AI Virtual Mouse

Submitted in partial fulfillment of the requirements

of the degree of

Masters of Computer Application

by

Adinath Dixit - *Roll No. 13*
Sarvesh Mahajan - *Roll No. 30*
Saurabh Tikam - *Roll No. 58*

Supervisor (s):

Mrs Ruchi Rautela Ma'am

Vivekanand Education Society's Institute Of Technology

University of Mumbai

2020 - 2021

CERTIFICATE

This is to certify that the project entitled **"AI VIRTUAL MOUSE"** is a bonafide work of "**Sarvesh Mahajan (30)**",**"Adinath Dixit (13),** **"Saurabh Tikam (58)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Postgraduate"** in **"Master of Computer Application".**

Ruchi Rautela

(Name and sign)
Supervisor/Guide

(Name and sign)
Co-Supervisor/Guide

(Name & Sign) Principal

(Name and sign) Head of Department

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Adinath Dixit (Roll No. 13)
Sarvesh Mahajan (Roll No. 30)
Saurabh tikam (Roll No. 58)

Date: 01/09/2021

Introduction

A mouse, in computing terms is a pointing device that detects two-dimensional movements relative to a surface. This movement is converted into the movement of a pointer on a display that allows to control the Graphical User Interface (GUI) on a computer platform. There are a lot of different types of mouse that have already existed in the modern days technology, there's the mechanical mouse that determines the movements by a hard rubber ball that rolls around as the mouse is moved. Years later, the optical mouse was introduced that replace the hard rubber ball to a LED sensor to detects table top movement and then sends off the information to the computer for processing. On the year 2004, the laser mouse was then introduced to improve the accuracy movement with the slightest hand movement, it overcome the limitations of the optical mouse which is the difficulties to track high-gloss surfaces. However, no matter how accurate can it be, there are still limitations exist within the mouse itself in both physical and technical terms. For example, a computer mouse is a consumable hardware device as it requires replacement in the long run, either the mouse buttons were degraded that causes inappropriate clicks, or the whole mouse was no longer detected by the computer itself.

Despite the limitations, the computer technology still continues to grow, so does the importance of the human computer interactions. Ever since the introduction of a mobile device that can be interact with touch screen technology, the world is starting to demand the same technology to be applied on every technological devices, this includes the desktop system. However, even though the touch screen technology for the desktop system is already exist, the price can be very steep.

Therefore, a virtual human computer interaction device that replaces the physical mouse or keyboard by using a webcam or any other image capturing devices can be an alternative way for the touch screen. This device which is the webcam will be constantly utilized by a software that monitors the gestures given by the user in order to process it and translate to motion of a pointes, as similar to a physical mouse.

Objectives :

The following describes the overall objectives of this project:

- To design to operate with the help of a webcam. The Virtual Mouse application will be operational with the help of a webcam, as the webcam are responsible to capture the images in real time. The application would not work if there are no webcam detected.
- To design a virtual input that can operate on all surface. The Virtual Mouse application will be operational on all surface and indoor environment, as long the users are facing the webcam while doing the motion gesture.
- To program the camera to continuously capturing the images, which the images will be analysed, by using various image processing techniques. As stated above, the Virtual Mouse application will be continuously capturing the images in real time, where the images will be undergo a series of process, this includes HSV conversion, Binary Image conversion, salt and pepper noise filtering, and more.
- To convert hand gesture/motion into mouse input that will be set to a particular screen position. The Virtual Mouse application will be programmed to detect the position of the defined colours where it will be set as the position of the mouse pointers. Furthermore, a combination of different colours may result in triggering different types of mouse events, such as the right/left clicks, scroll up/down, and more.

Purpose:

The purpose of this project is to develop a Virtual Mouse application that targets a few aspects of significant development. For starters, this project aims to eliminate the needs of having a physical mouse while able to interact with the computer system through webcam by using various image processing techniques. Other than that, this project aims to develop a Virtual Mouse application that can be operational on all kind of surfaces and environment.

