**ADVANCED AI VIRTUAL MOUSE**

**Abbreviations used**

|  |  |
| --- | --- |
| GUI | Graphical User Interface |
| HCI | Human Computer Interaction |
| MHI | Motion History Images |
| IDE | Integrated Development Environment |
| OpenCV | Open Source Computer Vision |
| NUI | Natural User Interface |

**ABSTRACT**

The PC mouse is one of the wondrous developments of people in the field of Human-Computer Interaction (HCI) innovation. In new age of innovation, remote mouse or a contact less mouse actually utilizes gadgets and isn't liberated from gadgets completely, since it utilizes power from the gadget or might be from outside power sources like battery and gain space and electric power, and also during COVID pandemic it is encouraged to make social separating and keep away from contacting things. Inside the projected AI virtual mouse utilizing hand signal framework, this constraint might be resolve by involving web camera or integrated camera to perceive the hand motions and fingers recognition. The algorithmic rule used in the framework utilizes the man-made consciousness and AI algorithmic rule. Upheld the hand signals, the gadget can be controlled well and can do left click, right click, and all PC gadget cursor perform while not the utilizing of the genuine mouse.

The proposed system will only require a webcam as an input device. The output from the camera will be displayed on the system’s screen so that it can be further calibrated by the user. The python dependencies that will be used for implementing this system are NumPy, math, and mouse.

This project consists of three main steps that are hand gesture tracking, features of hand region extraction and classification of these features. In this project hand gesture tracking is generated with Camshift (Continuously Adaptive Mean Shift) algorithm, features of hand gestures are extracted with bag of visual words. This also helps in advancing of AI implementation and computer vision.

Table of Contents

**INTRODUCTION1**

A. PROBLEM STATEMENT2

B. OBJECTIVE2

C. RelatedWork………………………………………………………………………………………………………………………………2

**LITERATURE SURVEY4**

ALGORITHMS USED5

A. GESTURE TRACKING6

B. MEDIA PIPE…………………………………………………………………………………………………………………………7

C.OPENCV MODULE………………………………………………………………………………………………………………..8

**SYSTEM DEVELOPMENT………………………………………………………………………..9**

\*Algorithm and techniques used………………………………………………………..10

\*coding………………………………………………………………………………………………11

\*outputs……………………………………………………………………………………………18

**PERFORMANCE ANALYSIS ……………………………………………………………………………………………….20**

**CONCLUSION AND FUTURE SCOPE……………………………………………………………………………………25**

**TEST CASES………………………………………………………………………………………………………………………27**

**REFERENCES…………………………………………………………………………………………………………………….28**

**INTRODUCTION**

In the new present day progresses at extends the areas of exaggerated reality and contraptions that we will generally will as a rule use in our existence, these devices have gotten diminished at ranges the sort of Bluetooth or far off advancements. This paper proposes Associate in nursing AI virtual mouse system that produces use of the hand signals and hand tip acknowledgment for performing articulations mouse limits at ranges the advantageous adaptable PC vision. The most impartial of the projected system is to perform device pointer works and material performs using a web camera or a characteristic camera at extends the smaller PC rather than using an obsolete mouse contraption.

Also, hand tip area by use helpful as PC vision is used as a HCI with the PC. With the usage of the AI virtual mouse system, we will follow the tip of the hand signal by using an intrinsic camera or net camera and play out the mouse pointer assignments and investigating work and together move the pointer with it.

While utilizing a remote or a Bluetooth mouse, a few gadgets particularly like the mouse, the contraption to interface with the pc, and also, battery to drive the mouse to control a utilized, yet all through this paper, the client utilizes his/her inbuilt camera or visual camera and utilizations his/her hand signs to deal with the PC mouse works out. Inside the projected construction, data interstate camera hinders this cycle the edges that square measure got accordingly see the different hand signals and hand tip developments shapes the specific mouse work.

