INTRUDER ALERT SYSTEM WITH FACIAL RECOGNITION AND OBJECT DETECTION

A PROJECT REPORT

Submitted by

KADHIRAVAN.G 211418104105

KALAIVANAN.K 211418104106

JAYAKANTH.V 211418104097

In partial fulfillment for the award of the

degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALARENGINEERINGCOLLEGE

(AnAutonomousInstitution,AffiliatedtoAnnaUniversity,Chennai)

MAY 2022

PANIMALARENGINEERING COLLEGE

(AnAutonomousInstitution,AffiliatedtoAnnaUniversity,Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"INTRUDER ALERT SYSTEM WITH FACIAL RECOGNITION AND OBJECT DETECTION"** is the bonafide work of "**KADHIRAVAN.G, KALAIVANAN.K, JAYAKANTH.V(211418104105, 211418104106, 211418104097)"** who carried out the project work under my supervision.

SIGNATURE

Dr.S.MURUGAVALLI,M.E.,Ph.D.,
HEAD OF THE DEPARTMENT

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

A.N.SASIKUMAR
SUPERVISOR
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above mentioned students were examined in the End Semester project Viva-Voice held on ...27.05.2022...

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We KADHIRAVAN.G, KALAIVANAN.K, JAYAKANTH.V (211418104105, 211418104106, 211418104097) hereby declare that report title "INTRUDER ALERT SYSTEM WITH FACIAL RECOGNITION AND OBJECT DETECTION" under the guidance of A.N.SASIKUMAR is the orginial work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**,**Dr.C.SAKTHIKUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr.S.MURUGAVALLI , M.E.,Ph.D.,**for the support extended throughout the project.

We would like to thank my project guide **Mr. A.N.SASIKUMAR, M.E.,**… and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

KADHIRAVAN.G, KALAIVANAN.K, JAYAKANTH.V

ABSTRACT

The program is to enhance the currently existing CCTV systems with machine learning to make the data retrieval and area monitoring secure and easy. The program gets the video feed from the CCTV cameras and mark the time when a vehicle or a person passes through the frame. The program will help people to save a lot of time at the time of data retrieval there is need for them to re-watch the whole recording looking for a single person or a vehicle. Also, this system can be used to monitor the people, vehicles, sports ball, dogs etc. when they appear in the frame. The time when any of the entities mentioned above enter the frame it will be recorded in the document which can be used as reference while rewatching for skipping the times when there is no necessary information in the video frame.

This project adds features like facial recognition, Age estimation, number plate recognition and animal detection in order to make it more effective while searching for event details when needed. The collected is stored in a timeline.csv file locally in the system which can be referred to check for events and to make the searching process efficient. All the number plates crossing the frame are recognized and the text in the plates are also documented along with the cropped pic of the number plate for reference.

LIST OF FIGURES

FIG NO.	FIGURE DESCRIPTION	PAGE NO.
4.1.1	**Class Diagram**	7
4.1.2	**Sequence Diagram**	9
5.1.1	**General Architecture**	10
5.3.1	**Finding All The Faces**	14
5.3.2	**Converted to b/w**	14
5.3.3	**Pixels Extraction**	15
5.3.4	**Pixel with Gradient**	16
5.3.5	**Simple Image**	17
5.3.6	**HOG Pattern**	17
5.3.7	**Identification With Age**	18
5.3.8	**Angle Normalization**	18
5.3.9	**Face Landmarks**	19
5.3.10	**Detection Using Face Landmarks**	20
5.3.11	**Centered image**	21
5.3.12	**Triplet Training**	23
5.3.13	**128 Measurements**	24
5.4.1	**MS COCO Detection**	25
5.4.2	**FPN Building Blocks**	26
5.4.3	**4 Types of pyramids**	27
5.4.4	**Parameter of NN**	29
5.4.5	**Example of YOLOv4**	29
7.1	**Multiple people in frame**	57
7.2	**Multiple people in long ranges**	58
7.3	**People and vehicles**	58
7.4	**Number Plate identification**	59
7.5	**Cropped Number Plate**	59
7.6	**Detect Multiple objects**	60
7.7	**Person Recognition**	61

7.8	**Number Plate Recognition**	61
7.9	**Saved Image**	62
7.10	**State and number extraction**	62
7.11	**Multiple People recognition**	62
7.12	**Logged Data**	63
7.13	**Gender and Age Recognition**	64

TABLE OF CONTENTS

CHAPTERNO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF FIGURES	v
1.	**INTRODUCTION**	1
	1.1 Problem Definition	1
2.	**LITERATURE SURVEY**	2
3.	**SYSTEM ANALYSIS**	6
	3.1 Existing System	6
	3.2 Proposed system	6
	3.4 Hardware Environment	6
	3.5 Software Environment	7
4.	**SYSTEM DESIGN**	7
	4.1 UML Diagrams	7
5.	**SYSTEM ARCHITECTURE**	10
	5.1 General Architecture	10
	5.2 The Design Of Motion Detection And Face Identification	11
	5.3 Algorithms	13
	5.4 Module Design Specification	30
6.	**SYSTEM IMPLEMENTATION**	32
	6.1 Explanation for important modules	33
	6.2 Working Python Codes	42

CHAPTERNO.	TITLE	PAGE NO.
7.	**SYSTEM TESTING/ PERFORMANCE ANALYSIS**	57
	7.1 Test Cases & Reports	57
8.	**CONCLUSION**	60
	8.1 Results& Discussion	61
	8.2 Conclusion and Future Enhancements	64
	REFERENCES	65

CHAPTER 1

1. INTRODUCTION

CCTV camera systems have become so common that even governments have started to install cams on each and every junction in many cities across India. Even though, there are CCTV systems installed and collects recordings on the events happening the particular field of view of the cameras, It's difficult for people to retrieve the particular bit of information from the very large data that's being recorded. Sometimes the time taken to acquire the particular event from the whole recording may not be worth it. Usually in order to retrieve the particular bit of data form the whole recording people tend to traverse through the recordings at a faster pace which again leads to a very chance of human error.

This reason creates a need for this program to make a video analyzing program which can eliminates the chance of human error by detecting and documenting the events occurring in short notations for easy accessing purpose. So, when it comes to creating a program for this purpose then, it can collect all the available to data and pre-organize all the necessary details form the events then and there as the events are being recorded so that the retrieval of data is easy. Check out the ways in which the details are collected and organized in this paper.

1.1 PROBLEM DEFINITION:

In the existing CCTV systems the problem that people commonly face is rewatching the whole footage to find any event that has occurred during the day. It takes a lot of time and effort for people to get the data about the event occurring for split second in a very long footage. This process needs a lot of attention and concentration to retrieve the necessary data. This may lead to a lot of human errors and the person may have to rewatch the whole video again and again.

CHAPTER 2

2. LITERATURE SURVEY

1) According to **Ade Nurhopipah** and **AgusHarjoko**(July 2018)
   - The use of CCTV has created a huge impact in todays world.
   - CCTV has changed the way of surveillance into an integrated intelligent control system. Facial recognition and motion detection is used in CCTV to make the system look effective and efficient.
   - Images are acquired and used for motion detection and if any motion is detected the time stamp and image information with detected motion will be stored. If the motion exceeds then facial detection will be used.
   - The images will be the input of facial detection. These images will be stored in the database.

2) According to **Apoorva Raghunandan**, **Mohana**, **Pakala Raghav**, **H. V. Ravish Aradhya** (Apr 2018)
   - The concept of Object detection is used in various fields in world.
   - This paper discusses various object detection methods for face detection, skin detection, color detection, shape detection.
   - This method also enables an option to detect the color of the shirt and pant who comes under the frame.

3) According to **Serhii Maksymenko** (Sept19)
   - A program can develop face recognition-based biometric identification system using OpenCV library, DLib and real-time streaming via video camera.
   - The steps involved here is to detect the face and needs to be recognized quickly. If the person is authorized, it allows access for the person.

- The application uses a deep neural network to find a face within its stream.

4) According to **Jayati GhoshDastidar** and **Rana Biswas** (Dec 2015)
   - The detection and tracking of objects is found as a challenging problem. They used computer vision and image processing these objects.
   - The static background information is collected with different illuminations. Then the real time images of those objects that come inside these frames are used to detect the human detection.
   - If that person enters the frame of the CCTV, then it will track the movement of the person and identify the person.

5) According to **Shraddha Mane** and **SupriyaMangale (**June 2018)
   - In this project the program detects the object and track with a computer vision algorithm there is a big problem
   - When two or more vehicles merge or combine with each other hence it uses tensor flow object detection API another problem is robust object detection
   - Which can be solve by CNN based object tracking algorithm the proposed approach achieved a accuracy of 90.88%.

6) According to **Andrej Vel'as, Milan Kutaj** and **Martin Ďurovec**(Oct 2017)
   - In this paper the intruder detection of protected zone can be performed within the object perimeter such as CCTV monitor screen systems.
   - It can be done at a certain level the intruder crosses the boundary points or at the certain level it can be monitored are crossing the detection zone area crossing systems it focussed on changing the camera parameter systems on a video based abilities.

- In experimental testing the camera system satisfies the video motion detection influences the success rate.

7) According to **Sonali Hargude, S.R. Idate**(Aug 2016)
    - Abandoned Object detection is an greater problem in computer vision and image processing technique in which system provides an powerful technique such as I-surveillance the process is used by background subtraction technique
    - Where the object left someone by mistake the system divides into two different phases Abandoned object detection
    - Human blob detection here these are the effective way to find the rejected objects which increases the efficiency of security and experiences the user friendly mechanism.

8) According to **B. Sachin Prabhu, Subramaniam Kalambur** and**Dinkar Sitaram**(Sept 2017)
    - Automatic license plate recognition has been deployed in many countries for traffic management, speed control and mainly used for stolen cars is the process of extracting the car licence plate via a live video feed
    - Extract the plate number and copies the crop number plate image there are some challenging for the system such as different angles, distance, resolution and illumination conditions
    - In this program it's using OpenALPR, k-NN and Convolutional Neural Networks (CNN) based techniques to detect the licence plate under the appropriate conditions it have good accuracy in both live video and still images.

9) According to **DhanarIntan Surya Saputra**and **Kamal Miftahul Amin** (Aug 2016)

- In this paper the human face detection is an application for computer vision in previous research paper about the facial recognition
- There is no matter of how healthy and how old and facial expression such as sad, normal or laugh in some paper there are focussing on the environmental conditions
- In complementary CCTV used in industries, military, airports, shops, offices, factories, and even today many housing have been using it will detects the actual condition of the human present under the camera
- This also used to detect the human faces who crosses the video frame.

10) According to **Ming-Hsuan Yang, D.J. Kriegman**and **N. Ahuja** (Aug 2002)

- The images contains the faces essential to vision based and computer based interaction theses process includes the face recognition and facial expression
- Assume that the faces in the images are identifies and to analyse the information efficiently by using the face detection algorithm
- Sometimes the face in images are have some problem in orientation and lighting conditions such problem faces the shape, color and texture of the given image
- This paper categorize and evaluates the algorithm data collection and benchmarking identifying the limitations and direction for the future research.

CHAPTER 3

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Currently as per our research there are systems which are capable of detecting motion in video feed by comparing the differences between the video frames. There are programs and systems which can perform object detection and face recognition but have not yet been implemented for such a cause as per our research.

3.2 PROPOSED SYSTEM

Over the recent years detecting human being in a video scene of surveillance system is attracting more attention due to its wide range of applications in abnormal event detection. In the project implementing a system where it is able to differentiate a person known to the system and people unknown to the system marked as an unauthorized person.

So, a software is created a software where it detects a person/object in CCTV and log in their data like facial recognition and detect objects like vehicles that pass through its colour and number of the vehicle while marking the time in a document. The recorded document makes the accessing of data very easy for future references.

3.3 HARDWARE ENVIRONMENT

- Processor: intel i7 8700k
- Ram: 16gb ddr4
- Hard disk: 2Tb
- GPU: Nvidia GeForce Rtx 2060
- (Phone /camera) or recorded video as input for processing

3.4 SOFTWARE ENVIRONMENT

- Artificial Intelligence- Machine Learning, Python, Object detection, data logging
- Developed in python using pycharm ide.
- HOGCV algorithm and Yolov4 is used for detection.
- The recorded data will be stored as a text document.
- Input will be take from a real-time running camera (phone cam for testing).

CHAPTER 4

4. SYSTEM DESIGN

4.1 UML DIAGRAMS

4.1.1 CLASS DIAGRAM

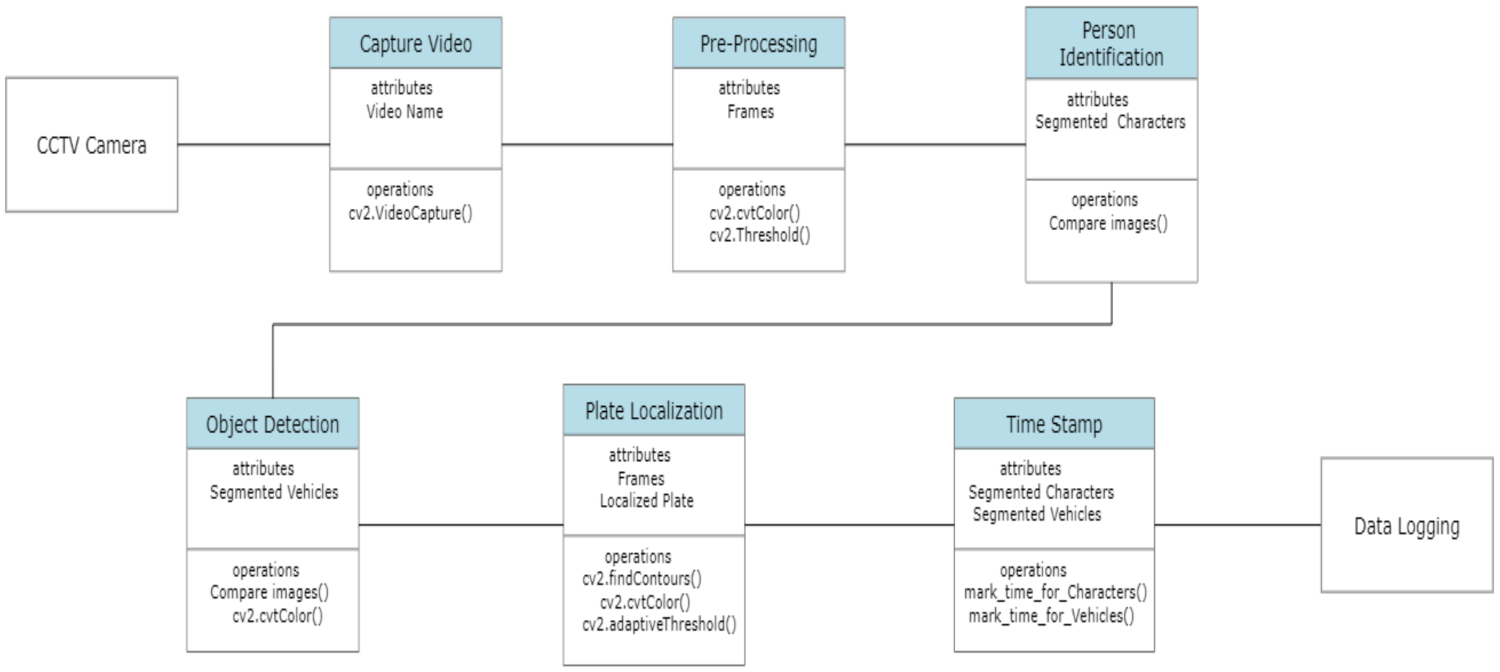


Fig 4.1.1

The above figure shows the class diagram of our system. It will capture the video who are nearby home are trying to entering the home by using an method cv2.VideoCapture() followed by an threshold (is an type of image segmentation where it can change the pixel of an image to easily analyze) and it will identify the person who entering into the home as known by using a method compare.

Images (is used to compare the image stored in the database) If unknown person crosses it marks as unknown all the data's are stored in database for the future purposes It is also used to detect the objects which are passes through the camera and named it as a car, bike or a person.

Plate Localization is an method used to locate the frames of the plate (contours – is used to extract the border of the number plate to easily localize it) cv2.FastNLMeansDenoising(Perform image denoising using Non-Local Means Denoising Algorithm) Adaptive threshold plays an important role where it calculated the value of the smaller regions such as(number plate).

It locates an time stamp when the object crosses the camera, it used to easily retrieve the exact footage which the person needs.

It can be both persons/vehicles All the details are bookmarked and stores in database all the details can be viewed whenever needs, It saves our time instead of watching whole recorded video.

