BANGLADESH UNIVERSITY OF BUSINESS AND TECHNOLOGY

(BUBT)

A Project Report on

Virtual Mouse

COURSE CODE: CSE 352

COURSE TITLE: Artificial Intelligence & Expert System Lab

SUBMITTED BY

NAME	ID
MD. EMON ALI	18192103154
MOSRAFUL HABIB	18192103172
MOINUL ISLAM	18192103180

SUBMITTED TO

Humayra Ahmed

Lecturer, Dept. of CSE

BUBT

Index

Introduction

A **virtual mouse** is software that allows users to give mouse inputs to a system without using an actual mouse. To the extreme it can also be called as hardware because it uses an ordinary web camera. A virtual mouse can usually be operated with multiple input devices, which may include an actual mouse or a computer keyboard. Virtual mouse which uses web camera works with the help of different image processing techniques.

In this the hand movements of a user is mapped into mouse
inputs. A web camera is set to take images continuously. Most laptops today are equipped with webcams, which have recently been used insecurity applications utilizing face recognition. In order to harness the full potential of a webcam, it can be used for vision based CC, which would effectively eliminate the need for a computer mouse or mouse pad. The usefulness of a webcam can also be greatly extended to other HCI application such as a sign language database or motion controller. Over the past decades there have been significant advancements in HCI technologies for gaming purposes, such as the Microsoft Kinect and Nintendo Wii. These gaming technologies provide a more natural and interactive means of playing videogames. Motion controls is the future of gaming and it have tremendously boosted the sales of video games, such as the Nintendo Wii which sold over 50 million consoles within a year of its release. HCI using hand gestures is very intuitive and effective for one to one interaction with computers and it provides a Natural User Interface (NUI). There has been extensive research towards novel devices and techniques for cursor control using hand gestures. Besides HCI, hand gesture recognition is also used in sign language recognition, which makes hand gesture recognition even more significant.

Methodologies

As modern technology of human computer interactions become important in our everyday lives, varieties of mouse with all kind of shapes and sizes were invented, from a casual office mouse to a hard-core gaming mouse. However, there are some limitations to these hardware as they are not as environmental friendly as it seems. For example, the physical mouse requires a flat surface to operate, not to mention that it requires a certain area to fully utilize the functions offered. Furthermore, some of these hardware are completely useless when it comes to interact with the computers remotely due to the cable lengths limitations, rendering it inaccessible.

The current system is comprised of a generic mouse and trackpad monitor control system, as well as the absence of a hand gesture control system. The use of a hand gesture to access the monitor screen from a distance is not possible. Even though it is primarily attempting to implement, the scope is simply limited in the virtual mouse field. The existing virtual mouse control system consists of simple mouse operations using a hand recognition system, in which we can control the mouse pointer, left click, right click, and drag, and so on. The use of hand recognition in the future will not be used. Even though there are a variety of systems for hand recognition, the system they used is static hand recognition, which is simply a recognition of the shape made by the hand and the definition of action for each shape made, which is limited to a few defined actions and causes a lot of confusion. As technology advances, there are more and more alternatives to using a mouse.

A special sensor (or built-in webcam) can track head movement to move the mouse pointer around on the screen. In the absence of a mouse button, the software's dwell delay feature is usually used. Clicking can also be accomplished with a well-placed switch.

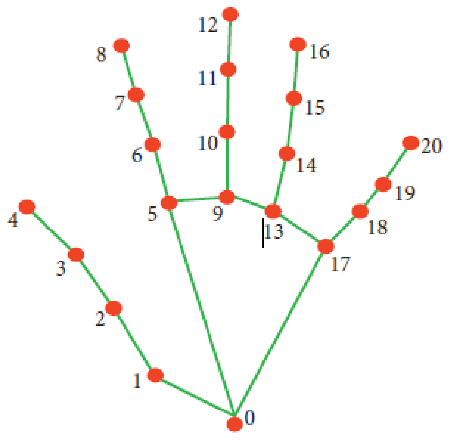
The various functions and conditions used in the system are explained in the flowchart of the real-time AI virtual mouse system in figure.

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.

