CHAPTER 1

INTRODUCTION

The most efficient and expressive way of human communication is through hand gestures and speech, which is universally accepted for communication. It is expressive enough for a dumb and deaf people to understand it. In this work, a real-world gesture system is proposed. Experimental setup of the system uses fixed position cost-effective web cam for high definition recording feature mounted on the top of the monitor of a computer or a fixed laptop camera. In addition to this, it uses a microphone to capture sound which is later processed to perform various mouse functions. Recognition and the interpretation of sign language or speech is one of the major issues for the communication with dumb and deaf people.

Python computer programming language has been used in the given project for the code, whereas OpenCV is used for computer vision to capture gestures. For hand tracking, the model in the proposed Virtual mouse system uses the MediaPipe package. The Python package Speech Recognition is used for voice instructions.

With the development technologies in the areas of augmented reality and devices that we use in our daily life, these devices are becoming compact in the form of Bluetooth or wireless technologies. This paper proposes an AI virtual mouse system that makes use of the hand gestures and hand tip detection for performing mouse functions in the computer using computer vision. The main objective of the proposed system is to perform computer mouse cursor functions and scroll function using a web camera or a built-in camera in the computer instead of using a traditional mouse device. Hand gesture and hand tip detection by using computer vision is used as a HCI [1] with the computer. With the use of the AI virtual mouse system, we can track the fingertip of the hand gesture by using a built-in camera or web camera and perform the mouse cursor operations and scrolling function and also move the cursor with it.

CHAPTER 2

LITERATURE SURVEY

2.1 What is Human-Computer Interaction (HCI)?

The study of how humans (users) interact with computers is known as human-computer interaction (HCI). It is a multidisciplinary field that deals with the design of computer technology. HCI began with computers and has now grown to embrace practically all aspects of information technology design.

The Meteoric Rise of HCI

When personal computers first became popular in the 1980s, HCI emerged at thesame time as machines like the IBM PC 5150, Commodore 64, and Apple Macintosh began to be utilised in homes and offices. For the first time, sophisticated electronic systems such as games units, word processors and accounting aids were available to general consumers for use. As a result, as computers grew in size to the point where they were room-sized, expensive tools created exclusively for professionals in specialised situations, the necessity to research human-computer interaction that was also efficient and simple for less experienced users grew in importance. Design, computer science, psychology, cognitive science, and human-factors engineering are just a few of the fields that have been incorporated into HCI.

Fig:2.1 Human computer interaction

Throughout the research on human-computer interaction, several variants of these algorithms have been developed in various fields including Engineering, Design,

social psychology, computer science, cognitive science, information security, sociology, and speech-language pathology are some of the fields covered.

The current research aims to create algorithms that lessen human reliance on hardware and strive for a more natural method of interacting with computers through hand gestures and speech.

The OpenCV library is utilised for computer vision, while the MediaPipe framework is used to recognise and monitor hand motions. The system also employs machine learning techniques to track and recognise hand gestures and tips.

2.2 MediaPipe

MediaPipe is a Google open-source framework that is used to apply in a machine learning pipeline. Because the MediaPipe framework is based on time series data, it is suitable for cross-platform development. The MediaPipe is a multimodal architecture that can be used with a variety of audio and video formats. The MediaPipe framework is used by developers to create and analyse systems using graphs, as well as to create systems for application development. The steps in a MediaPipe-enabled environment are carried out in the pipeline setup. The pipeline is scalable and runs ona range of platforms, including PCs, laptops, and mobile devices.

Performance evaluation, a framework for accessing sensor data, and a reusable collection of components known as calculators are the three primary components of the MediaPipe system.

In order to recognise and detect a hand or palm in real time, a single-shot detector model is used. The single-shot detector is used by MediaPipe. It is first trained for a palm detection model of hands in the hand detection module since palms of hands are easier to train and map. Furthermore, the non-maximum suppression is far more effective on small objects like hands and fists. The location of 21 joint or knuckle co-ordinates in the hand make up a model of hand map or landmark in hand gesture control model using mediaPipe.

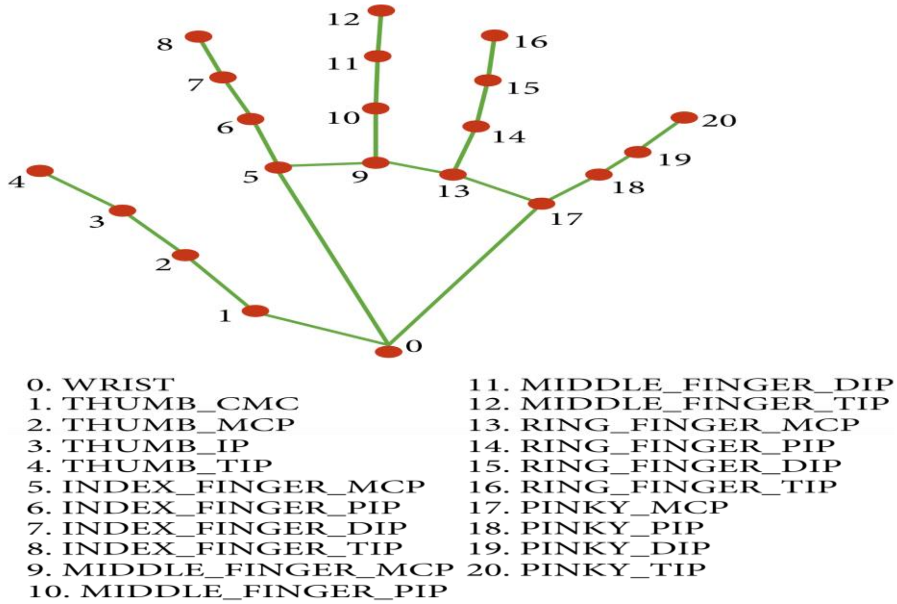


Fig:2.2 Hand palm coordinate

2.3 OpenCV

OpenCV is a real-time computer vision library that focuses on computer vision.Intel was the first to develop it.Under the open-source BSD licence, the library is cross platform and free to use.

OpenCV is written in C++ and has a C++-based user interface. Python, Java, and MATLAB/OCTAVE all have bindings.Open-source library of computer vision, image analysis, and machine learning. To do this, it has an infinity of algorithms that allow, just by writing a few lines of code, identifying faces, recognizing objects, classifying them, detecting hand.

2.4 Past Researches

We have come a long way in the field of human computer interaction. Gesture based mouse control was carried out by wearing gloves initially. Later, colour tips were alsoused for gesture recognition. Although such systems were not very accurate. The recognition accuracy is less due to use of gloves. Some users may not feel comfortable wearing gloves, and in some situations, recognition is not as accurate as it may be due to colour tip detection failure. Computer-based gesture detection systems have recently received some attention.

In 1990, Quam introduced a hardware-based approach that required the user to wear a DataGlove. Despite the fact that Quam's proposed method generates more precise results, certain of the gesture controls are difficult to execute with the system.

Zhengyou et al. proposed the Visual Panel interface system (2001). A quadrangle-shaped plane is used in this system, allowing the user to perform mouse operations with any tip-pointed interface instrument. Though the system can be operated contact free yet it does not solve the problem of surface area requirementand material handling.

Color tracking mouse stimulation was proposed by Kamran Niyazi et al. (2012). Using computer vision technology, the system tracks two colour tapes on the user's fingertips. One of the tapes will be used to control the cursor's movement, while the other will act as a trigger for the mouse's click events. Despite the fact that the proposed system handled the bulk of the issues, it only has a limited range ofcapabilities, as it can only perform fundamental actions such as cursor movements, left/right clicks, and double clicks.