4.1.2 SEQUENCE DIAGRAM

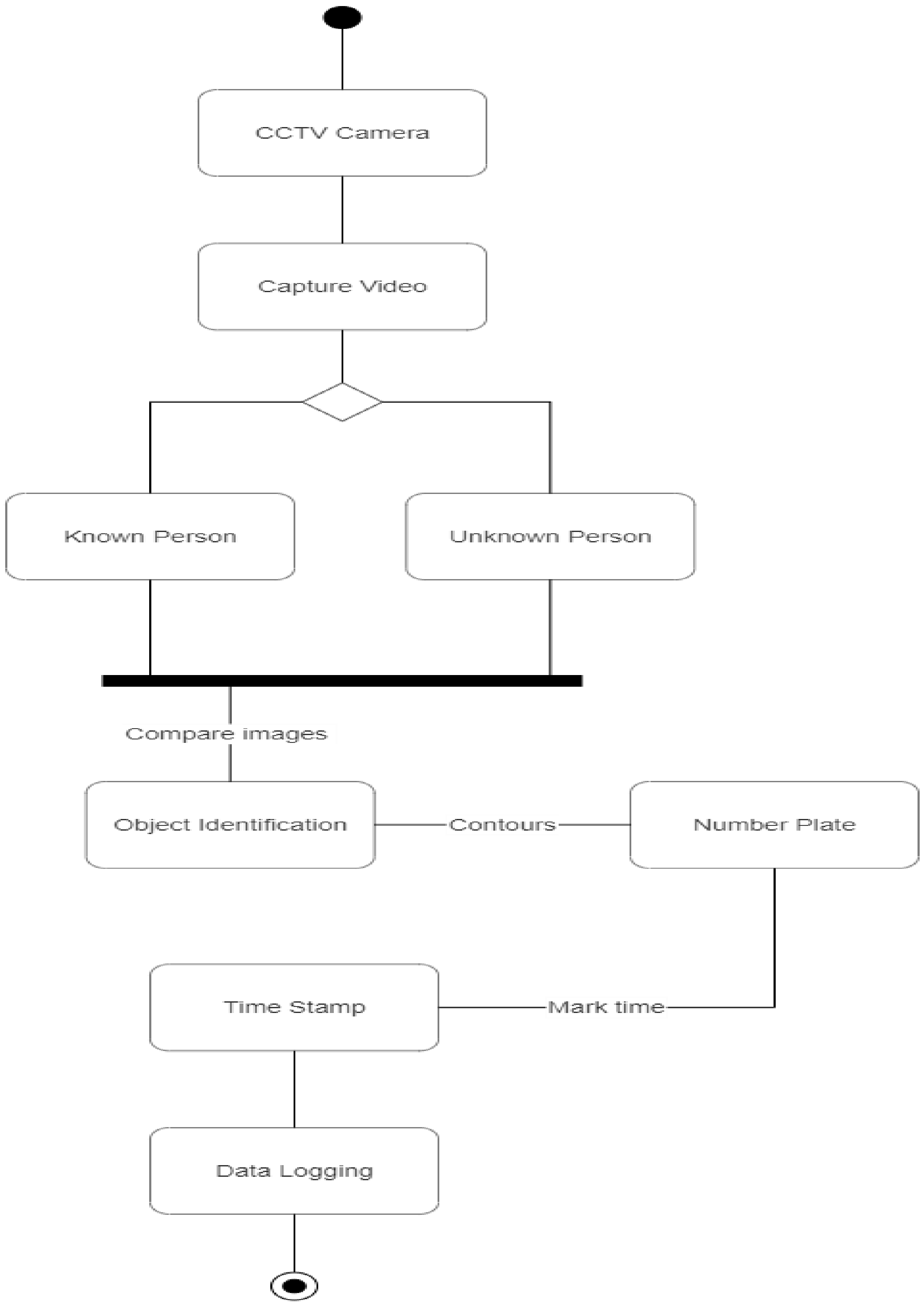


Fig 4.2.2

The above figure shows the sequence diagram of our system. Capture the video feed from CCTV Camera then it can easily identify from the video whether the person as a known or unknown. Then the program is identifying person and vehicle separately extracting all the details in the image/video and then it can encode the face for persons and identify the vehicles in the vehicle category. it use's contours for extracting the details in vehicles numbers plates or things like color of dress in the category of person. Then it is making bookmarks whenever a person or vehicle that cross the camera in the time graph. Time graph is shown in the footage of the CCTV camera when it can select the bookmarks for finding a vehicle or person whenever there is a need to check.

All the details of the person/vehicle and bookmarks in the time graph are then stored in the database. All the information can be retrieved for later use.

CHAPTER 5

5. SYSTEM ARCHITECTURE

5.1 GENERAL ARCHITECTURE

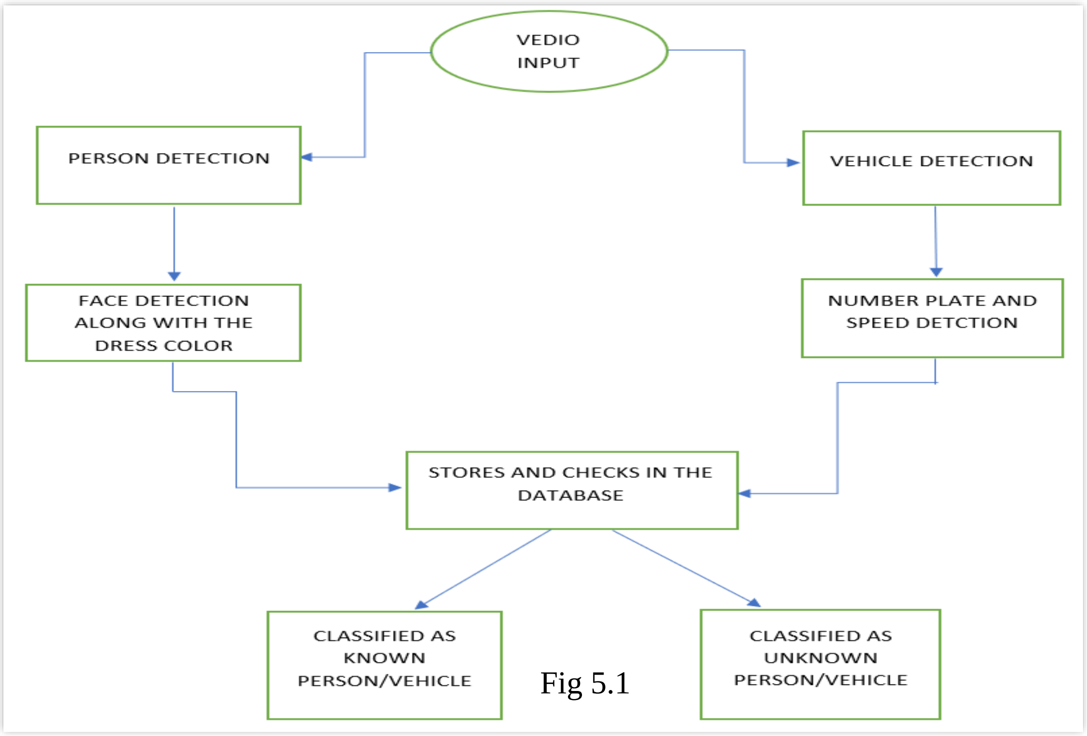

VEDIO INPUT
PERSON DETECTION
VEHICLE DETECTION
FACE DETECTION ALONG WITH THE DRESS COLOR
NUMBER PLATE AND SPEED DETCTION
STORES AND CHECKS IN THE DATABASE
CLASSIFIED AS KNOWN PERSON/VEHICLE
CLASSIFIED AS UNKNOWN PERSON/VEHICLE

Fig 5.1

From the above figure look at the general architecture of our system, where the program get's the video input from the CCTV and then it classifies the objects involved as person and vehicle that are captured in the CCTV. While a person looks at the side of person, he/she can detect the face of the person, also finding his age, his dress colour and also identify the gender of that person.

In the case of vehicle, a person is able to find the number plate and get the data in that number plate and with the help of the video footage people can get the speed at which the vehicle travelled i.e., the time clocked while passing that CCTV camera. Then these data's that are collected from both the vehicle and person the program store all the details in the particular database. The data needs to be verified if previously it's in the database. If the data is not present in the database it classifies as unknown person/vehicle and tries to encode the face and stores in the same database. If the data is already present then it classifies as a known person and give the details of that person/vehicle.

5.2 THE DESIGN OF MOTION DETECTION AND FACE IDENTIFICATION

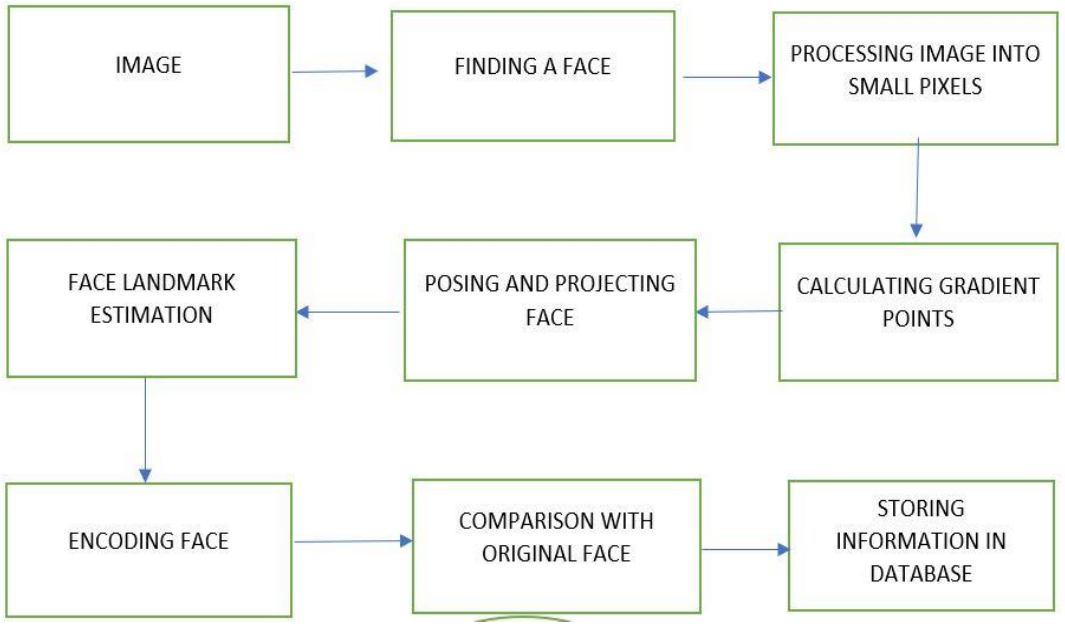


IMAGE
FINDING A FACE
PROCESSING IMAGE INTO SMALL PIXELS
CALCULATING GRADIENT POINTS
POSING AND PROJECTING FACE
FACE LANDMARK ESTIMATION
ENCODING FACE
COMPARISON WITH ORIGINAL FACE
STORING INFORMATION IN DATABASE

Fig 5.2

From the below figure it can be seen that input is given as image or a videotape from a CCTV footage. Also it’s relating the persons face from the footage and determining if the person is authorized person or unauthorized person. If the person is plant out as unauthorized also our system will start to descry the face.

To find faces in an image, it starts by making our image black and white. Also the program looks at every single pixel from the captured image bone at a time. For every single pixel, there is a need to look at the pixels that directly girding it. Also it ’ll have to find how dark the current pixel is compared to the pixels directly girding it. Also the algorithm will draw an arrow representing the direction in which the image is getting darker. However, it ends up with a illustration of numerous arrows, If it repeats that process for each and every single pixel present in the captured frame.

These arrows are called slants and they show the inflow from light to dark across the entire image. It'll make the image into small places of 16x16 pixels each. In each forecourt, it’ll count up how numerous slants point in each major direction. Also it replace the forecourt in the image with the arrow directions that were the strongest. The end result is the program makes the original image into a simple representation that captures the introductory structure of a face in a simple way. The coming step is (protuberance of the image. The program might know others when it looks at the left or right side of the same person as analogous but the system may not be suitable to identify the same.

To make this the program’s using the algorithm called face corner estimation. The introductory idea is that the program comes up with 68 specific points (called milestones) that live on every face’s specifications like shin, lips, eyes, etc.… Also it’ll will train a machine learning algorithm to be suitable to find these 68 specific points on any face. Now that find’s the eyes and mouth also it’s going to use introductory image metamorphoses like gyration and scale that save resemblant lines. This fashion is called affine metamorphoses. The coming step is to render the face, where training of the system to find the face.

Originally, load a know person face, also unknown person face, also a new persons face. Using the neural network if finds the fewest difference between each of the face. Once the system is trained for further than thousand time also it can easily render different faces and give the results.

Also the coming step is to compare the decoded face with the original face from the footage or the image. It can find that the decoded face looks a lot analogous to the original face. The final step is to store all the information regarding the details of the decoded face, It is suitable to use the data for unborn purpose, if it need to get the data for reference.

.3 ALGORITHMS

5.3.1 FACE RECOGNITION

Step 1: Finding all the Faces

- The first step in our channel is face discovery. Obviously it needs to detect the faces in a snap before it can try to tell them piecemeal!
- If you 've used any camera in the last 10 times, you 've presumably seen face discovery in action.
- Face discovery is a great point for cameras. It nowadays can detect faces of people so that it can adjust focus and saturation to provide a great shot . But it'll use it for a different purpose- chancing the areas of the image there's a need to pass on to the coming step in our channel.
- This snap shows the regular face recognition point in a regular smartphones and cameras

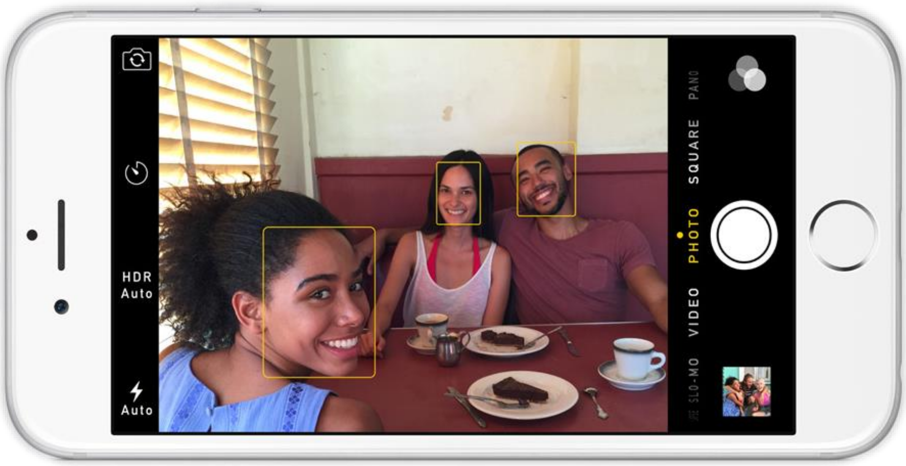


Fig 5.3.1

- Face discovery went mainstream in the early 2000's when Paul Viola and Michael Jones constructed a way to descry faces that was presto enough to run on cheap cameras.
- Still, much more dependable results live now. Program's going to use a system constructed in 2005 called Histogram of Acquainted Slants-or just Overeater for short. To find faces in an image, It'll start by making the image black and white because the program doesn't need colour data to find faces.

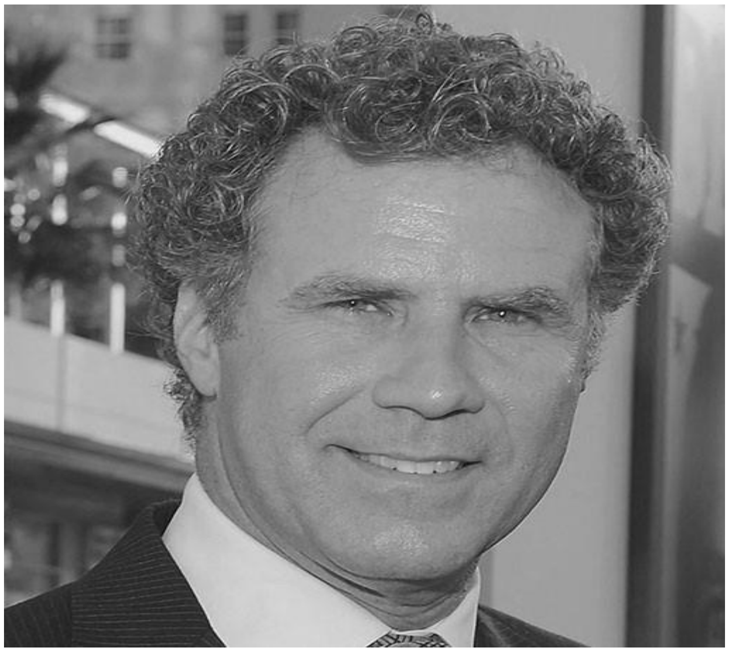


Fig 5.3.2

Also it'll look at every single pixel in our image one at a time. For every single pixel, The Program want's to look at the pixels that directly girding it.

