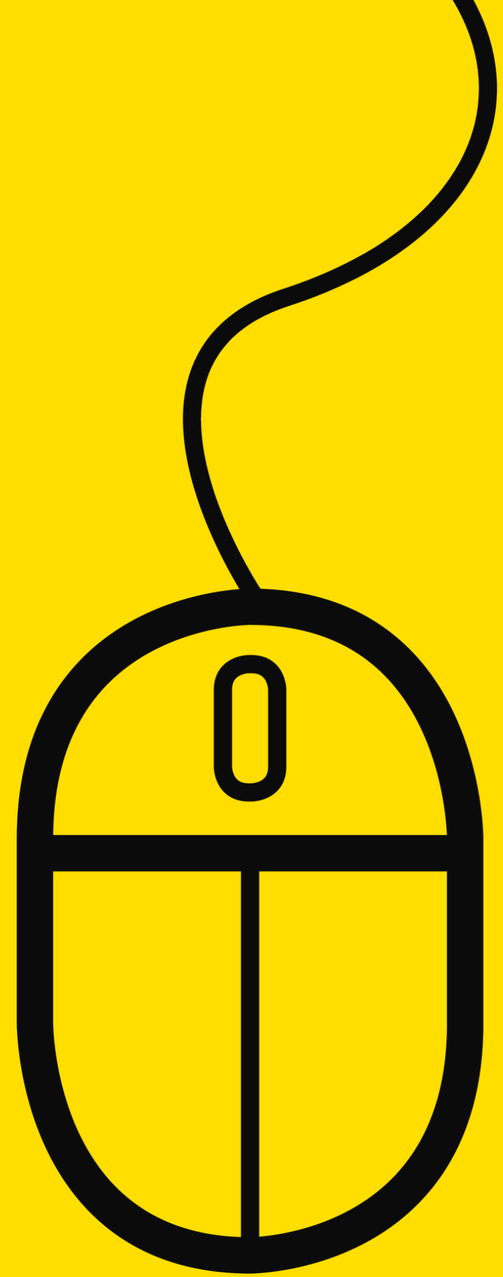
AI VIRTUAL MOUSE

312318205007 -Abirami S
312318205016- Angel M



IT8611 Mini-Project Laboratory.

Objective

Hand Gesture Recognition plays a key role in human-computer interactions. As we can see that there are so many new Technological advancements happening such as biometric authentication which we can see frequently in our smart phones, similarly hand gesture recognition is a modern way of humancomputer interaction i.e., we can control our system by showing our hands in front of webcam and hand gesture recognition can be useful for all kinds of people. Based upon this idea this project is presented. This presentation therefore upholds a detailed explanation to the algorithms and methodologies for the color detection and virtual mouse.

Motivation.

We have chosen this project with an interest of learning the direct interaction of humans with the consumer electronic devices . This takes the user experience to a whole new level. The gesture control technology would reduce our dependence on the age old peripheral devices hence it would reduce the overall complexity of the system. Initially this technology was considered in the field of gaming (like Xbox Kinect), but the application of motion/gesture control technology would be more diverse if we apply it to our other electronics like computers ,televisions, etc., for our day to day purposes like scrolling , selecting , clicking etc.

Our primary objective in doing this project was to build a device inspired from Leapmotion. So, we decided to build an introductory software implementation of the device which would eventually act as a virtual mouse

Abstract

Hand Gesture Recognition plays a key role in human-computer interactions. This application aims at providing the same. This project facilitates cursor control and volume control by means of simple navigation using the user's hand and whose motion is recorded as frames by means of video streaming. Hand gestures are indeed more efficient to bring about the underlying cause and the application is been developed over Python programming language.

LITERATURE SURVEY.

Author, Journal, Year	Title	Concept	Pros and Cons
International Research Journal of Engineering and Technology (IRJET) Nilesh Patil, Gaurav Sali, Nikhil Lokhande 2019	**Mouse on Finger Tips using ML and AI**	Focuses on the use of a Web Camera to develop a virtual human computer interaction device in the cost effective manner.	>Implementing cursor operations by means of the predefined colour pointer. >Background subtraction of the same colour pointer is not been implemented.
International Journal of Innovative Research in Technology Surajkumar S. Bele1 , Mayur R. Girsawale2 , Tejaswini G. Ganar3 , Monali P. Bawankar4 , Swati P. Raut5 , Ravindra R. Vaidya6 , Durgesh R. Chandurkar7 2019	**Hand Gesture Recognition with Switching and Mouse Movement Using MATLAB**	The use of object detection and image processing in MATLAB for the implementation of our proposed work proved to be practically successful and the movement of mouse cursor is achieved with a good precision accuracy.	>Can be integrated with Android devices. >Processing takes place from GUI to that of the cursour control therefore takes time in processing. >Use of colour cap may not be that efficient in case of backgrounds with the same colour.

International Journal of Advanced Research in Computer Engineering & Technology KRUTIKA JADHAV,FAHAD JAGIRDAR,SAYALI MANE,JAHANGIR SHAHABADI 2016	**VIRTUAL KEYBOARD AND VIRTUAL MOUSE**	Virtual Keyboard and Mouse system uses camera, blank paper for drawing the keyboard. This method is easy to use less expensive and even portable such as keyboard. However, the system developed is user friendly.	>Mouse system proved to be prominent by means of background detection and contour extraction. >Virtual Keyboard however implements image processing of the keyboard drawn on a paper rather can be done by gestures over a virtual keyboard image.
International Conference on Intelligent Computing & Optimization Dipankar Gupta, Emam Hossai,Mohammed Sazzad Hossain 2021	An Interactive Computer System with Gesture based Mouse and Keyboard.	interactive computer system which can operate without any physical keyboard and mouse. This system can be beneficial to everyone, especially to the paralyzed people who face difficulties to operate physical keyboard and mouse. We used computer vision so that user can type on virtual keyboard using a yellow-colored cap on his fingertip, and can also navigate to mouse controlling system by showing different number of fingers	>80% accuracy obtained when testing with a paralyzed person. >More relatable gestures of cursor control can be implemented.

