**A Project Report**

**On**

# 1. "Smart Attendance Management System"

**and**

# 2. "Virtual Cursor"

**Submitted for the course "Computer Vision" (EC6526)**

**Batch: 2020-2024**

**Department of Electronics & Communication Engineering**

**National Institute of Technology Patna**

**Submitted To:**

**Dr. Bambam Kumar**

**Assistant Professor, ECE, NIT Patna**

**Submitted By:**

1. **Pradeep Anand (2004091)**
2. **Amit Shukla (2004096)**
3. **Ayush Gupta (2004106)**
4. **Rustam Ali (2004126)**

## Abstract

**We've developed two projects as a part of Computer Vision (EC6526) Course. – 1. Smart Attendance Management System and 2. Virtual Cursor.**

**In the Smart Attendance Management System, automatic attendance of persons can be marked if their photos are present in the database. Also, if unknown person or someone who's photos are not available in the database, no attendance will be marked for that person.**

**In the Virtual Cursor project, computer screen can be virtually controlled and cursor can be moved with the help of fingers. Movement of cursor and clicks can be made without help of any physical mouse.**

**Both projects are made with the help of computer vision concepts and using OpenCV and Python.**

# Acknowledgement

**We would like to express my special thanks to our professor Dr. Bambam Kumar, Assistant Professor, NIT Patna who gave us the opportunity to develop the project and keep us motivated throughout the process. His teachings and explanations of conceptual and complex computer vision topics in a very easy manner helps us to be engaged in the project.**

**Also, by doing lots of research, we came to know about so many new things which helps us to build this project.**

**We're also thankful to all our friends who have supported this project by sharing their photos in order to be included in the database.**

**Lastly, we would like to acknowledge some of YouTube channels whose content make us learn and apply OpenCV and Python in a practical way through this project.**

**Murtaza's Workshop - Robotics and AI**

**Programming Hero**

## Table of Contents

# Introduction

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

Steps Involves in Computer Vision:-


| | | | |
|---|---|---|---|
| -Images<br>-Videos | Getting the data ready<br><br>-Standardization images<br>- Transformation of color and more... | -Find exclusive and differentiating information about the image. | - Learn from the extracted features to predict and classify objects. |

1. Image/Video Acquisition from Camera
2. Image Processing
3. Understanding Image/Feature extraction and Decision making

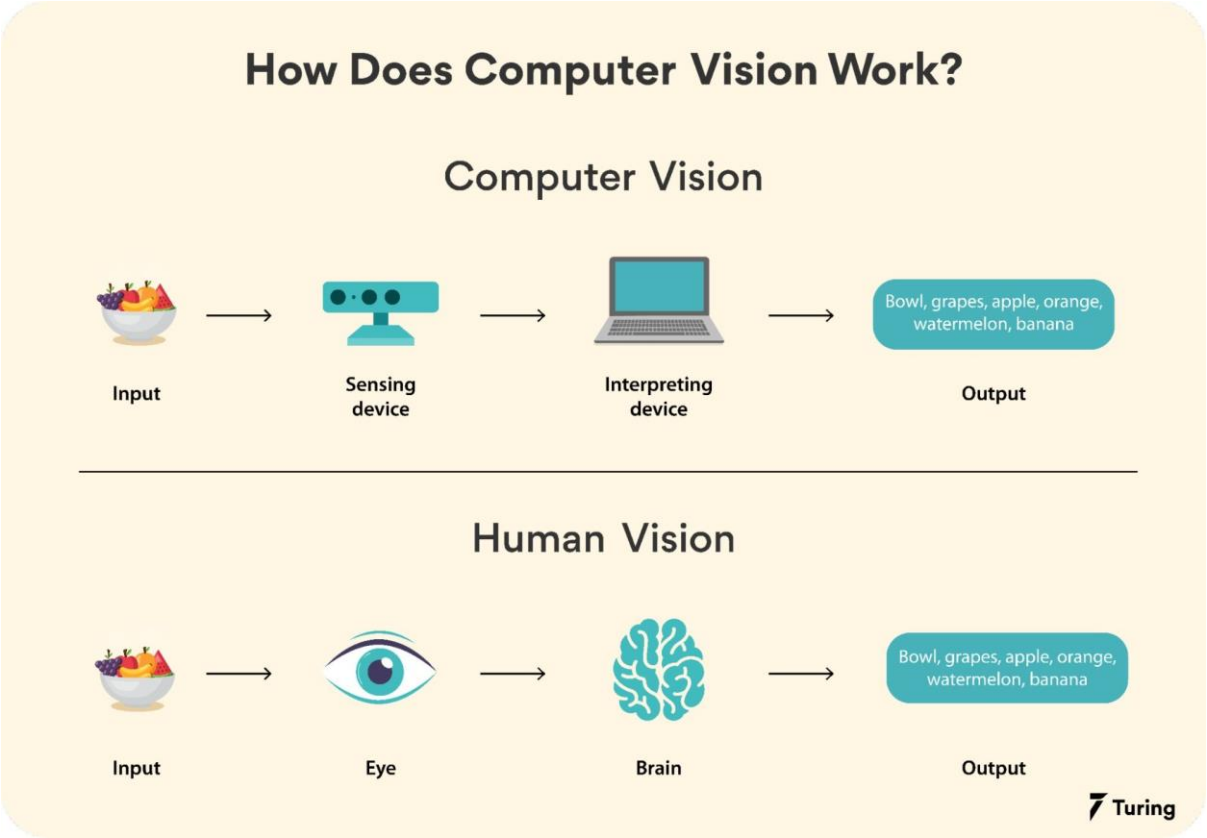
## How Does Computer Vision Work?

