AI ENABLED HAND GESTURE RECOGNITION SYSTEM BASED VIRTUAL MOUSE

A PROJECT REPORT

(PHASE II)

Submitted by

ARUN V **(Register No: 19TUCS009)**

JIJIN SURESH E **(Register No: 19TUCS060)**

JITHIN SURESH E **(Register No: 19TUCS061)**

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Accredited by NBA)

SRI KRISHNA COLLEGE OF TECHNOLOGY

An Autonomous Institution | Accredited by NAAC with 'A' Grade

Affiliated to Anna University | Approved by AICTE

KOVAIPUDUR, COIMBATORE 641042

MARCH 2023

SRI KRISHNA COLLEGE OF TECHNOLOGY
An Autonomous Institution | Accredited by NAAC with 'A' Grade
Affiliated to Anna University | Approved by AICTE
KOVAIPUDUR, COIMBATORE 641042

BONAFIDE CERTIFICATE

Certified that this project report **"AI ENABLED HAND GESTURE RECOGNITION SYSTEM BASED VIRTUAL MOUSE"** is the bonafide work of **"ARUN V., JIJIN SURESH E., JITHIN SURESH E."** who carried out the project work phase I under my supervision.

SIGNATURE
Ms. M.S. SRUTHI
SUPERVISOR
Assistant Professor,
Department of Computer Science
and Engineering
Sri Krishna College of Technology,
Coimbatore-641042.

SIGNATURE
Dr. R. VIDHYA
HEAD OF THE DEPARTMENT
Associate Professor,
Department of computer science
and Engineering
Sri Krishna College of Technology,
Coimbatore-641042.

Certified that the candidates were examined by us in the Project Phase I Viva Voice examination held on _____ at Sri Krishna College Of Technology Kovaipudur, Coimbatore – 641042 .

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

ABSTRACT

The proposed system provides a way to control the positions of the cursor with a bare hand without using a mechanical mouse. This project is made for efficient teaching purpose. The proposed system will only require a webcam as an input device. The software's that will be required to implement the proposed system are OpenCV and python. The output of 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, opencv, autopy, autogui and matplotlib.

The project promotes an approach for the Human Computer Interaction 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 of performing every task that the physical computer mouse can. In this the program will check the gesture of our hands in each frame of the camera using autopy and autogui library. Whenever the process is complete, the program will understand the meaning of the gesture shown and act accordingly.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

This project has been successfully completed owing comprehensive endurance many distinguished persons. First and foremost, we would like to thank the almighty, our family members, and friends for encouraging us to do this project.

We extend our sincere appreciation to **Dr. R. RAMESH KUMAR, M.E., Ph.D.,** the Dean Administration, for his kindness and unwavering support throughout the project work.

We extend our heartfelt thanks to our beloved Dean Academics and Students Affairs **Dr. P. MANJU, M.Tech., Ph.D.,** for her advice and ethics inculcated during the entire period of our study.

We would like to express our deep gratitude to **Dr. R. VIDHYA, M.Tech., Ph.D.,** the Head of the Department of Computer Science and Engineering, for her exceptional dedication and care towards success of this project.

We would like to extend our heartfelt gratitude to **Ms. M.S. SRUTHI, M.E.,** our supervisor, and **Mr. T. RAGHUNATHAN, M.E.,** the Project Co-ordinator of the Department of Computer Science and Engineering, for their mentorship and involvement were crucial in shaping our work and making it a success, and we are deeply grateful to them for their commitment to our project.

We would also like to express our gratitude to the teaching and non-teaching faculty members who supported us during this project. Their contributions and assistance were vital in helping us overcome challenges and achieve our goals.

TABLE OF CONTENT

TABLE OF CONTENT

LIST OF FIGURES

LIST OF FIGURES

INTRODUCTION

CHAPTER 1

INTRODUCTION

It has been generations since we have been using hand gestures for communicating in human society. The shaking of hands, Thumbs up and Thumbs down signs have been ever existing in the environment. It is believed that gestures are the easiest way of interacting with anyone. This project will get deep inside the AI virtual mouse system that produces use of the hand signals and hand tip acknowledgment for performing operations done by a physical mouse. 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 a mechanical mouse.

With the usage of the AI virtual mouse system, we will follow the tip of the hand signal by using an inbuilt camera or a physical camera and play out the mouse pointer assignments.

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.

Python programming language is utilized for empowering the AI virtual mouse structure, Open CV can't avoid being that the library for versatile PC vision is utilized at ranges the AI virtual mouse framework. Inside the projected AI virtual mouse utilizing hand signal, the model purposes the python Media-pipe bunch for the journey for the hands and the pursuit of the tip of the hands, what's more, Numpy, Autopy, and PyAuto GUI packs were utilized for propelling the screen of the PC for performing verbalizations limits like left-click, right-click, and more.

1.1 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 to be prepared to manage a real mouse. Moreover, in the COVID circumstance, it isn't safeguarded to include the devices reaching them as an eventual outcome of its 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 web cam. 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.

1.2 IMAGE PROCESSING

Image processing is a method to perform some operations on an image, to get an enhanced image, and or to extract some useful information from it. If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially to improve its quality”.

1.2.1 Digital image

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point. In another word An image is nothing more than a two-dimensional matrix (3-D in the case of color images) which is defined by the mathematical function f(x, y) at any point giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what color it should be. Image processing is signal processing in which the

input is an image and the output is an image or characteristics according to the requirement associated with that image.

