VIRTUAL MOUSE

A MINI PROJECT REPORT

Submitted by

VIJAYSARATHI N [710120205052]
SHRI BALAJI G [710120205040]
BASKAR S [710120205008]

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

ADITHYA INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI- 600025

MAY 2023

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

This mini project report "VIRTUAL MOUSE" is the Bonafide work of "VIJAYSARATHI N (710120205052), SHRI BALAJI G (710120205040), BASKAR S (710120205008)" who carried out the project work under my supervision

.

SIGNATURE
Dr.MISHMALA SUSHITH,ME,PhD.,
HEAD OF THE DEPARTMENT
DEPARTMENT OF INFORMATION TECHNOLOGY,
ADITHYA INSTITUTE OF TECHNOLOGY,
KURUMBAPALAYAM,
COIMBATORE-641107.

SIGNATURE
Ms.A.RAMYA, BE,ME.,
SUPERVISOR
ASSISTANT PROFESSOR,
DEPARTMENT OF INFORMATION TECHNOLOGY,
ADITHYA INSTITUTE OF TECHNOLOGY,
KURUMBAPALAYAM,
COIMBATORE-641107.

Submitted for the Anna University Viva-Voce examination held on ————————

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We are pleased to present the "**VIRTUAL MOUSE**" project and take the opportunity to express our profound gratitude to all of those people who helped us in the completion of this mini project.

We would like to express our deep sense of gratitude to the management of Adithya Institute of Technology, **Er. C. SUKUMARAN, Chairman and Mr. S. PRAVEEN KUMAR, Trustee,** for providing us with all facilities to carry out this project.

We would like to express our sincere thanks to our Trustee, **Dr.SRINITHI PRAVEEN KUMAR,** Adithya Institute of Technology, for providing us all facilities to carry out this project successfully.

We would like to express our gratitude to our Principal, **Dr.D. SOMASUNDARESWARI,** Adithya Institute of Technology, for providing us all facilities to carry out this project successfully.

We would like to express our profound thanks to our **Dr.MISHMALA SUSHITH** Head of the Department, Department of Information Technology**,** Adithya Institute of Technology, for her valuable suggestions and guidance throughout the course of the project.

We also express our sincere thanks to our guide **Ms.A.RAMYA, BE, M.E.,** Assistant Professor, Department of Information Technology. We also thank all the Faculty Members of our department for their help in making this project a successful one.

Finally, we take this opportunity to extend our deep appreciation to our family and friends, for all they meant to us during the crucial times of the completion of our project.

ABSTRACT

The main aim of developing this system is to provide cursor access to the computer without any special devices such as mouse or trackpad. In this project, an approach for Human computer Interaction (HCI), where tried to control the mouse cursor, movement and click events of the mouse using hand gestures. Hand gestures were acquired using a camera based on color detection technique. This method mainly focuses on the use of a Web Camera to develop a virtual human computer interaction device in a cost effective manner. As computer technology is growing up, the importance of human computer interaction is rapidly increasing. Most devices use Touch screen technology which cannot be affordable to all the applications. A virtual human based Computer interactive module such as virtual mouse, can be an alternative way for the traditional mouse or touch screen.

In conclusion, the virtual mouse represents a novel approach to human-computer interaction, providing users with an alternative input method for controlling the computer cursor. With its diverse range of input methods, potential applications, and accessibility benefits, virtual mice have the potential to revolutionize the way we interact with computers, making computing more intuitive, flexible, and inclusive.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

The virtual mouse is a technology that allows users to control their computer cursor without the need for a physical mouse. It provides an alternative input method that utilizes various techniques to track user movements and translate them into cursor actions on the computer screen. By eliminating the need for a physical device, virtual mice offer increased portability, flexibility, and accessibility.

The underlying principles of virtual mouse technology involve capturing and interpreting user input, tracking hand movements or other input signals, and transforming them into cursor actions. This typically involves the use of sensors, cameras, or touch-sensitive surfaces to detect and analyze user movements. Sophisticated algorithms and techniques are employed to accurately interpret and map these movements to cursor actions, ensuring precise control.

