AI VIRTUAL MOUSE

A PROJECT REPORT

Submitted By

KULDEEP GUPTA
(2100290140079)
PARTH MAURYA
(2100290140100)
AYUSH GUPTA
(2100290140044)

Submitted in partial fulfilment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Ms. Komal Salgotra
Teaching Assistant

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(JUNE 2023)

CERTIFICATE

Certified that **Kuldeep Gupta 2100290140079, Parth Maurya 2100290140100, Ayush Gupta 2100290140044** has/ have carried out the project work having "**AI VIRTUAL MOUSE**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Kuldeep Gupta (2100290140079)
Parth Maurya (2100290140100)
Ayush Gupta (2100290140044)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Ms. Komal Salgotra
Teaching Assistant
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Arun Tripathi
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

This project promotes an approach for the Human Computer Interaction (HCI) where cursor movement can be controlled using a real-time camera, it is an alternative to the current methods including manual input of buttons or changing the positions of a physical computer mouse. Instead, it utilizes a camera and computer vision technology to control various mouse events and is capable to performing every task that the physical computer mouse can.

The Virtual Mouse color recognition program will constantly acquire real-time images where the images will undergo a series of filtration and conversion. Whenever the process is complete, the program will apply the image processing technique to obtain the coordinates of the targeted colors position from the converted frames. After that, it will proceed to compare the existing colors within the frames with a list of color combinations, where different combinations consist of different mouse functions. If the current colors combination found a match, the program will execute the mouse function, which will be translated into an actual mouse function to the users' machine.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my thesis supervisor, **Ms. Komal Salgotra** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kinds of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Kuldeep Gupta

Parth Maurya

Ayush Gupta

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. PROJECT BACKGROUND

With the development of technologies in the areas of augmented reality and devices that we use in our daily life, these devices are becoming compact in the form of Bluetooth or wireless technologies. This paper proposes an AI virtual mouse system that makes use of hand gestures and hand tip detection for performing mouse functions in the computer using computer vision.

The main objective of the proposed system is to perform computer mouse cursor functions and scroll functions using a web camera or a built-in camera in the computer instead of using a traditional mouse device. Hand gesture and hand tip detection by using computer vision is used as a HCI with the computer. With the use of the AI virtual mouse system, we can track the fingertip of the hand gesture by using a built-in camera or web camera and perform the mouse cursor operations and scrolling function and also move the cursor with it. While using a wireless or a Bluetooth mouse, some devices such as the mouse, the dongle to connect to the PC and also a battery to power the mouse to operate are used, but in this paper, the user uses his/her built-in camera or a webcam and uses his/her hand gestures to control the computer mouse operations. In the proposed system, the web camera captures and then processes the frames that have been captured and then recognizes the various hand gestures and hand tip gestures and then performs the particular mouse function.

Python programming language is used for developing the AI virtual mouse system and also OpenCV which is the library for computer vision is used in the AI virtual mouse system. In the proposed AI virtual mouse system, the model makes use of the MediaPipe package for the tracking of the hands and for tracking of the tip of the hands and also Pynput, Autopy, and PyAuto GUI packages were used for moving around the window screen of the computer for performing functions such as left click, right click, and scrolling functions. The results of the proposed model showed a very high accuracy.

level, and the proposed model can work very well in real-world applications with the use of a CPU without the use of a GPU.

1.2. REVIEW OF THE PHYSICAL MOUSE

It is known that there are various types of physical computer mouse in modern technology, the following will discuss about the types and differences about the physical mouse.

1.2.1. MECHANICAL MOUSE

Known as the trackball mouse that is commonly used in the 1990s, the ball within the mouse is supported by two rotating rollers in order to detect the movement made by the ball itself. One roller detects the forward/backward motion while the other detects the left/right motion. The ball within the mouse is made of steel that was covered with a layer of hard rubber, so that the detection is more precise. The common functions included are the left/right buttons and a scroll-wheel. However, due to the constant friction made between the mouse ball and the rollers itself, the mouse is prone to degradation, as overtime usage may cause the rollers to degrade, thus causing it to unable to detect the motion properly, rendering it useless. Furthermore, the switches in the mouse buttons are no different as well, as long-term usage may cause the mechanics within to be loosed and will no longer detect any mouse clicks till it was disassembled and repaired.

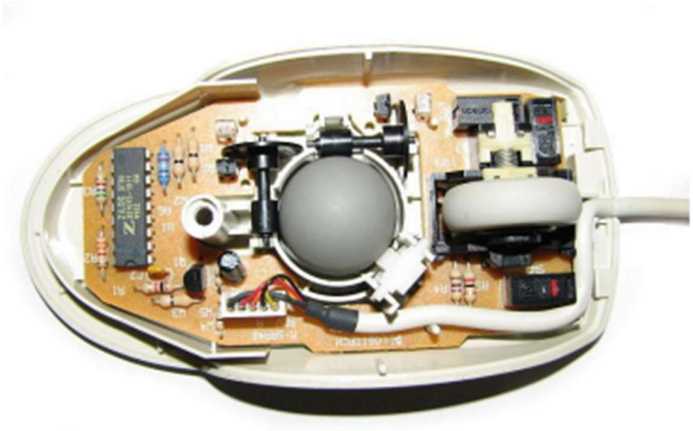

Figure 1.1 Mechanical mouse, with top cover removed.

The following table describes the advantages and disadvantages of the Mechanical Mouse.

Advantage Disadvantage	Advantage Disadvantage
• Allows the users to control the computer system by moving the mouse. • Provides precise mouse tracking movements	• Prone to degradation of the mouse rollers and button switches, causing faults. • Requires a flat surface to operate.

Table 1.1: Advantage and disadvantage of the Mechanical Mouse

1.2.2. OPTICAL AND LASER MOUSE

A mouse that is commonly used these days, the motions of optical mouse rely on the Light Emitting Diodes (LEDs) to detect movements relative to the underlying surface, while the laser mouse is an optical mouse that uses coherent laser lights. Compared to its predecessor, which is the mechanical mouse, the optical mouse no longer relies on the rollers to determine its movement, instead it uses an imaging array of photodiodes. The purpose of implementing this is to eliminate the limitations of degradation that plagues the current predecessor, giving it more durability while offering better resolution and precision. However, there's still some downside, even though the optical mouse is functional on most opaque diffuse surfaces, it's unable to detect motions on the polished surface.

Furthermore, long term usage without proper cleaning or maintenance may lead to dust particles trapping between the LEDs, which will cause both optical and laser mouse to have surface detection difficulties. Other than that, it's still prone to degradation of the button switches, which again will cause the mouse to function improperly unless it is disassembled and repaired.

The following table describes the advantages and disadvantages of the Optical and Laser Mouse.

Advantages Disadvantages	Advantages Disadvantages
• Allows better precision with lesser hand movements. • Longer life-span.	• Prone to button switches degradation. • Does not function properly while on a polished surface

Table 1.2: Advantage and disadvantage of the Optical and Laser Mouse

Figure 1.2 Optical Mouse, with top cover removed.

1.3. PROBLEM STATEMENT

The projected AI virtual mouse using hand signal structure could in like manner be familiar with beat issues inside the spot like things where there isn't any space to use a genuine mouse and set up for individuals who have issues in their grip and don't appear, apparently, to be prepared to manage a real mouse. Moreover, the COVID circumstance, it isn't safeguarded to include the devices by reaching them as an eventual outcome of it's intending to achieve what is happening of spread out of the disease by reaching the contraptions, that the projected AI virtual mouse could in like manner be

adjusted vanquished these issues since hand sign and hand Tip disclosure is used to manage the device mouse limits by using a camera or a characteristic camera like webcam.

While using a remote or a Bluetooth mouse, a couple of devices especially like the mouse, the contraption to connect with the pc, and besides, battery to drive the mouse to control a used, so all through this, the client uses his/her natural camera or visual camera and usages his/her hand movements to manage the PC mouse action.

The following describes the general problem that the current physical mouse suffers:

- Physical mouse is subjected to mechanical wear and tear.
- Physical mouse requires special hardware and surface to operate.
- A physical mouse is not easily adaptable to different environments and its performance varies depending on the environment.
- Mouse has limited functions even in present operational environments.
- All wired mouse and wireless mouse have its own lifespan.

