SMART CONTROL OF COMPUTERS WITH HAND GESTURES AND VOICE

Submitted by

A PROJECT REPORT

A.ELAVARASAN	**20202016**
V.JAYA KRISHNAN	**20202023**
A.NAVEEN KUMAR	**20202032**

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

INFORMATION TECHNOLOGY

PAAVAI ENGINEERING COLLEGE (AUTONOMOUS)

ANNA UNIVERSITY – CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project report " **SMART CONTROL OF COMPUTERS WITH HAND GESTURES AND VOICE**" is the work of "**A.ELAVARASAN (20202018)**", "**V.JAYA KRISHNAN(20202023)** " and "**A.NAVEEN KUMAR(20202032)"** who carried out the project work under my supervision.

SIGNATURE

Dr .B. VENKATESAN, M.E., Ph.D,
HEAD OF THE DEPARTMENT,
Department of Information Technology,
Paavai Engineering College,
Namakkal-637018.

SIGNATURE

Dr. G. MADASAMY RAJA, M.E., Ph.D.,
PROFESSOR ,
Department of Information Technology,
Paavai Engineering College,
Namakkal-637018.

Submitted for the University Examination held on ..……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We **"A. ELAVARASAN(20202018)"**, **"V. JAYA KRISHNAN(20202023) "** and "**A.NAVEEN KUMAR"** (20202032) hereby declare that the project report titled **"SMART CONTROL OF COMPUTERS WITH HAND GESTURES AND VOICE"** done by us under the guidance of **Dr. G. MADASAMY RAJA, M.E., Ph.D.,** at Paavai Engineering College is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology. Certified further that, to the best of our knowledge, the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on their or any other candidate.

1.

2.

3.

DATE:

PLACE:

SIGNAGTURE OF THE CANDIDATES

ACKNOWLEDGEMENT

A great deal of arduous work and effort has been spent in implementing this project work. Several special people have guided us and have contributed significantly to this work and so this becomes obligatory to record our thanks to them.

We express our profound gratitude to our honorable **Chairman, Shri.CA.N.V. NATARAJAN, B.Com., F.C.A.,** and also to our **Correspondent, Smt. MANGAI NATARAJAN, M.Sc.,** for providing all necessary facilities for the successful completion of our project.

We wish to express our sincere thanks to our respected **Director-Administration, Dr. K. K. RAMASAMY, M.E., Ph.D.,** for all the blessing and help provided during the period of project work.

We would like to thank our respected **Principal, Dr. M. PREM KUMAR, M.E., Ph.D.,** for allowing us to do this project and providing required time to complete the same.

We wish to express our deep gratitude to **Dr.B. VENKATESAN, M.E., Ph.D., Head of the Department,** for the able guidance and useful suggestions, which helped us for completing project work in time.

We express our sincere thanks **to Professor Dr. G. MADASAMY RAJA, M.E., Ph.D., Project Coordinator as well as our Project Guide** for the useful suggestions, which helped us for completing the project work in time.

We would like to extend our sincere thanks **to all our department staff members and our parents** for their advice and encouragement to do the project work with full interest and enthusiasm.

ABSTRACT

Human Computer Interaction is to improve the interaction between users and computers by making the computer more receptive to user needs. Human Computer Interaction with a personal computer today is not just limited to keyboard and mouse interaction. Interaction between humans comes from different sensory modes like gesture, speech, facial, voice and body expressions. Being able to interact with the system naturally is becoming ever more important in many fields of Human Computer Interaction. We have designed a robust marker less hand gesture recognition system which can efficiently track both static and dynamic hand gestures. Our system translates the detected voice and gesture into actions such as opening websites and launching applications like VLC Player and PowerPoint. The dynamic gesture is used to shuffle through the slides in presentation. The camera is used to detect the motion of the hand and move the cursor. We used the hand mark, which is likely a hand skeleton, to measure the motion of the hands. In recent years, the intersection of artificial intelligence (AI) and human mobility has received widespread attention due to its many applications in fields such as health, sports, security, and entertainment. This article provides an overview of machine learning using Python, a programming language widely used in AI research and development.

LIST OF ABBREVIATION

NLP - Natural Language Processing

CNN - Conclution Neural Network

MNN - Modular Nural Network

MP - Multilayer Perception

ADL - Advantages of Deep Learning

RNN - Recurrent Neural Networks

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Human movement recognition, the process of analyzing and understanding human actions from visual data, plays a crucial role in numerous applications, including surveillance, healthcare monitoring, gesture control systems, sports analytics, and virtual reality. With the advent of advanced machine learning and computer vision techniques, researchers have made significant strides in accurately detecting, tracking, and interpreting human movements. Python, with its rich ecosystem of libraries and frameworks, has emerged as a preferred choice for implementing movement recognition algorithms due to its simplicity, flexibility, and extensive community support. The primary goal of this era is to apprehend human hand gestures and translate them into commands that the computer can understand. For instance, a specific gesture may want to correspond to a command like “replica” or “paste,” or maybe working gadget features consisting of putting the computer to sleep or establishing a selected application.

Deep studying, especially Convolutional Neural Networks (CNNs), is frequently used to teach these structures to understand specific hand gestures. The device is skilled with heaps of pics of numerous hand gestures, allowing it to study and as it should be become aware of each gesture. This technology has a huge range of applications, from controlling presentations and gambling games to navigating through virtual surroundings. It’s a step toward creating a more immersive and interactive computing experience. However, growing a reliable hand gesture popularity gadget isn't always a trivial task. It calls for a complete

dataset of hand gestures and sophisticated gadget gaining knowledge of algorithms. Despite those challenges, the ability benefits of this generation make it a promising field of research. In end, clever computers using hand gestures are transforming the way we can interact with our digital devices, making the interplay greater intuitive and engaging. They represent the destiny of human-pc interaction, wherein instructions are as simple as a wave of the hand.

1.1 Python

Python is an interpreted, object -oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for uses such as a scripting or glue language to connect existing components together . Python's simple , easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages ,which encourages program modularity and code reuse. The python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Often , programmers use Python because of the increased productivity it provides . Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging python programs is easy a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expression, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in python itself, testifying to python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source the fast edit-test-debug cycle makes this simple approach very effective.

In this section, we provide practical tips on using deep learning models to recognize languages that use Python libraries such as TensorFlow and PyTorch. We discuss the advantages of using these libraries to build deep learning models, including ease of use, flexibility, and broad community support. Through code examples and tutorials, we show how to use TensorFlow and PyTorch to design, train, and evaluate CNNs and RNNs for cognitive processing. Additionally, we discuss best practices and optimization techniques to achieve optimal performance in deep learning-based cognitive processing.

1.2 Neural Network

Neural Networks are layers of nodes, much like the human brain is made up of neurons. Nodes within individual layers are connected to adjacent layers. The network is said to be deeper based on the number of layers it has. A single neuron in the human brain receives thousands of signals from other neurons. In an artificial neural network, signals travel between nodes and assign corresponding weights. A heavier weighted node will exert more effect on the next layer of nodes. The final layer compiles the weighted inputs to produce an output. Deep learning systems require powerful hardware because they have a large amount of data being processed and involve several complex mathematical calculations.

