PYTHON PROJECT

Presented by:

ABHIKALP SRIVASTAVA

B521002

C.S.E.

ABSTRACT

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 human computer 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 paper is presented. This paper provides a detailed explanation to the algorithms and methodologies for the colour detection and virtual mouse.

INTRODUCTION

In this high-tech world, it is almost impossible to imagine life without computers. The invention of computers is one of the greatest humankind inventions. Computers have become an essential part of almost every day use for individuals of every age. In daily life, we interact many times with computers to make our work easier. Thus, Human-Computer Interaction (HCI) has become a hot topic for research. In our daily life, vision and gestures are important approaches for communication among human beings, and the same role is played by the mouse in Graphical User Interface (GUI) based computers. So, a combined methodology can be used to make a better interactive system for Human-Computer Interaction. Computer vision techniques can be an alternative way for the touch screen and create a virtual human-computer interaction device using a webcam. In this project, a finger tracking-based virtual mouse application will be designed and implemented using a regular webcam. To implement this, we will be using the object tracking concept of Artificial Intelligence and the OpenCV module of Python.

SCOPE

Eye tracking which 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 require 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 colour caps to develop complex information about gesture provided by the user/human.

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 human computer 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 paper is presented. This paper provides a detailed explanation to the algorithms and methodologies for the colour detection and virtual mouse

PURPOSE

As the technology increase everything becomes virtualized such as speech recognition. Speech Recognition is used for recognition and translation of the spoken language into text. Thus, Speech Recognition can replace keyboards in the future, Similarly Eye Tracking which 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 require 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 complex information about gesture provided by the user/human

OVERVIEW OF THE SYSTEM

Existing System

A Computer Mouse is an input device that helps to point and to interact with whatever that is being pointed. There are so many types of mouse in the current trend, there's the mechanical mouse that consists of a single rubber ball which can rotate in any direction and the movement of the pointer is determined by the motion of that rubber ball. Later the mechanical mouse is replaced by the Optical Mouse. Optical Mouse consists of a led sensor to detect the movement of the pointer. Years Later the laser mouse was introduced to improve the accuracy and to overcome the drawbacks of the Optical Mouse. Later as the Technology has been increased drastically wireless mouse was introduced so as to enable hassle free movement of the mouse and to improve the accuracy. No Matter how much the accuracy of the mouse increases but 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 ad etc., as 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

Disadvantages

- 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.

Proposed System

As the technology increase everything becomes virtualized. Such as speech recognition, Speech Recognition is used for recognition and translation of the spoken language into text. Thus, Speech Recognition can replace keyboards in the future, Similarly Eye Tracking which 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 require 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 colour caps to develop a complex information about gesture provided by the user/human.

Advantages

- Virtual Mouse using Hand gesture recognition allows users to control mouse with the help of hand gestures.
- 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 are the applications the model is going to develop so the development of the mouse movement without using the system mouse

Modules: Implementation

- Collection information.
- Checking devices (like webcam) working properly or not.
- Collection tape or finger ribbon, which should be fit to the fingers.
- Import packages like NumPy, OpenCV, Autopy, Mediapipe.
- Implement the Open Gesture Operation.

ALGORITHM USED FOR HAND DETECTION

For High-fidelity hand and finger tracking MediaPipe library (on open-source cross-platform framework) and OpenCV library for implementing computer vision are being used. This algorithm utilizes machine learning concepts to track and detect hand gestures and finger tips.

MEDIAPIPE

The MediaPipe framework is used by the developer for building and analyzing the systems through graphs, and it also been used for developing the systems for the application purpose. MediaPipe library is used by developers for architecting and analyzing various models through graphs and many of these have been used for building applications. MediaPipe Hands utilizes an ML pipeline consisting of multiple models working together. The model that incorporated MediaPipe needs to work in a pipeline fashion. It consists of mainly graphs, nodes, streams and calculators. The MediaPipe framework is based on three fundamental parts; they are performance evaluation, framework for retrieving sensor data, and a collection of components which are called calculators , and they are reusable. A pipeline is a graph which consists of components called calculators, where each calculator is connected by streams in which the packets of data flow through. The pipeline is implemented as a MediaPipe graph that uses a hand landmark tracking subgraph from the hand landmark module, and renders using a dedicated hand renderer subgraph. The hand landmark tracking subgraph internally uses a hand landmark subgraph from the same module and a palm detection subgraph from the palm detection module.The calculators and streams combined create a data-flow diagram; the graph is created with MediaPipe where each node is a calculator and the nodes are connected by streams. Mediapipe offers open source cross-platform,customizable ML solutions for live and streaming media.This is helpful in many cases such as:

