font=("Arial", 20),bg="orange", fg="blue").place(x=700, y=530)
```

49. Generate Crowd Report Button:

```
Button(windowc, text="Generate  Crowd  Report", command=cam_gen_report, cursor="hand2", font=("Arial", 20),bg="gray", fg="blue").place(x=325, y=550)
```

50. For Camera:

```
lbl1 = tk.Label(windowc, text="DETECT  FROM\nCAMERA", font=("Arial", 50, "underline"), fg="brown")  # same way bg

        lbl1.place(x=230, y=20)
```

#BUTTONS FOR CAMERA WINDOW

51. Open Camera button:

```
Button(windowc, text="OPEN CAMERA", command=open_cam, cursor="hand2", font=("Arial", 20), bg="light green", fg="blue").place(x=370, y=230)

  info1 = tk.Label(windowc, font=("Arial", 30), fg="gray")  # same way bg

  info1.place(x=100, y=330)

  # info2 = tk.Label(windowc, font=("Arial", 30), fg="gray") #same way bg

  # info2.place(x=100, y=390)
```

52. Exit window:

```
  def exit_winc():
```

```
            if mbox.askokcancel("Exit", "Do you want to exit?", parent =
windowc):

                windowc.destroy()

        windowc.protocol("WM_DELETE_WINDOW", exit_winc)


    lbl1 = tk.Label(text="OPTIONS", font=("Arial", 50,
"underline"),fg="brown")  # same way bg

    lbl1.place(x=340, y=20)
```

53. image on the main window

```
      pathi = "Images/image1.jpg"

       imgi = ImageTk.PhotoImage(Image.open(pathi))

       paneli = tk.Label(window1, image = imgi)

       paneli.place(x = 90, y = 110)

       pathv = "Images/image2.png"

       imgv = ImageTk.PhotoImage(Image.open(pathv))

       panelv = tk.Label(window1, image = imgv)

       panelv.place(x = 700, y = 260)# 720, 260

      pathc = "Images/image3.jpg"

       imgc = ImageTk.PhotoImage(Image.open(pathc))

       panelc = tk.Label(window1, image = imgc)

       panelc.place(x = 90, y = 415)
```

54. Detect from Image Button:

```
      Button(window1, text="DETECT  FROM   IMAGE →",command=image_option,
      cursor="hand2", font=("Arial",30), bg = "light green", fg =
      "blue").place(x = 350, y = 150)
```

55. Detect from Video Button:

```
Button(window1, text="DETECT  FROM  VIDEO ⇥",command=video_option,
cursor="hand2", font=("Arial", 30), bg = "light blue", fg =
"blue").place(x = 110, y = 300) #90, 300
```

56. Detect from Camera Button:

```
Button(window1, text="DETECT FROM CAMERA ⇥",command=camera_option,
cursor="hand2", font=("Arial", 30), bg = "light green", fg =
"blue").place(x = 350, y = 450)
```

57. Function to Exit main window:

```
 def exit_win1():

     if mbox.askokcancel("Exit", "Do you want to exit?"):

         window1.destroy()
```

58. Exit button:

```
Button(window1, text="❌ EXIT",command=exit_win1,  cursor="hand2",
font=("Arial", 25), bg = "red", fg = "blue").place(x = 440, y = 600)


window1.protocol("WM_DELETE_WINDOW", exit_win1)

window1.mainloop()
```

Persondetection.py

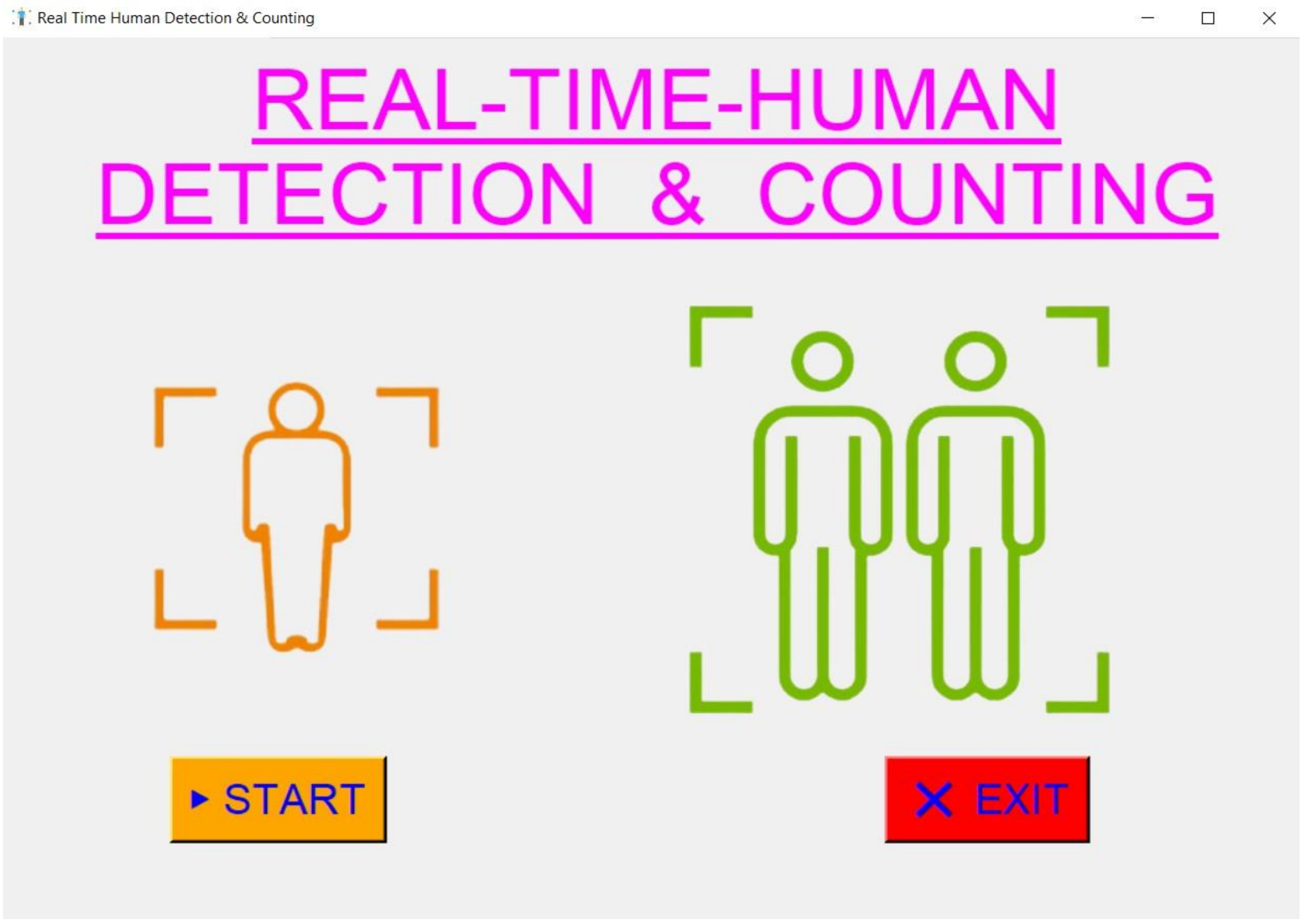
```
import numpy as np

import tensorflow as tf

# import cv2

import time

import os

# import tensorflow.compat.v1 as tf

import tensorflow._api.v2.compat.v1 as tf

tf.disable_v2_behavior()


class DetectorAPI:

    def __init__(self):

        path = os.path.dirname(os.path.realpath(__file__))

        self.path_to_ckpt = f'frozen_inference_graph.pb'

        self.detection_graph = tf.Graph()


        with self.detection_graph.as_default():

            od_graph_def = tf.GraphDef()

            with tf.gfile.GFile(self.path_to_ckpt, 'rb') as fid:

                serialized_graph = fid.read()

                od_graph_def.ParseFromString(serialized_graph)

                tf.import_graph_def(od_graph_def, name='')


        self.default_graph = self.detection_graph.as_default()

        self.sess = tf.Session(graph=self.detection_graph)
        # Definite input and output Tensors for detection_graph

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

        # Each box represents a part of the image where a particular object was
detected.

        self.detection_boxes =
self.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.

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

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

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



    def processFrame(self, image):

        # Expand dimensions since the trained_model expects images to have
shape: [1, None, None, 3]

        image_np_expanded = np.expand_dims(image, axis=0)

        # Actual detection.

        start_time = time.time()

        (boxes, scores, classes, num) = self.sess.run(

            [self.detection_boxes, self.detection_scores,

                self.detection_classes, self.num_detections],

            feed_dict={self.image_tensor: image_np_expanded})

        end_time = time.time()
        # print("Elapsed Time:", end_time-start_time)

        # print(self.image_tensor, image_np_expanded)

        im_height, im_width, _ = image.shape

        boxes_list = [None for i in range(boxes.shape[1])]


        for i in range(boxes.shape[1]):

            boxes_list[i] = (int(boxes[0, i, 0] * im_height),int(boxes[0, i,
1]*im_width),int(boxes[0, i, 2] * im_height),int(boxes[0, i, 3]*im_width))


        return boxes_list, scores[0].tolist(), [int(x) for x in
classes[0].tolist()], int(num[0])


    def close(self):

        self.sess.close()