To replicate click events, the system requires three fingers with three colour pointers, according to Kazim Sekeroglu (2010). The suggested system can detect pointers using colour information, track their motion, change the cursor according to the position of the pointer, and simulate single and double left or right mouse click events.

Chu-Feng Lien (2015) proposed a way for controlling the mouse cursor and click events using only one's fingertip. To interact with the system, the suggested system does not require hand motions or colour tracking; instead, it uses a feature called Motion History Images (MHI). Because the frame rates can't keep up with quickly moving objects, the proposed system can't detect them. Furthermore, because mouse click events occur when the finger is held in particular positions, this may cause the user to move their fingers constantly to avoid false alarms, which can be inconvenient.

CHAPTER 3

PROBLEM STATEMENT

3.1 Existing System

USB and Bluetooth are two different technologies used to connect a mouse to a computer.

A USB mouse connects to the computer via a USB port, which is a standard interface found on most modern computers. USB mice are typically plug-and-play, meaning that the computer will recognize and install the necessary drivers automatically when the mouse is plugged in.

A Bluetooth mouse, on the other hand, connects to the computer wirelessly using Bluetooth technology. Bluetooth mice require that the computer have Bluetooth capabilities, either built-in or via an external Bluetooth adapter. Once the mouse is paired with the computer, it can be used without the need for any cables or physical connections.

Both USB and Bluetooth mice have their advantages and disadvantages. USB mice are generally more reliable and have faster response times, since they are directly connected to the computer. Bluetooth mice, on the other hand, are more convenient since they don't require any cables, and can be used from a distance away from the computer.

Ultimately, the choice between a USB and Bluetooth mouse depends on personal preference and the specific needs of the user. If you value reliability and speed, a USB mouse may be a better choice. If you prioritize convenience and portability, a Bluetooth mouse may be the way to go.

3.2 Existing System Drawbacks

Both USB and Bluetooth mice have their drawbacks.

Some drawbacks of USB mice include:

1. Limited mobility: Since USB mice are connected to the computer via a cable, the range of movement is limited.

2. Cable clutter: The cable of a USB mouse can cause clutter around your workspace, which can be a problem if you have limited desk space.

3. Limited compatibility: Some older computers may not have USB ports, which means that you may not be able to use a USB mouse with them.

4. USB port congestion: If you have multiple USB devices, you may run out of USB ports on your computer, which can be a problem if you need to connect other devices.

Some drawbacks of Bluetooth mice include:

1. Connectivity issues: Bluetooth mice can sometimes experience connectivity issues, particularly if there is interference from other wireless devices.

2. Battery life: Since Bluetooth mice rely on batteries for power, you may need to replace or recharge the batteries frequently.

3. Lag: Bluetooth mice can sometimes experience lag, particularly if there is a lot of wireless interference or if the batteries are low.

4. Compatibility issues: Some older computers may not have Bluetooth capabilities, which means that you may not be able to use a Bluetooth mouse with them.

3.2 Proposed System

we people are targeting towards a lifestyle where everything can be controlled remotely without the involvement of any physical device such as the mouse, keyboards, etc. Not only is using a virtual mouse convenient, but it is also cost-effective.

The project's main goal is to create a hands-free virtual Mouse system that focuses on a few key applications in development. This project aims to eliminate the need for a physical mouse while allowing users to interact with the computer system via webcam and speech using various image and audio processing techniques. Thisproject seeks to create a Virtual Mouse programme that can be used in a variety of contexts and on a variety of surfaces.

3.2 Proposed System Objectives

- Design for mouse operation with the aid of a webcam. The Virtual Mouse technology works with the help of a webcam, which takes real-time photos and photographs. A webcam is required for the application to function.

- The cursor is assigned to a certain screen position when the hand gesture/motion is converted into a mouse operation. The Virtual Mouse application is set up to identify the position of the mouse pointers by detecting the position of the fingertips and knuckles on a defined hand colour and texture.

- Develop a multi user independent speech recognition system that captures voice in real-time with the help of a microphone and is able to retrieve folders, sub-folders, documents, copy, paste, left click, right click and double click by taking voice command and checking its validity.

- Create a voice-activated mouse system that works in tandem with the gesture-activated system.

CHAPTER 4

HARDWARE AND SOFTWARE REQUIREMENT

Hardware Requirement

The Virtual Mouse application requires the following hardware for development andexecution:

- Laptop or Computer Desktop

  To display what the webcam has taken, the virtual software will be started onthe laptop or computer desktop.

  The system will make use of (minimum requirements) Core2Duo processor (2nd generation)

  2 GB RAM (Main Memory)320 GB hard drive

  14-inch LCD monitor

- Webcam

  The image is acquired with a camera that will continue to take photos endlessly so that the application may process the image and calculate pixelposition.

  Resolution: 1.3 megapixels is the minimum required.

- Microphone

  Voice commands are recorded via the microphone. Until it is switched off or placed to sleep, it will continue to listen to all commands.

  Microphone should be capable of recognizing frequency range of 40Hz -16KHz

Software Requirement

The following software is required for the development and execution of the Virtual Mouse application:

- **Python Language**

  The Virtual Mouse application is coded in Python with the help of MicrosoftVisual Studio Code, an integrated development environment (IDE) for programming computer applications.
  Basic arithmetic, bit manipulation, indirection, comparisons, logical operations, and more are all available in the Python library.

- **Open CV Library**

  This software is also created with the help of OpenCV.
  OpenCV (Open Source Computer Vision) is a real-time computer vision library. OpenCV is capable of reading picture pixel values as well as real-time eye tracking and blink detection.

  Software will be using:
  OS : Window 10 Ultimate 64-bitLanguage : Python
  Tool Used : OpenCV and MediaPi