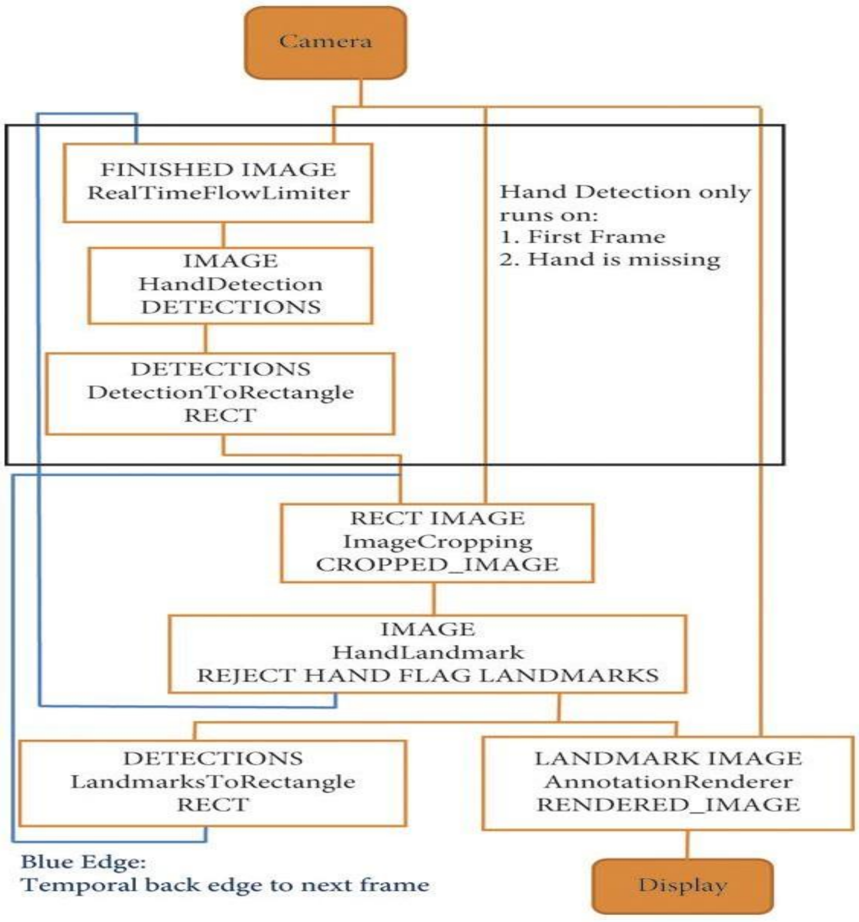
1.Selfie segmentation.

2.Face mesh

3.Human pose detection and tracking

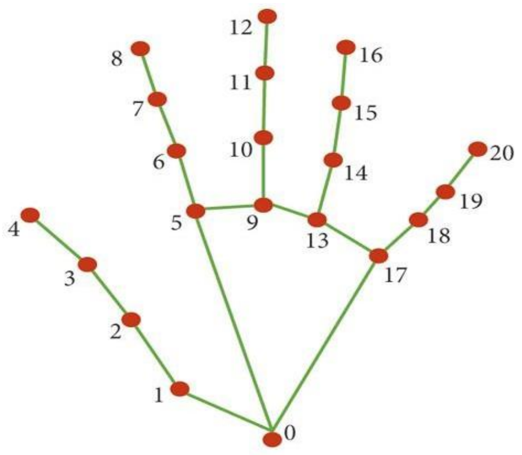
4.Holistic tracking

5.3D object detection.



OPENCV

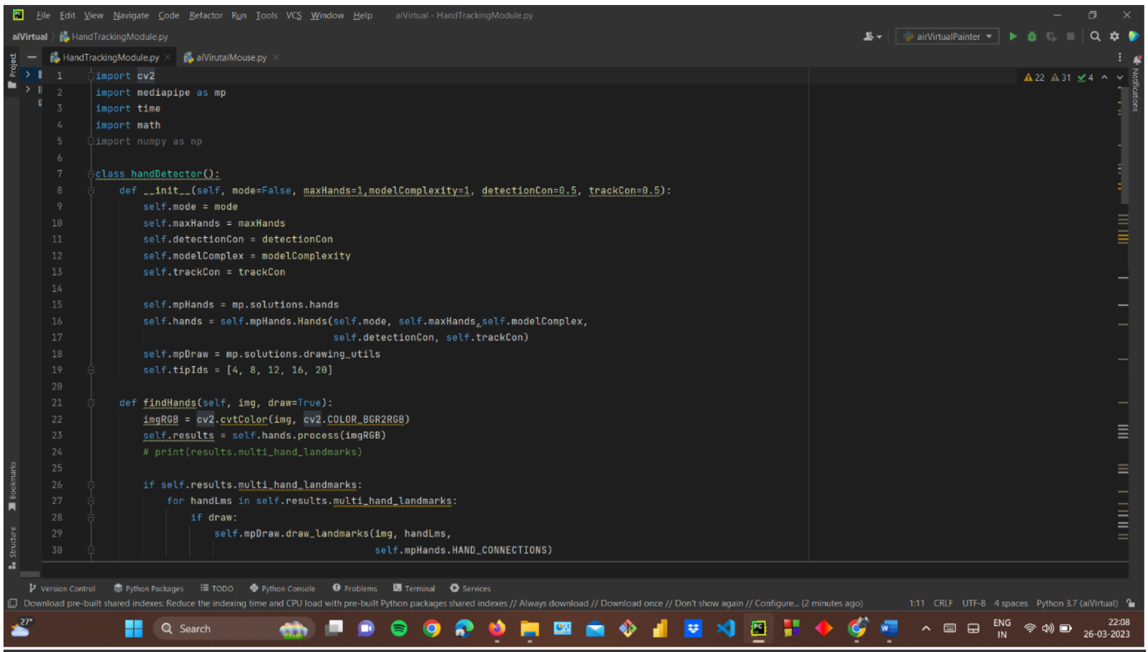
OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state of-the-art computer vision and machine learning algorithms. This library is written in the python language and it helps in the making of applications that used computer vision. In this model, OpenCV library is utilized for image and video processing and for detecting and analyzing faces and objects. The developing of hand gesture recognition using Python and OpenCV can be implemented by applying the theories of hand segmentation and the hand detection system which use the Haarcascade classifier.



