MONITORING SOCIAL DISTANCING USING IMAGE PROCESSING AND DETERMINING THE EFFICIENCY

Submitted In Partial Fulfillment for The Degree of Mechanical and Automation Engineering

SOMESH RAICHANDANI	02296403617
DHEERAJ MANN	00796403617
AADHAR DOGRA	35196403617
NILESH CHAND RAJWAR	35496403617

UNDER THE GUIDANCE OF

PROF. RAKESH CHANDER SAINI

(Asst Professor, Dept of Mechanical & Automation Engineering)

Maharaja Agrasen Institute of Technology, Delhi

CONTENTS

Abstract

In the time of the pandemic COVID-19, one of the most important parameters for the safety of people is social distancing. To incorporate this habit in people, they need to be monitored by authorities for prevention of the spread of the virus. A computer application is developed just to detect the violations in social distancing in a busy place, so people can be monitored.

The application is programmed in python programming language, incorporated with deep learning and shows social distancing violations in real time, using object detection and image processing.

The efficiency is calculated in terms of the accuracy and a cutoff angle after which the application will show deviation. The tools used to calculate accuracy and the cut off angle are laser distance meter and theodolite respectively. The application is then integrated with deep learning to train hundreds of models to increase accuracy in object detection.

LIST OF FIGURES

LIST OF TABLE

NOMENCLATURE

Open CV	Open Computer Vision
Tf	Tensorflow
Py	Python programming/extension
C	speed of light
T	time taken

CHAPTER 1 INTRODUCTION

This chapter delivers the insights upon the terms, tools and technologies used in the project. Each of the sections are categorized according to the chronology of using them and their meanings are defined in each section.

1.1 Social Distancing

Social distancing refers to the physical distancing between people, usually mandatory at a place of high physical risk, or in situations as critical as this pandemic, COVID-19. Social distancing is critical to avoid contagious diseases and it requires a distance of typically at least 6ft in length between two people. However it is often seen that lack of awareness leads to not following the rules and regulations and people end up with more sufferings in times of a disease.

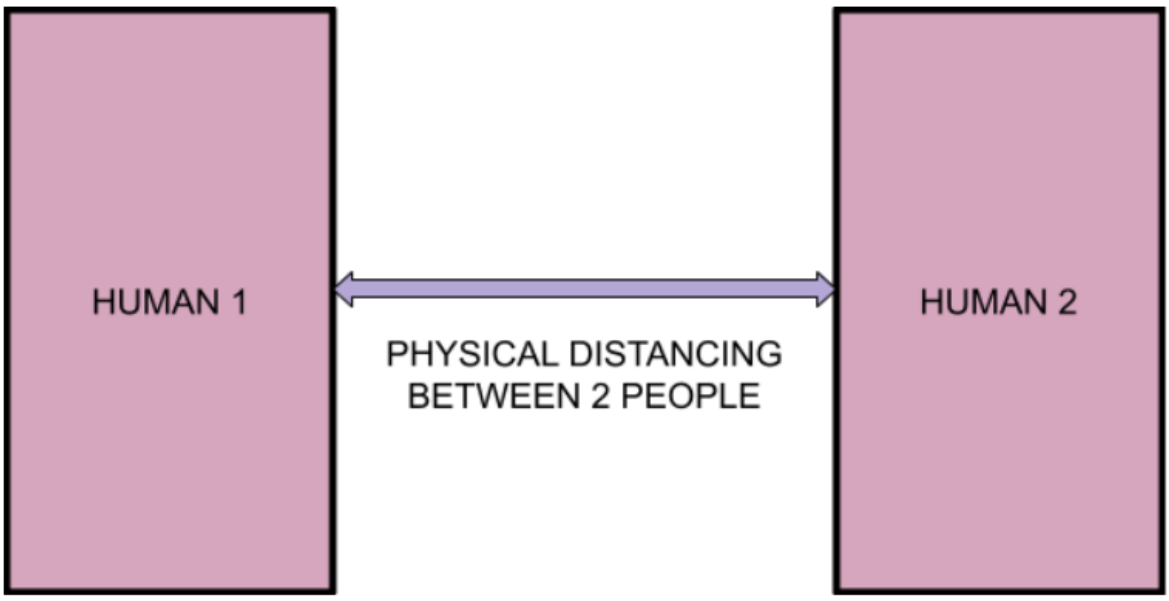

Fig 1.1 For Social distancing, the physical distance >= 6ft

Hence a computer application is a good solution to detect violations between people in a busy street so that they can be monitored much easily than if they were monitored manually. This will lead to the much-needed awareness between people if they know they are being monitored by technology rather than a person.

Social distancing is not only a solution in the times of a pandemic, but is also really helpful in places like hospitals where infectious and contagious diseases can spread very quickly. So, the application will prove to be helpful not only in this pandemic, but also any time and any place where social distancing will be required by people.

1.2 Object Detection

When a machine is able to detect the objects visible in its frame, the process is known as object detection. Object detection can also be defined as a computer technology related to computer vision and image processing that usually deals with detecting instances of semantic objects of a certain class in digital images and videos. Very well-researched domains of object detection include both face detection (facial recognition) and pedestrian detection.

It allows us to identify, detect and locate objects in an image or video. With this kind of identification, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labeling them. Then it can be processed further for various applications.

We can use a variety of tools and techniques to perform object detection. The Popular deep learning–based approaches are using convolutional neural networks (CNNs), such as R-CNN and YOLO, which automatically learn to detect objects within images.

We can choose from two key approaches to get started with object detection using deep learning:

1. Create and train a custom object detector.

To train a custom object detector from the very beginning, we need to design a network architecture to learn the features for the objects of interest. We need to compile a very large set of labeled data to train the CNN. The results of a custom object detector can be remarkable. That said, we need to manually set up the layers and weights in the CNN, which requires a lot of time and training data.

2. Use a pretrained object detector.

Many object detection workflows which use deep learning also apply transfer learning, an approach that enables we to start with a pretrained network and then fine-tune it for your application. This method always provides faster results because the object detectors have already been trained on thousands, or even millions, of images.

1.3 Image Processing

Image processing is defined as a method to perform various operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of pictorial processing in which input is an image and output may be image or characteristics or features associated with that image, depending on the application. Nowadays, image processing is among rapidly and fastest growing technologies. It forms a core research area within engineering and automation and robotics disciplines too.

Image processing typically includes these following three steps:

1. Importing the image via image acquisition tools.
2. Analyzing and manipulating the image.
3. Output in which result can be a changed image or report that is based on image analysis.

Image processing techniques help in manipulation of the digital images by using machines. The three general phases that all types of data have to undergo while using digital technique are pre-processing, enhancement, and display, information extraction, all of which can be performed with computer algorithms.

1.4 Video Processing

One field in particular is starting to gain more attention: Video. Most of the applications of computer vision today are based on images, with less focused on sequences of images, i.e. video frames. Essentially, a video is nothing but a collection of multiple images. Our brain perceives it as something moving because of our retinal qualities.

Video allows for much deeper situational understanding than images, because sequences of images provide new information about action. For an example, we can track an object through a sequence of images and understand its behavior to predict the next moves. We can track a human pose, and understand and predict the action taken with its action classification.

Hence, working on image frames will ultimately lead to our working on a video. We can use image processing to process videos by breaking down the video into smaller frames, just like still images.