1.4. MOTIVATION OF VIRTUAL MOUSE

It is fair to say that the Virtual Mouse will soon to be substituting the traditional physical mouse soon, as people are aiming towards the lifestyle where that every technological device can be controlled and interacted remotely without using any peripheral devices such as the remote, keyboards, etc. it doesn't just provide convenience, but it's cost effective as well.

1.4.1. CONVENIENT

It is known to interact with the computer system, users are required to use an actual physical mouse, which also requires a certain area of surface to operate, not to mention that it suffers from cable length limitations. Virtual Mouse requires none of it, as it is only a webcam to allow image capturing of user's hand position to determine the position of the pointers that the user wants it to be.

1.4.2. COST EFFECTIVE

A quality physical mouse normally costs from the range of 30 ringgit to a hefty 400 ringgit, depending on their functionality and features. Since the Virtual Mouse requires only a webcam, a physical mouse are no longer required, thus eliminating the need to purchase one, as a single webcam is sufficient enough to allow users to interact with the computer system through it, while some other portable computer system such as the laptop, are already supplied with a built-in webcam, could simply utilize the Virtual Mouse software without having any concerns about purchasing any external peripheral devices.

1.5. PROJECT SCOPE

The proposed AI virtual mouse has some limitations such as a small decrease in accuracy of the right click mouse function and also the model has some difficulties in executing clicking and dragging to select the text. These are some of the limitations of the proposed AI virtual mouse system, and these limitations will be overcome in our future work. Furthermore, the proposed method can be developed to handle the keyboard functionalities along with the mouse functionalities virtually which is another future scope of Human-Computer Interaction (HCI).

1.6. PROJECT OBJECTIVE

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 need of having a physical mouse while being 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 kinds of surfaces and environments.

The following describes the overall objectives of this project:

1. 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 is responsible for capturing the images in real time. The application would not work if there were no webcam detected.
2. To design a virtual input that can operate on all surfaces. The Virtual Mouse application will be operational on all surfaces and indoor environments, as long the users are facing the webcam while doing the motion gesture.
3. To program the camera to continuously capture the images, which the images will be analyzed, 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 undergo a series of processes, this includes HSV conversion, Binary Image conversion, salt, and pepper noise filtering, and more.
4. 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 colors where it will be set as the position of the mouse pointers. Furthermore, a combination of different colors may result in triggering different types of mouse events, such as the right/left clicks, scroll up/down, and more.

CHAPTER 2

LITERATURE

2.1. LITERATURE REVIEW

The current construction contains a nonexclusive mouse and trackpad screen control framework, as well as the mishap of a hand development control structure. The utilization of a hand development to get to the screen from a nice way is unimaginable.

No matter what how it is basically attempting to execute, the degree is just restricted in the virtual mouse field.

The current virtual mouse control structure contains direct mouse tasks utilizing a hand attestation framework, in which we have some control over the mouse pointer, left click, right snap, and drag, etc. The utilization of hand confirmation in the future won't be utilized. Despite how there are a gathering of frameworks for hand certification, the construction they utilized is static hand attestation, which is just a confirmation of the shape made by the hand and the meaning of activity for each shape made, which is restricted to a few depicted activities and makes a great deal of unsettling influence.

As progression drives, there are something else and more decisions rather than utilizing a mouse.

Coming up next are a piece of the techniques that were used: -

1. Camera Used in the Virtual Gesture Mouse project: Open- CV is python vision library that contains Associate in the organized AI virtual mouse structure depends upon the edges that are gotten by the camera in Associate in nursing passing computer.

2. Providing Input: Pictures in Computer Vision are portrayed as associations of numbers watching out for the discrete eclipsing or power values present in each picture pixel. Each picture is considered as information displayable in various ways, whether as collections of pixel values or either complex plot keeping an eye on the course of pixel powers.
3. Moving hand through the Window using rectangular area: The AI virtual mouse structure uses the informative algorithmic rule, and it changes over the co-ordinates of tip from the camera screen to the pc window full screen for the mouse.
4. Detect the Fingertips and doing the Mouse Cursor improvements.
5. In this construction, AI mouse is police evaluation that finger is up deceiving the spot co-ordinate of the finger that it'll found abuse the Media-Pipe and along these lines the specific bits of the fingers that region unit up, and according to that, the authentic mouse perform is played out its assignments.

Regardless, all of the systems under have its own game plan of checks. The usage of the head or eyes to control the cursor constantly can be risky to one's prosperity. This can induce different issues with flourishing. While using a touch screen, the client ought to stay aware of their accentuation on the screen constantly, which can cause drowsiness. By taking a gander at the going with systems, we want to make another endeavor that won't hurt the client's prosperity.

2.2. VISUAL PANEL

To overcome the stated problems, Zhengyou et al. (2001), proposed an interface system named Visual Panel that utilizes arbitrary quadrangle-shaped planar object as a panel to allow the user to use any tip-pointer tools to interact with the computer. The interaction movements will be captured, analyzed and implemented in the positions of the tip-pointer, resulting accurate and robust interaction with the computer. The overall system consists of panel tracker, tip-pointer tracker, holography, calculation and update, and action detector and event generator as it can simulate both mouse and keyboard.

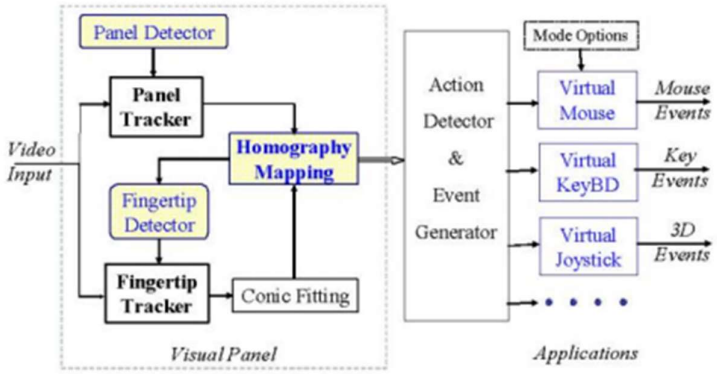

Figure 2.1: The system overview of Visual Panel (Zhengyou, Ying and Shafer, 2001)

2.3. PORTABLE VISION-BASED HUMAN COMPUTER INTERACTION (HCI)

Another "Ubiquitous Computing" approach proposed by Chu-Feng Lien (2015), requires only fingertips to control the mouse cursor and click events. The proposed system doesn't require hand-gestures nor color tracking in order to interact with the system, instead it utilizes a feature named Motion History Images (MHI), a method that is used to identify movements with a row of images in time.

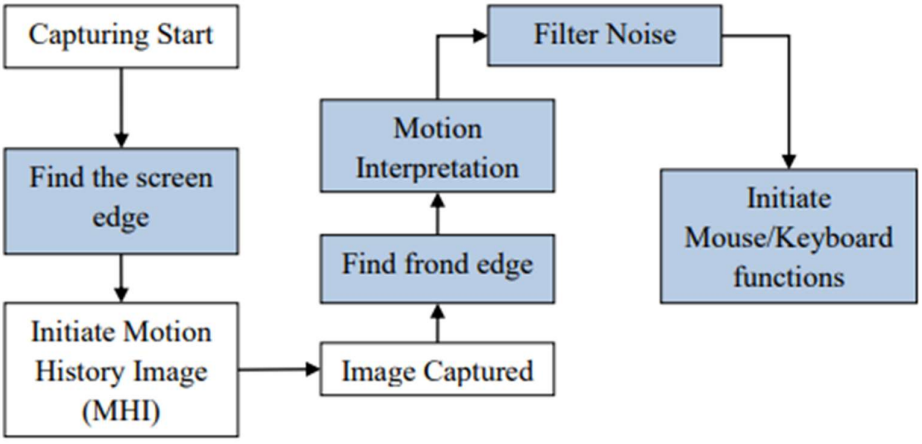

Figure 2.2: The Flow Chart of Portable Vision-Based Human Computer Interaction

CHAPTER 3

RELATED WORK

3.1. WORKING

There are some related works carried out on virtual mouse using hand gesture detection by wearing a glove in the hand and using color tips in the hands for gesture recognition, but they are no more accurate in mouse functions. The recognition is not so accurate because of wearing gloves; also, the gloves are also not suited for some users, and in some cases, the recognition is not so accurate because of the failure of detection of color tips. Some efforts have been made for camera-based detection of the hand gesture interface.