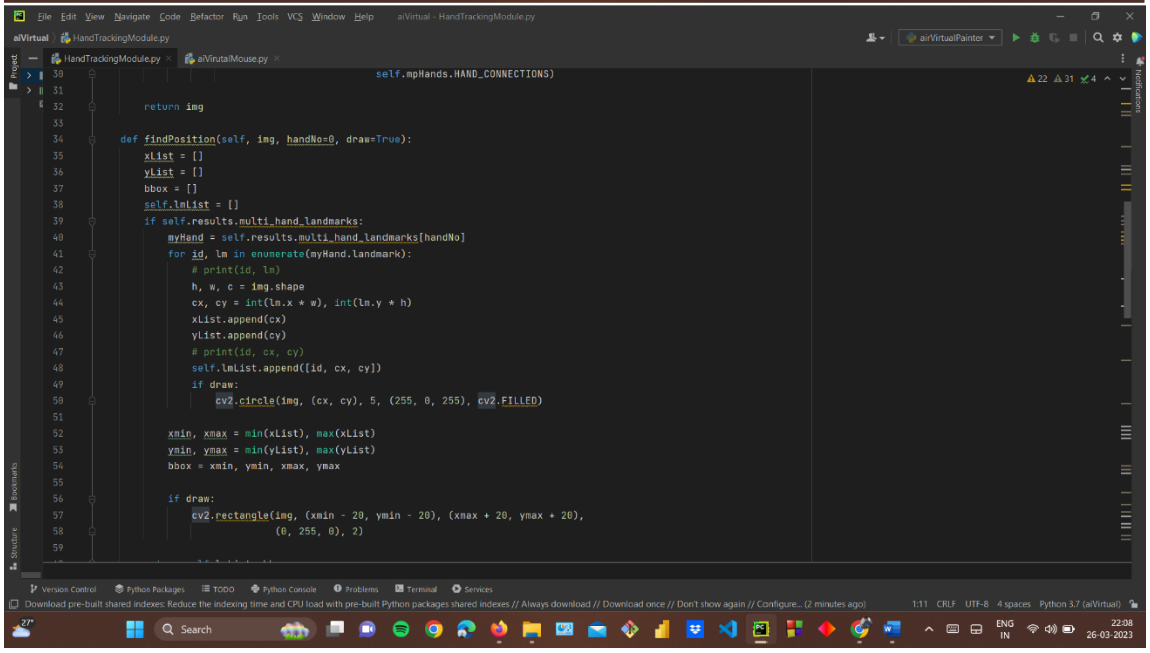
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP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