1.5 Python Programming

Python (programming language) is a high-level interpreted general-purpose language. Python's philosophy of design accomplishes emphasizing code readability with its notable use of significant indentation. Despite starting out as a hobby project, Python is now one of the most popular and widely used programming languages in the whole world. Besides web and software development, Python has various other uses, it is used for data analytics, machine learning, and even design. It's often called as a “scripting language” for applications. This means that it can automate specific series of tasks, making it more powerful and efficient. Consequently, Python is often used in software

applications, pages inside a web browser, the shells of operating systems, games and other types of developments.

We have used python as our programming languages to write the algorithms. The algorithms are used to detect objects in a frame and then the distance between the objects is calculated. The algorithm then determines the violation of social distancing according to the distance between the objects and counts the number of violations, and calculates some parameters too.

1.6 Computer Vision

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

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

Computer vision makes machines learn and trains them 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. Since 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 and easily surpass human capabilities. Computer vision is used in industries ranging from energy and utilities to manufacturing and automotive – and the market is continuing to grow.

We have used an open-source library to perform computer vision, known as Openo_cv, o_cv standing for Computer Vision, and included as a python library before working on the algorithm. Openo_cv uses its internal functions to perform Computer Vision and image processing, making the task of object detection much easier, faster and accurate.

1.7 **Theodolite**

Theodolite, a basic surveying instrument is basically used to measure horizontal and vertical angles. In its modern form it consists of a telescope which is mounted to swivel both horizontally and vertically. Leveling is accomplished with the aid of a spirit level and crosshairs in the telescope permit the accuracy in alignment with the object sighted. After the telescope is adjusted precisely, the two accompanying scales, vertical and horizontal, are observed and read.

Mounted on a tripod with adjustable legs, the theodolite is used in the field to obtain very precise angular measurements for triangulation in road building, tunnel alignment, civil-engineering, and other works. The transit is a variety of theodolite that has the telescope mounted so that it can be fully reversed, or transited. The phototheodolite, a combination camera and theodolite mounted on the same tripod, is used in terrestrial photogrammetry for map-making, measuring angles and other purposes.

1.8 Laser Distance Meter

A laser distance meter, also known as a laser tele-meter, is a range finder that uses a laser beam to determine the distance to an object, by shooting up a laser beam till the object. A laser distance meter emits a pulse of laser beam at a target object. The pulse then reflects off the target and bounces back to the sending device (in this case, a laser distance meter). This "time of flight" principle is based on the principle that the laser light travels at a fairly constant speed through the air

in Earth's atmosphere. At the inside of the laser meter, generally a simple computer quickly calculates the distance to target object. This method of distance calculation is capable of measuring the distance from the Earth to the moon within a distance of a few centimeters. Laser distance meters are usually also referred to as "range finders" or "laser range finders." Given the higher speeds at which the pulse travels and its focusing, this calculation is very accurate and precise over distances of feet or miles but loses accuracy over much closer or farther distances.

The distance between the meter and target is given by

$$D = c.t / 2$$

where c equals the speed of light and t equals the amount of time for the round trip between meter and target.

CHAPTER 2 LITERATURE REVIEW

This chapter delivers the insight of the rigorous literature survey on the research and development of image processing, object detection systems. Research works performed by various algorithms and different approaches have been reviewed and grouped accordingly. Research gaps and objectives have been identified and listed at the end of the chapter.

2.1 Literature review

Image modification produces a "new" image from two or more sources that highlight specific features or structures of interest, better than the original image input. Basic basic modification works for simple mathematical functionality in image data. Image deletion is often used to identify changes that have taken place between images collected on various dates. The main methods of image conversion are

• Hough Transform: used to find lines in a picture

• Radon Transform: used to recreate images from fan-beam fan and parallel-beam prediction data

• Discrete Cosine Transform: used for image and video compression to filter and analyze frequency

• Wavelet Transform: used to perform wavelet discrete analysis, denoise, and fuse images

Object Tracking: Object tracking is the process of finding an object (or multiple objects) in a sequence of images. It is one of the most important applications for computer vision, such as surveillance, computer communications, and medical imaging. Feature detection: A feature is defined as an “interesting” part of an image and features are used as a starting point for many computer viewing techniques. Since features are used as the starting point and first symbols for subsequent algorithms, the whole algorithm will generally be equal to its feature detector. Feature detection is the process of detecting

certain features of a visual stimulus, such as lines, edges or angles. It will be useful to make local decisions about the content of the location details (image structure) in the image.

With mobile devices such as smart phones, iPads and tablet pcs equipped with cameras, the need for photo editing apps is increasing. These applications require faster and low power consumption because the portable device is powered by the battery only. An easy way to increase device performance is to install old hardware with new hardware. Hardware technology depends on semiconductor technology. If a semiconductor chip contains a large number of transistors, the number of transistors increases. As overcrowding increases, current leaks also increase. We must therefore select a functional programming language to write an image processing application for mobile devices. These days, android became the most popular mobile device app. Android developers are introducing a new program to meet the needs of Smartphone users. Libraries such as OpenGL (Open Graphics Library) and Openo_cv (Open Computer Vision) are used for program development. An app that uses a camera usually includes a way to process images such as Gaussian, Median, Mean Laplacian, Sobel filter and others. In 2010, a new module was introduced in openo_cv to address GPU acceleration. Openo_cv uses a photo container called a o_cv :: Mat that exposes access to raw image information. In GPU module the container o_cv :: gpu :: GpuMat stores image data in GPU memory.

We may use this type of alarm to detect or prevent a criminal act. It uses easy motion detection to find the intruder. Motion detection is generally a software-reliant monitoring algorithm. Indicates the camera that it starts capturing the event when it detects movement. In image processing, the image is considered a two-dimensional signal. Video captured by the camera is analyzed via the OPENo_cv system to detect movement.

Steps to get motion detection using Openo_cv include:

1. Download video from camera o_cvCapture * capture = o_cvCaptureFromCAM (o_cv_CAP_ANY);

2. Download frame1 IplImage * frame1 = o_cvQueryFrame (hold);

3. Download frame2 IplImage * frame2 = o_cvQueryFrame (hold);

4. Blur the images slightly to reduce the blurring of sound (frame1, frame1_blur, Size (4, 4)); blurring (frame2, frame2_blur, Size (4, 4));

5. Find the absolute difference (frame 2_blur, frame1_blur, effect_frame); Flow chart for motion detection.

A study of how to make signi fi cance image processing and its application in the field of computer vision is conducted here. During image processing the input provided is an image and the output is high quality enhanced according to the techniques used. Image processing is often referred to as digitalimage processing. Our study provides a solid introduction to the ingalong process with classification techniques, the basic principles of computer vision and its ap-based applications that will be relevant to the use of images and computer research communities.

A digital image represents a real image as a set of numbers that can be stored and managed by a digital computer. To translate image numbers, we are divided into smaller areas called pixels (image elements). Foreach pixel, an imaging device that records a number, or small number of numbers, that define a particular pixel's property, such as its brightness (intensity) The numbers are arranged in rows and columns corresponding to the correct positions and pixels of the image. Digital photography has a few basic features. Another type of image. For example, a black-and-white image only records the

magnitude of light falling on pixels. The color image can be three colors, usually RGB (Red, Green, Blue) or four colors, CMYK (Cyan, Magenta, Yellow, black) .RGB images are commonly used on computer monitors and scanners, while CMYKimages is used in color printers. There are also non-visual images such as asultrasound or X-ray where audio or X-ray recording is recorded. Illegal images, pixel distance from the monitor is recorded. Resolutionis is expressed in the number of pixels per inch (ppi). High resolution provides a more detailed image. The computer monitor usually has a resolution of 100ppi, while the printer has a resolution from 300 ppi to more than 1440ppi. That's why the image looks so much better in print than in the monitor.