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 colour 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)
```



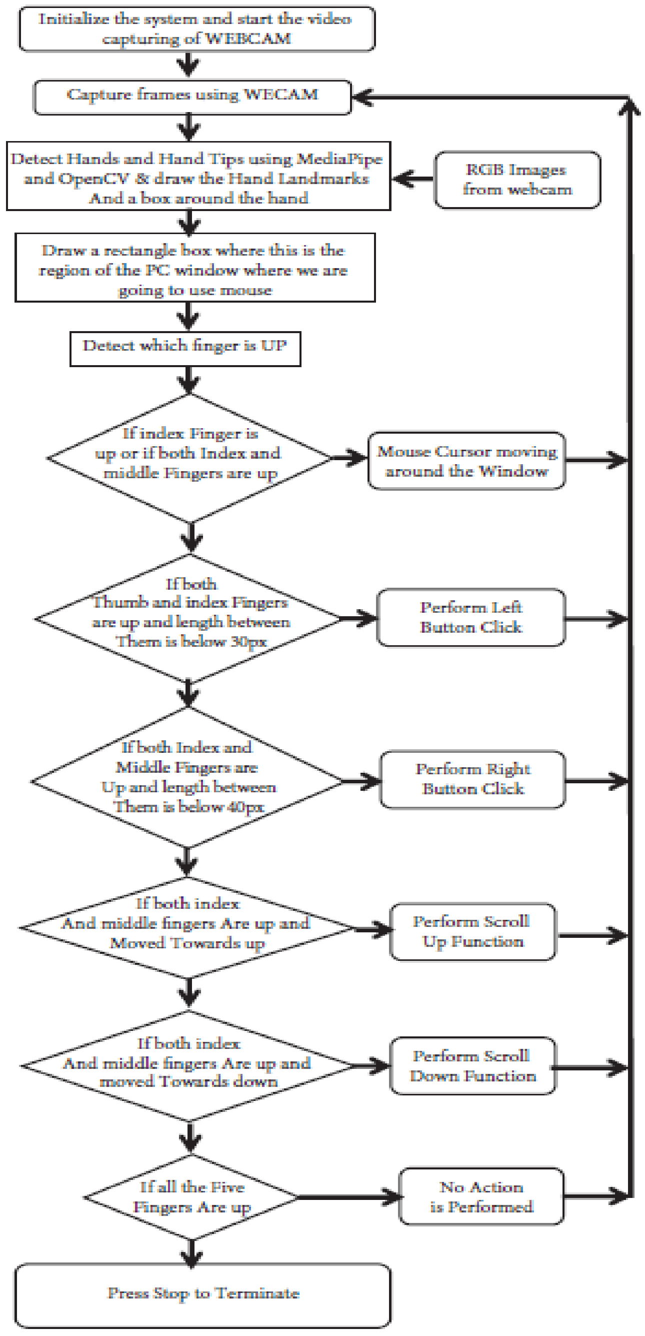
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP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

Rectangular Region for Moving through the Window. The AI virtual mouse system makes use of the transformational algorithm, and it converts the coordinates 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 , and according to that, the particular mouse function is performed.

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, as shown in Figure 7.

For the Mouse to Perform Left Button Click. If both the index finger with tip Id = 1 and the thumb finger with tip Id = 0 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 pynput.


Initialize the system and start the video capturing of WEBCAM
Capture frames using WECAM
Detect Hands and Hand Tips using MediaPipe and OpenCV & draw the Hand Landmarks And a box around the hand
RGB Images from webcam
Draw a rectangle box where this is the region of the PC window where we are going to use mouse
Detect which finger is UP
If index Finger is up or if both Index and middle Fingers are up
Mouse Cursor moving around the Window
If both Thumb and index Fingers are up and length between Them is below 30px
Perform Left Button Click
If both Index and Middle Fingers are Up and length between Them is below 40px
Perform Right Button Click
If both index And middle fingers Are up and Moved Towards up
Perform Scroll Up Function
If both index And middle fingers Are up and moved Towards down
Perform Scroll Down Function
If all the Five Fingers Are up
No Action is Performed
Press Stop to Terminate

Flow chart of AI Virtual Mouse

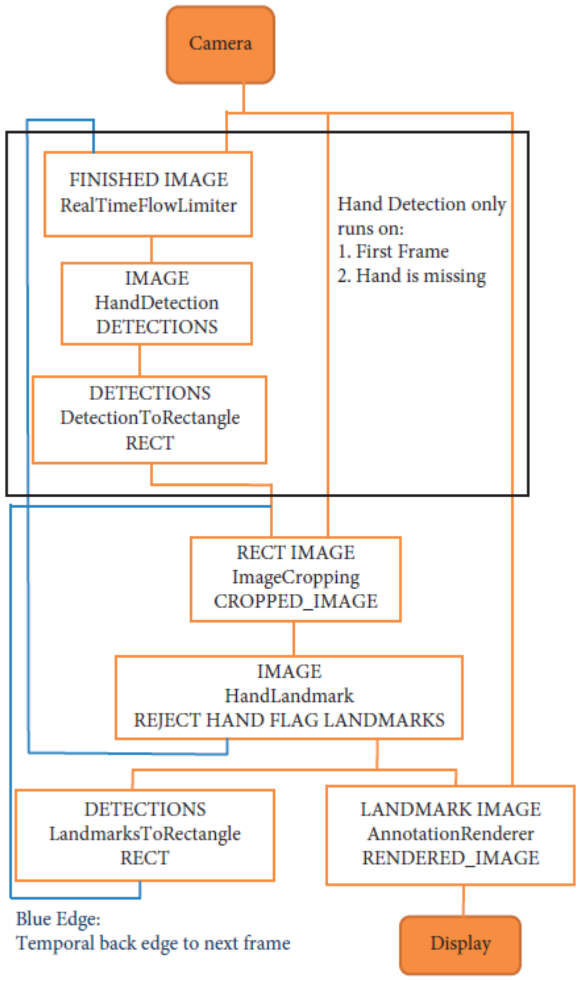
Algorithm and techniques used

For the purpose of detection of hand gestures and hand tracking, the MediaPipe framework is used, and OpenCV library is used for computer vision. The algorithm makes use of the machine learning concepts to track and
recognize the hand gestures and hand tip.

MediaPipe is a framework which is used for applying in a machine learning pipeline, and it is an opensource framework of Google. The MediaPipe framework is useful for cross platform development since the framework is built using the time series data. The MediaPipe framework is multimodal, where this framework can be applied to various audios and videos. 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. The steps involved in the system that uses MediaPipe are carried out in the pipeline configuration. The pipeline created can run in various platforms allowing scalability in mobile and desktops. 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. Developers are able to replace or define custom calculators anywhere in the graph creating their own application. 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.