<u>CODE</u>

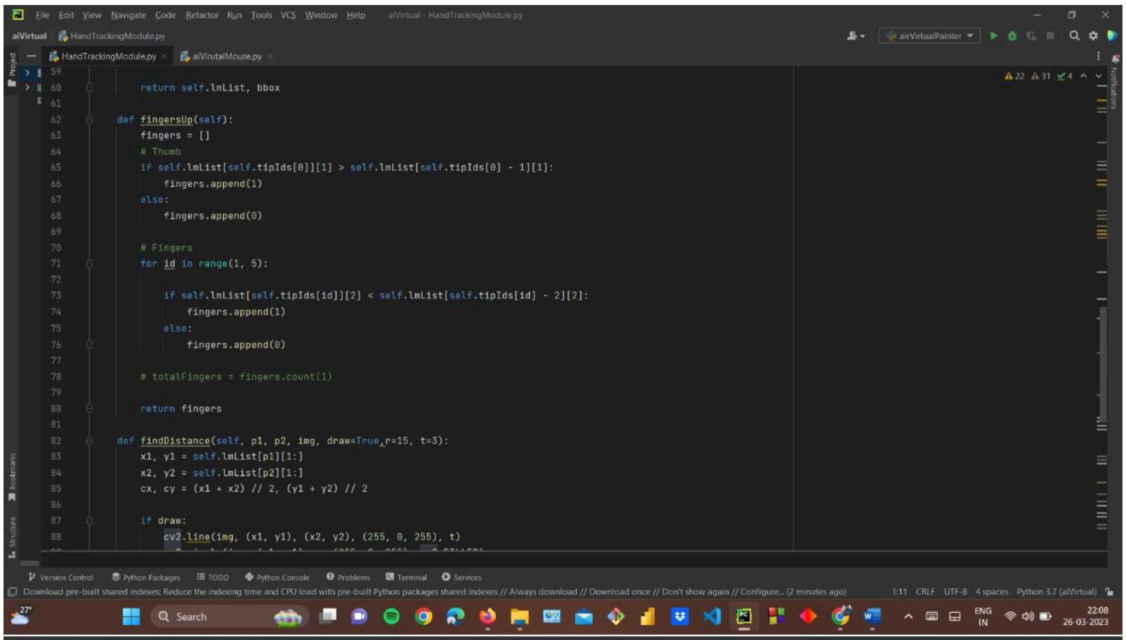
1. <u>HAND TRACKING MODULE.PY</u>



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

class handDetector():
    def __init__(self, mode=False, maxHands=1,modelComplexity=1, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.modelComplex = modelComplexity
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,self.modelComplex,
                                        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)
```

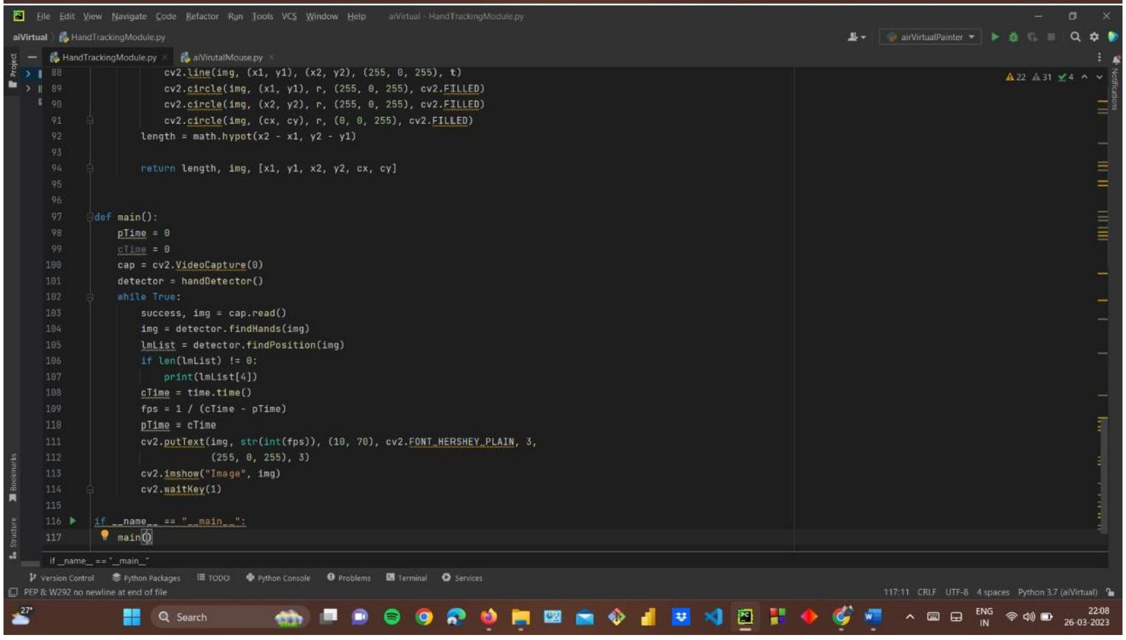


```python
                                               self.mpHands.HAND_CONNECTIONS)