CHAPTER 5

SYSTEM DESIGN AND ARCHITECTURE

5.1 Use Case Diagram

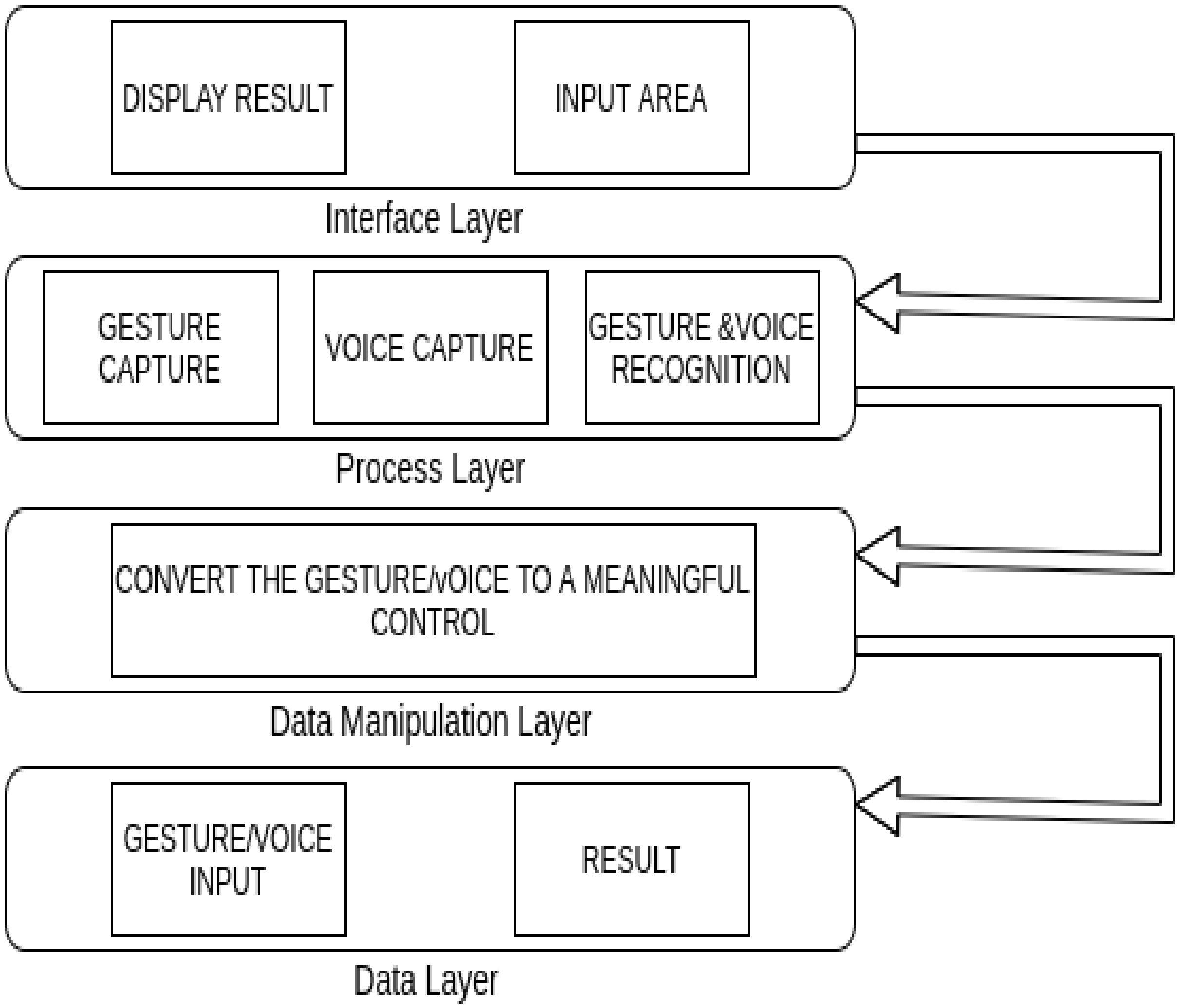


Fig:5.1 Use Case Diagram

5.2 Gesture Control Data Flow Diagram

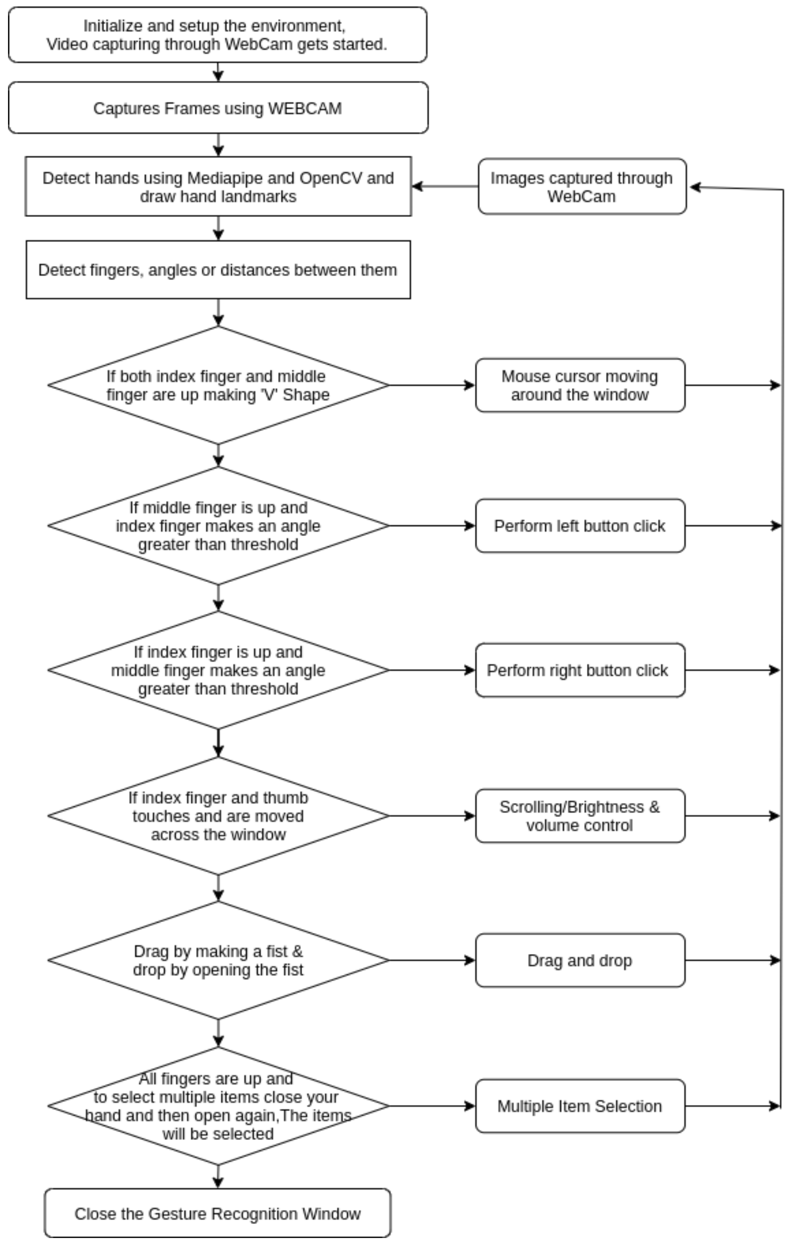


Fig:5.2 Gesture Control Data Flow Diagram

5.3 Voice Control Data Flow Diagram

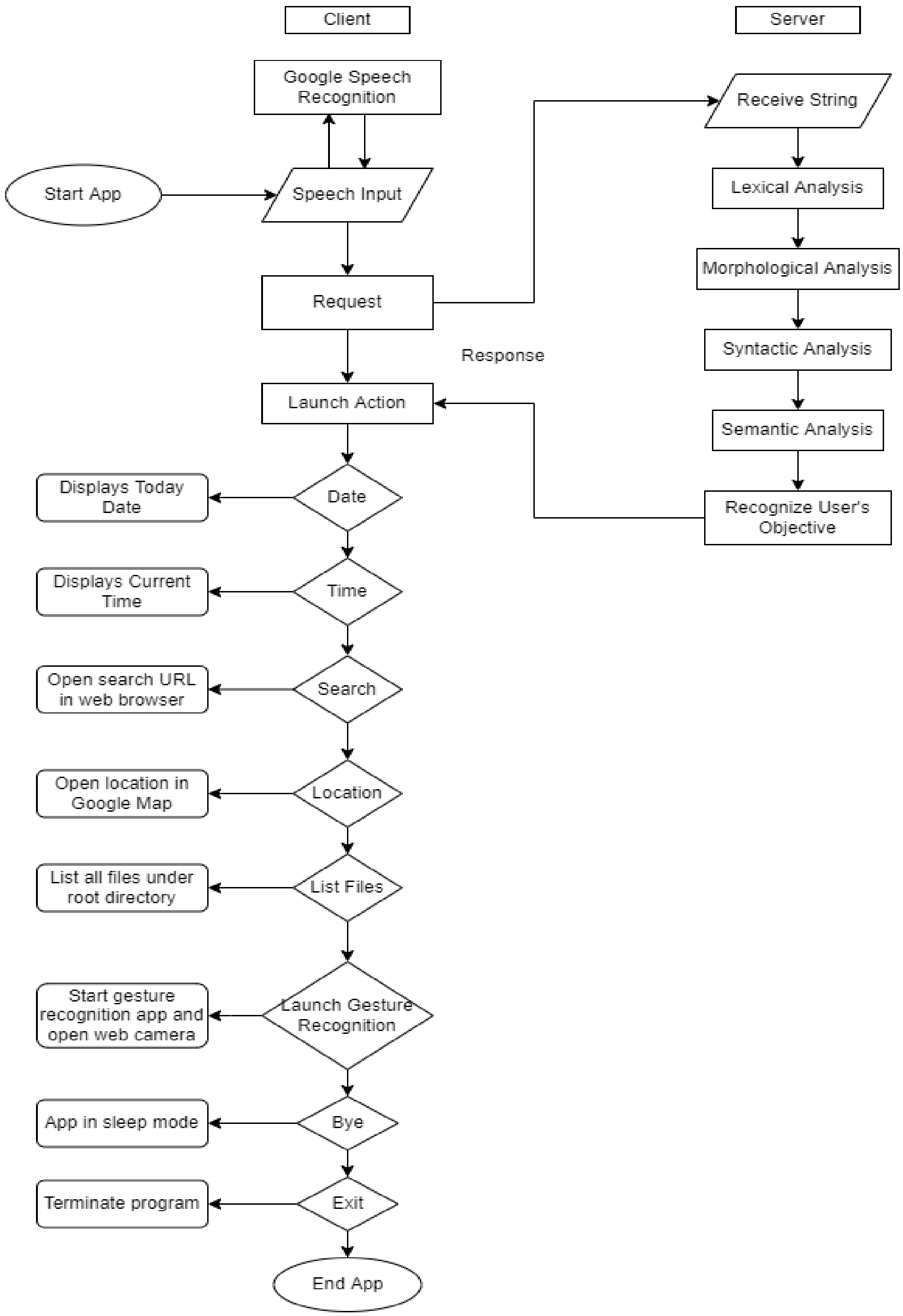


Fig:5.3 Voice Control Data Flow Diagram

CHAPTER 6

METHODOLOGY

6.1 Working of OpenCV

Computer works only with numbers. Everything we save in a computer like video, images, documents, etc is saved in a computer in the form of numbers.

In image processing pixels are converted into numbers.A pixel is the smallest unit of a digital image. The numbers can be used to calculate a number's intensity at any given pixel.OpenCV can work in a grey scale or BGR(Blue, Green, Red) format.

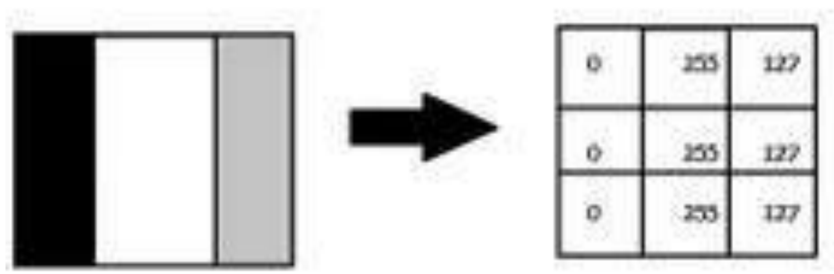


Fig:6.1 OpenCV Grey Scale

Images can be classified using:

1. Gray Scale Images

Image processing using this method involves converting the image into black and white format, where black is 0 and white is 255.

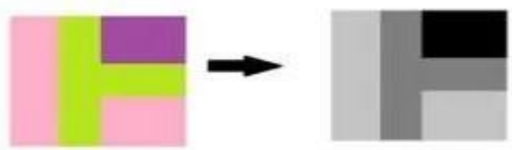


Fig:6.2 OpenCV Grey Scale

1. BGR format

The photos feature three colours: blue, green, and red. The computer extracts that value from each pixel and puts the results in an array to be interpreted. Images are represented as three channels blue, green and red.

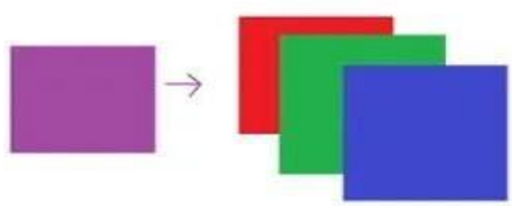


Fig:6.3 RGB Format

To identify the hand, the cumulative probability of B G R is employed

6.2 ML Pipeline(Mediapipe) for Hand Tracking and Gesture Recognition

Mediapipe is a Machine Learning system built on the collaboration of pipeline models.

What is ML Pipeline?

A pipeline joins several stages together so that the output of one is used as the input for the next. Pipeline makes it simple to train and test using the same preprocessing.