1.2.2 Image reading by a computer

We are humans we can easily make it out that is the image of a person who is me. But if we ask the computer “is it my photo?”. The computer can’t say anything because the computer is not figuring out it all on its own. The computer reads any image in a range of values between 0 and 255. For any color image, there are 3 primary channels - red, green, and blue.

1.3 IMPACT AND CONTRIBUTION

The Virtual Mouse application is expected to replace the current methods of utilizing a physical computer mouse where the mouse inputs and positions are done manually. This application offers a more effortless way to interact with the computer system, where every task can be done by gestures. Furthermore, the Virtual Mouse application could assist the monitor where he/she could interact with the computer system by just showing finger gestures to the webcam. Video conferencing is very popular nowadays. For this reason, most computer users use a webcam on their computer and most laptops have a built-in webcam. The proposed system which is webcam-based might be able to eliminate the need for a mouse partially. The process of interaction with a computer using hand gestures is a very interesting & effective approach to HCI (Human-Computer Interaction). There is some really good research on this interest. Hand gesture recognition technology is also popular in sign language recognition.

LITERATURE REVIEW

CHAPTER 2

LITERATURE REVIEW

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

2.1 VISUAL PANEL

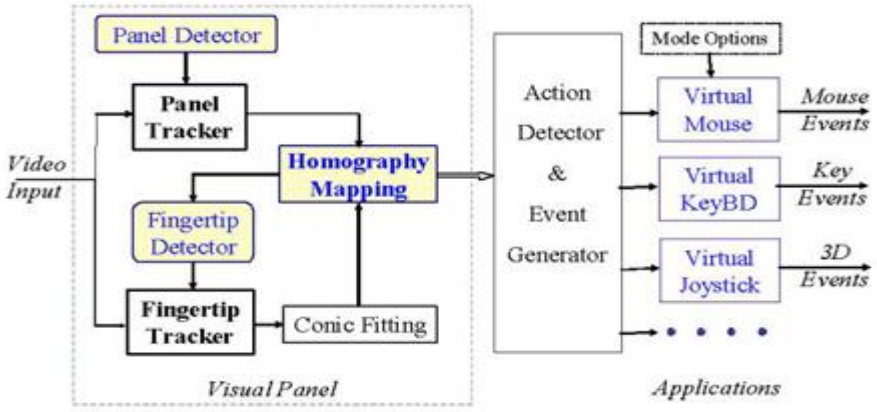

Figure 2.1. Flow chart of visual panel

To overcome the stated problems, Zhengyou et al. (2001), proposed an interface system named Visual Panel that utilizes an 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 implement in the positions of the tip-pointer, resulting in accurate and robust interaction with the

computer. The overall system consists of a panel tracker, tip-pointer tracker, holography, calculation and update, and action detector and event generator as it can simulate both mouse and keyboard. However, although the proposed system solved the issues of cable length limitations, it still requires a certain area and material to operate. Zhengyou et al., have mentioned that the system can accept any panel as long as it is quadrangle shaped, meaning any other shape besides the stated shape is not allowed. Figure 2.1. depicts flow chart of visual panel

2.2 MOUSE SIMULATION USING TWO COLOURED TAPES

Kamran Niyazi et al. (2012), mentioned that to solve the stated problem, a ubiquitous computing method is required. Thus, a colour tracking mouse simulation was proposed. The said system tracks two color tapes on the user's fingers by utilizing computer vision technology. One of the tapes will be used for controlling the movement of the cursor while the other will act as an agent to trigger the click events of the mouse.

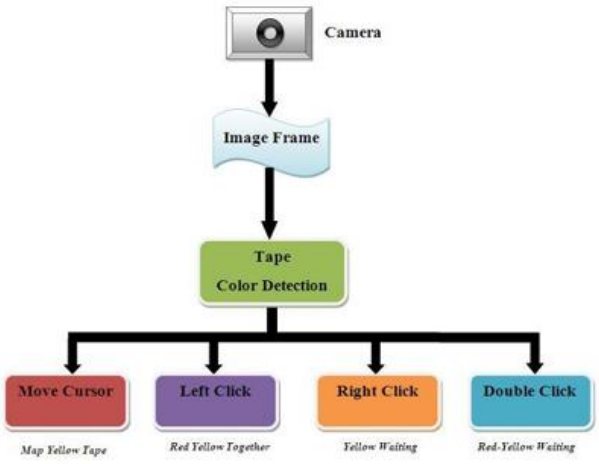

Figure 2.2. Flow chart of mouse simulation using two coloured tapes

To detect the colors, the system is first required to process the captured image by separating the hand pixels from the non-hand pixels, which can be done by a background subtraction scheme that segments the hand's movement information from the non-changing background scene. To implement this, the system requires capturing a pair of images to represent the static workplace from the camera view. When the subtraction process is complete, the system will undergo another process

that separates the RGB pixels to calculate the probability and differentiate the RGB values to determine which parts are the skin and which are not. After the process is completed, it will start detecting the defined color in the image, the image RGB pixels will be converted into an HSV color plane to eliminate the variation in shades of similar color. The resulting image will be converted to Binary Image and will undergo a filtering process to reduce the noise within the image. Even though the proposed system solved most of the stated issues, there are limited functions offered by the proposed system as it is merely able to perform common functions, such as cursor movements, left/right clicks, and double clicks. While other functions, such as the middle click and mouse scroll were ignored. Figure 2.2. depicts flow chart of mouse simulation using two coloured tapes

SYSTEM ANALYSIS

CHAPTER 3

SYSTEM ANALYSIS

This chapter will cover in detail about the existing and proposed system. At the end of this chapter, you will be clear about the existing system's drawback and the advantage of the proposed system.