        return img

    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)

```



```python
        return self.lmList, bbox

    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)
```

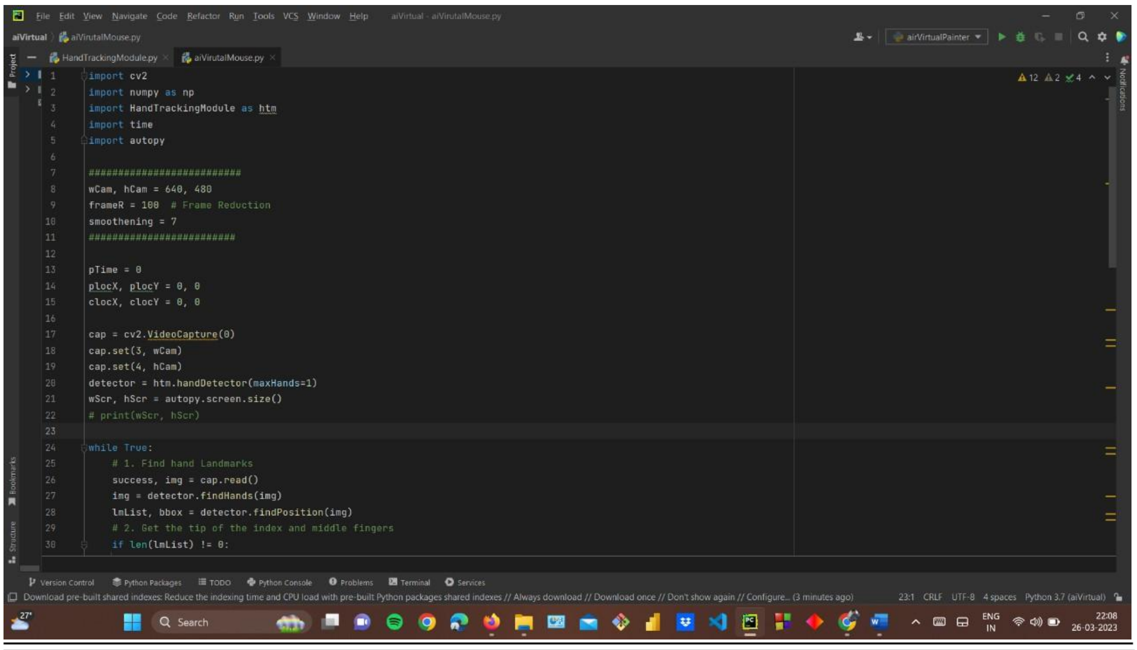
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download once // Don't show again // Configure... (2 minutes ago)



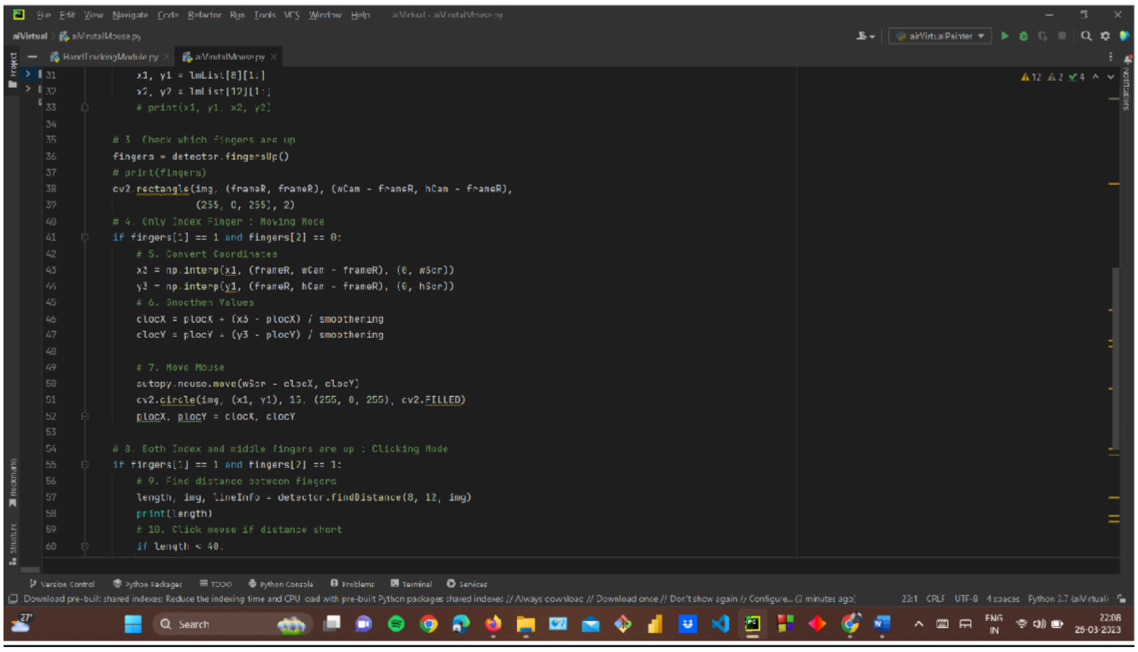
```python
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2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

        return length, img, [x1, y1, x2, y2, cx, cy]


def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList = 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()
```

PEP 8: W292 no newline at end of file

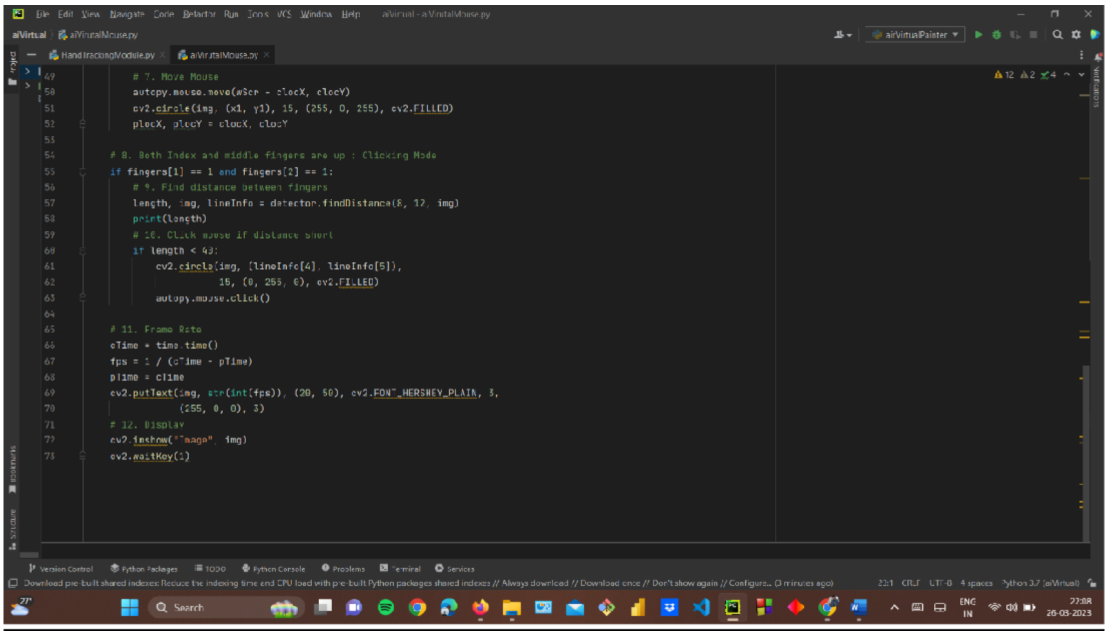
2. AiVirtualMouse.PY