The hand tracking method makes use of a machine learning pipeline that consists oftwo models that work together:

• A palm detector that uses an aligned hand bounding box to locate palms on a whole input image.
• A hand landmark model that uses the palm detector's clipped hand bounding box to produce high-fidelity results. landmarks in 2.5D

The following is a summary of the pipeline:

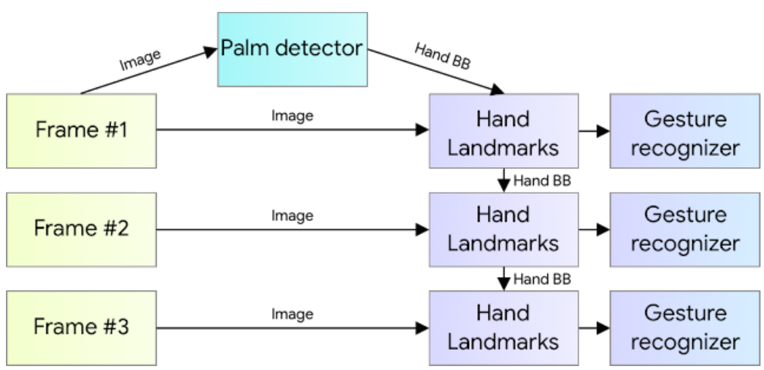


Fig:6.4 Palm Detection Model

Palm Detector Model

Hand detection is a tedious process as it requires identifying hands of various sizes, shapes, with deformities, etc. It is more complex than face detection as the contrast infeatures is far less than that in face.

We use palm detection model first as detecting palm or a fist is much easier than detecting a full hand with articulated fingers. Also palms are smaller therefore non suppression algorithm works better for it.

Hand Landmark Model

After detecting the palm using a palm detection model next a hand landmark model isused to detect 21 landmark points in 2.5 dimension. The Z depth is analysed using a Image Depth Map. The model recognises both partially and fully acclusioned hands perfectly.

The model has three outputs (see Figure 3):

1. 21 hand landmarks consisting of x, y, and relative depth.
2. A hand flag indicates the existence of a hand in the input image.
3. A binary classification of handedness, e.g. left or right hand.

The topology is the same as for the 21 landmarks. To avoid performing hand detection over and over for the entire frame, the probability of hand presence in a bounded crop is determined. The detector is triggered to reset tracking if the score falls below a threshold. We constructed a binary classification head to predict whetherthe input hand is left or right. Only the first frame or when the hand prediction shows that the hand is lost is the detector used.

For our project, the fingers are given Ids from 0 to 4.

