AI-Based Screen Controller with Hand Gesture

Submitted by

Rabia Luqman (BCS-19-12)
Muneeba Shahid (BCS-19-06)

Session

2019-2023

Supervised by

Ms Sehrish Raza

INSTITUTE OF COMPUTER SCIENCE&
INFORMATION TECHNOLOGY
THE WOMEN UNIVERSITY
MULTAN,PAKISTAN

FINAL APPROVAL

This is to certify that we have read this report submitted by ***Rabia Luqman and Muneeba Shahid*** and it is our judgment that this report is of sufficient standard to warrant its acceptance The Women University, Multan for the degree of BS (Computer Science).

Committee:

1. **External Examiner** ____________________

2. **Supervisor** ____________________
   Ms. Sehrish Raza
   Lecturer at
   The Women University, Multan

.

3. **Director** ____________________
   Dr. Khadija Kanwal
   Assistant Professor at
   The Women University, Multan

DEDICATION

To my Loving Parents & Great Teachers

This thesis stands as a testament to the profound impact you, my loving parents and great teachers, have had on my life. Your unwavering belief in me, unwavering support, and invaluable guidance have paved the way for my success. I am truly fortunate to have such remarkable individuals in my life who have shaped me into the person I am today.

ACKNOWLEDGMENT

All the praises, thanks and acknowledgments are for the creator of universe who gave me strength and enabled me to undertake and execute this task. First, I offer my gratitude to my supervisor, **Ms. Sehrish Raza**, who has supported me throughout my project with his patience and knowledge whilst allowing me to work in my own way.

I would like to express my gratitude to all those who gave me the possibility to complete this project. Thanks to all teachers of **CS&IT** who have guided me to the best. I would like to thank my **Family** and **Friends** those who helped me a lot and encouraged me in every despair moment and enabled me to face the challenges of this project. I want to express my special thanks to Institute of **CS&IT, WUM** for providing working environment and all other facilities that were needed throughout the academic session.

PROJECT BRIEF

PROJECT NAME	AI-Based Screen Controller with hand gesture
ORGANIZATION NAME	Institute of Computer Science and Technology
UNDERTAKEN BY	Rabia Luqman , Muneeba Shahid
SUPERVISED BY	Maam Sehrish Raza
STARTING DATE	January, 2023
COMPLETION DATE	July, 2023
COMPUTER USED	Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz 2.50 GHz 6.00 GB (5.89 GB usable)
OPERATING SYSTEM	64-bit operating system, x64-based processor Windows10 Pro
SOURCE LANGUAGE(S)	Python
DBMS USED	NO
TOOLS/PACKAGES	PyCharm/ Python

ABSTRACT

This thesis presents an AI-based screen controller that utilizes computer vision and hand gesture recognition techniques to enable hands-free interaction with a computer screen. The system captures video input from a webcam and employs the Hand Tracking Module to detect and track hand landmarks in real-time. By extending the index finger, users can control the mouse cursor, leveraging the program's mapping of hand coordinates to screen coordinates. Additionally, the system recognizes gestures such as extending both the index and middle fingers to simulate a mouse click, closing all fingers except the thumb for a right-click action, and extending the index and little fingers simultaneously to capture screenshots. This AI-powered screen controller offers a user-friendly and intuitive approach to screen navigation and interaction, providing an enhanced user experience and eliminating the need for conventional input devices.

TABLE OF CONTENTS

Contents **Page No.**

Chapter4
System Development & Implementation

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Background and Motivation

In recent years, AI-based technologies have gained significant popularity and have revolutionized various industries and fields. One area that has seen remarkable advancements is human-computer interaction. Traditional input devices, such as keyboards and mice, have limitations in terms of natural and intuitive interaction. Users often require additional training to master complex interfaces, leading to reduced productivity and user experience.
The motivation behind this research is to develop a more efficient and user-friendly screen controller using AI and hand gestures. By leveraging computer vision and machine learning techniques, the aim is to create a system that allows users to interact with their screens effortlessly, mimicking natural hand movements. This would not only enhance user experience but also open up new possibilities for applications in areas such as gaming, design, and virtual reality.

1.2 Problem Statement

The traditional input devices used for screen control have several limitations. Keyboards and mouse, although widely used, may not provide the most intuitive and efficient interaction experience. Users often need to memorize complex shortcuts and gestures, which can be time-consuming and prone to errors. There is a need for a more natural and user-friendly interface that simplifies screen control and improves overall productivity.

1.3 Objectives

The main objective of this research is to develop an AI-based screen controller using hand gestures. The specific objectives include

- Designing and implementing a hand tracking module that accurately detects and tracks hand movements in real-time.
- Developing algorithms to interpret hand gestures and map them to specific screen control actions.
- Integrating the screen controller with the operating system to enable seamless interaction with various applications and functionalities.
- Evaluating the performance and usability of the system through user studies and comparing it with traditional input devices.

1.4 Scope of the Research

This research focuses on the development of an AI-based screen controller using hand gestures. The scope of the research includes:

- Implementing a hand tracking module that utilizes computer vision techniques to detect and track hand movements with high accuracy.
- Creating an interface that translates hand gestures into specific screen control actions, such as cursor movement, clicking, scrolling, and other interactions.
- Testing and refining the system on various applications and use cases, including but not limited to gaming, design software, and virtual reality environments.

It is important to note that this research does not cover other forms of gesture recognition or non-hand-based input methods. The primary focus is on hand gestures and their application in screen control.

1.5 Thesis Organization

This thesis is organized into the following chapters:

- **Chapter 1: Introduction**
    - Provides an overview of the research topic and its significance
    - States the problem statement and objectives
    - Defines the scope of the research
    - Outlines the structure of the thesis
- **Chapter 2: Literature Review**
    - Reviews existing literature and research related to AI-based screen control and hand gesture recognition
    - Analyzes different approaches, algorithms, and technologies used in this field
    - Identifies gaps and limitations in the current body of knowledge
- **Chapter 3: Methodology**
    - Describes the methodology and approach used in developing the AI-based screen controller
    - Details the design of the hand tracking module and the algorithms for gesture interpretation
    - Explains the data collection process and the evaluation metrics used
- **Chapter 4: Implementation**
    - Presents the implementation details of the developed system
    - Describes the software architecture and the integration with the operating system
    - Provides insights into the challenges faced during implementation and their solutions
- **Chapter 5: Results and Evaluation**
    - Presents the results of the system's performance and usability evaluations
    - Analyzes the findings and compares them with traditional input devices
    - Discusses the strengths and limitations of the developed screen controller
- **Chapter 6: Conclusion and Future Work**
    - Summarizes the research findings and their implications
    - Discusses the contributions and achievements of the study
    - Provides recommendations for future enhancements and potential research directions

The structure of this thesis ensures a comprehensive exploration of the research topic, starting with an introduction and background and progressing through the implementation and evaluation stages.

Chapter 2
Literature Review

2.1 Gesture Recognition Techniques

Gesture recognition techniques play a crucial role in the field of human-computer interaction, enabling users to interact with computers and devices using hand gestures. There are various techniques employed for gesture recognition, including computer vision-based and machine learning-based approaches.

2.1.1 Computer Vision-based Gesture Recognition

Computer vision-based gesture recognition techniques utilize image processing and computer vision algorithms to detect and interpret hand gestures. These techniques typically involve extracting hand features, such as hand shape, position, and movement, from images or video frames. In the presented project, the Hand Tracking module employs computer vision techniques to detect and track hands in real-time using the Mediapipe library. It utilizes landmark detection to identify the positions of specific hand landmarks, allowing for gesture recognition based on hand poses and finger configurations.

2.1.2 Machine Learning-based Gesture Recognition

Machine learning-based gesture recognition techniques leverage the power of machine learning algorithms to recognize and classify hand gestures. These techniques involve training models on large datasets of hand gesture examples to learn patterns and relationships between input gestures and their corresponding outputs. The trained models can then be used to recognize and interpret new gestures. While the presented project focuses on computer vision-based techniques, machine learning algorithms can be integrated to enhance gesture recognition capabilities by training models to recognize specific hand gestures with high accuracy.

2.2 AI-based Human-Computer Interaction

AI-based human-computer interaction refers to the application of artificial intelligence techniques in enhancing the interaction between humans and computers. In the context of the project, the integration of hand tracking and gesture recognition with an AI virtual mouse allows users to control the mouse cursor on the screen using hand movements and gestures. This AI-powered interaction enables a more intuitive and natural way of interacting with computers, offering users greater flexibility and convenience.

2.3 Hand Tracking and Detection

Hand tracking and detection form the foundation of the presented project. The Hand Tracking module utilizes computer vision techniques and the Mediapipe library to detect and track hands in real-time video frames. It extracts hand landmarks and calculates their positions, allowing for precise tracking and analysis of hand movements and gestures. The module employs a hand detection model and hand landmark models provided by Mediapipe, enabling accurate and robust hand tracking capabilities.

2.4 Hand Gesture Recognition Systems

Hand gesture recognition systems are designed to interpret and recognize specific hand gestures performed by users. These systems utilize various techniques, such as computer vision, machine learning, or a combination of both, to analyze hand movements, hand configurations, and finger poses. The presented project implements hand gesture recognition by analyzing the positions of hand landmarks and the configuration of fingers. It recognizes gestures such as moving the mouse cursor, clicking, right-clicking, and taking screenshots based on the detected finger states and hand poses.

2.5 Summary

In summary, this literature review explored gesture recognition techniques, focusing on computer vision-based and machine learning-based approaches. It discussed the role of AI in human-computer interaction and the importance of hand tracking and detection in enabling gesture recognition. The review also highlighted the significance of hand gesture recognition systems in facilitating natural and intuitive user interactions with computers. The presented project leverages computer vision techniques, hand tracking, and gesture recognition to create an AI virtual mouse, enabling users to control the mouse cursor using hand gestures.