3.1 EXISTING SYSTEM

The existing system consists of a mouse that can be either wireless or wired to control the cursor, now we can use hand gestures to monitor the system. The existing virtual mouse control system consists of a simple mouse operation using the colored tips for detection which are captured by the webcam, hence colored fingers act as an object, and the webcam sense color like red, green, and blue color to monitor the system, whereas could perform basic mouse operations like minimize, drag, scroll up, scroll down, left-click right-click using hand gestures without any color the finger recognition system is not flexible. There is other virtual mouse software too which uses AR/VR method but it is so costly that it also sometimes lags and it also takes most of the system power.

Disadvantages:

- Prone to degradation of the mouse roller and button causing to be faulty, switches
- Requires a flat surface to operate
- It cannot work on super smooth surface
- AR/VR virtual mouse software takes most of the systems power
- AR/VR model-based software is expensive
- Mechanical Mouse or Optical laser mouse will not last longer

It is known that there are various types of physical computer mice in this modern world the following will discuss the types and differences between the physical mouse.

3.1.1 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 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 steel made that was and 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. Figure 3.1. depicts the mechanical mouse.

Figure 3.1. Mechanical mouse

However, due to the constant friction made between the mouse ball and the rollers themselves, 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 click till it was disassembled and repaired

3.1.2 Optical laser mouse

A mouse that is commonly used these days, the motions optical mouse relies on 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 the current plagues predecessor, giving it more durability while offering better resolution and precision. Figure 3.2. depicts the optical laser mouse.

Figure 3.2. Optical laser mouse

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 trapped between the LEDs, which will cause both optical and laser mouse having 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 was disassembled and repaired. The existing system uses static hand recognition like fingertip identification, hand shape, and Number of fingers to define action explicitly, which makes a system more complex to understand and difficult to use.

3.2 PROPOSED SYSTEM

It is fair to say that the Virtual Mouse will soon to be substituting the traditional physical mouse shortly, as people are aiming towards a lifestyle where every technological device can be controlled and interacted with 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.

3.2.1 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 the 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 the user's hand position to determine the position of the pointers that the user wants it to be. For example, the user will be able to remotely control and interact with the computer system by just facing the webcam or any other image-capturing devices and moving their fingers, thus eliminating the need to manually move the physical mouse, while able to interact with the computer system from few feet away.

3.2.2 Cost effective

A quality physical mouse normally costs from the range of 30 ringgit to a hefty 400 ringgit, depending on its functionality and features. Since the Virtual Mouse requires only a webcam, a physical mouse is 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.

3.3 PROJECT SCOPE

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

The scopes are:

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

The process of the application can be started when the user's gesture was captured in real-time by the webcam, and the captured image will be processed for segmentation to identify which pixel values are equal to the values of the defined color. After the segmentation is completed, the overall image will be converted to Binary Image where the identified pixels will be as white, while the rest are black. The position of the white segment in the image will be recorded and set as the position of the mouse p pointer, thus resulting in simulating the mouse pointer without using a physical computer mouse. The software application is compatible with the Windows platform. The functionality of the software will be coded with python programming language code with the titration of an external library that does the image processing known as OpenCV.

3.4 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 a webcam by using various image processing techniques. Other than that, this project aims to develop a Virtual Mouse application that can be operated on all kinds of surfaces and environments. The following describes the overall objectives of this project:

- To design to operate with the help of a webcam. The Virtual Mouse application will be operational with the help of a webcam, as the webcam is responsible to capture the images in real-time. The application would not work if there is no webcam detected.
- 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 as the motion gesture. users are facing the webcam while doing.
- To program the camera to continue capturing 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 be undergoing a series of processes, this includes HSV conversion, Binary Image conversion, salt and pepper noise filtering, and more.
- To convert hand gesture/motion into mouse input that will be set to a particular screen position.

3.5 APPLICATIONS

This work can easily replace the traditional mouse system that has been in existence for decades with the use of this algorithm the user can control the mouse without the fuss of any other hardware device this is done using hand gestures recognition with inputs from a webcam.

SYSTEM DESIGN

CHAPTER 4

SYSTEM DESIGN

This chapter will discuss in detail about the system architecture and about the modules used in the software. In the end of this chapter, you will be clear about system architecture and various modules used.