CHAPTER 7

IMPLEMENTATION

7.1 Gesture Control Code Snippet

```
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
class Gest(IntEnum):
class HLabel(IntEnum):
MINOR = 0
MAJOR = 1
class HandRecog:
def __init__(self, hand_label):
def update_hand_result(self, hand_result):
self.hand_result = hand_result

def get_signed_dist(self, point):
"""
returns signed euclidean distance between '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):
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
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
        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()

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

Output

Fig:7.1 Output Window

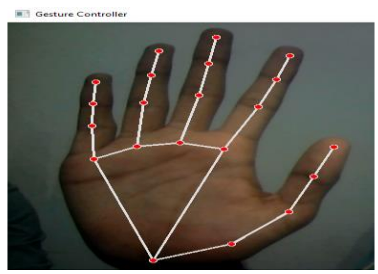


Fig:7.2 Palm Detection

7.2 Voice Control Code Snippet

```
import pyttsx3
import speech_recognition as sr
from datetime import date
import time
import webbrowser
import datetime
from pynput.keyboard import Key, Controller
import pyautogui
import sys
import os
from os import listdir
from os.path import isfile, join
```

```
import smtplib
import wikipedia
import Gesture_Controller
#import Gesture_Controller_Gloved as Gesture_Controller
import app

r = sr.Recognizer()
keyboard = Controller()
engine = pyttsx3.init('sapi5')
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

file_exp_status = False
files =[]
path = ''
is_awake = True #Bot status

def reply(audio):
app.ChatBot.addAppMsg(audio)

print(audio)
engine.say(audio)
engine.runAndWait()

def wish():
hour = int(datetime.datetime.now().hour)

if hour>=0 and hour<12:
reply("Good Morning!")
elif hour>=12 and hour<18:
reply("Good Afternoon!")
else:
reply("Good Evening!")
```

```
reply("I am Jerry, how may I help you?")

# Set Microphone parameters
with sr.Microphone() as source:
r.energy_threshold = 500
r.dynamic_energy_threshold = False

def record_audio():
with sr.Microphone() as source:
r.pause_threshold = 0.8
voice_data = ''
audio = r.listen(source, phrase_time_limit=5)

try:
voice_data = r.recognize_google(audio)
except sr.RequestError:
reply('Sorry my Service is down. Plz check your Internet connection')
except sr.UnknownValueError:
print('cant recognize')
pass
return voice_data.lower()

def respond(voice_data):
global file_exp_status, files, is_awake, path
print(voice_data)
voice_data.replace('jerry','')
app.eel.addUserMsg(voice_data)

if is_awake==False:
if 'wake up' in voice_data:
is_awake = True
wish()

elif 'hello' in voice_data:
```

```
wish()

elif 'what is your name' in voice_data:
reply('My name is Jerry!')

elif 'date' in voice_data:
reply(today.strftime("%B %d, %Y"))

elif 'time' in voice_data:
reply(str(datetime.datetime.now()).split(" ")[1].split('.')[0])

elif 'search' in voice_data:
reply('Searching for ' + voice_data.split('search')[1])
url = 'https://google.com/search?q=' + voice_data.split('search')[1]
try:
webbrowser.get().open(url)
reply('This is what I found')
except:
reply('Please check your Internet')

elif 'location' in voice_data:
reply('Which place are you looking for ?')
temp_audio = record_audio()
app.eel.addUserMsg(temp_audio)
reply('Locating...')
url = 'https://google.nl/maps/place/' + temp_audio + '/&amp;'
try:
webbrowser.get().open(url)
reply('This is what I found')
except:
reply('Please check your Internet')

elif ('bye' in voice_data) or ('by' in voice_data):
reply("Good bye! Have a nice day.")
```

```
is_awake = False

elif ('exit' in voice_data) or ('terminate' in voice_data):
if Gesture_Controller.GestureController.gc_mode:
Gesture_Controller.GestureController.gc_mode = 0
app.ChatBot.close()
sys.exit()

elif 'launch gesture recognition' in voice_data:
if Gesture_Controller.GestureController.gc_mode:
reply('Gesture recognition is already active')
else:
gc = Gesture_Controller.GestureController()
t = Thread(target = gc.start)
t.start()
reply('Launched Successfully')

elif ('stop gesture recognition' in voice_data) or ('top gesture recognition' in voice_data):
if Gesture_Controller.GestureController.gc_mode:
Gesture_Controller.GestureController.gc_mode = 0
reply('Gesture recognition stopped')
else:
reply('Gesture recognition is already inactive')

elif 'copy' in voice_data:
with keyboard.pressed(Key.ctrl):
keyboard.press('c')
keyboard.release('c')
reply('Copied')

elif 'page' in voice_data or 'pest' in voice_data or 'paste' in voice_data:
with keyboard.pressed(Key.ctrl):
keyboard.press('v')
```

```
keyboard.release('v')
reply('Pasted')

elif 'list' in voice_data:
counter = 0
path = 'C://'
files = listdir(path)
filestr = ""
for f in files:
counter+=1
print(str(counter) + ': ' + f)
filestr += str(counter) + ': ' + f + '<br>'
file_exp_status = True
reply('These are the files in your root directory')
app.ChatBot.addAppMsg(filestr)

elif file_exp_status == True:
counter = 0
if 'open' in voice_data:
if isfile(join(path,files[int(voice_data.split(' ')[-1])-1])):
os.startfile(path + files[int(voice_data.split(' ')[-1])-1])
file_exp_status = False
else:
try:
path = path + files[int(voice_data.split(' ')[-1])-1] + '//'
files = listdir(path)
filestr = ""
for f in files:
counter+=1
filestr += str(counter) + ': ' + f + '<br>'
print(str(counter) + ': ' + f)
reply('Opened Successfully')
```

```
app.ChatBot.addAppMsg(filestr)

except:
reply('You do not have permission to access this folder')

if 'back' in voice_data:
filestr = ""
if path == 'C://':
reply('Sorry, this is the root directory')
else:
a = path.split('//')[:-2]
path = '//'.join(a)
path += '//'
files = listdir(path)
for f in files:
counter+=1
filestr += str(counter) + ': ' + f + '<br>'
print(str(counter) + ': ' + f)
reply('ok')
app.ChatBot.addAppMsg(filestr)

else:
reply('I am not functioned to do this !')


t1 = Thread(target = app.ChatBot.start)
t1.start()

# Lock main thread until Chatbot has started
while not app.ChatBot.started:
time.sleep(0.5)

wish()
voice_data = None
while True:
```

```
if app.ChatBot.isUserInput():
#take input from GUI
voice_data = app.ChatBot.popUserInput()
else:
#take input from Voice
voice_data = record_audio()

#process voice_data
if 'jerry' in voice_data:
try:
#Handle sys.exit()
respond(voice_data)
except SystemExit:
reply("Exit Successfull")
break
except:
#some other exception got raised
print("EXCEPTION raised while closing.")