Chapter 3
System Architecture and Methodology

3.1 System Overview

The AI-based screen controller using hand gestures is designed to enable intuitive screen control through the recognition of hand movements and gestures. The system comprises several key components, including data collection and preprocessing, hand tracking module, gesture recognition module, and AI-based screen control algorithms. These components work together to process input hand gestures and perform corresponding screen control actions.

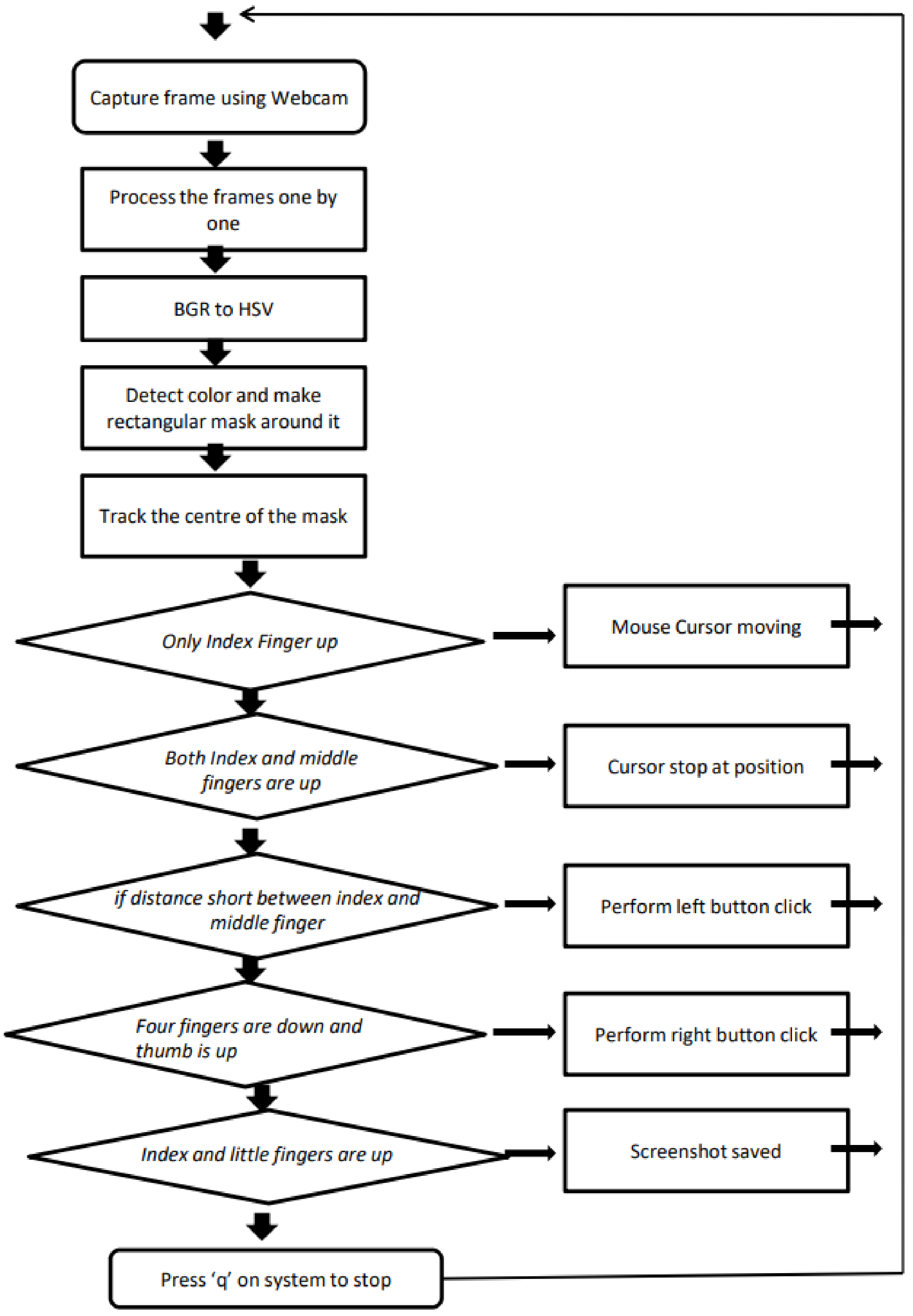

Capture frame using Webcam
Process the frames one by one
BGR to HSV
Detect color and make rectangular mask around it
Track the centre of the mask
Only Index Finger up
Mouse Cursor moving
Both Index and middle fingers are up
Cursor stop at position
if distance short between index and middle finger
Perform left button click
Four fingers are down and thumb is up
Perform right button click
Index and little fingers are up
Screenshot saved
Press 'q' on system to stop

Fig 3.1: System Overview

3.2 Block Diagram of system

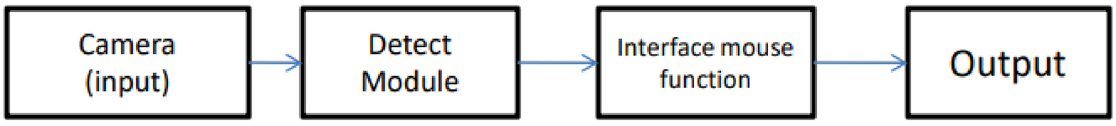


Fig 3.2: Block diagram of system

3.3 Flow chart of Modules of System

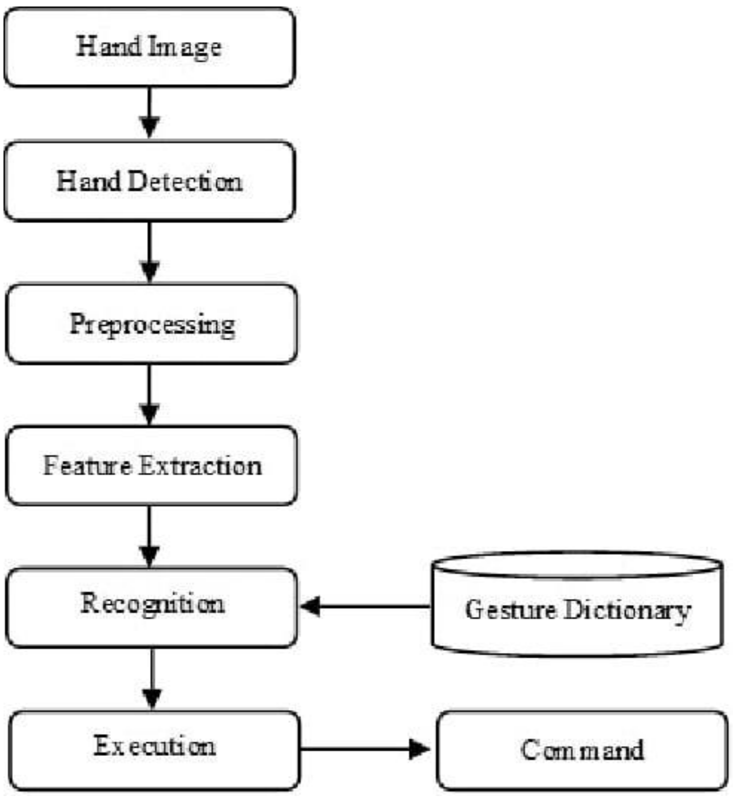


Fig 3.3: Flow chart

An AI virtual mouse with hand gestures would likely consist of the following modules:

- **Initialization:**
  Load AI model for hand gesture recognition
  Start video capture from a camera

- **Image Acquisition:**
  Capture a frame from the video
  Pre-process the image (resize, grayscale, etc.)

- **Hand Detection:**
  Use an object detection model to detect hands in the image
  If no hands are detected, go back to step 2

- **Hand Tracking:**
  Track the hand in subsequent frames to keep track of its movements.
  Store the hand's position and movements over time

- **Gesture Recognition**
  Use the hand's position and movements to recognize gestures.
  Map the recognized gestures to specific commands (e.g. swipe left to move back a page, swipe right to move forward)

- **Screen Control:**
  Execute the command associated with the recognized gesture
  Repeat steps 2 to 6 to continuously monitor and control the screen.

3.4 Use Case Diagram

Following diagram is the use case diagram of AI-Based screen controller using hand gestures. Use Case diagram helps you to easily understand the methodology of the system.