Python programming language is utilized for empowering the AI virtual mouse structure, Open CV the library for versatile PC vision is utilized at ranges for the AI virtual mouse framework. Inside the projected AI virtual mouse utilizing hand signal, the model purposes the python Media-pipe for the hand tracking of the hands and to detect the tip of the hands, math and PyAutoGUI packages were utilized for propelling the screen of the PC for performing verbalizations limits like left click, right click, and others. There the projected model show in a perfect world high accuracy level, and in this way the projected model can work respectably in clear application with the use of a cycle or where as not the utilization of PC GPU.

*A. Problem Statement*

The COVID circumstance, it isn't safeguarded to include the devices by reaching them as an eventual outcomes of the intention 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.

*B. Objective*

The fundamental target of the coordinated AI virtual Mouse steamiest urge substitute to the common and obsolete mouse construction to perform and the board as far as possible, and this could be accomplished with the assistance of an inside net camera that gets the hand developments and hand tip then, at that point, processes these lodgings to play out the specific mouse performs like left click, right click, and other performances. Thus, the projected structure will avoid COVID-19 spread out by discarding the human intervention and dependence of devices to control the PC.

Virtual Mouse Hand movement affirmation licenses clients to supervise mouse with the assistance of hand movements. System's automated web camera is used for following hand movements. PC vision systems were used for Motion affirmation. Open-CV python group includes as video get that is used to get information from a live video, essential issue we really want to perceive the applications the model goes to develop so the headway of the mouse improvement without reaching or using of the mouse.

*C. Related Work*

Here unit a few associated works allotted on virtual mouse exploitation hand motion location by wearing the hand gloves and likewise covering finger tips among the hands for motion acknowledgment, however they are presently further right in mouse capacities. The acknowledgment isn't in this way right as after effect of wearing gloves; also, those special gloves don't appear to be fitted to certain clients, and at times, the acknowledgment isn't subsequently right as consequences of the mistake of area of concealing tips.In1991,Quam presented partner early equipment based structure; all through this strategy, the client got to wear a data Glove. The extended structure by Quam gives consequences of higher exactness; assembled strong to perform assortment of the signal controls exploitation the framework. Neha ,Parul gandhi and Ashwini in 2014 extended a concentrate on "Cursor system abuse Hand Gesture Recognition."

Chaitanya thomas, Naveen kr, and Abhilash seth in 2019 expected "Virtual Mouse double-dealing Hand Gesture" where the model area relies upon colors. In any case, alone barely any mouse limits square measure performed.

**LITERATURE SURVEY**

The current construction is contained a nonexclusive mouse and track pad screen control framework, as well as the hand development control structure. The utilization of a hand development to get to the screen from a nice ways 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 click, and drag etc The utilization of hand confirmation in the future won't be utilized. Despite how they are 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:-

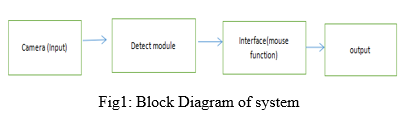
1. Camera Used in the AI Virtual Mouse project: Open- CV is python vision library that contains Associate in the organized AI virtual mouse structure depends upon the edges that are gotten by the camera in Associate in nursing passing computer.
2. Providing Input: Pictures in Computer Vision are portrayed 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 plots 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 Finger tips and doing the Mouse Cursor improvements
5. In this construction, AI mouse is police evaluation that finger is up deciding the spot co-ordinate of the particular finger that it'll find 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 has 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 endeavour that won't hurt the client's prosperity.

**ALGORTHIM USED For AI VIRTUAL MOUSE**

A. Gesture Tracking

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.



B. What is Media-Pipe?

Media-Pipe is a system that is utilized for applying in a different AI pipeline, partner with an open source structure of Google. The Media-Pipe system is useful for across stage improvement since the edge work is made 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 are 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 each mini-computer is associated by streams during which the parcels of information course through.