break
```

Output

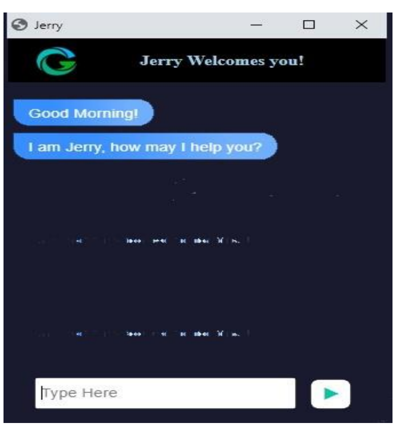
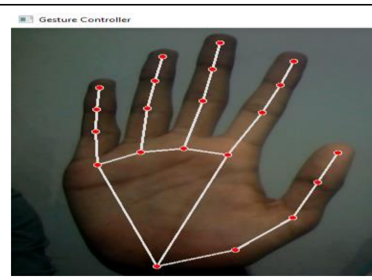


Fig:7.3 Output Window **Fig:7.4 Voice input, output window**

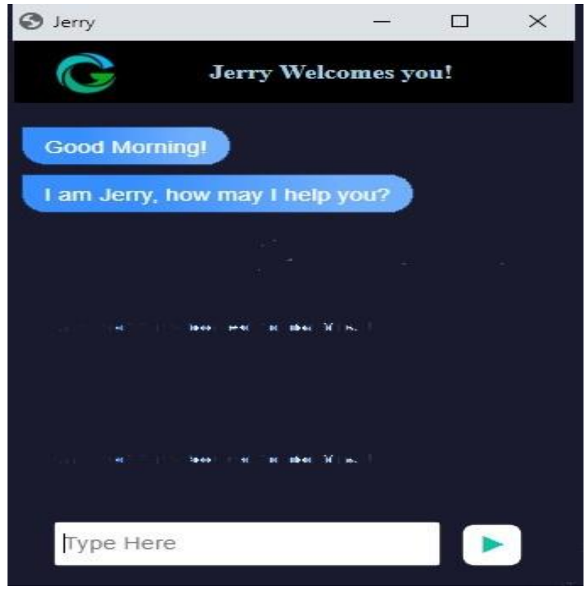
CHAPTER 8

SYSTEM TESTING

8.1 Gesture Control testing

Test Case	**Expected**	**Result**
Executing Gesture Control App	**A pop up of camera window**	
Detection of Hand	**Allotting co-ordinates to hands**	
Detection of objects other than hands	**No co-ordinates are allotted**	

8.2 Voice Control testing

Test Case	Expected	Result
Executing Voice Control App	**Windows pop up with appropriate wish**	
Giving voice command	**Process and perform the task**	

Giving in-correct voice command	**Displaying appropriate message**	

CHAPTER 9

SNAPSHOTS

9.1 Gesture Controller

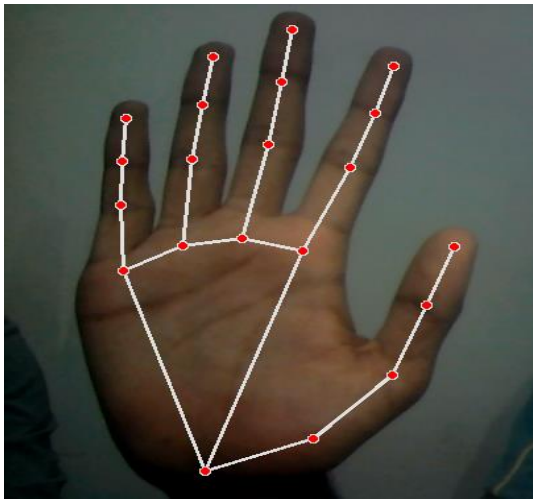
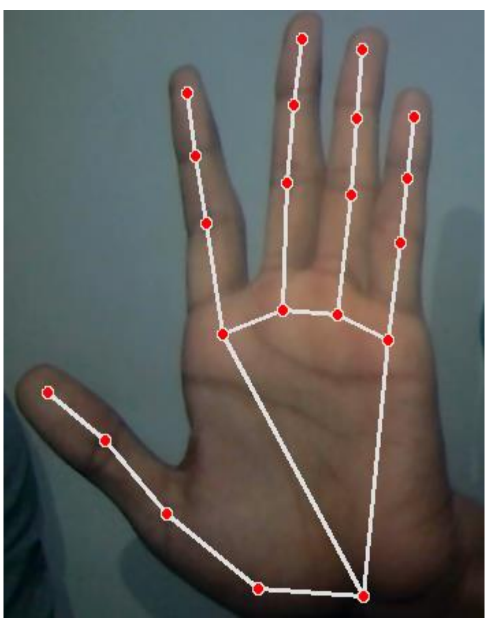


Fig:9.1 Neutral Gesture



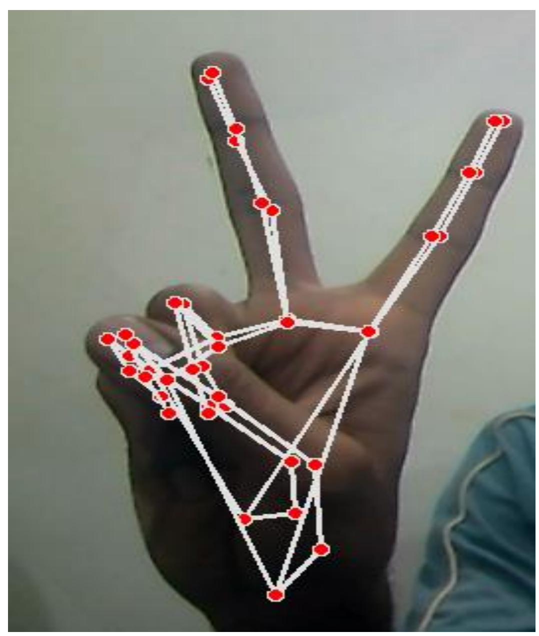


Fig:9.2 Drag and Drop

Fig:9.3 Move Cursor

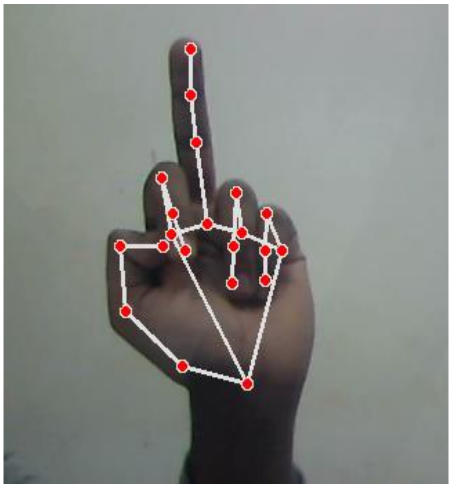


Fig:9.4 Left Click

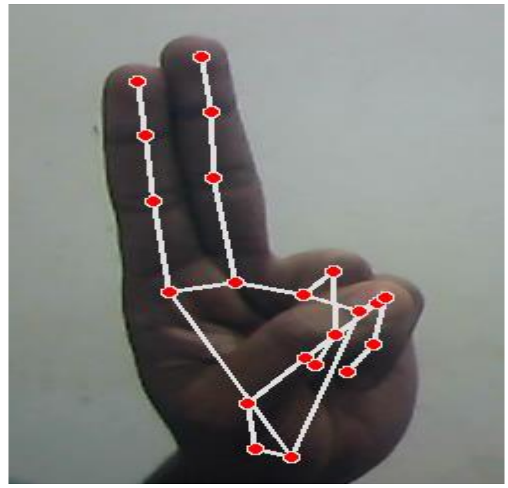


Fig:9.5 Double Click

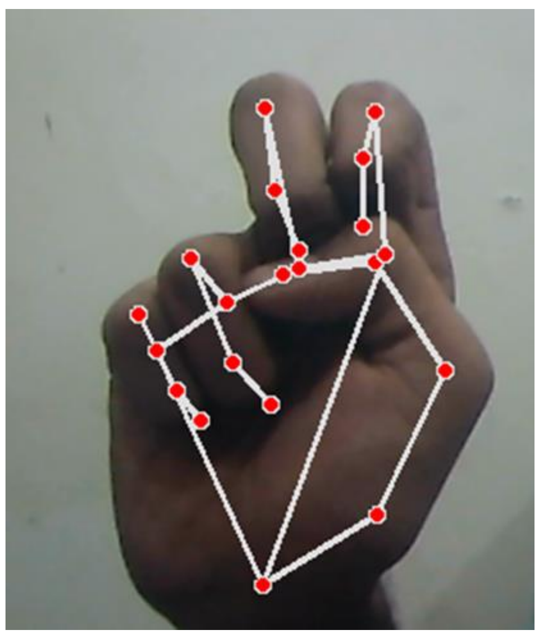


Fig:9.6 Drag

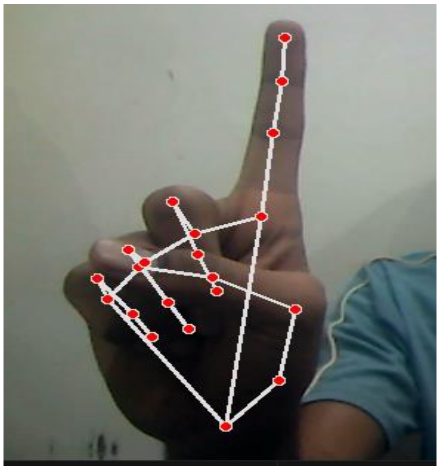


Fig:9.7Right Click

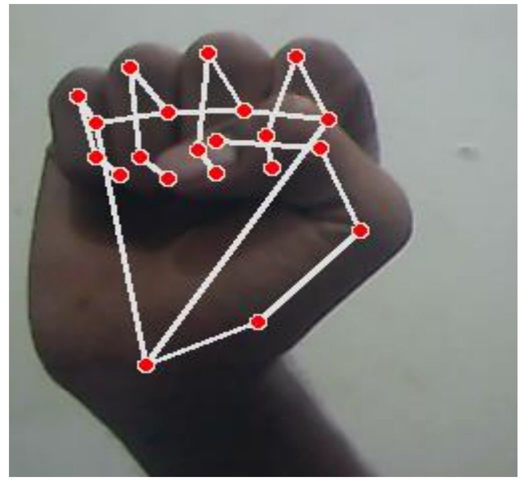


Fig:9.8 Multiple Item Select

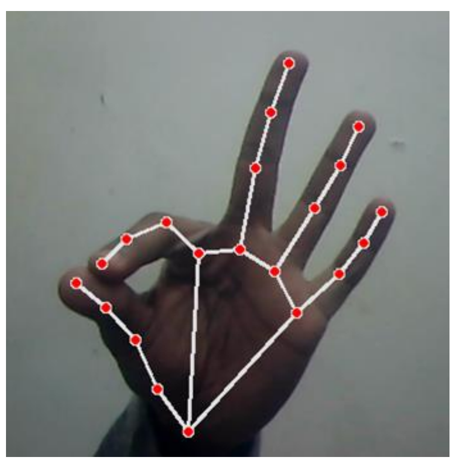


Fig:9.9 Volume Control

9.2 Voice Assistant Features

Fig:9.10 Launch/Stop Gesture Recognition

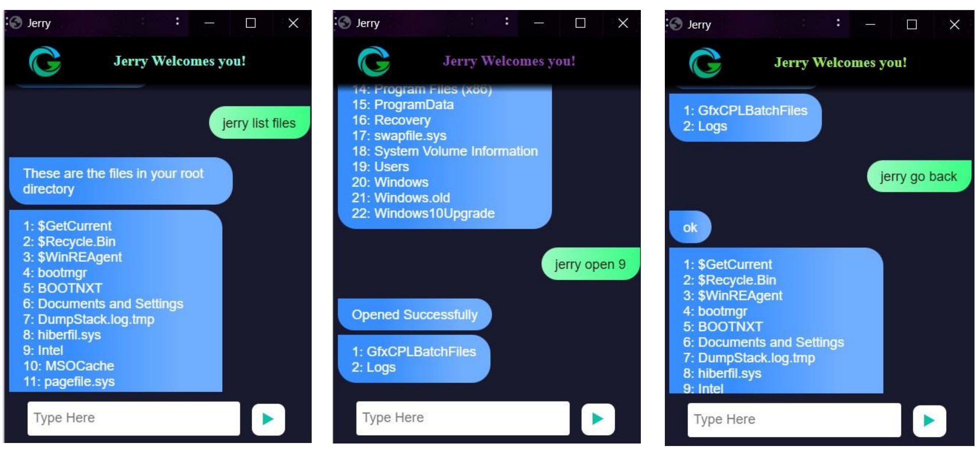


Fig:9.11 File Navigation

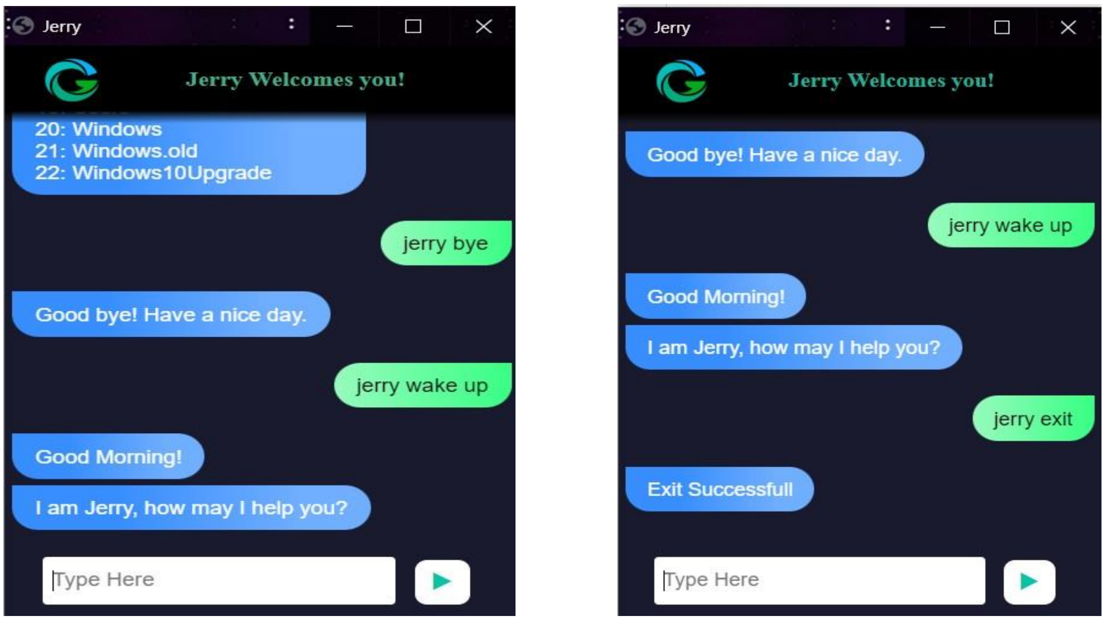


Fig:9.12 Sleep/Wakeup & Exit

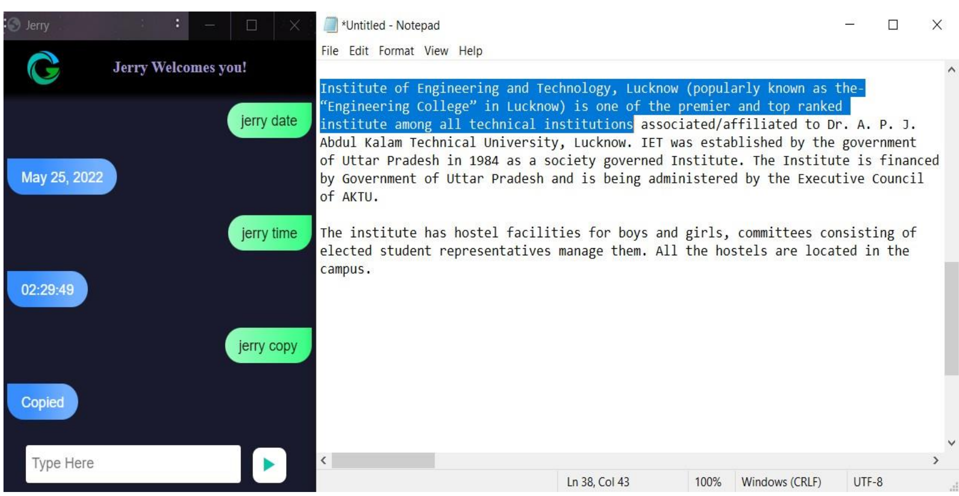


Fig:9.13 Current Date and Time

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

The already proposed models made significant improvements in the human control interaction with respect to mouse functions, yet they suffered from some of the drawbacks and limitations. Requirement of gloves, complex gestures, and limited functions are some of them. Through our project we have aimed at solving the above limitations by eliminating gloves and including most of the functions with the help of simple hand gestures. We have further Integrated this virtual system with speech recognition which will input commands like cut, copy, paste, etc and process it to perform the same.

The basic goal of the virtual mouse system is to control the mouse cursor and complete activities without needing a physical mouse by using hand gestures and voice commands. This proposed system is created by using a webcam (or any built-in camera) that recognises hand gestures and hand tip movement and processes these frames to perform the relevant mouse actions using the notion of speech recognitionto quickly follow voice commands and perform mouse activities.