1.3 Deep Learning

CNN is one of the variations of the multilayer Perceptron. CNN can contain more than 1 convolution layer and since it contains a convolution layer the network is very deep with fewer parameters. Artificial Neural Networks are used in various classification tasks like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Input Layer, Hidden Layer, Output Layer.

1.4 Implementation

The first step is to gather complete craft data. It allows you to take pictures or videos of different gestures from different angles, lighting conditions and backgrounds. The collected data are then preprocessed. This may include resizing images, converting buildings to grayscale (if color is not a factor), and adjusting pixel values accordingly. Deep learning models such as Convolutional Neural Network (CNN) are then trained on this pre-processed dataset.

The model learns to recognize gesture types by searching for patterns in the training data. Once the model is trained, it can be used for real-time gesture recognition. This involves receiving video streams from the network, preprocessing the video images in the same way as the training data, and then inserting these images into the trained image.

1.5 Objective

The goal of intelligent computer systems using gestures is to provide a simple and natural way for users to interact with their computers. Create a system that can accurately recognize gestures in real time. Each detected gesture must correspond to a specific command or action on the computer. This can be basic tasks like clicking and scrolling or more complex tasks like controlling movement.

The system should be accessible and provide the user with visual information about the alcohol detected and the actions that triggered it.The system should be versatile enough to be used for a variety of purposes, from presentations to games. The system should be able to recognize gestures and respond in real time, providing a seamless user experience. For users with physical disabilities, this technology can provide an alternative way to interact with their computers.

CHAPTER 2

LITERATURE REVIEW

Hand gesture recognition research is classified in three categories. First "Glove based Analysis" attaching sensor with gloves mechanical or optical to transduces flexion of fingers into electrical signals for hand posture determination and additional sensor for position of the hand. This sensor is usually an acoustic or a magnetic that attached to the glove. Look-up table software toolkit provided for some applications to recognize hand posture.

The first approach is "Vision based Analysis" that human beings get information from their surroundings, and this is probably most difficult approach to employ in satisfactory way. Many different implementations have been tested so far. One is to deploy 3-D model for the human hand. Several cameras attached to this model to determine parameters corresponding for matching images of the hand, palm orientation and joint angles to perform hand gesture classification. Lee and Kunii developed a hand gesture analysis system based on a three-dimensional hand skeleton model with 27 degrees of freedom. They incorporated five major constraints based on the human hand kinematics to reduce the model parameter space search. To simplify the model matching, specially marked gloves were used
.

The Third implementation is "Analysis of drawing gesture" use stylus as an input device. These drawing analysis lead to recognition of written text. Mechanical

sensing work has used for hand gesture recognition at vast level for direct and virtual environment manipulation. Mechanically sensing hand posture has many problems like electromagnetic noise, reliability, and accuracy. By visual sensing gesture interaction can be made potentially practical but it is most difficult problem for machines.

These two parts of body (Hand & Arm) have most attention among those people who study gestures in fact much reference only consider these two for gesture recognition. The majority of automatic recognition systems are for deictic gestures (pointing), emblematic gestures (isolated signs) and sign languages. Some are components of bimodal systems, integrated with speech recognition. Some produce precise hand and arm configuration while others only coarse motion .

Stark and Kohler developed the ZYKLOP system for recognizing hand poses and gestures in real-time. After segmenting the hand from the background and extracting features such as shape moments and fingertip positions, the hand posture is classified. Temporal gesture recognition is then performed on the sequence of hand poses and their motion trajectory. A small number of hand poses comprises the gesture catalog, while a sequence of these makes a gesture .

Similarly, Maggioni and Kammerer described the Gesture Computer, which recognized both hand gestures and head movements. There has been a lot of interest in creating devices to automatically interpret various sign languages to aid the deaf community. One of the first to use computer vision without requiring the user to wear anything special was built by Starner, who used HMMs to recognize a limited vocabulary of ASL sentences. The recognition of hand and arm gestures has been applied to entertainment applications.

Freeman developed a real-time system to recognize hand poses using image moments and orientations histograms and applied it to interactive video games. Cutler and Turk described a system for children to play virtual instruments and interact with life like characters by classifying measurements based on optical flow .

Conclusion

The current computer plays a important position in enhancing performance by using expediting procedures and lowering operation time. To input statistics, various styles of technology are utilized such as virtual keyboards, laser keyboards, and wireless mice. These modern methods facilitate the seamless transmission of information, similarly, optimizing the consumer experience. Additionally, continuous advancements in hardware additives like the mouse make sure that workloads are minimized, and reaction times are notably stepped forward. Overall, the blended utilization of these modern-day technology outcomes in a streamlined and efficient computing environment, empowering users to perform responsibilities with greater ease and speed. The evolution of pc tools and peripherals continues to pave the manner for greater productiveness and efficiency, supplying customers with the approach to maximize their talents and achieve top-quality overall performance in numerous computing duties.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System

- "Finger Motion Tracking System" that makes use of an external hardware devices called leap motion controller which is used to capture the movement of user's hands and fingers, allowing them to interact naturally with digital content.
- This method has an accuracy of 90.125%. To setup a virtual cursor and keyboard technology, convex hull algorithm is used which is a complex procedure.
- Multiple background hand gestures recognition can be a limitation. Color segmentation procedure is also a complex algorithm used in development of Hand Gesture based Mouse.
- The teaching of multi-touch and mid-air gestures is more difficult than that of single touch gestures.

3.1.1 Disadvantages of Existing System

- Threshold
- Low time response
- Inaccurate
- Huge time response

3.2 Proposed System

The proposed method involves utilizing motion capture technology to both execute and analyze the data. This advanced technique not only accelerates the processing time but also cuts down on the overall time required for execution, ultimately enhancing performance efficiency. By harnessing motion capture technology, tasks that previously demanded significant time and effort to complete can now be accomplished swiftly and with improved results. This streamlined process allows for more precise and detailed analysis of the data, contributing to better decision-making and overall outcomes. Embracing motion capture technology in data processing not only expedites the workflow but also elevates the quality of results, setting a solid foundation for enhanced performance across various applications**.**

3.2.1 Advantage of Proposed System

- Easy to detect the finger postures
- High accuracy
- Fast response
- Easy to track the moments
- User friendly
-

3.2.2Feasibility Study

A feasibility study for a modern computer using hand gestures project is essential to assess whether the project is viable, practical, and worth pursuing. Here are key aspects to consider in a feasibility study for a sign language-related project.

3.2.3 Technical Feasibility:

- Valuate the availability and maturity of the required technologies for sign recognition, accuracy, and performance. Ensure that the necessary hardware and software tools are readily accessible.
- Assess the availability of skilled professionals, including deep learning experts, computer vision specialists, and sign language experts, to work on the project.

3.2.4 Operational Feasibility:

- Resource Availability: Assess whether you can secure the necessary resources, including human resources, time, and infrastructure, to develop and maintain the project.
- Scalability: Determine whether the project can be scaled to accommodate a growing user base and evolving technological needs.