4.1 SYSTEM ARCHITECTURE

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

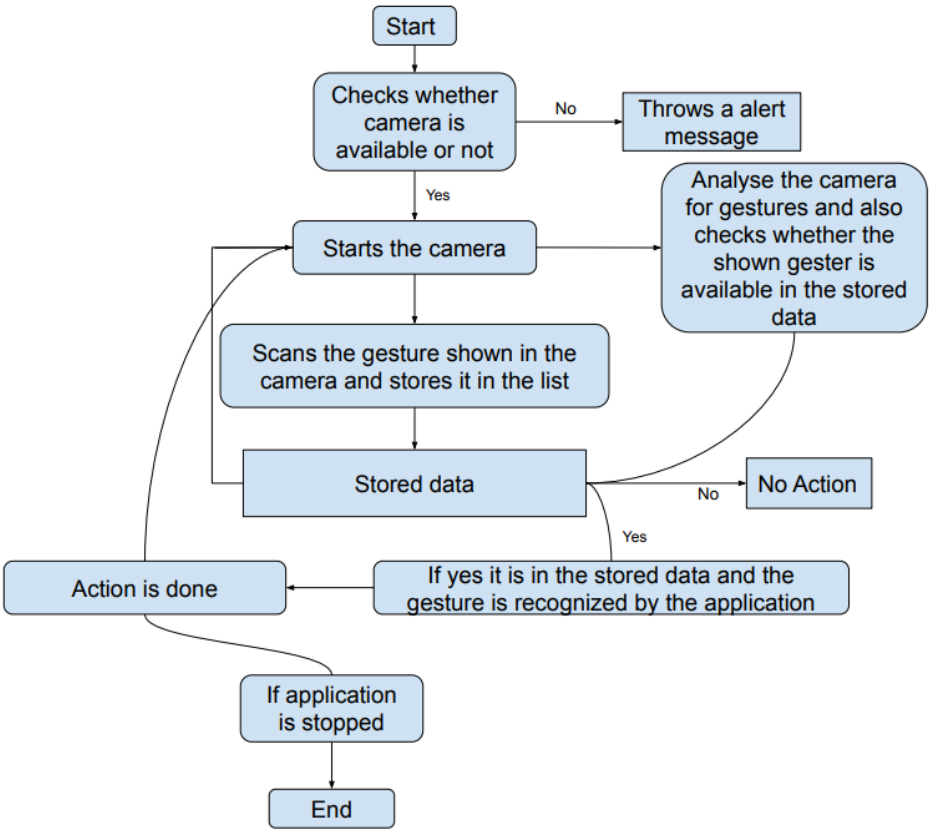

Figure 4.1 Flow chart of proposed system

4.2 MODULE DESCRIPTION

4.2.1 The camera used in the AI virtual mouse system

The proposed AI virtual mouse system is based on the frames that have been captured by the webcam in a laptop or PC. By using the Python computer vision library OpenCV, the video capture object is created and the web camera will start capturing video. The web camera captures and passes the frames to the AI virtual system. Figure 6.1. depicts hand recognized

4.2.2 Detecting the fingers

In this stage, we are detecting which finger is up using the tip Id of the respective finger that we found using the MediaPipe and the respective co-ordinates of the fingers that are up, and according to that, the particular mouse function is performed. Figure 6.2. depicts finger tip

4.2.3 For the mouse cursor moving around the window

both the index finger with tip Id = 1 and the middle finger with tip Id = 2 is up, the mouse cursor is made to move around the window of the computer using the AutoPy package of Python. Figure 6.3. depicts gesture for moving the cursor around the window

4.2.4 For The mouse to perform right button click

If the index finger is shown in the camera then it will recognize it and performs right click operation. Figure 6.4. depicts gesture for right click

4.2.5 For the mouse to perform left button click

If the thumb finger is shown in the camera then it will recognize it and performs left click operation. Figure 6.5. depicts gesture for left click

4.2.6 For the mouse to reduce or increase the sound

If the below gesture is shown and the hand is moving vertically then it will reduce the sound of the operating system. Figure 6.6. depicts gesture for increasing or reducing the sound

4.2.7 For the mouse to reduce or increase the brightness

If the below gesture is shown and the hand is moved horizontally then it will increase or reduce the sound of the operating system. Figure 6.7. depicts gesture for increasing or reducing the brightness

4.2.8 For the mouse to drag and drop

If the palm is closed and shown it will drag or drop the document from one folder to other. Figure 6.8. depicts gesture to drag and drop

SYSTEM SPECIFICATION

CHAPTER 5

SYSTEM SPECIFICATION

This chapter will discuss in detail about system specification needed to run this software. At the end of this chapter, you will have a detail knowledge about the system needs to run this software.

5.1 HARDWARE REQUIRMET

The following describes the hardware needed in order to execute and develop the Virtual Mouse application:

- Webcam

  Webcam is utilized for image processing, the webcam will continuously taking image in order for the program to process the image and find pixel position.

- Computer Desktop or Laptop

  The computer desktop or a laptop will be utilized to run the visual software to display what the webcam had captured. A notebook which is a small, lightweight, and inexpensive laptop computer is proposed to increase mobility. System will be using

Processor	:	Core2Duo
Main Memory	:	4GB RAM
Hard Disk	:	320GB
Display	:	14" Monitor

5.2 SOFTWARE REQUIRMENT

5.2.1 Python

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). Python has a simple syntax similar to the English language. Python has a syntax that allows developers to write programs with fewer lines than some other

programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated procedurally, in an object-oriented way, or in a functional way. Python can be used on a server to create web applications. Python can be used alongside software to create workflows. Python can connect to database systems. It can also read and modify files. Python can be used to handle big data and perform complex mathematics. Python can be used for rapid prototyping or production-ready software development.

5.2.2 MediaPipe

MediaPipe is a framework that is used for applications in a machine learning pipeline, and it is an open-source framework of Google. The MediaPipe framework is useful for cross-platform development since the framework is built using time series data. The MediaPipe framework is multimodal, where this framework can be applied to various audio and videos. The MediaPipe framework is used by the developer for building and analyzing the systems through graphs, and it has also been used for developing the systems for application purposes. The steps involved in the system that uses MediaPipe are carried out in the pipeline configuration. The pipeline created can run on various platforms allowing scalability in mobile and desktops. The MediaPipe framework is based on three fundamental parts; they are performance evaluation, a framework for retrieving sensor data, and a collection of components which are called calculators, and they are reusable. A pipeline is a graph that consists of components called calculators, where each calculator is connected by streams through which the packets of data flow through.