Existing System.

Features and drawbacks

From the survey, it was understood that to automate the use of existing hardware mouse software programs that implement the same function are been implemented. However, there doesn't seem to have a very evident use of it in real life as people are more resorted and comfortable towards the use of the hardware components itself. The virtual mouse and keyboard that have been published either as journals or research papers also do not find use in the physical environment due to its drawbacks that are mentioned in the next slide.

DRAWBACKS OF EXISTING SYSTEM.

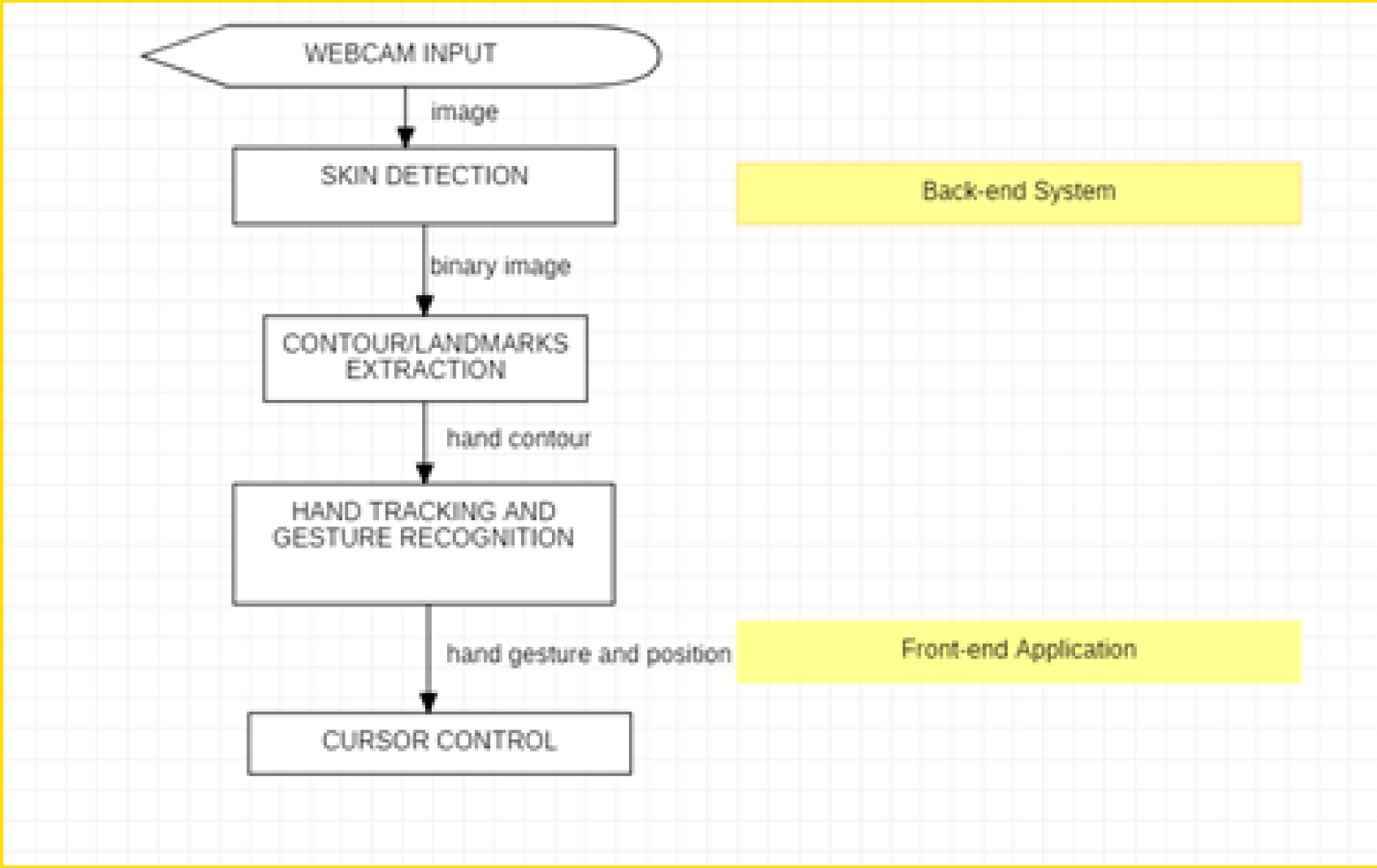
- There will always be limitations of the mouse as the mouse is a hardware input device and there can be some problems like mouse click not functioning properly.
- the mouse is a hardware device like any other physical object even the mouse will have a durability time within which is functional and after its durability time, we have to change the mouse.
- High Resolution is required for tracking.
- Very low accuracy.
- Not been implemented in the physical environment.
- All the features of the mouse cannot be implemented.
- Features do vary for right and left handed users.

Proposed System.

As the technology increase everything becomes virtualized. Such as speech recognition.Thus, Speech Recognition can replace keyboards in the future, Similarly Eye Tracking is used to control the mouse pointer with the help of our eye. Eye Tracking can replace mouse in the future.

Gestures can be in any form like hand image or pixel image or any human given pose that requires less computational difficulty or power for making the devices required for the recognitions to make work.
Different techniques are being proposed by the companies for gaining necessary information/data for recognition handmade gestures recognition models. Some models work with special devices such as data glove devices and color caps to develop a complex information about gesture provided by the user/human.