Computer Vision works in the same way as our Human Visual System does.

Following are major components of Computer Vision:-

a. Camera Sensor
b. Image Processor/CPU
c. AI model, Computer Neural Network (CNN) for decisions

This can be depicted as the diagram given below:-



## ❖ Research Paper Citation

---

Citation:- Face Recognition System

by **Shivam Singh** & **Prof. S. Graceline Jasmine**
Department of SCSE
Vellore Institute of Technology,
Chennai Tamil Nadu, India

**Published by :**
**International Journal of Engineering Research & Technology (IJERT)**
**ISSN: 2278-0181**
**Vol. 8 Issue 05, May-2019**
**IJERTV8IS050150**

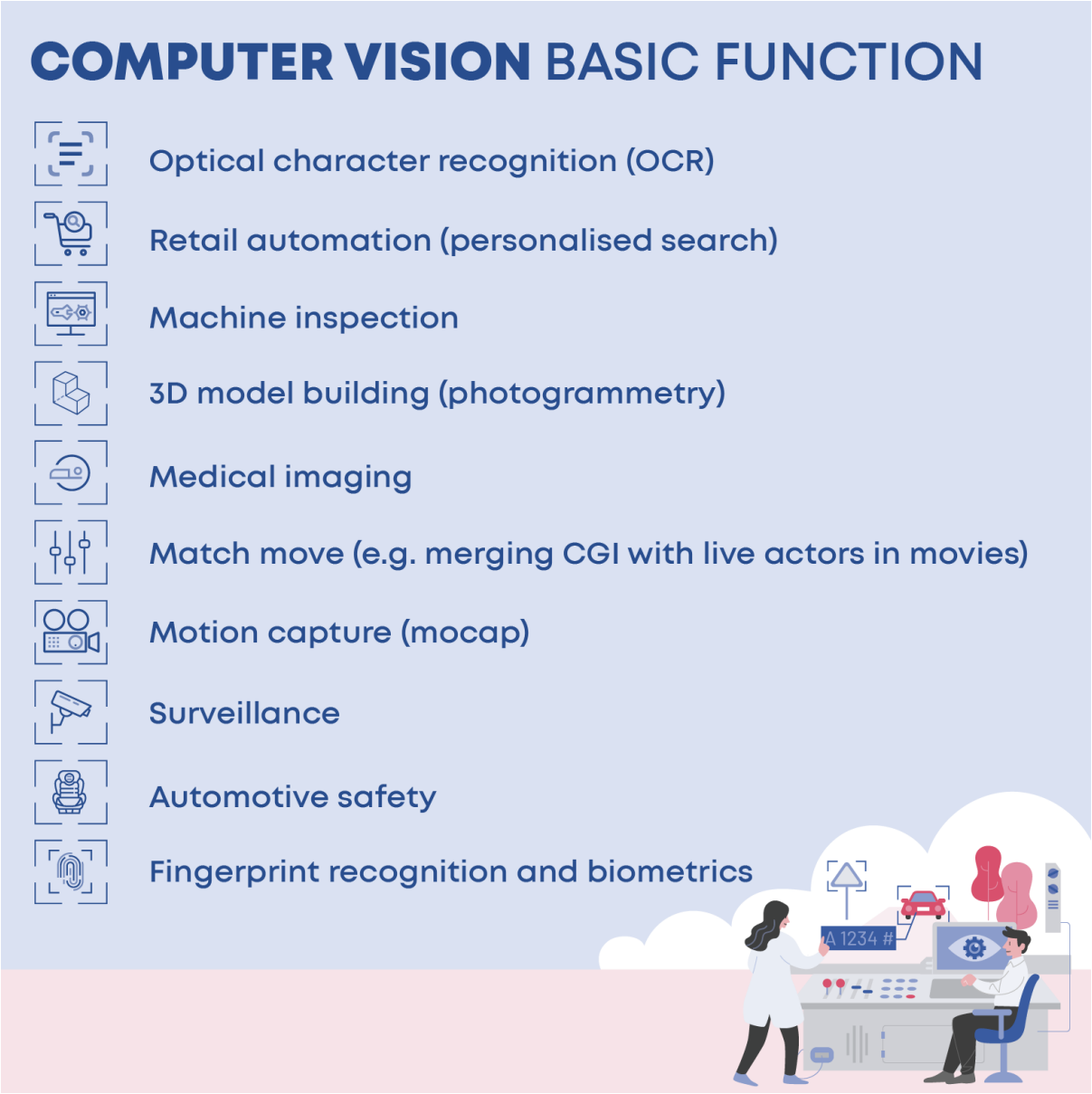
**www.ijert.org**

---

Mentioned research paper shows implementation of Face-Recognition Project based on face-recognition and it uses Haar-Cascade Algorithm and outdated Viola-Jones Algorithm to detect faces. Haar-Cascade algorithm works well but still not much efficient and modern way for face-recognition. Also, much optimized and great algorithm are available nowadays to detect faces.

So, in our project we've used **OpenFace** Algorithm (in place of Haar-Cascade) & Histogram of Oriented Gradients (**HOG**) (in place of Viola-Jones Algorithm) which is much reliable and efficient technique for detection framework. We can read more about these algorithms and how it is implemented in **Face-Recognition Module (Methodology) starts from Page no. – 14**.
It is already proved and tested, our algorithms are much more advanced and effective to generate desired and correct output with almost *95-99% accuracy*. Hence, we have achieved a success in implementing Face-Recognition based Attendance Management System with more effective and reliable algorithms.