        self.default_graph.close()
```

CHAPTER-VII

CODE OUTPUT

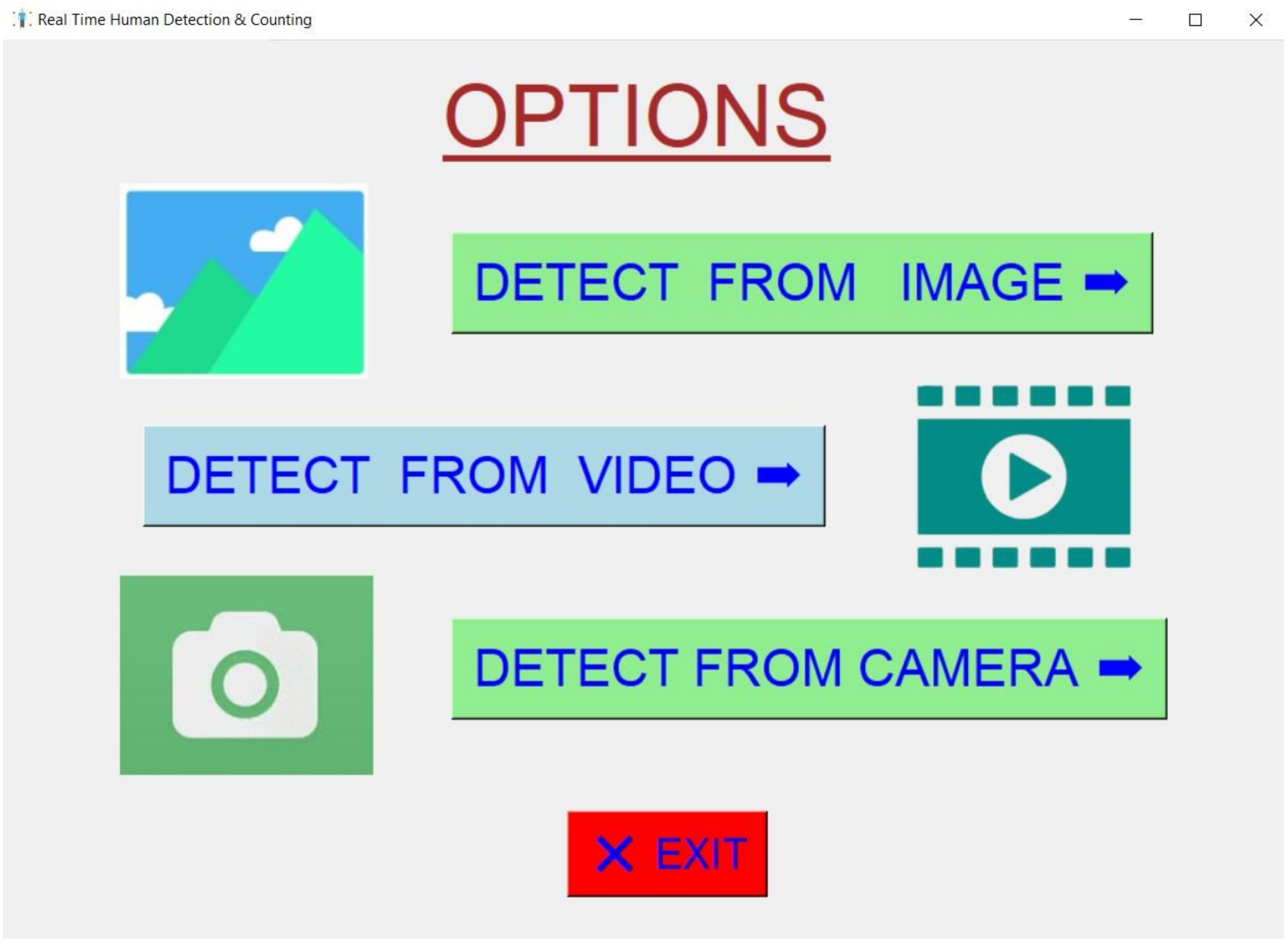
<u>Project GUI:</u>

Window 1:


Real Time Human Detection & Counting
REAL-TIME-HUMAN DETECTION & COUNTING
▸ START
✕ EXIT

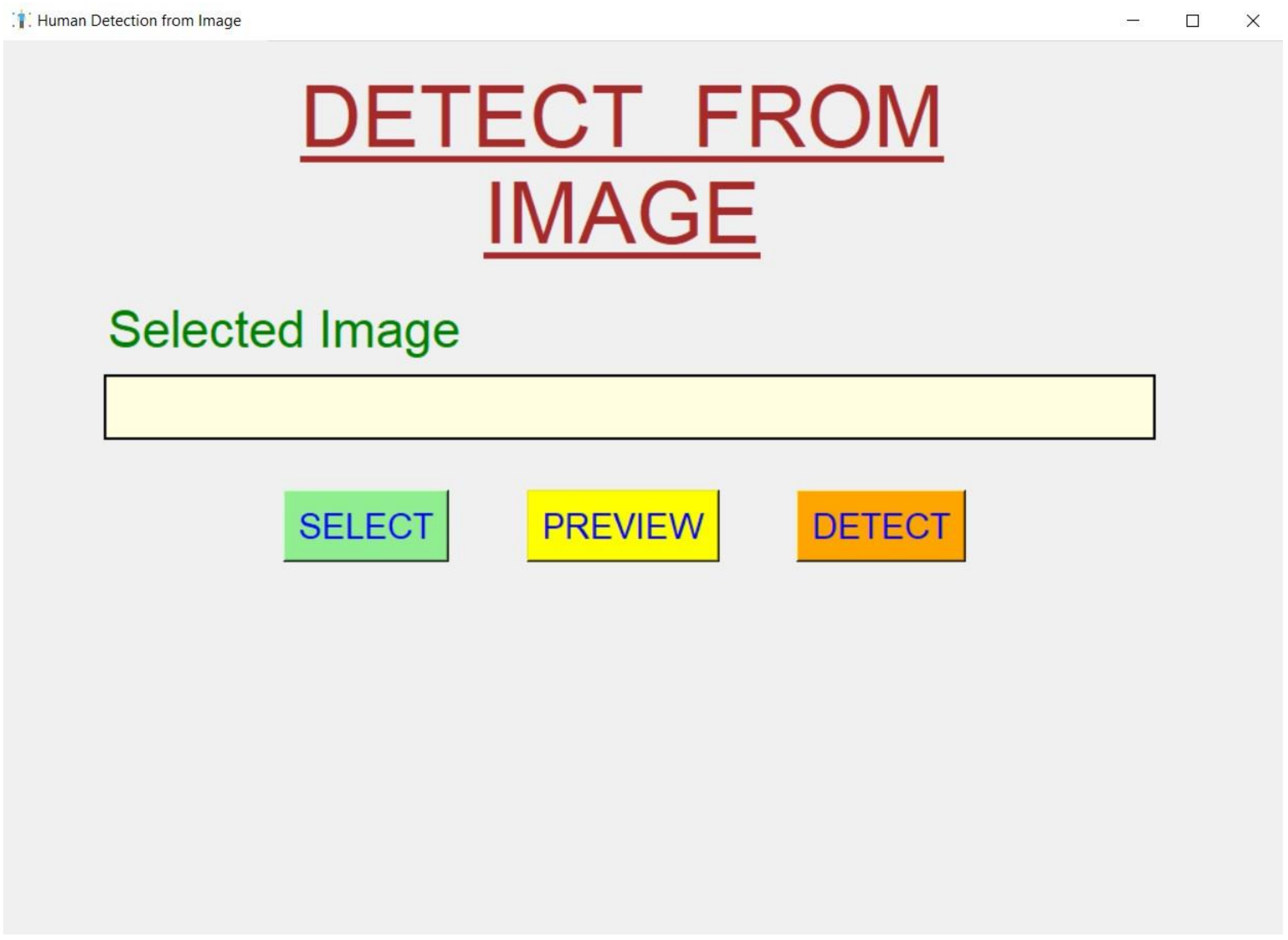
<u>Main Window</u>

Window 2



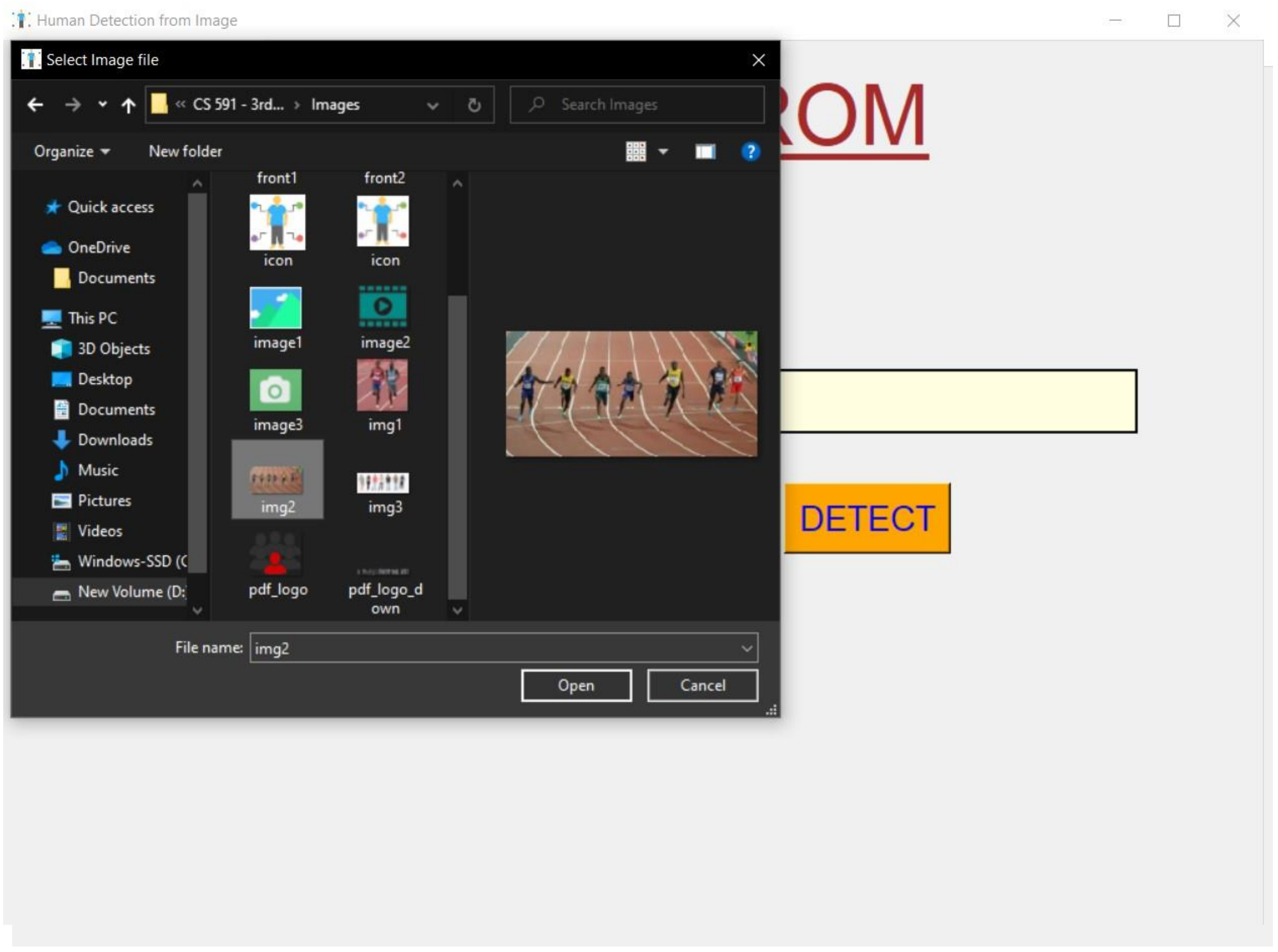
<u>Options Window</u>

Window 3



<u>Detect From Image Window</u>

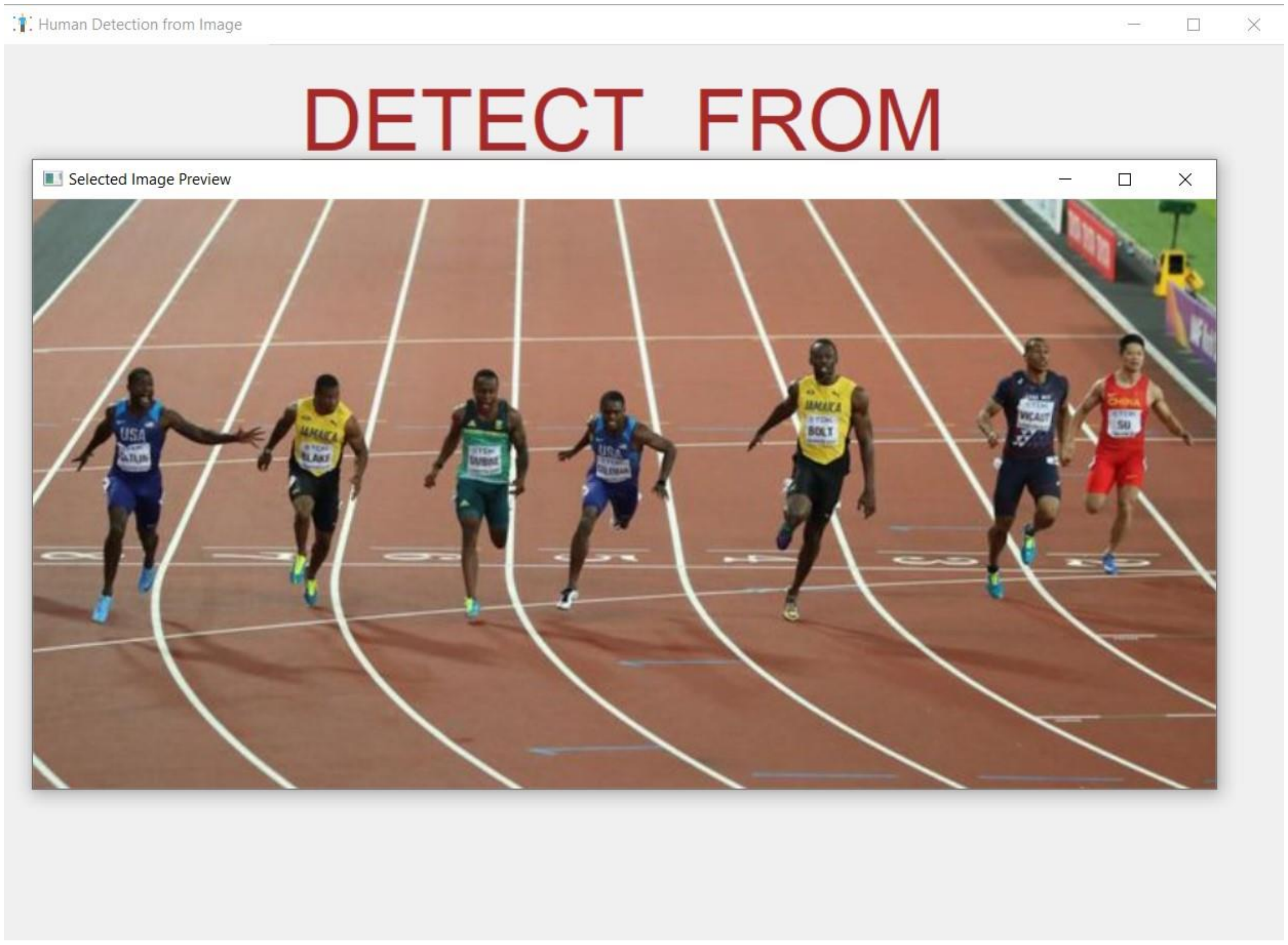
Window 4



File name: img2

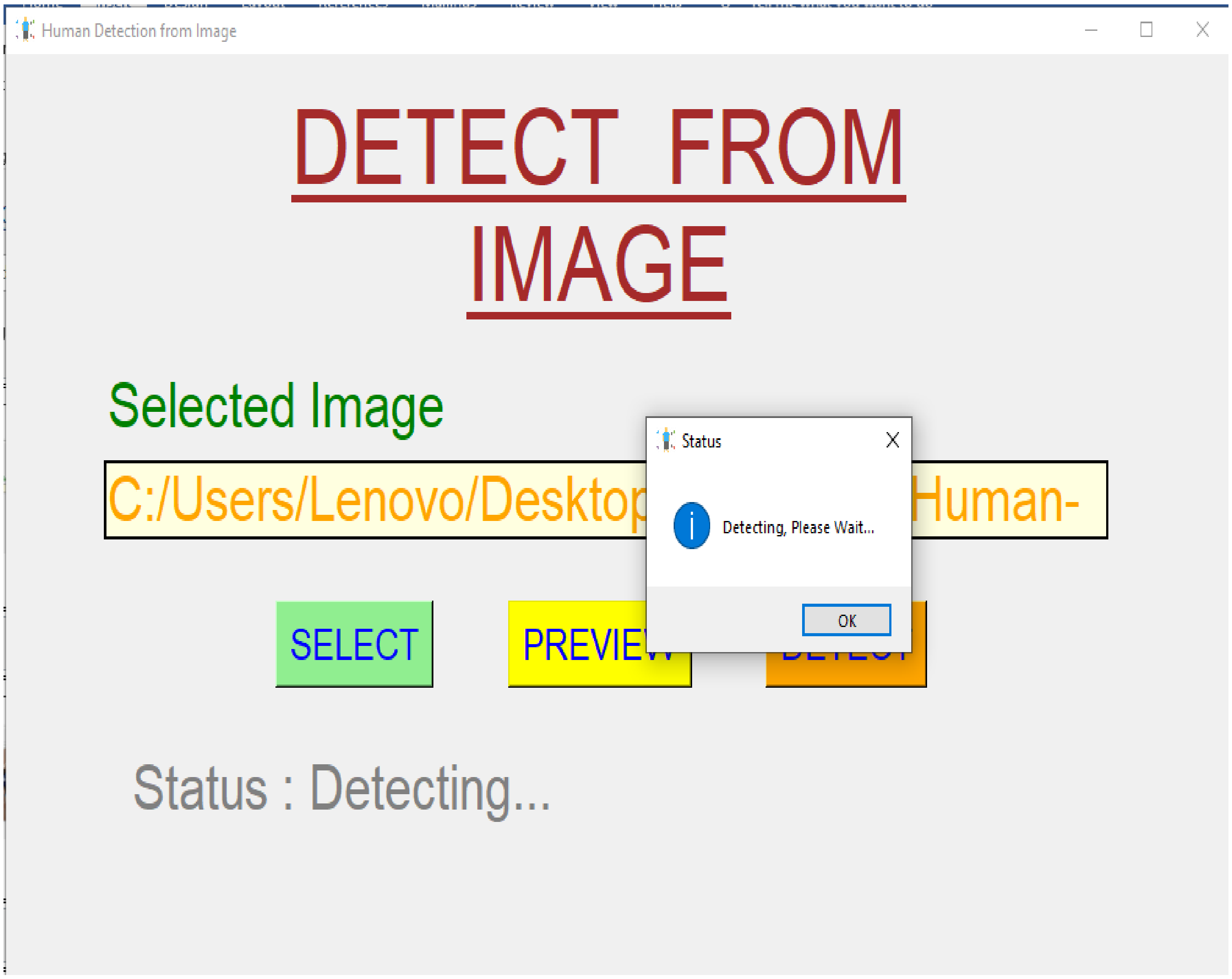
Select Images

Window 5



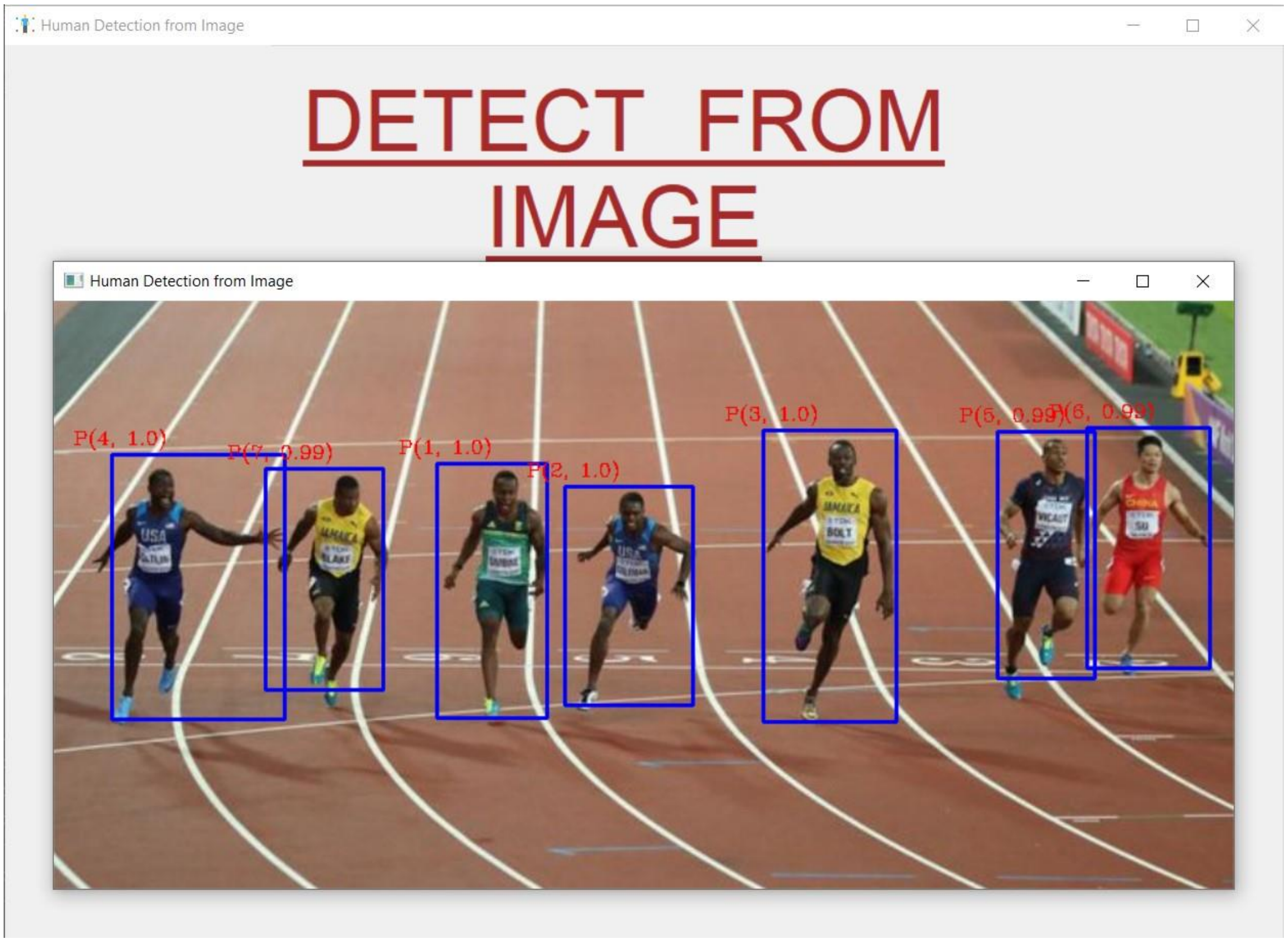
<u>Image Preview window</u>

Window 6



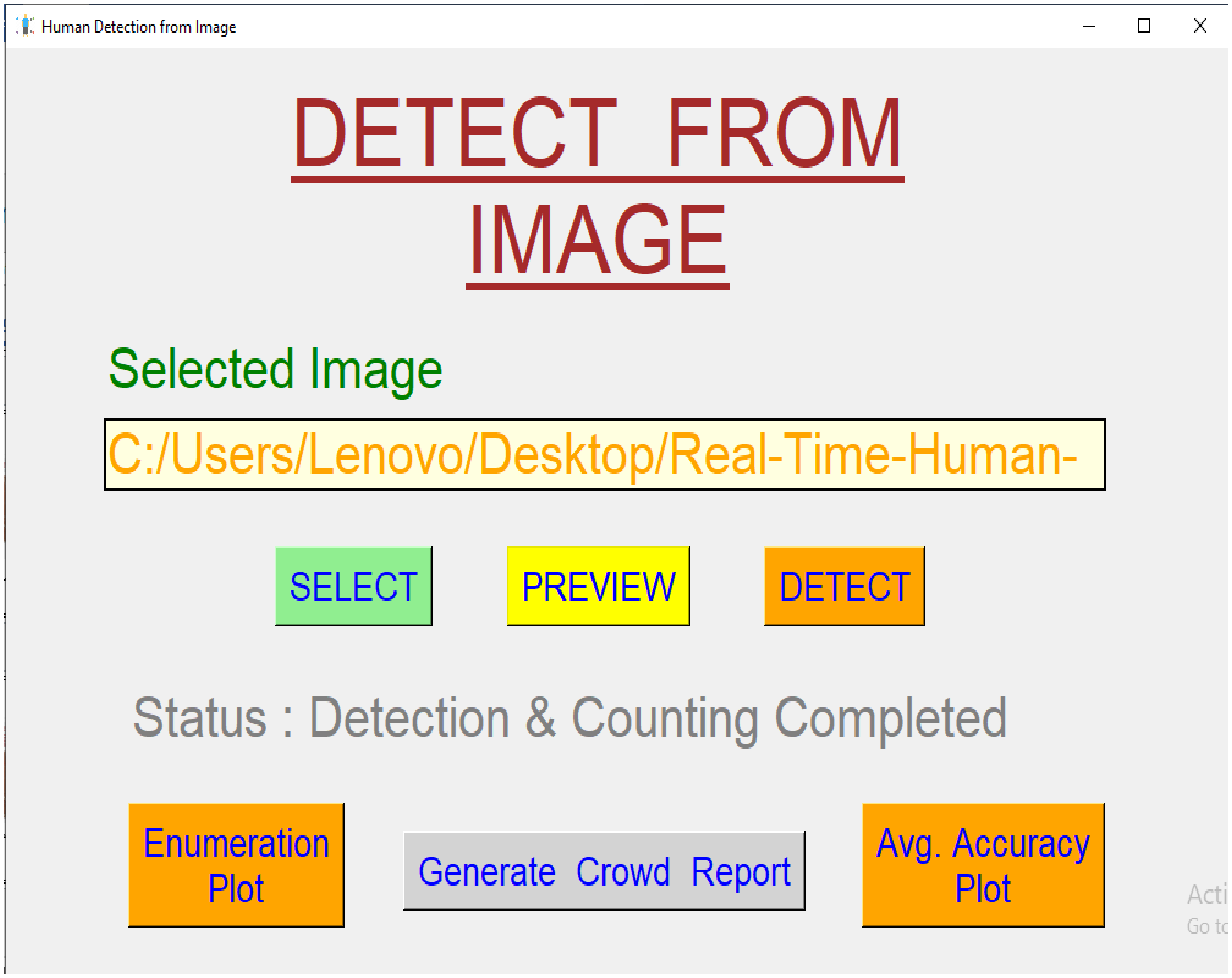
<u>Detecting Status Window</u>

Window 7



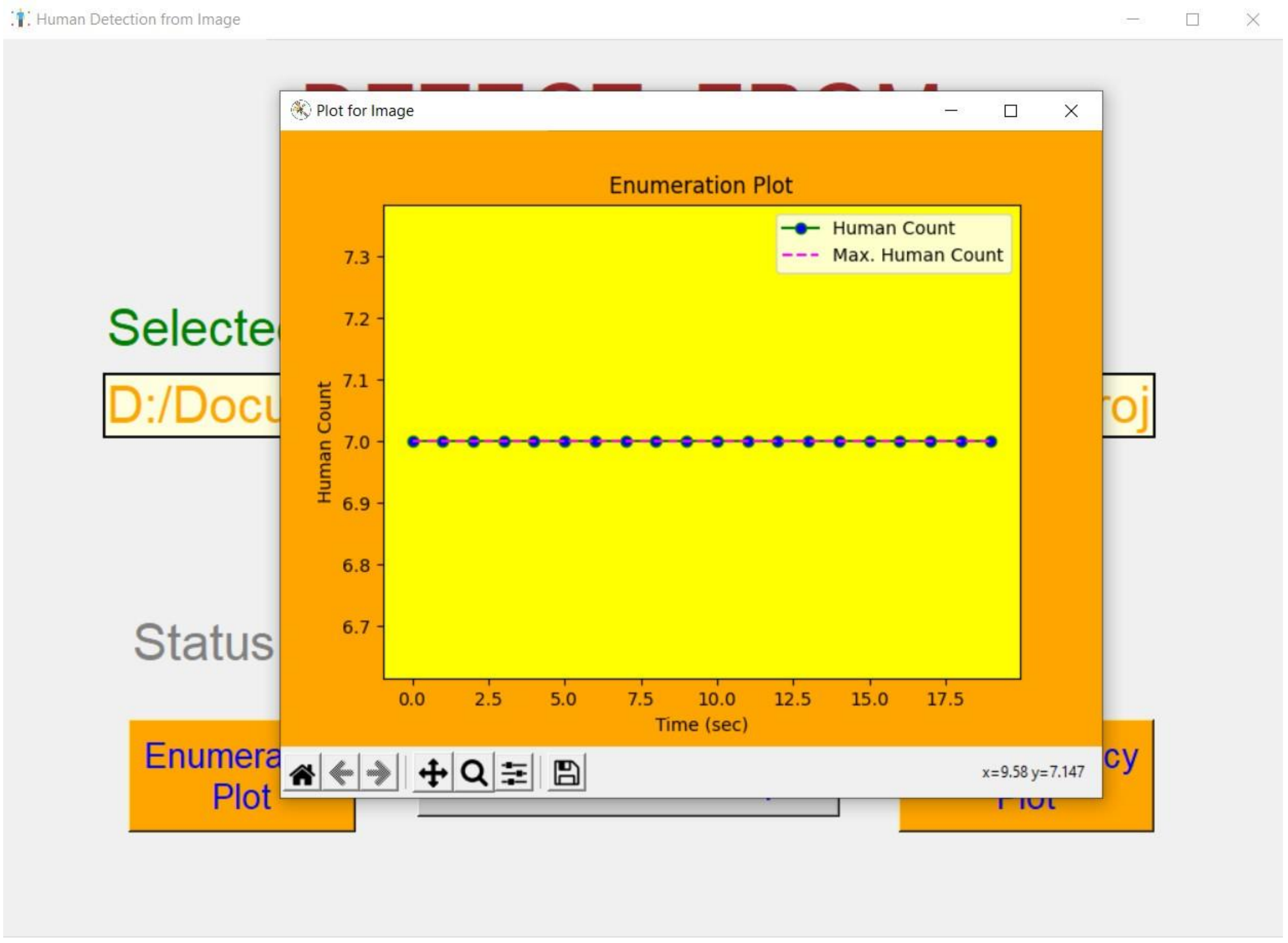
<u>Detecting Humans from Images</u>

Window 8


Human Detection from Image
DETECT FROM IMAGE
Selected Image
C:/Users/Lenovo/Desktop/Real-Time-Human-
SELECT
PREVIEW
DETECT
Status : Detection & Counting Completed
Enumeration Plot
Generate Crowd Report
Avg. Accuracy Plot

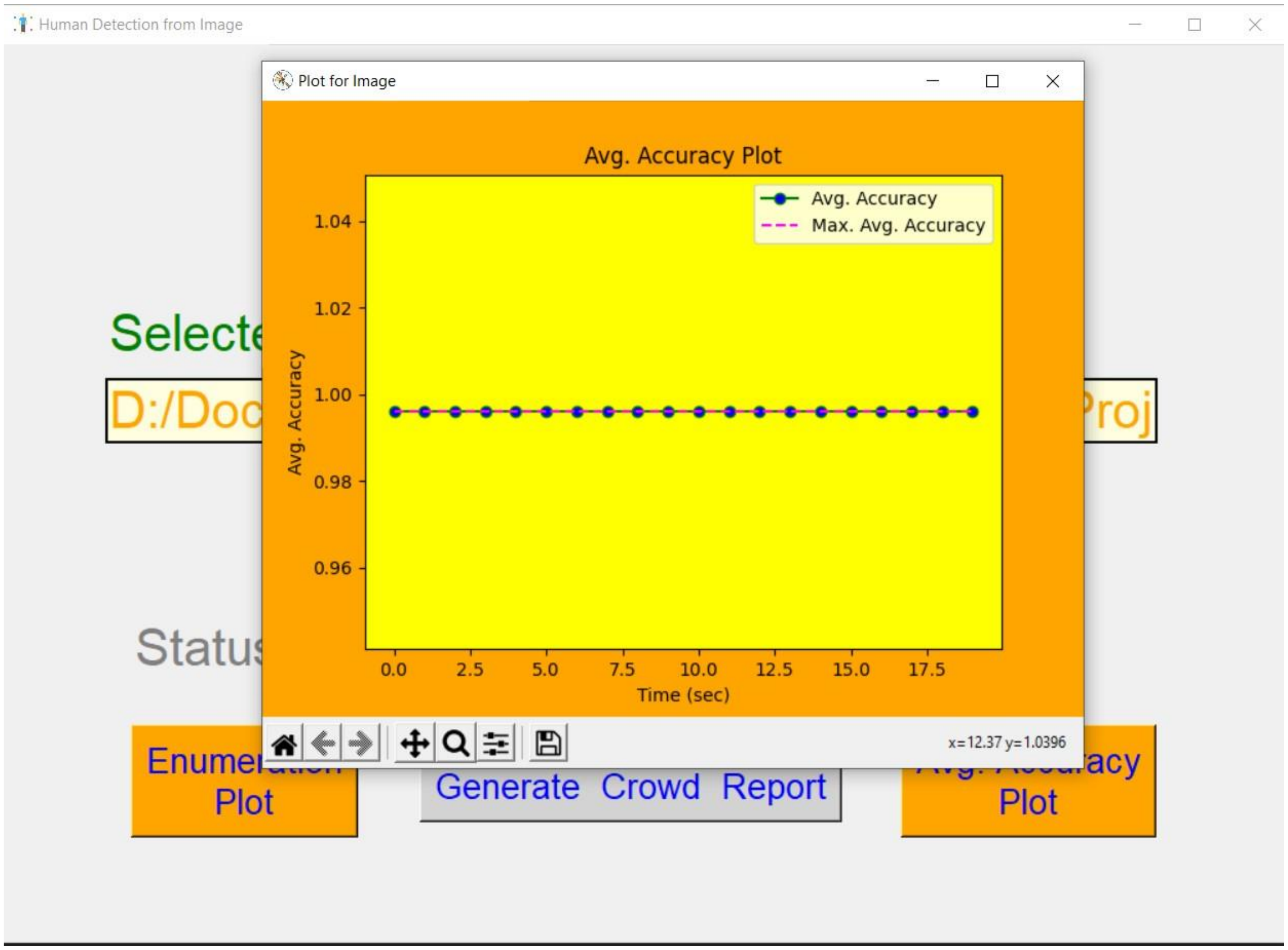
Detection Complete Window