Developers can replace or define custom calculators anywhere in the graph creating their applications. 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. The hand detection module 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 landmarks consists of locating 20 joint or knuckle coordinates in the hand region.

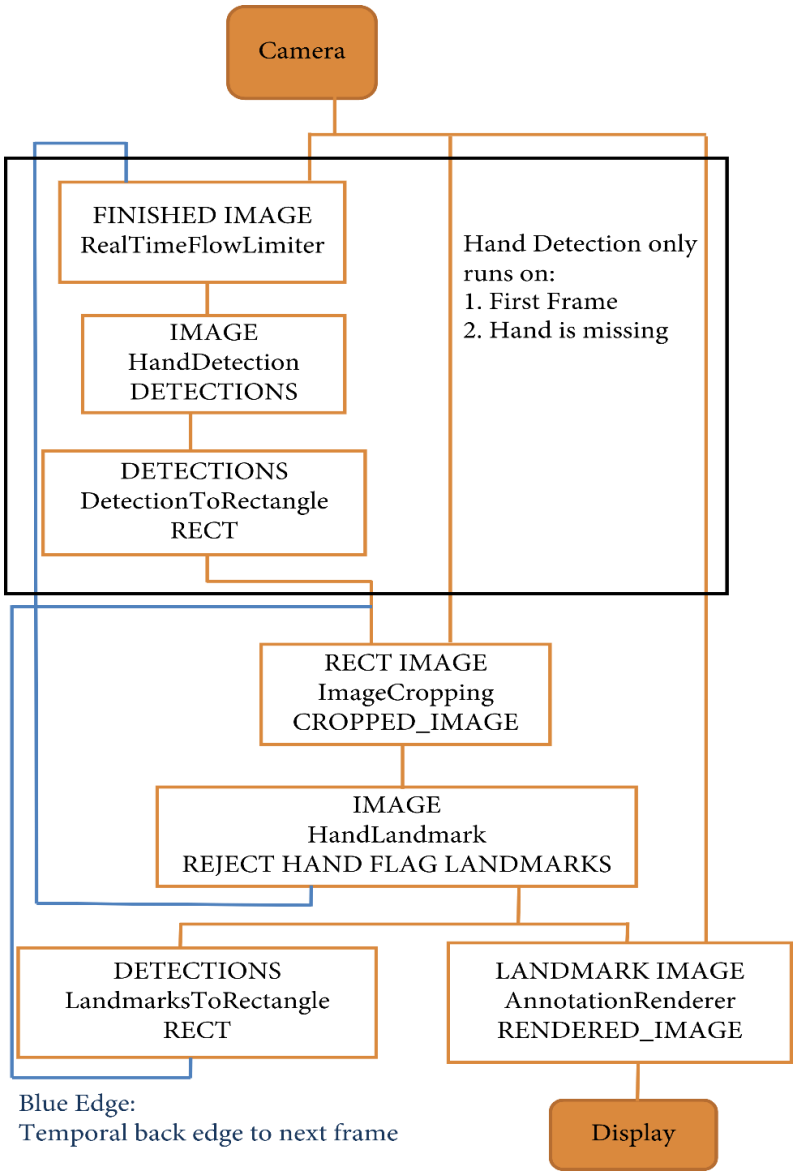

Figure 5.1. Flowchart of mediapipe algorithm

5.2.3 OpenCV

OpenCV is a computer vision library that 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. It is a huge open-source

library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify an image pattern and their various features we use vector space and perform mathematical operations on these features.

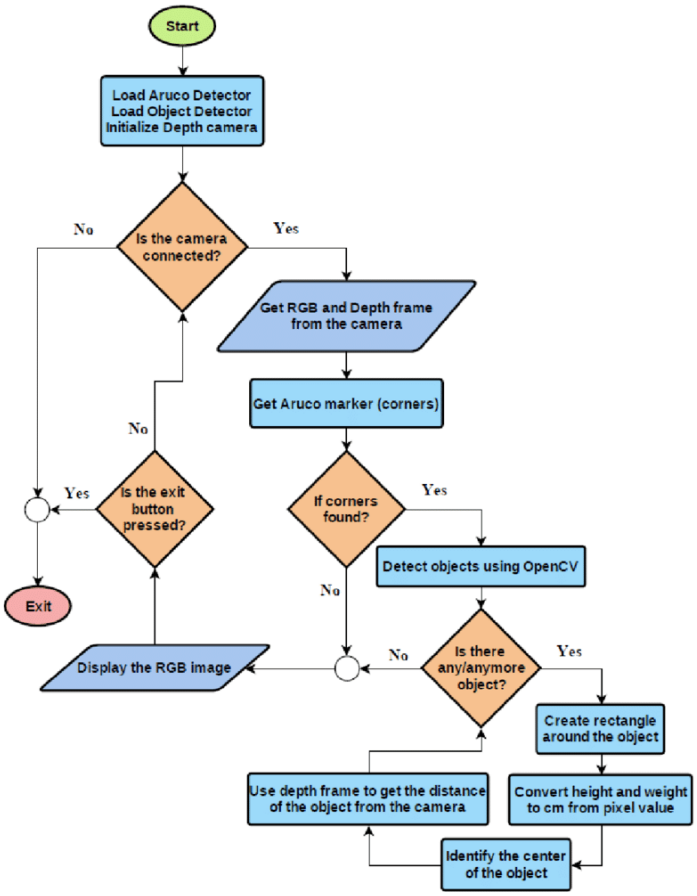

Figure 5.2. OpenCV algorithm flowchart

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python, and Java interfaces and supports Windows, Linux, Mac OS, iOS, and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing. lots of pieces of information that are present in the original image can be

obtained. Like if there is an image captured and there are two people and in that image, one person is wearing glasses, or watches, and more so with the help of OpenCV we can get all these types of information from the original image. It's the basic introduction to OpenCV we can continue the Applications and all the things in our upcoming articles.