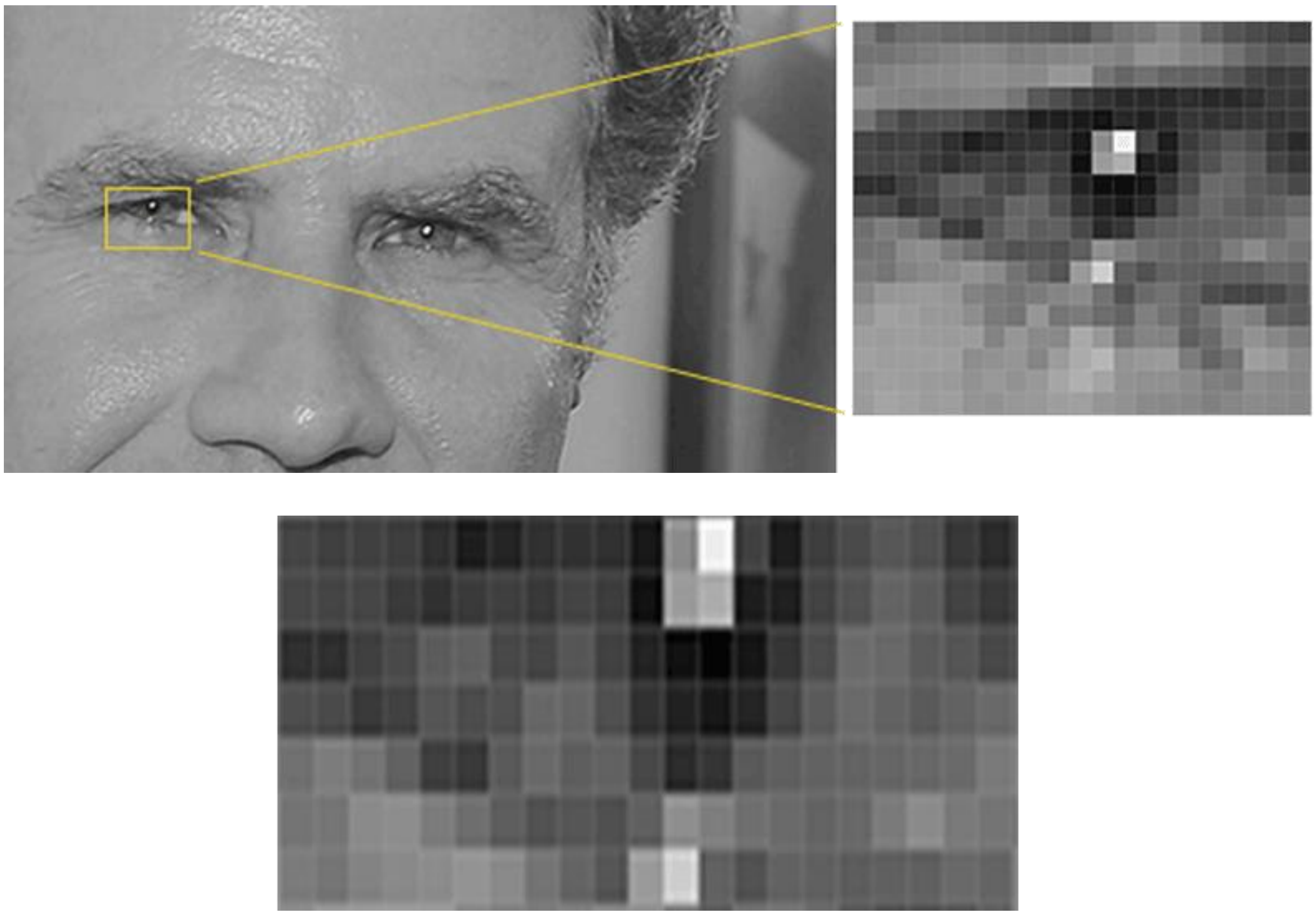


Fig 5.3.3

The idea of program is to figure out how dark the current pixel is compared to the pixels directly girding it. Also it needs to draw an arrow showing in which direction the image is getting darker

Still, you end up with every pixel being replaced by an arrow, If it repeats that process for every single pixel in the image. These arrows are called slants and they show the inflow from light to dark across the entire image

Fig 5.3.4

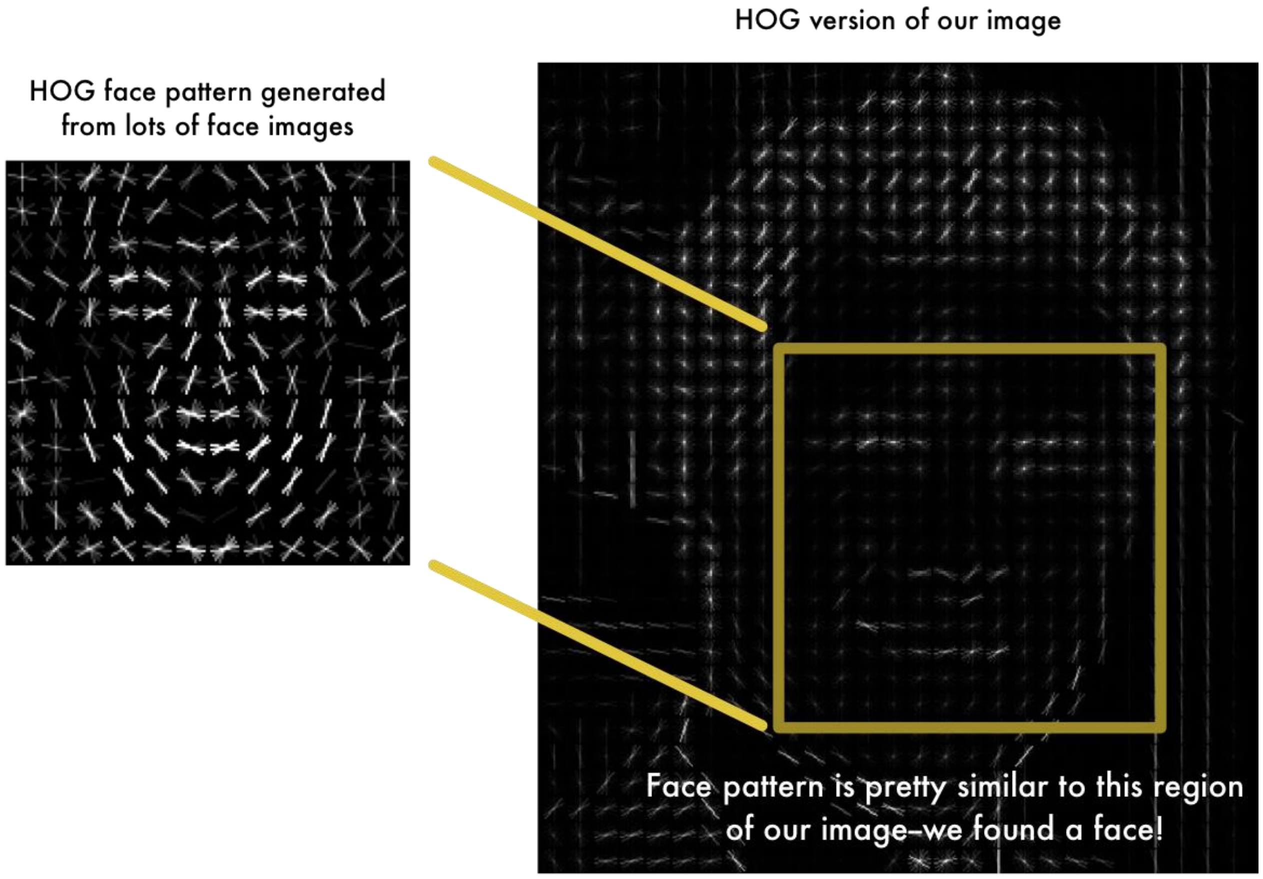
This might feel like a arbitrary thing to do, but there’s a really good reason for replacing the pixels with gradients. However, really dark images and really light images of the same person will have completely different pixel values, If it assay pixels directly. But by only considering the direction that brilliance changes, both really dark images and really bright images will end up with the same exact representation. That makes the problem a lot easier to break!

To do this, It’ll break up the image into small places of 16x16 pixels each. In each forecourt, It’ll count up how numerous slants point in each major direction. Also, it’ll replace that forecourt in the image with the arrow directions that were the strongest.

The end result is it turns the original image into a veritably simple representation that captures the introductory structure of a face in a simple way

Fig 5.3.5

To find faces in this HOG image, all the program have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces:


HOG version of our image
HOG face pattern generated from lots of face images
Face pattern is pretty similar to this region of our image–we found a face!

5.3.6

Using this technique, the program can now easily find faces in any image:

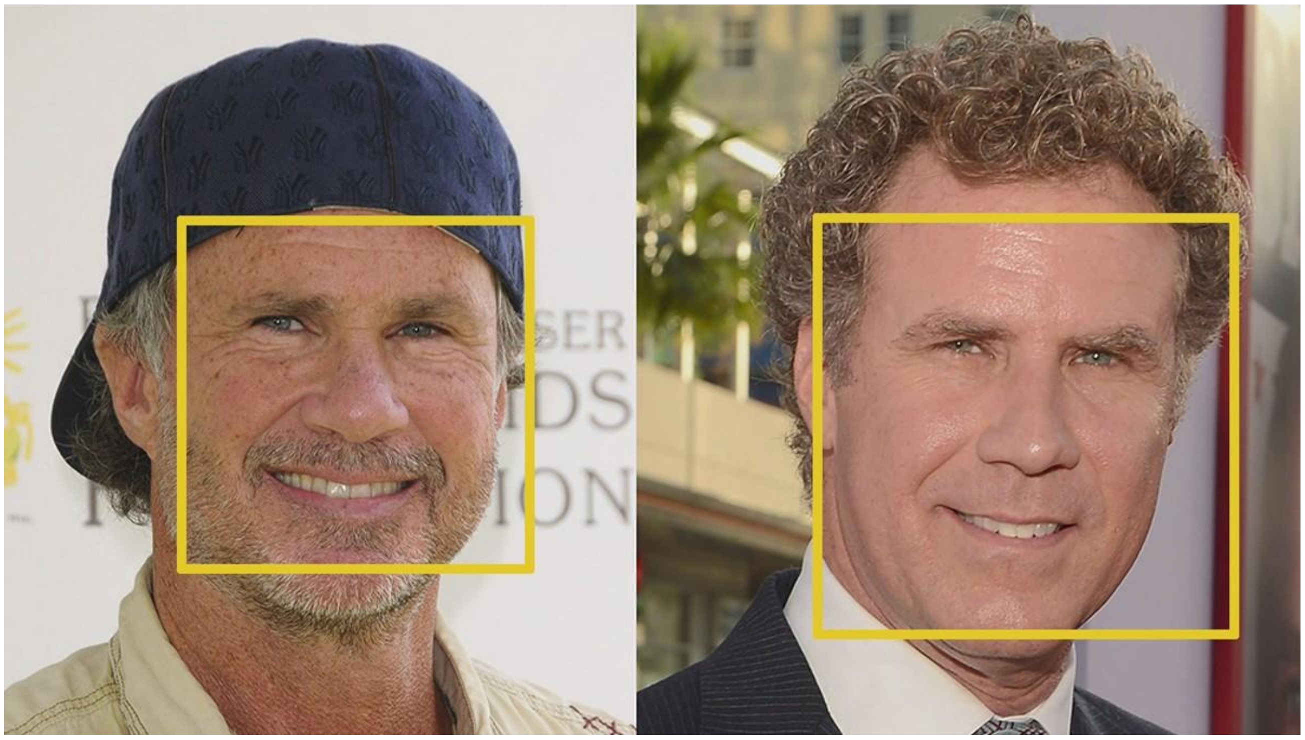


Fig 5.3.7

Step 2: Posing and Projecting Faces

- It isolated the faces in our image. But now it'll have to deal with the problem that faces turned different directions look totally different to a computer:

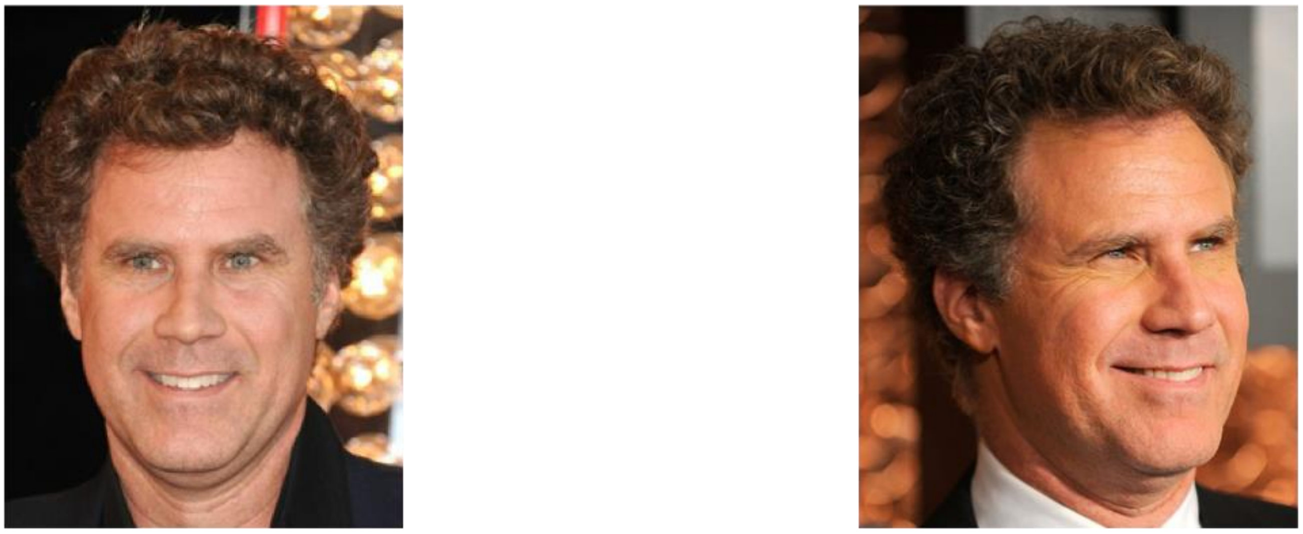


Fig 5.3.8

- Humans can fluently fete that both images are of Will Ferrell, but computers would see these filmland as two fully different people.
- To regard for this, the program will try to underpinning each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for the program to compare faces in the coming way.
- To do this, it's going to use an algorithm called face corner estimation. There are lots of ways to do this, but it's going to use the approach constructed in 2014 by Vahid Kazemi and Josephine Sullivan.
- The introductory idea is that it will come up with 68 specific points (called milestones) that live on every face-the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Also it will train a machine learning algorithm to be suitable to find these 68 specific points on any face:

Here's the result of locating the 68 face landmarks on our test image:

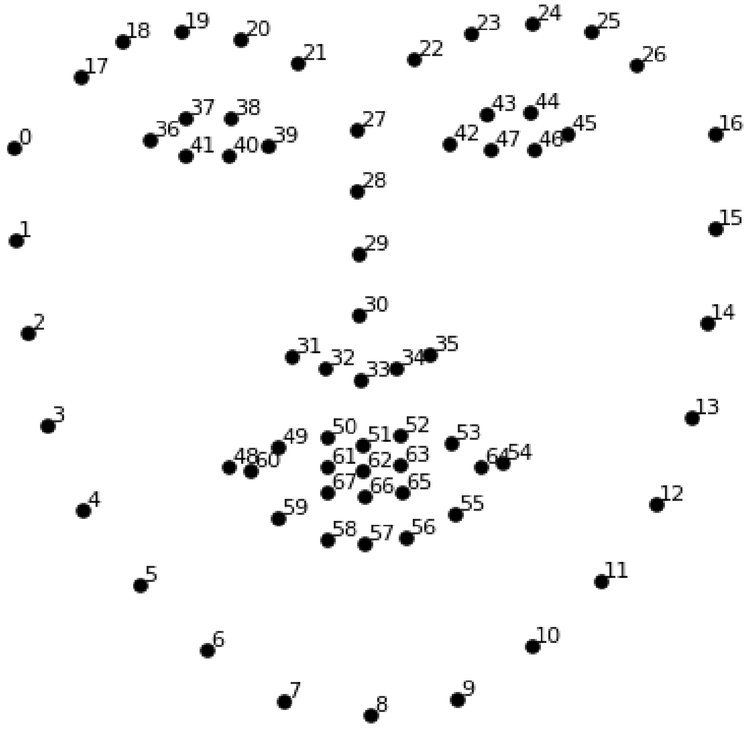


Fig 5.3.9

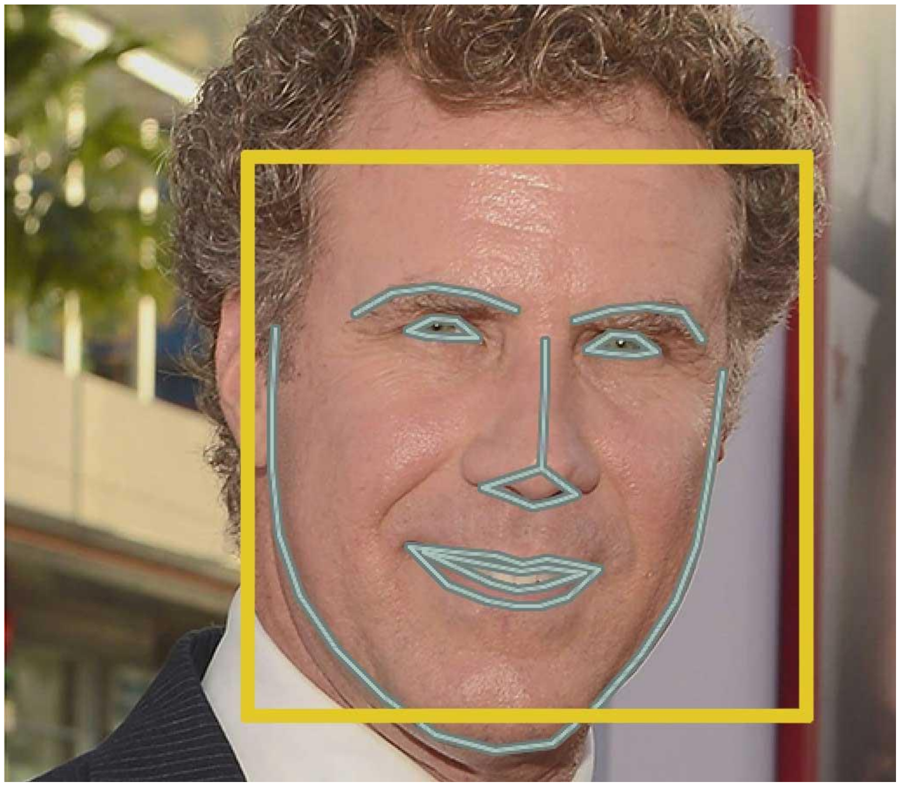


Fig 5.3.10

The 68 landmarks the will locate on every face. This image was created by Brandon Amos of CMU who works on OpenFace.

Since the program knows where the eyes and mouth are, it'll simply rotate, gauge and shear the image so that the eyes and mouth are centred as stylish as possible. The program won't do any fancy 3d foundations because that would introduce deformations into the image.

It's only going to use introductory image metamorphoses like gyration and scale that save resemblant lines (called affine metamorphoses).