Window 9



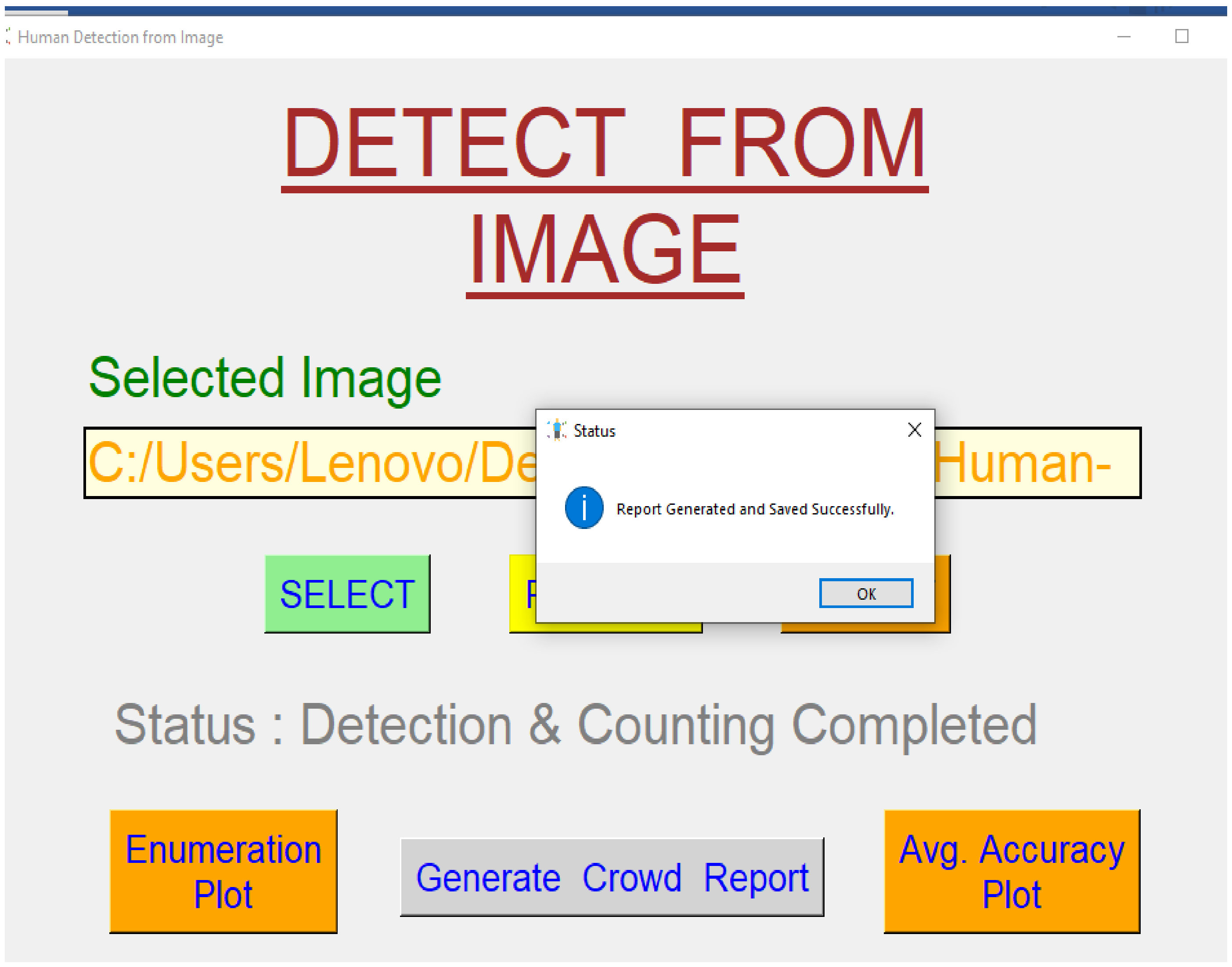
<u>Enumration Graph Plot</u>

Window 10



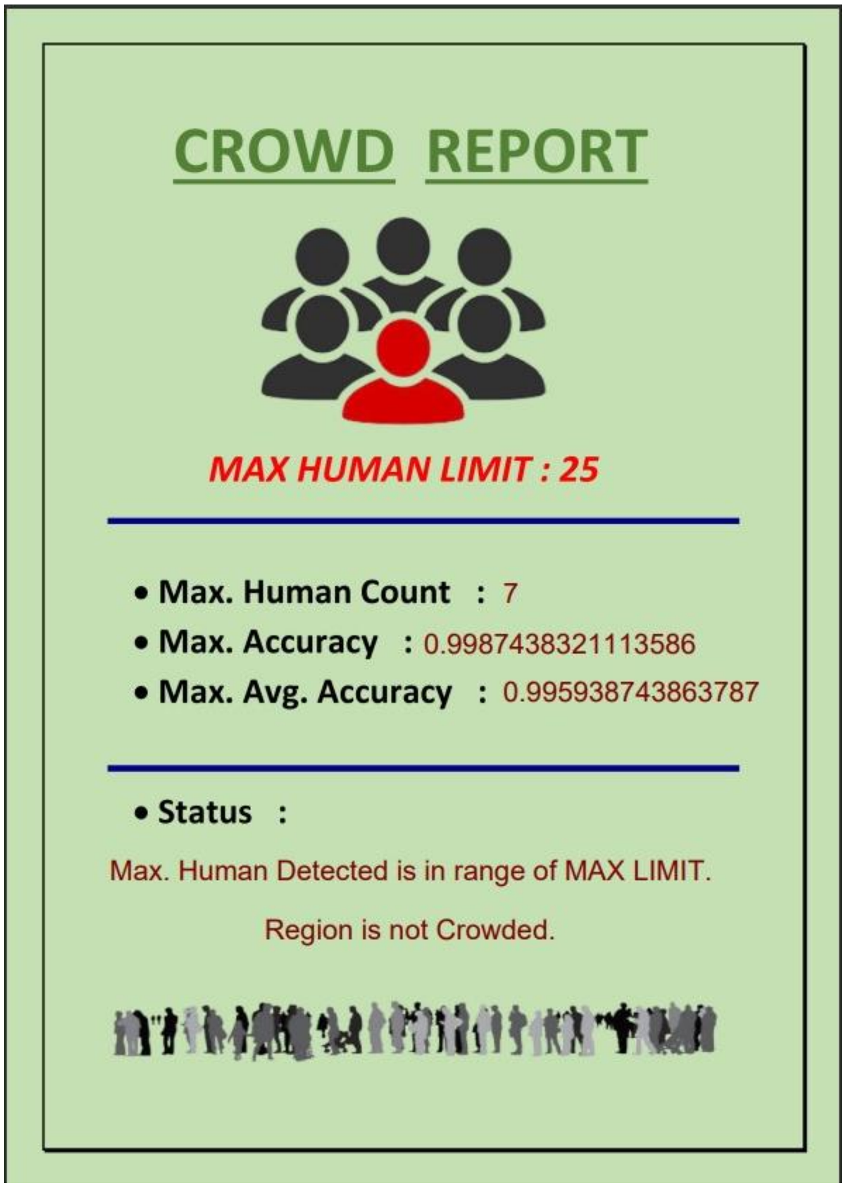
<u>Accuracy Graph plot</u>

Window 11



Crowd Report Generation Complete

Window 12



CROWD REPORT

MAX HUMAN LIMIT : 25

- Max. Human Count : 7
- Max. Accuracy : 0.9987438321113586
- Max. Avg. Accuracy : 0.995938743863787

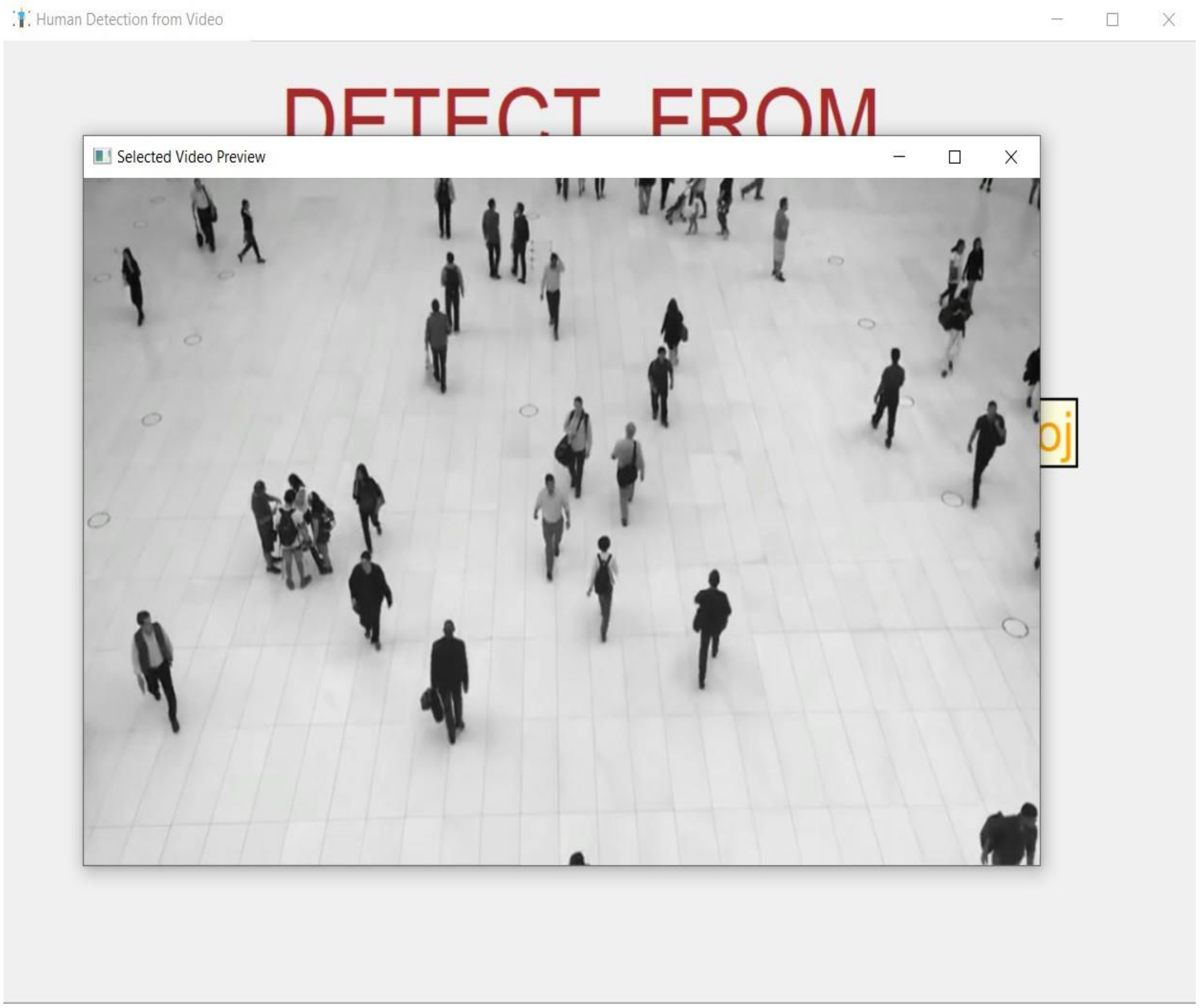
- Status :

Max. Human Detected is in range of MAX LIMIT.

Region is not Crowded.

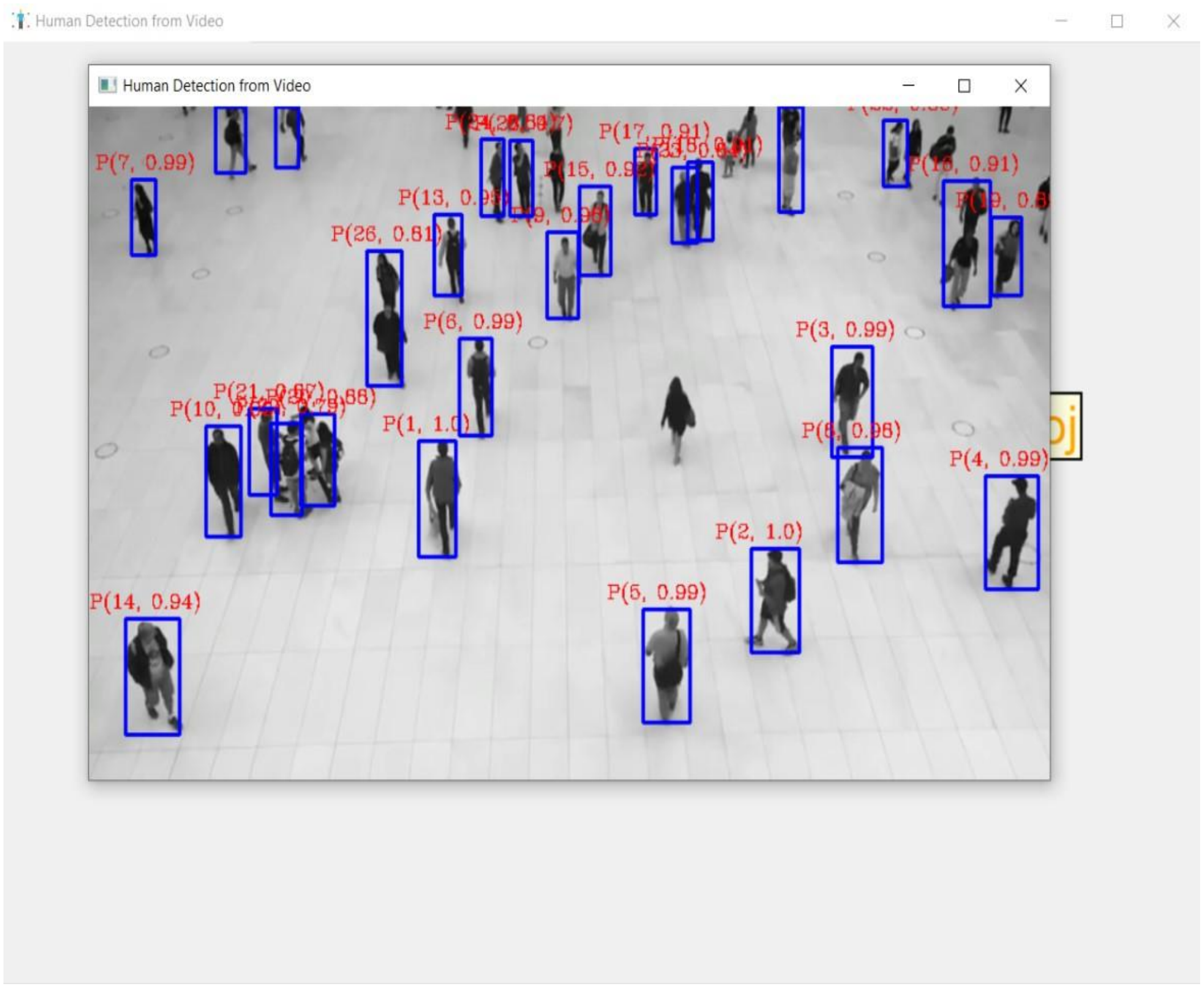
Crowd Report

Window 13



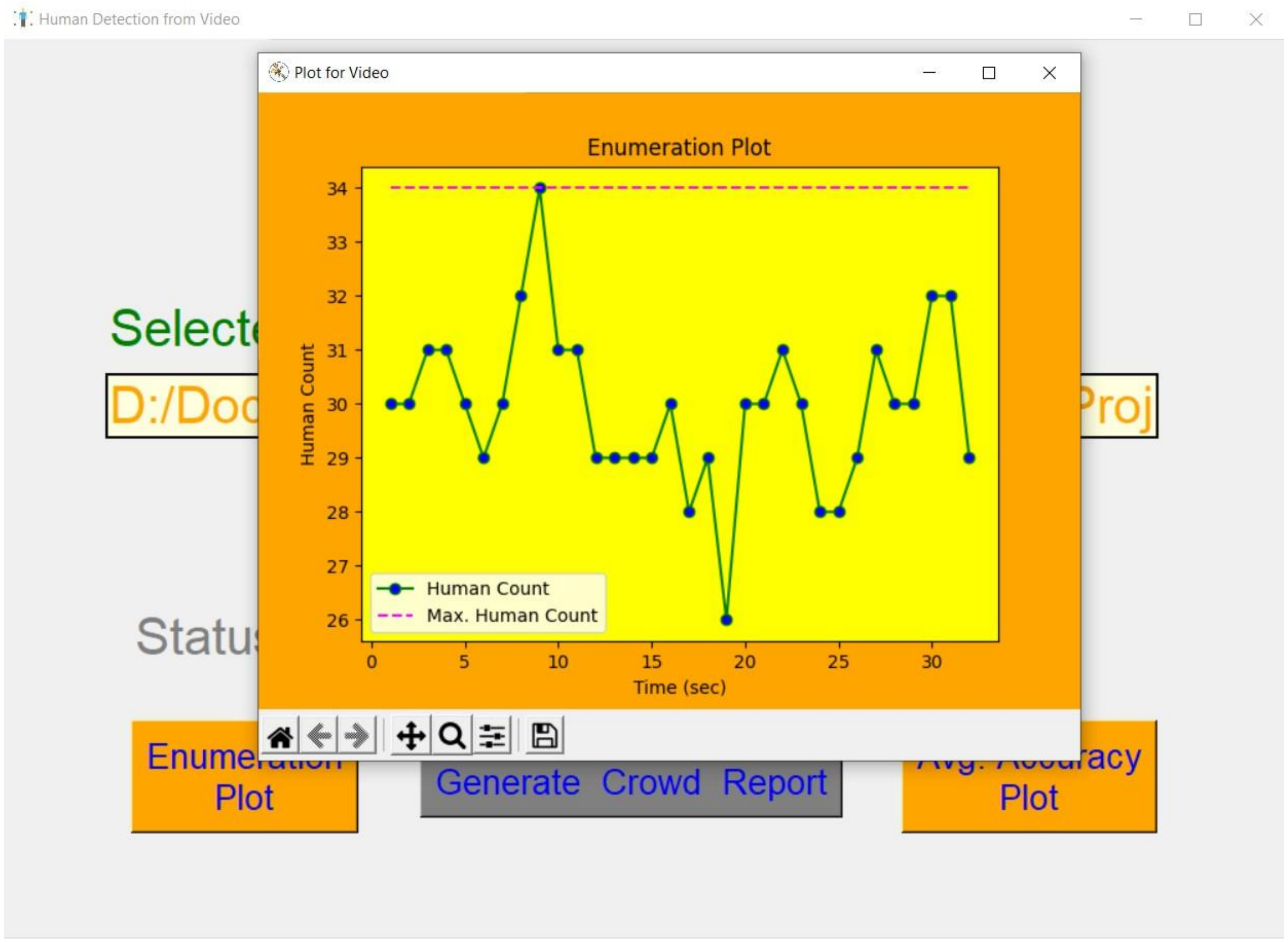
<u>Selected Video Preview window</u>

Window 14



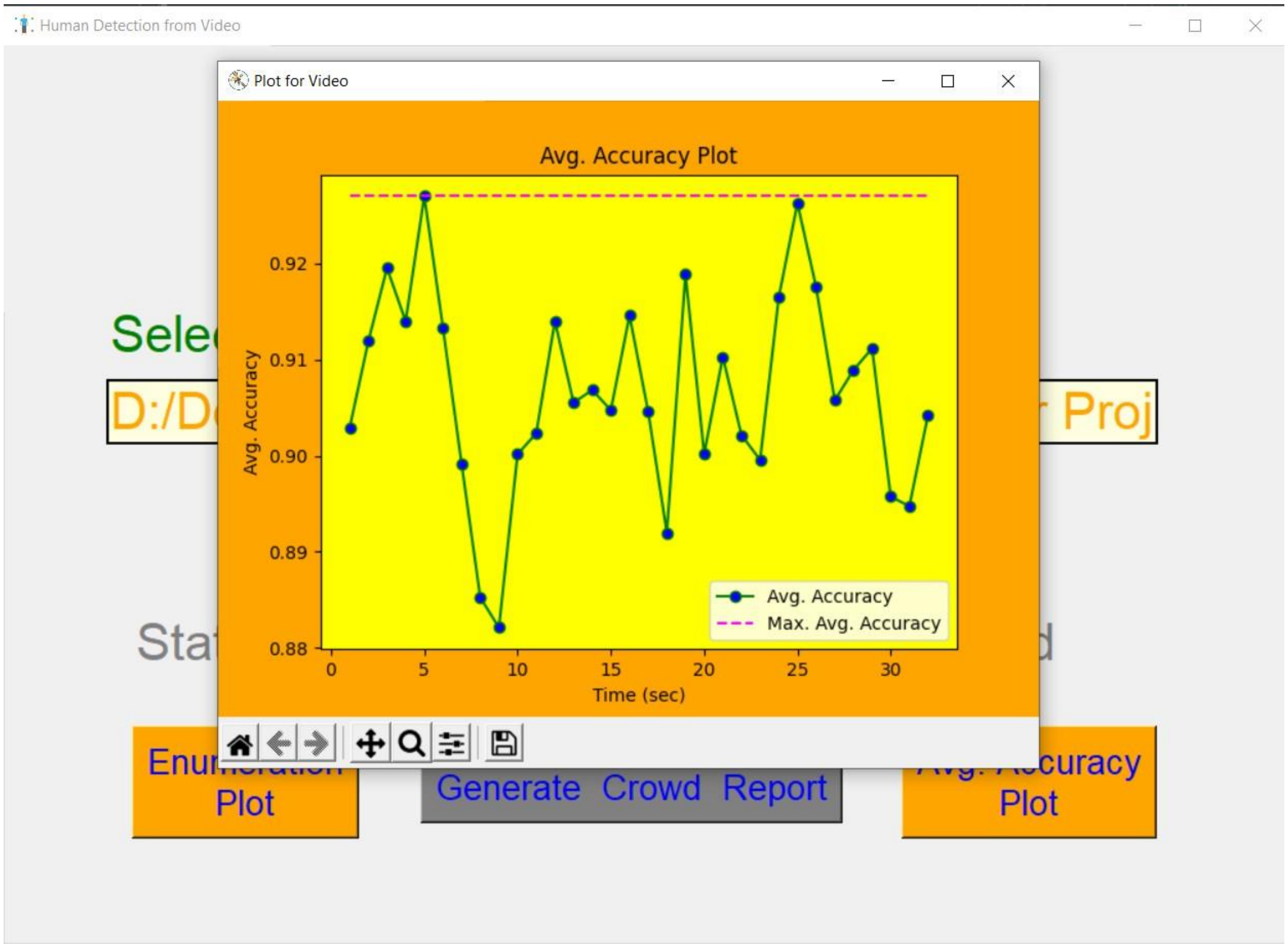
<u>**Detecting Humans from Video**</u>

Window 15



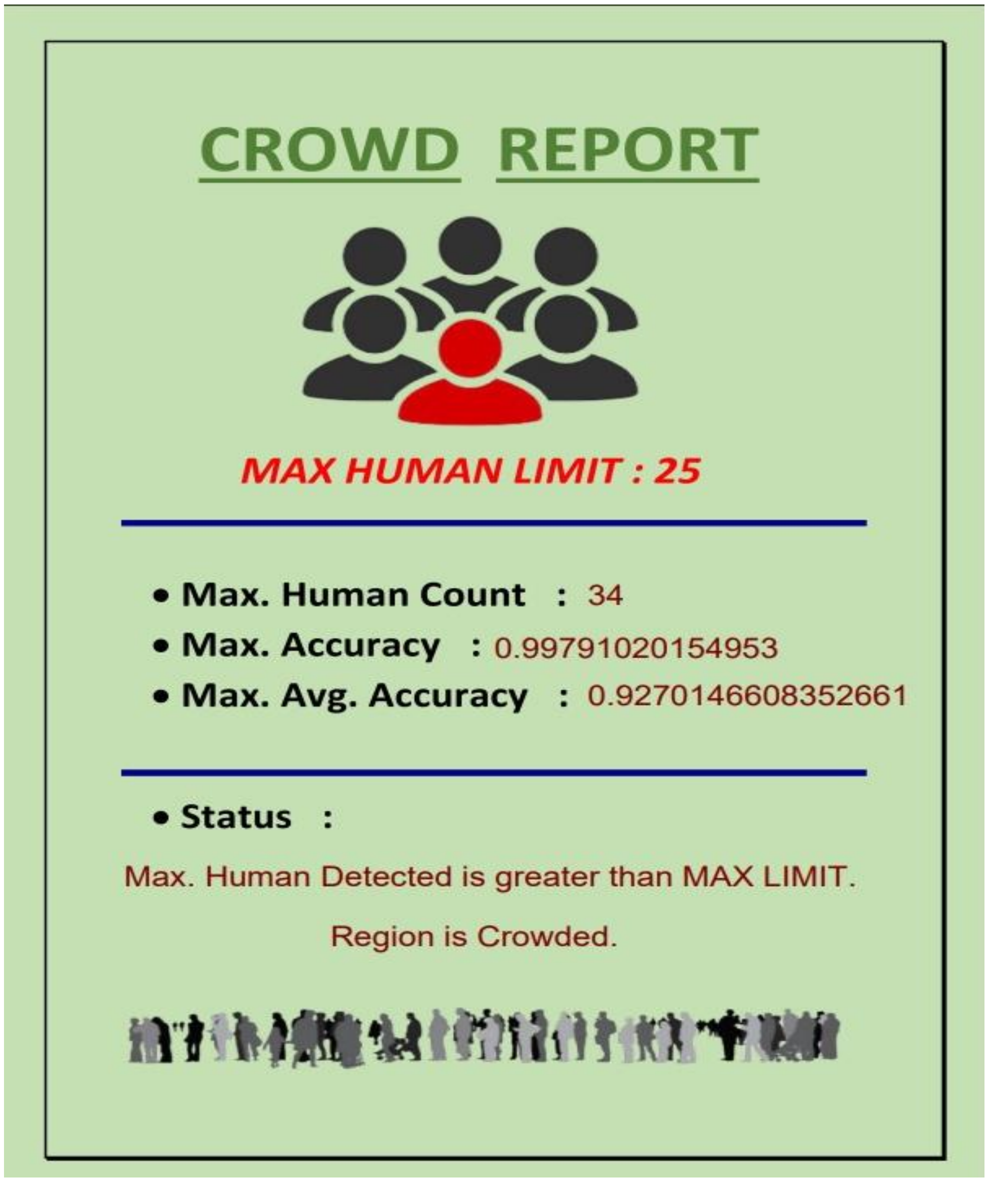
<u>Enumration Graph Plot</u>

Window 16



<u>**Accuracy Graph Plot**</u>

Window 17



CROWD REPORT

MAX HUMAN LIMIT : 25

- Max. Human Count : 34
- Max. Accuracy : 0.99791020154953
- Max. Avg. Accuracy : 0.9270146608352661

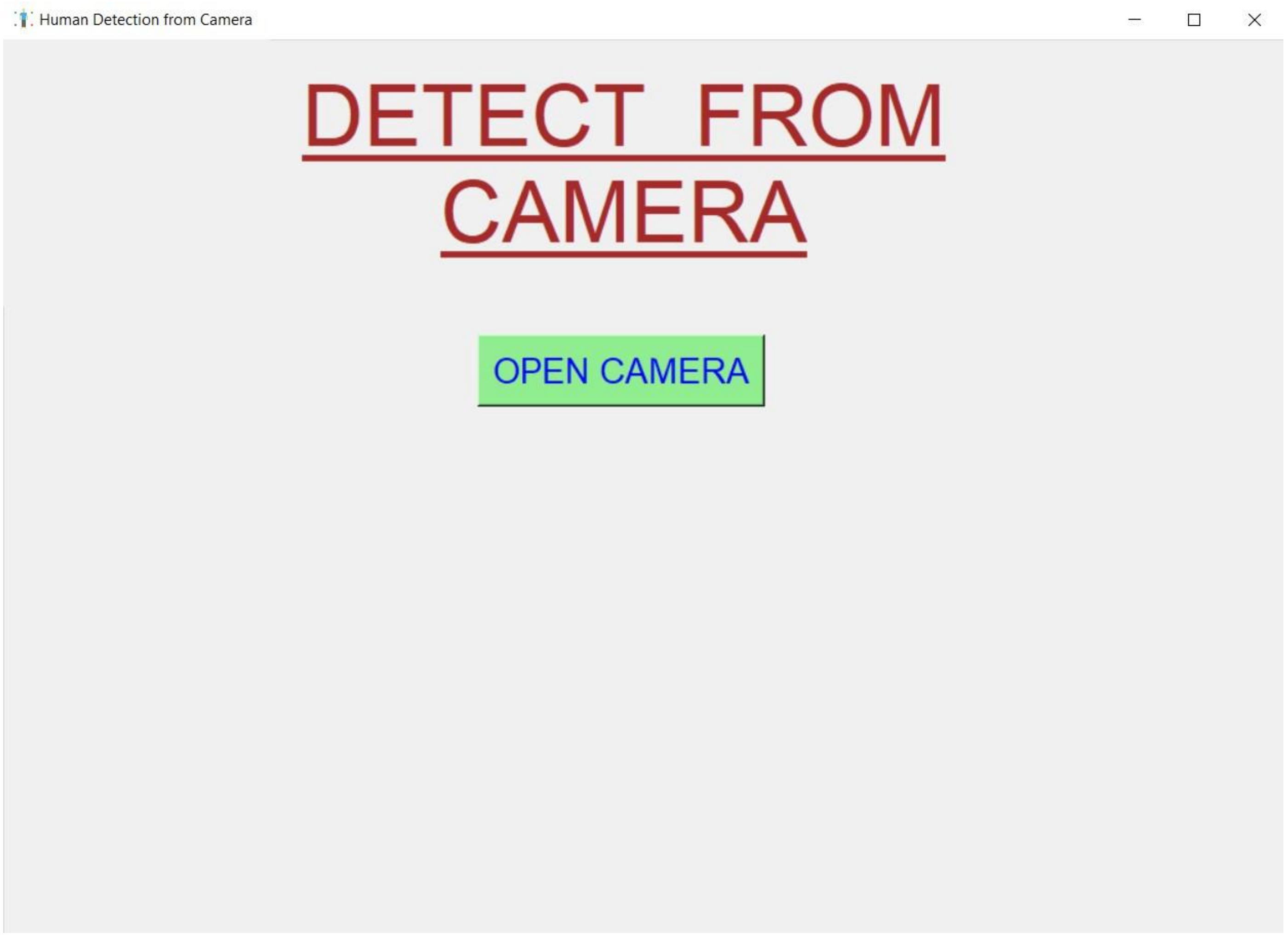
- Status :

Max. Human Detected is greater than MAX LIMIT.