Chapter 4

SYSTEM SPEFICATION

4.1 HARDWARE REQUIREMENTS

- Processor : Intel core processor 2.6.0 GHZ
- RAM : 4 GB
- Hard disk : 160 GB
- Compact Disk : 650 Mb
- Keyboard : Standard keyboard
- Monitor : 14-inch color monitor

4.2 SOFTWARE REQUIREMENTS

• Operating System : Windows OS

• Front End : PYTHON

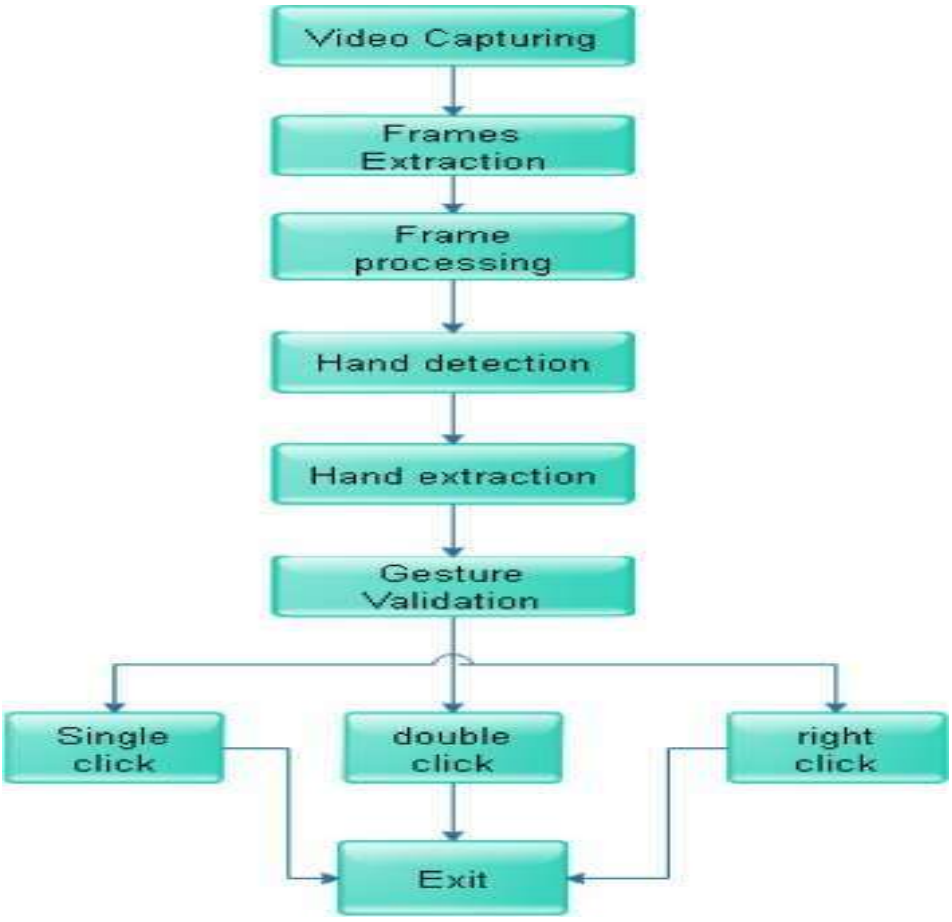
• IDE : PYCHARAM & VSCODE

• Application : WINDOWS APPLICATION

CHAPTER 5

SYSTEM DESIGN

Software architecture involves the high-level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability, and availability. Software architecture must describe its group of components, their connections, interactions among them and deployment configuration of all components.



Block diagram

5.1 Feature Extraction Technology

This section focuses on feature extraction technology used in conjunction with traditional machine learning algorithms for motion. Feature extraction plays an important role in identifying relevant information from raw sensor data or video frames. We discuss traditional extraction techniques such as time domain features, frequency domain features, and spatial features extracted from image or video frames. Additionally, we also provide Python code examples that demonstrate how to use extraction techniques to prepare input data for traditional learning machines such as SVM, k-NN, and random forest.

5.2 Neural Network

Neural Networks are layers of nodes, much like the human brain is made up of neurons. Nodes within individual layers are connected to adjacent layers. The network is said to be deeper based on the number of layers it has. A single neuron in the human brain receives thousands of signals from other neurons.

In an artificial neural network, signals travel between nodes and assign corresponding weights. A heavier weighted node will exert more effect on the next layer of nodes. The final layer compiles the weighted inputs to produce an output. Deep learning systems require powerful hardware because they have a large amount of data being processed and involve several complex mathematical calculation

5.3 Deep Learning Models

The analysis of deep learning models (such as CNN and RNN) and their applications in gesture recognition focuses on the use of Python libraries such as TensorFlow and PyTorch. Convolutional Neural Network (CNN) in Action Recognition This chapter provides an overview of the Convolutional Neural

Network (CNN) and its application in action recognition. CNNs are widely used in image and video processing due to their ability to capture features of the scene. We dive into the architecture of CNNs, including convolution, pooling, and all layers, and discuss how they lend themselves to motion. Using Python libraries such as TensorFlow and PyTorch, we show how to use CNN to identify and interpret human speech in video files.

5.4 Recurrent Neural Networks (RNN)

In Action Recognition we focus on Recurrent Neural Networks (RNN) and their applications in action recognition. RNNs are well-suited for modeling sequential data, making them suitable for predicting human behavior. We discuss RNN models that include recurrent techniques such as short-term temporal (LSTM) and recurring unit (GRU) and show how they can be applied to business experience. Through practical examples using Python libraries such as TensorFlow and PyTorch, we demonstrate the use of RNN to detect and classify human movements.

5.5 Python library

In this section, we provide practical tips on using deep learning models to recognize languages that use Python libraries such as TensorFlow and PyTorch. We discuss the advantages of using these libraries to build deep learning models, including ease of use, flexibility, and broad community support. Through code examples and tutorials, we show how to use TensorFlow and PyTorch to design, train, and evaluate CNNs and RNNs for cognitive processing. Additionally, we discuss best practices and optimization techniques to achieve optimal performance in deep learning-based cognitive processing.

CHAPTER 6

PROJECT DESCRIPTION

6.1 MODULES DESCRIPTION

A module is a separate unit of software or hardware. Typical characteristics of modular components include portability, which allows them to be used in a variety of systems and interoperability.