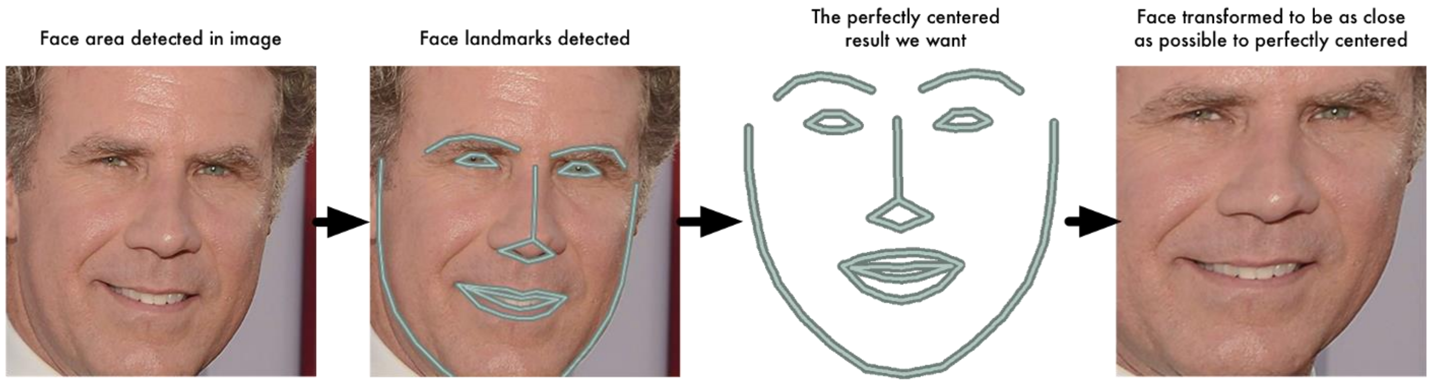

Face area detected in image
Face landmarks detected
The perfectly centered result we want
Face transformed to be as close as possible to perfectly centered

fig 5.3.11

Now no matter how the face is turned, the program is able to centre the eyes and mouth are in roughly the same position in the image. This will make the next step a lot more accurate.

Step 3: Encoding Faces

The simplest approach to face recognition is to directly compare the unknown face the program plant in Step 2 with all the filmland of people that have formerly been tagged. When found a preliminarily tagged face that looks veritably analogous to our unknown face, it must be the same person. But, There's a problem with that. A point like Facebook with billions of druggies and a trillion prints can't conceivably circle through every former-tagged face to compare it to every recently uploaded picture. That would take way too long. They need to be suitable to fete faces in milliseconds, not hours.

What is needed is a way to prize a many introductory measures from each face. Also, it could measure our unknown face the same way and find the given face with the closest measures.

The most reliable way to measure a face:

It turns out that the measures that feel egregious to humans (like eye colour) don't really make sense to a computer looking at individual pixels in an image. Experimenters have discovered that the most accurate approach is to let the computer figure out the measures

to collect itself. Deep literacy does a better job than humans at figuring out which corridor of a face are important to measure.
The result is to train a Deep Convolutional Neural Network (just like what’s done in Part But rather of training the network to fete filmland objects like it’s done last time, it’s going to train it to induce 128 measures for each face.

The training process works by looking at 3 face images at a time:

1. Cargo a training face image of a given person
2. Cargo another picture of the same given person
3. Cargo a picture of a completely different person

Also, the algorithm looks at the measures it's presently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measures it generates for# 1 and# 2 are slightly near while making sure the measures for# 2 and# 3 are slightly farther piecemeal
After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably induce 128 measures for each person. Any ten different filmland of the same person should give roughly the same measures.

Machine learning people call the 128 measures of each face an embedding. The idea of reducing complicated raw data like a picture into a list of computer-generated figures comes up a lot in machine literacy (especially in language restatement). The exact approach for faces the program using was constructed in 2015 by experimenters at Google but numerous analogous approaches live.

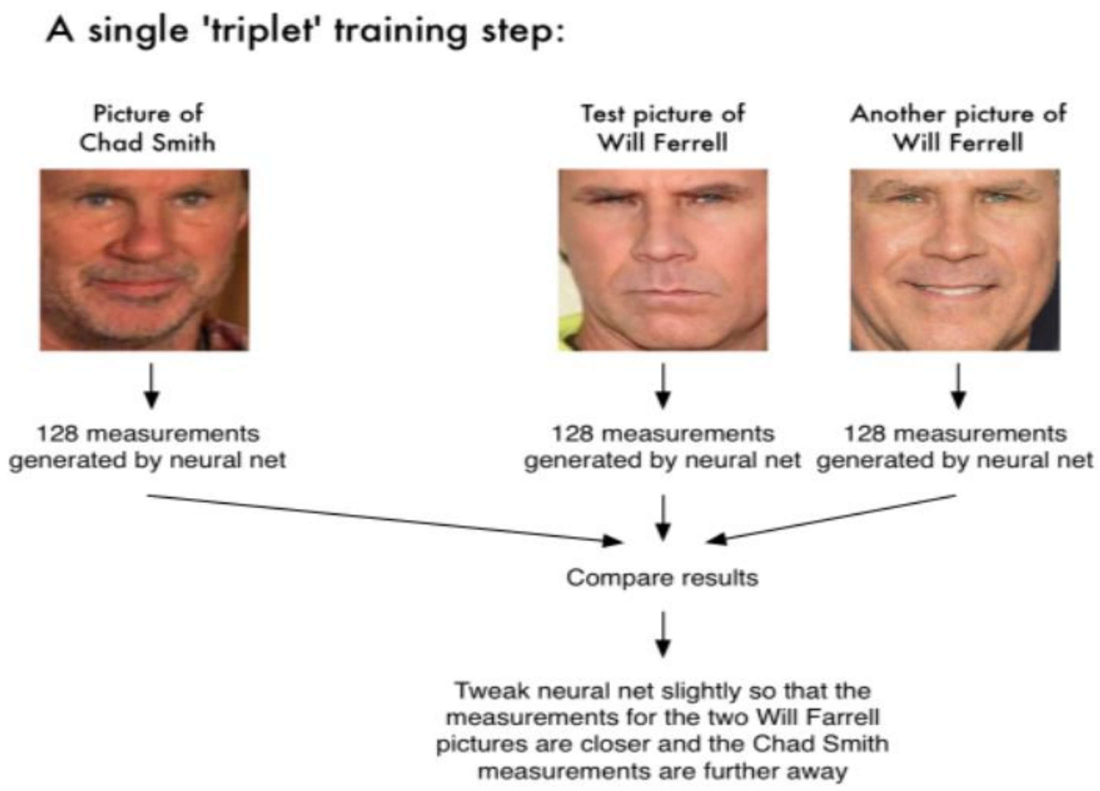
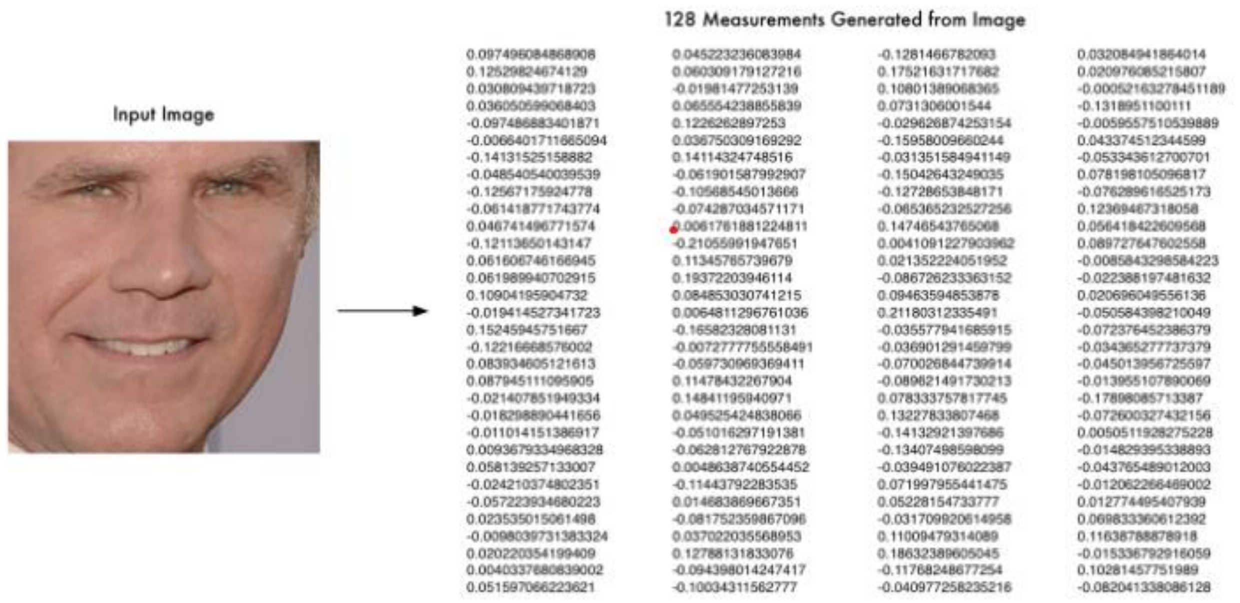

A single 'triplet' training step:
Picture of Chad Smith
Test picture of Will Ferrell
Another picture of Will Ferrell
128 measurements generated by neural net
128 measurements generated by neural net
128 measurements generated by neural net
Compare results
Tweak neural net slightly so that the measurements for the two Will Farrell pictures are closer and the Chad Smith measurements are further away

Fig 5.3.12

Encoding our face image:

- This process of training a convolutional neural network to affair face embeddings requires a lot of data and computer power. Indeed with an precious NVidia Tesla videotape card, it takes about 24 hours of nonstop training to get good delicacy.
- But once the network has been trained, it can induce measures for any face, So this step only needs to be done formerly.
- So all that need to be done is run our face images through their pre-trained network to get the 128 measures for each face. Then the measures for our test image



0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334968328	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

Fig 5.3.13

Step 4: Finding the person's name from the encoding:

- This last step is actually the easiest step in the whole process. All that have to be done is find the person in our database of known people who has the closest measurements to our test image.
- It's done using any basic machine learning classification algorithm. No fancy deep learning tricks are needed. It'll use a simple linear SVM classifier, but lots of classification algorithms could work.
- training a classifier that can take in the measurements from a new test image and tells which known person is the closest match should be done. Running this classifier takes milliseconds. The result of the classifier is the name of the person!

5.4.1 YOLOV4

You only look once (YOLO) is a family of one-stage object detectors that are fast and accurate.

- Utmost of the ultramodern accurate models bear numerous GPUs for training with a large mini-batch size, and doing this with one GPU makes the training really slow and impracticable.
- YOLO v4 addresses this issue by making an object sensor which can be trained on a single GPU with a lower mini-batch size.
- This makes it possible to train a super-fast and accurate object sensor with a single 1080 Ti or 2080 Ti GPU.
- YOLO v4 achieves state-of-the- art results at a real time speed on the MS COCO dataset with43.5 AP running at 65 FPS on a Tesla V100. Enough intriguing results!
- To achieve these results, they combine some features similar as Weighted-Residual- Connections (WRC),Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Tone-inimical- training (SAT) and Mish-activation, Mosaic data addition, DropBlock regularization, and CIoUlos

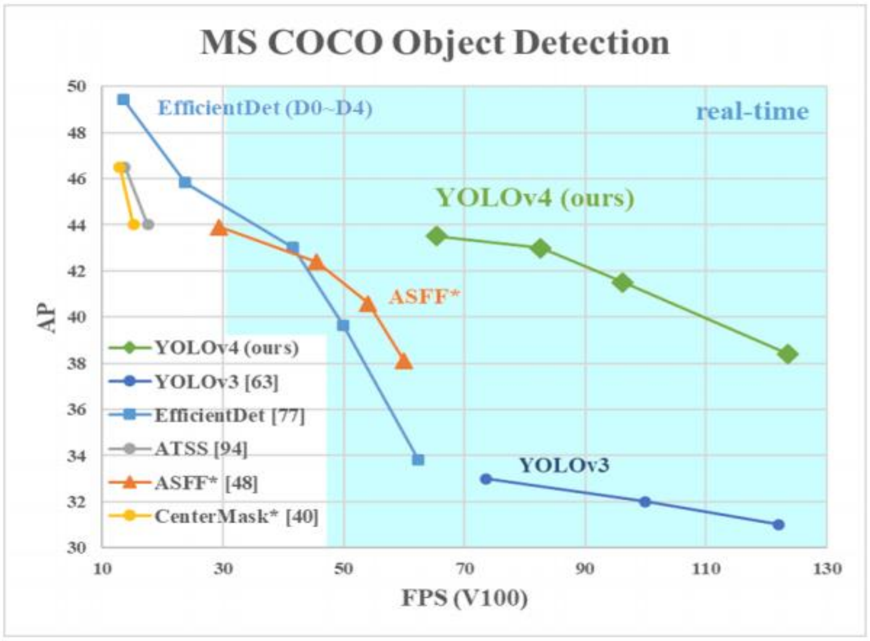


Fig 5.4.1

These are appertained to as universal features because they should work well singly from the computer vision tasks, datasets and models..

General Architecture of an Object Detector

Although YOLO are one- stage sensors, there are also two- stage sensors like R-CNN, fast R-CNN and faster R-CNN which are accurate but slow.

Backbone

- Models Similar as ResNet, DenseNet, VGG, etc, are used as point extractors. They're pre-trained on image bracket datasets, like ImageNet, and also OK-tuned on the discovery dataset.
- Turns out that, these networks that produce different situations of features with advanced semantics as the network gets deeper (further layers), are useful for ultimate corridor of the object discovery network.

Neck

- These are redundant layers that go in between the backbone and head. They're used to prize different point charts of different stages of the backbone.
- The neck part can be for illustration a FPN (1), PANet (2),Bi-FPN (3)
- YOLOv3 uses FPN to prize features of different scales from the backbone.

Feature Pyramid Network (FPN)

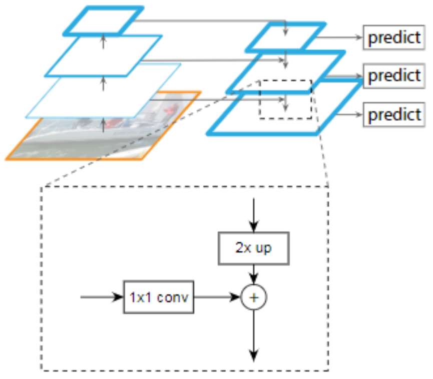


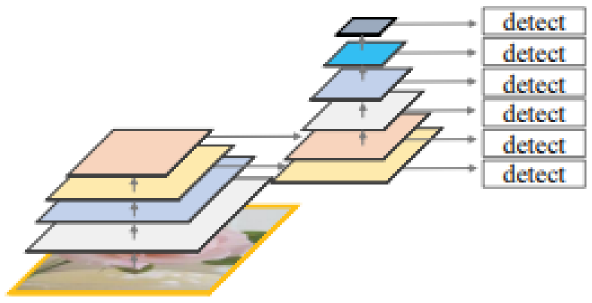
Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

Fig 5.4.2

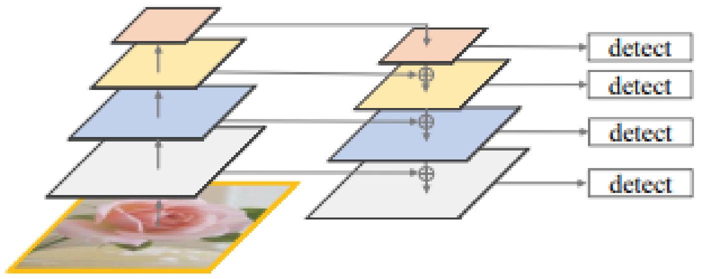
- Augments a standard convolutional network with a top-down pathway and side connections so the network efficiently constructs
- a rich, multi-scale point aggregate from a single resolution input image
  Each side connection merges the point maps from the bottom-up pathway to the top-down pathway, producing different aggregate situations.

Four types of Pyramids:

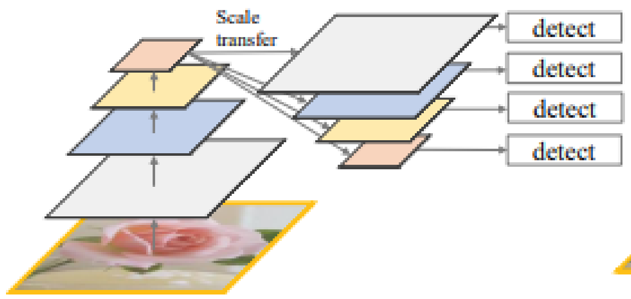
- Point Aggregate Networks can be applied to different chines models, and as an illustration, the original FPN (1) paper used ResNets. There are also numerous modules that integrate FPN in different ways, similar as SFAM (7), ASFF (9), and Bi-FPN (3). The image shows how features are uprooted from the backbone in a Single Shot Sensor armature (SSD). The image over shows also three other different types of aggregate networks



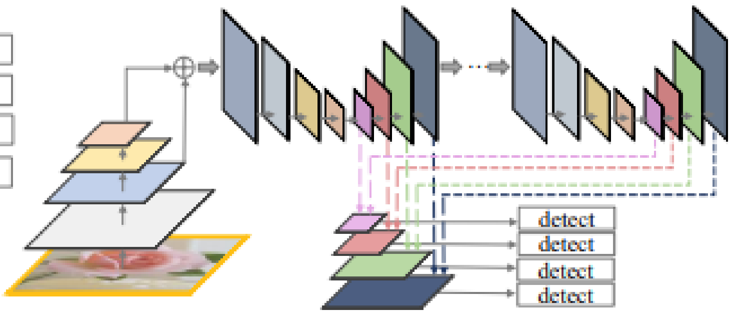
(a) SSD-style feature pyramid



(b) FPN-style feature pyramid



(c) STDN-style feature pyramid



(d) Our multi-level feature pyramid

Fig 5.4.3

Head

- This is a network in charge of actually doing the discovery part (bracket and retrogression) of bounding boxes. A single affair may look like (depending on the perpetration) 4 values describing the prognosticated bounding box (x, y, h, w) and the probability of k classes 1 (one redundant for background). Expostulated sensors anchor- grounded, like YOLO.