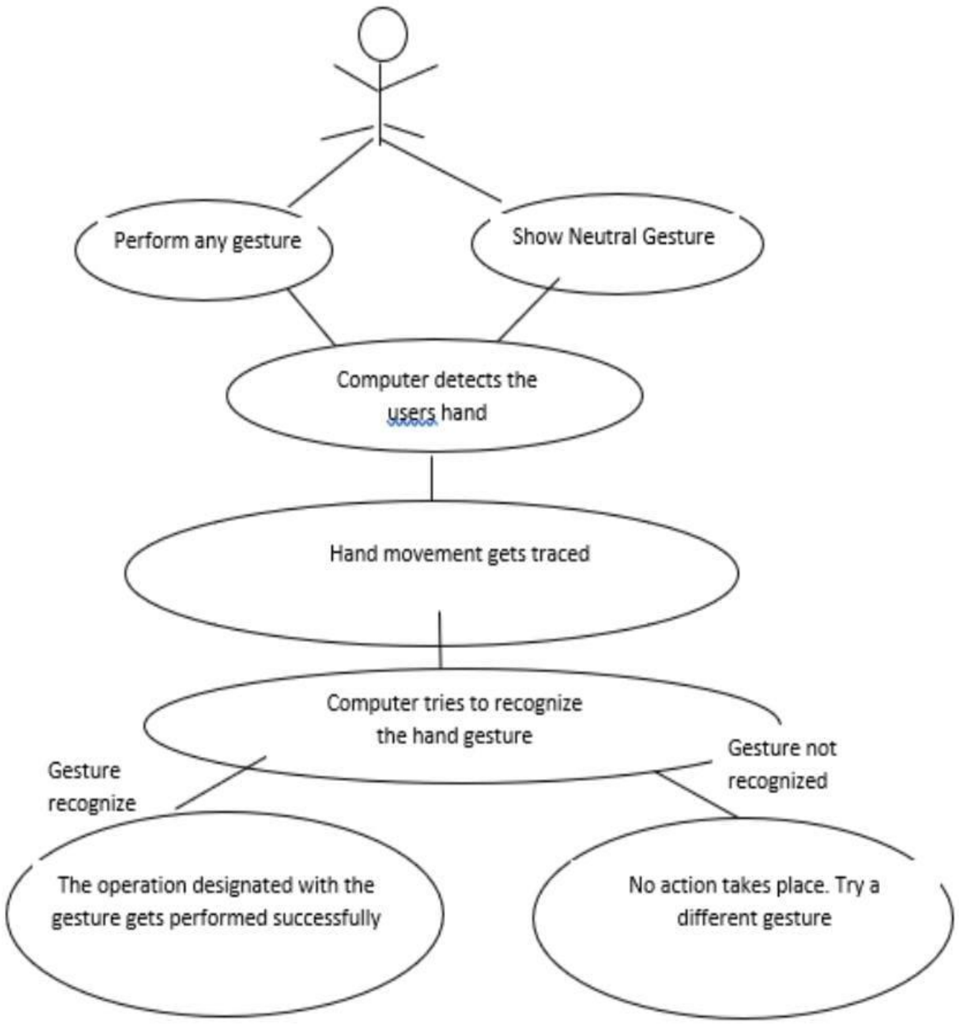

Perform any gesture
Show Neutral Gesture
Computer detects the users hand
Hand movement gets traced
Computer tries to recognize the hand gesture
Gesture recognize
Gesture not recognized
The operation designated with the gesture gets performed successfully
No action takes place. Try a different gesture

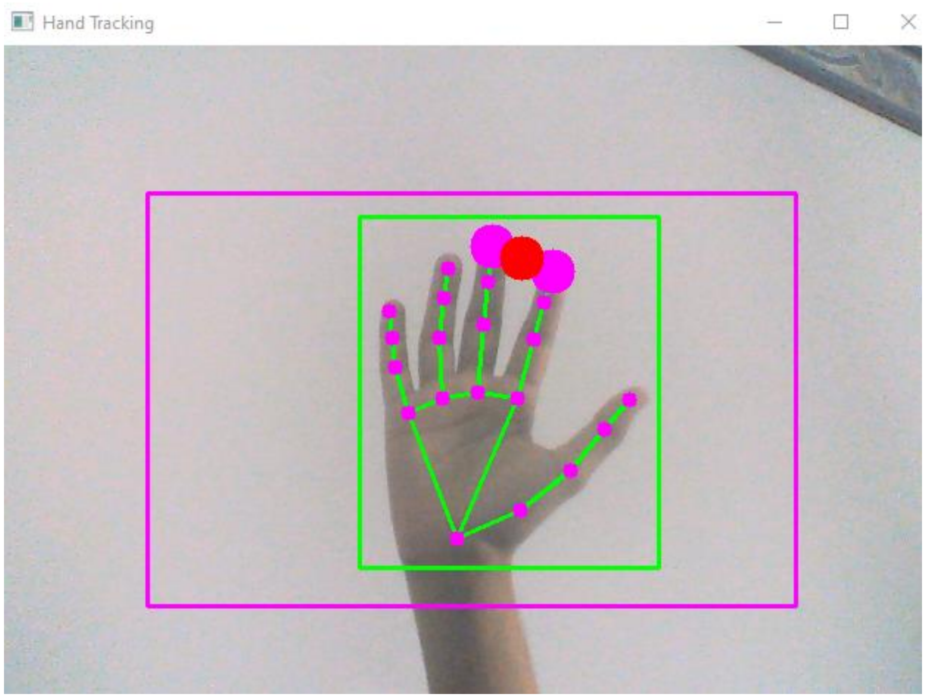
Fig 3.4: Use Case Diagram

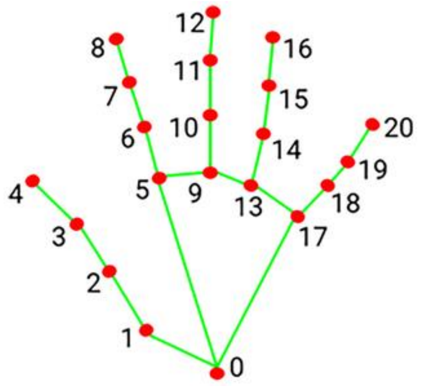
3.5 Data Collection and Preprocessing

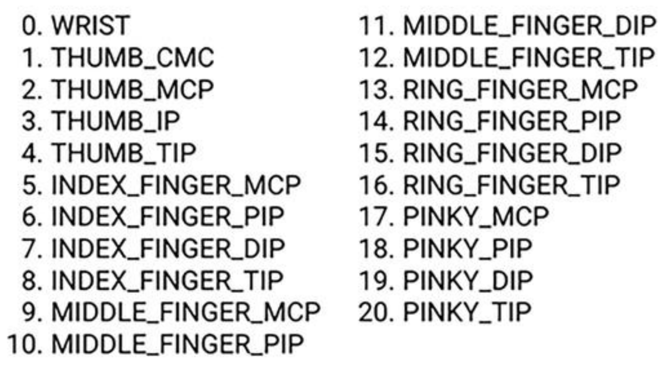
To train and evaluate the system, a dataset of hand gesture examples is collected and preprocessed. The data collection process involves capturing hand movements and gestures from the user. The collected data is then preprocessed to enhance its quality and effectiveness. This includes techniques such as resizing images, normalizing values, and extracting relevant features from the data.

3.6 Hand Tracking Module

The hand tracking module is responsible for detecting and tracking hand movements in real-time. It utilizes computer vision techniques and libraries such as OpenCV and mediapipe. The module processes input images or frames to detect the presence of hands and track their movements. The module also extracts hand landmarks, which represent key points on the hand, such as fingertips and joints.







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

Fig 3. 5: Hand Co-Ordinates **Table 3. 5: Hand Co-Ordinates**

3.7 Gesture Recognition Module

The gesture recognition module interprets the hand gestures detected by the hand tracking module and maps them to specific screen control actions. This module utilizes the finger positions and their configurations to recognize various gestures. The module employs machine learning techniques such as support vector machines (SVM) or deep learning algorithms for gesture classification

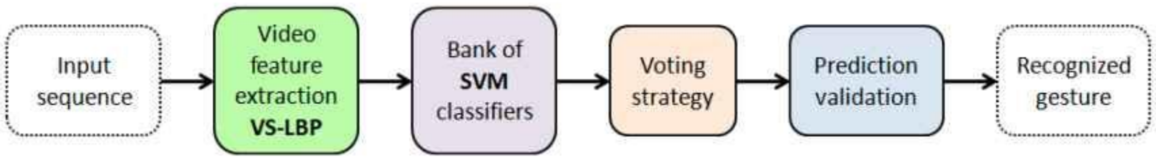


Fig 3.6: Block diagram of gesture recognition Module

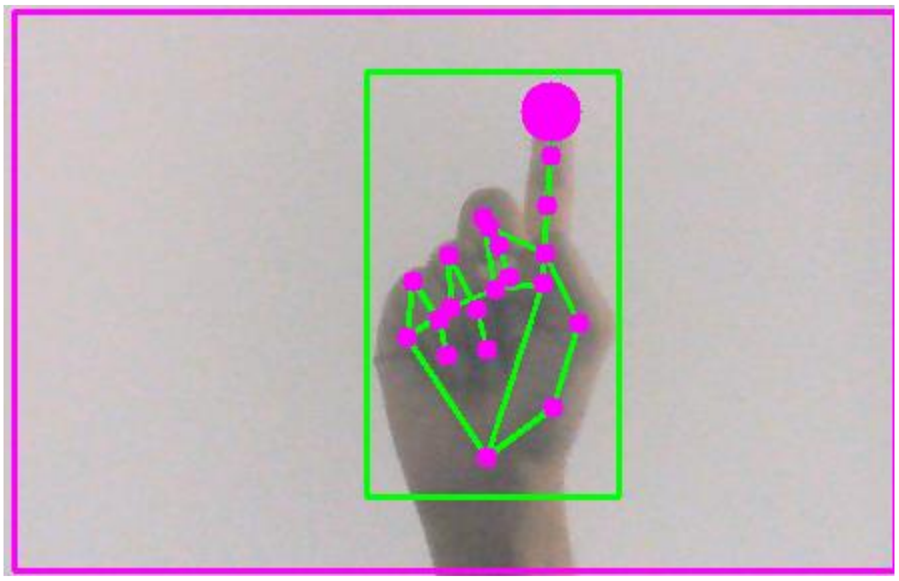
3.8 AI-based Screen Control Algorithms

This section describes the AI-based screen control algorithms implemented in the system. The algorithms utilize the hand gestures detected and recognized to perform various screen control actions. The key modes include:

3.8.1 Moving Mode

In moving mode, the system tracks the movement of the hand and maps it to cursor movements on the screen. The hand's coordinates are converted and used to control the position of the mouse pointer.