Scope:

Virtual Mouse that will soon to be introduced to replace the physical computer mouse to promote convenience while still able to accurately interact and control the computer system. To do that, the software requires to be fast enough to capture an process every image, in order to successfully trac the user's gesture. Therefore, this project will develop a software application with the aid of the latest software coding technique and the open-source computer vision library also known as the OpenCV. The scope of the project is as below:

- Real time application.
- User friendly application.
- Removes the requirement of having a physical mouse

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 color 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

REQUIREMENT ANALYSIS

Software and Hardware Requirements:

Software :- PyCharm IDE

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the *Python* language. It is developed by the Czech company JetBrains (formerly known as IntelliJ).It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License,and there is also Professional Edition with extra features – released under a proprietary license.

Packages :-

- **opencv- python:** OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.
- **dlib -** dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world

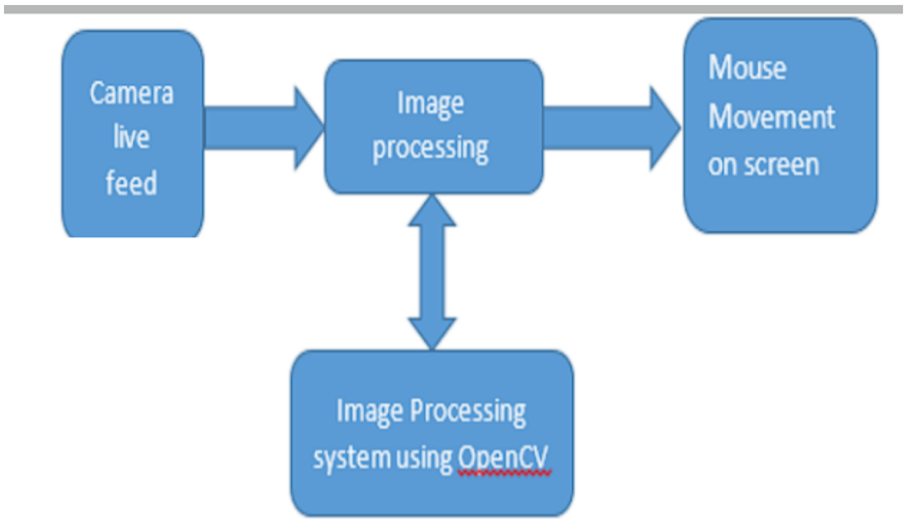
problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge.

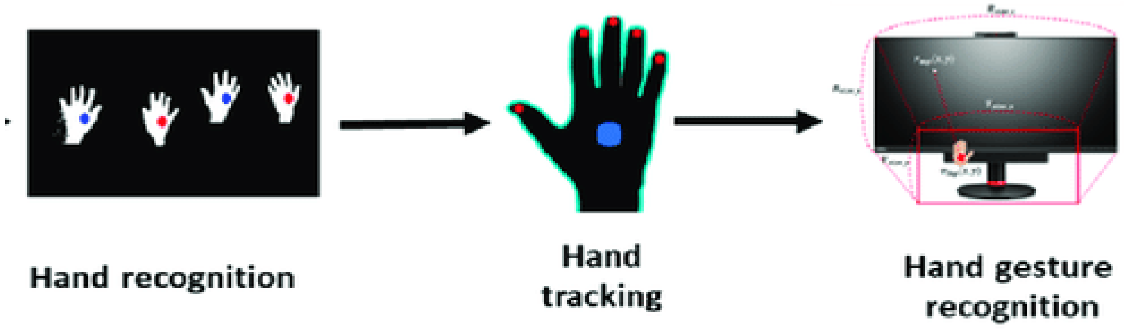
- ❖ **numpy -** NumPy brings the computational power of languages like C and Fortran to Python, a language much easier to learn and use. With this power comes simplicity: a solution in NumPy is often clear and elegant.
- ❖ **Autopy -** AutoPy is a simple, cross-platform GUI automation library for Python. It includes functions for controlling the keyboard and mouse, finding colors and bitmaps on-screen, and displaying alerts.