• Module 1: Hand image acquisition

• Module 2: Plot points using hand skeleton

• Module 3:Gesture recognition

• Module 4:Voice recognition

• Module 5: Track the moments

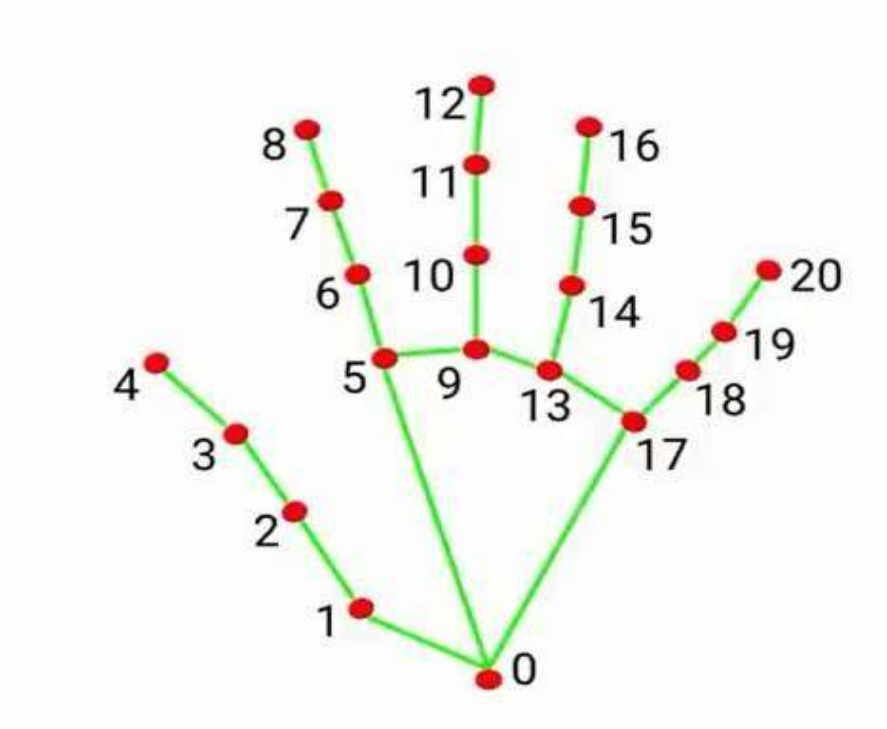
6.1.1 Hand Image Acquisition

During daily life, the hand gesture serves as a natural mode of communication, especially prevalent among individuals who may face challenges with speech or hearing. Yet, the realm of human-computer interaction is increasingly embracing gesture-based systems, paving the way for various application scenarios. One such example is the ability to capture hand images in real-time through input from a built-in camera connected to the system. Gesture recognition, a well-established field, has captured attention for decades, with two primary methods in use today: one reliant on professional settings and the other leveraging image processing techniques. However, the efficacy of gesture recognition solely based on the 16 extracted features through image processing still poses limitations. Notably, hand images obtained from a web camera are

stored in the system's memory, showcasing how technological advancements continue to shape our interactions and experiences.

6.1.2 Plot Points Using Hand Skeleton

The Hand Landmarked uses a model bundle with two packaged models: a palm detection model and a hand landmarks detection model. You need a model bundle that contains both these models to run this task. The hand landmark model bundle detects the key point localization of 21 hand-knuckle coordinates within the detected hand regions. The model was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds. The hand landmarker model bundle contains a palm detection model and a hand landmarks detection model. The Palm detection model locates hands within the input image, and the hand landmarks detection model identifies specific hand landmarks on the cropped hand image defined by the palm detection model.



Hand Landmark

Hand Landmark points

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

Since running the palm detection model is time consuming, when in video or live stream running mode, Hand Landmarker uses the bounding box defined by the hand landmarks model in one frame to localize the region of hands for subsequent frames. Hand Landmarker only re-triggers the palm detection model if the hand landmarks model no longer identifies the presence of hands or fails to track the hands within the frame. This reduces the number of times Hand Landmarker tiggers the palm detection model.

6.1.3 Gesture Recognition

The Gesture Recognizer task is an advanced tool that empowers users to effortlessly identify and interpret hand gestures in real time. It not only offers real-time recognition of hand gestures but also supplies detailed results concerning the recognized gestures and hand landmarks. One of the key benefits of this task is that it allows the mapping of specific hand gestures to trigger certain application functionalities, providing a seamless user experience.

Through its utilization of a machine learning model, the task can easily process image data, whether in the form of static images or a continuous stream, granting flexibility in data input methods. The key outputs of this task include the precise localization of hand landmarks in both image and world coordinates, enabling accurate spatial analysis. Additionally, the task provides information about handedness, distinguishing between left and right hands, which can be vital for certain applications. Furthermore, the task categorizes hand gestures among multiple hands, minimizing ambiguity and enhancing the overall interpretability of the results.

Overall, Gesture Recognizer task stands out as a comprehensive solution for hand gesture recognition, bridging the gap between image analysis and user interaction. Its robust features and versatile functionalities make it a valuable asset for developers seeking to incorporate intuitive hand gesture controls into their applications.

6.1.4 Voice Recognition

Speech recognition technology, a critical component of artificial intelligence (AI) systems, allows machines to simulate human behavior by observing and learning from their environment. This capability enables computers and software to comprehend human speech, facilitating faster and more accurate data processing. Moreover, speech recognition models are integral to voice assistant programs like Siri and Alexa, revolutionizing the way users interact with technology through natural language communication.

In the realm of AI, Natural Language Processing (NLP) plays a fundamental rolein transforming human language data into a format that machines can comprehend. By incorporating speech recognition and AI within NLP frameworks, the efficiency and precision of language processing are significantly enhanced, ushering in a new era of automated communication.

Businesses across various industries are harnessing the power of speech-to-text software and AI-driven speech recognition to optimize their operations and enhance customer satisfaction. The synergistic combination of speech recognition AI and natural language processing allows companies to transcribe important conversations, such as calls and meetings, with unparalleled accuracy and speed. Tech giants like Apple, Google, and Amazon are at the forefront of utilizing AI-powered speech recognition solutions to deliver seamless customer interactions and elevate user experiences to new heights.

6.1.5Track moments

Pointing with the gesture is accomplished by moving the cursor until the pointer is positioned over a desired area or object on the screen. This pointing action is fundamental in interacting with digital interfaces as it allows users to select, click on, or interact with various elements displayed on the screen.

When it comes to interaction techniques, left clicking using touch involves using both thumb and index finger to initiate an action. This action is often used for selecting objects, activating functions, or pressing buttons within an application or on a webpage.

On the other hand, right-clicking using touch requires the use of both index fingers to trigger a context-sensitive menu. This menu is designed to provide

CHAPTER 7

SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results This chapter provides a comprehensive overview of the various testing methods utilized to create a bug-free system. In order to achieve high-quality results, it is essential to apply diverse testing techniques throughout different stages of project development. The primary goal of testing is to identify and rectify any potential errors present in the system. Testing involves meticulously examining every possible fault or vulnerability within a work product, serving as a means to assess the functionality of individual components, subassemblies, or the final product itself. The process aims to validate that the software system successfully conforms to specified requirements and user expectations, ensuring that it operates without unacceptable failures. There exists a range of test types, each tailored to address specific testing prerequisites. System testing, for instance, focuses on verifying the overall integrated software system's compliance with requirements, ensuring consistent and foreseeable outcomes. An exemplary instance of system testing is the configuration-oriented system integration test, which aligns with process descriptions and flows, accentuating predetermined process connections and integration touchpoints.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

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

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

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

8.2 Future Enhancement

The Hand Gesture recognition is moving at tremendous speed for the futuristic products and services and major companies are developing technology based on the hand gesture system and that includes companies like Microsoft, Samsung, Sony and it includes the devices like Laptop, Handheld devices, Professional and LED lights. The verticals include, where the Gesture technology is and will be

evident are Entertainment, Artificial Intelligence, Education and Medical and Automation fields. And with lot of Research and Development in the field of Gesture Recognition Field, the use and adoption will become more cost effective and cheaper. It’s a brilliant feature turning data into features with mix of technology and Human wave.