ARCHITECTURE DIAGRAM.



Modules.

1. Hand Detection
2. Hand Contour Extraction
3. Hand Tracking
4. Gesture Recognition
5. Cursor and Volume Control

Hand Detection

RGB Image is obtained by means of video streaming where each and every frame is taken into consideration.

Hand Contour Extraction

Locating and examining the landmarks of the palm raised.

Hand Tracking

Keeping a record of which finger is raised and tracing the movements of the hand from frame to frame.

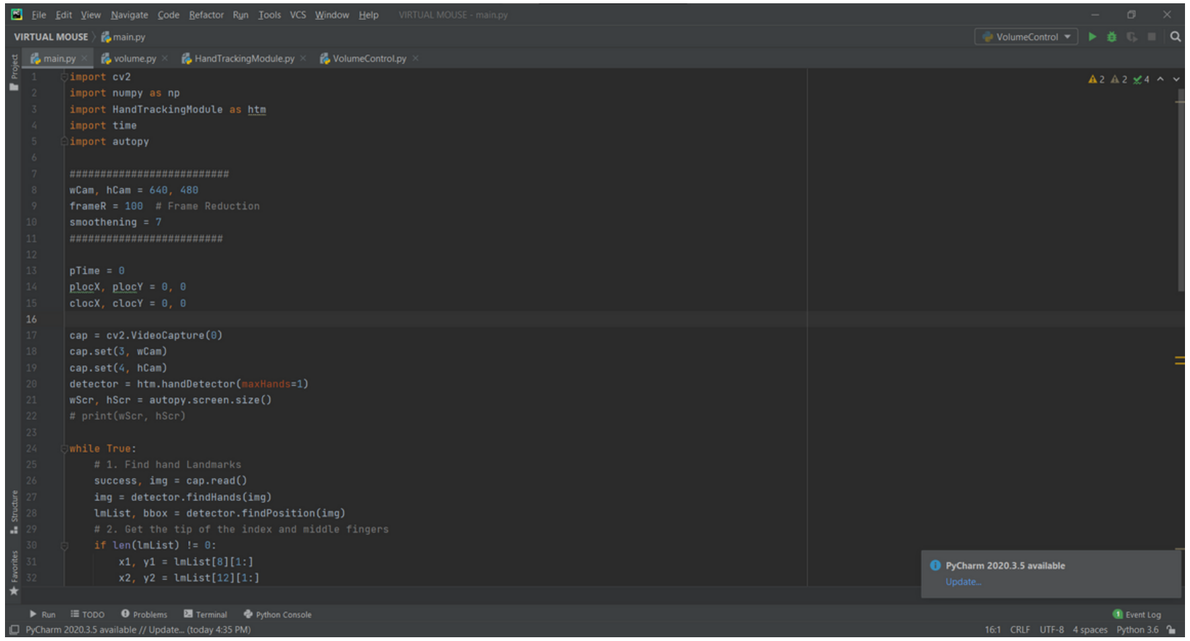
Gesture Recognition

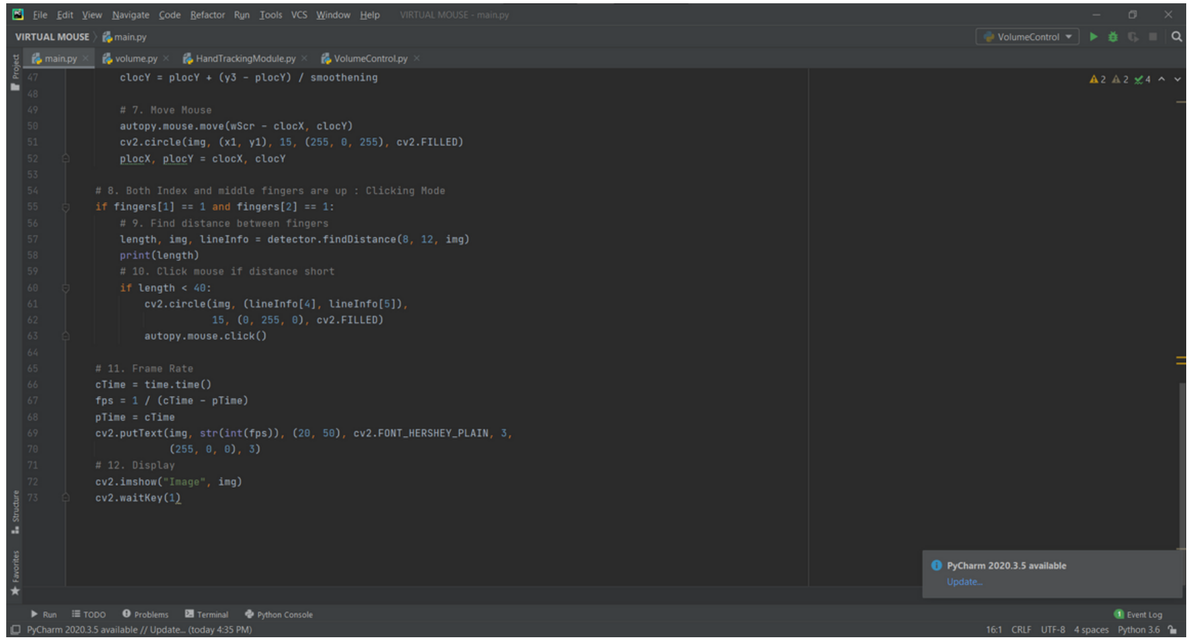
The movement obtained from tracing the frames is examined as per backend information to find the kind of gesture.