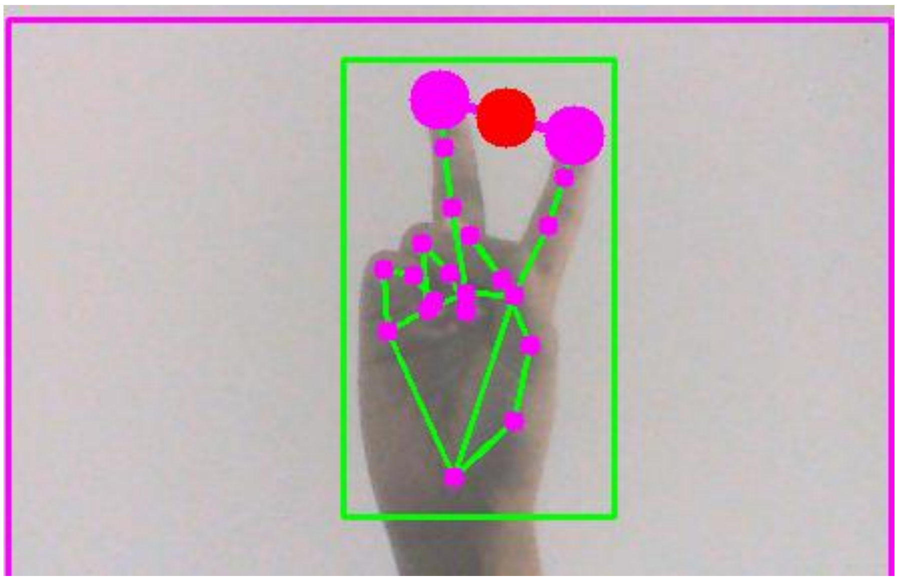
- When index finger is up and all other are down = Mouse cursor moving



3.8.2 Stop Mode

Stop mode enables the user to stop at specific position.

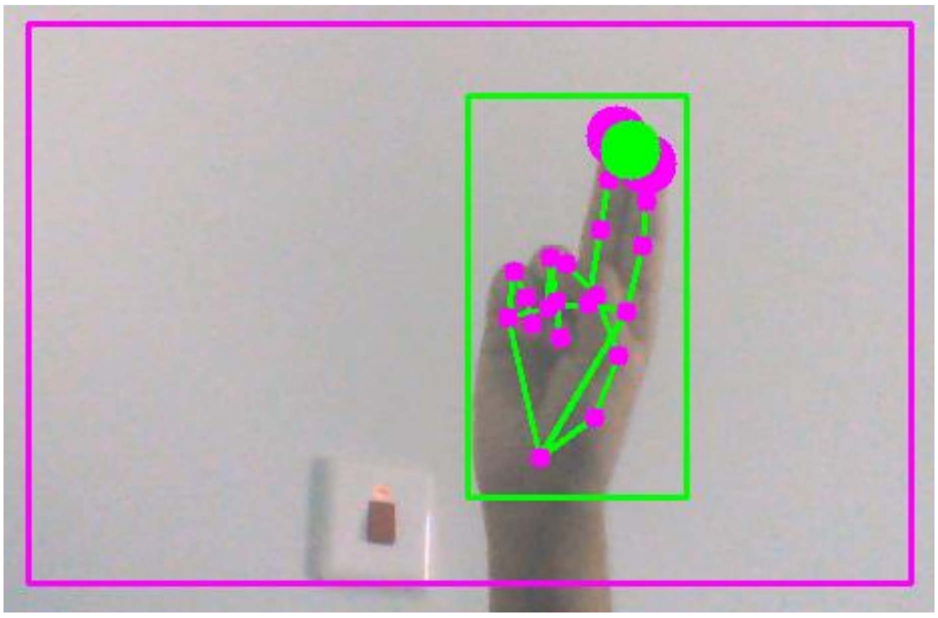
- When index and middle finger up = Cursor stop at that position



3.8.3 Clicking Mode

Clicking mode enables the user to perform a mouse click action using hand gestures. The system detects when the index and middle fingers are raised and determines the distance between them. If the distance is below a certain threshold, a mouse click action is triggered.

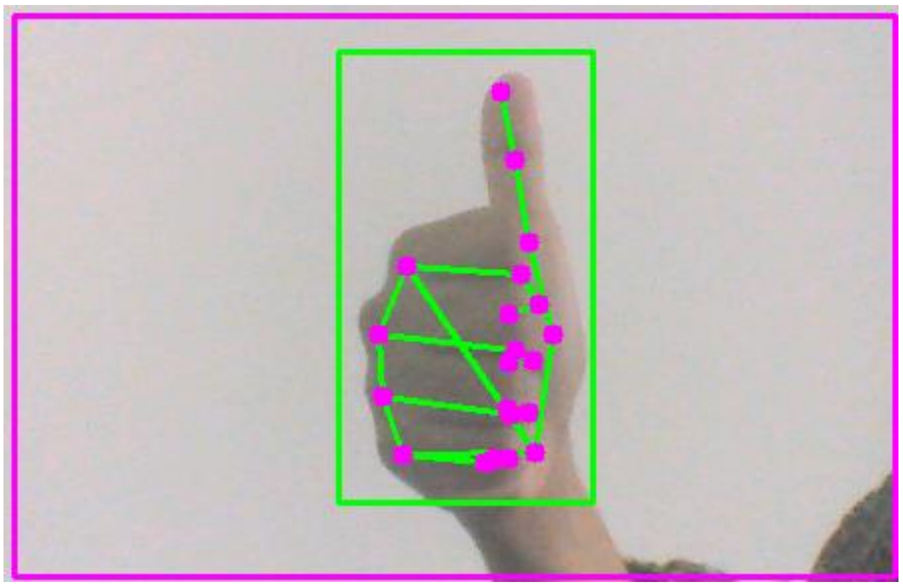
- When distance is smaller between index and middle finger = Left Click



3.8.3 Right-clicking Mode

Right-clicking mode allows the user to perform a right-click action using hand gestures. The system detects when all four fingers are down and the thumb is raised, triggering a right-click action.

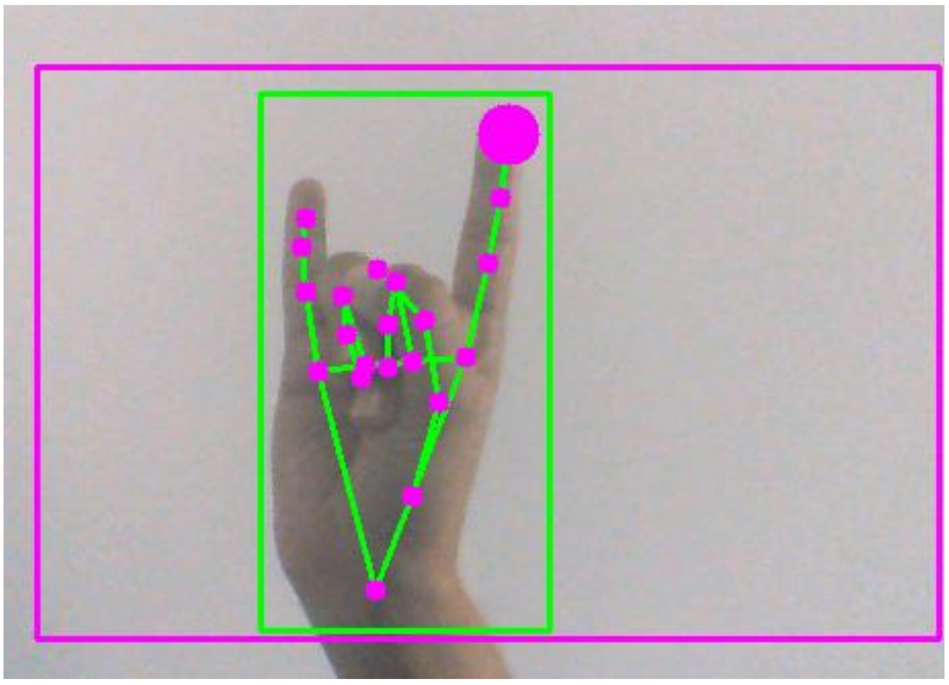
- Thumb is raised = Right Click

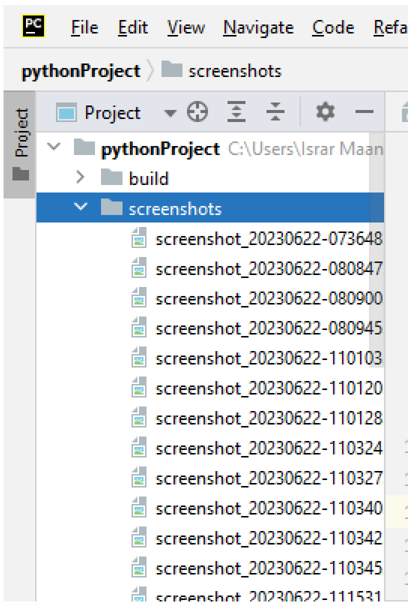


3.8.4 Screenshot Mode

In screenshot mode, the user can capture a screenshot of the screen using hand gestures. By raising the index and little fingers while keeping other fingers down, the system captures a screenshot and saves it with a timestamp.

- Little and index finger are up = Screenshot Captured





```
pythonProject 〉 screenshots
pythonProject C:\Users\Israr Maan
  build
  screenshots
    screenshot_20230622-073648
    screenshot_20230622-080847
    screenshot_20230622-080900
    screenshot_20230622-080945
    screenshot_20230622-110103
    screenshot_20230622-110120
    screenshot_20230622-110128
    screenshot_20230622-110324
    screenshot_20230622-110327
    screenshot_20230622-110340
    screenshot_20230622-110342
    screenshot_20230622-110345
    screenshot_20230622-111531
```

Fig 3.8.4 Saved screenshot

3.9 Summary

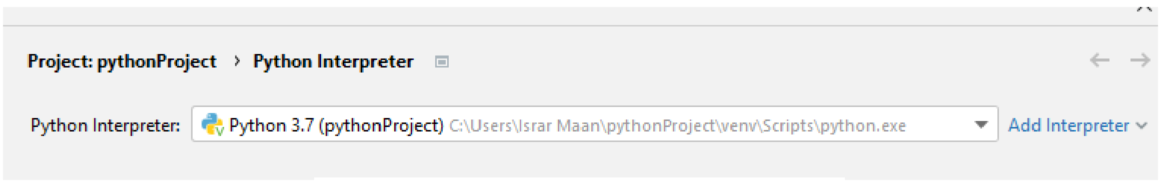
In this chapter, we presented the system architecture and methodology of the AI-based screen controller using hand gestures. The system comprises several components, including data collection and preprocessing, hand tracking module, gesture recognition module, and AI-based screen control algorithms. The UML class diagrams provided a visual representation of the internal structure and functionality of the hand tracking and gesture recognition modules. The AI-based screen control algorithms enable intuitive screen control by interpreting hand gestures and performing corresponding actions. The next chapter will focus on the implementation and evaluation of the system.