1.2 PROBLEM STATEMENT

The virtual mouse is to provide an alternative input method for computer users who are unable to use a physical mouse due to various disabilities or physical limitations. The virtual mouse aims to improve the accessibility and usability of computers for people with disabilities, enabling them to interact with technology more effectively and efficiently. It also aims to provide a customizable and versatile input method that can be adapted to meet the specific needs

CHAPTER 2
LITERATURE SURVEY

2.1 OBJECTIVE

Virtual mouse is to provide a user interface input device that simulates the functionality of a physical mouse. It allows users to control the cursor on a computer screen by using movements and gestures, typically through a touchpad, touchscreen, or other sensor-based technologies.

The virtual mouse serves several purposes, including:

- ➢ Cursor control: The primary objective is to enable users to move the cursor on the screen, perform point-and-click actions, and interact with graphical user interfaces (GUIs) or applications.
- ➢ Navigation: The virtual mouse allows users to navigate through menus, options, and various elements within an operating system, software, or website.
- ➢ Gestural input: Many virtual mouse implementations support gestures, enabling users to perform actions such as scrolling, zooming, swiping, and rotating by using specific finger movements or multi-touch gestures.
- ➢ Accessibility: Virtual mice can be an essential accessibility tool for individuals who have difficulty using physical mice or other traditional input devices. They provide an alternative method of interaction that accommodates different needs and abilities.
- ➢ Mobile devices: On mobile devices, where physical mice are not typically available, virtual mice enable precise cursor control and mimic the functionality of a traditional mouse. This is particularly useful for tasks like text selection, editing, and interacting with applications.
- ➢ Remote access: In remote desktop or virtual machine environments, a virtual mouse allows users to control a computer or server from a different location by emulating mouse movements and clicks over the network.

Overall, the objective of a virtual mouse is to provide a flexible and intuitive input mechanism that replicates the functionality of a physical mouse, enhancing user interaction and facilitating various tasks on computers, mobile devices, and other systems

2.2 LITERATURE REVIEW

- By using an IR camera and an IR pen that serves as a virtual marker, Ashish Mehtar and Ramanath Nayak created the Virtual Mouse on March 12, 2015. It also makes use of hardware. The existing Virtual Marker is altered to serve as both a mouse pointer and a marker, giving it access to all of the mouse's features. Disadvantage
  IR PEN and IR Camera required
  Expensive

- By Sherin Mohammed and VH Preetha in 2018, their Hand Gesture-Virtual Mouse for Human Computer Interaction utilized OpenCV and Python and required the deployment of two expensive cameras. Their study focuses on the development of hand gestures in 3-D space for human computer interface systems using two cameras in position.

  Disadvantage

  Requires heavy GPU processor (Due to 3D modeling)

  Requires 2 Cameras

- By using computer vision to operate the mouse cursor, the Virtual Mouse Using Object Tracking research by Monali Shetty, Christina Daniel, Manthan Bhatkar, and Ofrin Lopez in 2020 captures hand gestures from a camera using an HSV color recognition method.
  Disadvantage
  Hue color changes according to lighting.
- They suggest employing hand motion recognition and fingertip identification to control a virtual mouse in their article. Two techniques are used in their investigation to track the fingers: hand gesture detection and the use of coloured caps.

  Disadvantage

  Require special coloured caps.

CHAPTER 3

EXISTING SYSTEM

3.1 EXISTING SYSTEM

- The existing system may require additional cameras and It may require IR camera and IR sensors
- The remote accessing of the monitor screen using the hand gesture is unavailable.

  Even-though it is largely trying to implement the scope is simply restricted in the field of virtual mouse.

3.2 DRAWBACKS OF EXISTING SYSTEM

- The major problem in Virtual mouse software requires additional devices.
- Special devices may cost higher than any other physical mouse..
- Tracking the cursor was complicated when the user's movements are fast.
- There was a shortage and sometimes unavailability of modules for respective OS platforms (i.e) Linux and MAC.