Image processing is a way to convert an image to digital and perform certain functions on it, in order to get an enhanced image or extract useful information from it. The type of signal time in which an image is inserted, such as a video frame or a picture and output can be the icon symbols associated with that image. Usually the Image Processing process involves treating images as two equal symbols while using existing editing methods. It is one of the fastest growing technologies today, with its applications in various aspects of business. Image Processingforms core core area within engineering and computer science disciplinestoo.Processing images basically involves the following three steps:

- Image import with image capture tools.
- Image analysis and management.
- Effect on which an image or report based on image analysis can be altered. Low image processing systems include:
- Edge detection
- Separation.
- Property classification.
- Identification and matching.

Show image: Use the function o_cv2.imshow () to display a picture in a window. Windows automatically include image size. The first issue is the name of the string window. the second issue is our image. We can build as many windows as we like, but with different windows names. Next, the function o_cv2.waitKey (), used in Openo_cv keyboard binding, performs the rendering of an image uploaded in a previous step. It takes a number indicating the time in the second-second of the offer. Basically, we use this function to wait for specific specifications until we encounter a keyboard event. The program stops at this point, then waits for we to press any key to continue. If we do not pass any argument, or if we pass 0 as an argument, this function waits for an inde ely nitely keyboard event.Finally, the function o_cv2.destroyAllWindows () destroys all the windows we have built.

2.2 Literature gap

A thorough investigation and literature survey on the image processing and object detection has been done successfully. A significant amount of work computer vision has been done in the survey. Openo_cv library from python has been used to incorporate image processing functions for the algorithms to work. Work has been done to apply the object detection using these algorithms to detect distance between people in real-time in a video for social distance monitoring purposes. Few researchers conducted the experimental work on humans considered as objects for image processing and object detection.

2.3 Objectives of the dissertation work

• Detection of people in a given video frame and detect distance between them.

• Differentiating each individual and mark the number of social distancing violations in the frame in real-time.

• Finding the accuracy of the tracker system and also a cut-off angle after which it would show aberrations and hence determine the efficiency in the terms of accuracy and cut-off angle found.

CHAPTER 3 METHODOLOGY

This chapter focuses on the fabrication work of the tracking system and the main application. The fabrication work, components of each unit and main unit is discussed in detail. The testing of different configurations of the application with the same setup is also discussed in this chapter. The measurement of accuracy and cut-off angle according to specific distance ranges is also included in this chapter. This chapter also focuses on the trouble shooting of various aspects in order to attain maximum efficiency.

3.1 Object Detection

The first step towards detecting a human in a video would be to detect a human in an image. But before that, it is required to detect any object in a given frame.

For example, image classification is straightforward, but the difference between doing something and finding an object can be confusing, especially when all three of these functions can be called equally as object recognition.

Image classification involves assigning a category label to an image, and the process involves drawing a binding box around one or more objects in an image. Obtaining an item is a big challenge and combines these two functions and then draws a binding box around each item of interest in the image and assigns it to a class label. Collectively, all of these problems are called object recognition.

Object recognition is a common term to describe a collection of related computer viewing activities that include identifying objects in digital images.

Image classification involves predicting a category of one object in an image. Localization of an object refers to seeing the location of one or more objects in an image and drawing a box full of their width. Object acquisition combines these two functions and is localized and separates one or more objects in the image.

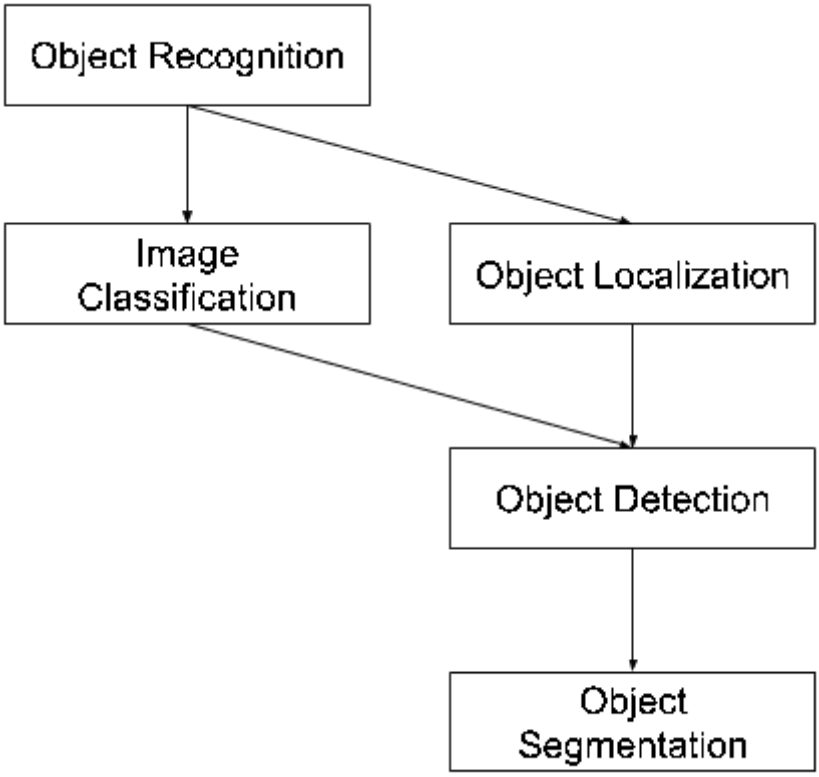

Fig 3.1 Object Recognition Process

Therefore, we can distinguish between these three computer viewing functions:

Graphics: Predict the type or category of an object in a picture.

Input: A picture with one object, such as a picture.

Output: Class label (e.g. one or more numbers labeled on class labels).

Geographic Object: Detect the presence of objects in an image and indicate their location with a mandatory box.

Input: A picture with one or more objects, such as a picture.

Output: One or more binding boxes (e.g. defined by point, width and height).

Object discovery: Find the presence of objects and a box that meets the types or classes of objects found in the image.

Input: A picture with one or more objects, such as a picture.

Output: One or more binding boxes (e.g. defined by point, width and height), and the label label of each binding box.

Another extension to this breakdown of computer-view functions is the separation of an object, also called “object modeling” or “semantic separation,” in which the objects of a known object are indicated by highlighting the pixels of an object instead of a compound connecting box.

In this breakdown, we see that object recognition refers to a range of challenging computer recognition tasks.

The source code written for object dxetection is as follows for the following output generated:

```
from imageai.DetectionModule import ObjectDetection

import os


execution_path = os.getcwd()


detector = ObjectDetection()
```

```
detector.setModelTypeAsRetinaNet()
detector.setModelPath( os.path.join(execution_path , "resnet50_coco_best_v2.1.0.h5"))
detector.loadModel()
detectionsNumber = detector.detectObjectsFromImage(input_image=os.path.join(execution_path , "image.jpg"), output_image_path=os.path.join(execution_path , "imagenew.jpg"))

for eachObject in detections:
    print(Object["name"] , " : " , Object["percentage_probability"] )
```

Fig 3.2 Output of the algorithm for detecting an object

3.2 Detecting People in Images

Finding a successful person in a photo or video means that we are building an app that will marry object discovery and image sharing. The technology that allows we to access objects in image data is slightly different from the visual separation tools currently used in many industries.