Region is Crowded.

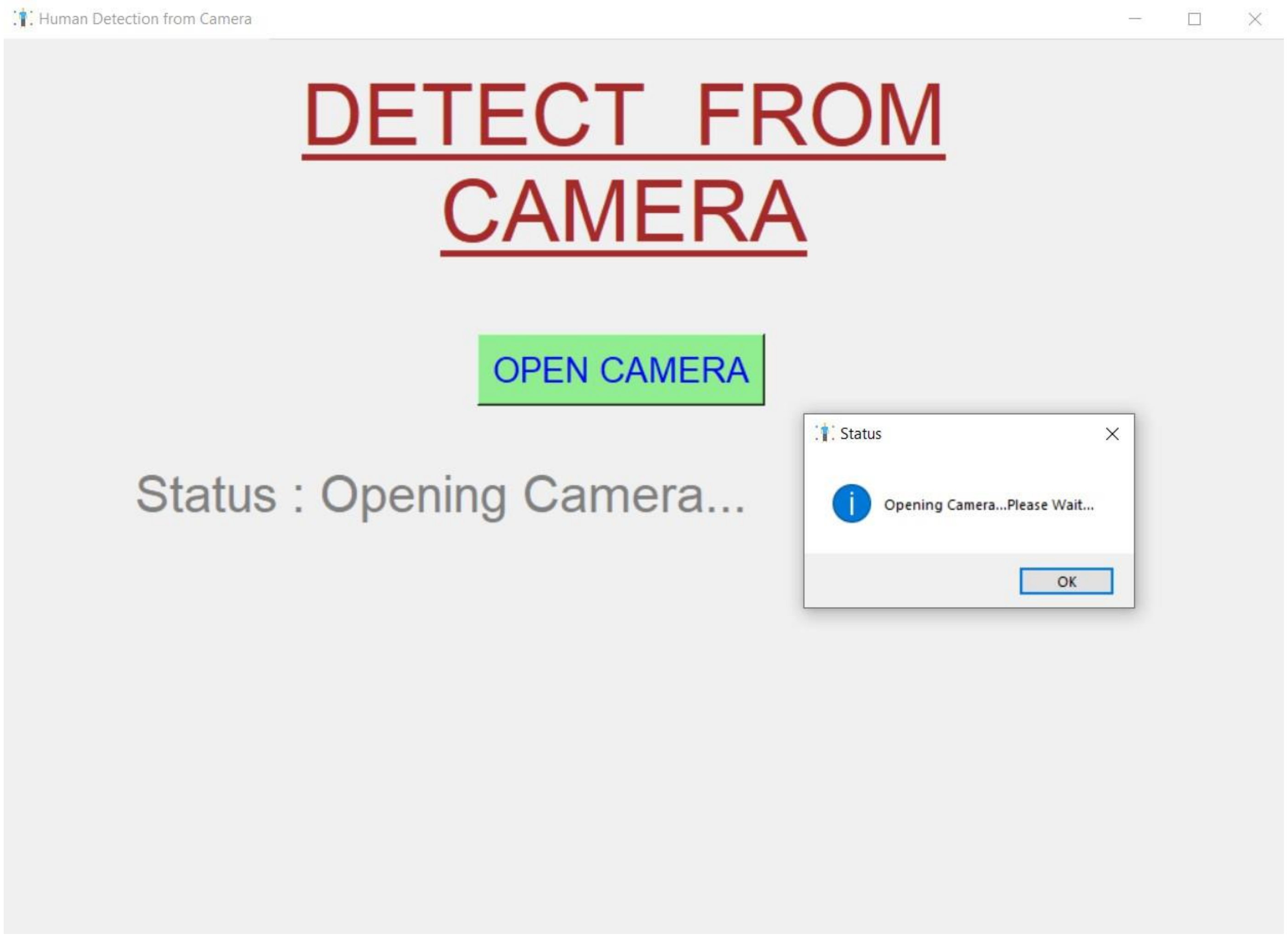
<u>Crowd Report of Video</u>

Window 18



<u>Detect From Camera Window</u>

Window 19

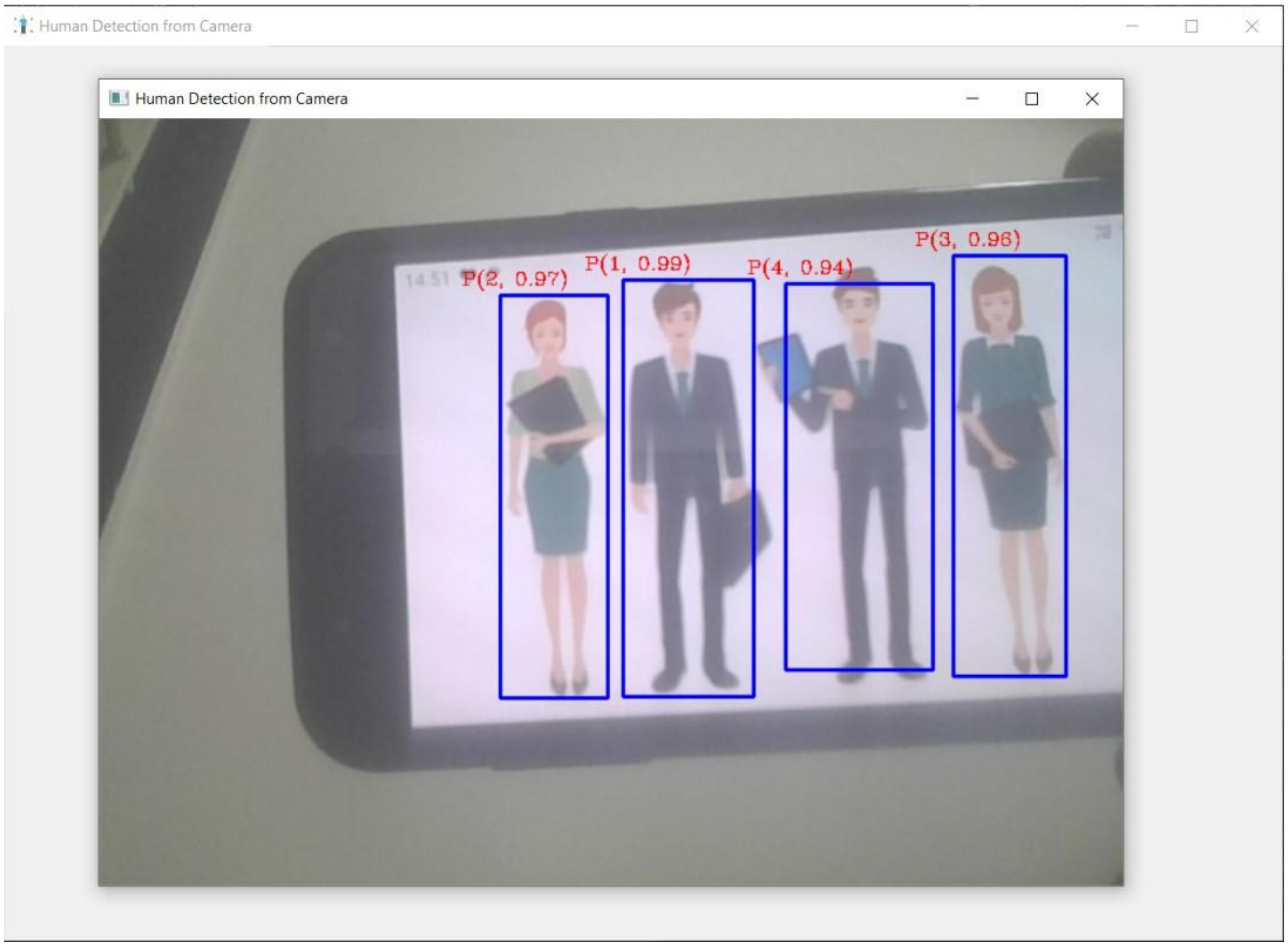


Status : Opening Camera...

Opening Camera...Please Wait...

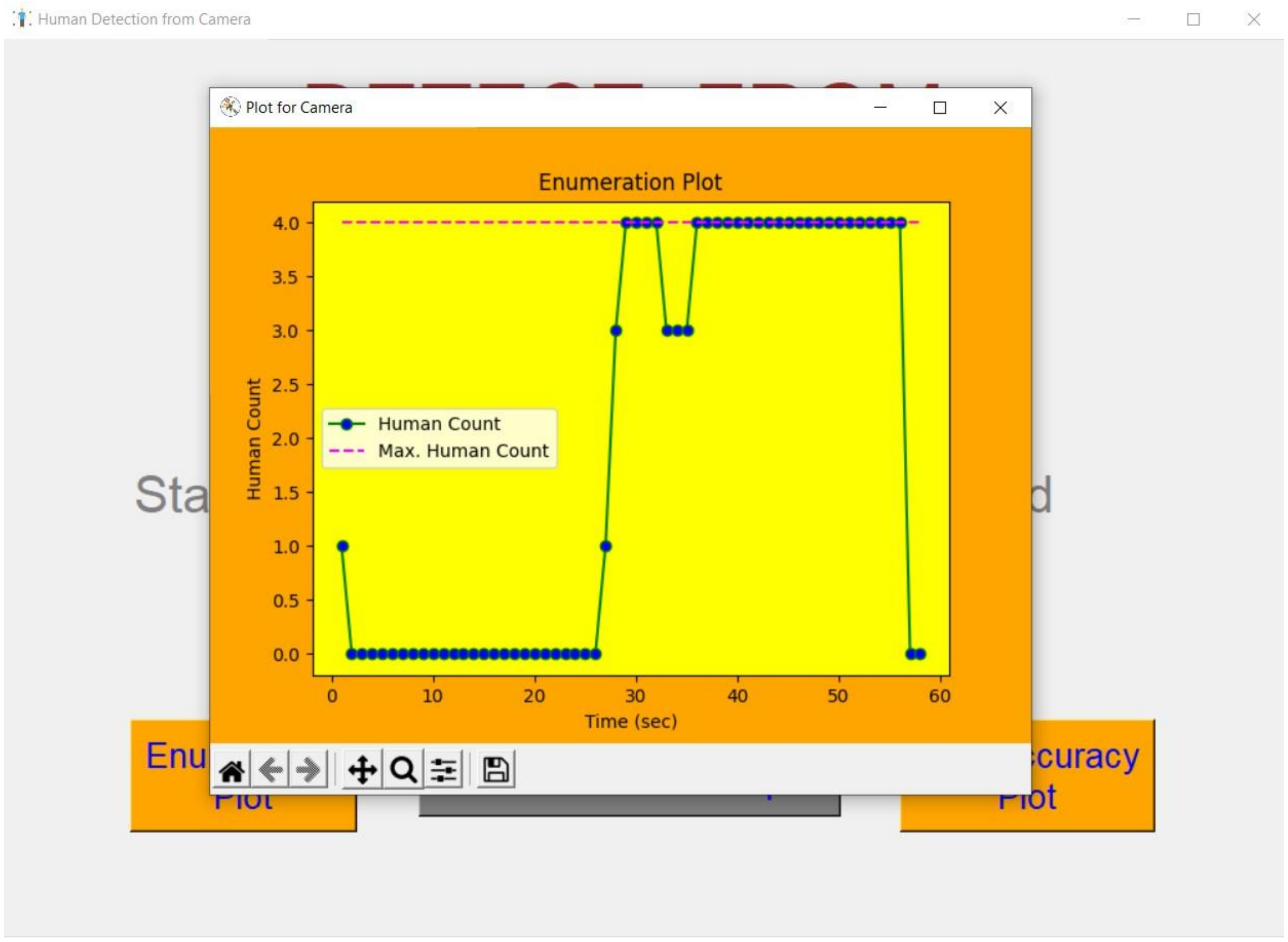
Camera Opening image

Window 20



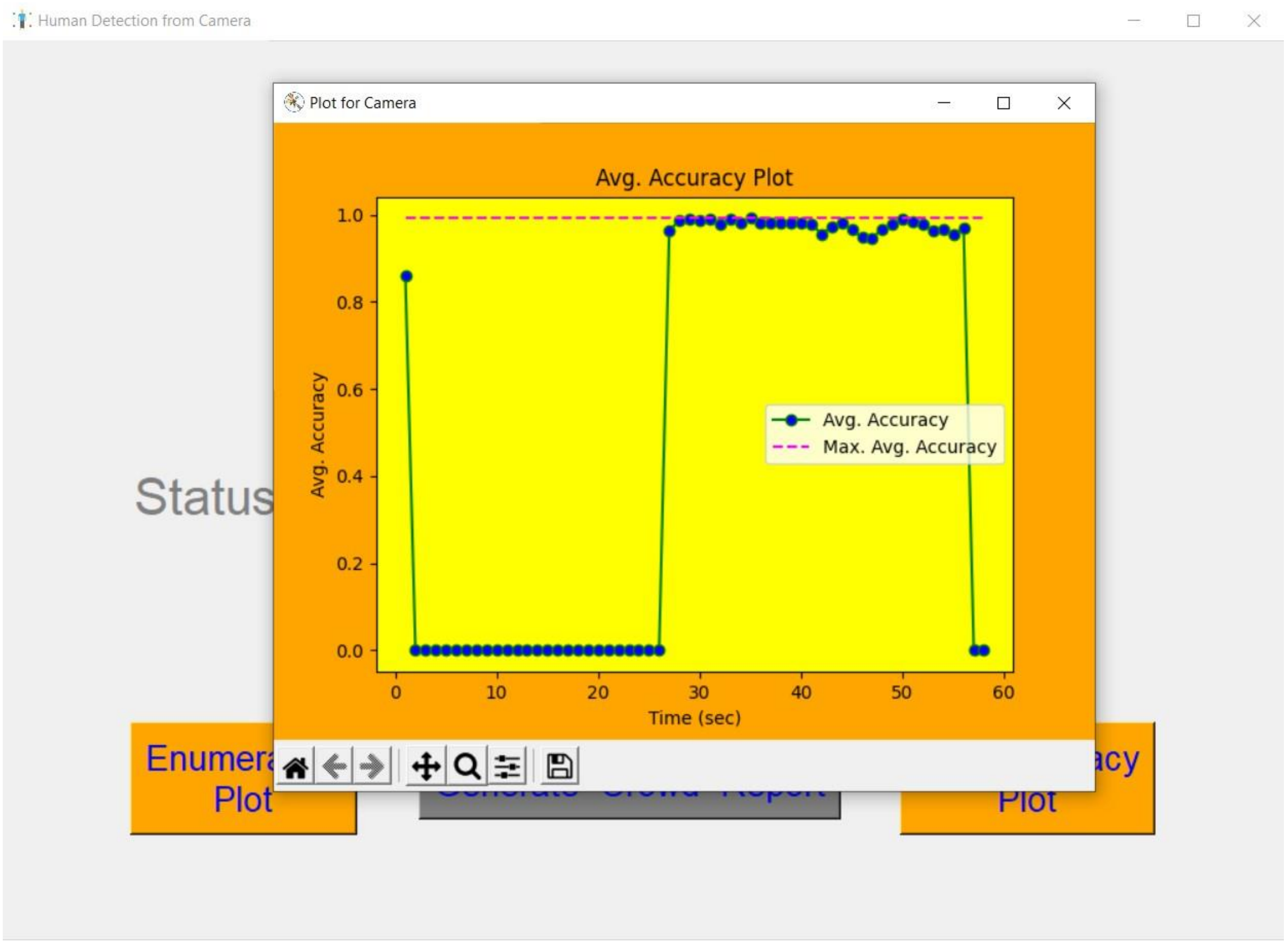
<u>Detecting from Live camera Feed</u>

Window 21



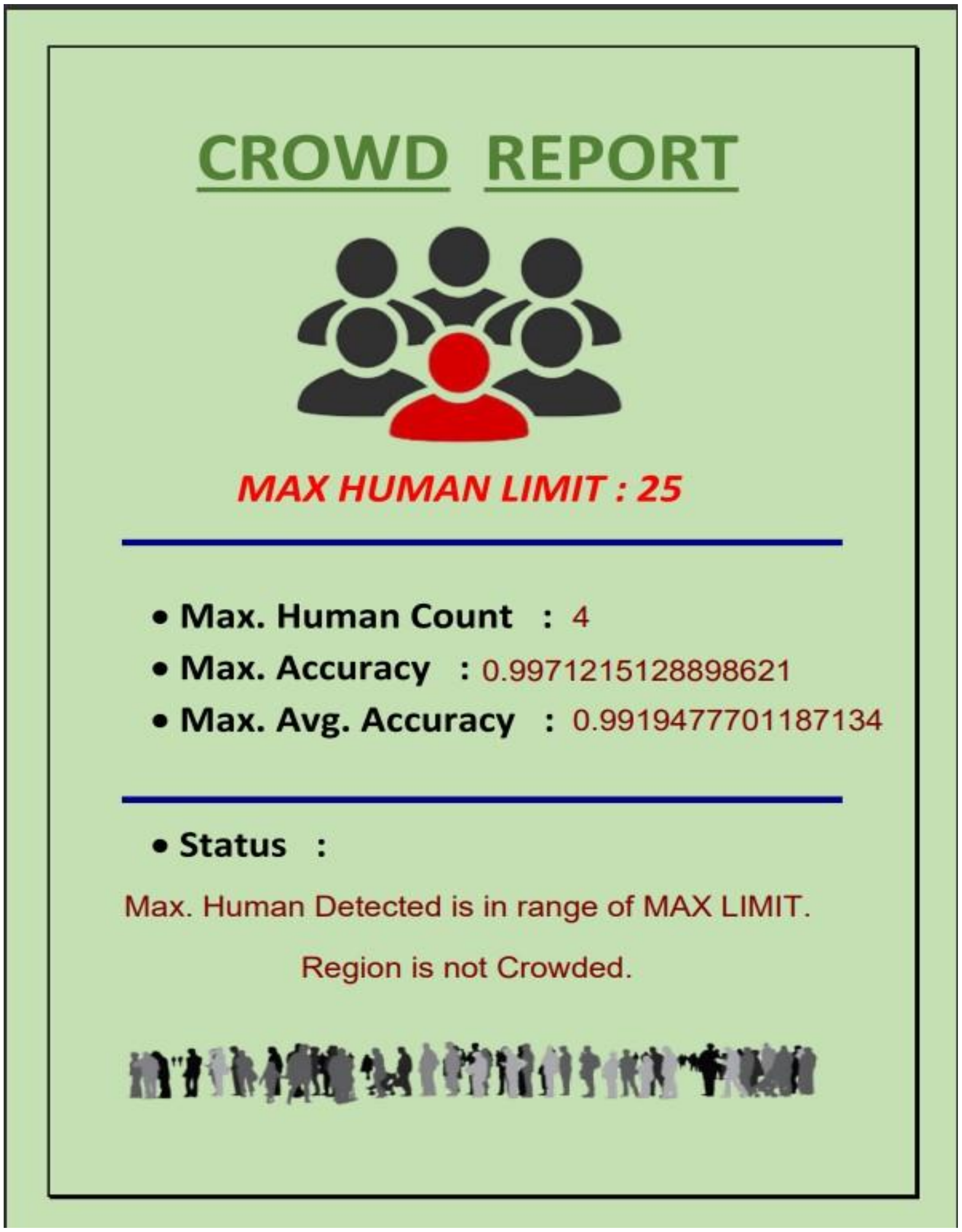
<u>Enumration Graph Plot</u>

Window 22



Accuracy Graph Plot

Window 23



CROWD REPORT

MAX HUMAN LIMIT : 25

- Max. Human Count : 4
- Max. Accuracy : 0.9971215128898621
- Max. Avg. Accuracy : 0.9919477701187134

- Status :

Max. Human Detected is in range of MAX LIMIT.

Region is not Crowded.

<u>Crowd Report of live Camera Detection</u>

CHAPTER-VIII

CONCLUSION

CONCLUSION

As we mentioned before in abstract, Detecting human beings accurately is one of the major topics of vision research due to its wide range of applications. A challenging aspect is to process the image obtained from a surveillance video as it has low resolution. A review of the available detection techniques is presented.