<u>Hardware Requirements :-</u>

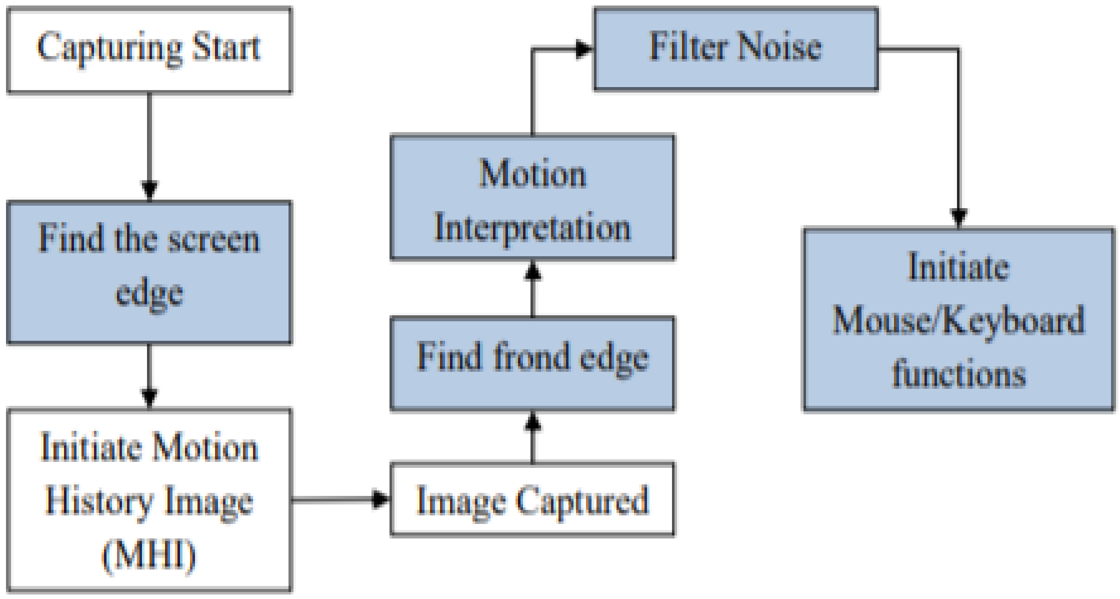
- ➔ 64-bit versions of Microsoft Windows 10, 8, 7 (SP1)
- ➔ 4 GB RAM minimum, 8 GB RAM recommended
- ➔ 1.5 GB hard disk space + at least 1 GB for caches
- ➔ 1024×768 minimum screen resolution
- ➔ Python 2.7, or Python 3.5 or newer
- ➔ Web Cam