CHAPTER 9

APPENDIX

9.1 Source Code

```
from tkinter import *
from tkinter.ttk import Progressbar
import os

root = Tk()
image = PhotoImage(file='images//2.png')
height = 239
width = 217
x=(root.winfo_screenwidth()//2)-(width//2)
y=(root.winfo_screenheight()//2)-(height//2)
root.geometry('{}x{}+{}+{}'.format(width,height,x,y))
root.overrideredirect(1)

root.wm_attributes('-topmost',True)
root.lift()
root.wm_attributes("-topmost",True)
root.wm_attributes("-disable",True)
root.wm_attributes("-transparentcolor","white")
bg_label = Label(root,image=image,bg='white')
bg_label.place(x=0,y=0)

Progr_label = Label(root,text="Please Wait...",font=('Comic Sans
MS',13,'bold'))
```

```
button1 = Button(root, text="Calculate", command=minimizeGUI,
Place=BOTTOM)
button1.pack()
button2 = Button(root, text="Calculate", command=exitGUI ,Place=BOTTOM)
button2.pack()




root.mainloop()

import os
from main import *

recognizer = sr.Recognizer()

def recognize_speech():
    with sr.Microphone() as source:
        print("Application name:")
        recognizer.adjust_for_ambient_noise(source, duration=1)
        audio = recognizer.listen(source)

    try:
        recognized_text = recognizer.recognize_google(audio)
        return recognized_text
    except sr.UnknownValueError:
        return "Could not understand audio"
    except sr.RequestError as e:
        return "Could not request results; {0}".format(e)
```

```
def file():
 speak("what application would you like to open")
 while True:

    recognized_text = recognize_speech()
    print("Application:", recognized_text)

    if "vs code" in recognized_text:
        speak("opening")
        vscodepath="C:\\Users\\elava\\AppData\\Local\\Programs\\Microsoft VS
Code\\Code.exe"
        os.startfile(vscodepath)
    elif "Adobe Reader" in recognized_text:
                speak("opening")
                vscodepath="C:\\Program Files (x86)\\Adobe\\Acrobat
DC\\Acrobat\\Acrobat.exe"
                os.startfile(vscodepath)
    elif "Photoshop" in recognized_text:
                speak("opening")
                vscodepath="C:\\Program Files\\Adobe\\Adobe Photoshop
2022\\Photoshop.exe"
                os.startfile(vscodepath)
    elif "brave Browser" in recognized_text:
                speak("opening")
                vscodepath="C:\\Program Files\\BraveSoftware\\Brave-
Browser\\Application\\brave.exe"
                os.startfile(vscodepath)
    elif "Excel" in recognized_text:
```

```
            speak("opening")
            vscodepath="C:\\Program Files\\Microsoft
Office\\root\\Office16\\EXCEL.EXE"
            os.startfile(vscodepath)
    elif "PowerPoint" in recognized_text:
            speak("opening")
            vscodepath="C:\\Program Files\\Microsoft
Office\\root\\Office16\\POWERPNT.EXE"
            os.startfile(vscodepath)
    elif "Outlook" in recognized_text:
            speak("opening")
            vscodepath="C:\\Program Files\\Microsoft
Office\\root\\Office16\\OUTLOOK.EXE"
            os.startfile(vscodepath)
    elif "Visual Studio" in recognized_text:
            speak("opening")
            vscodepath="C:\\Program Files\\Microsoft Visual
Studio\\2022\\Community\\Common7\\IDE\\devenv.exe"
            os.startfile(vscodepath)
    elif "Chrome" in recognized_text:
            speak("opening")
            vscodepath="C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe"
            os.startfile(vscodepath)
    elif "Microsoft edge" in recognized_text:
            speak("opening")
            vscodepath="C:\\Program Files
(x86)\\Microsoft\\Edge\\Application\\msedge.exe"
            os.startfile(vscodepath)
```

```
    elif "Firefox" in recognized_text:
            speak("opening")
            vscodepath="C:\\Program Files\\Mozilla Firefox\\firefox.exe"
            os.startfile(vscodepath)
    elif "exit" in recognized_text:
        speak("exit from the application model")
        break
    else:
       speak("it is invalide command,try again")


from main import *

recognizer = sr.Recognizer()

engine = pyttsx3.init()
engine = pyttsx3.init('sapi5')
voice = engine.getProperty('voices')
engine.setProperty('voice',voice[1].id)
engine.setProperty('rate', 140.5)
engine.setProperty('volume', 30)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def recognize_speech():
    with sr.Microphone() as source:
        print("website name:")
```

```
        recognizer.adjust_for_ambient_noise(source, duration=1)
        audio = recognizer.listen(source)

    try:
        recognized_text = recognizer.recognize_google(audio)
        return recognized_text
    except sr.UnknownValueError:
        return "Could not understand audio"
    except sr.RequestError as e:
        return "Could not request results; {0}".format(e)

def lib():
 speak("ask any questions")
 print("RULE 1 ==> wh type question accepted")
 print("RULE 2 ==> if you want details ask detail about")
 while True:

    recognized_text = recognize_speech()
    print("website:", recognized_text)
    if "details" in recognized_text:
                    speak("searching in")
                    recognized_text=recognized_text.replace("details","")
                    results=wikipedia.summary(recognized_text,sentences=10)
                    speak("Details about")
                    print(results)
                    speak(results)

    elif "what is" in recognized_text:
                    speak("searching in")
```

```
            recognized_text=recognized_text.replace("what is","")
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "why" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("why","")
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "when" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("when","")
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "where" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("where","")
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "which" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("which","")
```

```
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "who" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("who","")
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "whom" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("whom","")
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "whose" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("whose","")
            results=wikipedia.summary(recognized_text,sentences=5)
            speak("Details about")
            print(results)
            speak(results)
    elif "how" in recognized_text:
            speak("searching in")
            recognized_text=recognized_text.replace("How","")
            results=wikipedia.summary(recognized_text,sentences=5)
```

```
                speak("Details about")
                print(results)
                speak(results)
    elif "explain briefly" in recognized_text:
                speak("searching in")
                recognized_text=recognized_text.replace("How","")
                results=wikipedia.summary(recognized_text,sentences=20)
                speak("Details about")
                print(results)
                speak(results)
    elif "exit" in  recognized_text:
            speak("exit from the library model")
            break
    else:
            speak("try again")

import os
import subprocess
import  pyttsx3
from main import *
import pyautogui
import wmi
import cv2

engine = pyttsx3.init()
engine = pyttsx3.init('sapi5')
voice = engine.getProperty('voices')
engine.setProperty('voice',voice[1].id)
engine.setProperty('rate', 140.5)
```

```
engine.setProperty('volume', 30)


def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def restart_windows():
    os.system("shutdown /r /f /t 0")
def power_off_windows():
    os.system("shutdown /s /f /t 0")
def open_this_pc():
    subprocess.run(["explorer", "/select,", "::{20D04FE0-3AEA-1069-A2D8-
08002B30309D}"])
def open_start_menu():
    pyautogui.hotkey('win')
def open_windows_settings():
    subprocess.run("start ms-settings:", shell=True)
def enable_bluetooth():
    c = wmi.WMI()
    for interface in c.Win32_PnPEntity():
        if "Bluetooth" in interface.Caption:
            interface.Enable()
def open_bluetooth_settings():
    subprocess.run("start ms-settings:bluetooth", shell=True)
def disable_bluetooth():
    c = wmi.WMI()
    for interface in c.Win32_PnPEntity():
        if "Bluetooth" in interface.Caption:
            interface.Disable()
def open_wifi_settings():
```

```
    subprocess.run("start ms-availablenetworks:", shell=True)
def open_wireless_cast_settings():
    subprocess.run("start ms-settings-connectabledevices:", shell=True)
def open_projector_settings():
    subprocess.run("start ms-settings-displays:project", shell=True)
def open_camera():
    cap = cv2.VideoCapture(0) # 0 represents the default camera (usually built-
in)

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        cv2.imshow('Camera Feed', frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()



recognizer = sr.Recognizer()

def recognize_speech():
    with sr.Microphone() as source:
        print("Windows operation:")
        recognizer.adjust_for_ambient_noise(source, duration=1)
```

```
        audio = recognizer.listen(source)

    try:
        recognized_text = recognizer.recognize_google(audio)
        return recognized_text
    except sr.UnknownValueError:
        return "Could not understand audio"
    except sr.RequestError as e:
        return "Could not request results; {0}".format(e)

def cmdprogram():
 speak("command mode existing")
 while True:

    recognized_text = recognize_speech()
    print("Application:", recognized_text)

    if "shutdown" in recognized_text:
        speak("shutdowning")
        power_off_windows()
    elif "restart" in recognized_text:
        speak("restarting")
        restart_windows()
    elif "this PC" in recognized_text:
        speak("this pc is opening")