Single-shot detector model is used for detecting and recognizing a hand or palm in real time. The single-shot detector model is used by the MediaPipe. First, in the hand detection module, it is first trained for a palm detection model because it is easier to train palms. Furthermore, the non maximum suppression works significantly better on small objects such as palms or fists . A model of hand landmark consists of locating joint or knuckle co-ordinates in the hand region,

OpenCV is a computer vision library which contains image-processing algorithms for object detection. OpenCV is a library of python programming language, and real-time computer vision applications can be developed by using the computer vision library. The OpenCV library is used in image and video processing and also analysis such as face detection and object detection .



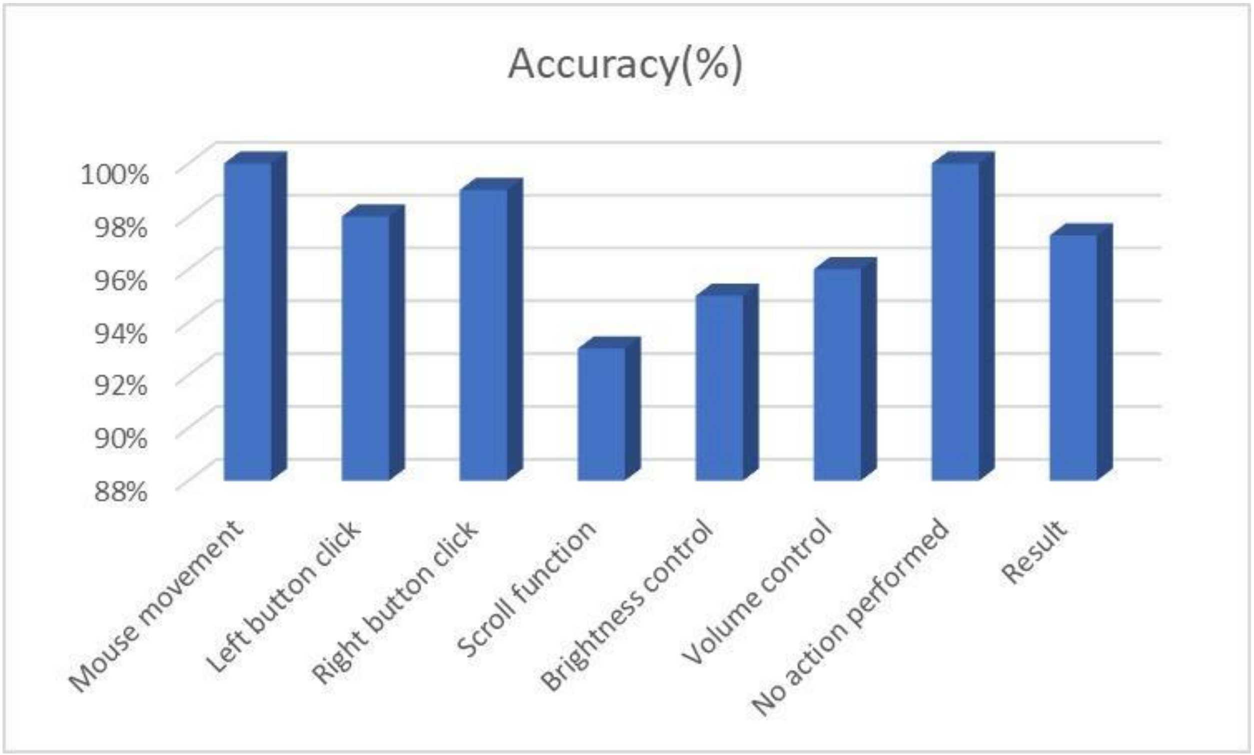
In the proposed AI virtual mouse system, the concept of advancing the human-computer interaction using computer vision is given.

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. An experimental test has been conducted to summarize the results shown in Table .

The test was performed 25 times by 4 persons resulting in 600 gestures with manual labeling, and this test has been made in different light conditions and at different distances from the screen, and each person tested the AI virtual mouse system 10 times in normal light conditions, 5 times in faint light conditions, 5 times in close distance from the webcam, and 5 times in long distance from the webcam, and the experimental results are tabulated in Table.

Mouse function performed	Success	Failure	Accuracy(%)
Mouse movement	100	0	100%
Left button click	98	2	98%
Right button click	99	1	99%
Scroll function	93	7	85%
Brightness control	95	5	95%
Volume control	96	4	86%
No action performed	100	0	100%
Result	681	19	90.01%