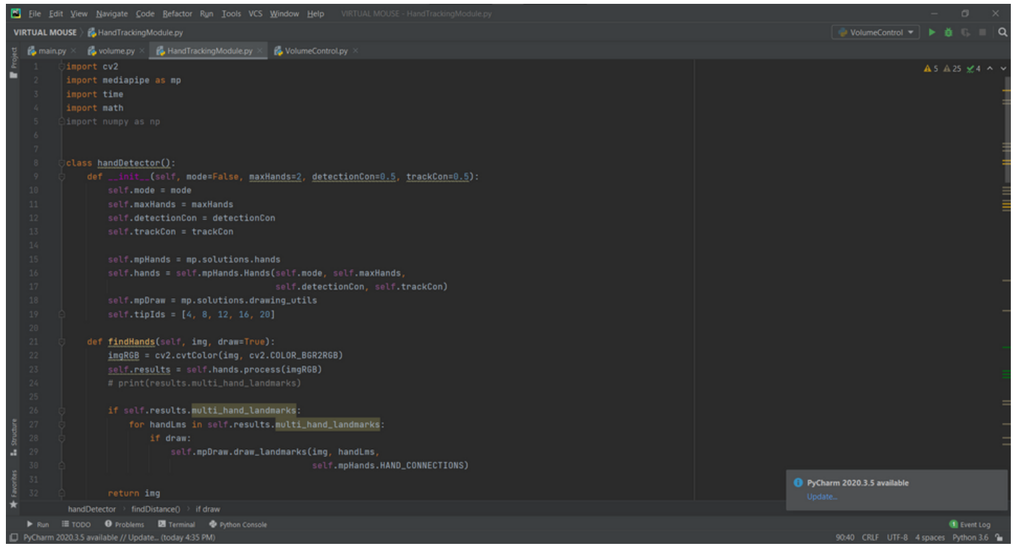
Cursour and Volume Control.

As per gesture, the cursor is moved or volume is changed accordingly.

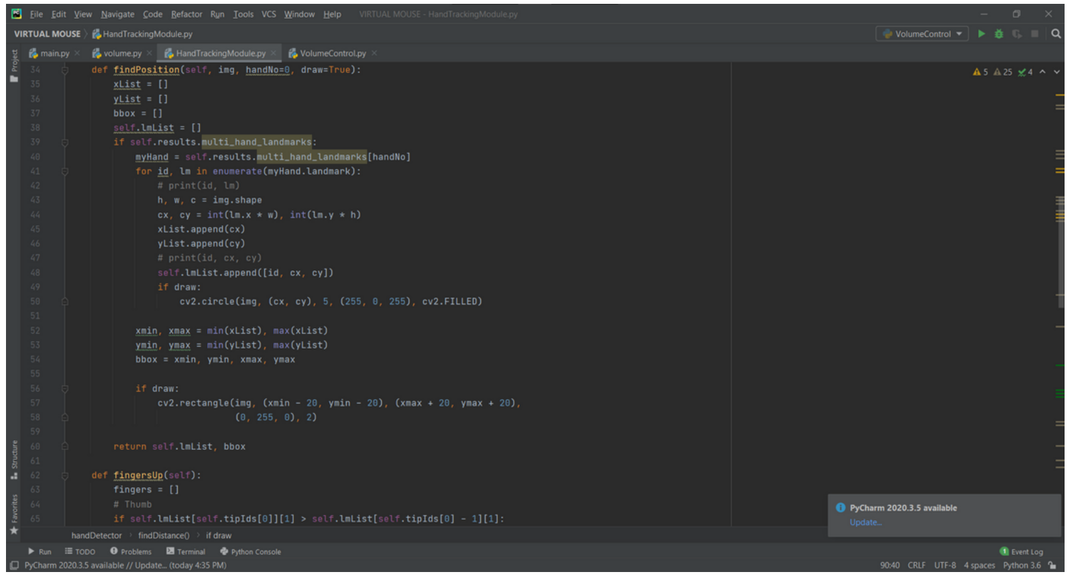
Code.


File Edit View Navigate Code Refactor Run Tools VCS Window Help VIRTUAL MOUSE - main.py
VIRTUAL MOUSE main.py VolumeControl
main.py volume.py HandTrackingModule.py VolumeControl.py
import cv2
import numpy as np
import HandTrackingModule as htm
import time
import autopy
#########################
wCam, hCam = 640, 480
frameR = 100 # Frame Reduction
smoothening = 7
#########################
pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0
cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
detector = htm.handDetector(maxHands=1)
wScr, hScr = autopy.screen.size()
# print(wScr, hScr)
while True:
# 1. Find hand Landmarks
success, img = cap.read()
img = detector.findHands(img)
lmList, bbox = detector.findPosition(img)
# 2. Get the tip of the index and middle fingers
if len(lmList) != 0:
x1, y1 = lmList[8][1:]
x2, y2 = lmList[12][1:]
PyCharm 2020.3.5 available
Update...
Run TODO Problems Terminal Python Console Event Log
PyCharm 2020.3.5 available // Update... (today 4:35 PM)
16:1 CRLF UTF-8 4 spaces Python 3.6