        open_this_pc()
    elif "open start" in recognized_text:
        speak("opening start menu")
        open_start_menu()
```

```
elif "open settings" in recognized_text:
    speak("opening setting")
    open_windows_settings()
elif "turn on Bluetooth" in recognized_text:
    speak("turn on Bluetooth")
    enable_bluetooth()
elif "turn off Bluetooth" in recognized_text:
    speak("turn off Bluetooth")
    disable_bluetooth()
elif "open Bluetooth settings" in recognized_text:
    speak("open Bluetooth settings")
    open_bluetooth_settings()
elif "open Wi-Fi settings" in recognized_text:
    speak("open wifi settings")
    open_wifi_settings()
elif "open cast settings" in recognized_text:
    speak("open wireless cast settings")
    open_wireless_cast_settings()
elif "open projector settings" in recognized_text:
    speak("open proujector settings")
    open_projector_settings()
elif "open camera" in recognized_text:
    speak("opening web camera")
    open_camera()
elif "exit" in recognized_text:
    speak("exit from the system model")
    break
else:
    speak("try again ")
```

```
import webbrowser
from main import *

recognizer = sr.Recognizer()

engine = pyttsx3.init()
engine = pyttsx3.init('sapi5')
voice = engine.getProperty('voices')
engine.setProperty('voice',voice[1].id)
engine.setProperty('rate', 140.5)
engine.setProperty('volume', 30)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def recognize_speech():
    with sr.Microphone() as source:
        print("website name:")
        recognizer.adjust_for_ambient_noise(source, duration=1)
        audio = recognizer.listen(source)

    try:
        recognized_text = recognizer.recognize_google(audio)
        return recognized_text
    except sr.UnknownValueError:
        return "Could not understand audio"
    except sr.RequestError as e:
```

```
        return "Could not request results; {0}".format(e)

def website():
 speak("what website would you like to open")
 while True:

   recognized_text = recognize_speech()
   print("website:", recognized_text)

   if "open WhatsApp" in recognized_text:
      speak("opening")
      url = "https://web.whatsapp.com/"
      webbrowser.open(url)
   elif "open Google" in recognized_text:
      speak("opening google")
      url="https://www.google.com/"
      webbrowser.open(url)
   elif "open YouTube" in recognized_text:
      speak("opening google")
      url="https://www.youtube.com/"
      webbrowser.open(url)
   elif "open Facebook" in recognized_text:
      speak("opening google")
      url="https://www.facebook.com/"
      webbrowser.open(url)
   elif "open FB" in recognized_text:
      speak("opening google")
      url="https://www.facebook.com/"
      webbrowser.open(url)
```

```
elif "open Instagram" in recognized_text:
    speak("opening google")
    url="https://www.instagram.com//"
    webbrowser.open(url)
elif "open insta" in recognized_text:
    speak("opening google")
    url="https://www.instagram.com//"
    webbrowser.open(url)
elif "open office" in recognized_text:
    speak("opening google")
    url="https://www.office.com/"
    webbrowser.open(url)
elif "open Microsoft office" in recognized_text:
    speak("opening google")
    url="https://www.office.com/"
    webbrowser.open(url)
elif "open open AI" in recognized_text:
    speak("opening google")
    url="https://openai.com/"
    webbrowser.open(url)
elif "open bing" in recognized_text:
    speak("opening google")
    url="https://www.bing.com/"
    webbrowser.open(url)
elif "open Gmail" in recognized_text:
    speak("opening google")
    url="https://mail.google.com/"
    webbrowser.open(url)
elif "open Microsoft" in recognized_text:
```

```
        speak("opening google")
        url="https://account.microsoft.com/"
        webbrowser.open(url)
    elif "open Flipkart" in recognized_text:
        speak("opening google")
        url="https://www.flipkart.com/"
        webbrowser.open(url)
import speech_recognition as sr
import pyttsx3
import datetime
import wikipedia
from data_collection import file_operation
from data_collection import systemoper
from data_collection import web
from data_collection import library

engine = pyttsx3.init()
engine = pyttsx3.init('sapi5')
voice = engine.getProperty('voices')
engine.setProperty('voice',voice[1].id)
engine.setProperty('rate', 140.5)
engine.setProperty('volume', 30)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def toDaysDate():
    hour = datetime.datetime.now().hour
```

```
    if hour >= 6 and hour<12:
        engine.say("Good morning sir!")
    elif hour >= 12 and hour<16:
        engine.say("Good afternoon sir!")
    elif hour >=16 and hour<18:
        engine.say("good evening sir!")
    else:
        engine.say("Good night")
    text_to_speak = "i am master Elavarasan Artificial Intelligence"
    #speak("hi my self is Helen")
    speak(text_to_speak)
    speak("welcome to the ai world")
recognizer = sr.Recognizer()

def recognize_speech():
    with sr.Microphone() as source:
        print("I AM READY TO HELP")
        recognizer.adjust_for_ambient_noise(source, duration=1)
        audio = recognizer.listen(source)

    try:
        recognized_text = recognizer.recognize_google(audio)
        return recognized_text
    except sr.UnknownValueError:
        return "Could not understand audio"
    except sr.RequestError as e:
        return "Could not request results; {0}".format(e)
def recognize_speech1():
    with sr.Microphone() as source:
```

```
        print("Search Datas:")
        recognizer.adjust_for_ambient_noise(source, duration=1)
        audio = recognizer.listen(source)

    try:
        recognized_text = recognizer.recognize_google(audio)
        return recognized_text
    except sr.UnknownValueError:
        return "Could not understand audio"
    except sr.RequestError as e:
        return "Could not request results; {0}".format(e)

if__name__== "__main__":
    toDaysDate()

    while True:

            recognized_text = recognize_speech1()
            print("You said:", recognized_text)
            if "hello" in recognized_text:
                speak("hi how are you")
            elif "what about you" in recognized_text:
                speak("I am fine")
            elif "I am fine" in recognized_text:
                speak("what about you")
            elif "tell about you" in recognized_text:
                speak("i am a voice assistent,i solve your questions and i can
help you")
                speak("i am developed by Elavarasan A")
```

```
            speak("i developed for help bliend people and future purpos")
        elif "about yourself" in recognized_text:
            speak("i am a voice assistent,i solve your questions and i can
help you")
            speak("i am developed by Elavarasan A")
            speak("i developed for help bliend people and future purpos")
        elif "thank you" in recognized_text:
            speak("you are welcome")
        elif "exit" in recognized_text:
            speak("Thank you for using me")
            break
        elif "open applications" in recognized_text:
            file_operation.file()
        elif "change system mode" in recognized_text:
            systemoper.cmdprogram()
        elif "open website" in recognized_text:
            web.website()
        elif "open library" in recognized_text:
            library.lib()


        else:
            speak("try again")


import cv2
import mediapipe as mp
import pyautogui
cap = cv2.VideoCapture(0)
```

```
hand_detector = mp.solutions.hands.Hands()
drawing_utils = mp.solutions.drawing_utils
screen_width,screen_height = pyautogui.size()
index_y=0
while True:
    _, frame = cap.read()
    frame = cv2.flip(frame,1)
    frame_height,frame_width,_ = frame.shape
    rgb_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
    output = hand_detector.process(rgb_frame)
    hands = output.multi_hand_landmarks
    if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(frame,hand)
            landmarks = hand.landmark
            for id, landmark in enumerate(landmarks):
                x = int(landmark.x*frame_width)
                y = int(landmark.y*frame_height)
                if id == 8:
                    cv2.circle(img=frame,center=(x,y),radius=10, color=(0,255,255))
                    index_x = screen_width/frame_width*x
                    index_y = screen_height/frame_height*y
                    pyautogui.moveTo(index_x,index_y)
                if id == 4:
                    cv2.circle(img=frame,center=(x,y),radius=10, color=(0,255,255))
                    thum_x = screen_width/frame_width*x
                    thum_y = screen_height/frame_height*y
                    print('double click',abs(index_y-thum_y))
                    if abs(index_y-thum_y) < 40:
```

```
                pyautogui.click(clicks=2)
                pyautogui.sleep(1)
                print('click')
            if id == 12:
                cv2.circle(img=frame,center=(x,y),radius=10, color=(0,255,255))
                mid_x = screen_width/frame_width*x
                mid_y = screen_height/frame_height*y
                print('single click',abs(index_y-mid_y))
                if abs(index_y-mid_y) < 25:
                    pyautogui.click()
                    pyautogui.sleep(1)
                    print('click')
            if id == 5:
                cv2.circle(img=frame,center=(x,y),radius=10, color=(0,255,255))
                scr_x = screen_width/frame_width*x
                scr_y = screen_height/frame_height*y
                print('scroll',abs(index_y-scr_y))
                if abs(index_y-scr_y) < 55:
                    pyautogui.scroll(-170)
                    pyautogui.sleep(1)
                    print('click')


    cv2.imshow('VR Mouse',frame)
    cv2.waitKey(1)
import subprocess
proc1 = subprocess.Popen(['python','main.py'])
proc2 = subprocess.Popen(['python','mouse.py'])
proc1.wait()
proc2.wait()
```