From Table , it can be seen that the proposed AI virtual mouse system had achieved an accuracy of about 97%. From this 97% accuracy of the proposed AI virtual mouse system, we come to know that the system has performed well. As seen in Table, the accuracy is low for “Scroll function” as this is the hardest gesture for the computer to understand. The accuracy for scroll function is low because the gesture used for performing the particular mouse function is harder. Also, the accuracy is very good and high for all the other gestures. Compared to previous approaches for virtual mouse, our model worked very well with 97% accuracy. The graph of accuracy is shown in Figure.


Accuracy(%)
100%
98%
96%
94%
92%
90%
88%
Mouse movement
Left button click
Right button click
Scroll function
Brightness control
Volume control
No action performed
Result

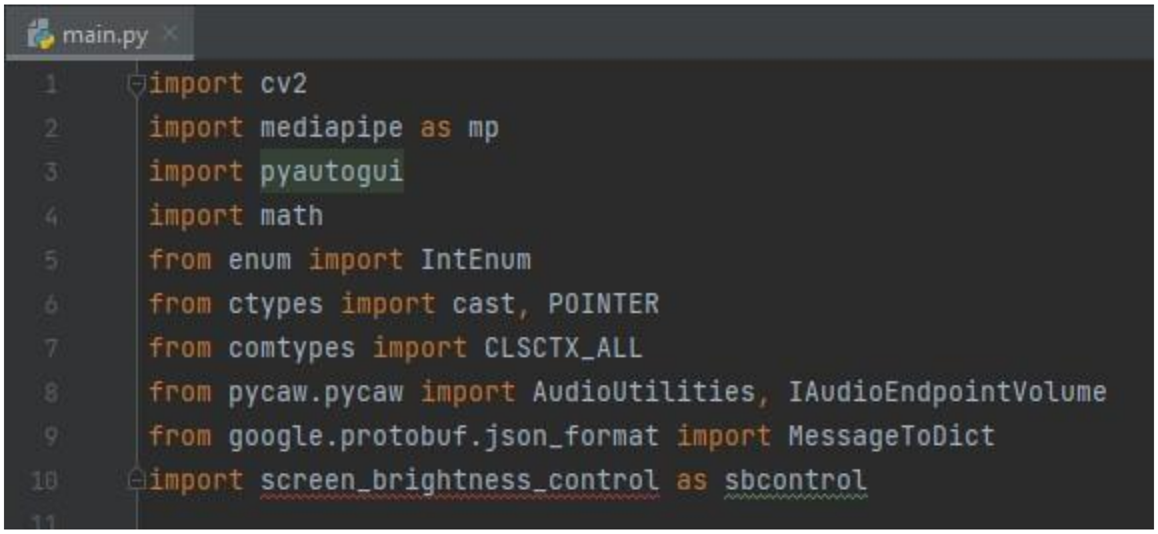
Implementation & Result

Test Cases **:**

Test case id	**Scenario**	**Boundary Value**	**Expected Result**	**Actual Result**	**Status**
1	Used in normal environment.	>90%	In normal environment hand gestures can be recognized easily.	Hand gestures got easily recognized and work properly.	**Passed**
2	Used in bright environment.	>60%	In brighter environment, software should work fine as it easily detects the hand movements but in a more brighter conditions it may not detect the hand gestures as expected.	In bright conditions the software works very well.	**Passed**
3	Used in dark environment	<30%	In dark environment, It should work properly.	In dark environment software didn't work properly in detecting hand gestures.	**Failed**
4	Used at a near distance (15cm) from the web cam.	>80%	At this distance, this software should perform perfectly.	It works fine and all features works properly.	**Passed**
5	Used at a far distance (35cm) from the web cam.	>95%	At this distance, this software should work fine.	At this distance, it is working properly.	**Passed**
6	Used at a farther distance (60cm) from the web cam.	>60%	At this distance, their will be some problem in detecting hand gestures but it should work fine.	At this distance, The functions of this software works properly.	**Passed**

Code with screenshots:

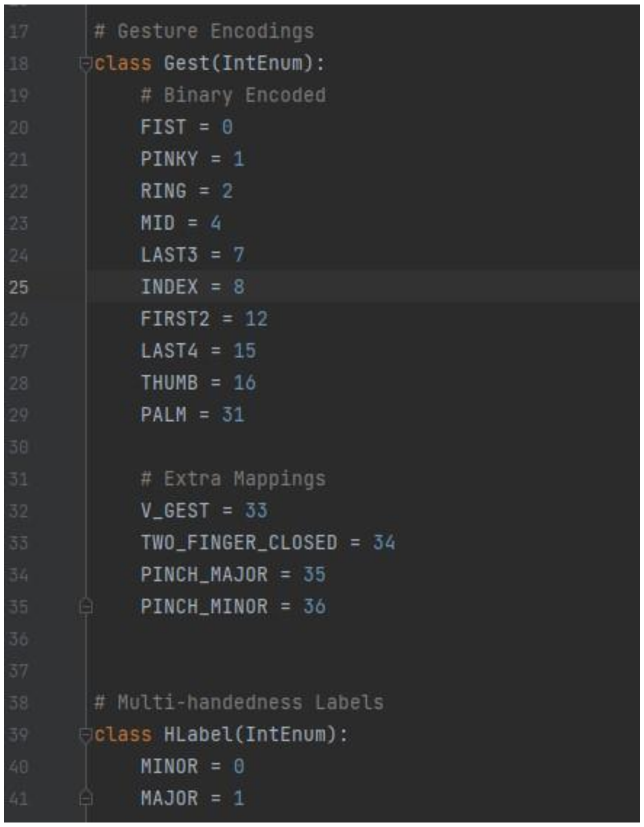
Imported packages for this program



main.py

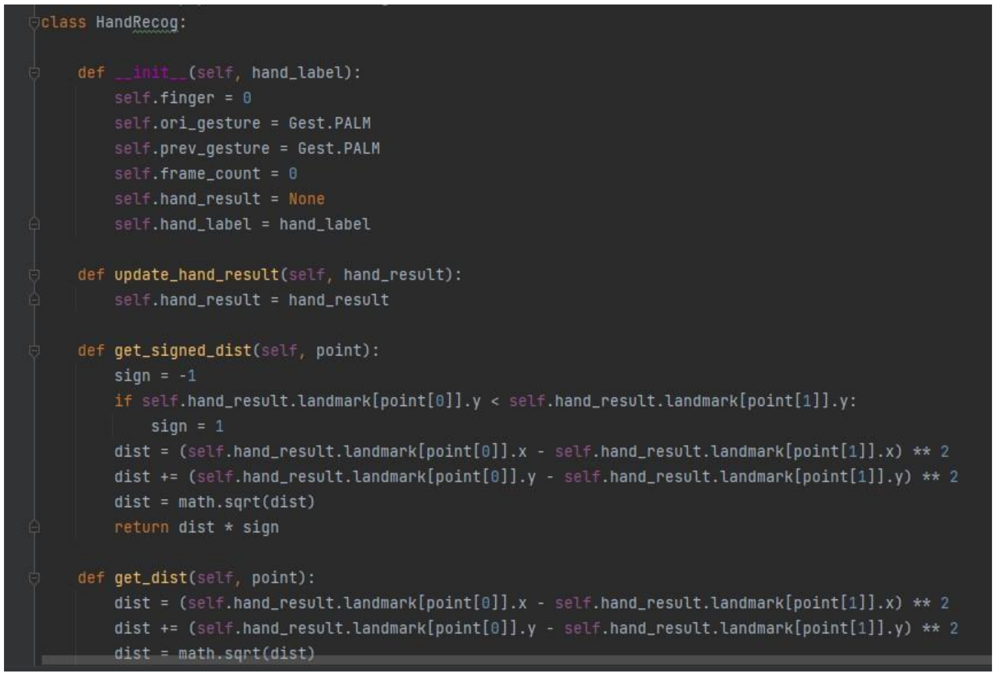
```python
import cv2
import mediapipe as mp
import pyautogui
import math
from enum import IntEnum
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
from google.protobuf.json_format import MessageToDict
import screen_brightness_control as sbcontrol
```

Important variables



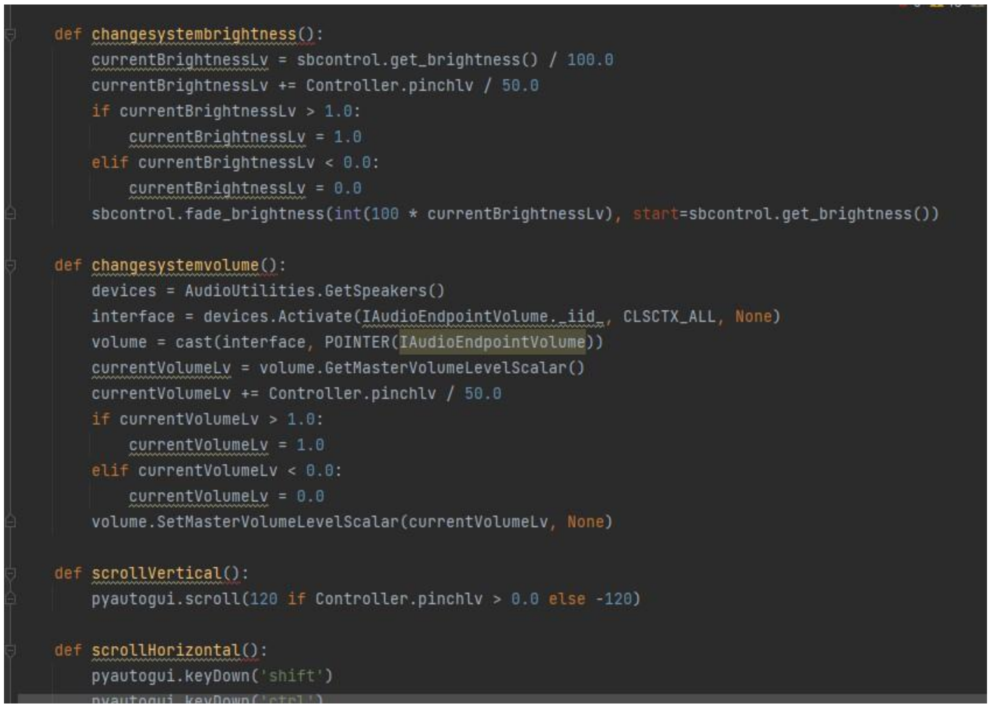
```python
# Gesture Encodings
class Gest(IntEnum):
    # Binary Encoded
    FIST = 0
    PINKY = 1
    RING = 2
    MID = 4
    LAST3 = 7
    INDEX = 8
    FIRST2 = 12
    LAST4 = 15
    THUMB = 16
    PALM = 31