The model upon rigorous testing has come out to be highly accurate and sophisticated showing enormous improvements with respect to prior existing models. Since the proposed model has been tested for high sophistication, the virtual mouse can be used for real-time applications. Because the proposed mouse system may be operated digitally utilising hand gestures and voice commands rather than the traditional physical mouse, it will be of more value in combating the propagation of viruses like COVID-19 in the current context.

Virtual Mouse will be introduced soon to replace the conventional computer mouse, making it easier for users to connect with and administer their computers. In order to correctly track the user's gesture, the software must be fast enough to capture and process every image and speech command.

Other features and improvements could be added to make the application moreuser-friendly, accurate, and adaptable in different contexts. The following are the enhancements and functionalities that are required:

Smart Recognition Algorithm

Using the palm and numerous fingers, additional functions such as enlarging and reducing the window, and so on, can be implemented.

Better Performance

The response time is largely influenced by the machine's hardware, which includes the processor's processing speed, the amount of RAM available, and the webcam's characteristics. As a result, when the software is performed on a respectable machine with a webcam that operates well in various lighting conditions and a better quality microphone that can detect voice instructions correctly and rapidly, the programme may perform better.

REFERENCES

- N. Li, X. Chen, Y. Feng and J. Huang, "Human–Computer Interaction Cognitive Behavior Modeling of Command and Control Systems", IEEE Internet of Things Journal, vol. 9, no. 14, pp. 12723-12736, July 2022.

- V. V. Reddy, T. Dhyanchand, G. V. Krishna and S. Maheshwaram, "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition", 2020 IEEE-HYDCON, pp. 1-5, 2022.

- X. Zhang, Y. Sun and Y. Zhang, "Evolutionary Game and Collaboration Mechanism of Human–Computer Interaction for Future Intelligent Aircraft Cockpit Based on System Dynamics", IEEE Transactions on Human-Machine Systems, vol. 52, no. 1, pp. 87-98, Feb. 2022.

- L. Dan, "Construction and design of visual information platform in human-computer interaction", 2022 IEEE Asia-Pacific Conference on Image Processing, pp. 152-157, 2022.

- Y. Shen, L. Yang, E. S. L. Ho and H. P. H. Shum, "Interaction-Based Human Activity Comparison", IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 8, pp. 2620-2633, Aug. 2020.

- M. Nazar, M. M. Alam, E. Yafi and M. M. Su'ud, "A Systematic Review of Human–Computer Interaction and Explainable Artificial Intelligence in Healthcare With Artificial Intelligence Techniques", *IEEE Access*, vol. 9, pp. 153316-153348, 2021.