Bag of freebies (BoF)

- Styles that can make the object sensor admit better delicacy without adding the conclusion cost.
- An illustration of BoF is data addition, which increases the conception capability of the model. To do this the program can do print-metric deformations like changing the brilliance, achromatism, discrepancy and noise or it can do geometric deformation of an image, like rotating it, cropping, etc.
- The cost function of the retrogression network also applies to the order. The traditional thing is to apply Mean Squared error to perform retrogression on the equals.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

YOLO V4 design

- **Backbone**: It uses the CSPDarknet53 as the point-extractor model for the GPU interpretation. For the VPU (Vision Processing Unit) they consider using EfficientNet-nonfat-MixNet-GhostNet or MobileNetV3. Look at the GPU interpretation for now.

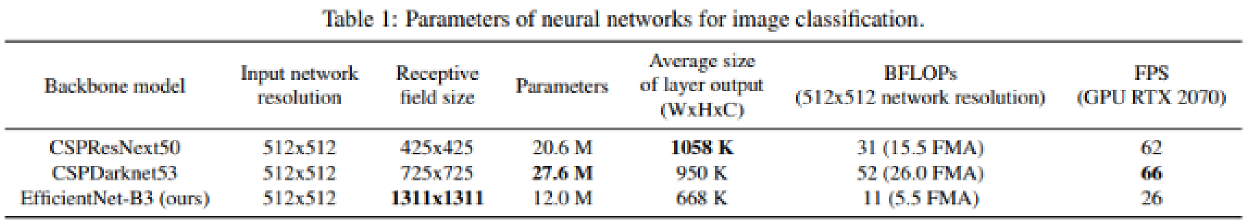


Table 1: Parameters of neural networks for image classification.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	31 (15.5 FMA)	62
CSPDarknet53	512x512	725x725	**27.6 M**	950 K	52 (26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	11 (5.5 FMA)	26

Fig 5.4.4

- Certain Chines are more suitable for bracket than for discovery. For illustration, CSPDarknet53 showed to be better than CSPResNext50 in terms of detecting objects, and CSPResNext50 better than CSPDarknet53 for image bracket.
- **Neck**: They use Spatial aggregate pooling (SPP) and Path Aggregation Network (Visage). The ultimate isn't identical to the original Visage, but a modified interpretation which replaces the addition with a concatenation.
- **Head**: They use the same as YOLO v3.
- These are the heads applied at different scales of the network, for detecting different-size objects. The number of channels is 255 because of (80 *classes* + 1 *for objectness* + 4 *coordinates*) * 3 *anchors.*

Example of YOLO V4

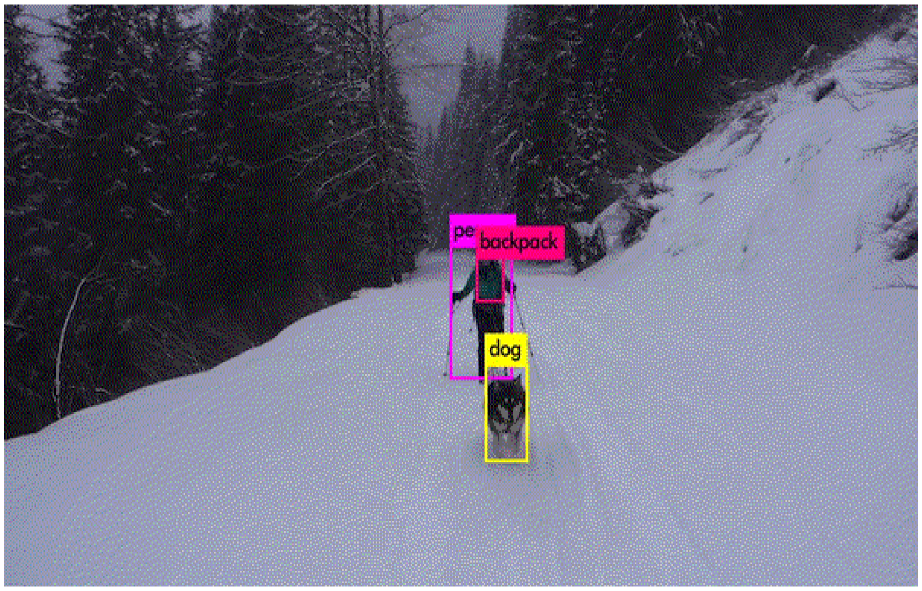


Fig 5.4..5

- YOLO v4 achieves state-of-the- art results (43.5 AP) for real- time object discovery and is suitable to run at a speed of 65 FPS on a V100 GPU. If you want lower delicacy but much advanced FPS.

5.4 MODULE DESIGN SPECIFICATION

IMPLEMENTATION MODULES

- **OpenCV-python**
- **Pytesseract**
- **NumPy**
- **Datetime**
- **Face_recognition**
- **Os**
- **Glob**

These are the modules used for the implementaion of the cctv monitioring system.

OpenCV-python:

- OpenCV is a huge open- source library for computer vision, machine literacy, and image processing. OpenCV supports a wide variety of programming languages like Python, C, Java, etc. It can reuse images and vids to identify objects, faces, or indeed the handwriting of a mortal.
- When it's integrated with colorful libraries, similar as NumPy which is a largely optimized library for numerical operations, also the number of munitions increases in your Arsenal i.e. whatever operations one can do in NumPy can be combined with OpenCV.
- Opencv is used it in this design to represent the places where faces can be plant in the camera frame.

Pytesseract:

- Pytesseract or Python-tesseract is an Optical Character Recognition (OCR) tool for Python. It will read and recognize the text in images, license plates etc. Python-tesseract is actually a wrapper class or a package for Google's Tesseract-OCR Engine.
- It is also useful and regarded as a stand-alone invocation script to tesseract, as it can easily read all image types supported by the Pillow and Leptonica imaging libraries, which mainly includes – jpg, png, gif, bmp, tiff etc. Also additionally, if it is used as a script, Python-tesseract will also print the recognized text instead of writing it to a file. And it uses this module to record text form image and write it down in the document in digital form.

NumPy:

- In Python lists that serve the purpose of arrays are found, but they're slow to reuse. NumPy aims to give an array object that's over to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray veritably easy. Arrays are veritably constantly used in data wisdom, where speed and coffers are veritably important.

Datetime:

- The datetime module inventories classes for manipulating dates and times. While date and time computation is supported, the focus of the perpetration is on effective trait birth for affair formatting and manipulation.
- A date in Python isn't a data type of its own, but the program import's a module named datetime to work with dates as date objects.

Face_recognition:

- Fete and manipulate faces from Python or from the command line with the world's simplest face recognition library. Erected using dlib's state-of-the-art face recognition erected with deep literacy. The model has an delicacy of 99.38 on the Labeled Faces in the Wild standard. This also provides a simpleface_recognition command line tool that lets the program to do face recognition on a brochure of images from the command line.

Os:

- The Os module in Python provides functions for creating and removing a directory (brochure), costing its contents, changing and relating the current directory, etc. You first need to import the OS module to interact with the underpinning operating system.

Glob:

- In Python, the glob module is used to recoup lines/ pathnames matching a specified pattern. It will follow the pattern of standard path unix rules expansion.   It's also predicted that according to marks it's faster than other styles to match pathnames in directories. It uses as a wildcards ("*,?, (ranges)) incremental from exact string quest to make path recovery more simple and accessible.Implementation for facial recognition.

CHAPTER 6

6 SYSTEM IMPLEMENTATION

- This program is created and developed in Python Programming Language. There are several pre-defined modules that have been imported and made use of in the process. For getting the input the stock laptop camera and pre-recorded videos for certain instances are used.

- The modules used are listed and their uses have been explained in the upcoming slides along with code snippets of implementation.

A. Implementation for facial recognition

```
while True:
    ret, frame = cap.read()
    #to show time and date
    if ret == True:
        font = cv2.FONT_HERSHEY_SIMPLEX
        text = 'Width: ' + str(cap.get(3)) + ' Height:' + str(cap.get(4))
datet = str(datetime.datetime.now())
        """frame = cv2.putText(frame, text, (10, 100), font, 1,
                            (0, 255, 255), 2, cv2.LINE_AA)"""
        frame = cv2.putText(frame, datet, (10, 25), font, 0.5,
                            (0, 255, 255), 2, cv2.LINE_AA)
    # Detect Faces
face_locations, face_names, face_times = sfr.detect_known_faces(frame)
    for face_loc, name, times in zip(face_locations, face_names, face_times):
        y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
 cv2.putText(frame, name,(x1, y1 - 10),
    cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
            #cv2.putText(frame, times, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX,
    1, (0, 0, 200), 2)
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)
            print(times)
     cv2.imshow("Frame", frame)
```

In this module the program uses camera to read the faces of people who enters into the frame. If a person enters to video frame it will start to analyse the entered person face and if that person's is already encoded (known to us) it show the name of that person and also shows us the accurate time of entry and stay of that particular person and also gives the date in the video feed itself. The name of the person will be displayed on top of a rectangular box shaped frame. Unknown is mentioned if an unknown person enters into the frame it shows as unknown with corresponding date and time.

B. Detecting people in the cam frame

```
def detect(frame):
  bounding_box_cordinates, weights = HOGCV.detectMultiScale(frame,
  winStride=(4, 4), padding=(8, 8), scale=1.03)
    person = 1
    for x, y, w, h in bounding_box_cordinates:
      cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
      cv2.putText(frame, f'person {person}', (x, y),
  cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
      person += 1

   cv2.putText(frame, 'Status : Detecting ', (40, 40),
  cv2.FONT_HERSHEY_DUPLEX, 0.8, (255, 0, 0), 2)
    cv2.putText(frame, f'Total Persons : {person - 1}', (40, 70),
  cv2.FONT_HERSHEY_DUPLEX, 0.8, (255, 0, 0), 2)
    cv2.imshow('output', frame)
    return frame
  detect(frame)
```

In this module the program is going to detect the person in the video frame by an bounding boxes coordinates using a method called as HOGCV detect Multiscale which means, first it will convert the image into black and white then look at each and every single pixels of the image and compare how the dark pixel compared with others by using the HOG pattern it can easily identify a persons face and encode it for future identification.

In any image or a video frame if two person faces are with different viewing angles can be easily identified by humans but computer cannot find it so, it can be done successfully with the help of face landmark estimation algorithm along with 68 faces landmark technique, here eyes, nose and mouth angles are straightened as much as possible so that it can easily identify whether the current person visible in the frame is already known(previously encoded) if encoded already then the person’s name is returned.

C. Detecting number plates and identifying the state where the vehicle registration belongs

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program
Files\Tesseract-OCR\tesseract.exe'
cascade= cv2.CascadeClassifier("haarcascade_russian_plate_number.xml")
states={"AN":"Andaman and Nicobar","AP":"AndhraPradesh","AR":"Arunachal
Pradesh","AS":"Assam","BR":"Bihar","CH":"Chandigarh","DN":"Dadra and
Nagar Haveli","DD":"Daman and
Diu","DL":"Delhi","GA":"Goa","GJ":"Gujarat",
"HR":"Haryana","HP":"HimachalPradesh","JK":"Jammu and
Kashmir","KA":"Karnataka","KL":"Kerala","LD":"Lakshadweep","MP":"Madh
ya
Pradesh","MH":"Maharashtra","MN":"Manipur","ML":"Meghalaya","MZ":"Miz
oram","NL":"Nagaland","OD":"Odissa","PY":"Pondicherry","PN":"Punjab","RJ
":"Rajasthan","SK":"Sikkim","TN":"TamilNadu","TR":"Tripura","UP":"Uttar
Pradesh", "WB":"West
Bengal","CG":"Chhattisgarh","TS":"Telangana","JH":"Jharkhand","UK":"Uttara
khand"}
```

Here pytesseract module is used to identify every alphabet given in a picture. And the module CascadeClassifier is used for identifying the words and numbers and give the exact details in the number plate. In our module it specifies all the states of India to the system and when the system finds any cars number plate in the frame it tries to get the details from the number plate and identify which state it belongs to and then its stored in the database for further reference.

The number plates varieties(languages and fonts) are very huge in number so that exact text extraction is not possible so, it also uses this opportunity to crop the image of the plate for recovery.

D.Extracting the text from the number plates of vehicles

```
def extract_num(img_name):
img = cv2.imread(img_name) ## Reading Image
   # Converting into Gray
  gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  # Detecting plate
nplate = cascade.detectMultiScale(gray,1.1,4)
  for (x,y,w,h) in nplate:
    # Crop a portion of plate
a,b = (int(0.02*img.shape[0]), int(0.025*img.shape[1]))
    plate = img[y+a:y+h-a, x+b:x+w-b, :]
    # make image more darker to identify the LPR
```

```
        ## iMAGE PROCESSING
        kernel = np.ones((1, 1), np.uint8)
        plate = cv2.dilate(plate, kernel, iterations=1)
        plate = cv2.erode(plate, kernel, iterations=1)
plate_gray = cv2.cvtColor(plate,cv2.COLOR_BGR2GRAY)
        (thresh, plate) = cv2.threshold(plate_gray, 127, 255,
cv2.THRESH_BINARY)
# Feed Image to OCR engine
        read = pytesseract.image_to_string(plate)
        read = ''.join(e for e in read if e.isalnum())
        print(read)
        stat = read[0:2]
        try:
        # Fetch the State information
print('Car Belongs to',states[stat])
        except:
print('State not recognised!!')
        print(read)
        cv2.rectangle(img, (x,y), (x+w, y+h), (51,51,255), 2)
        cv2.rectangle(img, (x, y - 40), (x + w, y),(51,51,255) , -1)
        cv2.putText(img,read, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(255, 255, 255), 2)
        cv2.imshow('PLate',plate)
        # Save & display result image
        cv2.imwrite('plate.jpg', plate)


  cv2.imshow("Result", img)
```

```
    cv2.imwrite('result.jpg',img)
extract_num(image.jpg)
```

Here in this module, the program is extracting the image of the vehicle from the frame. The image is then made to turn into grey scale mode. Then using Cascade command, it is used to detect the number plate from the image. Then the part of the number plate is cropped from the picture and shown is the shape of rectangle. The cropped image of the plate stores the number and state of the vehicle registration in a document.

Then using the pytesseract module the program gets every alphabet from the number plate and it’s displayed over the rectangle box over the cropped image of the number plate. Then from the number plates our system tries to identify the state which the car belongs to and store it in the database. Then image is stored in the database using datalogging along with the date and time it was seen in the frame and the image is displayed within a rectangular box where the number plate is shown and also the information in the number plate is also displayed.

When the same vehicle is again seen int frame our system identifies from the previous data in the database and displays to the user.

E. SAVED VIDEO

```
def set_saved_video(input_video, output_video, size):
fourcc = cv2.VideoWriter_fourcc(*"MJPG")
    fps = int(input_video.get(cv2.CAP_PROP_FPS))
    video = cv2.VideoWriter(output_video, fourcc, fps, size)
    return video
```

In this function the program record the video feed from the cctv camera and the recorded video is saved in the form of MJPG format it is used to find the person who is known and unknown to us and can be seen whenever there’s need and for the future references.

F. VIDEO CAPTURE

```
def video_capture(frame_queue, darknet_image_queue):
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:


            break
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
frame_resized = cv2.resize(frame_rgb, (darknet_width, darknet_height),
                           interpolation=cv2.INTER_LINEAR)
frame_queue.put(frame)

img_for_detect = darknet.make_image(darknet_width, darknet_height, 3)
darknet.copy_image_from_bytes(img_for_detect, frame_resized.tobytes())
darknet_image_queue.put(img_for_detect)
cap.release()
```

In this function the cctv camera recorded video will be capture and frames are spilted for an different uses are the spilted frames are combined atlast for the output result using a function called darknet(Darknet is an open source neural network framework which is written in both C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.)

G. INFERENCE

```
def inference(darknet_image_queue, detections_queue, fps_queue):
    while cap.isOpened():
```

```
    darknet_image = darknet_image_queue.get()
    prev_time = time.time()
        detections = darknet.detect_image(network, class_names, darknet_image,
    thresh=args.thresh)
    detections_queue.put(detections)
        fps = int(1/(time.time() - prev_time))
    fps_queue.put(fps)
    print("FPS: {}".format(fps))
    darknet.print_detections(detections, args.ext_output)
    darknet.free_image(darknet_image)
    cap.release()
```

In this function the resultant output will be comes under the frame of inference where the persons and objects are detected and stores the data in an data logging for easy retrievel and for the further use.

H. DRAWING

```
def drawing(frame_queue, detections_queue, fps_queue):
random.seed(3)  # deterministic bboxcolors
    video = set_saved_video(cap, args.out_filename, (video_width, video_height))
    count=i=0
    while cap.isOpened():
        frame = frame_queue.get()
        detections = detections_queue.get()
        fps = fps_queue.get()
detections_adjusted = []
```

```
extract_num(frame)
    if frame is not None:
        for label, confidence, bbox in detections:
bbox_adjusted = convert2original(frame, bbox)
            if str(label)=='person'or str(label)=='car'or str(label)=='bicycle'or
str(label)=='bus'or str(label)=='truck'or str(label)=='motorbike' or str(label)=='sports
ball':
detections_adjusted.append((str(label), confidence, bbox_adjusted))
        #print(bbox_adjusted)
        image = darknet.draw_boxes(detections_adjusted, frame, class_colors)
        #ret, frame1 = cap.read()
        font = cv2.FONT_HERSHEY_SIMPLEX
        text = 'Width: ' + str(cap.get(3)) + ' Height:' + str(cap.get(4))
datet = str(datetime.datetime.now())
        """frame = cv2.putText(frame, text, (10, 100), font, 1,
                        (0, 255, 255), 2, cv2.LINE_AA)"""
        frame = cv2.putText(frame, datet, (10, 25), font, 0.5,
                        (0, 255, 255), 2, cv2.LINE_AA)
```

In this function it will draw the bbox in the recorded video frame and shows whether it is an person or the objects and writes the name of the person who enters the frame or shows unknown and it is also detects the vehicles as well as motorcycles, football and mark the time at data logging and saves the file with an extension of csv format.

Python code for the program:
Darknet.py:

```
#!/usr/bin/env python3

"""
Python 3 wrapper for identifying objects in images

Running the script requires opencv-python to be installed (`pip install opencv-
python`)
Directly viewing or returning bounding-boxed images requires scikit-image to be
installed (`pip install scikit-image`)
Use pip3 instead of pip on some systems to be sure to install modules for python3
"""

from ctypes import *
import math
import random
import os
import numpy as np


class BOX(Structure):
    _fields_ = [("x", c_float),
                ("y", c_float),
                ("w", c_float),
                ("h", c_float)]


class DETECTION(Structure):
    _fields_ = [("bbox", BOX),
                ("classes", c_int),
                ("best_class_idx", c_int),
                ("prob", POINTER(c_float)),
                ("mask", POINTER(c_float)),
                ("objectness", c_float),
                ("sort_class", c_int),
                ("uc", POINTER(c_float)),
                ("points", c_int),
                ("embeddings", POINTER(c_float)),
                ("embedding_size", c_int),
                ("sim", c_float),
                ("track_id", c_int)]

class DETNUMPAIR(Structure):
    _fields_ = [("num", c_int),
                ("dets", POINTER(DETECTION))]


class IMAGE(Structure):
    _fields_ = [("w", c_int),
                ("h", c_int),
                ("c", c_int),
                ("data", POINTER(c_float))]


class METADATA(Structure):
    _fields_ = [("classes", c_int),
                ("names", POINTER(c_char_p))]


def network_width(net):
    return lib.network_width(net)


def network_height(net):
    return lib.network_height(net)


def bbox2points(bbox):
    """
    From bounding box yolo format
    to corner points cv2 rectangle
    """
    x, y, w, h = bbox
    xmin = int(round(x - (w / 2)))
    xmax = int(round(x + (w / 2)))
    ymin = int(round(y - (h / 2)))
    ymax = int(round(y + (h / 2)))
    return xmin, ymin, xmax, ymax


def class_colors(names):
    """
    Create a dict with one random BGR color for each
    class name
    """
    return {name: (
        random.randint(0, 255),
        random.randint(0, 255),
        random.randint(0, 255)) for name in names}


def load_network(config_file, data_file, weights, batch_size=1):
    """
    load model description and weights from config files
    args:
        config_file (str): path to .cfg model file
        data_file (str): path to .data model file
        weights (str): path to weights
    returns:
        network: trained model
        class_names
        class_colors
    """
    network = load_net_custom(
        config_file.encode("ascii"),
        weights.encode("ascii"), 0, batch_size)
    metadata = load_meta(data_file.encode("ascii"))
    class_names = [metadata.names[i].decode("ascii") for i in range(metadata.classes)]
    colors = class_colors(class_names)
    return network, class_names, colors


def print_detections(detections, coordinates=False):
    print("\nObjects:")
    for label, confidence, bbox in detections:
        x, y, w, h = bbox
        if coordinates:
            print("{}: {}%    (left_x: {:.0f}   top_y:  {:.0f}   width:   {:.0f}   height:
{:.0f})".format(label, confidence, x, y, w, h))
        else:
            print("{}: {}%".format(label, confidence))
```

```
def draw_boxes(detections, image, colors):
    import cv2
    for label, confidence, bbox in detections:
        left, top, right, bottom = bbox2points(bbox)
        cv2.rectangle(image, (left, top), (right, bottom), colors[label], 1)
        cv2.putText(image, "{} [{:.2f}]".format(label, float(confidence)),
                    (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                    colors[label], 2)
    return image


def decode_detection(detections):
    decoded = []
    for label, confidence, bbox in detections:
        confidence = str(round(confidence * 100, 2))
        decoded.append((str(label), confidence, bbox))
    return decoded

# https://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-
python/
# Malisiewicz et al.
def non_max_suppression_fast(detections, overlap_thresh):
    boxes = []
    for detection in detections:
        _, _, _, (x, y, w, h) = detection
        x1 = x - w / 2
        y1 = y - h / 2
        x2 = x + w / 2
        y2 = y + h / 2
        boxes.append(np.array([x1, y1, x2, y2]))
    boxes_array = np.array(boxes)

    # initialize the list of picked indexes
    pick = []
    # grab the coordinates of the bounding boxes
    x1 = boxes_array[:, 0]
    y1 = boxes_array[:, 1]
    x2 = boxes_array[:, 2]
    y2 = boxes_array[:, 3]
    # compute the area of the bounding boxes and sort the bounding
    # boxes by the bottom-right y-coordinate of the bounding box
    area = (x2 - x1 + 1) * (y2 - y1 + 1)
    idxs = np.argsort(y2)
    # keep looping while some indexes still remain in the indexes
    # list
    while len(idxs) > 0:
        # grab the last index in the indexes list and add the
        # index value to the list of picked indexes
        last = len(idxs) - 1
        i = idxs[last]
        pick.append(i)
        # find the largest (x, y) coordinates for the start of
        # the bounding box and the smallest (x, y) coordinates
        # for the end of the bounding box
        xx1 = np.maximum(x1[i], x1[idxs[:last]])
        yy1 = np.maximum(y1[i], y1[idxs[:last]])
        xx2 = np.minimum(x2[i], x2[idxs[:last]])
        yy2 = np.minimum(y2[i], y2[idxs[:last]])
        # compute the width and height of the bounding box
        w = np.maximum(0, xx2 - xx1 + 1)
        h = np.maximum(0, yy2 - yy1 + 1)
        # compute the ratio of overlap
        overlap = (w * h) / area[idxs[:last]]
        # delete all indexes from the index list that have
        idxs = np.delete(idxs, np.concatenate(([last],
                                               np.where(overlap > overlap_thresh)[0])))
        # return only the bounding boxes that were picked using the
        # integer data type
    return [detections[i] for i in pick]

def remove_negatives(detections, class_names, num):
    """
    Remove all classes with 0% confidence within the detection
    """
    predictions = []
    for j in range(num):
        for idx, name in enumerate(class_names):
            if detections[j].prob[idx] > 0:
                bbox = detections[j].bbox
                bbox = (bbox.x, bbox.y, bbox.w, bbox.h)
                predictions.append((name, detections[j].prob[idx], (bbox)))
    return predictions


def remove_negatives_faster(detections, class_names, num):
    """
    Faster version of remove_negatives (very useful when using yolo9000)
    """
    predictions = []
    for j in range(num):
        if detections[j].best_class_idx == -1:
            continue
        name = class_names[detections[j].best_class_idx]
        bbox = detections[j].bbox
        bbox = (bbox.x, bbox.y, bbox.w, bbox.h)
        predictions.append((name, detections[j].prob[detections[j].best_class_idx],
bbox))
    return predictions


def detect_image(network, class_names, image, thresh=.5, hier_thresh=.5, nms=.45):
    """
        Returns a list with highest confidence class and their bbox
    """
    pnum = pointer(c_int(0))
    predict_image(network, image)
    detections = get_network_boxes(network, image.w, image.h,
                                   thresh, hier_thresh, None, 0, pnum, 0)
    num = pnum[0]
    if nms:
        do_nms_sort(detections, num, len(class_names), nms)
    predictions = remove_negatives(detections, class_names, num)
    predictions = decode_detection(predictions)
    free_detections(detections, num)
    return sorted(predictions, key=lambda x: x[1])

if os.name == "posix":
    cwd = os.path.dirname(__file__)
    lib = CDLL(cwd + "/libdarknet.so", RTLD_GLOBAL)
elif os.name == "nt":
    cwd = os.path.dirname(__file__)
    os.environ['PATH'] = cwd + ';' + os.environ['PATH']
    lib = CDLL("darknet.dll", RTLD_GLOBAL)
else:
    print("Unsupported OS")
    exit
lib.network_width.argtypes = [c_void_p]
lib.network_width.restype = c_int
lib.network_height.argtypes = [c_void_p]
lib.network_height.restype = c_int
copy_image_from_bytes = lib.copy_image_from_bytes
copy_image_from_bytes.argtypes = [IMAGE,c_char_p]
predict = lib.network_predict_ptr
predict.argtypes = [c_void_p, POINTER(c_float)]
predict.restype = POINTER(c_float)
set_gpu = lib.cuda_set_device
init_cpu = lib.init_cpu
make_image = lib.make_image
make_image.argtypes = [c_int, c_int, c_int]
make_image.restype = IMAGE
get_network_boxes = lib.get_network_boxes
get_network_boxes.argtypes = [c_void_p, c_int, c_int, c_float, c_float,
POINTER(c_int), c_int, POINTER(c_int), c_int]
get_network_boxes.restype = POINTER(DETECTION)
make_network_boxes = lib.make_network_boxes
make_network_boxes.argtypes = [c_void_p]
make_network_boxes.restype = POINTER(DETECTION)
free_detections = lib.free_detections
free_detections.argtypes = [POINTER(DETECTION), c_int]
free_batch_detections = lib.free_batch_detections
free_batch_detections.argtypes = [POINTER(DETNUMPAIR), c_int]
free_ptrs = lib.free_ptrs
free_ptrs.argtypes = [POINTER(c_void_p), c_int]
network_predict = lib.network_predict_ptr
network_predict.argtypes = [c_void_p, POINTER(c_float)]
reset_rnn = lib.reset_rnn
reset_rnn.argtypes = [c_void_p]
load_net = lib.load_network
load_net.argtypes = [c_char_p, c_char_p, c_int]
load_net.restype = c_void_p
load_net_custom = lib.load_network_custom
load_net_custom.argtypes = [c_char_p, c_char_p, c_int, c_int]
load_net_custom.restype = c_void_p
```

```
free_network_ptr = lib.free_network_ptr
free_network_ptr.argtypes = [c_void_p]
free_network_ptr.restype = c_void_p
do_nms_obj = lib.do_nms_obj
do_nms_obj.argtypes = [POINTER(DETECTION), c_int, c_int, c_float]
do_nms_sort = lib.do_nms_sort
do_nms_sort.argtypes = [POINTER(DETECTION), c_int, c_int, c_float]
free_image = lib.free_image
free_image.argtypes = [IMAGE]
letterbox_image = lib.letterbox_image
letterbox_image.argtypes = [IMAGE, c_int, c_int]
letterbox_image.restype = IMAGE
load_meta = lib.get_metadata
lib.get_metadata.argtypes = [c_char_p]
lib.get_metadata.restype = METADATA
load_image = lib.load_image_color
load_image.argtypes = [c_char_p, c_int, c_int]
load_image.restype = IMAGE
rgbgr_image = lib.rgbgr_image
rgbgr_image.argtypes = [IMAGE]
predict_image = lib.network_predict_image
predict_image.argtypes = [c_void_p, IMAGE]
predict_image.restype = POINTER(c_float)
predict_image_letterbox = lib.network_predict_image_letterbox
predict_image_letterbox.argtypes = [c_void_p, IMAGE]
predict_image_letterbox.restype = POINTER(c_float)
network_predict_batch = lib.network_predict_batch
network_predict_batch.argtypes = [c_void_p, IMAGE, c_int, c_int, c_int,
                                  c_float, c_float, POINTER(c_int), c_int, c_int]
network_predict_batch.restype = POINTER(DETNUMPAIR)
```

Darknet_video.py:

```
import sys
sys.path.insert(1,'C:/Users/gopal/Downloads/source-code-face-
recognition/sourcecode')
import datetime
from ctypes import *
import random
import os
import csv
import PIL.Image
import cv2
import time
import darknet
import argparse
from PIL import Image
from threading import Thread, enumerate
from queue import Queue
import os
import pytesseract
import numpy as np
import datetime
from simple_facerec import SimpleFacerec


def parser():
    parser = argparse.ArgumentParser(description="YOLO Object Detection")
    '''parser.add_argument("--input", type=str,
default="C:/Users/gopal/Downloads/video.MOV",
                        help="video source. If empty, uses webcam 0 stream")'''
    parser.add_argument("--input", type=str, default=0,
                        help="video source. If empty, uses webcam 0 stream")
    parser.add_argument("--out_filename", type=str, default="",
                        help="inference video name. Not saved if empty")
    parser.add_argument("--weights", default="yolov4.weights",
                        help="yolo weights path")
    parser.add_argument("--dont_show", action='store_true',
                        help="windown inference display. For headless systems")
    parser.add_argument("--ext_output", action='store_true',
                        help="display bbox coordinates of detected objects")
    parser.add_argument("--config_file", default="./cfg/yolov4.cfg",
                        help="path to config file")
    parser.add_argument("--data_file", default="./cfg/coco.data",
                        help="path to data file")
    parser.add_argument("--thresh", type=float, default=.25,
                        help="remove detections with confidence below this value")
    return parser.parse_args()


def str2int(video_path):
    """
    argparse returns and string althout webcam uses int (0, 1 ...)
    Cast to int if needed
    """
    try:
        return int(video_path)
    except ValueError:
        return video_path


def check_arguments_errors(args):
    assert 0 < args.thresh < 1, "Threshold should be a float between zero and one (non-
inclusive)"
    if not os.path.exists(args.config_file):
        raise(ValueError("Invalid config path
{}".format(os.path.abspath(args.config_file))))
    if not os.path.exists(args.weights):
        raise(ValueError("Invalid weight path
{}".format(os.path.abspath(args.weights))))
    if not os.path.exists(args.data_file):
        raise(ValueError("Invalid data file path
{}".format(os.path.abspath(args.data_file))))
    if str2int(args.input) == str and not os.path.exists(args.input):
        raise(ValueError("Invalid video path {}".format(os.path.abspath(args.input))))


def set_saved_video(input_video, output_video, size):
    fourcc = cv2.VideoWriter_fourcc(*"MJPG")
    fps = int(input_video.get(cv2.CAP_PROP_FPS))
    video = cv2.VideoWriter(output_video, fourcc, fps, size)
    return video


def convert2relative(bbox):
    """
    YOLO format use relative coordinates for annotation
    """
    x, y, w, h  = bbox
    _height     = darknet_height
    _width      = darknet_width
    return x/_width, y/_height, w/_width, h/_height


def convert2original(image, bbox):
    x, y, w, h = convert2relative(bbox)

    image_h, image_w, __ = image.shape

    orig_x       = int(x * image_w)
    orig_y       = int(y * image_h)
    orig_width   = int(w * image_w)
    orig_height  = int(h * image_h)

    bbox_converted = (orig_x, orig_y, orig_width, orig_height)

    return bbox_converted


def convert4cropping(image, bbox):
    x, y, w, h = convert2relative(bbox)

    image_h, image_w, __ = image.shape

    orig_left    = int((x - w / 2.) * image_w)
    orig_right   = int((x + w / 2.) * image_w)
    orig_top     = int((y - h / 2.) * image_h)
    orig_bottom  = int((y + h / 2.) * image_h)

    if (orig_left < 0): orig_left = 0
    if (orig_right > image_w - 1): orig_right = image_w - 1
    if (orig_top < 0): orig_top = 0
    if (orig_bottom > image_h - 1): orig_bottom = image_h - 1

    bbox_cropping = (orig_left, orig_top, orig_right, orig_bottom)

    return bbox_cropping


def video_capture(frame_queue, darknet_image_queue):
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_resized = cv2.resize(frame_rgb, (darknet_width, darknet_height),
                            interpolation=cv2.INTER_LINEAR)
        frame_queue.put(frame)
        img_for_detect = darknet.make_image(darknet_width, darknet_height, 3)
        darknet.copy_image_from_bytes(img_for_detect, frame_resized.tobytes())
        darknet_image_queue.put(img_for_detect)
    cap.release()


def inference(darknet_image_queue, detections_queue, fps_queue):
    while cap.isOpened():
        darknet_image = darknet_image_queue.get()
        prev_time = time.time()
        detections = darknet.detect_image(network, class_names, darknet_image,
thresh=args.thresh)
        detections_queue.put(detections)
        fps = int(1/(time.time() - prev_time))
        fps_queue.put(fps)
        print("FPS: {}".format(fps))
        darknet.print_detections(detections, args.ext_output)
        darknet.free_image(darknet_image)
    cap.release()


def drawing(frame_queue, detections_queue, fps_queue):
    random.seed(3)  # deterministic bbox colors
    video = set_saved_video(cap, args.out_filename, (video_width, video_height))

    count=i=0
    while cap.isOpened():
        frame = frame_queue.get()
        detections = detections_queue.get()
        fps = fps_queue.get()
        detections_adjusted = []
        extract_num(frame)
        if frame is not None:
            for label, confidence, bbox in detections:
                bbox_adjusted = convert2original(frame, bbox)
                if str(label)=='person'or str(label)=='car'or str(label)=='bicycle'or
str(label)=='bus'or str(label)=='truck'or str(label)=='motorbike' or str(label)=='sports
ball':
                    detections_adjusted.append((str(label), confidence, bbox_adjusted))
            #print(bbox_adjusted)
            image = darknet.draw_boxes(detections_adjusted, frame, class_colors)
            #ret, frame1 = cap.read()
            font = cv2.FONT_HERSHEY_SIMPLEX
            text = 'Width: ' + str(cap.get(3)) + ' Height:' + str(cap.get(4))
            datet = str(datetime.datetime.now())
            """frame = cv2.putText(frame, text, (10, 100), font, 1,
                        (0, 255, 255), 2, cv2.LINE_AA)"""
            frame = cv2.putText(frame, datet, (10, 25), font, 0.5,
                        (0, 255, 255), 2, cv2.LINE_AA)
            # Detect Faces
            padding = 20
            face_locations, face_names, face_times = sfr.detect_known_faces(frame)
            for face_loc, name, times in zip(face_locations, face_names, face_times):
                y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]

                cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX,
1, (0, 0, 200), 2)
                # cv2.putText(frame, times, (x1, y1 - 10),
cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)
                print(times)
                writer.writerow(times)
                #face=frame[y1:x1,x2:y2]
                #face = frame[max(0, face_loc[1] - padding):min(face_loc[3] + padding,
frame.shape[0] - 1),
                #max(0, face_loc[0] - padding):min(face_loc[2] + padding,
frame.shape[1] - 1)]
                blob = cv2.dnn.blobFromImage(frame, 1.0, (227, 227),
MODEL_MEAN_VALUES, swapRB=False)
                genderNet.setInput(blob)
                genderPred = genderNet.forward()
                gender = genderList[genderPred[0].argmax()]

                ageNet.setInput(blob)
                agePred = ageNet.forward()
                age = ageList[agePred[0].argmax()]

                label = "{},{}".format(gender, age)
                cv2.rectangle(frame, (face_loc[0], face_loc[1] - 30), (face_loc[2],
face_loc[1]), (0, 255, 0), -1)
```