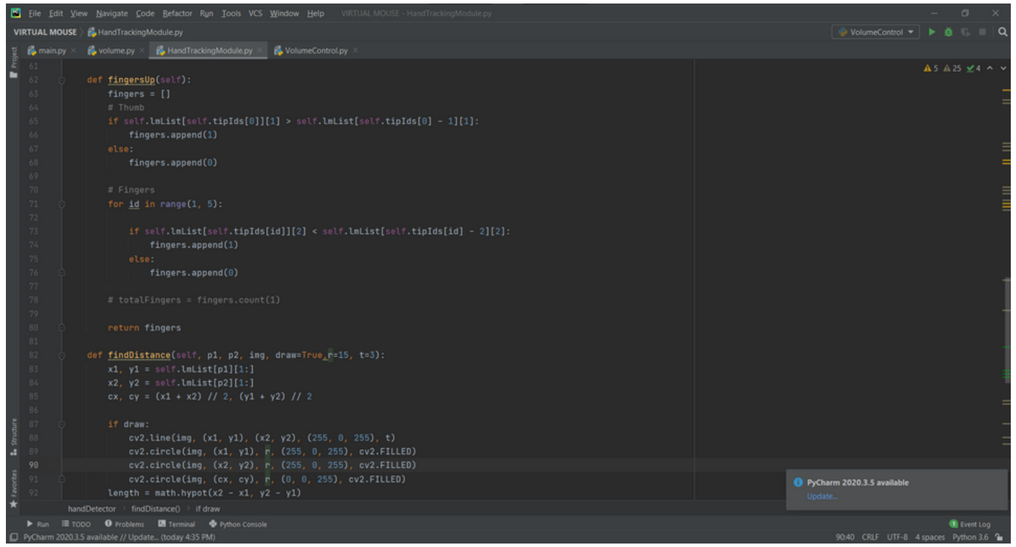

File Edit View Navigate Code Refactor Run Tools VCS Window Help VIRTUAL MOUSE - main.py
VIRTUAL MOUSE main.py VolumeControl
main.py volume.py HandTrackingModule.py VolumeControl.py
clocY = plocY + (y3 - plocY) / smoothening
# 7. Move Mouse
autopy.mouse.move(wScr - clocX, clocY)
cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
plocX, plocY = clocX, clocY
# 8. Both Index and middle fingers are up : Clicking Mode
if fingers[1] == 1 and fingers[2] == 1:
# 9. Find distance between fingers
length, img, lineInfo = detector.findDistance(8, 12, img)
print(length)
# 10. Click mouse if distance short
if length < 40:
cv2.circle(img, (lineInfo[4], lineInfo[5]),
15, (0, 255, 0), cv2.FILLED)
autopy.mouse.click()
# 11. Frame Rate
cTime = time.time()
fps = 1 / (cTime - pTime)
pTime = cTime
cv2.putText(img, str(int(fps)), (20, 50), cv2.FONT_HERSHEY_PLAIN, 3,
(255, 0, 0), 3)
# 12. Display
cv2.imshow("Image", img)
cv2.waitKey(1)
PyCharm 2020.3.5 available
Update...
Run TODO Problems Terminal Python Console Event Log
PyCharm 2020.3.5 available // Update... (today 4:35 PM)
16:1 CRLF UTF-8 4 spaces Python 3.6



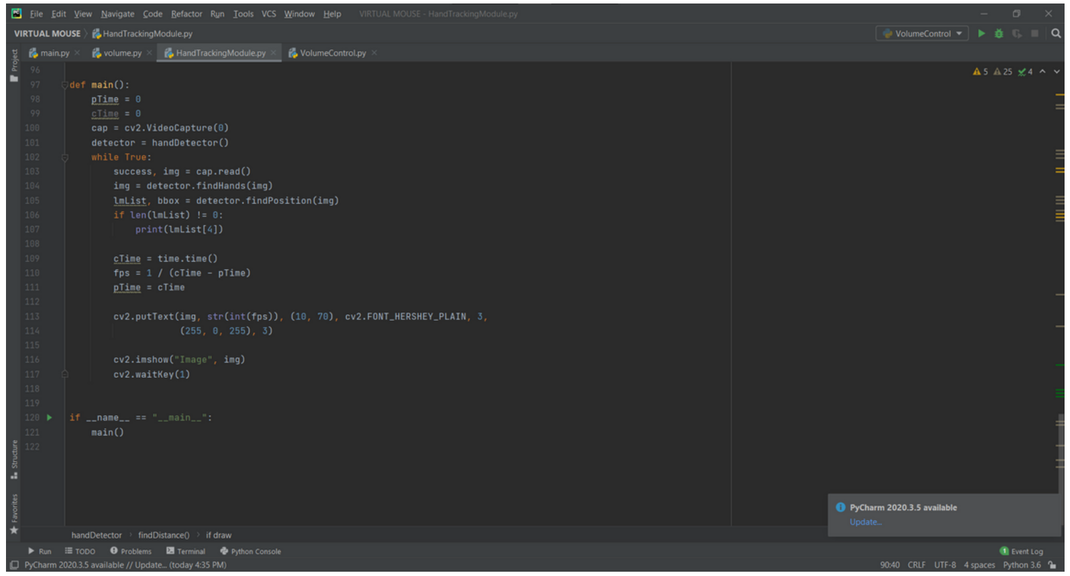
```python
import cv2
import mediapipe as mp
import time
import math
import numpy as np


class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detectionCon, self.trackCon)

        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        # print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img
```