- A. de Souza Vieira, M. Ribeiro Filho and C. de Salles Soares Neto, "Production and Evaluation of an Educational Process for Human–Computer Interaction (HCI) Courses", IEEE Transactions on Education, vol. 64, no. 2, pp. 172-179, May 2021.

- Himanshu Bansal, Rijwan Khan, "A review paper on human computer interaction" International Journals of advanced research in Computer Science and Software Engineering, Volume 8, Issue 4, April 2018

- N. Subhash Chandra, T. Venu, P. Srikanth, "A Real-Time Static & Dynamic Hand Gesture Recognition System" International Journal of Engineering Inventions Volume 4, Issue 12, August 2015

- S. Shiriam, B. Nagaraj, J. Jaya, "Deep learning based real time AI Virtual Mouse system using computer vision to avoid COVID-19 spread", Hindawi Journal of Healthcare Engineering, October 2021.

- Hritik Joshi, Nitin Waybhase, Ratnesh Litoria, "Towards controlling mouse through hand gestures: A novel and efficient approach", Medi-caps University, May 2022

- Mohhamad Rafi, Khan Sohail, Shaikh Huda, "Control mouse and computer system using voice commands", International Journal of Research in Engineering and Technology", Volume 5, Issue 3, March 2016

- C. Jeon, O.-J. Kwon, D. Shin and D. Shin, "Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction", *IEEE Access*, vol. 5, pp. 25181-25188, 2017.

- Aashni Haria, Archanasri Subramanian, Nivedhitha Asokkumar, Shristi Poddar and Jyothi S Nayak, "Hand Gesture Recognition for Human Computer Interaction", Procedia Computer Science, vol. 115, 2017.

- R. Agrawal and N. Gupta, "Real Time Hand Gesture Recognition for Human Computer Interaction", 2016 IEEE 6th International Conference on Advanced Computing (IACC), pp. 470-475, 2016.

- S. M. S. Shajideen and V. H. *Preetha*, Hand Gestures - Virtual Mouse for Human Computer Interaction, pp. 543-546, 2018.

- E. A. Lavrov, A. A. Volosiuk, N. B. Pasko, V. P. Gonchar and G. K. Kozhevnikov, Computer Simulation of Discrete Human-Machine Interaction for Providing Reliability and Cybersecurity of Critical Systems, pp. 67-70, 2018.

- K. Marasek, A. Romanowski and M. Sikorski, Emerging trends and novel approaches in interaction design, pp. 1231-1234, 2017

- Y. Ji, Y. Yang, F. Shen, H. T. Shen and X. Li, "A Survey of Human Action Analysis in HRI Applications", IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 7, pp. 2114-2128, July 2020.

- J. Xu, H. Wang, J. Zhang and L. Cai, "Robust Hand Gesture Recognition Based on RGB-D Data for Natural Human–Computer Interaction", *IEEE Access*, vol. 10, pp. 54549-54562, 2022.

- Y. Wang, "Design of Online Reading System Based on Human–Computer Interaction Information Fusion Technology", 2022 IEEE Asia-Pacific Conference on Image Processing Electronics and Computers (IPEC), pp. 1325-1327, 2022.