```
            cv2.putText(frame, label, (face_loc[0], face_loc[1] - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2,
                        cv2.LINE_AA)
        #cv2.imshow("Age-Gender", frame)
        '''count += 1
        x, y, z, h = darknet.bbox2points(bbox_adjusted)
        imcap = image[x:h, y:z]
        if(count>=0 and float(confidence)>99):
            cv2.imwrite(os.path.join("C:/Users/gopal/OneDrive/Desktop/darknet-
master/pics",'Frame' + str(datetime.datetime.now()) + '.jpg'), imcap)
        i+=1
        count=0'''


data=(str(label),str((datetime.datetime.now().strftime("%H:%M:%S"))),str(datetime.d
atetime.today().strftime("%Y:%m:%d")))
        print(data)
        writer.writerow(data)
        '''r = open('timeline.csv', 'r')
        reader = r.readlines(-1)
        if (data[1] - reader[1]):
            print(str(reader))'''
        cv2.imshow('Inference', image)
        if not args.dont_show:
            cv2.imshow('Inference', image)
        if args.out_filename is not None:
            video.write(image)
        if cv2.waitKey(fps) == 27:
            break
    cap.release()
    video.release()
    cv2.destroyAllWindows()
def detect(frame):
    bounding_box_cordinates, weights = HOGCV.detectMultiScale(frame,
winStride=(4, 4), padding=(8, 8), scale=1.03)

    person = 1
    for x, y, w, h in bounding_box_cordinates:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(frame, f'person {person}', (x, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
```

```
        person += 1

    cv2.putText(frame, 'Status : Detecting ', (40, 40),
cv2.FONT_HERSHEY_DUPLEX, 0.8, (255, 0, 0), 2)
    cv2.putText(frame, f'Total Persons : {person - 1}', (40, 70),
cv2.FONT_HERSHEY_DUPLEX, 0.8, (255, 0, 0), 2)
    cv2.imshow('output', frame)
    return frame
def extract_num(img_name):
    img = img_name
    # img = cv2.imread(img_name) ## Reading Image
    # Converting into Gray
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Detecting plate
    nplate = cascade.detectMultiScale(gray, 1.1, 4)
    #nplate =
cv2.CascadeClassifier('haarcascade_russian_plate_number.xml').detectMultiScale(gra
y, 1.1, 4)
    for (x, y, w, h) in nplate:
        # Crop a portion of plate
        a, b = (int(0.02 * img.shape[0]), int(0.025 * img.shape[1]))
        plate = img[y + a:y + h - a, x + b:x + w - b + 25, :]
        # make image more darker to identify the LPR
        ## iMAGE PROCESSING
        kernel = np.ones((1, 1), np.uint8)
        plate = cv2.dilate(plate, kernel, iterations=1)
        plate = cv2.erode(plate, kernel, iterations=1)
        plate_gray = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)
        (thresh, plate) = cv2.threshold(plate_gray, 127, 255, cv2.THRESH_BINARY)
        # Feed Image to OCR engine
        read = pytesseract.image_to_string(plate)
        read = ''.join(e for e in read if e.isalnum())
        print(read)
        if((len(read) >= 5 and len(read) <= 10) and not os.path.isfile(str(read)+'.jpg')):
            data=[str(read),str(datetime.datetime.now())]
            writer.writerow(data)
        stat = read[0:2]
        try:
            # Fetch the State information
            print('Car Belongs to', states[stat])
        except FileNotFoundError:
          continue
        except:
          print('State not recognised!!')

        print(read)
        cv2.rectangle(img, (x, y), (x + w, y + h), (51, 51, 255), 2)
        cv2.rectangle(img, (x, y - 40), (x + w, y), (51, 51, 255), -1)
        cv2.putText(img, read, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,
255, 255), 2)
        # cv2.imshow('PLate',plate)
        # Save & display result image
        if ((len(read) >= 5 and len(read) <= 10) and not os.path.isfile(str(read)+'.jpg')):
          cv2.imwrite(os.path.join("C:/Users/gopal/OneDrive/Desktop/darknet-
master/pics", 'plate' + read + '.jpg'), plate)
        else:
          cv2.imwrite(os.path.join("C:/Users/gopal/OneDrive/Desktop/darknet-
master/pics",'plateUNKNOWN.jpg'),plate)

    # cv2.imshow("Result", img)
    # cv2.imwrite('result.jpg',img)
    # cv2.waitKey(0)
    # cv2.destroyAllWindows()

if __name__ == '__main__':
    HOGCV = cv2.HOGDescriptor()
    HOGCV.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
    # Encode faces from a folder
    sfr = SimpleFacerec()
    sfr.load_encoding_images("images/")
    pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-
OCR\tesseract.exe'
    cascade = cv2.CascadeClassifier("haarcascade_russian_plate_number.xml")
    states = {"AN": "Andaman and Nicobar", "AP": "Andhra Pradesh", "AR":
"Arunachal Pradesh", "AS": "Assam",
              "BR": "Bihar",
              "CH": "Chandigarh", "DN": "Dadra and Nagar Haveli", "DD": "Daman and
Diu", "DL": "Delhi", "GA": "Goa",
              "GJ": "Gujarat",
              "HR": "Haryana", "HP": "Himachal Pradesh", "JK": "Jammu and Kashmir",
"KA": "Karnataka", "KL": "Kerala",
            "LD": "Lakshadweep", "MP": "Madhya Pradesh", "MH": "Maharashtra", 
"MN": "Manipur", "ML": "Meghalaya",
            "MZ": "Mizoram", "NL": "Nagaland", "OD": "Odissa", "PY": "Pondicherry", 
"PN": "Punjab", "RJ": "Rajasthan",
            "SK": "Sikkim", "TN": "TamilNadu", "TR": "Tripura", "UP": "Uttar 
Pradesh", "WB": "West Bengal",
            "CG": "Chhattisgarh", "TS": "Telangana", "JH": "Jharkhand", "UK": 
"Uttarakhand"}
    faceProto = "opencv_face_detector.pbtxt"
    faceModel = "opencv_face_detector_uint8.pb"

    ageProto = "age_deploy.prototxt"
    ageModel = "age_net.caffemodel"

    genderProto = "gender_deploy.prototxt"
    genderModel = "gender_net.caffemodel"

    faceNet = cv2.dnn.readNet(faceModel, faceProto)
    ageNet = cv2.dnn.readNet(ageModel, ageProto)
    genderNet = cv2.dnn.readNet(genderModel, genderProto)

    MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
    ageList = ['(0-2)', '(4-6)', '(8-12)', '(15-20)', '(21-32)', '(38-43)', '(48-53)', '(60-100)']
    genderList = ['Male', 'Female']
    data = []
    f=open("timeline.csv",'w')
    writer=csv.writer(f)
    frame_queue = Queue()
    darknet_image_queue = Queue(maxsize=1)
    detections_queue = Queue(maxsize=1)
    fps_queue = Queue(maxsize=1)

    args = parser()
    check_arguments_errors(args)
    network, class_names, class_colors = darknet.load_network(
            args.config_file,
            args.data_file,
            args.weights,
            batch_size=1
        )
    darknet_width = darknet.network_width(network)
    darknet_height = darknet.network_height(network)
    input_path = str2int(args.input)
    cap = cv2.VideoCapture(input_path)
    print(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    print(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    video_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    video_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    Thread(target=video_capture, args=(frame_queue, darknet_image_queue)).start()
    Thread(target=inference, args=(darknet_image_queue, detections_queue,
fps_queue)).start()
    Thread(target=drawing, args=(frame_queue, detections_queue, fps_queue)).start()
```

CHAPTER 7

7 SYSTEM TESTING / PERFORMANCE ANALYSIS

The testcases and their reports from the testing of the programs are shown below. The results of the program is shown below in the next section.

TESTCASES AND REPORTS:

Test case 1:

A case where many people are in cam range:

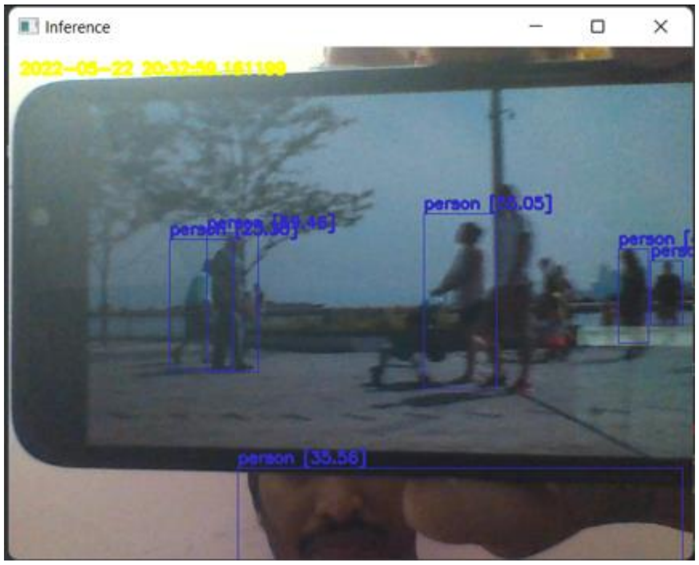


Fig 7.1

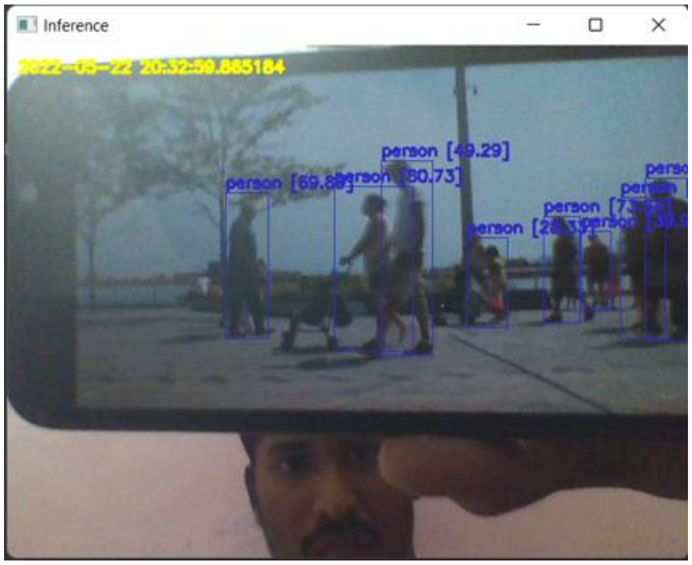


Fig 7.2

- ➢ The program was able to detect even more than 10 people in the frame flawlessly during live video testing without any problem. It was also able to data log the people in the frame without any issues.

Test case 2:

A case where people and vehicles come in frame together:

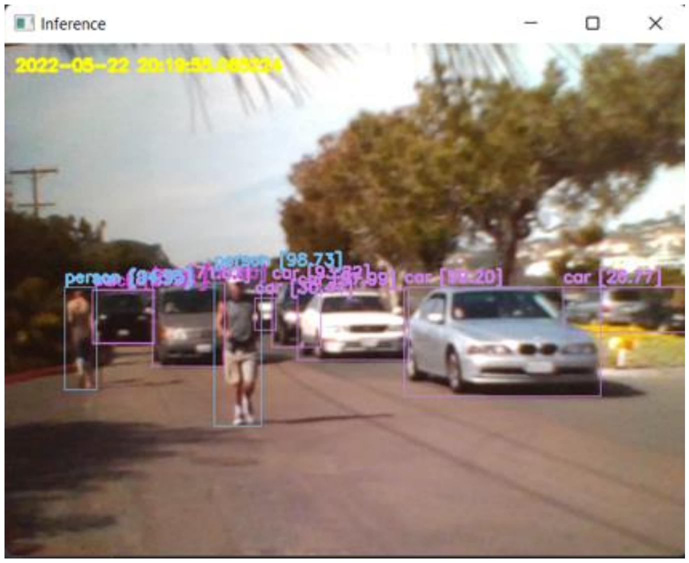


Fig 7.3

- The program is able to identify all people in the frame alongside vehicles without any issues. Since this video of people and cars is shown to camera using a mobile phone screen the number plates were not clear for the program to recognize and crop.

Test case 3:

A case where a car with a num plate visible in frame comes inside

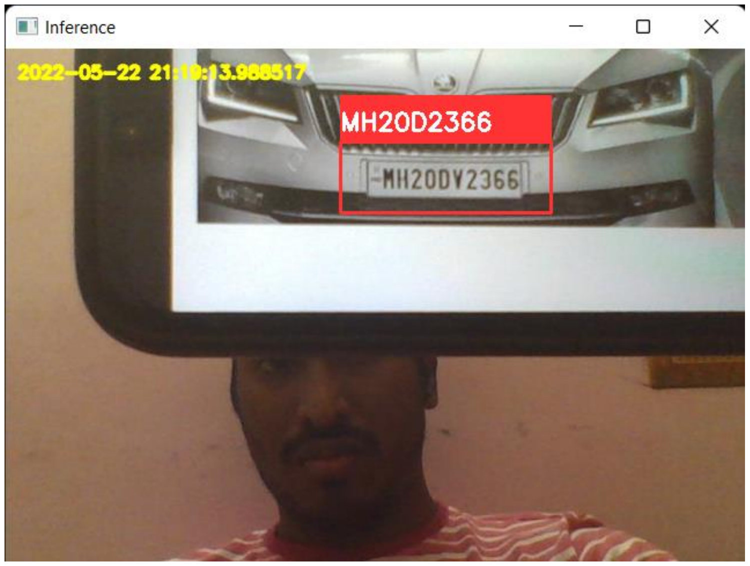


Fig 7.4

- The program was able to find the number plate in frame and was able to recognize the text written in the number plate for the most part except it left a character.
- But it did crop out the number plate and save it in the data logging directory.
- The cropped plate is shown below:

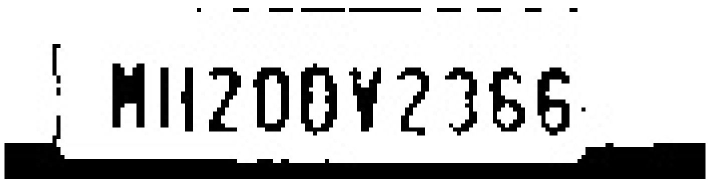


Fig 7.5

Test case 4:

A case where it detects person, his or her age along with sports ball come in the frame.

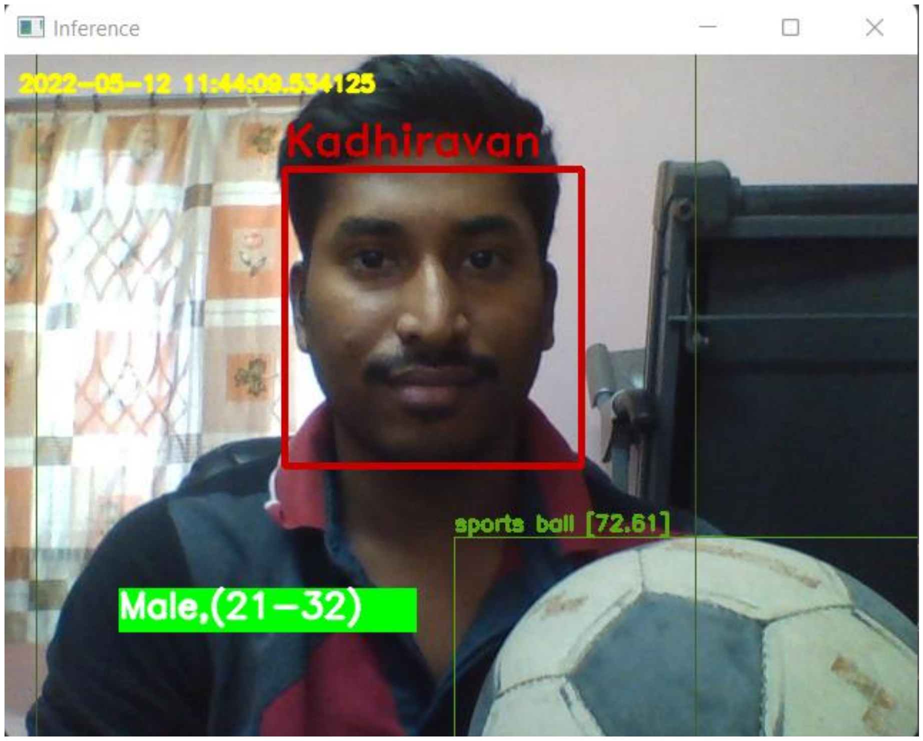


Fig 7.6

The program was able to detect all the parameters 1ike age, ball and the person also documenting the events.

CHAPTER 8

8 CONCLUSION

RESULTS AND DISCUSSION:

- The results from the above code using object detection are shown in the upcoming slides. The Ai detects the person in the frame using the reference photo provided to it.
- It not just detects but also notes the time the particular person is in the frame.

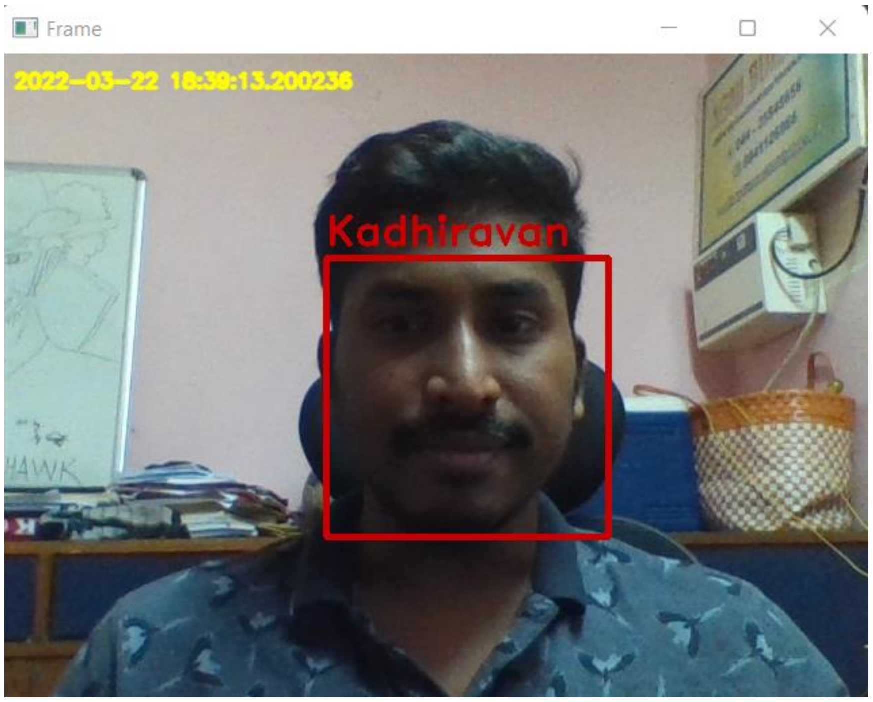


Fig 7.7

- It detects the number plates of the vehicles in frame and stores the cropped image of the plate while storing the number and state of the vehicle registration in a document.

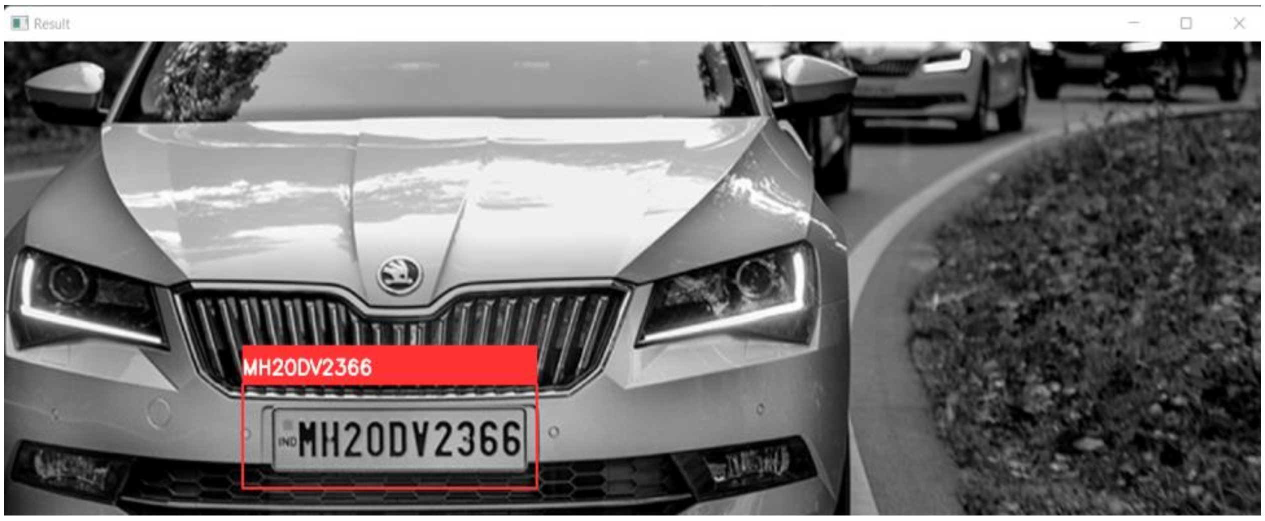


Fig 7.8

➢ The cropped image is shown here

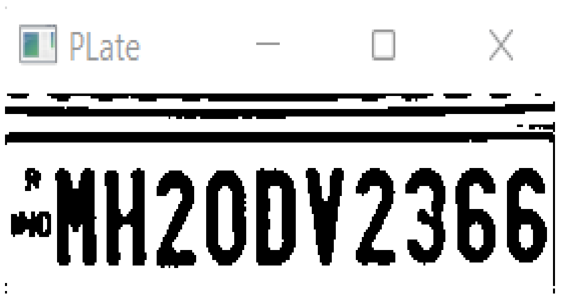
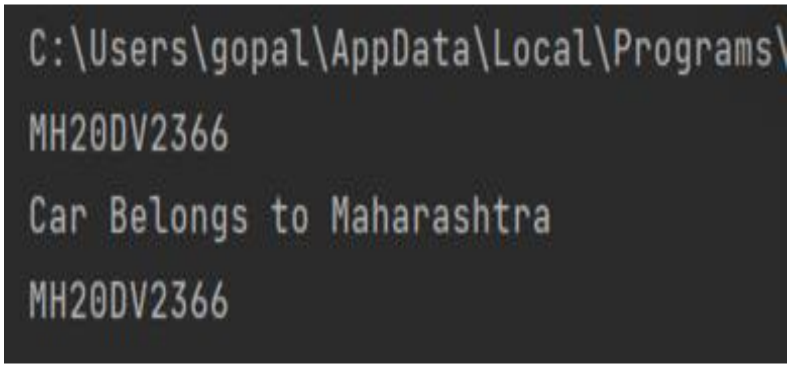


Fig 7.9