SYSTEM DESIGN

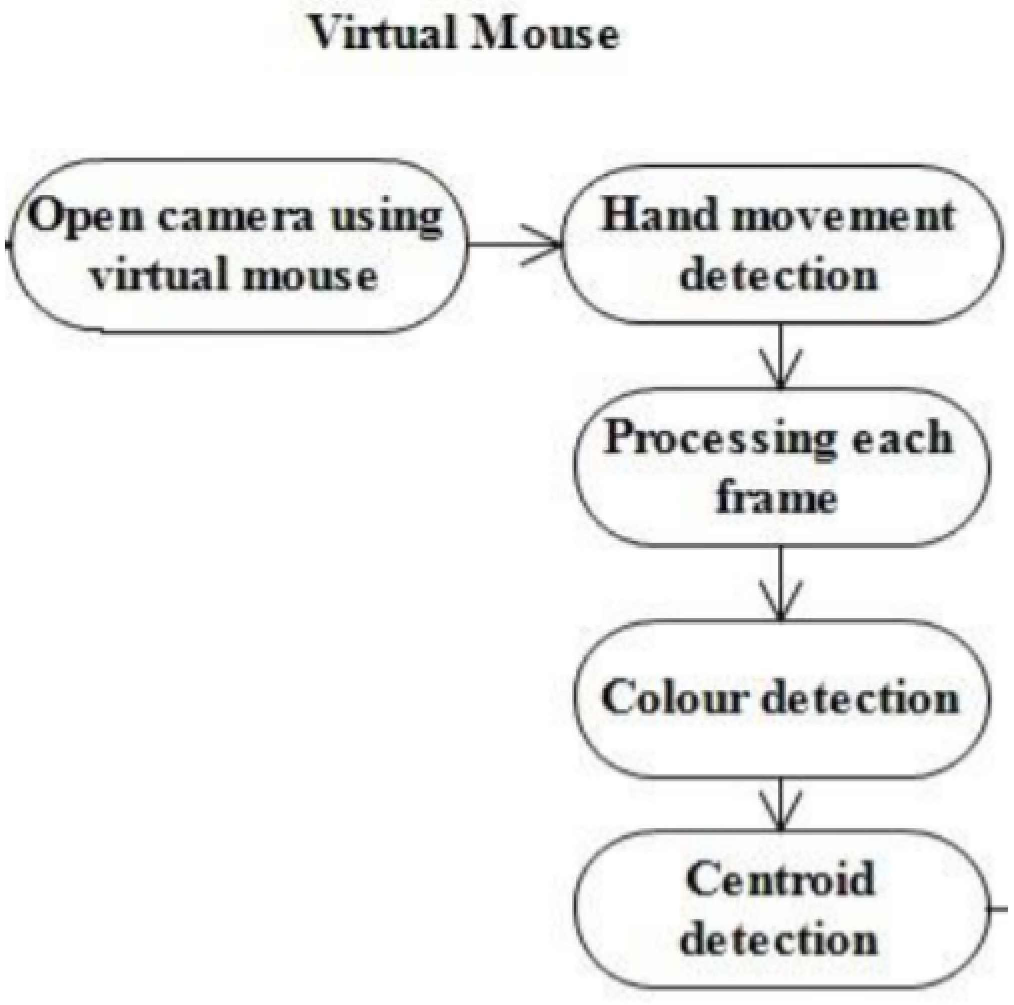




Flow Chart:


Capturing Start
Find the screen edge
Initiate Motion History Image (MHI)
Image Captured
Find frond edge
Motion Interpretation
Filter Noise
Initiate Mouse/Keyboard functions

Activity Diagram :



Code

HandTrackingModule.py

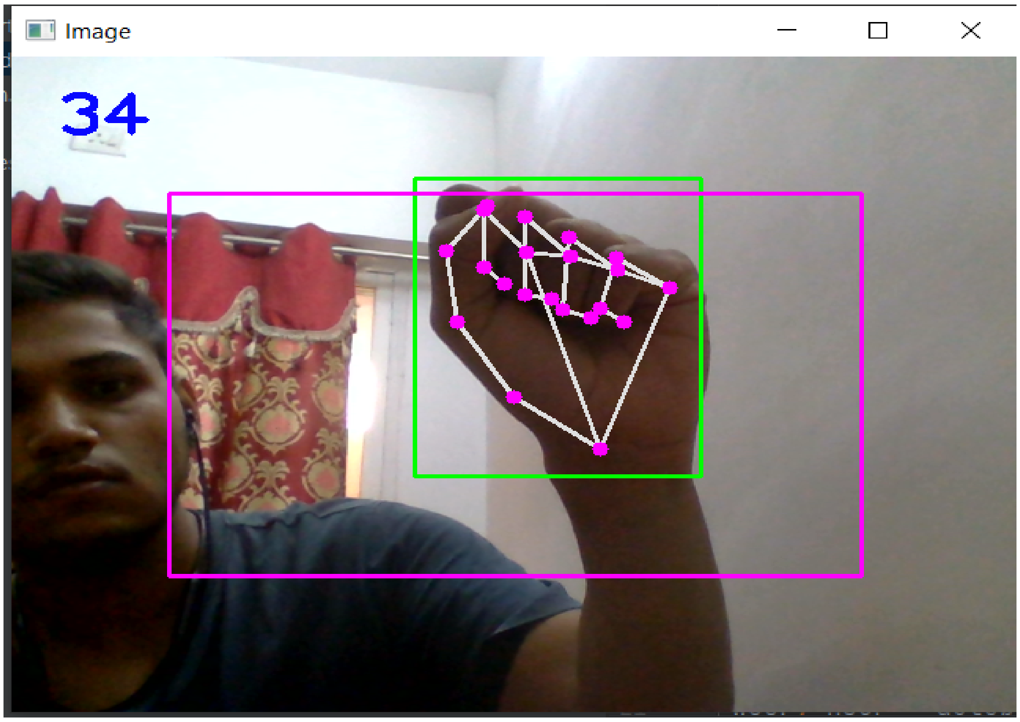
```
import cv2
import mediapipe as mp
import time
import math
import numpy as np


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

    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]:
            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]:
```