```python
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:
```



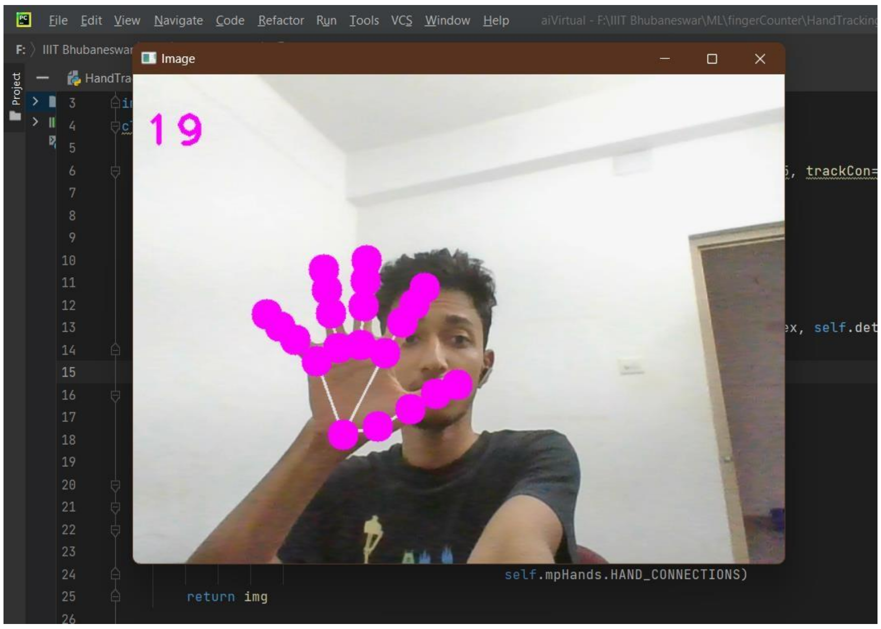
```python
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]
        # print(x1, y1, x2, y2)

    # 3. Check which fingers are up
    fingers = detector.fingersUp()
    # print(fingers)
    cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),
                  (255, 0, 255), 2)
    # 4. Only Index Finger : Moving Mode
    if fingers[1] == 1 and fingers[2] == 0:
        # 5. Convert Coordinates
        x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
        y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))
        # 6. Smoothen Values
        clocX = plocX + (x3 - plocX) / smoothening
        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:
```



```python
    # 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)
```

The Camera Used in the AI Virtual Mouse System

The proposed AI virtual mouse system is based on the frames that have been captured by the webcam in a laptop or PC. By using the Python computer vision library OpenCV, the video capture object is created and the web camera will start capturing video, as shown in Figure. The web camera captures and passes the frames to the AI virtual system.