First of all, there is now a set framework for getting things done in video with varying degrees of accuracy. Pairing the target location of an object in an image with an understanding of the object category means that your application can distinguish between a person in a single image region compared to an object that can be disrupted as a person, such as a mannequin in a commercial area.

Computer vision acquisition performs three different functions:

1. We select items from the background images

2. We propose things as belonging to a particular category - people, in this case - using possible points

3. Defines the parameters of the proposed population by x-y origin and height and length of numbers

At the highest level, there are two things to consider when approaching a photo finder using computer vision apps. First, there is the technical side - how we can see people in a photo or video. The second part is what we can do with the results, and it has to do

with the quality of the profits we get from your app. to solve the problem of how to find objects in photos or video data first with a formal image classification. First, the tool would use input algorithms for the purpose of identifying areas of interest. The machine will come up with a variety of items based on the settings. The final steps of adoption are to differentiate objects based on models, to apply the boundaries of opportunities and to retrieve classrooms and areas within the framework of final approved proposals.

The category we would want, in this case, is the people. Applications get these human objects in the visual field with pre-trained processing blocks by inserting a large number of images through an in-depth study program. These repair elements are known as models, and can be trained to see almost anything people can see.

When we use human detection in your computer vision application, we can use a pre-trained model or provide the model ourselves. If we provide an additional model, the device will be better able to see the features we want and learn how to improve them in the future.

After uninstalling, it will be up to us how we can use it. Our use case will determine a variety of details, such as access limits. The one-stop-finding tool for one situation may not work for others.

The equipment has performed well over the past few years in performing these tasks due to advances in model training and in-depth multilayer processing. For example, Deepface and DeepID, the first two technologies of the download and comparison feature have worked very well as there are many examples facing consumers now: People now use technology to unlock their tablets and phones, for example.

Cloud processing represented an important development in the o_cv, which puts powerful resources in the hands of developers everywhere. However, the new o_cv platforms like Openo_cv provide easy access to open API platforms, giving developers more flexibility. Now, it is possible to build in-depth learning programs on edge devices. We do not need to be computer experts and we do not need to rely on cloud connectivity to use computer recognition services, such as object detection, processing and analysis of your images. Businesses of all sizes can now build and ship high-end computer vision applications to devices with limited resources, low power.

The following code detects a persons front and side view in a given frame in real time, provided with a person in a frame.

Code to detect a person in a frame:

```
def detect(frame):
    bounding_box_cordinates, weights =  openo_cv.detectMultiScale(frame, winStride = (4, 4), padding = (8, 8), scale = 1.03)

    person = 1
    for x,y,w,h in bounding_box_cordinates:
        o_cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 2)
        0 + _o_cv2.putText(frame, f'person {person}', (x,y), o_cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1)
        person += 1

    0 + _o_cv2.putText(frame, 'Status : Detecting ', (40,40), o_cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)
```

```
    1 + _o_cv2.putText(frame, f'Total Persons : {person-1}', (40,70),
o_cv2.FONT_HERSHEY_DUPLEX, 0.8, (255,0,0), 2)

    o_cv2.imshow('output', frame)

    return frame
```

The output of the code is as follows:

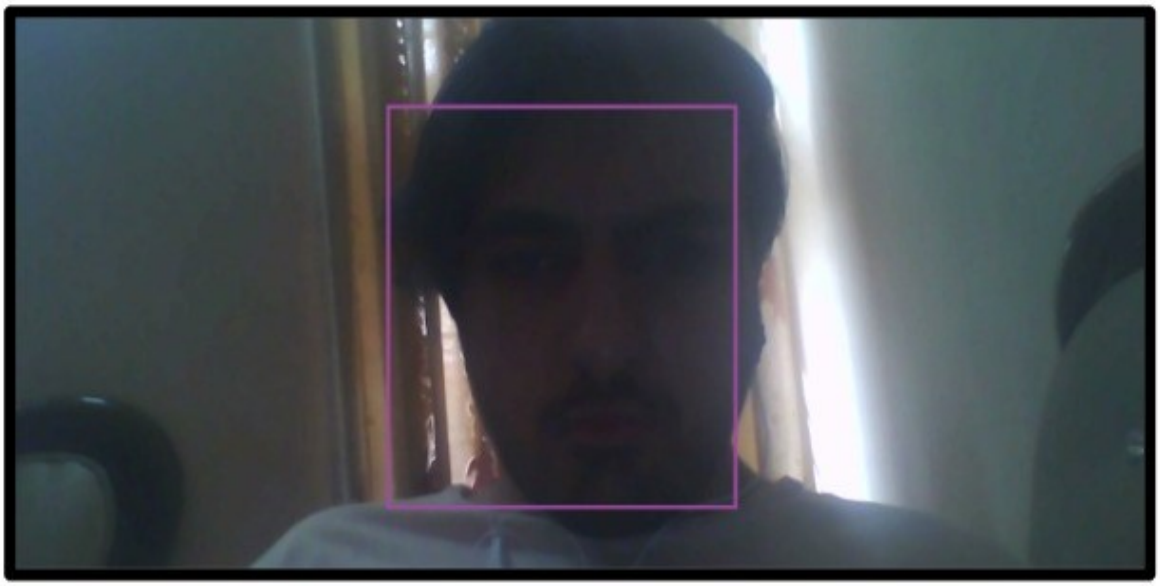

Fig 3.3 Front view of detection of a human

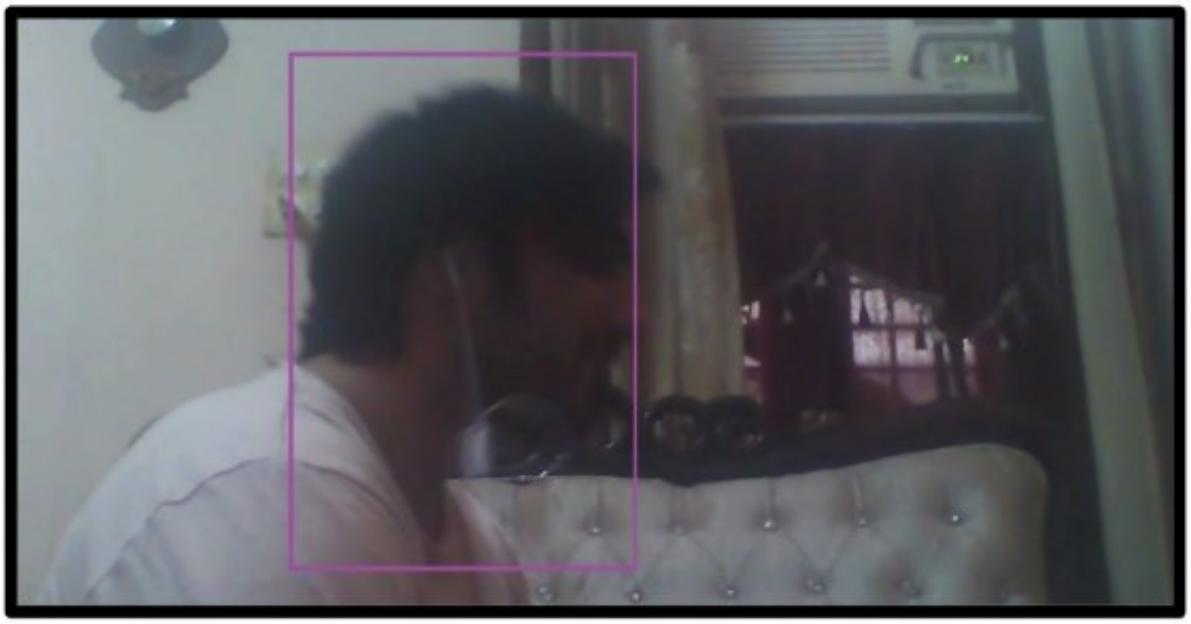

Fig 3.4 Side view of detection of a human

Code to detect a person from a given camera with output:

```
def detectByCamera(writer):
    video = o_cv2.VideoCapture(0)
    print('Detecting people…')

    while True:
        check, frame = video.read()

        frame = detect(frame)
        if writer is not None:
```

```
            writer.write(frame)


        key = o_cv2.waitKey(1)
        if key == ord('q'):
            break

video.release()
o_cv2.destroyAllWindows()
```

3.3 Detecting distance between objects

The detection of an object, the Zoom

We have to find a way to get the object that we are going to use it to zoom out(or in this case, of course, our “blue” or “yellow” sensor. Most likely, it will do this by using one of the methods described in my previous posts; as RGB and HSV detection.