In 1990, Quam introduced an early hardware-based system; in this system, the user should wear a Data Glove [2]. The proposed system by Quam although gives results of higher accuracy, it is difficult to perform some of the gesture controls using the system.

Dung-Hua Liou, ChenChiung Hsieh, and David Lee in 2010 [3] proposed a study on "A Real-Time Hand Gesture Recognition System Using Motion History Image." The main limitation of this model is more complicated hand gestures.

Monika B. Gandhi, Sneha U. Dudhane, and Ashwini M. Patil in 2013 [4] proposed a study on "Cursor Control System Using Hand Gesture Recognition." In this work, the limitation is stored frames are needed to be processed for hand segmentation and skin pixel detection.

Vinay Kr. Pasi, Saurabh Singh, and Pooja Kumari in 2016 [5] proposed "Cursor Control using Hand Gestures" in the IJCA Journal. The system proposes the different bands to perform different functions of the mouse. The limitation is it depends on various colors to perform mouse functions.

CHAPTER 4

ALGORTHIM USED

4.1. ALGORTHIM USED FOR AI VIRTUAL MOUSE

For the characteristic of area of hand signals and hand development, the Media Pipe system is utilized, and Open-CV library is utilized for PC machine vision the standard purposes the AI contemplations to keep and see the hand developments and fingertip.

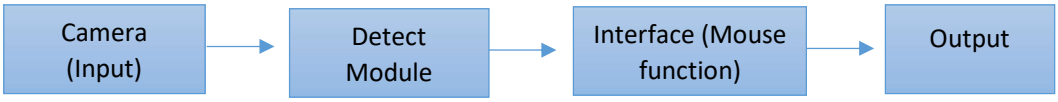

Figure 4.1. Block Diagram of System

4.2. MEDIA-PIPE

Media-Pipe is a system that is utilized for applying in a different AI pipeline, partnered with an open-source structure of Google. The Media-Pipe system is useful for across stage improvement since the edge work is made abuse the measurement data. The Media-Pipe structure is multi-modular, any place this system is frequently applied to differed sounds and recordings. The Media-Pipe structure is utilized by the engineer for building and breaking down the frameworks through diagrams, and it conjointly been utilized for fostering the frameworks for the machine reason.

The means worried inside the framework that utilizes Media- Pipe square measure administrated inside the line setup. The pipeline made will run in various stages allowing quantity friability in portable and work areas. The Media-Pipe structure is predicated on three rudimentary parts, they're execution investigation, system for recovering identifier data, and a gathering of parts that square measure known as mini-

computers and those they square measure reusable. A pipeline might be a chart that comprises of parts known as number cruncher any place where each minicomputer is associated by streams during which the parcels of information course through.

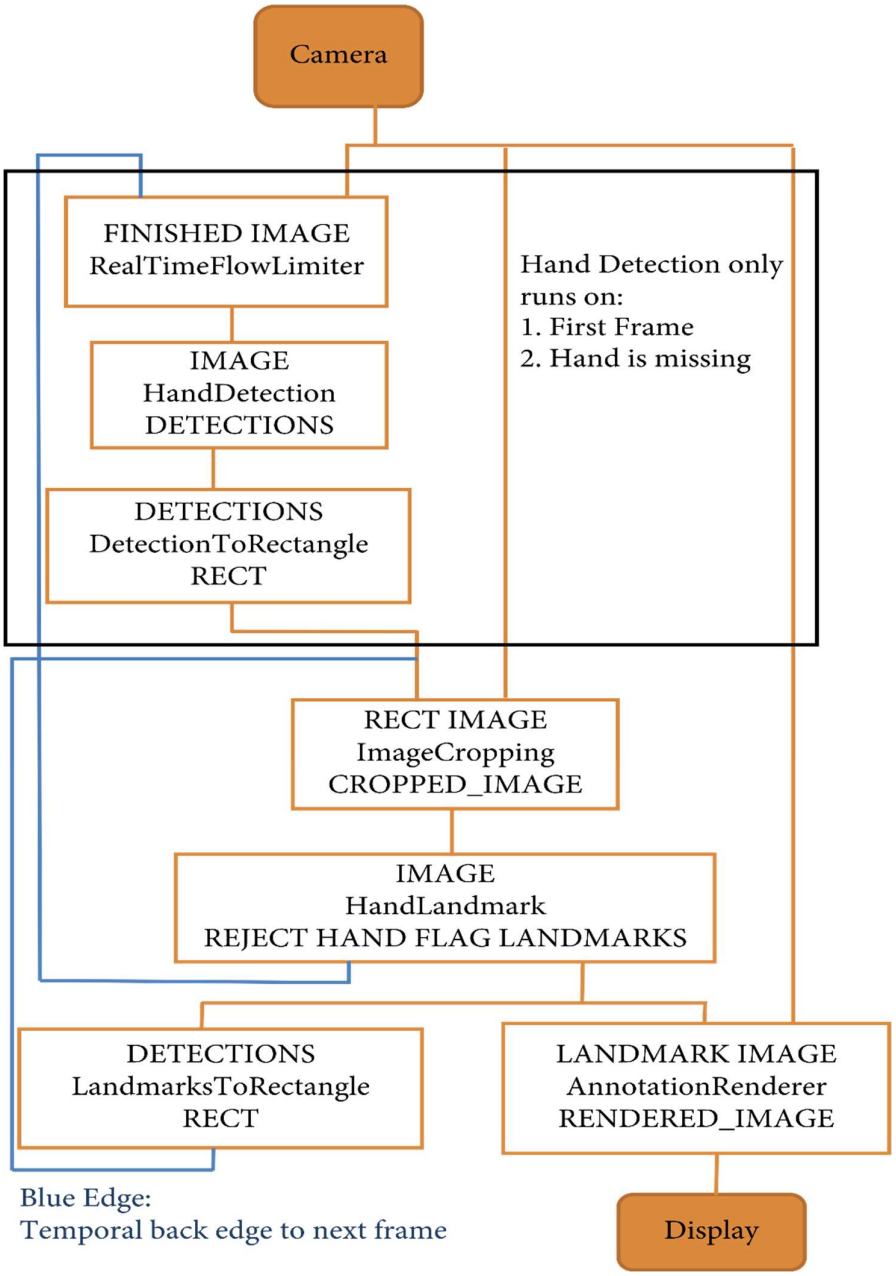

Figure 4.2. Media- Pipe hand tip recognition layout

Single shot is utilized for location and perceiving a finger and palm progressively exploitation journal PC net cam. Finder framework is utilized by the Media Pipe, in the Hand discovery module of python, its style for a finger and hand recognition model because of its easy to mentor hand. The planned model of hand reason mark comprises of 21 joint reason and co-ordinates inside the hand, as displayed in Fig4.3.

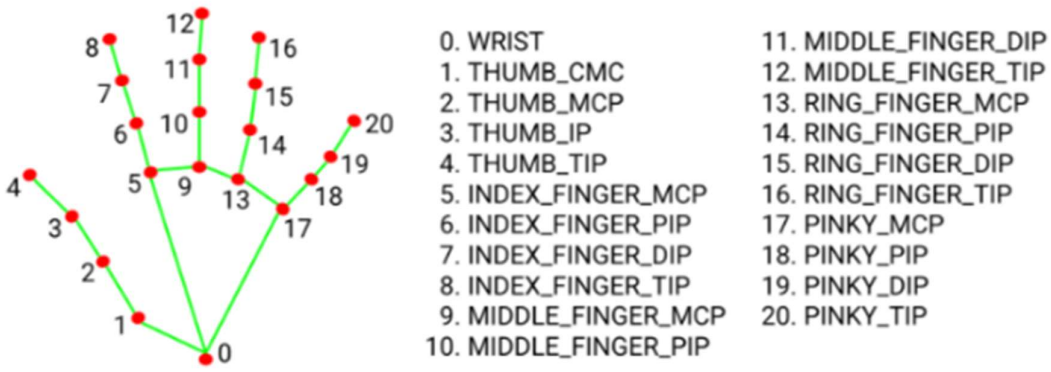

Figure 4.3. Co- ordinates of hand dots.