```python
, trackCon=
ex, self.det
self.mpHands.HAND_CONNECTIONS)
return img
```

Capturing the Video and Processing

The AI virtual mouse system uses the webcam where each frame is captured till the termination of the program. The video frames are processed from BGR to RGB color space to find the hands in the video frame by frame as shown in the following code:

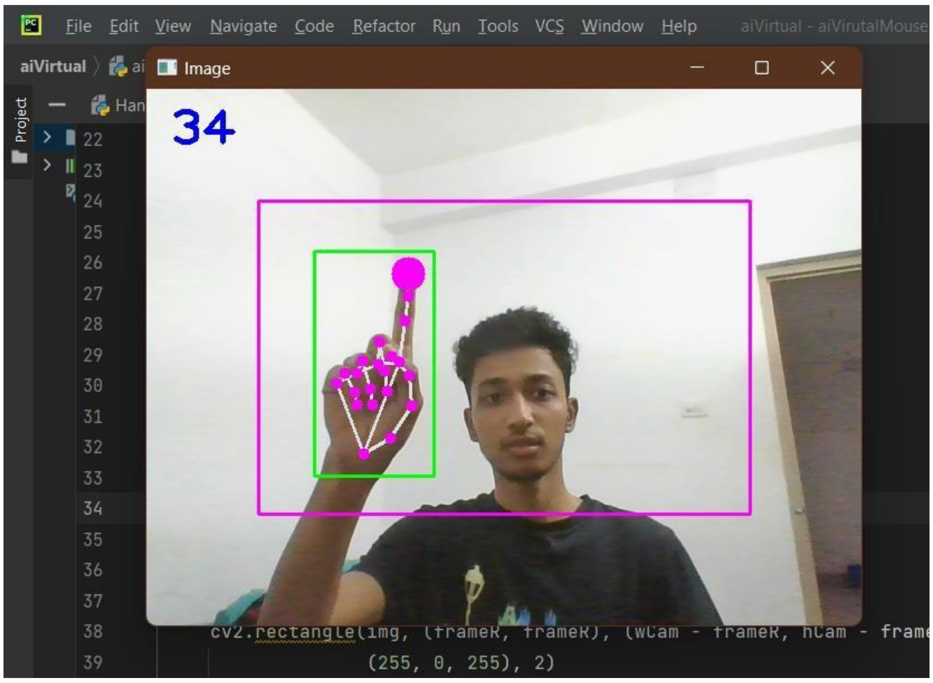
```
def findHands(self, img, draw = True):
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
self.results = self.hands.process(imgRGB)
```

(Virtual Screen Matching) Rectangular Region for Moving through the Window

The AI virtual mouse system makes use of the transformational algorithm, and it converts the co-ordinates of fingertip from the webcam screen to the computer window full screen for controlling the mouse. When the hands are detected and when we find which finger is up for performing the specific mouse function, a rectangular box is drawn with respect to the computer window in the webcam region where we move throughout the window using the mouse cursor,

Detecting Which Finger Is Up and Performing the Particular Mouse Function

In this stage, we are detecting which finger is up using the tip Id of the respective finger that we found using the MediaPipe and the respective co-ordinates of the fingers that are up, as shown, and according to that, the particular mouse function is performed.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
aiVirtual - aiVirutalMouse
aiVirtual
Image
Project
Han
34