Version

The implementation of this concept has already been presented at other things. First of all, the image is converted to grayscale and blurred it a bit.

```
Image = o_cv2.imread(args[“image”])

gray = o_cv2.o_cvtColor(image, o_cv2.COLOR_BGR2GRAY)

cv.show = o_cv2.GaussianBlur(gray, (7, 7), 0)
```

With the following boundary detection (Canny) comes into the picture with the grays, to be followed by a 1-to set up and remove the repetition and the removal of all the external noise.

```
Sharp edged = o_cv2.Canny(gray, 50, 100)

sharp edged = o_cv2.dilate(edged, None, repeat=1)

sharp edged = o_cv2.erode(edged, None, repeat=1)
```

Polygon on the boundary of the detection map before.

```
Cnts = o_cv2.findContours(edged.the copy (), o_cv2.RETR_EXTERNAL,
 o_cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if imutils.is_o_cv2() else cnts[1]
```

The outlines are listed in order from the left to the right, which means that in this example, the left-most object is used as the scale.

```
(cnts,_) = contours. Sort_contours (cnts)
color = ((0, 0, 255), (240, 0, 159), (0, 165, 255), (255, 255, 0),
(255, 0, 255))
refObj = None
```

Each contour is in focus, unless we expand it to a rather large size, it is ignored. We may also, in the calculation of the bounding box around the main mass of the contour and put a dot in the upper left -, right -, left-and right-hand corners of the earth. In fact, we can calculate the average of the automatic transmission

```
for c in cnts:
if o_cv2. Contourea(c)
 box = o_cv2.minAreaRect(c)
 box = o_cv2. To resume it.BoxPoints(box) if imutils.is_o_cv2() else
o_cv2.boxPoints(box)
 box = array(box, dtype=”int”)
```

box box = perspective.order_points(box)

cX =, for example, the average(box [:, 0])

cY =, for example, the average(box [:, 1])

Once the program is to identify the first (far left) is a model of objects in our project. To determine the center point of the object, and then calculate the instant distances between the center points to create a new link object.

If refObj, it is not:

(tl, tr, br, bl) = box

(tlblX, tlblY) = midpoint(tl, bl)

(trbrX, trbrY) = midpoint(tr, br)

D = dist.euclid((tlblX, tlblY), (trbrX, trbrY))

_refObj = (box (cX, cY), (D / args[“width”])

continue

We will draw the outlines of the figure, and set the reference coordinates and the coordinates of the object in the study of the effect provided in the next section.

Orig = image.copy()

o_cv2.drawContours(orig, [box.type (int”)], -1, (0, 255, 0), 2)

```
o_cv2.drawContours(orig, [refObj[0].type (int")], -1, (0, 255, 0), 2)

refCoords = eg vstack([refObj[0], refObj[1]])
objCoords = eg vstack([box (cX, cY)])
```

In the end, we make use of the reference co-ordinates of the object's coordinates, and show that the distance vector from each of the four corners of the reference objects at the appropriate angle, the object is found, the reference distance to calculate the value of the exact scaling of the distance.

```
# loop over the original points
for(_xA, _yA), (xB, yB), color) in zip(refCoords, objCoords, color):
# drawing-circles, up to date, points, and
 # connect them with a line.
 O_cv2.the circle(orig, (int(xA), int(yA)), 5, color, -1)
 o_cv2.the circle(orig, (int(xp), int(yB)), 5, color, -1)
 o_cv2.line(orig, (int(xA), int(yA)), (int) (xp), int(yB)),
color, 2)

 # in the calculation of the instant and the distance between the coordinates of
# and then, in the distance, in pixels, of the distance (in pixels).
 # of units
 D = dist. Euclid((xA, yA), (xB, yB)) / refObj[2]
```

```
(mX, mY) = center(xA, yA), (xB, yB))
 openo_cv2.putText(orig, "{:.1 f} (s).- size(D)), (int mX, int my - 10)),
o_cv2.FONT_HERSHEY_SIMPLEX, 0.55, color, 2)

 # display the resulting
o_cv2-image. Imshow ("Image", org)
 o_cv2.waitKey(0)
```

3.4 Classifying People According To Distancing

1. Pre-Processed images taken by a video camera for detecting the contours of the components of the images (using Python)
2. Classification of each contour as a face or not (also using Python)
3. Approximation of the distance between faces detected by the camera

For accomplishing the Python parts, we have used the OpenCV library, which implements most of the state-of-the-art algorithms for computer vision. It has been written in C++ but has bindings for both Python and Java. It also has an Android SDK that could help extending the solution for cellular mobile devices. Such a mobile application might be specifically useful for live and on demand assessments of social distancing using Python.

Steps 1 and 2 in our process are generally straightforward because OpenCV provides methods to get results in a single line of code. The following

code written in Python creates a function faces_dist, which takes care of detecting all the faces in an image:

```
def faces_dist(the_classifier, ref_width, ref_pix):
    ratio_px_cm = ref_width / ref_pix
    cap = cv2.VideoCapture(0)
    while True:
        _, img = cap.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # detect all the faces on the image provided
        faces = classifier.detectMultiScale( gray, 1.1, 4)
        annotate_faces( img, faces, ratio_px_cm)

        k = cv2.waitKey(30) & 0xff
        if k==27:
            break
    # releasing the camera
    cap.release()
```