```
C:\Users\gopal\AppData\Local\Programs\
MH20DV2366
Car Belongs to Maharashtra
MH20DV2366
```

Fig 7.10

➢ The Ai can detect multiple people in the frame at the same time.

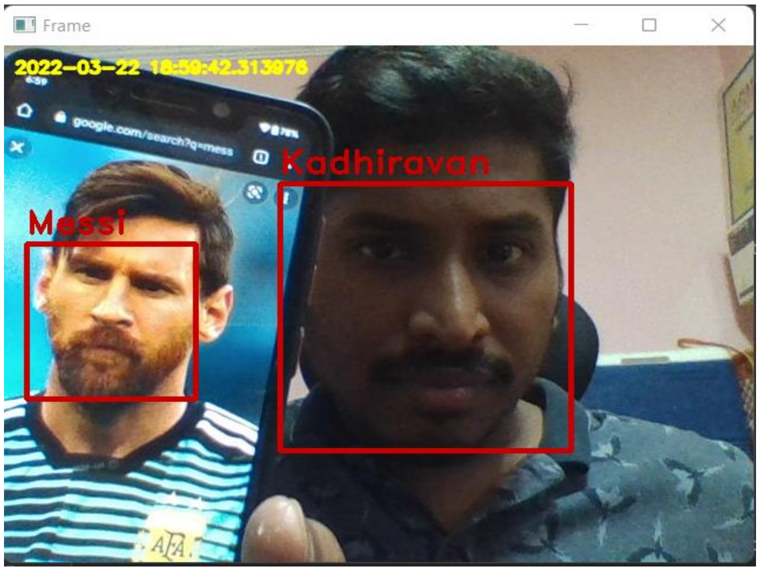
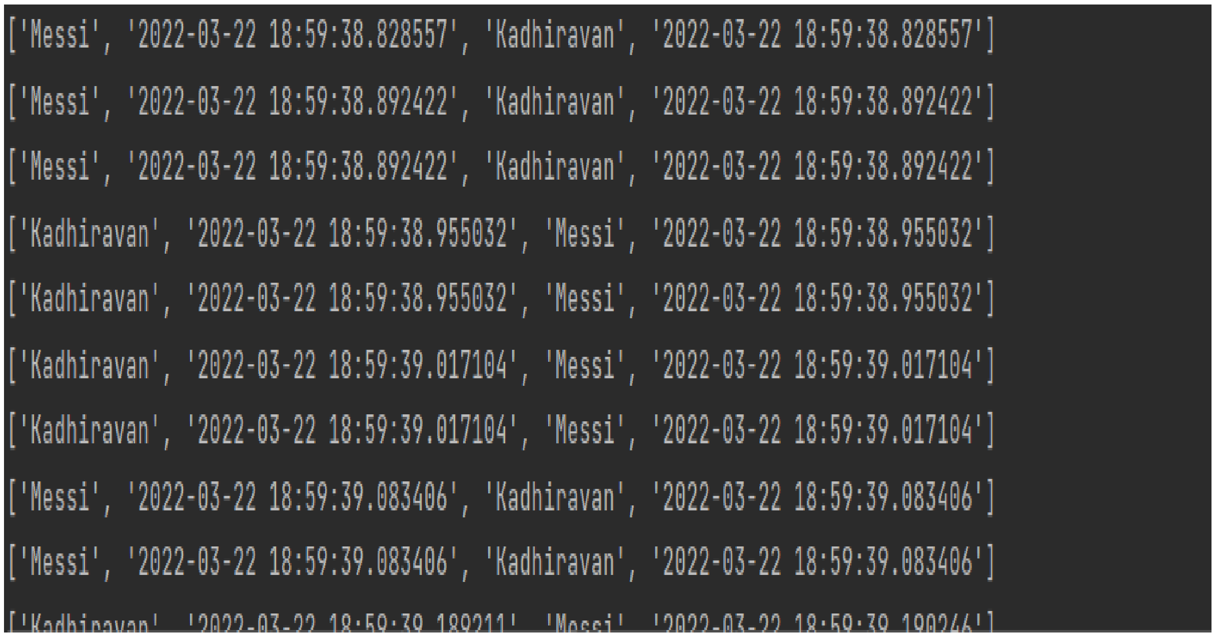


Fig 7.11

- Here look at an example of a person which the system detects as “KADHIRAVAN” and at the same time he shows a image in his phone that the system detects as “MESSI” since the data of MESSI is already present in our database.



```
['Messi', '2022-03-22 18:59:38.828557', 'Kadhiravan', '2022-03-22 18:59:38.828557']
['Messi', '2022-03-22 18:59:38.892422', 'Kadhiravan', '2022-03-22 18:59:38.892422']
['Messi', '2022-03-22 18:59:38.892422', 'Kadhiravan', '2022-03-22 18:59:38.892422']
['Kadhiravan', '2022-03-22 18:59:38.955032', 'Messi', '2022-03-22 18:59:38.955032']
['Kadhiravan', '2022-03-22 18:59:38.955032', 'Messi', '2022-03-22 18:59:38.955032']
['Kadhiravan', '2022-03-22 18:59:39.017104', 'Messi', '2022-03-22 18:59:39.017104']
['Kadhiravan', '2022-03-22 18:59:39.017104', 'Messi', '2022-03-22 18:59:39.017104']
['Messi', '2022-03-22 18:59:39.083406', 'Kadhiravan', '2022-03-22 18:59:39.083406']
['Messi', '2022-03-22 18:59:39.083406', 'Kadhiravan', '2022-03-22 18:59:39.083406']
['Kadhiravan', '2022-03-22 18:59:39.189211', 'Messi', '2022-03-22 18:59:39.190246']
```

Fig 7.12

- This image shows the details of the person who enters the frame.
- When a known person enters the field the system recognizes and shows its name along with the time he enters the frame.
- When an unknown person enters it shows them as unknown along with the time then enter the frame.
- Then it tries to encode their face and stores them in the database.
- It shows the recorded document with time logging.

- This module is used to detect the age and gender of the person.

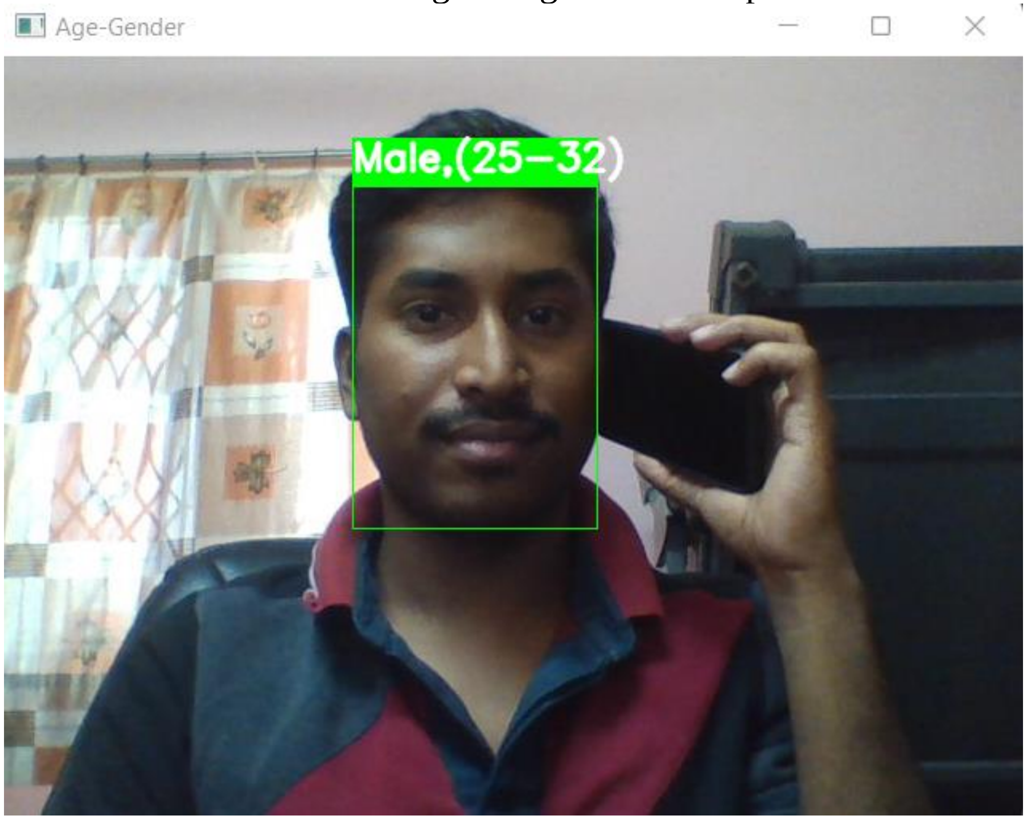


Fig 7.13

8.2 CONCLUSION

- Detecting a human being accurately is a major topic of vision research due to its wide range of applications and it is also a challenging process to obtained the image from the low resolution and the major applications of human and object detection of surveillance video are reviewed.
- A discussion is to made to point a future work needed to improve the human detection process in surveillance video. These includes exploiting a multi-view approach and adopting an improved model based images.
- Nowadays, time is everything, in such a fast paced world even if sometime can be saved using this program then its worthwhile to give it a shot since it saves a lot of time from investigation.

FUTURE ENHANCEMENTS

- The program can be developed many new features within this project that help to the user to detect any mistake that takes place. Detecting the count in groups and identifying the dimensions of the person and detecting the types of vehicle and also detection of animals can also be added to this work.
- implementation to determine the actions on what the people are doing in the frame of camera.

REFERENCES:

[1] Nurhopipah, Ade & Harjoko, Agus. (2018). Motion Detection and Face Recognition for CCTV Surveillance System. IJCCS (Indonesian Journal of Computing and Cybernetics Systems). 12. 107. 10.22146/ijccs.18198.

[2] A. Jalal and S. Kamal, "Real-time life logging via a depth silhouette-based human activity recognition system for smart home services," 2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2014, pp. 74-80, doi: 10.1109/AVSS.2014.6918647.

[3] Serhii Maksymenko. (2019). "How to build a face detection and recognition system". https://towardsdatascience.com/how-to-build-a-face-detection-and-recognition-system-f5c2cdfbeb8c

[4] A. Raghunandan, Mohana, P. Raghav and H. V. R. Aradhya, "Object Detection Algorithms for Video Surveillance Applications," 2018 International Conference on

Communication and Signal Processing (ICCSP), 2018, pp. 0563-0568, doi: 10.1109/ICCSP.2018.8524461.

[5] S. Mane and S. Mangale, "Moving Object Detection and Tracking Using Convolutional Neural Networks," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 1809-1813, doi: 10.1109/ICCONS.2018.8662921.

[6] J. Chun and S. Park, "RGB-D Model Based Human Detection and Tracking Using 3D CCTV," 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2018, pp. 758-762, doi: 10.1109/UEMCON.2018.8796713.

[7] H. Tanikella, S. Namkoong and B. L. Smith, "Design of a multiple-target based automated camera repositioning system for integrating CCTV with video image vehicle detection systems," Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749), 2004, pp. 848-852, doi: 10.1109/ITSC.2004.1399014.

[8] A. Dabrowski, D. Cetnarowicz, P. Pawlowski and M. Stankiewicz, "People recognition and tracking methods for control of viewpoint in CCTV systems," 2011 20th European Conference on Circuit Theory and Design (ECCTD), 2011, pp. 849-852, doi: 10.1109/ECCTD.2011.6043826.

[9] S. Hargude and S. R. Idate, "i-surveillance: Intelligent surveillance system using background subtraction technique," 2016 International Conference on Computing

Communication Control and automation (ICCUBEA), 2016, pp. 1-5, doi: 10.1109/ICCUBEA.2016.7860046.

[10] J. G. Dastidar and R. Biswas, "Tracking Human Intrusion through a CCTV," 2015 International Conference on Computational Intelligence and Communication Networks (CICN), 2015, pp. 461-465, doi: 10.1109/CICN.2015.95.