The detection process occurs in two steps: object detection and object classification. In this project, we choose tensorflow approach which is a very efficient algorithm for human detection and give accurate results.

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. At the end of this project, a discussion is made to point the future work needed to improve the human detection process in many areas especially surveillance videos. These include exploiting a multi-view approach and adopting an improved model based on localized parts of the image.

FUTURE SCOPE

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.

The outcomes in this research are based on results that involve only sample datasets. It is necessary that additional datasets should be considered for the evaluation of different classification problems as the information growth in the recent technology is extending to heights beyond assumptions. Recent field of technology is growing and data are by nature dynamic. Hence, further classification of the entire system needs to be implemented right from the scratch since the results from the old process have become obsolete. The scope of future work can deal with using CNN, which can increase the efficiency of the system, the existing system and processes the new incoming data more efficiently. More specifically, the models with CNN can be used in categorization process to improve the following aspects in each type of problems.

The real time human detection system can be enhanced in future, by including more low-level features such as shape and spatial location features apart from optimizing the weights and learning rate of the neural network. In this research work, while evaluating the fitness of an individual, the consideration is given only to the occurrence frequencies of an individual in the image, video and from web camera and not the date and time at which the person was detected. So, the positioning (date and time) of the person in the image, video and in web camera should be taken into account when evaluating the count of an individual in