Chapter 4
System Development & Implementation

4.1 Tools and Language

- **Python 3.7 :**

Python programming language was the primary language used for developing the hand tracking mouse control system. Python offers simplicity, readability, and a vast array of libraries that make it suitable for computer vision and machine learning tasks.



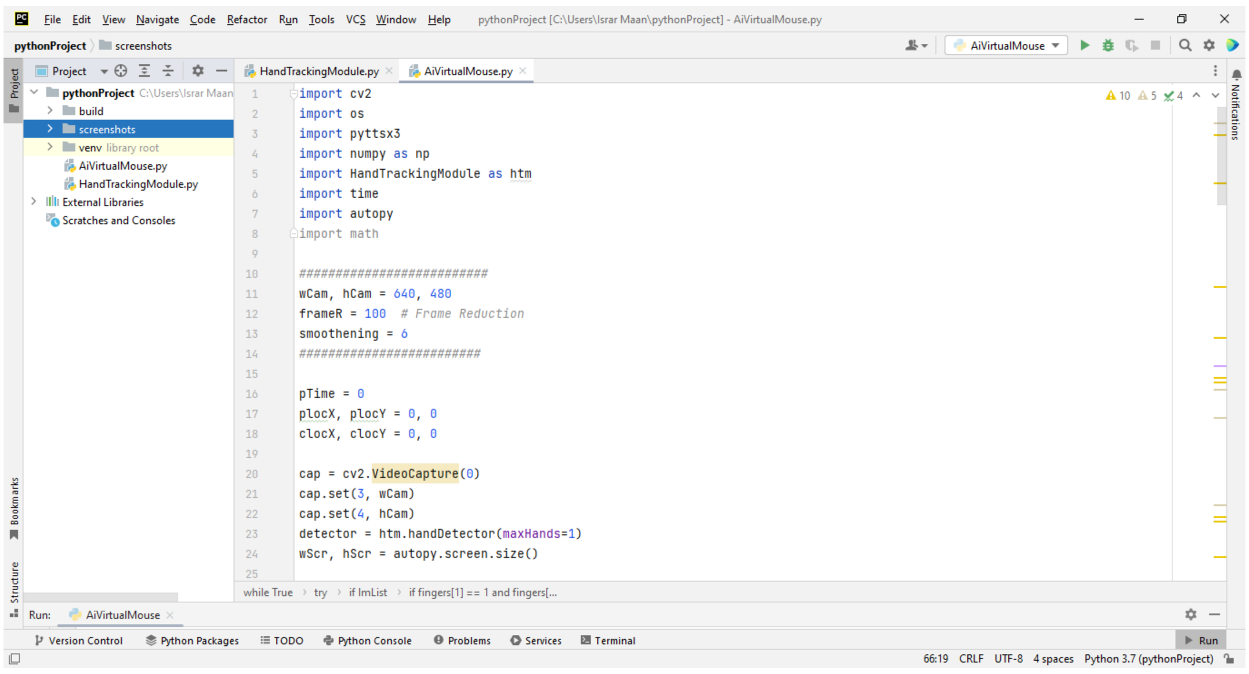
Project: pythonProject › Python Interpreter

Python Interpreter: Python 3.7 (pythonProject) C:\Users\Israr Maan\pythonProject\venv\Scripts\python.exe

Python 3.7 as an interpreter

- **PyCharm:**

PyCharm is an integrated development environment (IDE) specifically designed for Python development. It provides features like code editing, debugging, and version control, which streamline the development process.



File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject [C:\Users\Israr Maan\pythonProject] - AiVirtualMouse.py
pythonProject › screenshots
AiVirtualMouse
Project
pythonProject C:\Users\Israr Maan
build
screenshots
venv library root
AiVirtualMouse.py
HandTrackingModule.py
External Libraries
Scratches and Consoles
HandTrackingModule.py
AiVirtualMouse.py
Bookmarks
Structure
Notifications

```
import cv2
import os
import pyttsx3
import numpy as np
import HandTrackingModule as htm
import time
import autopy
import math

##########################
wCam, hCam = 640, 480
frameR = 100  # Frame Reduction
smoothening = 6
#########################

pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0

cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
detector = htm.handDetector(maxHands=1)
wScr, hScr = autopy.screen.size()

```

while True › try › if lmList › if fingers[1] == 1 and fingers[...
Run: AiVirtualMouse
Version Control
Python Packages
TODO
Python Console
Problems
Services
Terminal
Run
66:19 CRLF UTF-8 4 spaces Python 3.7 (pythonProject)

PyCharm environment

- **OpenCV:**

  OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision library used for image and video processing tasks. It provides various functions and algorithms for handling images, videos, and webcam inputs. In this project, OpenCV was used for video capture, image processing, and rendering.

- **Mediapipe:**

  Mediapipe is a framework developed by Google that provides a comprehensive set of tools and algorithms for building applications involving real-time perception. The Hand Tracking Module used in your project is part of the Mediapipe library. It offers pre-trained models for hand detection and tracking, making it easy to extract hand landmarks and gestures.

- **Autopy:**

  Autopy is a cross-platform library that allows interaction with the mouse, keyboard, and screen. It provides functionalities to control mouse movements, clicks, and keyboard inputs. In your project, Autopy was utilized for moving the mouse cursor and performing mouse clicks based on hand gestures.

- **Numpy:**

  Numpy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions. Numpy was used in your project for various mathematical calculations and array manipulations.

  g. pyttsx3: pyttsx3 is a Python library that provides a simple interface for text-to-speech synthesis. It enables the system to speak out messages or notifications. In your project, pyttsx3 was employed to generate audio notifications when capturing screenshots or performing specific hand gestures.

4.1.1 Libraries and their Purposes:

- OpenCV:

  Used for video capture, image processing, and rendering. It provides essential functions for handling image data, such as color space conversions, filtering, and drawing.

- **Mediapipe:**

  Integrated the Hand Tracking Module from Mediapipe, which offers pre-trained models for hand detection and tracking. It allows the extraction of hand landmarks and the ability to recognize hand gestures.

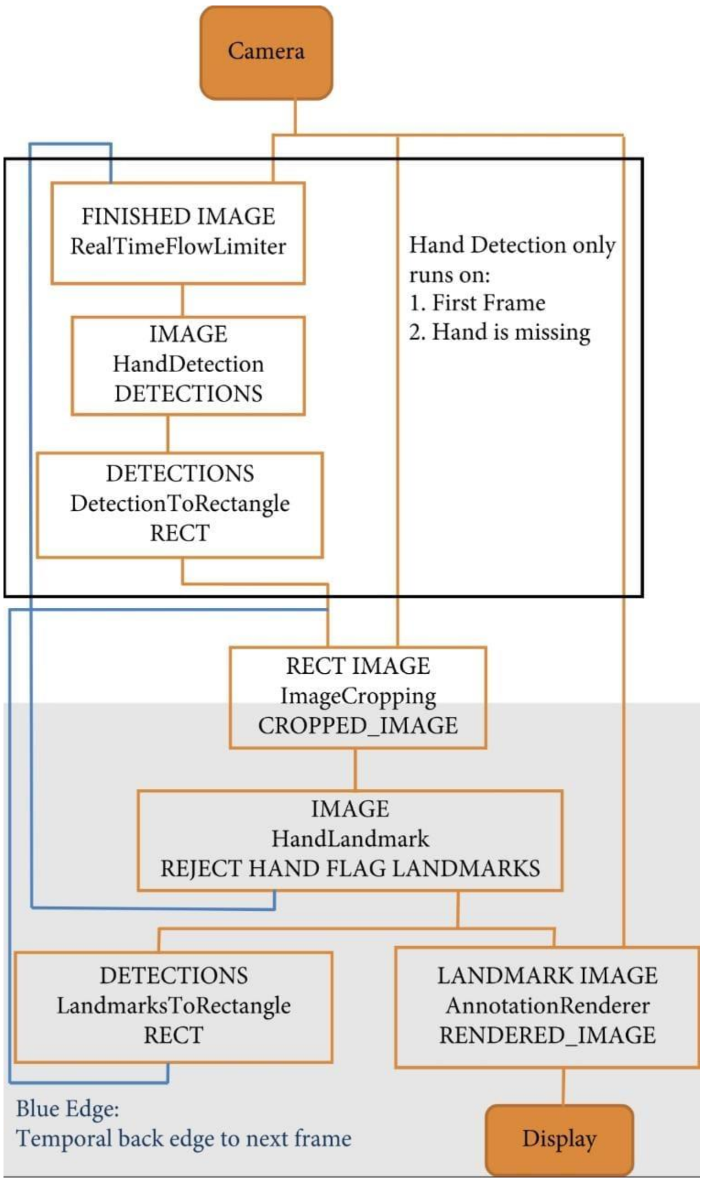


Fig: mediapipe hand recognition graph

- **Autopy:**

  Utilized Autopy library to control mouse movements and perform mouse clicks based on hand gestures. It provides an interface to interact with the mouse, keyboard, and screen.