4.3. Open-CV MODULE

PC vision is an interaction by which we can comprehend the pictures and recordings, how they are put away and how we can control and recover information from them. PC Vision is the base or generally utilized for Artificial Intelligence. The primary Open- CV form was 1.0. Open-CV is delivered under a BSD permit and thus it's free for both scholar and business use. It has C++, C, Python and Java connection points and supports Windows, Linux, Mac OS, iOS and Android. At the point when Open-CV was planned the fundamental center was continuous applications for computational productivity.

1. **Pseudo code algorithm for edge-detection**
   a. Start

   b. Import pyhon OpenCV 3- input section.

   define the normal value for ‘A’ ‘A’ value to be divided by 500.
   define the edge algorithm parameters - picture and intensity.
   define height X and width Y of an picture define the edge.

c. Recognizing section

for all height X and width Y pixels in range extract pixel values
top and bottom
left and right
top_left and top_right bottom_left and bottom_right extract differences
difference I = top minus bottom difference II = left minus right
extract total diff
total diff. = diff. I + diff. II
total diff. = normal (total diff.) * intensity extract pixels of the image
picture_pix = image [X , Y] extract edge_image
edge_picture [X , Y] = picture_pix * total diff

d. Output

Display input picture
Display input picture converted to
gray scale Display edge

e. End

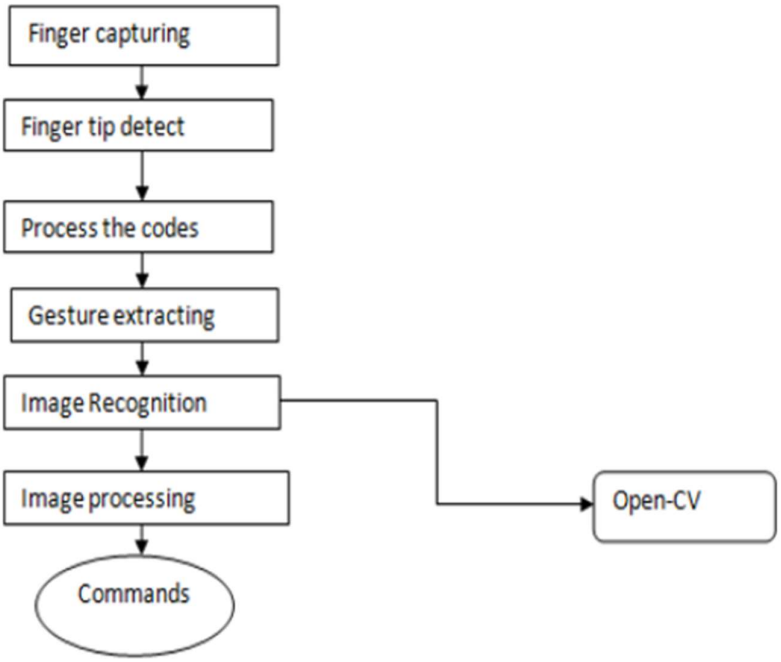

Figure 4.4. Flow Chart

4.4. METHODOLOGY

Pre-processing or to be specific picture handling is an earlier advance in PC vision, where the objective is to change over a picture into a structure reasonable for additional investigation. Instances of tasks, for example, openness rectification, shading adjusting, picture sound decrease, or expanding picture sharpness are exceptionally significant and very consideration requesting to accomplish adequate outcomes.

For this article, I propose to introduce a part of the typically used picture taking care of methodology using an outstandingly notable Computer Vision library, Open-CV. I'll endeavor to portray immediately how each movement works and spotlight more on dealing with the point even more basically, giving you all the code, you truly need so you have a functioning experience of the material.

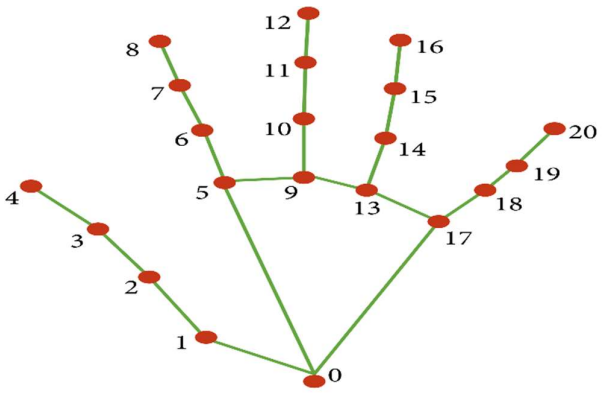

Figure 4.5. Working Co-ordinates

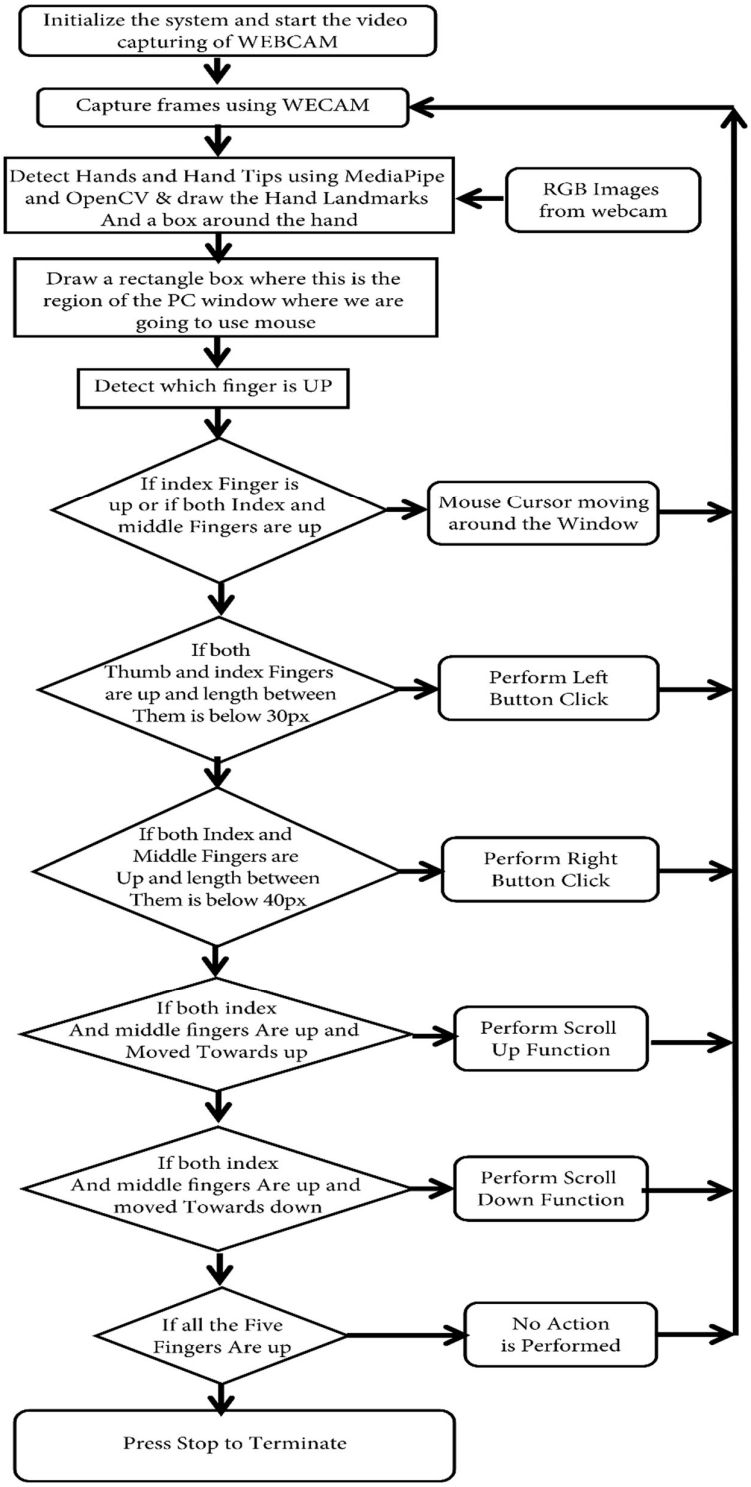

Figure 4.6. Flowchart of the real-time AI virtual mouse system.