5.2.4 AutoGUI

PyAutoGUI lets your Python scripts control the mouse and keyboard to automate interactions with other applications. The API is designed to be simple. PyAutoGUI works on Windows, macOS, and Linux, and runs on Python 2 and 3. The features of PyAutoGui are

- Moving the mouse and clicking in the windows of other applications.
- Sending keystrokes to applications
- Take screenshots, and given an image
- Locate an application's window, and move, resize, maximize, minimize.
- Display alert and message boxes.

5.2.5 NumPy

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents due to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy. Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python

packages are available; SciPy is a library that adds more MATLAB be like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality.

Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing, or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as a universal data structure in OpenCV for images, extracted feature points, filter kernels, and many more vastly simplifies the programming workflow and debugging.

SYSTEM IMPLEMENTATION

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SOURCE CODE

6.1.1 Handtracking

```
import cv2
import mediapipe as mp
import time
import operations
import autopy

cap = cv2.VideoCapture(0)

mpHands = mp.solutions.hands
hands = mpHands.Hands()
mpDraw = mp.solutions.drawing_utils

pTime = 0
cTime = 0

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)
    print(results.multi_hand_landmarks)

    if results.multi_hand_landmarks:
        for handLms in results.multi_hand_landmarks:
            for id, lm in enumerate(handLms.landmark):
                # print(id,lm)
                h, w, c = img.shape
                cx, cy = int(lm.x*w), int(lm.y*h)
                print(id, cx,cy)
                cv2.circle(img, (cx,cy), 15, (255,0,255), cv2.FILLED)

            mpDraw.draw_landmarks(img, handLms,
mpHands.HAND_CONNECTIONS)
```

```
    cTime = time.time()
    fps = 1/(cTime-pTime)
    pTime = cTime

    cv2.putText(img,str(int(fps)),(10,70), cv2.FONT_HERSHEY_PLAIN, 3, (255, 0,
255), 3)

    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

6.1.2 Operations

```
import cv2
import numpy as np
import trackmodule as tm
import time
import autopy
import pyautogui
import time
from inputimeout import inputimeout
import threading

Movment = []
Right_click = []
left_click = []
double_click = []
scroll = []


width, height = 640, 480

cap = cv2.VideoCapture(0)
cap.set(3, width)
cap.set(4, height)
detector = tm.handDetector(maxHands=1)
wscreen,hscreen = autopy.screen.size()


# def gesture():
#     threading.Timer(10.0, gesture).start()
#     return detector.fingersUp()

print("please give a gesture to do mouse movment")
o = str(input("Enter Y to scan your hand : "))
if o == 'Y' or o == 'y':
    while True:
```

```
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)

        if len(lmList) != 0:
            x1, y1 = lmList[8][1:]
            x2, y2 = lmList[12][1:]

            if len(Movment) != 100:
                Movment.append(detector.fingersUp())
                print(Movment)
                if len(Movment) == 100:
                    Movment = Movment[-1]
                    break

        cv2.imshow("Images", img)
        cv2.waitKey(1)
    print(Movment)

else:
    print("please scan your gesture to enable mouse movment")

print("please give a gesture to do right click operation")
o = str(input("Enter Y to scan your hand : "))
if o == 'Y' or o == 'y':
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)

        if len(lmList) != 0:
            x1, y1 = lmList[8][1:]
            x2, y2 = lmList[12][1:]

            if len(Right_click) != 100:
                Right_click.append(detector.fingersUp())
                print(Right_click)
                if len(Right_click) == 100:
                    Right_click = Right_click[-1]
                    break

        cv2.imshow("Images", img)
        cv2.waitKey(1)
    print(Right_click)
```

```
else:
    print("please scan your gesture to enable right click operation")

print("please give a gesture to do left click operation")
r = str(input("Enter Y to scan your hand : "))
if r == 'Y' or r == 'y':
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)

        if len(lmList) != 0:
            x1, y1 = lmList[8][1:]
            x2, y2 = lmList[12][1:]

            if len(left_click) != 100:
                left_click.append(detector.fingersUp())
                print(left_click)
                if len(left_click) == 100:
                    left_click = left_click[-1]
                    break

        cv2.imshow("Images", img)
        cv2.waitKey(1)
    print(left_click)

else:
    print("please scan your gesture to enable left click operation")

print("please give a gesture to do double click operation")
i = str(input("Enter Y to scan your hand : "))
if i == 'Y' or i == 'y':
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)

        if len(lmList) != 0:
            x1, y1 = lmList[8][1:]
            x2, y2 = lmList[12][1:]

            if len(double_click) != 100:
                double_click.append(detector.fingersUp())
                print(double_click)
                if len(double_click) == 100:
```

```
                double_click = double_click[-1]
                break

    cv2.imshow("Images", img)
    cv2.waitKey(1)
    print(double_click)

else:
    print("please scan your gesture to enable double click operation")


c = []
c.append(Movment)
c.append(Right_click)
c.append(left_click)
c.append(double_click)

print("the input gestures are : ")
print(c)

smoothening = 7
px,py = 0,0
cx,cy = 0,0

while True:
    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bbox = detector.findPosition(img)

    if len(lmList) != 0:
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]