- **Numpy:**

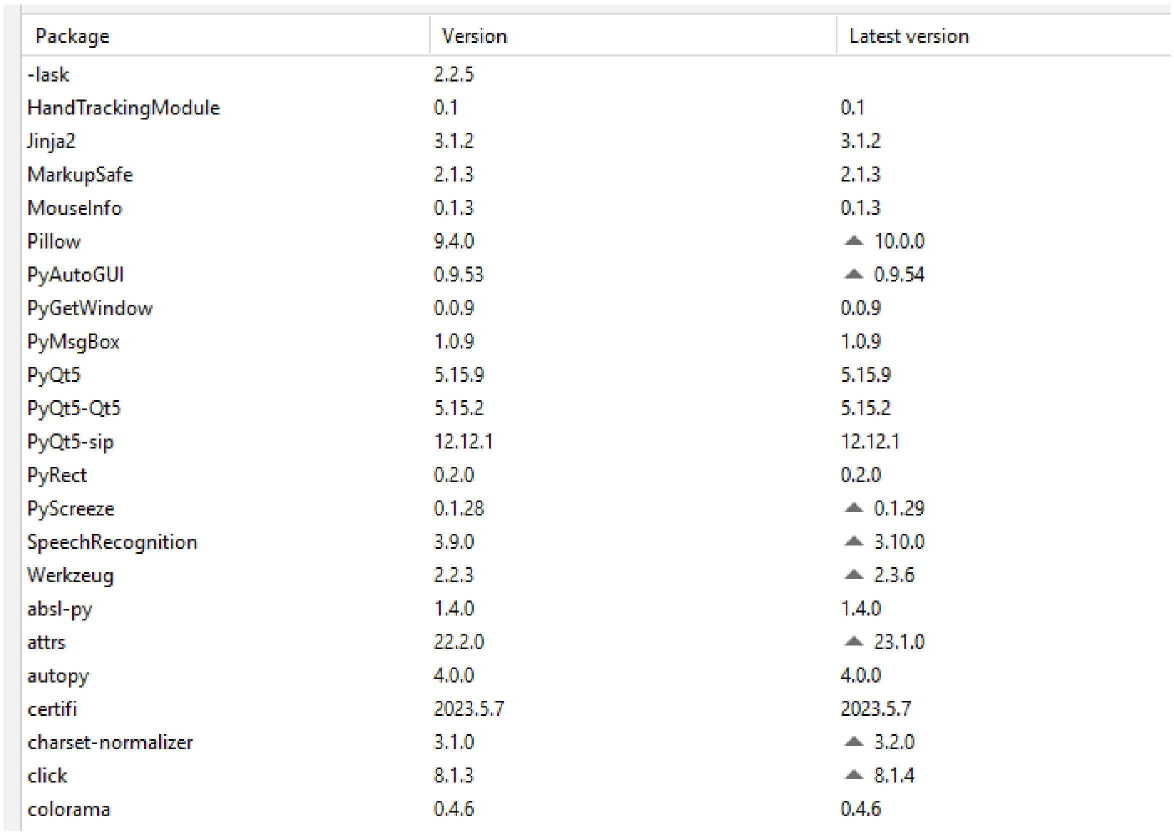
  Utilized Numpy for mathematical calculations, array manipulation, and handling multi-dimensional arrays. It simplifies complex mathematical operations in Python.

- **pyttsx3:**

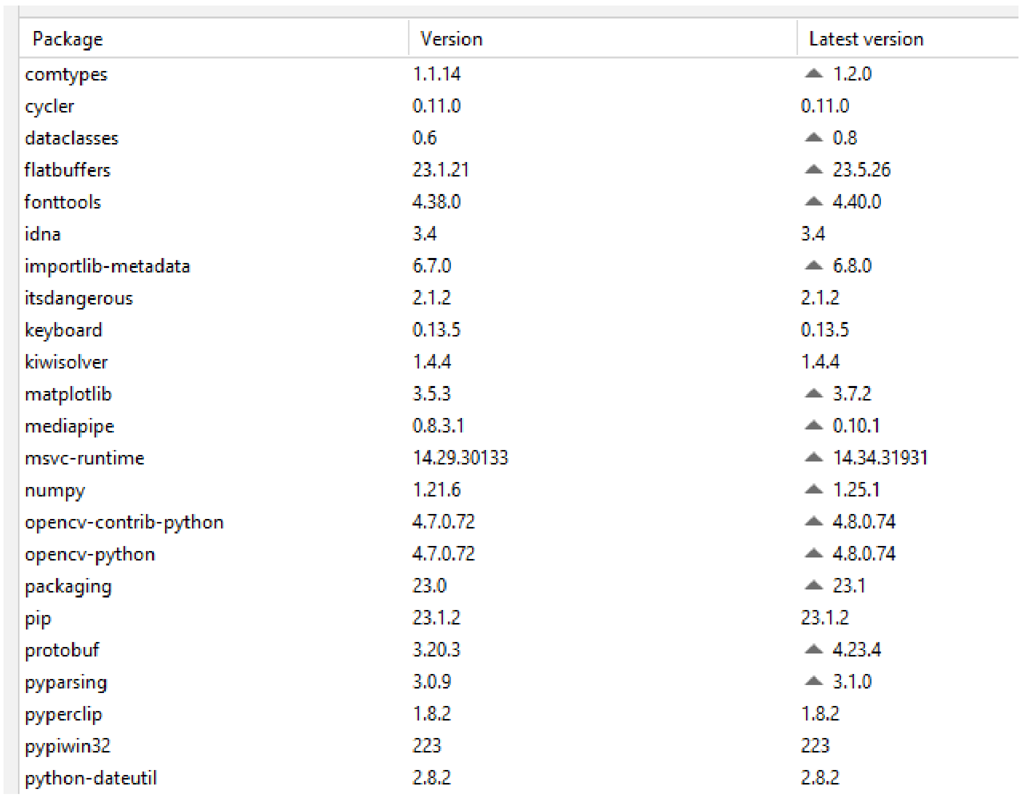
  Incorporated pyttsx3 library for text-to-speech synthesis, allowing the system to generate audio notifications or messages. It converts text-based messages into audible speech.

The combination of these tools, languages, and libraries enabled the development of an efficient hand tracking mouse control system. The integration of OpenCV and Mediapipe provided robust hand detection and tracking capabilities, while Autopy facilitated precise mouse control based on hand gestures. Numpy was instrumental in performing mathematical calculations, and pyttsx3 added audio notifications to enhance the user experience.

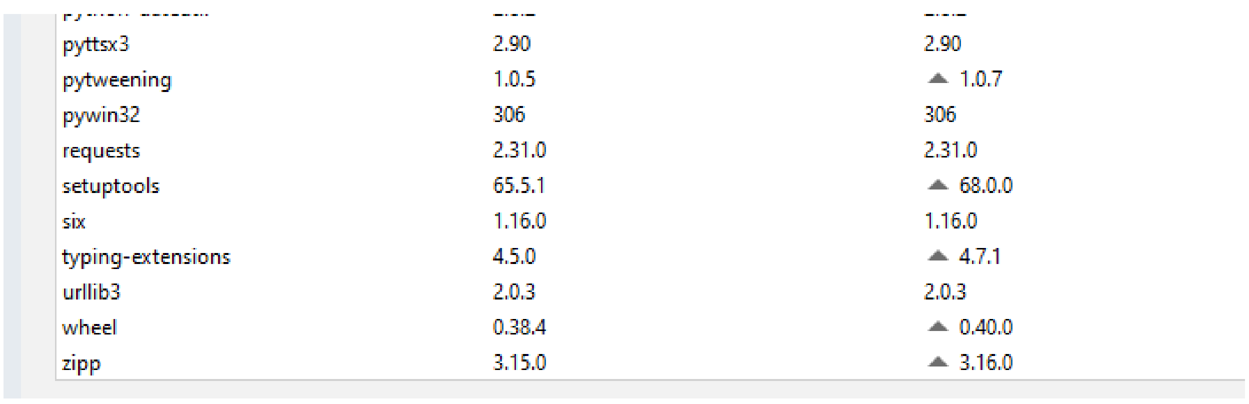
4.1.2 Python Packages table



Package	Version	Latest version
-lask	2.2.5	
HandTrackingModule	0.1	0.1
Jinja2	3.1.2	3.1.2
MarkupSafe	2.1.3	2.1.3
MouseInfo	0.1.3	0.1.3
Pillow	9.4.0	▲ 10.0.0
PyAutoGUI	0.9.53	▲ 0.9.54
PyGetWindow	0.0.9	0.0.9
PyMsgBox	1.0.9	1.0.9
PyQt5	5.15.9	5.15.9
PyQt5-Qt5	5.15.2	5.15.2
PyQt5-sip	12.12.1	12.12.1
PyRect	0.2.0	0.2.0
PyScreeze	0.1.28	▲ 0.1.29
SpeechRecognition	3.9.0	▲ 3.10.0
Werkzeug	2.2.3	▲ 2.3.6
absl-py	1.4.0	1.4.0
attrs	22.2.0	▲ 23.1.0
autopy	4.0.0	4.0.0
certifi	2023.5.7	2023.5.7
charset-normalizer	3.1.0	▲ 3.2.0
click	8.1.3	▲ 8.1.4
colorama	0.4.6	0.4.6



Package	Version	Latest version
comtypes	1.1.14	▲ 1.2.0
cycler	0.11.0	0.11.0
dataclasses	0.6	▲ 0.8
flatbuffers	23.1.21	▲ 23.5.26
fonttools	4.38.0	▲ 4.40.0
idna	3.4	3.4
importlib-metadata	6.7.0	▲ 6.8.0
itsdangerous	2.1.2	2.1.2
keyboard	0.13.5	0.13.5
kiwisolver	1.4.4	1.4.4
matplotlib	3.5.3	▲ 3.7.2
mediapipe	0.8.3.1	▲ 0.10.1
msvc-runtime	14.29.30133	▲ 14.34.31931
numpy	1.21.6	▲ 1.25.1
opencv-contrib-python	4.7.0.72	▲ 4.8.0.74
opencv-python	4.7.0.72	▲ 4.8.0.74
packaging	23.0	▲ 23.1
pip	23.1.2	23.1.2
protobuf	3.20.3	▲ 4.23.4
pyparsing	3.0.9	▲ 3.1.0
pyperclip	1.8.2	1.8.2
pypiwin32	223	223
python-dateutil	2.8.2	2.8.2



[illegible]	[illegible]	[illegible]
pyttsx3	2.90	2.90
pytweening	1.0.5	▲ 1.0.7
pywin32	306	306
requests	2.31.0	2.31.0
setuptools	65.5.1	▲ 68.0.0
six	1.16.0	1.16.0
typing-extensions	4.5.0	▲ 4.7.1
urllib3	2.0.3	2.0.3
wheel	0.38.4	▲ 0.40.0
zipp	3.15.0	▲ 3.16.0

All of these are the list of packages used in this project, here I also specify their latest versions and which version I used in this project.