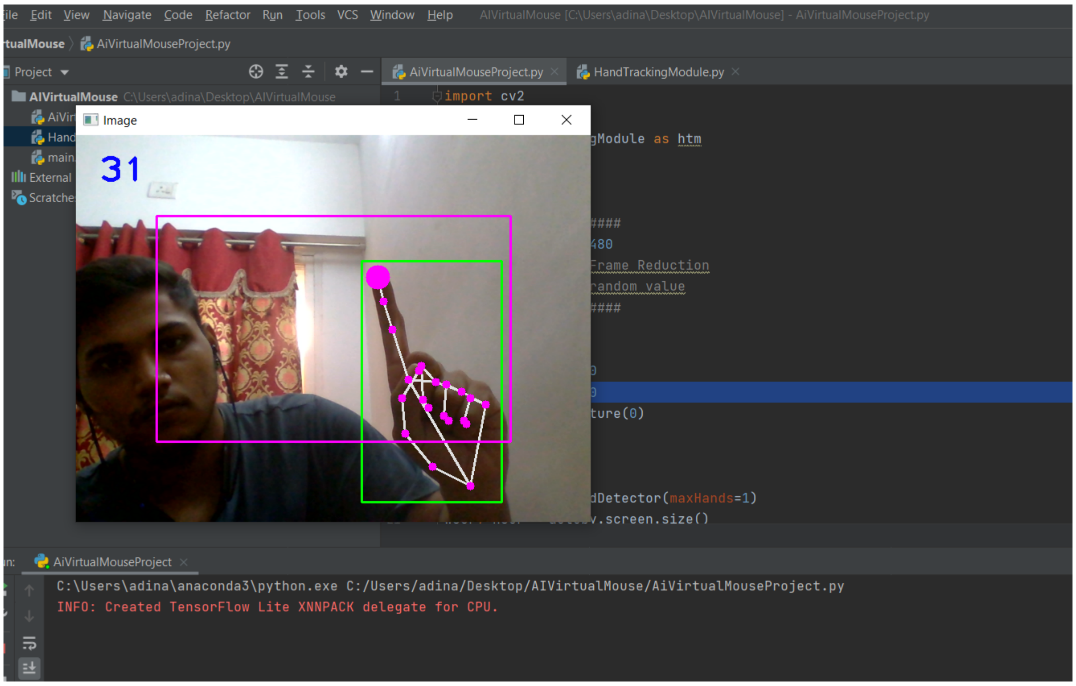
```
                fingers.append(1)
            else:
                fingers.append(0)
    # totalFingers = fingers.count(1)
        return fingers

    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)

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

def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(1)
    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()
```

AIVirtualMouse.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  #random value
######################

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:
    # Step1: Find the landmarks
    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bbox = detector.findPosition(img)
    # Step2: Get the tip of the index and middle finger
    if len(lmList) != 0:
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]

        # Step3: Check which fingers are up
        fingers = detector.fingersUp()
        cv2.rectangle(img, (frameR, frameR), (wCam -
frameR, hCam - frameR),
                      (255, 0, 255), 2)

        # Step4: Only Index Finger: Moving Mode
        if fingers[1] == 1 and fingers[2] == 0:

            # Step5: Convert the coordinates
            x3 = np.interp(x1, (frameR, wCam-frameR), (0,
wScr))
            y3 = np.interp(y1, (frameR, hCam-frameR), (0,
hScr))

            # Step6: Smooth Values
            clocX = plocX + (x3 - plocX) / smoothening
            clocY = plocY + (y3 - plocY) / smoothening

            # Step7: Move Mouse
            autopy.mouse.move(wScr - clocX, clocY)
            cv2.circle(img, (x1, y1), 15, (255, 0, 255),
cv2.FILLED)
            plocX, plocY = clocX, clocY

        # Step8: Both Index and middle are up: Clicking
Mode
        if fingers[1] == 1 and fingers[2] == 1:

            # Step9: Find distance between fingers
            length, img, lineInfo =
detector.findDistance(8, 12, img)

            # Step10: Click mouse if distance short
            if length < 40:
                cv2.circle(img, (lineInfo[4],
lineInfo[5]), 15, (0, 255, 0), cv2.FILLED)
                autopy.mouse.click()

    # Step11: Frame rate
    cTime = time.time()
    fps = 1/(cTime-pTime)
    pTime = cTime
    cv2.putText(img, str(int(fps)), (28, 58),
cv2.FONT_HERSHEY_PLAIN, 3, (255, 8, 8), 3)