        # mouse movment
        b = c[0]
        if detector.fingersUp() == b:
            o1 = np.interp(x1, (0,width), (0,wscreen))
            o2 = np.interp(y1, (0,height), (0,hscreen))
            cx = px + (o1 - px) / smoothening
            cy = py + (o2 - py) / smoothening
            autopy.mouse.move(wscreen-cx,cy)
            px, py = cx, cy
        # right click
        if detector.fingersUp() == c[1]:
            o1 = np.interp(x1, (0,width), (0,wscreen))
            o2 = np.interp(y1, (0,height), (0,hscreen))
```

```
            cx = px + (o1 - px) / smoothening
            cy = py + (o2 - py) / smoothening
            pyautogui.click(wscreen-cx,cy,button='right')
        # left click
        if detector.fingersUp() == c[2]:
            o1 = np.interp(x1, (0,width), (0,wscreen))
            o2 = np.interp(y1, (0,height), (0,hscreen))
            cx = px + (o1 - px) / smoothening
            cy = py + (o2 - py) / smoothening
            pyautogui.click(cx,cy)
        # double click
        if detector.fingersUp() == c[3]:
            o1 = np.interp(x1, (0,width), (0,wscreen))
            o2 = np.interp(y1, (0,height), (0,hscreen))
            cx = px + (o1 - px) / smoothening
            cy = py + (o2 - py) / smoothening
            pyautogui.click(cx,cy,clicks=2)

    cv2.imshow("Image", img)
    cv2.waitKey(1)
```

6.1.3 Trackmodule

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


class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

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

```
        # print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img

    def findPosition(self, img, handNo=0, draw=True):
        xList = []
        yList = []
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                # print(id, lm)
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)
                # print(id, cx, cy)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
                              (0, 255, 0), 2)

        return self.lmList, bbox

    def fingersUp(self):
        fingers = []
        # Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)
```

```
        # Fingers
        for id in range(1, 5):

            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)

        # totalFingers = fingers.count(1)

        return fingers

    def findDistance(self, p1, p2, img, draw=True,r=15, t=3):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

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

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


def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList)

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)
```

```
        cv2.imshow("Image", img)
        cv2.waitKey(1)


if __name__ == "__main__":
    main()
```

6.2 OUTPUT SCREENSHOTS

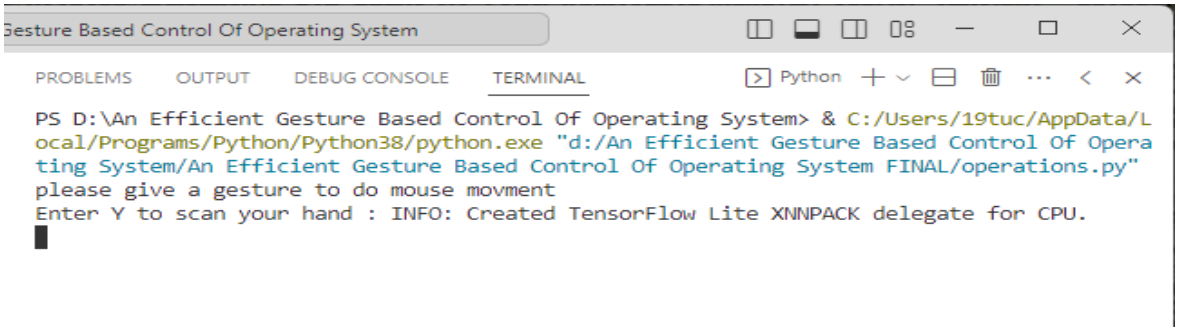

Figure 6.1: gesture input console image

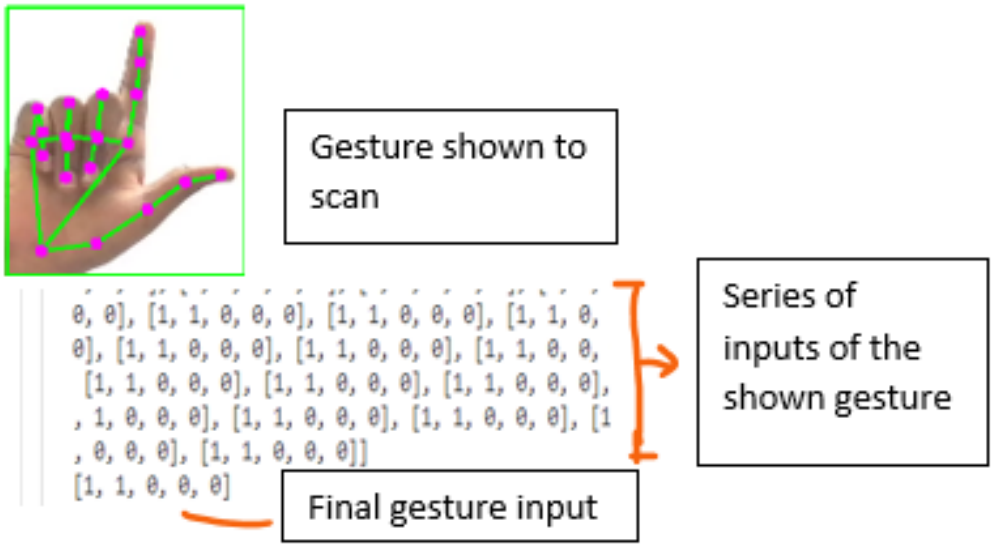

Figure 6.2: Gesture scanned and determined the input

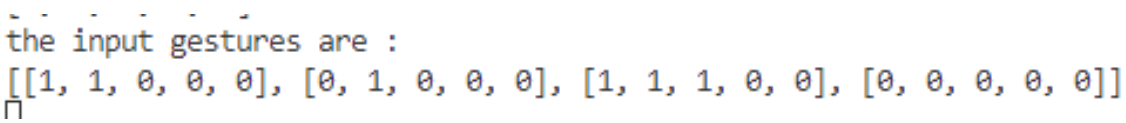

Figure 6.3: All gestures grouped into 2d list

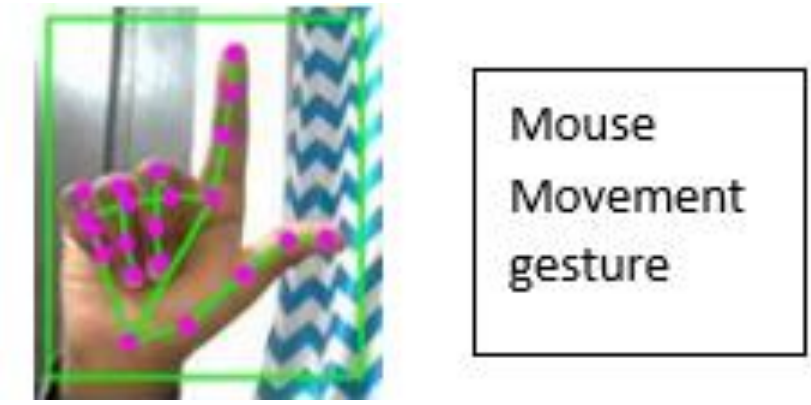

Figure 6.4: Gesture given by a user to the application

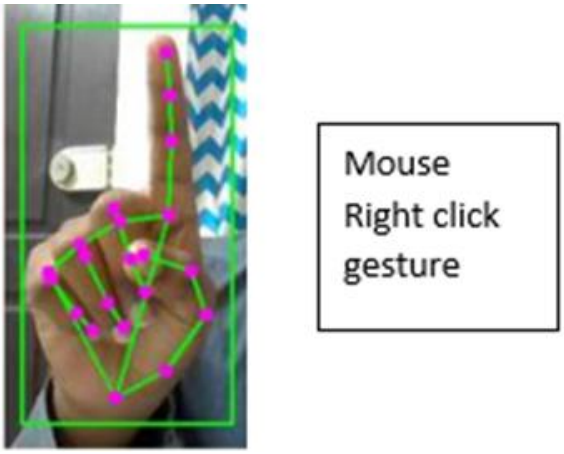

Figure 6.5: Mouse right click event gesture

Figure 6.6: Mouse left click event gesture

Figure 6.7: double click event gesture