4.2 Hand Tracking Module

4.2.1 Code Overview

The Hand Tracking module is a crucial component of the project that enables the detection, tracking, and analysis of hand movements. It consists of a class called **handDetector** that encapsulates the functionality related to hand detection and tracking. The code is implemented using the OpenCV and Mediapipe libraries.
The **handDetector** class is initialized with parameters such as mode, maxHands, detectionCon, and trackCon, which control the behavior of the hand tracking algorithm. The **mpHands** and **mpDraw** objects are used from the Mediapipe library for hand tracking and landmark visualization, respectively. Additionally, the **tipIds** list is defined to identify specific landmarks on the hand.

4.2.2 Hand Detection and Tracking

In the **findHands** method of the **handDetector** class, the input image is processed to detect and track hands using the **hands.process** function. The processed results are then used to draw landmarks on the image if the hands are detected. The **draw_landmarks** function from **mpDraw** is employed for this purpose. The method returns the image with the drawn landmarks.

4.2.3 Landmark Detection and Visualization

The **findPosition** method in the **handDetector** class is responsible for extracting the positions of landmarks on the hand. It takes the processed image as input and retrieves the landmarks' coordinates by iterating through the **landmark** attribute of the detected hand. The x and y coordinates are normalized and converted to pixel values based on the image dimensions. These coordinates are stored in the **lmList** list.
Additionally, the method calculates the bounding box (bbox) that encompasses the hand by finding the minimum and maximum x and y values from the landmark coordinates. The bounding box is visualized on the image using the **rectangle** function from OpenCV. The method returns the **lmList** and the bounding box coordinates.

4.2.4 Finger Tracking and Gesture Recognition

The **fingersUp** method in the **handDetector** class determines which fingers are extended based on the landmark positions. It compares the y-coordinate of specific landmarks (stored in the **tipIds** list) to detect whether the corresponding fingers are up or down. The method returns a list of binary values indicating the state of each finger.

4.2.5 Distance Calculation

The **findDistance** method in the **handDetector** class calculates the Euclidean distance between two specified landmarks on the hand. It takes the indices of the two landmarks, the image, and optional parameters for visualization (radius and line thickness). The method retrieves the coordinates of the landmarks from the **lmList** and calculates the distance using the **hypot** function from the math module. It also draws a line and circles representing the landmarks on the image if the **draw** parameter is set to true. The method

returns the distance, the modified image, and the coordinates of the line and circles.
The Hand Tracking module provides essential functionality for the project, including hand detection, landmark visualization, finger tracking, and distance calculation. These features lay the foundation for the subsequent AI Virtual Mouse module, which utilizes the hand tracking capabilities to enable mouse control through hand gestures.

4.3 Ai Virtual Mouse Coding

In the Ai Virtual Mouse section of the project, the code enables controlling the mouse cursor on the screen using hand gestures. It utilizes the Hand Tracking module to detect and track hand movements in real-time. The following parts describe the functionality and implementation of different features in the code.

4.3.1 Hand Tracking Initialization

The Hand Tracking module is initialized, setting the maximum number of hands to track, detection and tracking confidence levels, and the Hand Tracking Module object for hand detection and tracking.

- **wCam** and **hCam**: Width and height of the video frame from the camera.
- **frameR**: Frame reduction value for defining the region of interest (ROI).
- **smoothening**: Smoothening factor for stabilizing cursor movement.

4.3.2 Hand Tracking and Gesture Recognition Loop

The main loop continuously reads frames from the video capture, performs hand tracking, detects hand gestures, and interacts with the mouse accordingly.

- The loop runs indefinitely until interrupted.
- Within the loop, it reads frames from the video capture and passes them to the Hand Tracking module for detecting and tracking hands.
- The **findHands** function draws landmarks on the detected hands in the image.

4.3.3 Moving the Mouse Cursor

- This part checks if the index finger is up and the middle finger is down, indicating moving mode.
- It converts the hand coordinates to screen coordinates using interpolation.
- The cursor movement is smoothened to provide a smoother user experience.
- The **autopy.mouse.move** function moves the mouse cursor based on the calculated coordinates.

4.3.4 Clicking the Mouse

- This part checks if both the index and middle fingers are up, indicating clicking mode.
- It calculates the distance between the fingertips using the **findDistance** function from the Hand Tracking module.
- If the distance is short (less than 40 pixels), it simulates a mouse click using the **autopy.mouse.click** function.

4.3.5 Right-Clicking

- This part checks if all the fingers except the thumb are down and the thumb is up, indicating right-clicking mode.
- It simulates a right-click by pressing and releasing the right mouse button using the **autopy.mouse.toggle** function.

4.3.6 Taking Screenshots

- This part checks if the index and little fingers are up, indicating screenshot mode.
- It waits for half a second to avoid accidental triggering of the screenshot action.
- The code captures a screenshot using the **autopy.bitmap.capture_screen** function and saves it with a timestamp as the filename.
- Additionally, it utilizes the text-to-speech engine (**pyttsx3**) to speak a message confirming that the screenshot has been saved.

4.3.7 Hand Presence and Closed Fist Detection

- This part updates the **hand_closed** variable based on whether all fingers are down, indicating a closed fist.
- If the hand is not present, the **hand_closed** variable is reset to False.
- The image with hand landmarks and visualizations is displayed using **cv2.imshow**.
- The loop continues until the 'q' key is pressed.
- After the loop, the video capture is released and all OpenCV windows are closed.

.

Chapter 5
User Guide

5.1 Introduction

The user guide provides detailed instructions on how to use the Hand Tracking Module for various hand gestures and actions. This chapter will guide users through the setup, usage, and available features of the module.

5.2 System Requirements

Before using the Hand Tracking Module, ensure that your system meets the following requirements:

- ✓ **Operating System:** 64-bit operating system, x64-based processor
- ✓ **Python:** The module requires Python 3.7 or later.

5.3 Installation

To install the Hand Tracking Module, follow these steps:

- Open a terminal or command prompt.
- Create a new Python virtual environment (recommended).
- Activate the virtual environment.
- Install the necessary dependencies
- Download the Hand Tracking Module code
- Extract the downloaded file and navigate to the module directory.
- Run the command
- The module is now installed and ready to use.

5.4 Getting Started

To start using the Hand Tracking Module, follow these steps:

1. Ensure that your webcam is connected and functional.
2. Open a terminal or command prompt.
3. Activate the Python virtual environment (if applicable).
4. Navigate to the directory containing the Hand Tracking Module code.
5. Run the program
6. The module will start capturing video from the webcam and detecting hand gestures.

5.5 Hand Gestures and Actions

The Hand Tracking Module supports various hand gestures and actions. This section provides an overview of each gesture and explains how to perform the associated actions.

- **Moving the Mouse**

  Gesture: Pointing with a single finger (usually index finger).
  Action: Move the hand in the air to control the mouse pointer on the screen.
  Usage: Use this gesture to navigate the mouse pointer and interact with applications.

- **Clicking and Double-Clicking**

  Gesture: Show the index and middle fingers raised.
  Action: Tap the thumb and index/middle finger together to perform a left-click or double-click action.
  Usage: Use this gesture to select items, activate buttons, or open files/folders with a single or double-click.

- **Right-Clicking**

  Gesture: Close all fingers except the thumb.
  Action: Tap the thumb and fingers together to perform a right-click action.
  Usage: Use this gesture to access context menus, open additional options, or perform right-click actions.

- **Taking Screenshots**

  Gesture: Raise the index and little fingers while keeping the other fingers closed.
  Action: Hold the gesture for a short duration to capture a screenshot of the screen.
  Usage: Use this gesture to capture and save screenshots of the current screen or application.

- **Zoom In and Out**

  Gesture: Move the hand closer to or away from the webcam.
  Action: Move the hand closer to zoom in or move the hand away to zoom out.
  Usage: Use this gesture to zoom in or out on the screen, adjust the zoom level, or scale content.

5.6 Troubleshooting

If you encounter any issues or difficulties while using the Hand Tracking Module, try the following troubleshooting steps:

- Ensure that the webcam is properly connected and functional.
- Check the system requirements to ensure compatibility.
- Make sure the Hand Tracking Module is properly installed and up to date.
- Restart the module or the computer if necessary.
- Check for any error messages or warnings in the terminal or command prompt.
- Refer to the documentation or support resources for further assistance.

5.7 Conclusion

The Hand Tracking Module provides a powerful tool for tracking hand gestures and performing actions using the webcam. By following this user guide, users can easily set up and utilize the module's features for various applications. For any further questions or support, refer to the documentation or reach out to the developer community.