    # Extra Mappings
    V_GEST = 33
    TWO_FINGER_CLOSED = 34
    PINCH_MAJOR = 35
    PINCH_MINOR = 36


# Multi-handedness Labels
class HLabel(IntEnum):
    MINOR = 0
    MAJOR = 1
```

For recognizing hand gesture



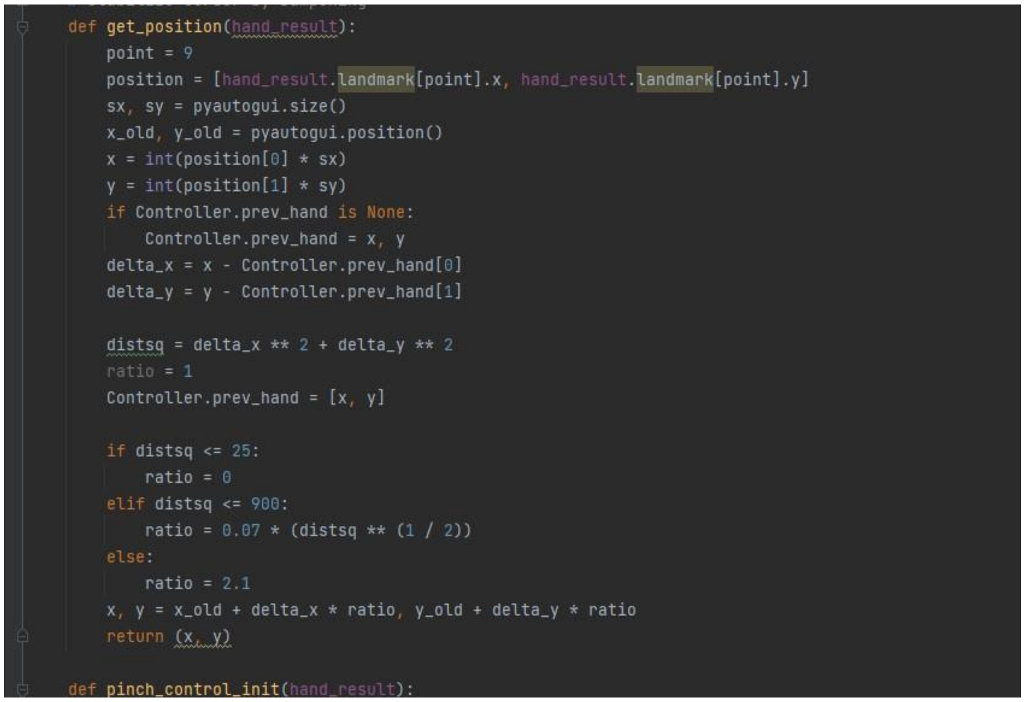
```
class HandRecog:

    def __init__(self, hand_label):
        self.finger = 0
        self.ori_gesture = Gest.PALM
        self.prev_gesture = Gest.PALM
        self.frame_count = 0
        self.hand_result = None
        self.hand_label = hand_label

    def update_hand_result(self, hand_result):
        self.hand_result = hand_result

    def get_signed_dist(self, point):
        sign = -1
        if self.hand_result.landmark[point[0]].y < self.hand_result.landmark[point[1]].y:
            sign = 1
        dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x) ** 2
        dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y) ** 2
        dist = math.sqrt(dist)
        return dist * sign

    def get_dist(self, point):
        dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x) ** 2
        dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y) ** 2
        dist = math.sqrt(dist)
```

Reaction by program of each gesture



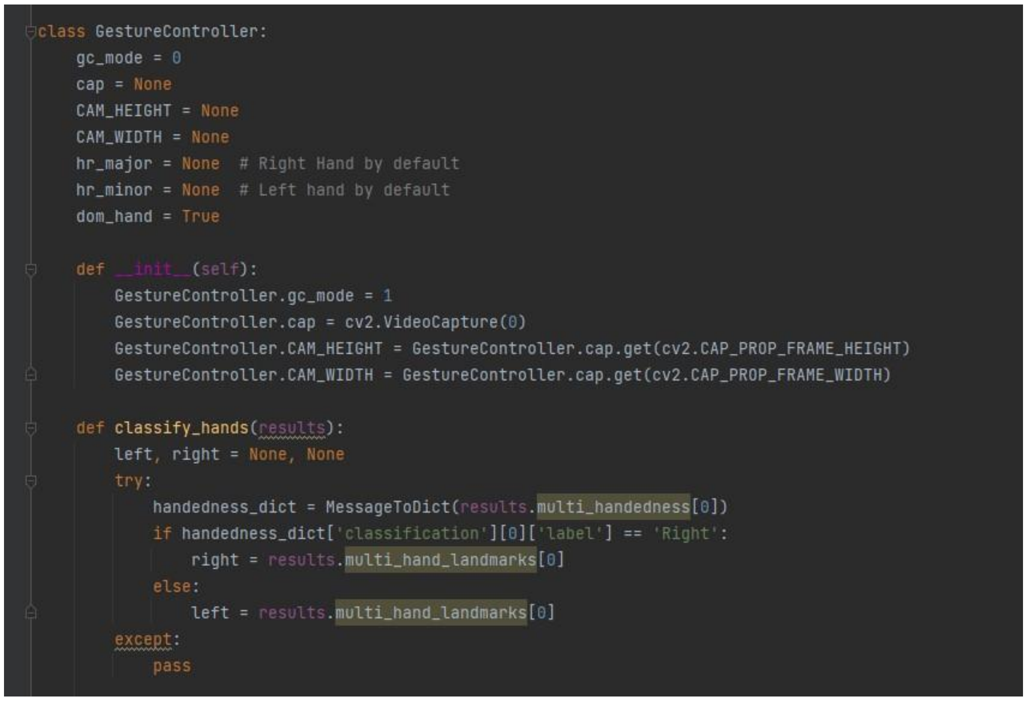
```
def changesystembrightness():
    currentBrightnessLv = sbcontrol.get_brightness() / 100.0
    currentBrightnessLv += Controller.pinchlv / 50.0
    if currentBrightnessLv > 1.0:
        currentBrightnessLv = 1.0
    elif currentBrightnessLv < 0.0:
        currentBrightnessLv = 0.0
    sbcontrol.fade_brightness(int(100 * currentBrightnessLv), start=sbcontrol.get_brightness())