**Applications of Computer Vision**



# Synopsis of the Projects (Introduction)

## 1. Smart Attendance Management System

| | |
|---|---|
| Title of The Project | Smart Attendance Management System |
| Category | **Face-Recognition** & **AI** (Computer Vision) |
| Objective | Attendance of a person can be marked by recognizing his/her face by a computer system. |
| Languages & Technology Used | **Python** (v3.7) & **OpenCV** |
| Modules & Library Used | face-recognition (1.3.0), opencv-python (4.7.0.72), cmake (3.17.2) |
| Software/IDE Used | PyCharm Community Edition (v2022.3.3), Windows 11 |
| Minimum Hardware Requirements | OS – Windows 7 or Higher<br>RAM – 4 GB<br>Hard Disk Space – 100-128 GB |

## 2. Virtual Cursor

| | |
|---|---|
| Title of The Project | Virtual Cursor (AI Virtual Mouse) |
| Category | Hand-Detection & AI (Computer Vision) |
| Objective | To control computer screen cursor with fingers and also clicks can be made (without a physical mouse) |
| Language & Technology Used | **Python** (v3.7) & **OpenCV** |
| Modules/Library Used | PyAutoGUI (0.9.5), mediapipe (0.8.10) opencv-python (4.6.0.66) |
| Software/IDE Used | PyCharm Community Edition (v2022.3.3), Windows 11 |
| Minimum Hardware Requirements | OS – Windows 7 or Higher<br>RAM – 4 GB<br>Hard Disk Space – 100-128 GB |

# Beginners Guide to OpenCV (Data/Module Used)

- **What is OpenCV?**

OpenCV is an open source cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

- **Features of OpenCV Library**

Using OpenCV library, we can –

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc.

- **Installation Guide**

***NOTE***:- In order to use OpenCV library on our system, we can either download the OpenCV software on our system from https://opencv.org/releases/ OR we can directly download library on our IDE (PyCharm) and import it (type - import cv2)

**Installing OpenCV through command on the command prompt: type (or paste) "pip install opencv-python" command on your command prompt and hit enter. The download and installation will automatically take place. Note that before installation through pip, you need to download and install any Python version (most recommended – Python v3.7) for the system.**

**OpenCV Important Functions and Properties (At a Glance):-**

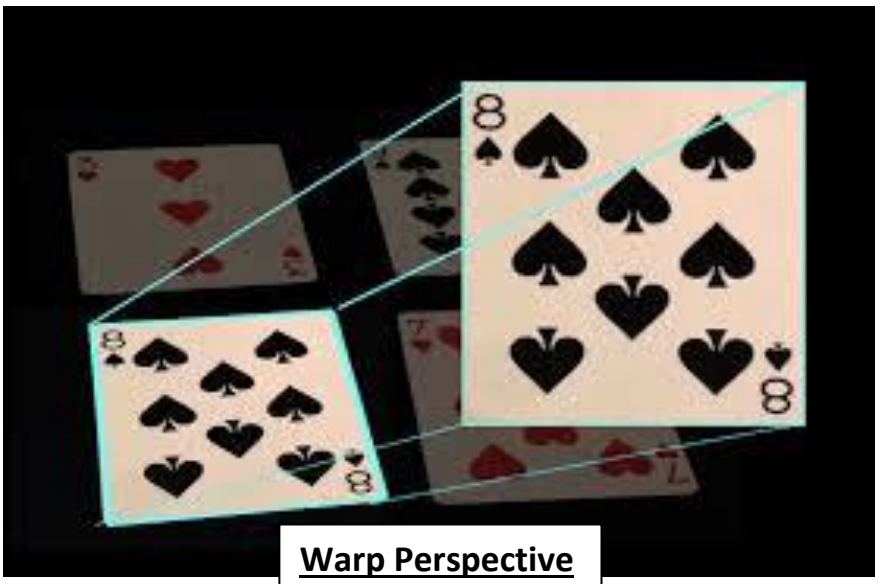
| Description/Task | Function/Properties or Code |
|---|---|
| Read Images | **cv2.imread("source location with format type")** |
| Display/show Images/Videos/Webcam | **cv2.imshow("Name", img_var)** |
| Create Delay | **cv2.waitKey(numbers in miliseconds)**<br>Put 0 for infinite delay/wait |
| Show Video/Webcam | **cap = cv2.videoCapture("source location with format type")**<br>**While True:**<br>**Success,img = cap.read()**<br>**Cv2.imshow("Video/Webcam", img)**<br>**if cv2.waitKey(1) & 0xFF == ord('q')**<br>**break**<br>- For Webcam – **cv2.videoCapture(webcam no.)**<br>Rest everything is same as video.<br>- Put 0 for using default webcam. |
| Setting different parameters such as height, width, brightness etc. for video/webcam | **cap.set(ID_no, parameter value)**<br>For width – ID no. is 3<br>For height – ID no. is 4<br>For Brightness – ID no. is 10 …….and so on |
| Conver RGB Image to Gray Image | cv2.cvtColor(img,cv2,COLOR_BGR2GRAY) |
| Blur Image | **cv2.GaussianBlur(img,(Kernel Size,Sigma X)**<br>- Kernel Size should be odd – 3*3, 5*5 etc.<br>- Take Sigma X as 0 |
| Edge Detection in Image (Canny Image) | **Cv2.canny(img, Threshold1, Threshold2)**<br>Threshold value increases => Less no. of edges detected |
| Dialate Image<br>- To add thickness to Canny Image | **Cv2.dilate(cannyImg, Kernel, iterations=value)**<br>- To create Kernel<br>Kernel = np.ones((odd size), np.unit8)<br>- Iterations value increases => More thickness |
| Erose Image (opposite of Dialation) | **Cv2.erode(img, kernel, iterations=value)** |