```
please give a gesture to do right click operation
Enter Y to scan your hand :
```

Figure 6.8: message that mentions about for which operation we are going to set a gesture

CONCLUSION

CHAPTER 7

CONCLUSION

The main objective of the virtual mouse system is to control the mouse 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 that detects the hand gestures and hand tips and processes these frames to perform the particular mouse functions. From the results of the model, we can conclude that the proposed virtual mouse system has performed very well and has 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 virtual mouse can be used for real-world applications, the proposed mouse system can be used virtually using hand gestures without using the traditional physical mouse.

REFERENCES

REFERENCES

[1] Tsai, T. H., Huang, C. C., & Zhang, K. L. (2015, June). Embedded virtual mouse system by using hand gesture recognition. In *2015 IEEE International Conference on Consumer Electronics-Taiwan* (pp. 352353). IEEE.

[2] Farooq, J., & Ali, M. B. (2014, April). Real time hand gesture recognition for computer interaction. In *2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE)* (pp. 73-77). IEEE.

[3] Rautaray, S. S. (2012). Real time hand gesture recognition system for dynamic applications. *International Journal of ubicomp (IJU), 3*(1).

[4] Shibly, K. H., Dey, S. K., Islam, M. A., & Showrav, S. I. (2019, May). Design and development of hand gesture based virtual mouse. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)* (pp. 1-5). IEEE.

[5] Rautaray, S. S., & Agrawal, A. (2011, December). Interaction with virtual game through hand gesture recognition. In *2011 International Conference on Multimedia, Signal Processing and Communication Technologies* (pp. 244247). IEEE.

[6] Aksaç, A., Öztürk, O., & Özyer, T. (2011, December). Real-time multiobjective hand posture/gesture recognition by using distance classifiers and finite state machine for virtual mouse operations. In 2011 7th International Conference on Electrical and Electronics Engineering (ELECO) (pp. II-457). IEEE.

[7] Tsai, T. H., Huang, C. C., & Zhang, K. L. (2020). Design of hand gesture recognition system for human-computer interaction. *Multimedia tools and applications*, *79*(9), 5989-6007.

[8] Hu, C., Liang, L., Ma, S., & Lu, H. (2000, October). Virtual Mouse—inputting device by hand gesture tracking and recognition. In *International Conference on Multimodal Interfaces* (pp. 88-95). Springer, Berlin, Heidelberg.

[9] Hu, C., Liang, L., Ma, S., & Lu, H. (2000, October). Virtual Mouse— inputting device by hand gesture tracking and recognition. In *International Conference on Multimodal Interfaces* (pp. 88-95).

Springer, Berlin, Heidelberg.