def changesystemvolume():
    devices = AudioUtilities.GetSpeakers()
    interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
    volume = cast(interface, POINTER(IAudioEndpointVolume))
    currentVolumeLv = volume.GetMasterVolumeLevelScalar()
    currentVolumeLv += Controller.pinchlv / 50.0
    if currentVolumeLv > 1.0:
        currentVolumeLv = 1.0
    elif currentVolumeLv < 0.0:
        currentVolumeLv = 0.0
    volume.SetMasterVolumeLevelScalar(currentVolumeLv, None)

def scrollVertical():
    pyautogui.scroll(120 if Controller.pinchlv > 0.0 else -120)

def scrollHorizontal():
    pyautogui.keyDown('shift')
    pyautogui.keyDown('ctrl')
```

Axis measurement by program of each gesture



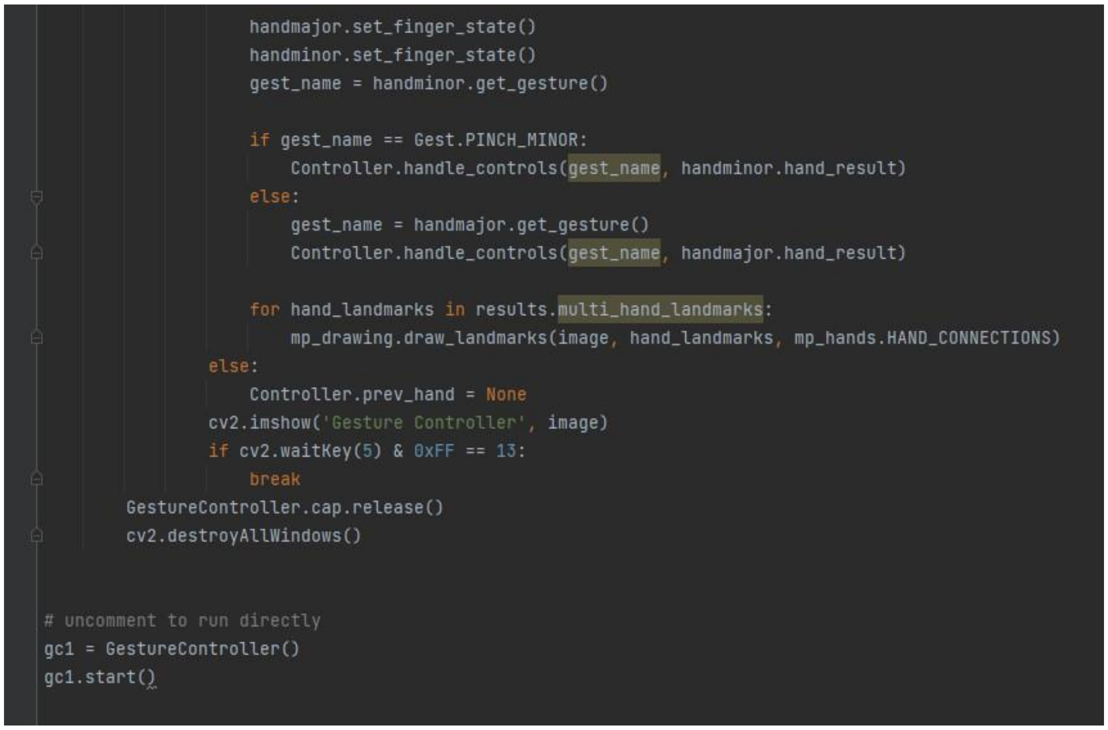
```
def get_position(hand_result):
    point = 9
    position = [hand_result.landmark[point].x, hand_result.landmark[point].y]
    sx, sy = pyautogui.size()
    x_old, y_old = pyautogui.position()
    x = int(position[0] * sx)
    y = int(position[1] * sy)
    if Controller.prev_hand is None:
        Controller.prev_hand = x, y
    delta_x = x - Controller.prev_hand[0]
    delta_y = y - Controller.prev_hand[1]

    distsq = delta_x ** 2 + delta_y ** 2
    ratio = 1
    Controller.prev_hand = [x, y]

    if distsq <= 25:
        ratio = 0
    elif distsq <= 900:
        ratio = 0.07 * (distsq ** (1 / 2))
    else:
        ratio = 2.1
    x, y = x_old + delta_x * ratio, y_old + delta_y * ratio
    return (x, y)