| | |
|---|---|
| Get Shape of Image | **Img.shape** – height, width, channel no.<br>For BGR channel – it will give value 3 as channel no. |
| Resize Image (change pixel size not the quality) | **Cv2.resize(img,width, height)** |
| Crop Image | **Img[hc1:hc2, wc1:wc2]** – matrix method<br>- hc is height coordinate<br>- wc is width coordinate |
| Create Image | **Img = np.zeros((width,height,channel no), np.unit8)**<br>- zeros for creating black image<br>- ones for creating white image |
| Color Image | **Img[:] = B,G,R**<br>- no value before and after colon (:) means color will be applied for whole image. So, if we want to add color to specific position we can define location separated by colon (:). |
| Draw line over Image | **Cv2.line(img,(wc),(hc),(color),thickness)**<br>- width coordinate can be get by img.shape[1]<br>- height coordinate can be get by img.shape[0] |
| Draw Rectangle on Image | **Cv2.rectangle(img,(wc),(hc),(color),thickness)** |
| Draw Circle on Image | **Cv2.circle(img,(center coordinate), radius, (color), thickness)** |
| Text on Image | **Cv2.putText(img,"Text",(begining coordinate),font,font scale, color, thickness)** |
| Join Images<br>**Problems** –<br>1. Don't work with different color images<br>2. Can't resize stacked images | - Horizontal Join –<br>**np.hstack((img1,img2))**<br>- Vertical Join –<br>**np.vstack((img1,img2))** |

## Warp Perspective

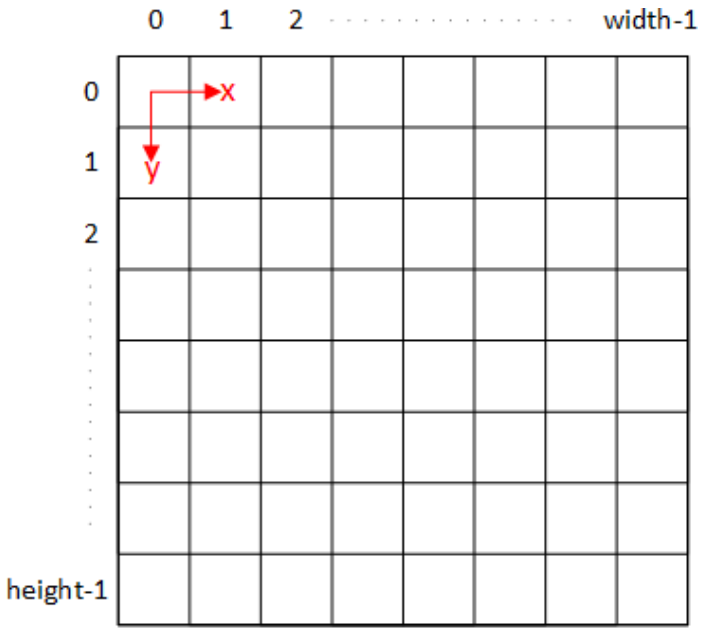
**Perspective Warp algorithm** allows for correcting perspective distortion caused by camera misalignment with respect to the object plane being captured.

**pt1 = np.float32([[c1],[c2],[c3],[c4]])**
**pt2 = np.float32([[0,0],[width,0],[0,height],[width,height]])**
**matrix = cv2.getPerspectiveTransform(pt1,pt2)**
**OutputImg = cv2.warpPerspective(img,matrix,(width,height))**



**Warp Perspective**

**Important point to be noted –**




Buffer/Pixel Coordinates (integer)

# Face-Recognition Module (Methodology)

Face recognition is the process of detecting or confirming the identity of individuals based on facial images. Various companies and applications often use a facial recognition system to verify the identity of people in videos, photos, or the real world.

To use face-recognition in our project, we have to download and install face-recognition module by **Adam Geitgey** in our system directly or on PyCharm IDE.

## How Face Recognition Works:-

### *Step 1: Finding all the Faces*

The first step in our pipeline is *face detection.* It can be achieved by old technique – Viola-Jones object detection framework proposed in 2001 by Paul Viola and Michael Jones. Much reliable and effective solution exist now. This is called Histogram of Oriented Gradients (HOG) and invented in 2005.

### *Step 2: Posing and Projecting Faces*

Next step is to create warp of isolated image so that the eyes and lips are always in the sample place in the image. we are going to use an algorithm called **face landmark estimation**.

We'll come up with 68 landmark points on each face - the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. And train a machine learning algorithm to get these landmarks on the face. Now according to these landmark points we'll apply some simple image transformations (Affine Transformations) to warp image and get closer and centered view of the face on image.

### *Step 3: Encoding Faces*

Now, we'll get measurements of face on image and compare it with unknown face measurements. To do this, we need to train or use a already trained deep convolutional neural network. It will generate 128 measurements for each face in the image.

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

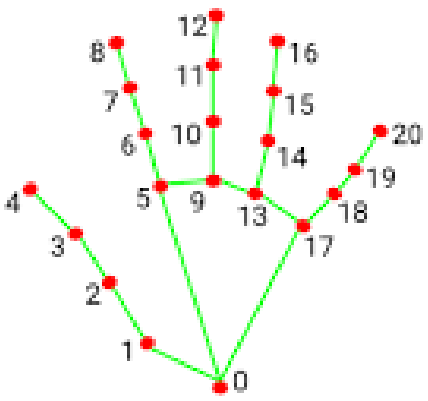
a. Load a training face image of a known person
b. Load another picture of the same known person
c. Load a picture of a totally different person
   These 128 measurements of each face are called **embedding.**

This training of deep convolutional neural network requires lot of data and computational power. So we need not to train it. We can use trained one by OpenFace developed by Brandon Amos and team and is based on the CVPR 2015 paper FaceNet: A Unified Embedding for Face Recognition and Clustering by Florian Schroff, Dmitry Kalenichenko, and James Philbin at Google.