9.2 OUTPUT SCREENSHOTS

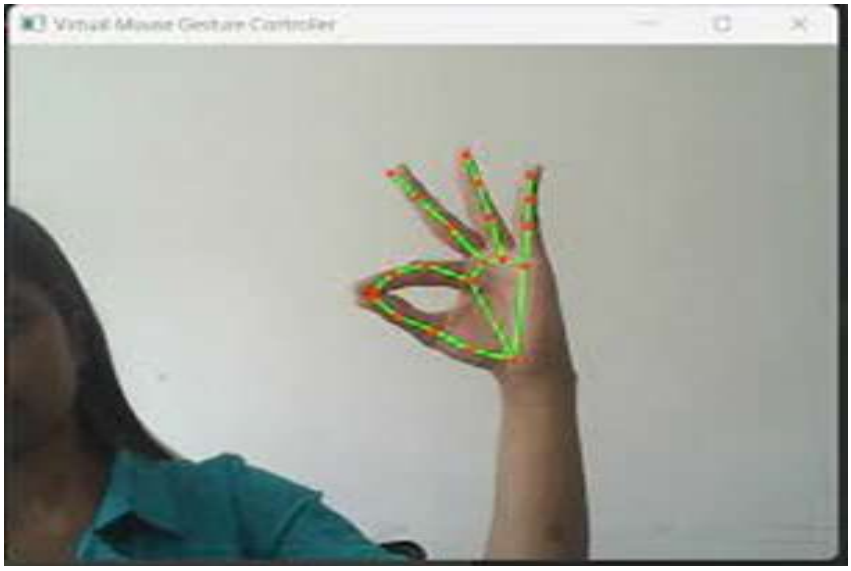


Fig 9.1 Input

Fig 9.2 Output

Fig 9.3 Palymouth

Fig 9.4 Startup Program

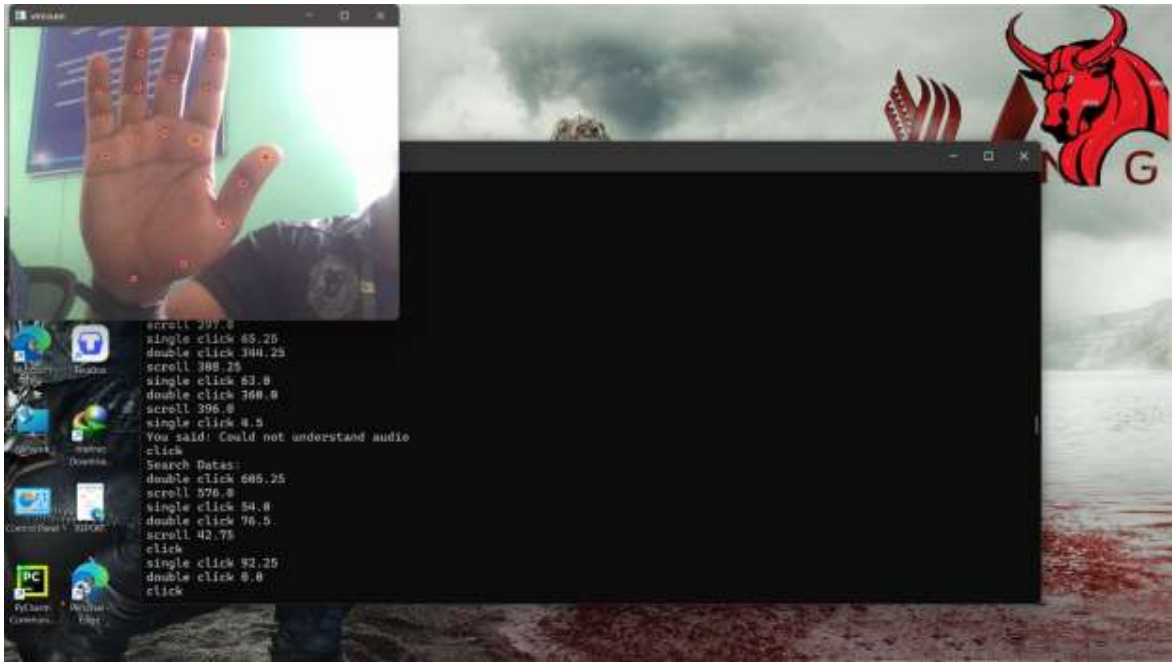


Fig 9.5 Hand Recognition and Points

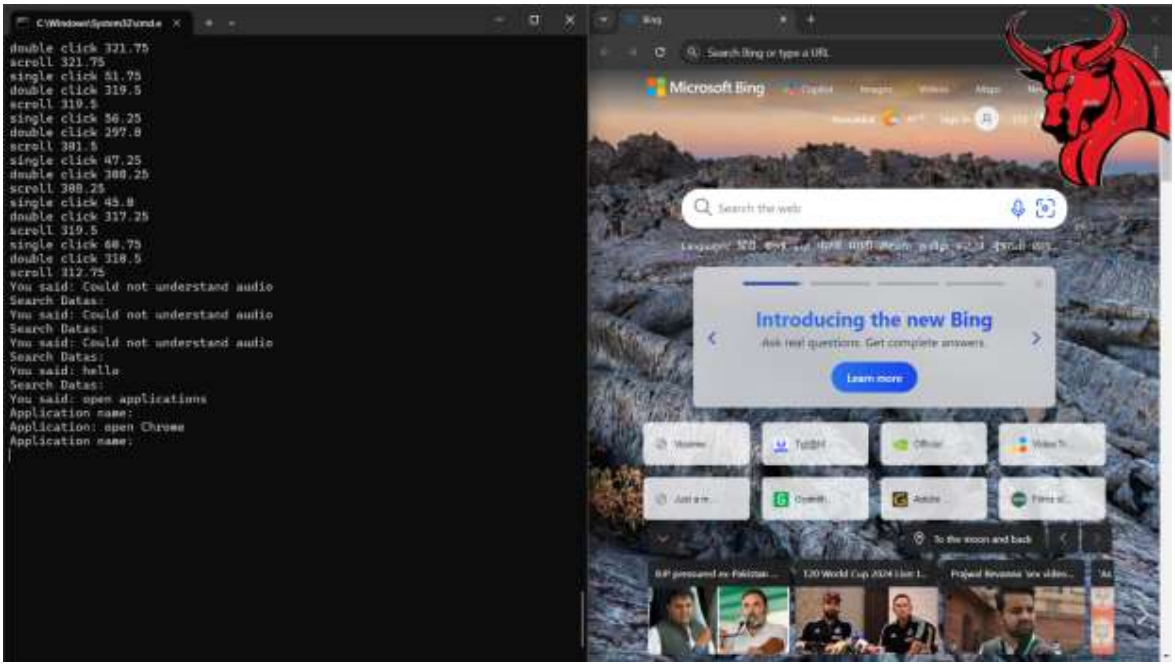


Fig 9.5 System Controling Using Voice

Fig 9.6 Output of Voicegpt

CHAPTER 10

REFERENCE

[1] T. Yang, Y. Xu, and "A Hidden Markov Model for Gesture Recognition", CMURI-TR-94 10, Robotics Institute, Carnegie Mellon Univ, Pittsburgh, PA, May 1994.

[2] Bretzner, L., Laptev, I., & Lindeberg, T. (2002). Hand gesture recognition using multi-scale color features, hierarchical models and particle filtering. Paper presented at the Proceedings of the fifth IEEE International Conference on Automatic face gesture recognition.

[3] Parvini, F., & Shahabi, C. (2007). An algorithmic approach for static and dynamic gesture recognition utilizing mechanical and biomechanical

[4] Murthy, G., & Jadon, R. (2009). A review of vision-based hand gesture recognition. International Journal of Information Technology and Knowledge Management, 2(2), 405-410.

[5] Sulochana M. Nadgeri, Dr. S.D. Sawarkar, Mr.A.D.Gawande, "Hand gesturerecognition using CAMSHIFT Algorithm", Third International Conference on Emerging Trends in Engineering and Technology, IEEE, 2010, India

[6] Li, L., & Zhang, L. (2012). Corner Detection of Hand Gesture. TELKOMNIKA Indonesia Journal of Electrical Engineering, 10(8), 2088-2094.

[7] Gupta, S., Jaafar, J., & Ahmad, W. F. W. (2012). Static hand gesture recognition using local Gabor filter. Procedia Engineering, 41, 827-832.

[8] Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Advances in neural information processing systems.

[9] Hasan, H., & Abdul-Kareem, S. (2014). Retracted article: Human-computer interaction using vision-based hand gesture recognition systems: A survey. Neural Computing and Applications, 25(2), 251-261.

[10] Nagashree R N, Stafford Michahial, Aishwarya G N, Beebi Hajira Azeez, Jayalakshmi M R, R Krupa Rani, “Hand gesture recognition using support vector machine”, The International Journal of Engineering and Science (IJES), June 2015.

[11] Ahuja, M. K., & Singh, A. (2015). Static vision-based Hand Gesture recognition using principal component analysis. Paper presented at the 2015 IEEE 3rd International Conference on MOOCs, Innovation, and Technology in Education (MITE).

[12] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553),436-444.