def pinch_control_init(hand_result):
```

Webcam starting from main function



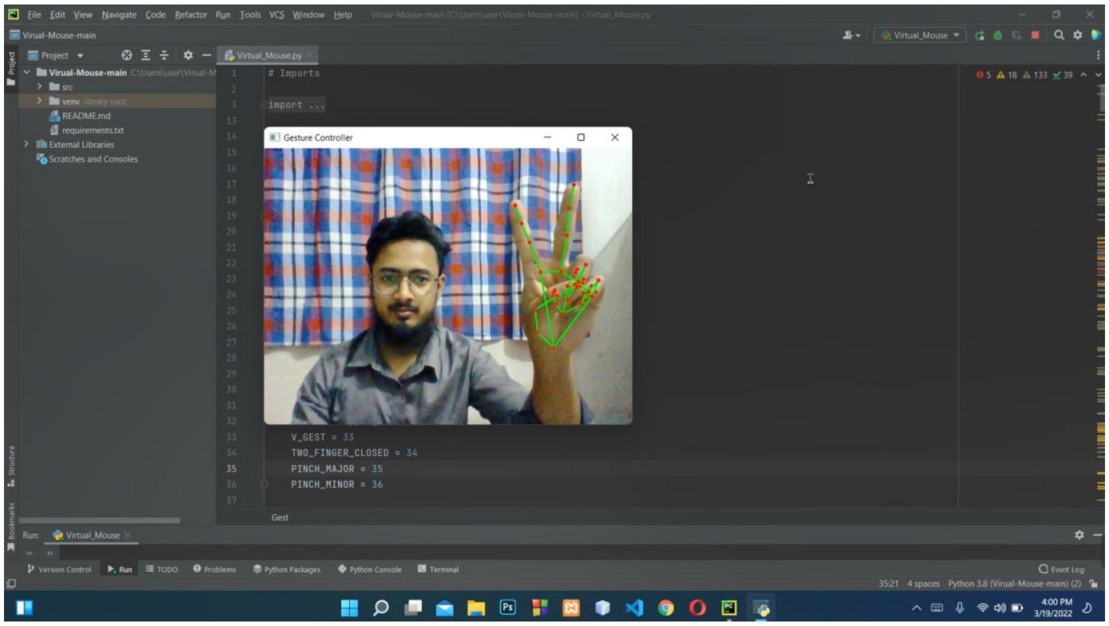
```
class GestureController:
    gc_mode = 0
    cap = None
    CAM_HEIGHT = None
    CAM_WIDTH = None
    hr_major = None  # Right Hand by default
    hr_minor = None  # Left hand by default
    dom_hand = True

    def __init__(self):
        GestureController.gc_mode = 1
        GestureController.cap = cv2.VideoCapture(0)
        GestureController.CAM_HEIGHT = GestureController.cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
        GestureController.CAM_WIDTH = GestureController.cap.get(cv2.CAP_PROP_FRAME_WIDTH)

    def classify_hands(results):
        left, right = None, None
        try:
            handedness_dict = MessageToDict(results.multi_handedness[0])
            if handedness_dict['classification'][0]['label'] == 'Right':
                right = results.multi_hand_landmarks[0]
            else:
                left = results.multi_hand_landmarks[0]
        except:
            pass
```

Parameter of different function

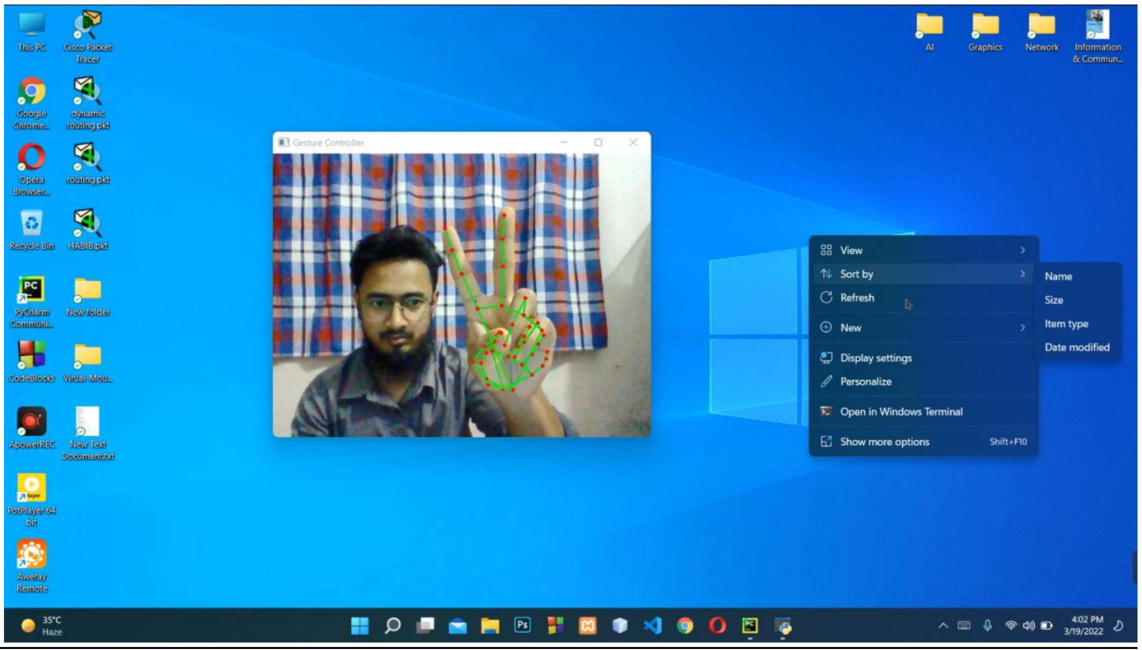


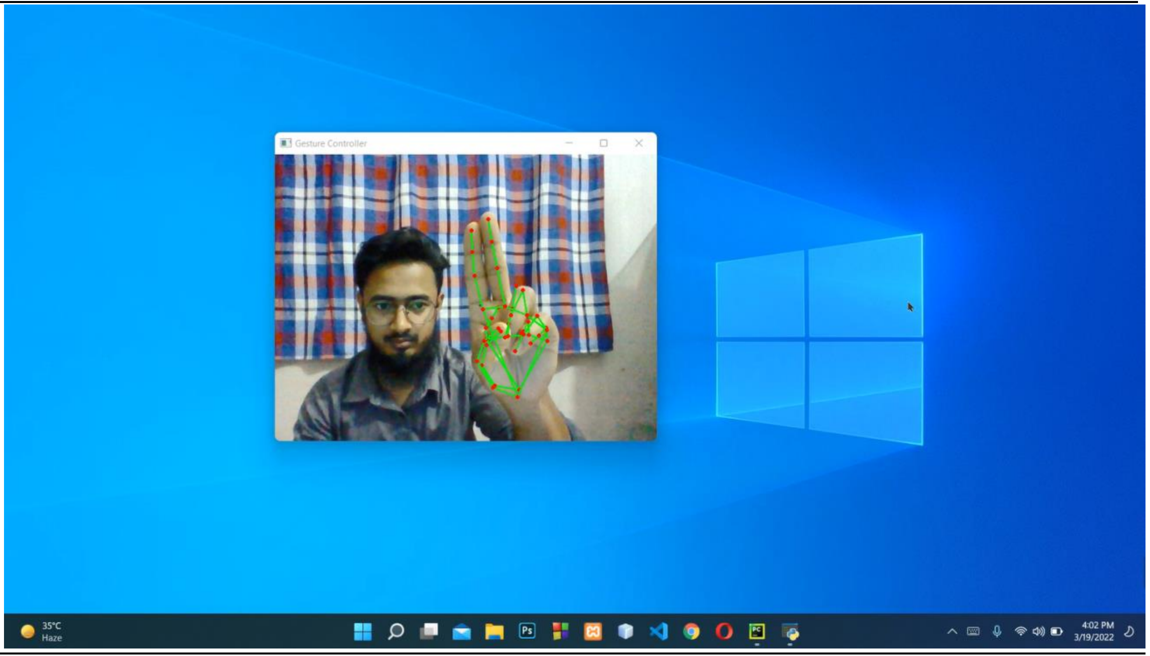
```
                handmajor.set_finger_state()
                handminor.set_finger_state()
                gest_name = handminor.get_gesture()