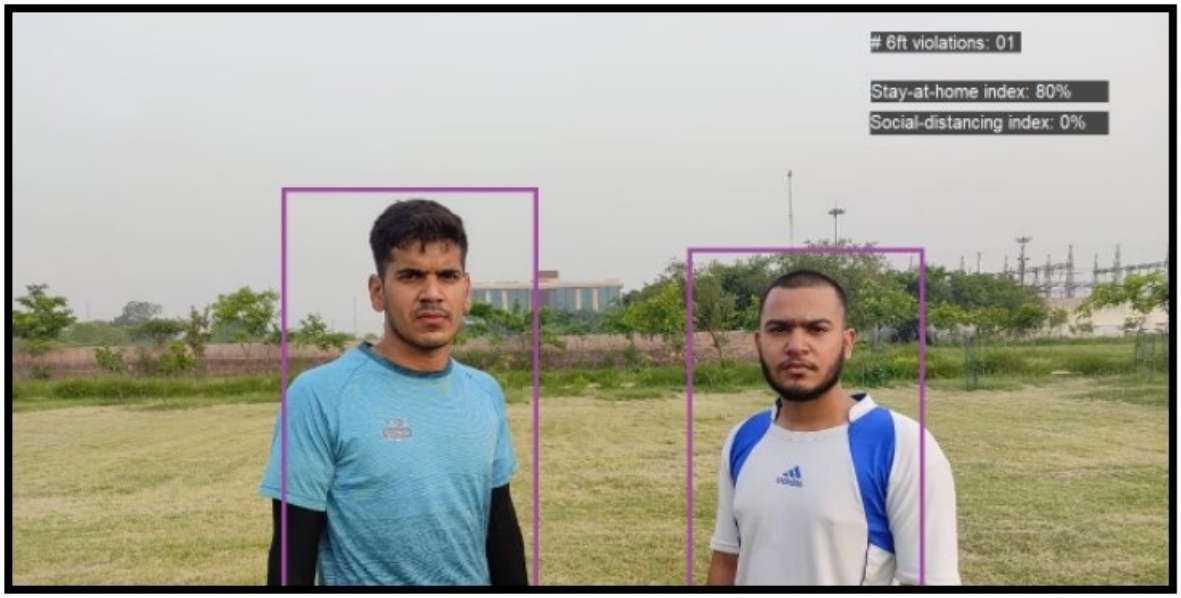

Fig 3.5 Distance recognition 1

After starting the camera in video mode, we enter into an infinite loop to process each image streamed from the video. To simplify identification, the routine transforms the image to grayscale, and then, using the classifier passed as an argument, we get a list of tuples that contains the X, Y coordinates for each face detected along with the corresponding Width and Height which gives us a rectangle.

```
def annotate_faces( img, faces, ratio_px_cm ):
    points = []
    for (x, y, w, h) in faces:
        center = (x+(int(w/2)), y+(int(h/2)))
        cv2.circle( img, center, 2, (0,255,0),2)
```

```
                for p in points:
                        ed = euclidean_dist( p, center ) * ratio_px_cm
                        color = (0,255,0)
                        if ed < MIN_DIST:
                                color = (0,0,255)
                        # draw a rectangle over each detected face in the
frame
                        cv2.rectangle( img, (x, y), (x+w, y+h), color, 2)
                        # put the distance as text over the face's rectangle
                        cv2.putText( img, "%scm" % (ed),
                                (x, h -10), cv2.FONT_HERSHEY_SIMPLEX,
                                0.5, color, 2)
                        # draw a line between the faces detected
                        cv2.line( img, center, p, color, 5)
                points.append( center )

        cv2.imshow('img', img )
```

Object Detection Using Python

We have used a classifier to do human face detection. In simple manner, a classifier can be thought of as a function that chooses a category for a given object. In our case, the classifier is provided by the OpenCV library. OpenCV has already been pre-trained against an extensive dataset of

images featuring human faces. As the result, it can be highly reliable in detecting areas of an image that correspond to human faces. But because real world use cases can differ, OpenCV provides two models for detection:

Local Binary Pattern (LBP) divides the image into small quadrants (3×3 pixels) and checks if the quadrants surrounding the center are darker or not. If they are, it assigns them to 1; if not 0. Given this information, the algorithm checks a feature vector against the pre-trained ones and returns if an area is a face or not. This classifier is fast, but prone to error with higher false positives.

Haar Classifier is based on the features in adjacent rectangular sections of the image whose intensities are computed and compared. The Habr-like classifier is slower than the given LBP, but typically far more accurate.

Our use case for object detection is quite specific since it pertains to social distancing: we need to detect faces, which both classifiers are suited for. But we also need to build an area that represents the most prominent part of the face. This is due to the order to calculate the distance between faces, we first need to approximate the distance the faces are from the camera, which will be based on a comparing manner of the widths of the facial areas detected.

As a result, we have chosen the Habr Classifier since in our tests the rectangle detected for the face was better than that approximated by LBP.

Approximating Distances

Now that we can reliably detect the faces in each image, for step 3: calculating the distance between the faces. That is, to approximate the distance between the centroid of each rectangle drawn. To accomplish that, we have to calculate the ratio between pixels and cm measured from a known distance for a known object reference. The reference_pixels function calculates this value which is used in the faces_dist function described earlier:

Given these parameters, our code will calculate the focal length of the camera, and then use detect_faces.py to determine the approximate distances between the faces the camera detects. To stop the infinite cycle, just press the ESC key. The GitHub repo also contains reference images and two datasets for OpenCV's classifiers.

classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')#classifier = cv2.CascadeClassifier('lbpcascade_frontalface_improved.xml')

Fig 3.6 Distance recognition 2

```
IMG_SRC = sys.argv[1]
REF_DISTANCE = float(sys.argv[2])
REF_WIDTH = float(sys.argv[3])
FL = largest_marker_focal_len( IMG_SRC, REF_DISTANCE, REF_WIDTH )
faces_dist(classifier, REF_WIDTH, REF_WIDTH, FL)
```

3.5 Working of the algorithm in monitoring social distancing

1. The first step is to import all the pre-required libraries.

```
Import cv2
import os
import matplotlib
import pandas
import time as t
import csv
```

2. The second step is to aquire the frame relative to the screen and in the frame also, receive the area in which the algorithm has to detect people and the distance between them.

```
While(0):
    def get_mouse_points(event, x, y, flags, param):
        # marked 4 points on the frame zero of the video that will be warped
        # marked 2 points on the frame zero of the video that are 6 feet away
        global mouseX, mouseY, mouse_pts
        if event == cv2.EVENT_LBUTTONDOWN:
            mouseX, mouseY = x, y
            #cv2(image, (x, y), 10, (0, 0, 0), 0)
```

```
if “mouse_pts” not in globals():
        mouse_pts = []
mouse_pts.append((x, y))
print(“Point detected”)
print(mouse_pts)
```

3. The next step is to link the video(if pre recorded) to the path of the algorithm so that the algorithm can work upjn it and localize the file.

```
# Command-line input setup
parser = argparse.ArgumentParser(description=”SocialDistancing”)
parser.add_argument(
        “—videopath”, type=str, default=”sample_download”, help=”Path to the video file”
)
args = parser.parse_args()

input_video = args.videopath

# Define a DNN model
# Bird Eye View
# Color: 195,0,255
DNN = model()
```

```
# Get video handle
cap = cv2.VideoCapture(input_video)
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
fps = int(cap.get(cv2.CAP_PROP_FPS))
```