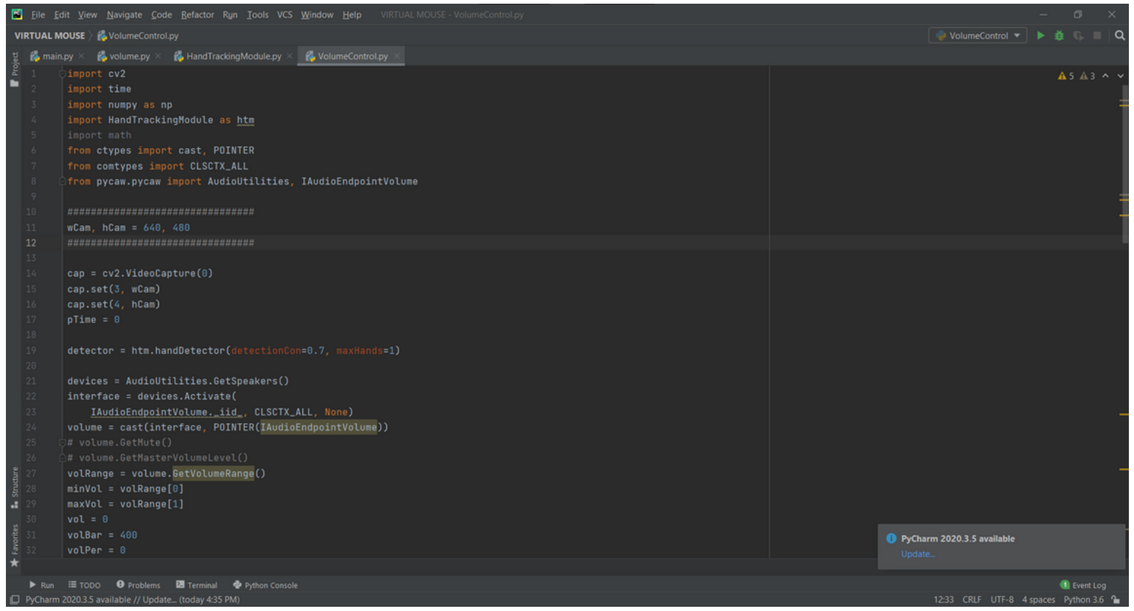
```python
    def findPosition(self, img, handNo=0, draw=True):
        xList = []
        yList = []
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                # print(id, lm)
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)
                # print(id, cx, cy)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
                              (0, 255, 0), 2)

        return self.lmList, bbox

    def fingersUp(self):
        fingers = []
        # Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
```



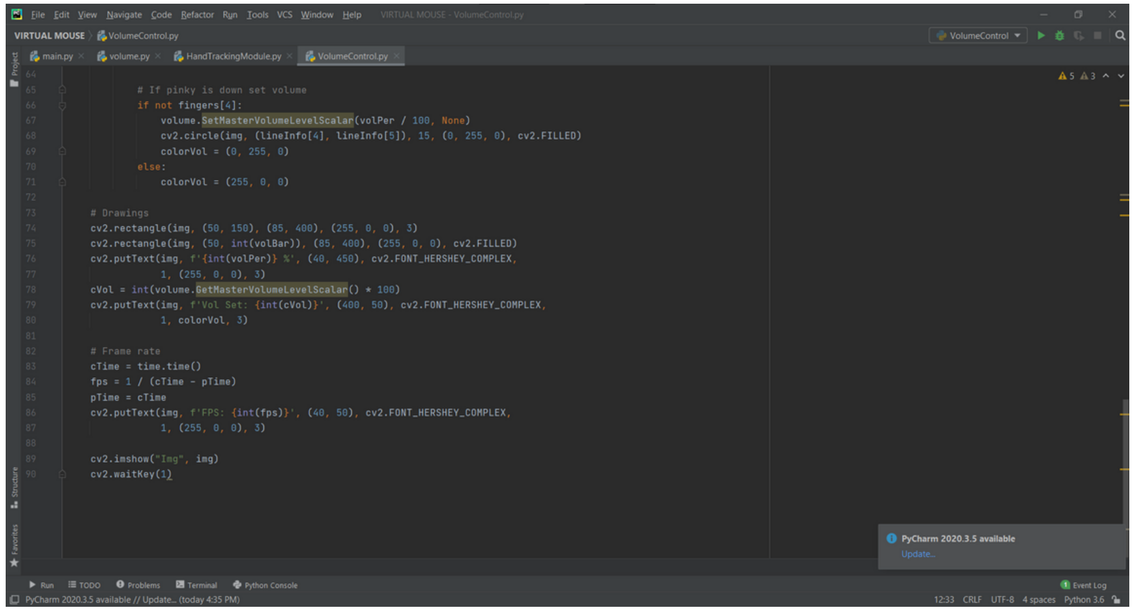
```python
    def fingersUp(self):
        fingers = []
        # Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)

        # Fingers
        for id in range(1, 5):

            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)

        # totalFingers = fingers.count(1)

        return fingers

    def findDistance(self, p1, p2, img, draw=True,r=15, t=3):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)
```



```python
def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)


if __name__ == "__main__":
    main()
```



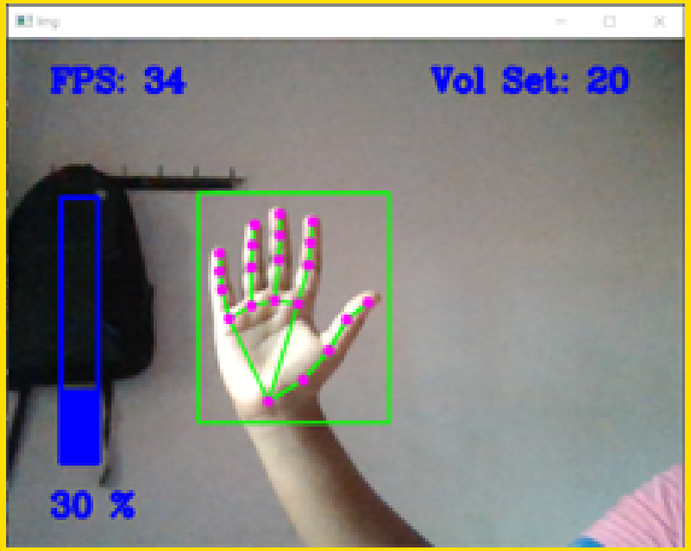
```python
import cv2
import time
import numpy as np
import HandTrackingModule as htm
import math
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

################################
wCam, hCam = 640, 480
################################

cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
pTime = 0

detector = htm.handDetector(detectionCon=0.7, maxHands=1)

devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(
    IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))
# volume.GetMute()
# volume.GetMasterVolumeLevel()
volRange = volume.GetVolumeRange()
minVol = volRange[0]
maxVol = volRange[1]
vol = 0
volBar = 400
volPer = 0
```