Chapter 6
Conclusion and Future Work

6.1 Limitations and Challenges:

While the proposed system offers a promising solution for hand gesture recognition and AI-based human-computer interaction, it is important to acknowledge the limitations and challenges that need to be addressed:

- **Occlusion issues:**

  One of the primary challenges faced by the system is handling occlusion. When the hand is partially or fully occluded, such as when fingers overlap or objects obstruct the hand, it can lead to inaccurate hand tracking and gesture recognition. To overcome this limitation, advanced techniques such as 3D hand tracking or multi-camera setups can be explored to improve the system's robustness in occlusion scenarios.

- **Lighting conditions**:

  Variations in lighting conditions can significantly impact the system's performance. Low light conditions can lead to difficulties in detecting hand landmarks accurately, while strong backlighting can result in overexposed images, making hand tracking challenging. To address this, adaptive algorithms that dynamically adjust the image processing parameters based on the lighting conditions can be implemented. Additionally, exploring techniques like infrared-based hand tracking can provide more reliable results in diverse lighting environments.

- **Complex gestures:**

  Recognizing complex gestures involving multiple fingers or intricate hand movements can be a challenging task. Certain gestures may have similar hand configurations, making it difficult to distinguish them accurately. To overcome this, incorporating advanced machine learning algorithms, such as deep learning models, can help capture more complex patterns and improve gesture recognition accuracy. Additionally, integrating temporal information by considering the sequence of hand poses over time can enhance the system's ability to recognize dynamic gestures.

6.2 Future Enhancements:

To further advance the proposed system and address its limitations, the following future enhancements can be considered:

- **Advanced machine learning algorithms:**

  Expanding the system's capabilities by exploring and integrating advanced machine learning techniques, such as deep learning models, can significantly improve the accuracy and robustness of gesture recognition. These models can learn complex patterns and representations from large-scale training data, enabling the system to recognize a wider range of gestures accurately.

- **Refinement of hand tracking module**:

  Continuously refining the hand tracking module is crucial for achieving more accurate and reliable hand tracking. Improvements can be made to handle occlusion and lighting challenges by exploring novel algorithms that leverage multiple sensors or depth information. Additionally, incorporating hand pose estimation techniques can provide more detailed information about hand articulation, enabling precise tracking even in challenging conditions.

- **User customization and adaptation**:

  Providing options for users to calibrate the system according to their hand shape and size can enhance the user experience and improve gesture recognition accuracy. By allowing users to input their hand profiles or leveraging user-specific training, the system can adapt to individual variations, ensuring personalized interaction.

- **Multi-user support:**

  Extending the system's capabilities to support multiple users simultaneously can enable collaborative interactions and expand its applicability in scenarios such as interactive presentations or group activities. Techniques like multi-camera setups or advanced depth sensing can be explored to differentiate between multiple users' hands and enable simultaneous tracking and interaction.

- **Integration of additional features:**

  To enhance the overall usability and flexibility of the system, integrating additional features can be considered. For example, incorporating voice commands alongside hand gestures can provide users with multiple interaction modalities.

Additionally, integrating hand gesture shortcuts for common tasks or applications can further streamline the user experience and improve efficiency.

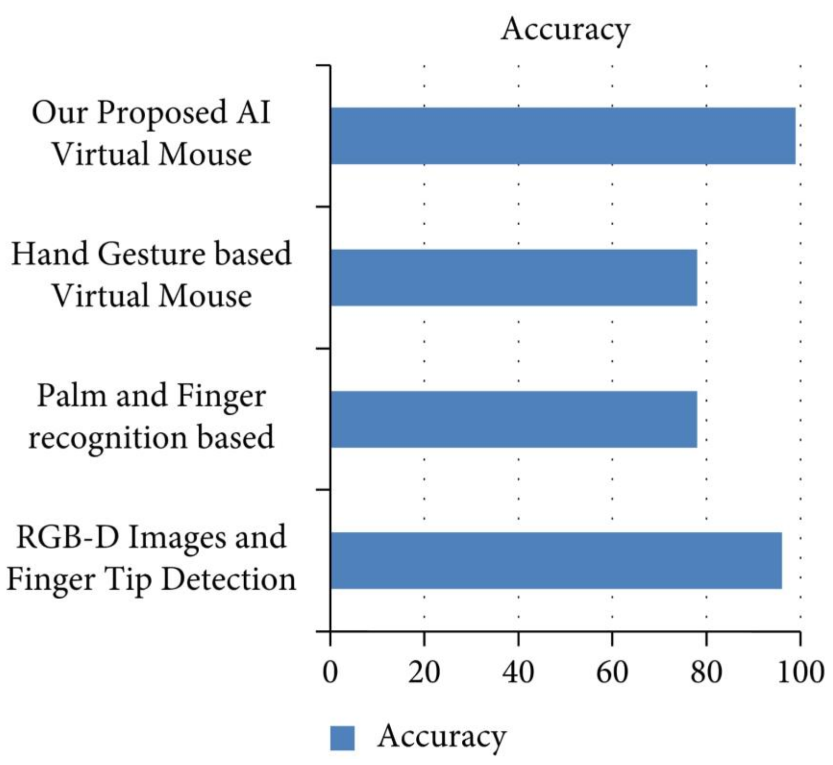
- **Cross-platform compatibility:**

  Expanding the system's compatibility to various platforms and operating systems can increase its accessibility and reach. Supporting different devices, such as smartphones, tablets, or wearable devices, can enable users to interact with a wide range of digital interfaces beyond traditional desktop environments. This cross-platform compatibility can be achieved through platform-specific adaptations and optimizations.

By addressing these limitations and exploring the suggested future enhancements, the proposed system can further improve its accuracy, usability, and adaptability, making it a more powerful tool for AI-based human-computer interaction.

6.3 Accuracy and Precision:

One of the critical factors in assessing the effectiveness of a virtual mouse is its accuracy and precision in tracking hand movements. The AI algorithms should be capable of accurately interpreting hand gestures and translating them into precise cursor movements on the screen. Evaluating the system's performance in different scenarios, such as varying lighting conditions or complex hand motions, can provide insights into its reliability.



6.4 Summary of Contributions:

In this research, we developed an AI-based virtual mouse system that utilizes machine learning algorithms to enable users to control their computer cursor using hand gestures. We designed and implemented a deep learning model capable of accurately recognizing and interpreting hand movements, allowing for intuitive and precise control. The system demonstrated promising results in terms of accuracy and usability, providing a viable alternative to traditional mouse input methods.

6.5 Comparison between Existing AI-Based Screen Controller and New AI-Based Screen Controller:

Existing AI-Based Screen Controller:

1. **Hardware Cost:** The existing AI-based screen controller relies on a hardware setup that incurs significant costs, including specialized sensors or cameras, processing units, and other peripherals. These hardware components contribute to the overall expense of implementing the system.

2. **Limited Features:** The existing controller offers a limited range of features compared to the new AI-based screen controller. It may provide basic functionalities such as hand detection and gesture recognition, but lacks advanced capabilities.

3. **Flaw:** One notable flaw in the existing controller is its termination behavior when a keyboard input is used. This means that if the user attempts to use the keyboard while the controller is active, it abruptly terminates the control process. This limitation restricts the seamless integration of keyboard and controller interactions.

New AI-Based Screen Controller:

1. **Cost-Effective:** In contrast to the existing solution, the new AI-based screen controller offers a cost-effective alternative. By leveraging existing computing devices such as webcams and standard processing units, it eliminates the need for additional expensive hardware components. This significantly reduces the implementation costs, making it more accessible and affordable.

2. **Enhanced Features:** The new controller introduces a range of enhanced features that surpass the capabilities of the existing solution. In addition to hand detection and gesture recognition, it incorporates additional functionalities such as screenshot capture and right-click capability. These features enhance the user experience and provide added convenience, facilitating seamless control and interaction with the screen.

3. **Improved Compatibility:** Unlike the existing controller, the new AI-based screen controller ensures improved compatibility with hardware components. It seamlessly integrates with standard keyboards and mice without terminating the control process. This enables users to freely use the keyboard while simultaneously benefiting from the controller's functionalities, resulting in a smoother and more efficient workflow.

By considering these detailed aspects, it becomes evident that the new AI-based screen controller surpasses the existing solution in terms of cost-effectiveness, feature set, and compatibility with hardware components. The new controller's ability to provide advanced features without the need for expensive hardware, while ensuring a seamless user experience with keyboard integration, makes it a highly desirable and superior choice.

6.6 Implications and Applications:

The AI-based virtual mouse opens up a wide range of implications and applications. It can be integrated into assistive technologies to empower individuals with disabilities, enabling them to navigate digital interfaces and perform tasks independently. Additionally, it has the potential to enhance virtual reality and augmented reality experiences, allowing for more natural and immersive interactions. Moreover, it can find applications in gaming, design, and other domains that require precise cursor control.

6.7 Final Thoughts:

The AI-based virtual mouse system presented in this research holds great promise for revolutionizing the way we interact with computers. It offers a novel and accessible input modality, opening up new possibilities for individuals with disabilities and providing an alternative option for all users. By further refining the system, exploring new applications, and addressing user feedback, we can continue to advance the field of human-computer interaction and create more inclusive and intuitive computing experiences for everyone.

Bibliography:

D.-H. Liou, D. Lee, and C.-C. Hsieh, "A real time hand gesture recognition system using motion history image," in Proceedings of the 2010 2nd International Conference on Signal Processing Systems, IEEE, Dalian, China, July 2010.

L. Thomas, "Virtual mouse using hand gesture," International Research Journal of Engineering and Technology (IRJET, vol. 5, no. 4, 2018.

S. U. Dudhane, "Cursor control system using hand gesture recognition," IJARCCE, vol. 2, no. 5, 2013.

D. L. Quam, "Gesture recognition with a DataGlove," IEEE Conference on Aerospace and Electronics, vol. 2, pp. 755–760, 1990.

Katona, "A review of human–computer interaction and virtual reality research fields in cognitive InfoCommunications," Applied Sciences, vol. 11, no. 6, p. 2646, 2021.