But in this project we've used Haar-Cascade which is a feature-based object detection algorithm to detect objects from images. A cascade function is trained on lots of positive and negative images for detection. The algorithm does not require extensive computation and can run in real-time. It provides many trained models (XML Files) for millions of object types including face and body. According to our application and need we can use Haar-Cascade XML Files in our project.

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

This is the last step and requires any classification algorithm such as SVM classifier. The classifier takes measurements from new test image and match/compare with the images present in our database and show result such as name of the person etc.

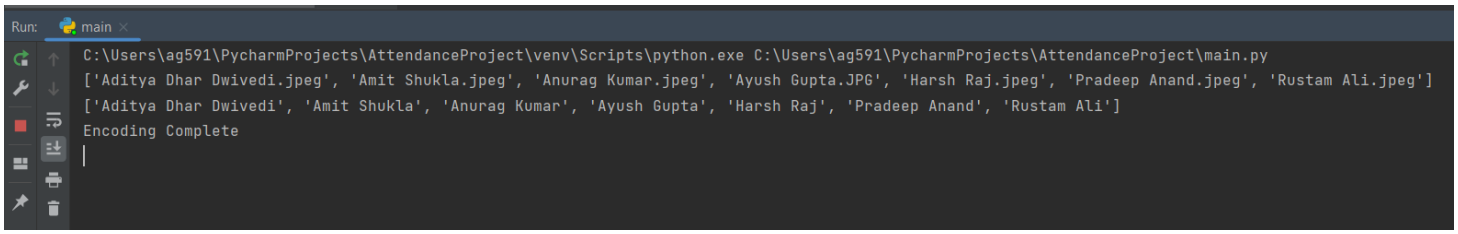



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

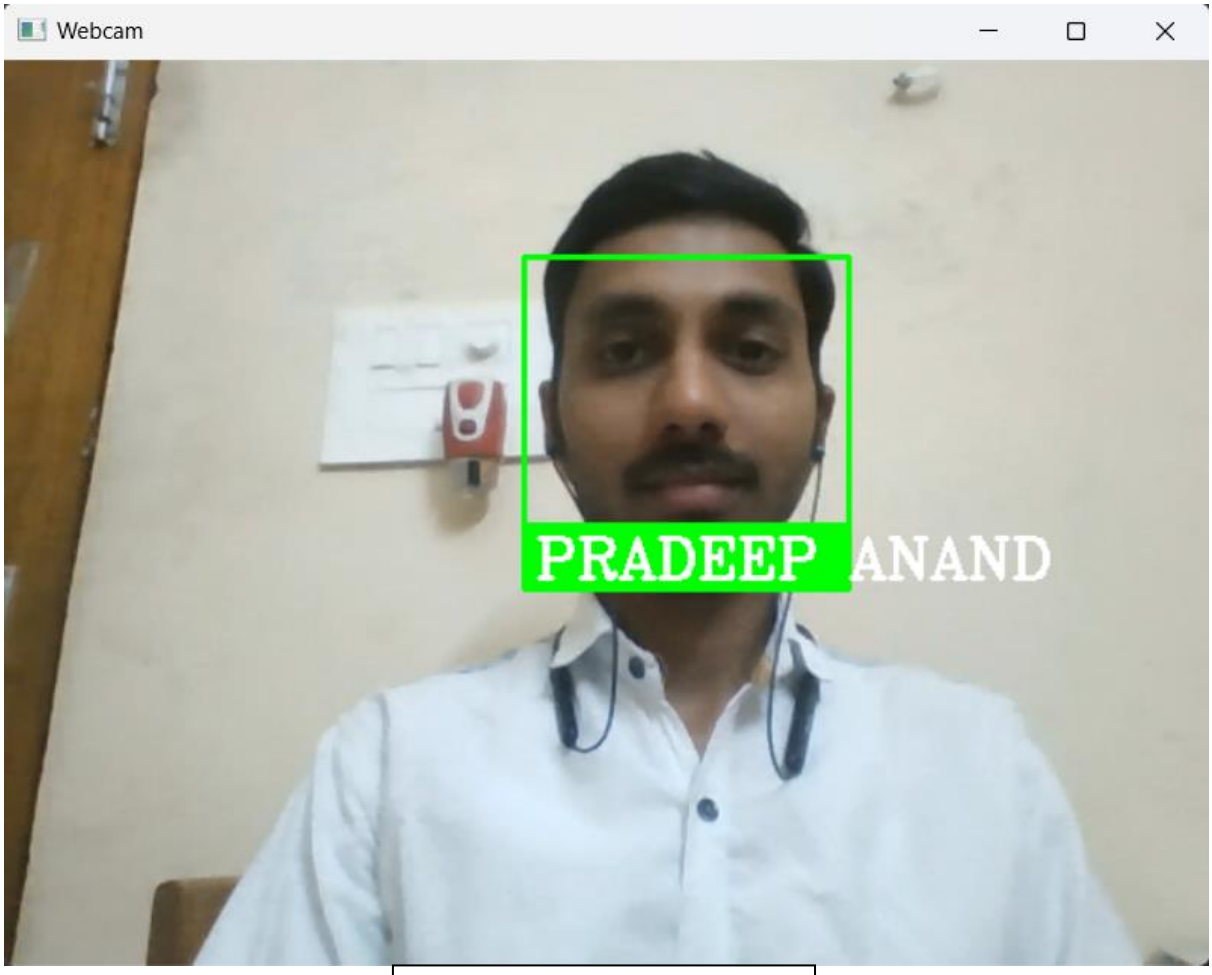
**Hand Tracking Points**

# Results/Output of the Project

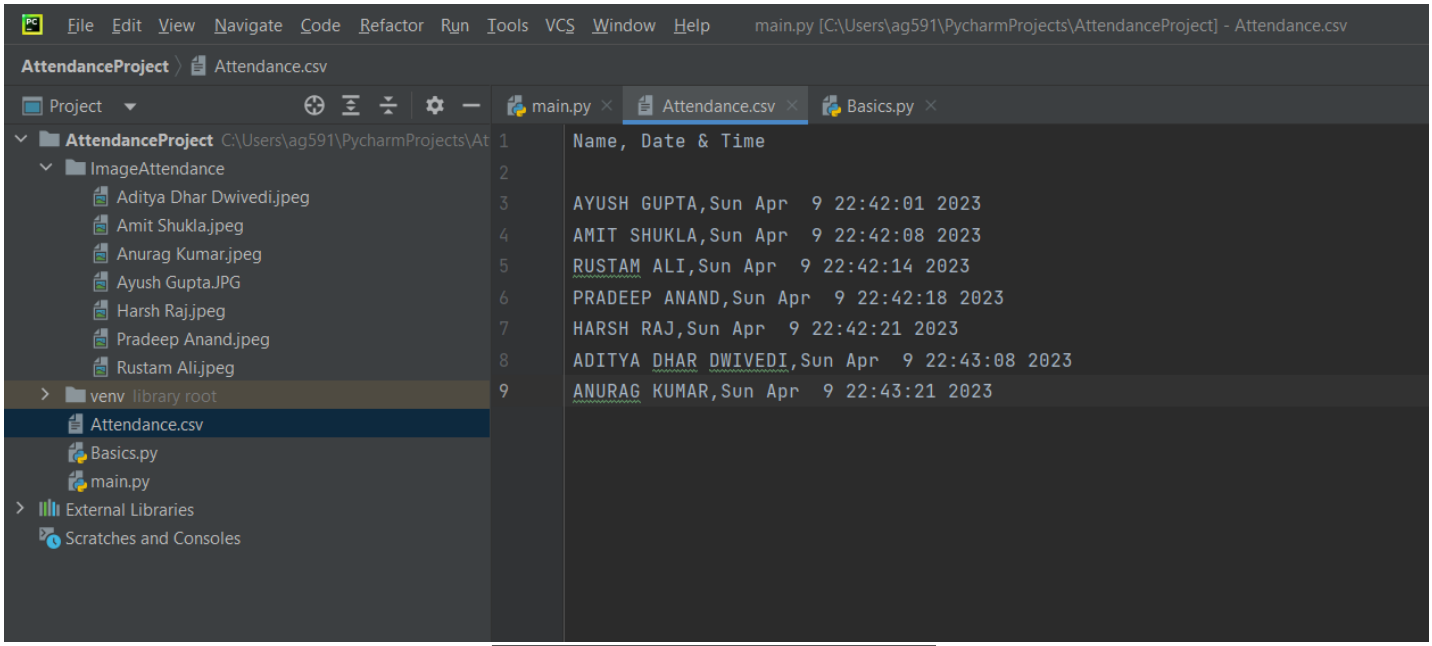
## a. Smart Attendance Management System