    # Step12: Display
    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

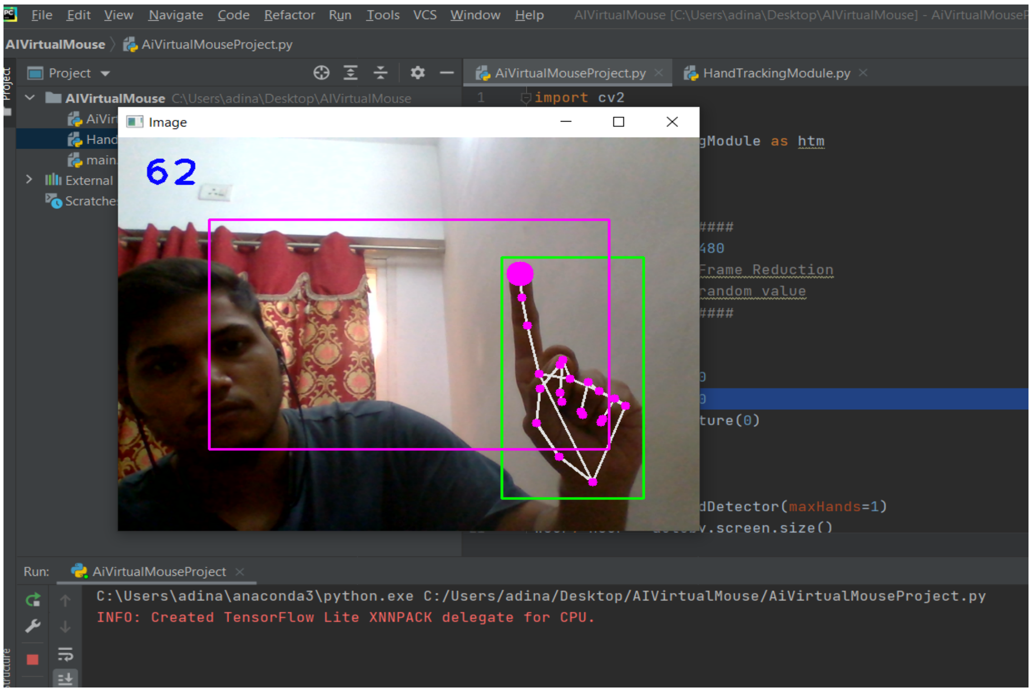
SCREENSHOTS OF PROJECT




File Edit View Navigate Code Refactor Run Tools VCS Window Help
AIVirtualMouse [C:\Users\adina\Desktop\AIVirtualMouse] - AiVirtualMouseProject.py
tualMouse
AiVirtualMouseProject.py
Project
AiVirtualMouseProject.py
HandTrackingModule.py
AIVirtualMouse C:\Users\adina\Desktop\AIVirtualMouse
AiVir
Hand
main
External
Scratche
Image
31
AiVirtualMouseProject