The number cruncher and streams joined produce an information stream outline; the diagram is made with Media-Pipe any place each hub might be an adding machine and thus the hubs square measure associated by stream.

*The* proposed AI virtual mouse system is based on the framesthat have been captured by the webcam in a laptop or PC. Byusing the Python computer vision library OpenCV, thevideo 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)

*C. 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 python 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

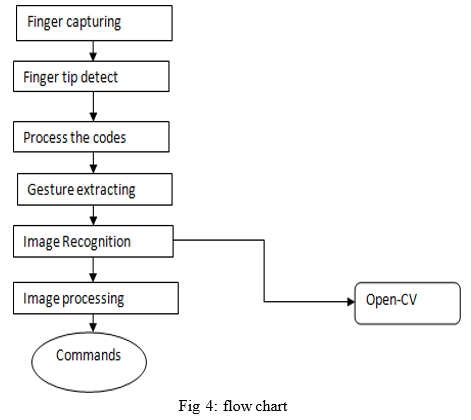
d. Output

Display input picture

Display input picture converted to

gray scale Display edge

e. End



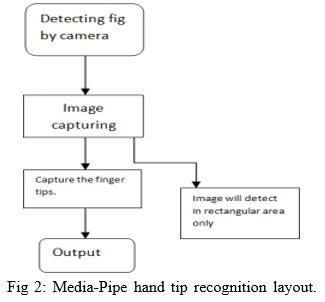
**III. METHODOLOGY**

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.

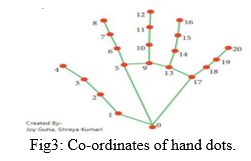
For this article, I propose to introduce a part of the typically used picture taking care of methodology using an outstandingly notable Computer Vision library, Open-CV. I'll endeavor to portray immediately the manner by which each movement works and spotlight more on dealing with the point even more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

*A. Camera Used in the Virtual Gesture Mouse project*

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 (Blue, Green, and Red), Grayscale with pixel values fluctuating from 0 (dull) to 255 (white), and twofold portraying dim or white characteristics (0 or 1) specifically**.**



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 it's easy to mentor hand. The planed model of hand reason mark comprises of 21 joint reason and co-ordinates inside the hand, as displayed in Fig3.



**System Development**

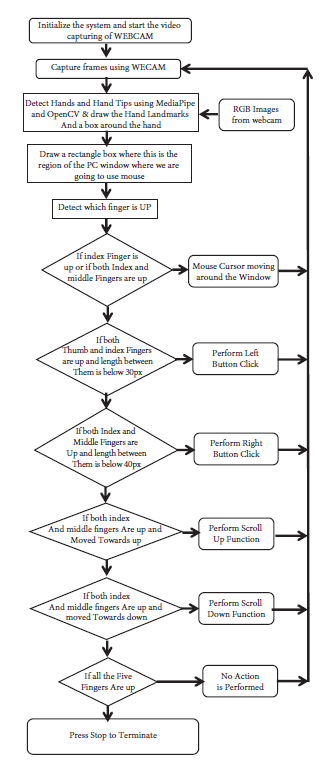
The various functions and conditions used in the system are explained in the flowchart of the real-time AI virtual mouse system in figure.

*Rectangular Region for Moving through the Window.* The AI virtual mouse system makesuse of the transformational algorithm, and it converts the coordinatesof fingertip from the webcam screen to thecomputer 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 whichfinger 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 theindex finger with tip Id = 1 and the middle finger with tipId = 2 are up, the mouse cursor is made to move around thewindow of the computer using the AutoPy package ofPython, as shown in Figure 7.

*For the Mouse to Perform Left Button Click.* If both theindex finger with tip Id = 1 and the thumb finger with tip Id = 0are up and the distance between the two fingers is lesser than30px, the computer is made to perform the left mouse button click using the pynput.