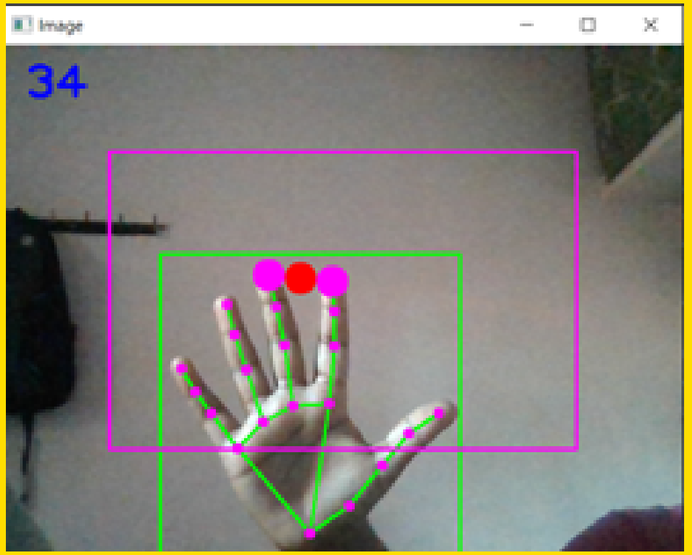


```python
            # If pinky is down set volume
            if not fingers[4]:
                volume.SetMasterVolumeLevelScalar(volPer / 100, None)
                cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0), cv2.FILLED)
                colorVol = (0, 255, 0)
            else:
                colorVol = (255, 0, 0)

    # Drawings
    cv2.rectangle(img, (50, 150), (85, 400), (255, 0, 0), 3)
    cv2.rectangle(img, (50, int(volBar)), (85, 400), (255, 0, 0), cv2.FILLED)
    cv2.putText(img, f'{int(volPer)} %', (40, 450), cv2.FONT_HERSHEY_COMPLEX,
                1, (255, 0, 0), 3)
    cVol = int(volume.GetMasterVolumeLevelScalar() * 100)
    cv2.putText(img, f'Vol Set: {int(cVol)}', (400, 50), cv2.FONT_HERSHEY_COMPLEX,
                1, colorVol, 3)

    # Frame rate
    cTime = time.time()
    fps = 1 / (cTime - pTime)
    pTime = cTime
    cv2.putText(img, f'FPS: {int(fps)}', (40, 50), cv2.FONT_HERSHEY_COMPLEX,
                1, (255, 0, 0), 3)

    cv2.imshow("Img", img)
    cv2.waitKey(1)
```

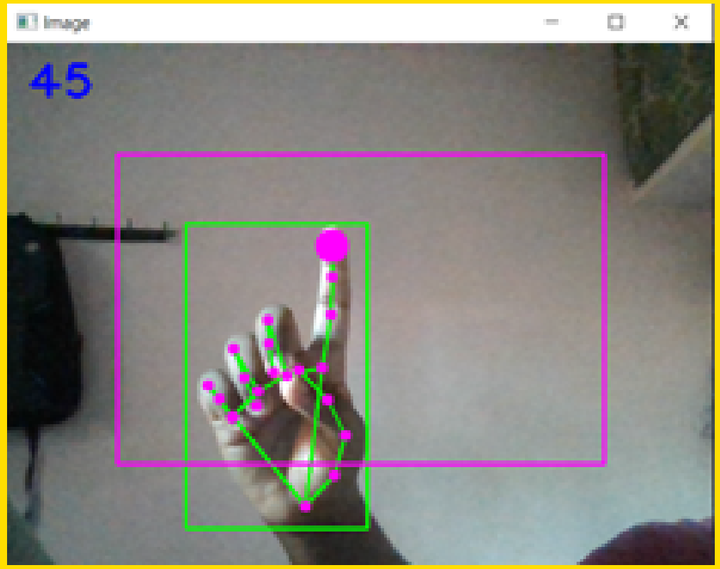
Ouputs.

Identifying Landmarks.