                if gest_name == Gest.PINCH_MINOR:
                    Controller.handle_controls(gest_name, handminor.hand_result)
                else:
                    gest_name = handmajor.get_gesture()
                    Controller.handle_controls(gest_name, handmajor.hand_result)

                for hand_landmarks in results.multi_hand_landmarks:
                    mp_drawing.draw_landmarks(image, hand_landmarks, mp_hands.HAND_CONNECTIONS)
            else:
                Controller.prev_hand = None
            cv2.imshow('Gesture Controller', image)
            if cv2.waitKey(5) & 0xFF == 13:
                break
        GestureController.cap.release()
        cv2.destroyAllWindows()


# uncomment to run directly
gc1 = GestureController()
gc1.start()
```

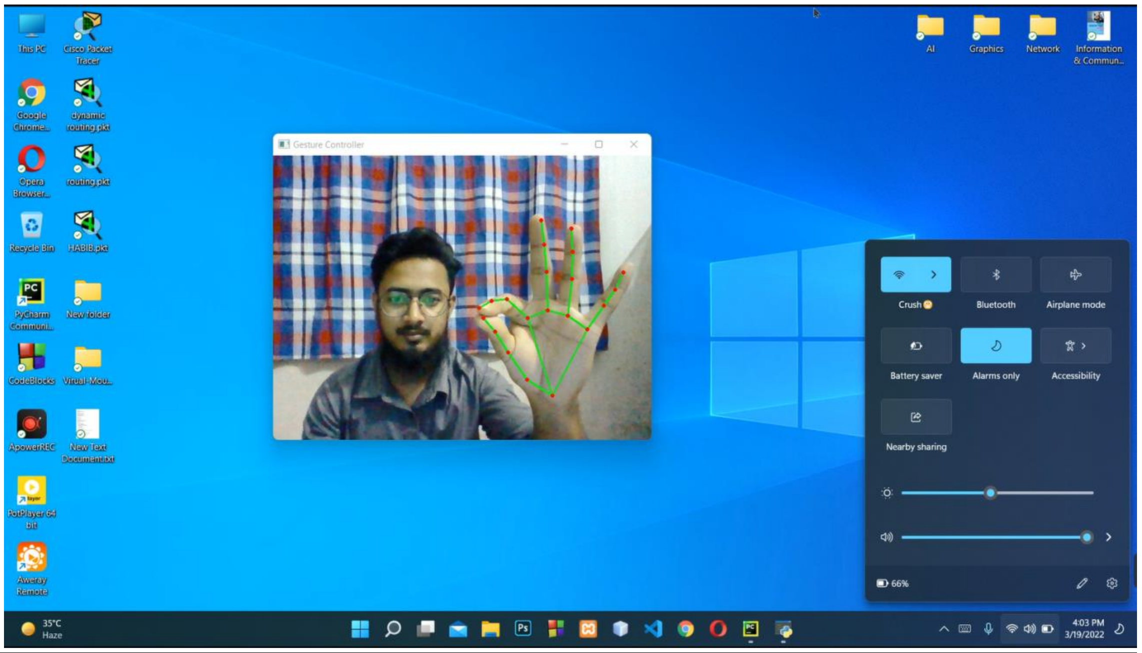
Implementation of program



```
# Imports

import ...

    V_GEST = 33
    TWO_FINGER_CLOSED = 34
    PINCH_MAJOR = 35
    PINCH_MINOR = 36
```


Gesture Controller
View
Sort by
Refresh
New
Display settings
Personalize
Open in Windows Terminal
Show more options
Shift+F10
Name
Size
Item type
Date modified


Gesture Controller


Gesture Controller
Crush
Bluetooth
Airplane mode
Battery saver
Alarms only
Accessibility
Nearby sharing
66%
35°C
Haze
4:03 PM
3/19/2022

Future work & Conclusion

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. After implanting such type of application there is big replacement of physical mouse i.e., there is no need of any physical mouse. Each & every movement of physical mouse is done with this motion tracking mouse (virtual mouse).

There are several features and improvements needed in order for the program to be more user friendly, accurate, and flexible in various environments. The following describes the improvements and the features required:

a) Smart Movement: Due to the current recognition process are limited within 25cm radius, an adaptive zoom in/out functions are required to improve the covered distance, where it can automatically adjust the focus rate based on the distance between the users and the webcam.

b) Better Accuracy & Performance: The response time are heavily relying on the hardware of the machine, this includes the processing speed of the processor, the size of the available RAM, and the available features of webcam. Therefore, the program may have better performance when it's running on a decent machine with a webcam that performs better in different types of lightings.

c) Mobile Application: In future this web application also able to use on Android devices, where touch screen concept is replaced by hand gestures.