****

**Flow chart of AI Virtual Mouse**

**Algorithm and techniques used**

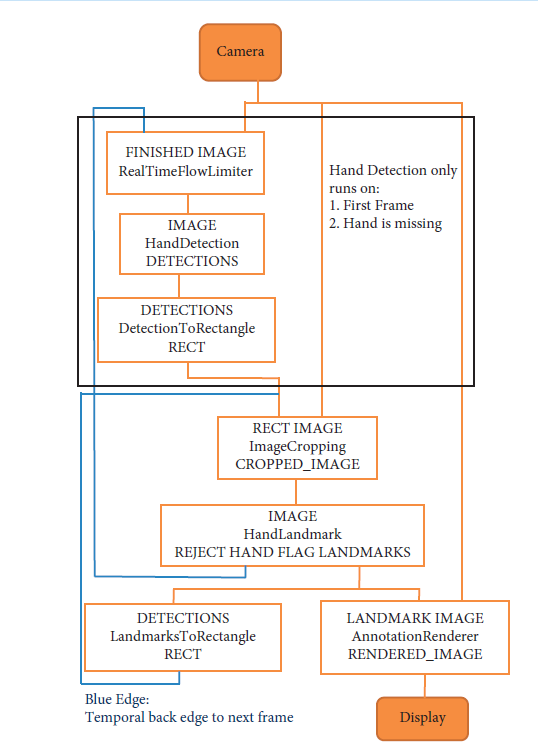
For the purpose of detection of hand gestures and hand tracking, the MediaPipe framework is used, and OpenCV library is used for computer vision. The algorithm makes use of the machine learning concepts to track and

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 .

****

**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\_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')

        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.scrollVertical)

        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, Controller.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\_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

        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\_tracking\_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.HAND\_CONNECTIONS)

                else:

                    Controller.prev\_hand = None

                cv2.imshow('Gesture Controller', image)

                if cv2.waitKey(5) & 0xFF == 13:

                    break

        GestureController.cap.release()

        cv2.destroyAllWindows()

gc1 = GestureController()

gc1.start()

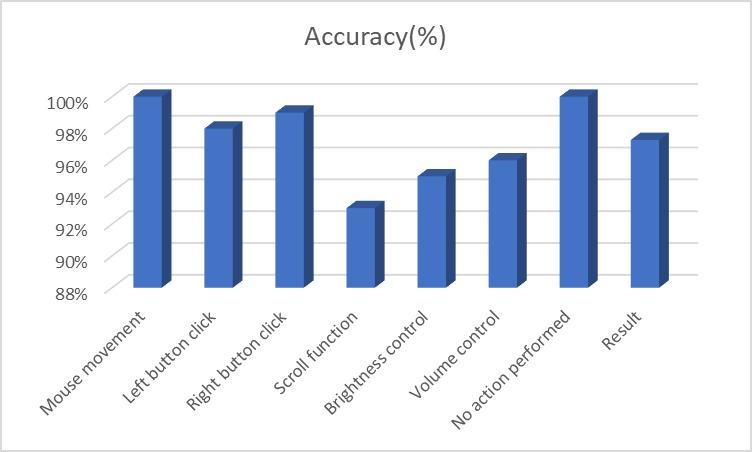
**Performance Analysis**

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 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.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 93% |
| Brightness control | 95 | 5 | 95% |
| Volume control | 96 | 4 | 96% |
| No action performed | 100 | 0 | 100% |
| Result | 681 | 19 | 97.28% |

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   
  
****

**Conclusion and Future Scope**

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 touchscreen concept is replaced by hand gestures.

**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** |

**References**

1) OpenCV Website – www.opencv.org

2) MSDN Microsoft developers network – www.msdn.microsoft.com

3) Code project – www.codeproject.com/Articles/498193/Mouse-Control-via-Webcam

4) Aniket Tatipamula’s Blog - http://anikettatipamula.blogspot.in/2012/02/hand-gesture-using-opencv.html

5) Microsoft Research Paper- http://research.microsoft.com/en-us/um/people/awf/bmvc02/project.pdf