3.3 PROPOSED SYSTEM

In this project, To make use of the laptop camera or web-cam and by recognizing the hand gesture we could control the mouse and perform basic operations like mouse pointer controlling, select and deselect using left click, and a quick access. The project done is a “Zero Cost” hand recognition system for laptops, which uses stable Opencv algorithms to determine the hand, hand movements and by assigning an action for each movement.

3.4 ADVANTAGES OF PROPOSED SYSTEM

- Portability: One significant advantage of virtual mice is their portability. Users can control the computer cursor on any compatible device without the need to carry a physical mouse. This is particularly useful for individuals who frequently switch between multiple devices or work remotely. Whether using a laptop, tablet, or a touch-enabled screen, virtual mice provide a consistent and convenient input method.
- Accessibility: Virtual mice offer alternative input mechanisms that cater to individuals with physical disabilities. They provide customizable options that can accommodate various impairments, such as limited dexterity or mobility. Users can choose input methods that best suit their abilities, including hand gestures, touch gestures, or eye-tracking, making computing more accessible and inclusive.
- Flexibility: Virtual mouse technology provides flexibility in terms of input methods. Users can choose from different options based on their preferences and available technology. Whether it's using hand gestures, touch-sensitive surfaces, or eye movements, virtual mice offer a range of input methods that can be tailored to individual needs and preferences.
- Immersive Experiences: Virtual mice find applications in virtual reality (VR) and augmented reality (AR) environments. In these immersive settings, traditional physical mice may not be practical or provide the desired level of interaction. Virtual mice enable users to navigate and interact with virtual objects and environments more intuitively, enhancing the overall immersive experience.
- Reduced Hardware Dependencies: With virtual mice, there is no need for additional physical hardware. Users can control the computer cursor using existing sensors, cameras, or touch-sensitive surfaces. This eliminates the requirement for a dedicated mouse device and reduces the clutter of additional peripherals. It also simplifies setup and reduces maintenance requirements.
- Gesture-Based Control: Virtual mice often utilize gesture-based control methods, such

as hand movements or touch gestures. This can offer a more natural and intuitive way of interacting with the computer. Users can perform gestures that mimic real-world actions, enhancing the user experience and potentially reducing the learning curve associated with traditional mouse input.

- Improved Precision and Control: Virtual mice leverage advanced algorithms and tracking techniques to accurately interpret user input and translate it into precise cursor actions. With proper calibration and optimization, virtual mice can offer high levels of control, allowing users to perform fine-grained movements and tasks with accuracy.
- Integration with Existing Systems: Virtual mouse technology can integrate with existing software and operating systems, allowing users to seamlessly incorporate it into their computing environment. This compatibility ensures that users can enjoy the benefits of virtual mice without significant disruptions or the need for major software or hardware upgrades.

\

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

- Front-End : Python
- Back-End : Python
- Operating System : Windows 7,Windows 8,Windows 10, Linux
- System Type : 32-bit or 64-bit Operating System

4.2 SOFTWARE DESCRIPTION

4.2.1 Python Introduction

Python can be used in the development of virtual mouse applications. Python is a versatile programming language known for its simplicity, readability, and extensive library support. It provides various libraries and frameworks that can aid in creating virtual mouse functionality. Here are a few ways Python can be used in virtual mouse development:

1. Image Processing: Python offers powerful image processing libraries such as OpenCV and scikit-image. These libraries can be used to analyze input from cameras or sensors to detect hand gestures or other input signals. They provide functions for image manipulation, object detection, and tracking, which are essential for capturing and interpreting user movements.
2. Gesture Recognition: Python can be utilized to implement gesture recognition algorithms. Using machine learning techniques, Python libraries such as scikit-learn or TensorFlow can train models to recognize specific hand gestures or touch gestures. These models can then be integrated into the virtual mouse application to map recognized gestures to cursor actions.
3. User Interface: Python's GUI libraries like Tkinter or PyQt can be used to create the graphical user interface (GUI) of the virtual mouse application. The GUI can include buttons, sliders, or touch-sensitive areas that users can interact with to control the cursor. These libraries provide tools for handling user input events and updating the cursor position based on user interactions.
4. Event Handling: Python can handle events such as mouse clicks or touch events and translate them into cursor actions. Libraries like Pygame or Pyglet offer functionality to capture and process input events from various input sources. By mapping these events to cursor movements or clicks, Python enables the virtual mouse application to respond to user interactions.
5. Integration with Operating System: Python provides modules for interacting with the operating system's input/output functionalities. Using libraries like pyautogui or pynput.

6. System Integration: Python's versatility enables it to communicate with other system components or peripherals. For example, Python can interact with eye-tracking devices, touch-sensitive surfaces, or other specialized hardware to capture input for virtual mouse control. Libraries like PySerial can facilitate communication with external devices.

Python's ease of use, extensive libraries, and community support make it a popular choice for virtual mouse development. Its versatility allows developers to implement various aspects of virtual mouse functionality, including image processing, gesture recognition, user interface, event handling, system integration, and more.

4.2.2 Common Uses of Python

Python is a versatile programming language widely used in various domains and applications. Some common uses of Python include:

1. Web Development: Python is frequently used for web development, thanks to frameworks such as Django and Flask. These frameworks provide tools and libraries for building scalable, secure, and dynamic web applications.
2. Data Science and Machine Learning: Python is immensely popular in the field of data science and machine learning. Libraries such as NumPy, Pandas, and SciPy offer powerful tools for data manipulation, analysis, and scientific computing. Additionally, frameworks like TensorFlow and PyTorch enable developers to build and deploy machine learning models efficiently.
3. Scripting and Automation: Python's simplicity and readability make it an ideal choice for scripting and automation tasks. It allows developers to write scripts to automate repetitive tasks, system administration, file manipulation, or data processing.
4. Scientific Computing: Python's extensive library ecosystem, including NumPy, SciPy, and Matplotlib, makes it a valuable tool for scientific computing. Researchers and scientists use Python for simulations, data visualization, statistical analysis, and computational modeling.
5. Natural Language Processing (NLP): Python offers libraries like NLTK (Natural Language Toolkit) and spaCy that facilitate NLP tasks such as text processing, sentiment analysis, language translation, and information extraction.
6. Web Scraping: Python's libraries, such as BeautifulSoup and Scrapy, enable web scraping, which involves extracting data from websites. This capability is valuable for tasks like data collection, market research, and building data-driven applications.
7. Internet of Things (IoT): Python is frequently used in IoT projects due to its simplicity and availability of libraries like MQTT and Requests. It enables developers to build IoT applications, interact with sensors, and communicate with IoT devices.

8. Desktop GUI Applications: Python's GUI libraries, including Tkinter, PyQt, and wxPython, allow developers to create cross-platform desktop applications with graphical user interfaces (GUIs). These applications can range from simple utilities to complex software.

9. Game Development: Python is utilized in game development, both for building complete games using frameworks like Pygame and for scripting game logic in game engines such as Unity or Unreal Engine.

10. Education: Python's simplicity and readability make it a popular choice for teaching programming concepts and introductory computer science courses. Its syntax and extensive documentation make it accessible for beginners.

These are just a few examples of Python's common uses. Python's versatility, robustness, and extensive library ecosystem make it suitable for a wide range of applications and industries.

4.3. DATABASE

- **Database** also called **electronic database**, any collection of data or information which is specially organized for rapid search and retrieval by a computer. Databases are structured to facilitate the storage, retrieval, modification, and deletion of data in conjunction with various data-processing operations.

- A database is stored as a file or a set of files. The information in these files may be broken down into records, each of which consists of one or more fields. Fields are the basic units of data storage, and each field typically contains information pertaining to one aspect or attribute of the entity described by the database. Records are also organized into tables that include information about relationships between its various fields.