CHAPTER 5

UI DESIGN

5.1. CAMERA USED IN THE VIRTUAL GESTURE MOUSE

Open-CV is python vision library that contains Associate in the organized AI virtual mouse system depends upon the edges that are gotten by the camera in Associate in nursing passing PC. Pictures can be conveyed in concealing layered with 3 channels, Grayscale with pixel values fluctuating from 0 (dull) to 255 (white), and twofold portraying dim or white characteristics (0 or 1) specifically.

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.

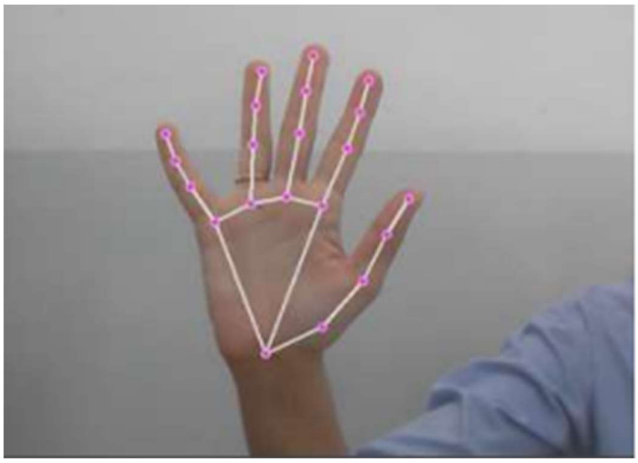

Figure 5.1 Camera Used in the Virtual Gesture

5.2. MOVING HAND THROUGH THE WINDOW USING RECTANGULAR AREA

The AI virtual mouse framework utilizes the instructive algorithmic rule, and it changes over the co-ordinates of tip from the camera screen to the pc window full screen for the mouse. Whenever the hands unit saw and keeping in mind that we've missing to see that finger is up for topic the specific mouse performs, associate in nursing rectangular box is attracted concerning the pc window at ranges the camera locale any spot we've a penchant to will every now and again move all through the window plan the mouse pointer, as displayed fig.

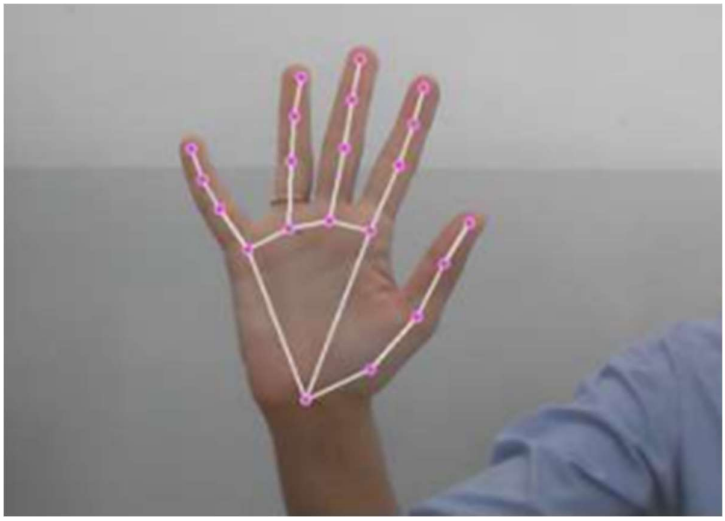

Figure 5.2 Moving Hand Gesture

5.3. 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 less than 30px and like both the tips get attached, the computer is made to perform the left mouse button.

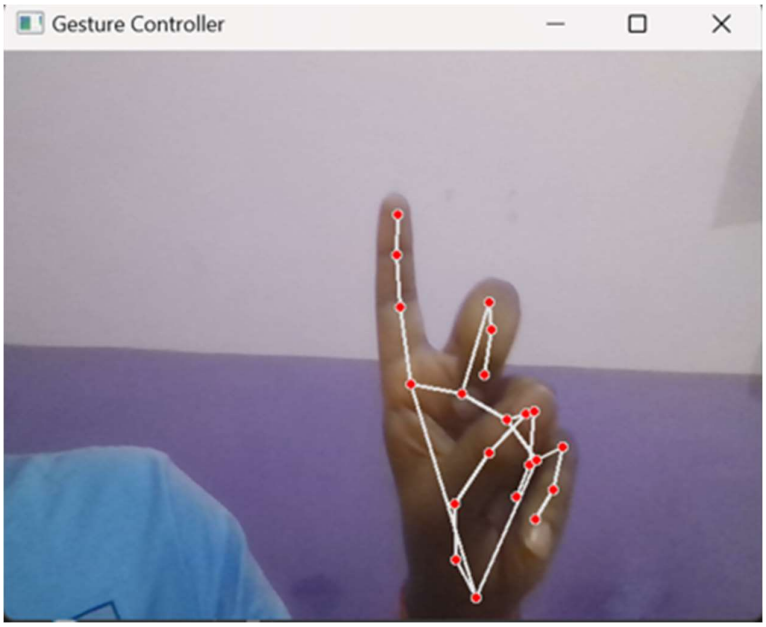

Figure 5.3 Left Button Gesture

5.4. FOR THE MOUSE TO PERFORM RIGHT BUTTON CLICK

If 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 greater than 40px, rest of the finger are in downward direction, the computer is made to perform the right mouse button click.

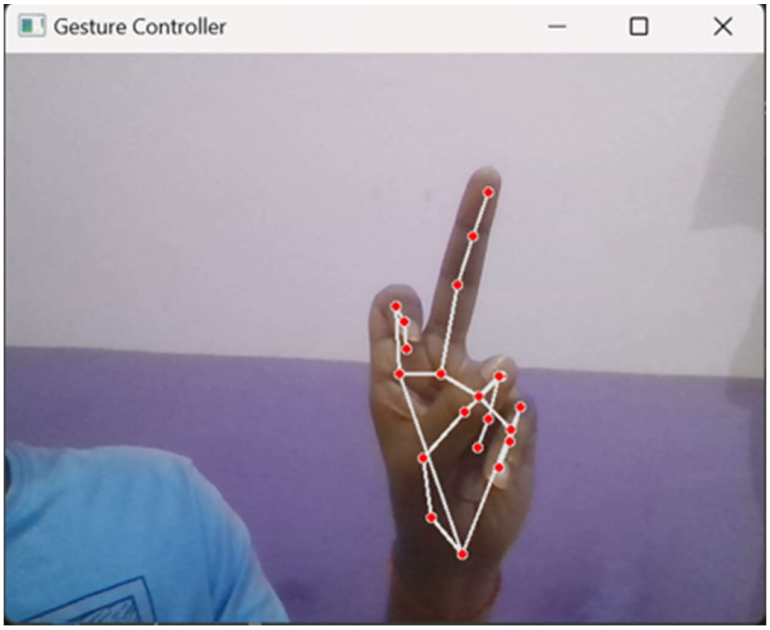

Figure 5.4 Right Button Gesture

5.5. FOR THE MOUSE TO PERFORM DOUBLE CLICK

If 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 40px, and other finger are also in upward direction but the distance between them is greater than 40px, the computer is made to perform the double click.