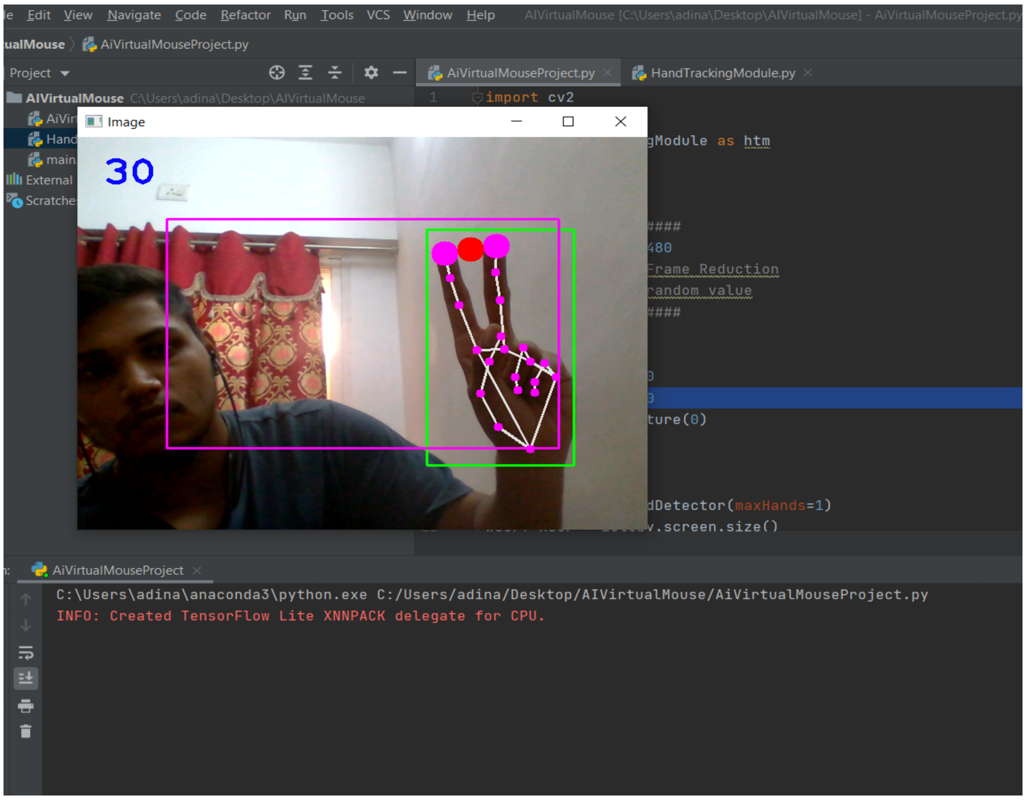
```
import cv2
gModule as htm
####
480
Frame Reduction
random value
####
ture(0)
dDetector(maxHands=1)
.screen.size()
```

```
C:\Users\adina\anaconda3\python.exe C:/Users/adina/Desktop/AIVirtualMouse/AiVirtualMouseProject.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
AIVirtualMouse [C:\Users\adina\Desktop\AIVirtualMouse] - AiVirtualMouse
AIVirtualMouse 〉 AiVirtualMouseProject.py
Project
AiVirtualMouseProject.py
HandTrackingModule.py
AIVirtualMouse C:\Users\adina\Desktop\AIVirtualMouse
External
Scratches
Image
62
Run: AiVirtualMouseProject

```
import cv2
gModule as htm
####
480
Frame Reduction
random value
####
0
0
ture(0)
dDetector(maxHands=1)
v.screen.size()
```

```
C:\Users\adina\anaconda3\python.exe C:/Users/adina/Desktop/AIVirtualMouse/AiVirtualMouseProject.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```


Edit View Navigate Code Refactor Run Tools VCS Window Help
AIVirtualMouse [C:\Users\adina\Desktop\AIVirtualMouse] - AiVirtualMouseProject.py
AiVirtualMouseProject.py
Project
AiVirtualMouseProject.py
HandTrackingModule.py
AIVirtualMouse C:\Users\adina\Desktop\AIVirtualMouse
External
Scratches
Image
30
AiVirtualMouseProject

```
import cv2
gModule as htm
####
480
Frame Reduction
random value
####
0
0
ture(0)
dDetector(maxHands=1)
v.screen.size()
```

```
C:\Users\adina\anaconda3\python.exe C:/Users/adina/Desktop/AIVirtualMouse/AiVirtualMouseProject.py
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Conclusion

In conclusion, it’s no surprised that the physical mouse will be replaced by a virtual non-physical mouse in the Human-Computer Interactions , where every mouse movements can be executed with a swift of your fingers everywhere and anytime without any environmental restrictions. This project had develop a virtual mouse program with the purpose of replacing the generic physical mouse without sacrificing the accuracy and efficiency, it is able to recognize movements, combinations, and translate them into actual mouse functions. Due to accuracy and efficiency plays an important role in making the program as useful as an actual physical mouse, a few techniques had to be implemented.

First and foremost, the coordinates that are in charge of handling the cursor movements are averaged based on a collections of coordinates, the purpose of this technique is to reduce and stabilize the sensitivity of cursor movements, as slight movement might lead to unwanted cursor movements.

The purpose of this implementation is to promote convenience in controlling the program without much of a hassle. Therefore, actual mouse functions can be triggered accurately with minimum trial and errors.

Reference

- https://www.geeksforgeeks.org/opencv-python-tutorial/
- https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv python/
- https://www.youtube.com/watch?v=NZde8Xt78Iw
- https://www.tutorialspoint.com/numpy/index.htm
- https://www.youtube.com/watch?v=oXlwWbU8l2o