- Although database is applied loosely to any collection of information in computer files, a database in the strict sense provides cross-referencing capabilities. Using keywords and various sorting commands, users can rapidly search, rearrange, group, and select the fields in many records to retrieve or create reports on particular aggregates of data. Database records and files must be organized to allow retrieval of the information. The user provides a string of characters, and the computer searches the database for a corresponding sequence and provides the source materials in which those characters appear.

- The many users of a large database must be able to manipulate the information within it quickly at any given time. Moreover, large businesses and other organizations tend to build up many independent files containing related and even overlapping data, and their data-processing activities often require the linking of data from several files.

- The information in many databases consists of natural-language texts of documents; number-oriented databases primarily contain information such as statistics, tables, financial data, and raw scientific and technical data. Small databases can be maintained on personal-computer systems and used by individuals at home.

CHAPTER 5
SYSTEM DESIGN

5.1 CONCEPT DIAGRAM

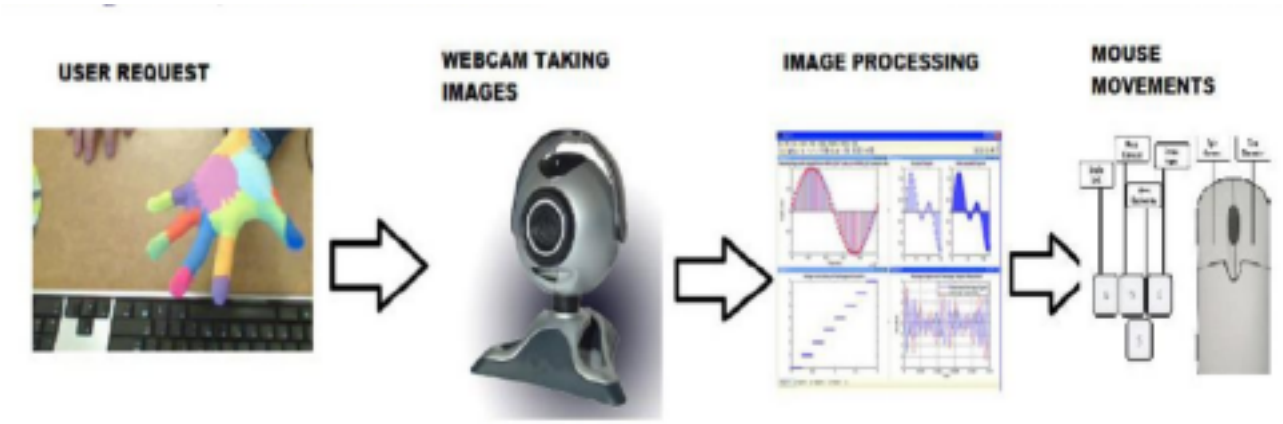


Fig 5.1.1 Concept diagram of proposed diagram

The concept of a virtual mouse is particularly useful in situations where a physical mouse is not available or practical, such as on touchscreen devices or when using virtual reality (VR) or augmented reality (AR) systems. It provides an alternative means of interacting with the user interface, offering similar cursor control and functionality as a physical mouse. The system manages all the activities from Users interaction both from camera and mouse till the process is closed.