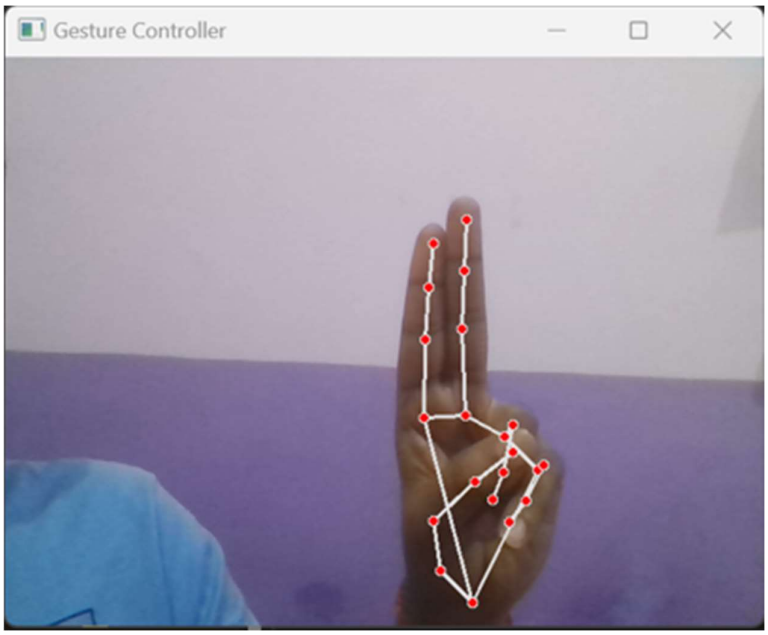

Figure 5.5 Double click Gesture.

5.6. 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.

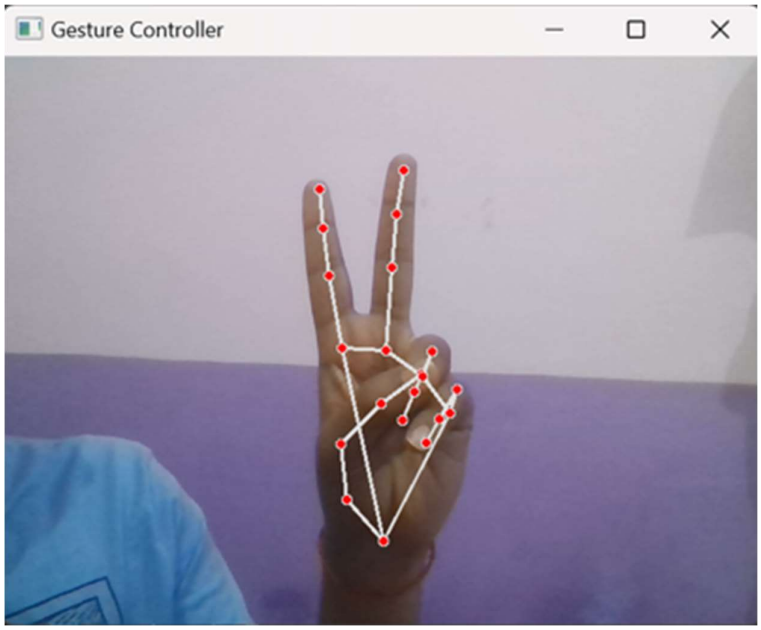

Figure 5.6 Mouse Cursor Moving Gesture

5.7. FOR CONTROLLING BRIGHTNESS, VOLUME, OR SCROLLING

Pinch gesture is commonly associated with tasks such as zooming or resizing objects, it is not typically used for controlling brightness, volume, or scrolling in a virtual mouse context. These functions are usually controlled through different gestures or input methods. Let's explore how these tasks are typically accomplished:

1. **Controlling Brightness:** Adjusting the brightness of a display is often achieved through dedicated keys on a keyboard, function keys, or software controls in the operating system. Some devices also have ambient light sensors that automatically adjust the brightness based on the environment.

2. **Scrolling:** Scrolling up and down on a virtual mouse is typically performed through other methods. In touch-enabled devices, you can typically scroll by swiping your finger up or down on the screen or using dedicated scroll bars. In a virtual mouse context, scrolling is often achieved by moving the mouse pointer over a scrollable area and using the mouse wheel or trackpad gestures, such as two-finger swipe or two-finger scrolling.