[13] Pujan Ziaie, Thomas M ¨uller , Mary Ellen Foster , and Alois Knoll“A Naïve Bayes Classifier with Distance Weighting for Hand-Gesture Recognition “,Munich,Dept. of Informatics VI, Robotics and Embedded Systems,Germany.

[14] Abadi, M., et al. (2016). TensorFlow: A system for large-scale machine learning. In the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16).

[15] Paszke, A., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems.

[16] Raghu Veera Chowdary, M Nagendra Babu, Thadigotla Venkata Subbareddy, Bommepall IEEE international conference on advanced communications, control and computing technologies, 2014.

[17] GRS Murthy, RS Jadon. Hand gesture recognition using neural networks IEEE 2nd international advance computing conference (IACC), 134-138, 2010.

[18] Yanmin Zhu, Zhibo Yang, Bo Yuan .Vision based hand gesture recognition international conference on service sciences (ICSS), 260-265, 2013.

[19] Alexandra Stefan, Vassilis Athitsos, Jonathan Alon, Stan Sclaroff. Translation and scale-invariant gesture recognition in complex scenes, international conference on Pervasive Technologies Related to Assistive Environments, 1-8, 2008.

[20] Mustafa, Mohammed. "A study on Arabic sign language recognition for differently abled using advanced machine learning classifiers." J. Ambient Intell. Humaniz. Comput. 12.3 (2021): 4101-4115.

[21] Camgoz, Necati Cihan, et al. "Sign language transformers: Joint end-to end sign language recognition and translation." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[22] Wadhawan, Ankita, and Parteek Kumar. "Sign language recognition systems: A decade systematic literature review." Archives of Computational Methods in Engineering 28.3 (2021): 785-813.

[23] Adaloglou, Nikolas, et al. "A comprehensive study on sign language recognition methods." arXiv preprint arXiv:2007.12530 2.2 (2020).

[24] Li, Dongxu, et al. "Transferring cross-domain knowledge for video sign language recognition." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[25] Jaramillo-Yánez, Andrés, Marco E. Benalcázar, and Elisa Mena Maldonado. "Realtime hand gesture recognition using surface electromyography and machine learning: a systematic literature review." Sensors 20.9 (2020): 2467.

[26] Sahoo, Ashok Kumar, Pradeepta Kumar Sarangi, and Parul Goyal. "Indian sign language recognition using soft computing techniques." Machine Vision Inspection Systems: Image Processing, Concepts, Methodologies and Applications 1 (2020): 37-65.