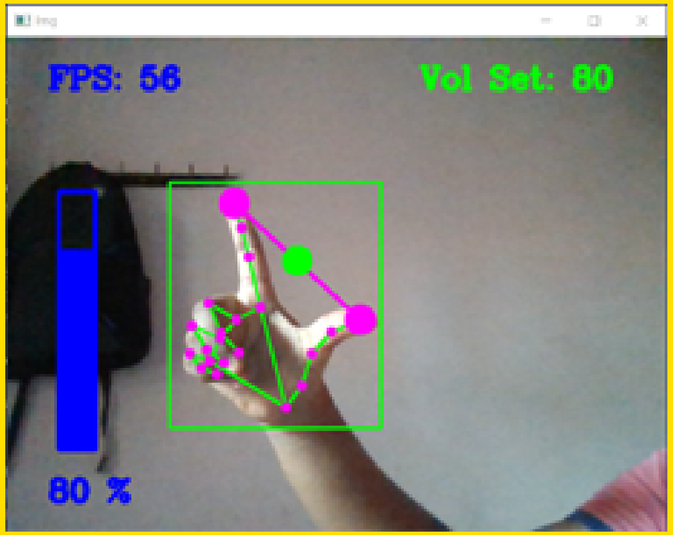




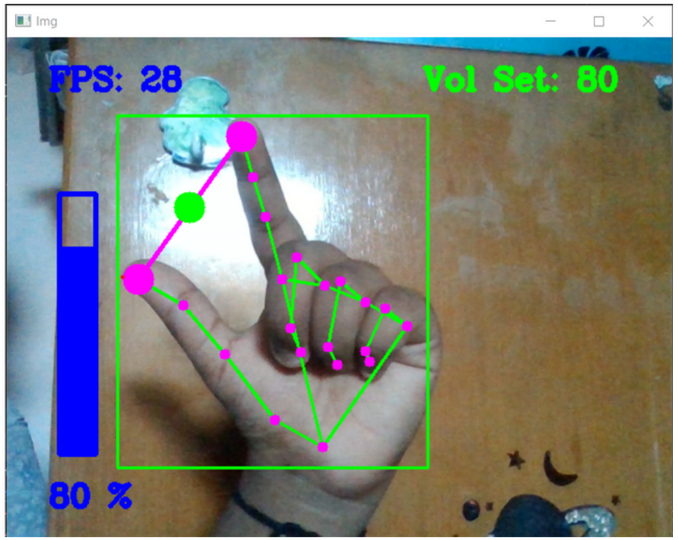
Navigating using index finger.

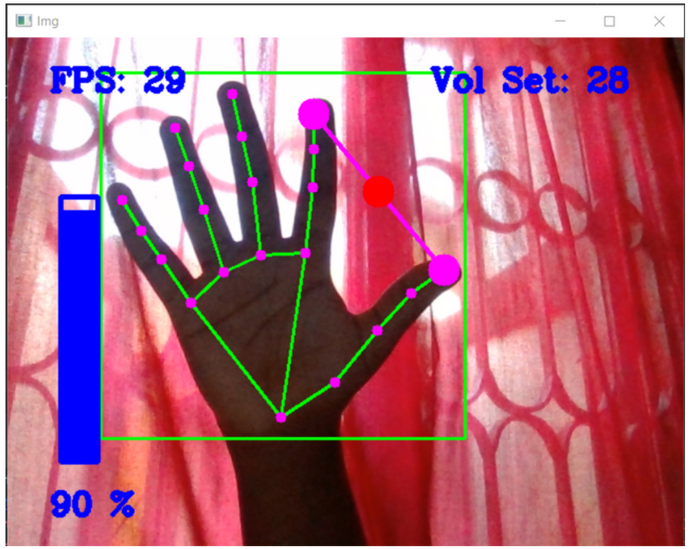


Adjusting Volume.

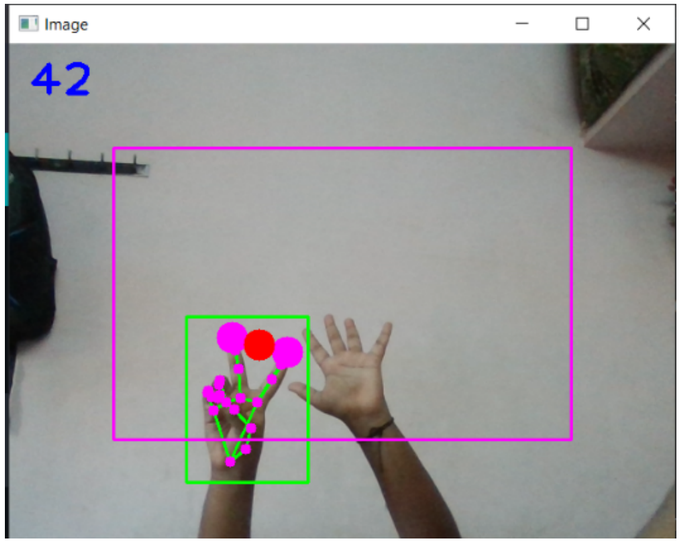


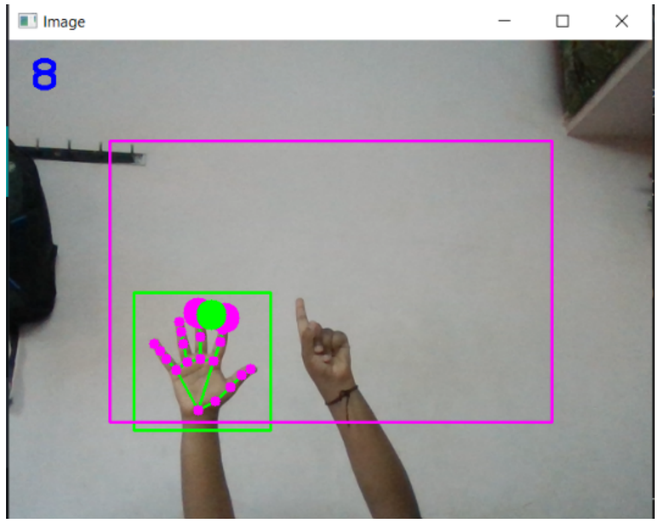
Detecting Hands (Implementing Background Subtraction)





Detecting Hands (In case of Multiple Hands)





Advantages of Proposed System and Scope

- Virtual Mouse using Hand gesture recognition allows users to control mouse with the help of hand gestures.
- The system’s webcam is used for tracking hand gestures.
- Computer vision techniques are used for gesture recognition. OpenCV consists of a package called video capture which is used to capture data from a live video.
- Main thing we need to identify is the applications , the model is going to develop so the development of the mouse movement without using the system mouse
- More reasonable and relatable gestures can be implemented using the Python language.
- Virtual Mouse can also be developed using CNN.
- Advancement can be made by replcing hand gestures with that of eye ball movement.

References

i. Autopy Mouse control- https://www.autopy.org/documentation/api-reference/mouse.html
ii. OpenCV Website – https://opencv.org/
iii. Hand Recognition- https://www.analyticsvidhya.com/blog/2021/05/hand-gesture-recognition-using-colour-based-technology/
iv. Pycaw Audio Control- https://pypi.org/project/pycaw/

THANK YOU.