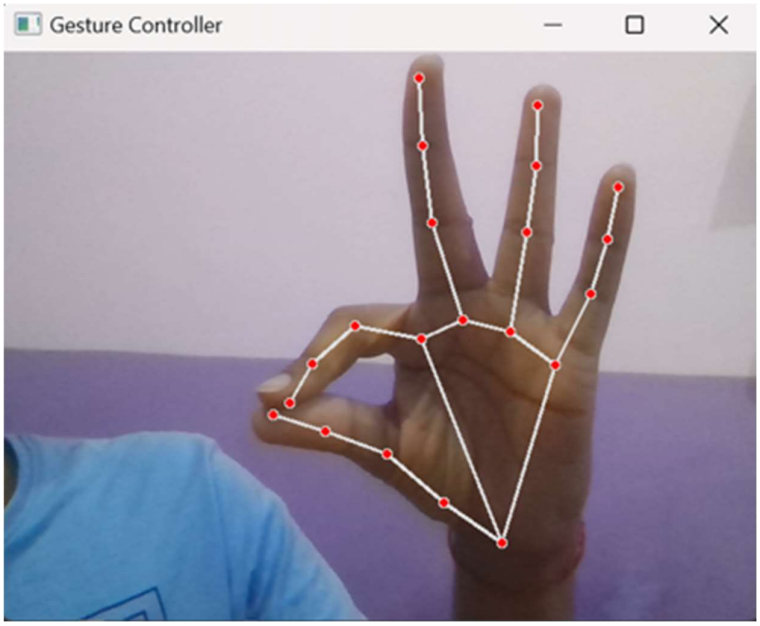

Figure 5.7 Pinch Gestures

CHAPTER 6

CODING

```
# Imports

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

pyautogui.FAILSAFE = False
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

# 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

# Convert Mediapipe Landmarks to recognizable Gestures
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)
      return dist

  def get_dz(self,point):
      return abs(self.hand_result.landmark[point[0]].z –
self.hand_result.landmark[point[1]].z)

  # Function to find Gesture Encoding using current finger_state.
  # Finger_state: 1 if finger is open, else 0
  def set_finger_state(self):
      if self.hand_result == None:
          return

      points = [[8,5,0],[12,9,0],[16,13,0],[20,17,0]]
      self.finger = 0
      self.finger = self.finger | 0 #thumb
      for idx,point in enumerate(points):

          dist = self.get_signed_dist(point[:2])
          dist2 = self.get_signed_dist(point[1:])

          try:
              ratio = round(dist/dist2,1)
          except:
              ratio = round(dist1/0.01,1)

          self.finger = self.finger << 1
          if ratio > 0.5 :
              self.finger = self.finger | 1


  # Handling Fluctations due to noise
  def get_gesture(self):
```

```
if self.hand_result == None:
        return Gest.PALM

    current_gesture = Gest.PALM
    if self.finger in [Gest.LAST3,Gest.LAST4] and self.get_dist([8,4]) < 0.05:
        if self.hand_label == HLabel.MINOR :
            current_gesture = Gest.PINCH_MINOR
        else:
            current_gesture = Gest.PINCH_MAJOR

    elif Gest.FIRST2 == self.finger :
        point = [[8,12],[5,9]]
        dist1 = self.get_dist(point[0])
        dist2 = self.get_dist(point[1])
        ratio = dist1/dist2
        if ratio > 1.7:
            current_gesture = Gest.V_GEST
        else:
            if self.get_dz([8,12]) < 0.1:
                current_gesture =  Gest.TWO_FINGER_CLOSED
            else:
                current_gesture =  Gest.MID

    else:
        current_gesture =  self.finger

    if current_gesture == self.prev_gesture:
        self.frame_count += 1
    else:
        self.frame_count = 0

    self.prev_gesture = current_gesture

    if self.frame_count > 4 :
        self.ori_gesture = current_gesture
    return self.ori_gesture
```

```
# Executes commands according to detected gestures
class Controller:
    tx_old = 0
    ty_old = 0
    trial = True
    flag = False
    grabflag = False
    pinchmajorflag = False
    pinchminorflag = False
    pinchstartxcoord = None
    pinchstartycoord = None
    pinchdirectionflag = None
    prevpinchlv = 0
    pinchlv = 0
    framecount = 0
    prev_hand = None
    pinch_threshold = 0.3

    def getpinchylv(hand_result):
        dist = round((Controller.pinchstartycoord - hand_result.landmark[8].y)*10,1)
        return dist

    def getpinchxlv(hand_result):
        dist = round((hand_result.landmark[8].x - Controller.pinchstartxcoord)*10,1)
        return dist

    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_brig
htness())
```

```
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')
        pyautogui.scroll(-120 if Controller.pinchlv>0.0 else 120)
        pyautogui.keyUp('ctrl')
        pyautogui.keyUp('shift')

    # Locate Hand to get Cursor Position
    # Stabilize cursor by Dampening
    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):
        Controller.pinchstartxcoord = hand_result.landmark[8].x
        Controller.pinchstartycoord = hand_result.landmark[8].y
        Controller.pinchlv = 0
        Controller.prevpinchlv = 0
        Controller.framecount = 0

    # Hold final position for 5 frames to change status
    def pinch_control(hand_result, controlHorizontal, controlVertical):
        if Controller.framecount == 5:
            Controller.framecount = 0
            Controller.pinchlv = Controller.prevpinchlv

            if Controller.pinchdirectionflag == True:
                controlHorizontal() #x

            elif Controller.pinchdirectionflag == False:
                controlVertical() #y

        lvx =  Controller.getpinchxlv(hand_result)
        lvy =  Controller.getpinchylv(hand_result)

        if abs(lvy) > abs(lvx) and abs(lvy) > Controller.pinch_threshold:
            Controller.pinchdirectionflag = False
            if abs(Controller.prevpinchlv – lvy) < Controller.pinch_threshold:
                Controller.framecount += 1
            else:
```

```
            Controller.prevpinchlv = lvy
            Controller.framecount = 0

        elif abs(lvx) > Controller.pinch_threshold:
            Controller.pinchdirectionflag = True
            if abs(Controller.prevpinchlv – lvx) < Controller.pinch_threshold:
                Controller.framecount += 1
            else:
                Controller.prevpinchlv = lvx
                Controller.framecount = 0

    def handle_controls(gesture, hand_result):
        x,y = None,None
        if gesture != Gest.PALM :
            x,y = Controller.get_position(hand_result)

        # flag reset
        if gesture != Gest.FIST and Controller.grabflag:
            Controller.grabflag = False
            pyautogui.mouseUp(button = “left”)

        if gesture != Gest.PINCH_MAJOR and Controller.pinchmajorflag:
            Controller.pinchmajorflag = False

        if gesture != Gest.PINCH_MINOR and Controller.pinchminorflag:
            Controller.pinchminorflag = False

        # implementation
        if gesture == Gest.V_GEST:
            Controller.flag = True
            pyautogui.moveTo(x, y, duration = 0.1)

        elif gesture == Gest.FIST:
            if not Controller.grabflag :
                Controller.grabflag = True
                pyautogui.mouseDown(button = “left”)
            pyautogui.moveTo(x, y, duration = 0.1)
```

```
 elif gesture == Gest.MID and Controller.flag:
            pyautogui.click()
            Controller.flag = False

        elif gesture == Gest.INDEX and Controller.flag:
            pyautogui.click(button='right')
            Controller.flag = False

        elif gesture == Gest.TWO_FINGER_CLOSED and Controller.flag:
            pyautogui.doubleClick()
            Controller.flag = False

        elif gesture == Gest.PINCH_MINOR:
            if Controller.pinchminorflag == False:
                Controller.pinch_control_init(hand_result)
                Controller.pinchminorflag = True
            Controller.pinch_control(hand_result,Controller.scrollHorizontal, Controller.scr
ollVertical)

        elif gesture == Gest.PINCH_MAJOR:
            if Controller.pinchmajorflag == False:
                Controller.pinch_control_init(hand_result)
                Controller.pinchmajorflag = True
            Controller.pinch_control(hand_result,Controller.changesystembrightness, Contr
oller.changesystemvolume)

'''
---------------------------------------- Main Class ----------------------------------------
    Entry point of Gesture Controller
'''

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_F
RAME_HEIGHT)
        GestureController.CAM_WIDTH = GestureController.cap.get(cv2.CAP_PROP_F
RAME_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

        try:
            handedness_dict = MessageToDict(results.multi_handedness[1])
            if handedness_dict['classification'][0]['label'] == 'Right':
                right = results.multi_hand_landmarks[1]
            else :
                left = results.multi_hand_landmarks[1]
        except:
            pass

        if GestureController.dom_hand == True:
            GestureController.hr_major = right
            GestureController.hr_minor = left
        else :
            GestureController.hr_major = left
            GestureController.hr_minor = right

    def start(self):
```

```
        handmajor = HandRecog(HLabel.MAJOR)
        handminor = HandRecog(Hlabel.MINOR)

        with mp_hands.Hands(max_num_hands = 2,min_detection_confidence=0.5, min_t
racking_confidence=0.5) as hands:
            while GestureController.cap.isOpened() and GestureController.gc_mode:
                success, image = GestureController.cap.read()

                if not success:
                    print("Ignoring empty camera frame.")
                    continue

                image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)

                image.flags.writeable = False
                results = hands.process(image)

                image.flags.writeable = True
                image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

                if results.multi_hand_landmarks:
                    GestureController.classify_hands(results)
                    handmajor.update_hand_result(GestureController.hr_major)
                    handminor.update_hand_result(GestureController.hr_minor)

                    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.HAN
D_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()
```

CHAPTER 7

EXPERIMENTAL RESULTS AND EVALUATION

7.1. 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 fingertip 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 7.1. The test was performed 25 times by 4 persons resulting in 600 gestures with manual labelling, 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-7.1.

Mouse function performed	Success	Failure	Accuracy (%)
Mouse movement	100	0	100%
Left button click	98	2	98%
Right buttons click	99	1	99%
Scroll function	93	7	93%
Brightness control	95	5	95%
Volume control	96	4	96%
No action performed	100	0	100%
Result	681	19	97.28%

Table 7.1 Experimental Result

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 7.1.

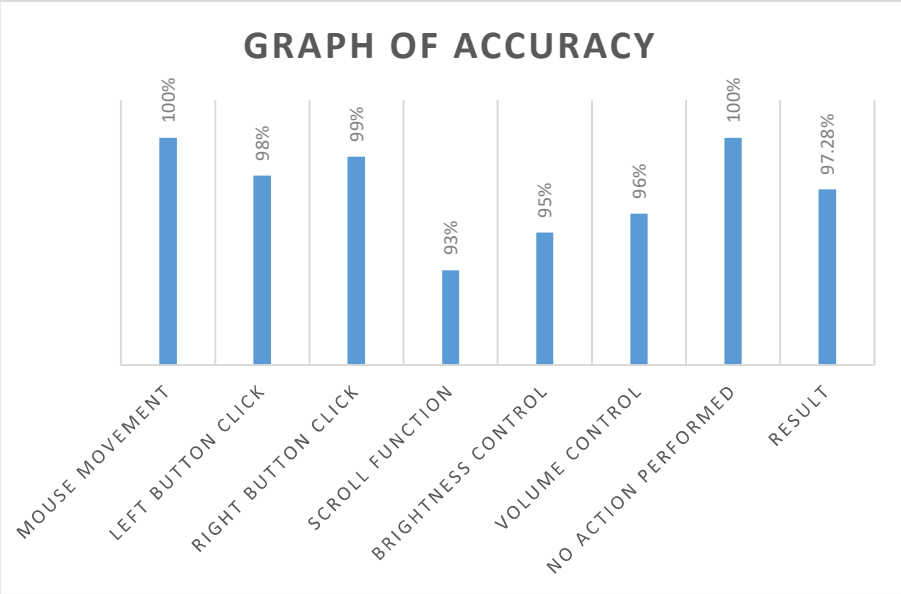

Figure 7.1 Graph of Accuracy

Table 7.2 shows a comparison between the existing models and the proposed AI virtual mouse model in terms of accuracy.

Existing models	Accuracy (%)
Virtual mouse system using RGB-D images and fingertip detection [16]	96.13
Palm and finger recognition based [17]	78
Hand gesture-based virtual mouse [18]	78
The proposed AI virtual mouse system	99

Table 7.2

From Table 7.2, 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 fingertip detection, and also, the model is helpful in controlling the PC like a physical mouse but in the virtual mode. Figure 7.2 shows a graph of comparison between the models.