```
cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frame
              (255, 0, 255), 2)
```

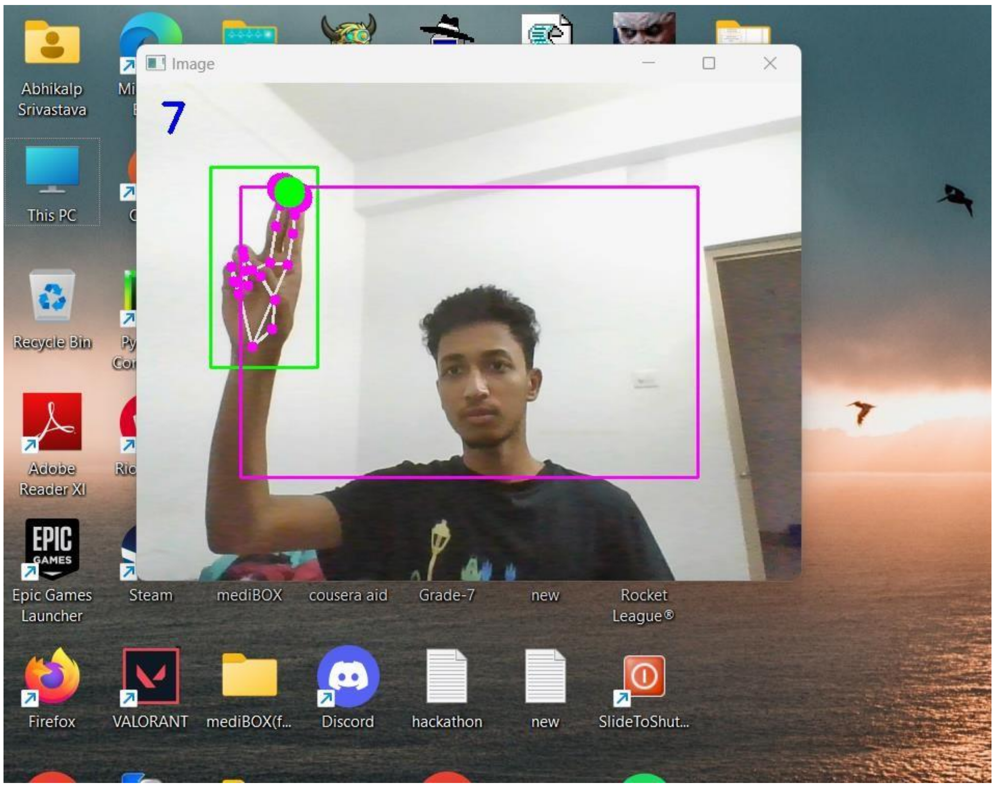
Mouse Functions Depending on the Hand Gestures and Hand Tip Detection Using Computer Vision

For the Mouse Cursor Moving around the Computer Window

If the index finger is up with tip Id = 1 or both the index finger with tip Id = 1 and the middle finger with tip Id = 2 are up, the mouse cursor is made to move around the window of the computer using the AutoPy package of Python.

For the Mouse to Perform Left Button Click

If both the index finger with tip Id = 1 and the middle finger with tip Id = 2 are up and the distance between the two fingers is lesser than 30px, the computer is made to perform the left mouse button click using the Autopy Python package

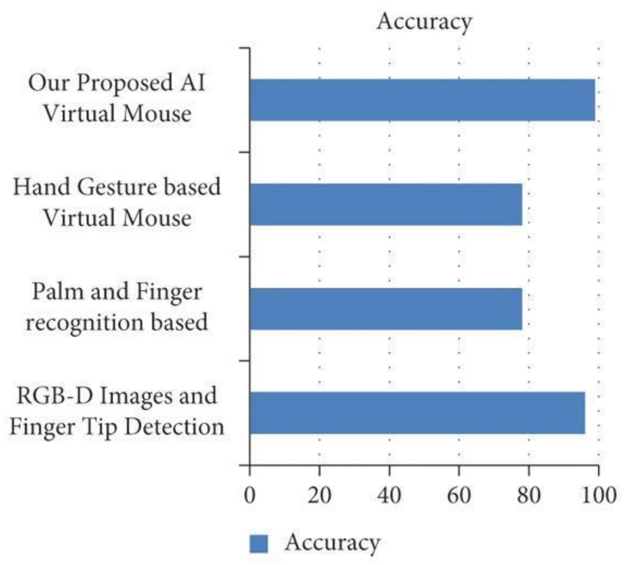


Experimental Results and Evaluation

In the proposed AI virtual mouse system, the concept of advancing the human-computer interaction using computer vision is given.

Cross comparison of the testing of the AI virtual mouse system is difficult because only limited numbers of datasets are available. The hand gestures and finger tip detection have been tested in various illumination conditions and also been tested with different distances from the webcam for tracking of the hand gesture and hand tip detection.

From Table , it is evident that the proposed AI virtual mouse has performed very well in terms of accuracy when compared to the other virtual mouse models. The novelty of the proposed model is that it can perform most of the mouse functions such as left click, right click, scroll up, scroll down, and mouse cursor movement using finger tip detection, and also, the model is helpful in controlling the PC like a physical mouse but in the virtual mode.



FUTURE SCOPE

This AI virtual mouse has some drawbacks such as drop in accuracy of some functions like right click operation and inability to perform other mouse functions such as dragging and dropping, and selecting text. Another major limitation is that this model cannot function in the dark or low light settings. These drawbacks can be addressed in the future and can be overcome. Apart from the above-mentioned, additionally key board capability can be incorporated to emulate keyboard functions along with the mouse operations which proves to be the scope for the future.

APPLICATIONS

The artificial-intelligence based virtual mouse has a plethora of application use cases. It could be used where there is restricted space for a conventional mouse and can be utilized in cases where the use of a traditional mouse device is not possible. This system rids of the need for a physical mouse and it can be used in its absence. Some Applications:

i. The proposed AI virtual mouse has an accuracy of approximately 98% which is larger than other systems that have been previously implemented.
ii. It can be utilized in the areas of automation for robots and other systems without the need of a physical device.
iii. The same system can be used for creating drawings both in two dimensions and three dimensions.
iv. People who cannot use a physical mouse can make use of this system to perform mouse like functions.
v. In the areas of virtual reality and augmented reality, the current system can be extended to work with such upcoming technologies.

CONCLUSION

The main objective of the proposed virtual AI mouse is to furnish an alternative to the conventional physical mouse that provides mouse functions with the help of computer vision enabled computer that houses a web camera which recognizes fingers and hand gestures and processes the captured frames and uses a machine learning algorithm to execute the defined mouse functions like moving the cursor, left click function. After testing we have concluded that the proposed virtual mouse system has worked exceedingly well and with greater accuracy when compared to previously proposed models mentioned in the related work and the current system has overcome the drawbacks of the other systems. As such, this proposed AI based virtual mouse system can be used in real-time and in real-world applications.

REFERENCES

[1]. Google, MP, https://ai.googleblog.com/2019/08/on-devicereal-time-hand-tracking-with.html. [2]. S. U. Dudhane, —Cursor control system using hand gesture recognition.

[2]. Hsieh, Chen-Chiung, Dung-HuaLiou, and David Lee. "A real time hand gesture recognition system using motion history image." 2010 2nd international conference on signal processing systems, Vol. 2. IEEE, 2010.

[3]. D.-S. Tran, N.-H. Ho, H.-J. Yang, S.-H.Kim, and G. S. Lee, —Real-time virtual mouse system using RGB-D images and fingertip detection,‖ Multimedia Tools and Applications Multimedia Tools and Applications, vol. 80, no. 7, pp. 10473–10490, 2021.

[4]. https://www.analyticsvidhya.com/blog/2021/07/building-a-hand-tracking-system-using-opencv/

[5].https://docs.opencv.org/4.x/da/dd0/tutorial_table_of_content_video.html

[6].https://www.javatpoint.com/opencv

[7].https://google.github.io/mediapipe/solutions/hands.html