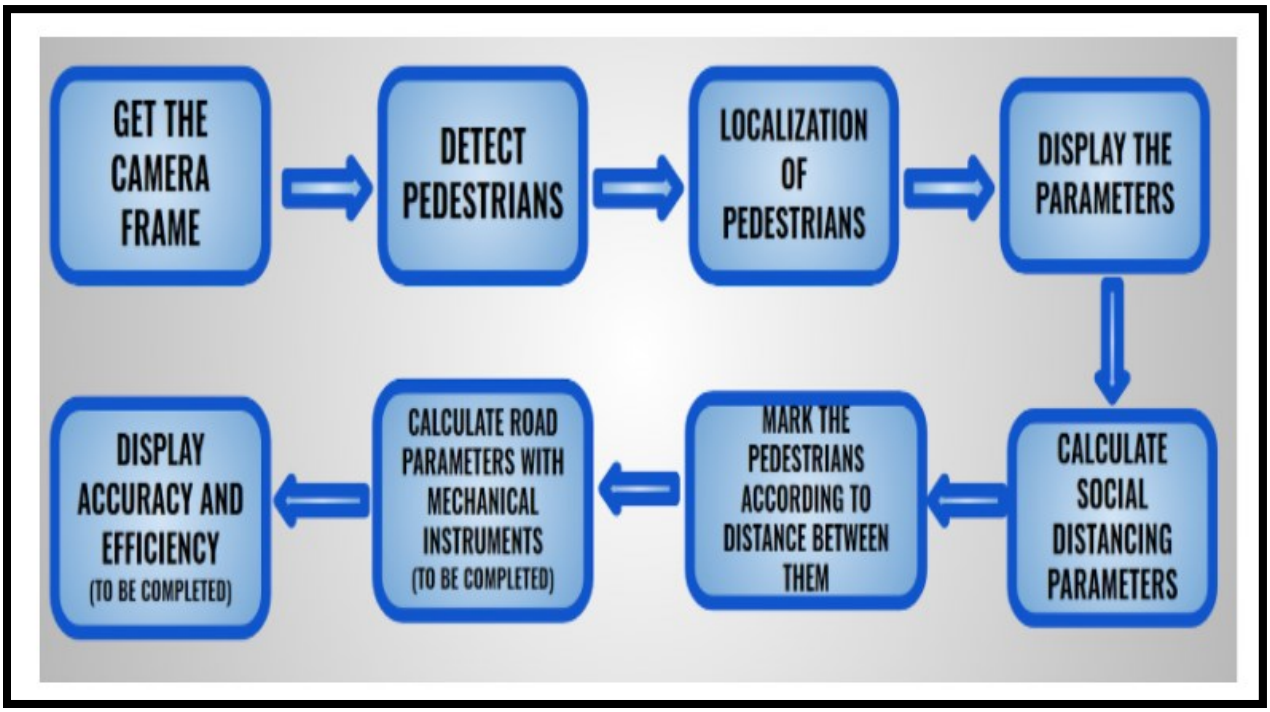

Fig 3.7 Working of algorithm

```
scale_w = 1.2 / 2
scale_h = 4 / 2
```

4. Detecting people using OpenCV and calculating the distance between them in real-time or in a pre recorded video using image processing in each video frame.

```
SOLID_BACK_COLOR = (41, 41, 41)
# Setup video writer
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_movie = cv2.VideoWriter("Pedestrian_detect.avi", fourcc, fps, (width, height))
bird_movie = cv2.VideoWriter(
        fourcc, fps, (int(width * scale_w), int(height * scale_h))
)
# Initialize necessary variables
frame_num = 0
total_pedestrians_detected = 0
total_six_feet_violations = 0
total_pairs = 0
abs_six_feet_violations = 0
pedestrian_per_sec = 0
sh_index = 1
sc_index = 1

cv2.namedWindow("image")
cv2.setMouseCallback("image", get_mouse_points)
num_mouse_points = 0
first_frame_display = True
```

```
# Process each frame, until end of video
while cap.isOpened():
    frame_num += 1
    ret, frame = cap.read()

    if not ret:
        print("end of the video file...")
        break

    frame_h = frame.shape[0]
    frame_w = frame.shape[1]

    if frame_num == 1:

# Ask user to mark parallel points and two points 6 feet apart. Order bl, br, tr, tl, p1, p2
            while True:
                image = frame
                cv2.imshow("image", image)
                cv2.waitKey(1)
                if len(mouse_pts) == 6:
                    cv2.destroyWindow("image")
```

```
                    break
                first_frame_display = False
            four_points = mouse_pts

            # Get perspective
            M, Minv = get_camera_perspective(frame,
four_points[0:4])
            pts = src = np.float32(np.array([four_points[4:]]))
            warped_pt = cv2.perspectiveTransform(pts, M)[0]
            d_thresh = np.sqrt(
                (warped_pt[0][0] - warped_pt[1][0]) ** 2
                + (warped_pt[0][1] - warped_pt[1][1]) ** 2
            )
            bird_image = np.zeros(
                (int(frame_h * scale_h), int(frame_w *
scale_w), 3), np.uint8
            )

            bird_image[:] = SOLID_BACK_COLOR
            pedestrian_detect = frame

        print("Processing frame: ", frame_num)
```

```
# draw polygon of detected object
pts = np.array(
        [four_points[0], four_points[1], four_points[3],
four_points[2]], np.int32
)
cv2.polylines(frame, [pts], True, (0, 255, 255), thickness=4)

# Detect person and bounding boxes using DNN
pedestrian_boxes, num_pedestrians = DNN.detect_pedestrians(frame)
```

5. Calculation of social distancing parameters like stay-at-home index and social distancing index is completed using the following algorithm. This is important so as to summarize the whole frame in a singular line so that a manual authority can easily understand these parameters if the place of the frame is very busy.

```
if len(pedestrian_boxes) > 0:
        pedestrian_detect = plot_pedestrian_boxes_on_image(frame,
pedestrian_boxes)
        warped_pts, bird_image = plot_points_on_bird_eye_view(
            frame, pedestrian_boxes, M, scale_w, scale_h
        )
```

```
                six_feet_violations, ten_feet_violations, pairs =
plot_lines_between_nodes(
                        warped_pts, bird_image, d_thresh
                )
                # plot_violation_rectangles(pedestrian_boxes, )
                total_pedestrians_detected += num_pedestrians
                total_pairs += pairs

                total_six_feet_violations += six_feet_violations / fps
                abs_six_feet_violations += six_feet_violations
                pedestrian_per_sec, sh_index =
calculate_stay_at_home_index(
                        total_pedestrians_detected, frame_num, fps
                )
```

6. This piece of code displays the final calculated parameters.

```
        last_h = 75
        text = "# 6ft violations: " + str(int(total_six_feet_violations))
        pedestrian_detect, last_h = put_t

        if total_pairs != 0:
sc_index = 1 - abs_six_feet_violations / total_pairs
```

```
                text = "Social-distancing Index: " + str(np.round(100 *
sc_index, 1)) + "%"
                pedestrian_detect, last_h = put_text(pedestrian_detect, text,
text_offset_y=last_h)


                cv2.imshow(  pedestrian_detect)
                cv2.waitKey(1)
                output_movie.write(pedestrian_detect)
                bird_movie.write(bird_image)
```

3.5 Efficiency Detection

The efficiency of the final tracker can be displayed using two parameters:

1. Accuracy of the tracking application

2. A Cut-off angle after which it shows aberrations in prediction.

As discussed earlier, the mechanical instruments required to calculate accuracy and cut-off angle would be a theodolite and a laser distance meter.

3.5.1 Measuring error using laser distance meter

The laser distance meter is used to find the real distance between two people and then the observations are compared with whether the tracker shows a violation or not. The following observation table shows if the violations are shown or not shown based on the distance, with reading with close proximity to 6ft where the chances of error are quite higher.

Based on the observations (in the close range of 6ft) the error occurring in the tracker is derived and noted down to calculate accuracy. The laser distance meter provides us a way to calculate accurate and precise measurements so as to find even a small error and improve the tracker further. The following table shows these observations: -

OBSERVATION TABLE 1

Laser Distance Meter measurements

Actual Distance (ft)	Violation shown	Actual Violation	Deviation from 6ft, if error observed
5.8	Yes	Yes, as d < 6ft	NA
5.85	No	Yes, as d < 6ft	0.15
5.9	No	Yes, as d < 6ft	0.1
5.95	No	Yes, as d < 6ft	0.05
6.0	Yes	No, as d >= 6ft	0
6.05	Yes	No, as d >= 6ft	-0.05
6.10	No	No, as d >= 6ft	NA

Table 3.1 Laser distance meter readings

The observation table -1 shows the differences in actual violation and violation shown by the tracker, thereby giving us the error in detection and in turn the accuracy will be calculated.