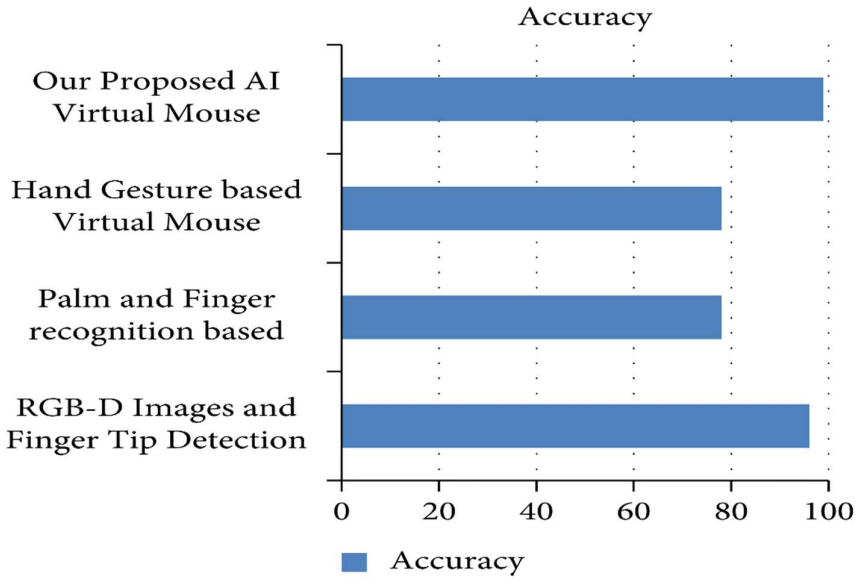

Figure 7.2

CHAPTER 8

TESTING

8.1. 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 environments, software should work fine as it easily detects the hand movements but in a 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, there will be some problem in detecting hand gestures, but it should work fine.	At this distance, the function of this software works properly.	Passed

CHAPTER 9

APPLICATIONS

9.1. OVERVIEW

The AI virtual mouse system is useful for many applications; it can be used to reduce the space for using the physical mouse, and it can be used in situations where we cannot use the physical mouse. The system eliminates the usage of devices, and it improves human-computer interaction.

9.2. MAJOR APPLICATIONS:

1. The proposed model has a greater accuracy of 99% which is far greater than the that of other proposed models for virtual mouse, and it has many applications.
2. Amidst the COVID-19 situation, it is not safe to use the devices by touching them because it may result in a possible situation of spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to control the PC mouse functions without using the physical mouse.
3. The system can be used to control robots and automation systems without the usage of devices.
4. 2D and 3D images can be drawn using the AI virtual system using the hand gestures.
5. AI virtual mouse can be used to play virtual reality- and augmented reality-based games without the wireless or wired mouse devices.
6. Persons with problems in their hands can use this system to control the mouse functions in the computer.
7. In the field of robotics, the proposed system like HCI can be used for controlling robots.
8. In designing and architecture, the proposed system can be used for designing virtually for prototyping.

CHAPTER 10

CONCLUSIONS

10.1. OVERVIEW

The main objective of the AI virtual mouse system is to control the mouse cursor functions by using hand gestures instead of using a physical mouse. The proposed system can be achieved by using a webcam or a built-in camera which detects the hand gestures and hand tip and processes these frames to perform the particular mouse functions.

From the results of the model, we can come to a conclusion that the proposed AI virtual mouse system has performed very well and has a greater accuracy compared to the existing models and also the model overcomes most of the limitations of the existing systems. Since the proposed model has greater accuracy, the AI virtual mouse can be used for real-world applications and also it can be used to reduce the spread of COVID-19, since the proposed mouse system can be used virtually using hand gestures without using the traditional physical mouse.

The model has some limitations such as small decrease in accuracy in right click mouse function and some difficulties in clicking and dragging to select the text. Hence, we will work next to overcome these limitations by improving the fingertip detection algorithm to produce more accurate results.

10.2. LIMITATION

In this project, there are several existing problems that may hinder the results of color recognition. One of the problems is the environmental factor during the recognition phase that takes place. The recognition process is highly sensitive on the intensity of brightness, as immense brightness or darkness may cause the targeted colors to be undetected within the captured frames. Besides that, distance is also the one of the problems that may affect the color recognition results, as the current detection region can support up to 25cm radius, any display of colors exceed the mentioned distance will be considered as a noise and be filtered off.

Furthermore, the performance of the program is highly dependent on the users' hardware, as processor speed and/or resolutions taken from the webcam could have an effect on performance load. Therefore, the slower the processing speed and/or the higher the resolutions, the longer time is required to process a single frame.

10.3. FUTURE WORKS

There are several features and improvements needed 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 Recognition Algorithm**
   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 Performance**
   The response time is heavily relied 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 lighting.

REFERENCES

1. J. Katona, "A review of human–computer interaction and virtual reality research fields in cognitive Info Communications," Applied Sciences, vol. 11, no. 6, p. 2646, 2021.View at: Publisher Site | Google Scholar
2. D. L. Quam, "Gesture recognition with a Data Glove," IEEE Conference on Aerospace and Electronics, vol. 2, pp. 755–760, 1990. View at: Publisher Site | Google Scholar
3. D.-H. Liou, D. Lee, and C.-C. Hsieh, "A real time hand gesture recognition system using motion history image," in Proceedings of the 2010 2nd International Conference on Signal Processing Systems, IEEE, Dalian, China, July 2010. View at: Publisher Site | Google Scholar
4. S.U.Dudhane, "Cursor control system using hand gesture recognition," IJARCCE, vol. 2, no. 5, 2013. View at: Google Scholar
5. K. P. Vinay, "Cursor control using hand gestures," International Journal of Critical Accounting, vol. 0975–8887, 2016. View at: Google Scholar
6. L. Thomas, "Virtual mouse using hand gesture," International Research Journal of Engineering and Technology (IRJET, vol. 5, no. 4, 2018. View at: Google Scholar
7. P. Nandhini, J. Jaya, and J. George, "Computer vision system for food quality evaluation—a review," in Proceedings of the 2013 International Conference on Current Trends in Engineering and Technology (ICCTET), pp. 85–87, Coimbatore, India, July 2013. View at: Publisher Site | Google Scholar
8. J. Jaya and K. Thanushkodi, "Implementation of certain system for medical image diagnosis," European Journal of Scientific Research, vol. 53, no. 4, pp. 561–567, 2011. View at: Google Scholar.
9. P. Nandhini and J. Jaya, "Image segmentation for food quality evaluation using computer vision system," International Journal of Engineering Research and Applications, vol. 4, no. 2, pp. 1–3, 2014. View at: Google Scholar.

10. J. Jaya and K. Thanushkodi, “Implementation of classification system for medical images,” European Journal of Scientific Research, vol. 53, no. 4, pp. 561–569, 2011.View at: Google Scholar
11. J. T. Camillo Lugaresi, “MediaPipe: A Framework for Building Perception Pipelines,” 2019, https://arxiv.org/abs/1906.08172. View at: Google Scholar
12. Google, MP, https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html.
13. V. Bazarevsky and G. R. Fan Zhang. On-Device, MediaPipe for Real-Time Hand Tracking.
14. K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, “Realtime computer vision with OpenCV,” Queue, vol. 10, no. 4, pp. 40–56, 2012. View at: Publisher Site | Google Scholar
15. https://www.tutorialspoint.com› OpenCV.
16. 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.View at: Publisher Site | Google Scholar
17. A Haria, A. Subramanian, N. Asokkumar, S. Poddar, and J. S. Nayak, “Hand gesture recognition for human computer interaction,” Procedia Computer Science, vol. 115, pp. 367–374, 2017. View at: Publisher Site | Google Scholar
18. K. H. Shibly, S. Kumar Dey, M. A. Islam, and S. Iftekhar Showrav, “Design and development of hand gesture based virtual mouse,” in Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), pp. 1–5, Dhaka, Bangladesh, May 2019.View at: Publisher Site | Google Scholar