future works. When these enhancements are incorporated in the detection system, it would help further improve the performance and be useful for applications meant for the explicit detection system.

Reference

1. Stauffer C, Grimson W: Adaptive background mixture models for real-time tracking. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1999). Piscataway: IEEE; 1999:246-252.
2. Xiaofei J, Honghai L: Advances in view-invariant human motion analysis: a review. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 2010, 40(1):13-24.
3. T.Zhao and R.Nevatia,Tracking multiple humans in complex situations,IEEE Trans, Pattern Anal.Mach.Intell., vol.26, no.9, pp.1208-1221, sep.2004.
4. T.Zhao,R.Nevatia,and B.Wu, Segmentation and tracking of multiple humans in crowded environments, IEEE Trans.Pattern Anal.Mach.Intell.,vol30,no.7,pp.1198-1211,Jul.2008.
5. J.Rittscher,P.H.Tu, and N.Krahnstoever, Simultaneous estimation of segmentation and shape, in Proc,IEEE Cnf.Comput.vis .Pattern Recog.,2005,pp.486-493.
6. A.Mohan,C.Papageorgiou , and T.Poggio, Example-based object detection in images by components, IEEE Trans.Pattern Anal.Mach.Intell.,vol 23.no.4, pp.349-361,Apr.2001.
7. Ya-Li Hou, and Grantham K.H.Pang, People counting and human detection in a challenging situation, IEEE Trans.sys.man and cybernetics.,vol.41,No.1, pp.24-33, Jan.2011.
8. N.Dalal and B.Triggs, Histograms of oriented gradients for human detection, in Proc.IEEE Conf.Comp.Vis.Pattern Recog, pp.886-893,2005.
9. B.Wu and R.Nevatia, Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors, Int.J.Comput.Vis., vol.75,no.2, pp.247-266, Nov.2007.
10. G.J.Brostow and R.Cipolla, Unsupervised Bayesian detection of independent motion in crowds, in Proc.IEEE Conf.Comp.Vis.Pattern Recog, pp.594-601,2006.
11. V.Rabaud and S.Belongie, Counting crowded moving objects, in Proc.IEEE Conf.Comput Vis.Pattern Recog.,pp.705-711,2006.
12. A.C.Davies,J.H.Yin, and S.A.Velastin, Crowd monitoring using image processing, Electron.Commun,Eng.J.,vol.7,no.1, pp. 37-47, Feb1995.
13. H.celik, A.Hanjalic, and E.A.Hendriks, Towards a robust solution to people counting, in Proc IEEE Int. Conf.Image Process.,pp.2401-2404,2006.