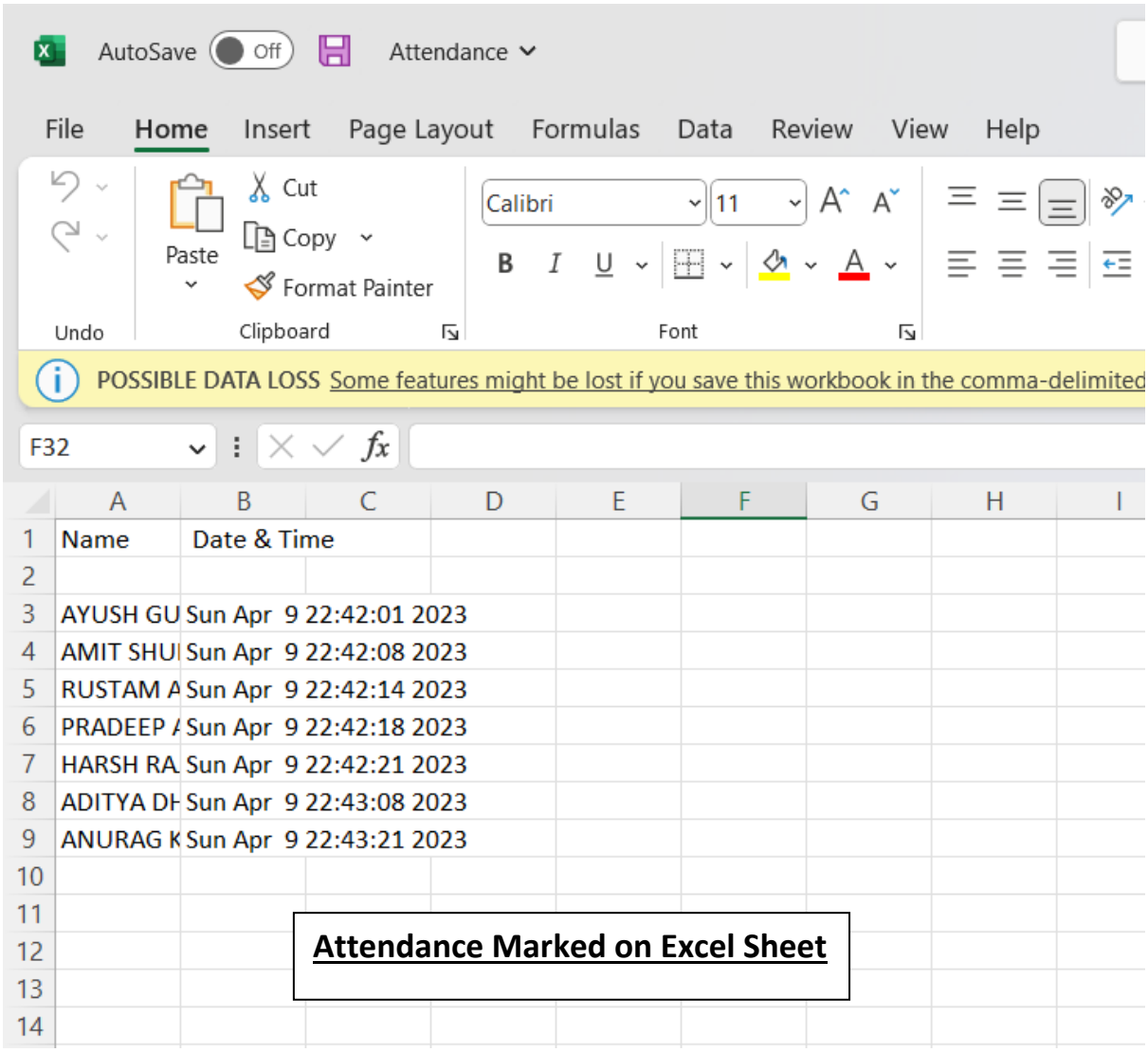

**Database Encoding**




**WebCam Detecting Face**

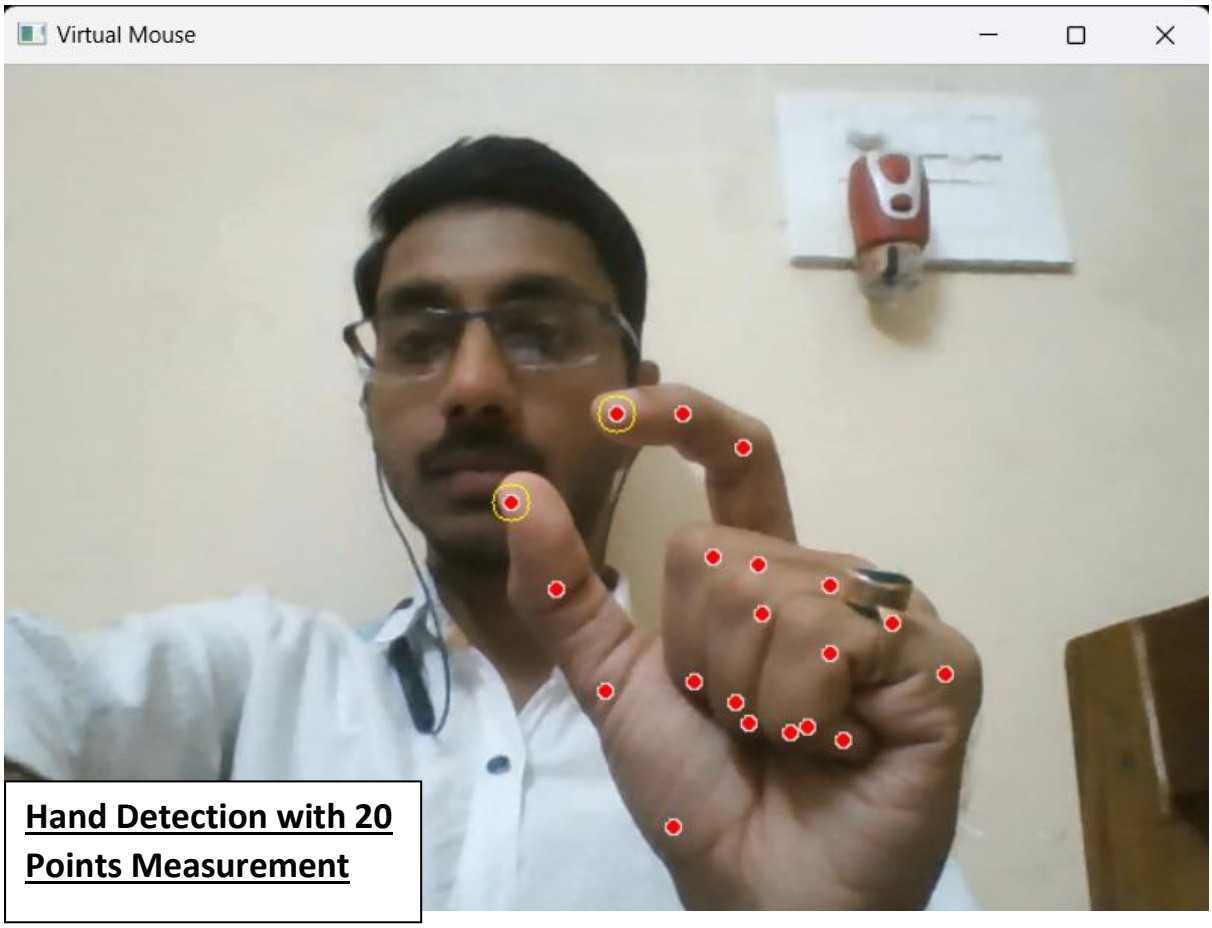



**Attendance Marked with Date & Time in .csv file**

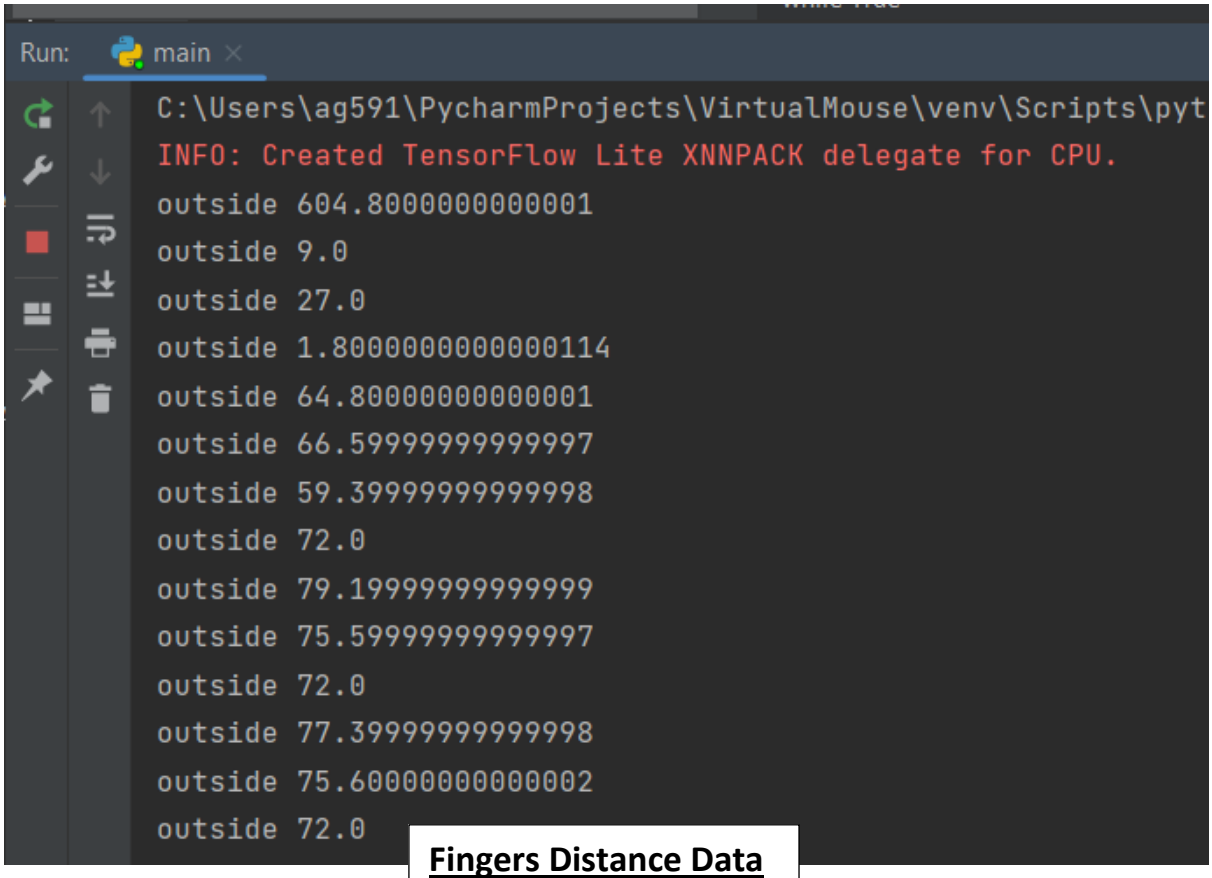



**Attendance Marked on Excel Sheet**

## b. Virtual Cursor




**Hand Detection with 20 Points Measurement**




**Fingers Distance Data**

# Future Scope of The Project

a. **Smart Attendance Management System**
   - This is a software model of the project. It can be implemented as embedded part to be install in any space.
   - More features like maintaining separate files for attendance with subject, date & time etc. can be inserted in future.
   - This project doesn't serve much more user experience as it is console-based. GUI can be applied in the project.

b. **Virtual Cursor**
   - More accurate cursor movements and less-delay clicks needs to be implemented.
   - Its very hard to move cursor/pointer at the bottom of the screen. This is a bug. We need to fix it.
   - It doesn't support features like double tap to select, move multiple lines with two fingers etc. as provided by physical mouse and its drivers. We can add these features in the future.

# Conclusion

We've successfully created and tested the project. It is running smoothly and giving output as we expected. In **Face-Recognition based Attendance Management System** project, we've tried and tested many images of different person and it is easily recognizing the faces and marking the attendance on .csv file with current date and time of entry. But in some cases – like Low Light, face is not properly focused on webcam, faces upon something is there etc., it is either not recognizing the faces or recognizing faces incorrectly for a moment. But moreover, it is recognizing faces with almost 95-99% accuracy.

And in **Virtual Cursor** project, there needs more improvements. Cursor is moving smoothly with the help of fingers and touch is working. Only problem we've encountered is that cursor is not moving to the most bottom part of the screen as fingers are going out of range from the scope of webcam view. Also, sometimes, we've experienced a little lag and delay in touch response. But moreover, this is also working fine and can be used when it will get improved.

# References


- LEARN OPENCV in 3 HOURS with Python
- Cited Research Paper - Face Recognition System by **Shivam Singh** & **Prof. S. Graceline Jasmine**
- Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning – Article by Adam Geitgey
- Official GitHub Repository of Face-Recognition library
- Haar-Cascade GitHub Repository