3.1.1 Measuring elevated vertical angle using Theodolite

The need for measuring vertical angles is simply to find a cut-off angle till which the tracker works perfectly. This is necessary to find so that we do not get confused later on about the working of tracker on elevated surfaces like fly-overs and stairs.

The following observation table shows the measurements of vertical angles taken with the help of a theodolite and thereby, find a cut-off angle. It is seen that for larger deviations, the cut-off angle is greater, which is good for the efficiency of the application.

OBSERVATION TABLE 2

Theodolite measurements

Inclination angle (degrees)	Error in observation for 0.5 ft deviation(6.5 ft)	Error in observation for 1 ft deviation(7 ft)
15.2	No	No
21.2	No	No
27.5	No	No
32.4	No	No
35.7	Yes	No
38.6	Yes	Yes

Table 3.2 Theodolite readings

CHAPTER 4 RESULTS AND DISCUSSION

The chapter presents the experimental analysis and results carried out by experimenting with the tracking application made. It also presents the deductions made by the theodolite and laser distance meter used earlier to find the cut-off angle and accuracy respectively. Point to point discussions are made so as to reach towards the final result of the project.

4.1 Determination of the accuracy of the tracker

- The accuracy is simply determined by subtracting the error from 100.
- The error earlier found with the help of laser distance meter is calculated by simply taking the mean of all the deviations and dividing it by total readings taken.
- The error is found out to be 1.16%.
- Error is found from the deviations from correct observations in close proximity to 6ft range, where chances of finding an error are maximum.
- It is verified that no error is found except in the readings taken in the observation table.
- Hence the accuracy is found out to be 98.83%.

4.1 Determination of the cut-off angle

• The cut-off angle is found from two separate deviations, that are 0.5 ft and 1 ft respectively.

• The cut off angle is measured using elevated surfaces vertical angles measured by theodolite.

• The cut off angle for 0.5 ft deviation is found out to be 35.7 degrees.

• The cut off angle for 1 ft deviation is found out to be 38.6 degrees.

• Hence, the cut off angle is taken as 35.7 degrees.

CHAPTER 5 CONCLUSIONS AND FUTURE SCOPE

This chapter focuses on the observations and outcomes of the experimental investigation of the tracking application made. The immediate results are also taken into consideration to determine the conclusion. Also, an attempt has been made to determine the future scope, opportunities and possible outcomes of the project in the future.

5.1 Conclusions

• The project has been designed successfully on the terms of completion of the application for tracking people.

• The tracker is successful in detecting people, calculating distance between them, determine if social distancing violations are happening or not.

• It also can take the count of social distancing violations made in addition to detection and calculate parameters like social distancing index and stay-at-home index.

• The calculations are further made to determine the accuracy of the tracking application and also the cut-off angle till the app is completely reliable. Most of the elevations on which people are made to walk are less than 30 degrees in vertical inclinations. The tracking application stands out with a cut-off angle of 35.7 degrees and hence can be said to be extremely reliable.

• The algorithm is also exposed to training with the help of deep learning to further improve its accuracy. The neural network models help increase the accuracy using the pre-built data models which are applied through the tensorflow library in python.

5.2 Future Scope

• The usability of the tracker is increased multiple-folds in the time of a pandemic like COVID-19. However, there are other applications too which will always remain, like hospitals or other places where social distancing is required to prevent spread of a disease or even separate people at a distance in a busy place.

• It is expected to be much more accurate also with the increase in development of Artificial Intelligence over time, which is sky-rocketing these days and expected to grow much more in this decade. With the help of advanced neural networking, there might be a time that the tracker is developed error-free and without any cut-off angle also.

REFERENCES

1) abdul kadhar.K, Mohaideen & Anand, G.. (2021). Basics of Python Programming.

2) mm, Malathi & Balaji, Sinthia. (2021). An Advanced Image Processing Prototype for Corrosion Finding Using Image Processing. Journal of Computational and Theoretical

3) Banda, Francis. (2020). OVERVIEW of IMAGE PROCESSING OVERVIEW OF IMAGE PROCESSING Contents.

4) Liang, Wei & Xu, Pengfei & Guo, Ling & Bai, Heng & Zhou, Yang & Chen, Feng. (2021). A survey of 3D object detection. Multimedia Tools and Applications.

5) Nikolenko, Sergey. (2021). Deep Learning and Optimization.

6) D., Kavitha. (2021). Multiple Object Recognition Using OpenCV. Revista Gestão Inovação e Tecnologias.

7) Panchal, Omkar. (2021). Road Sign Recognition and Lane Detection using CNN with OpenCV. International Journal for Research in Applied Science and Engineering Technology.

8) ouchra, hafsa & belangour, abdessamad. (2021). Object detection approaches in images: a survey.

9) Cunningham, Padraig & Kathirgamanathan, Bahavathy & Delany, Sarah. (2021). Feature Selection Tutorial with Python Examples.

10) Zhao, Qianji & Shang, Zequn. (2021). Deep learning and Its Development. Journal of Physics: Conference Series. Gollapudi, Sunila. (2019). OpenCV with Python.

11) Pettersen, Bjørn. (2020). Introducing theodolites for mapping in Norway. Norsk Geografisk Tidsskrift - Norwegian Journal of Geography.

12) Subero, Armstrong. (2021). Programming Microcontrollers with Python: Experience the Power of Embedded Python.

13) Chen, Shengyi & Fan, Chaoran & Xu, Zhilei & Zhang, Feng & Li, Xiaoxue. (2021). An approach of target photometric measuring for theodolite visible light imaging. Journal of Physics: Conference Series.

14) El-Sheikha, Zaki. (2020). A Method of Theodolite Installation at the Intersection of Two perpendicular lines (Dept.C). MEJ. Mansoura Engineering Journal.

15) Lingmei, Li & Ruijun, Lu & Yuanyao, Li & Hongguang, Liu. (2019). Principle and verification of hand-held laser distance meter.

16) Ketkar, Nikhil & Moolayil, Jojo. (2021). Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch.

17) Zheng, Zhaohui & Ye, Rongguang & Wang, Ping & Wang, Jun & Ren, Dongwei & Zuo, Wangmeng. (2021). Localization Distillation for Object Detection.

18) Hong, Hyungi & Chung, Mokdong. (2020). Performance Improvement on Object Detection for the Specific Domain Object Detecting.

19) Leng, Jiaxu & Liu, Ying. (2021). Context augmentation for object detection. Applied Intelligence.

20) Cheng, Keyang & Wang, Ning & Li, Maozhen. (2021). Interpretability of Deep Learning: A Survey.

OVERALL EXPERIENCE

The most important purpose for going on a journey is not to just reach there, but explore the places that came in between, and we took great pleasure in doing just that. From the beginning till the very end, it has not only been an in-depth introduction to what can be the future to human detection using cameras, monitoring social distancing using an algorithm. After the completion of the theoretical knowledge, we are more impelled than ever before to fabricate the tracker application and substantiate our learnings and observations. We are grateful to everyone who heled us directly or indirectly in the successful completion of our project. Besides, it has been a thorough and an outright experience. We are keen to know what the future holds for us and how we as young engineers can amend the methods and techniques for a better and more efficient world.