5.2 DATA FLOW DIAGRAM

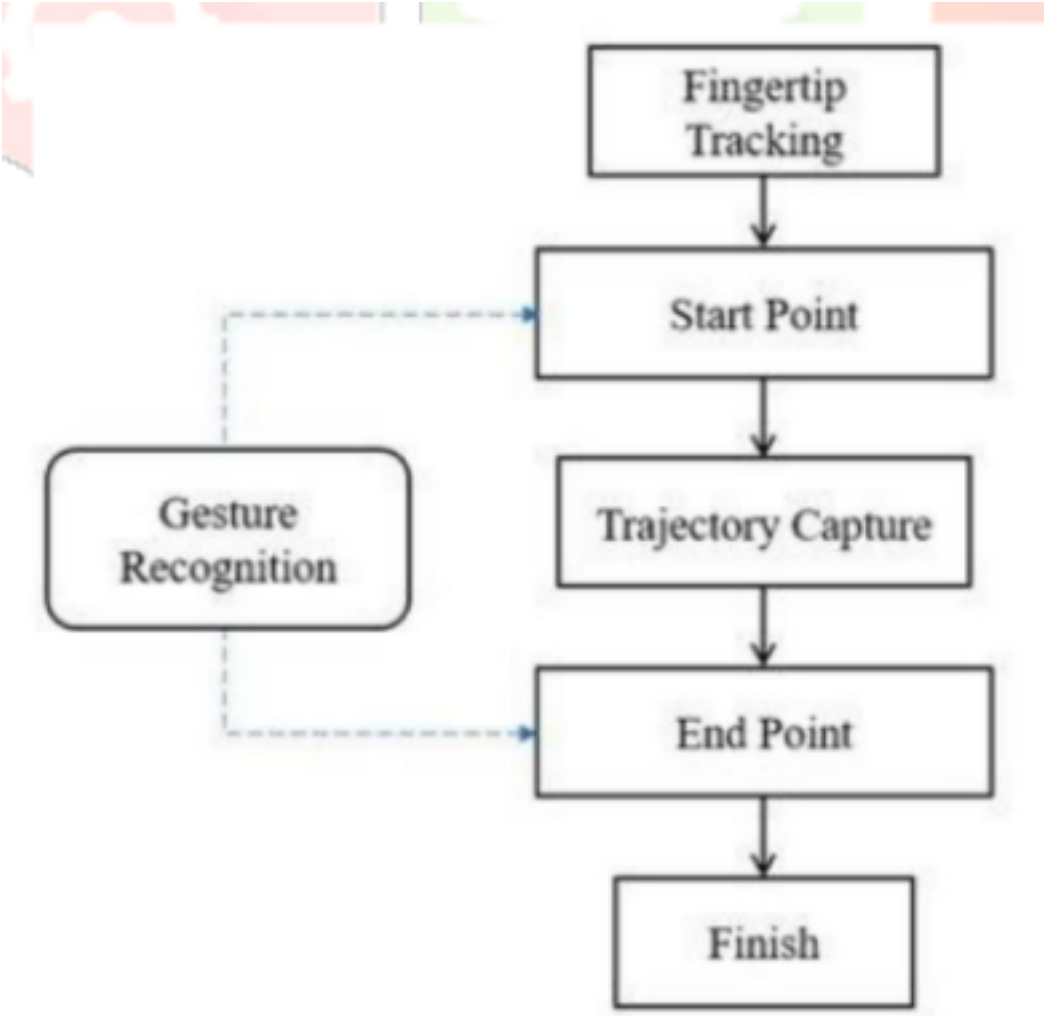


Figure : 5.2 Data flow and its process

5.3 System Architecture

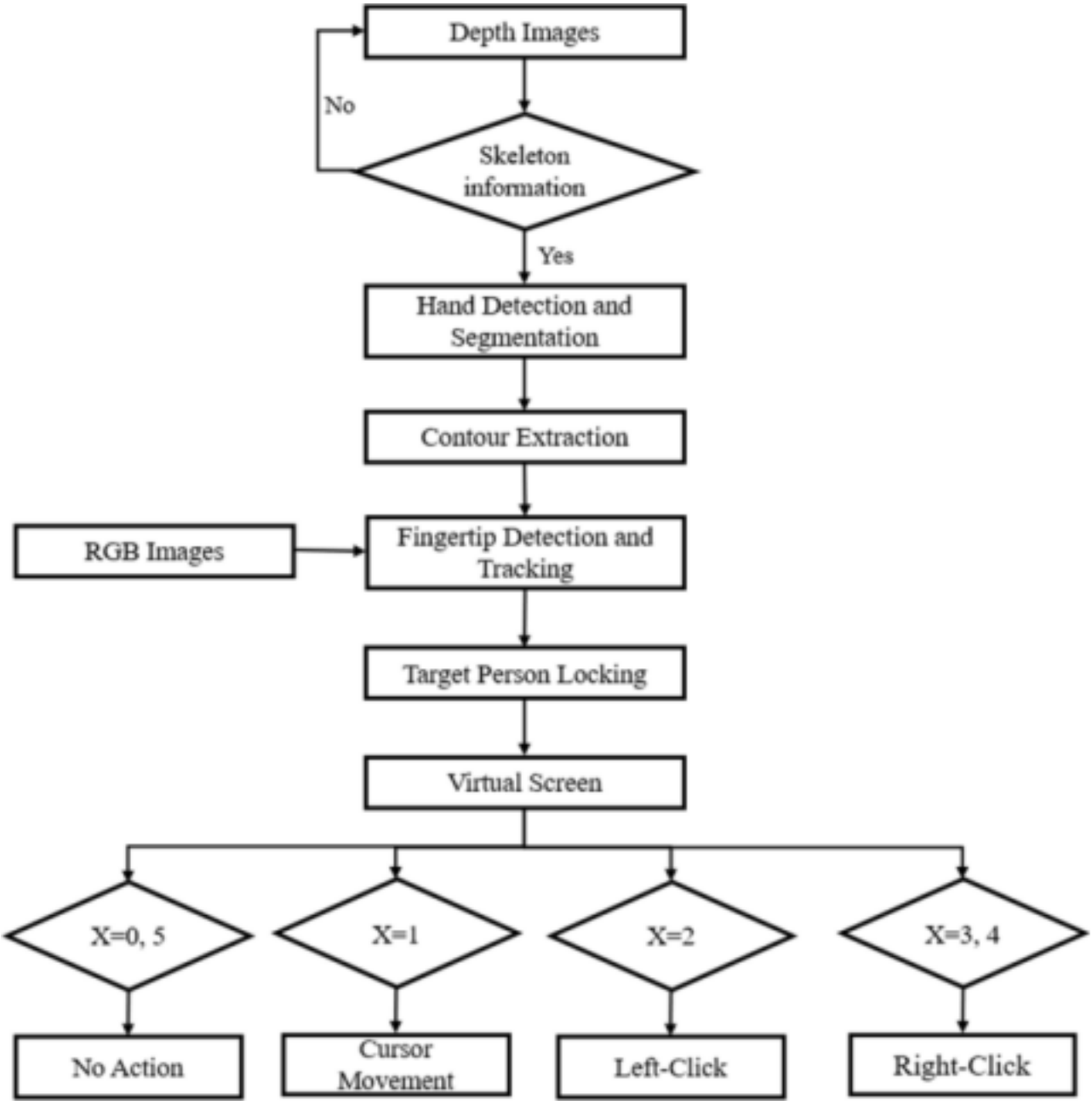


Fig 5.3. Flow diagram for Implementation

CHAPTER 6
MODULES

6.1 MODULE DESIGN

- Input Module

  (a) Opencv

- Process Module

  (a) Mediapipe

  (b) Numpy

  (c) Math

  (d) Time

- Output Module

  (a) autopy

6.2 MODULE DESCRIPTION

The Python Package Index (PyPI) hosts thousands of additional modules created by the Python community, covering areas such as web development, machine learning, natural language processing, image processing, and more.

6.2.1 Input Module

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning software library. It provides a wide range of functions and algorithms for image and video processing, object detection and recognition, feature extraction, camera calibration, and more. OpenCV is written in C++ but has Python bindings, making it accessible and widely used in the Python programming language.

Here are some key features and functionalities of OpenCV:

1. Image and Video Processing: OpenCV allows you to load, manipulate, and process images and videos. It provides functions for basic operations like resizing, cropping, rotation, and color space conversions. You can also apply filters, perform morphological operations, and handle image enhancement tasks.

2. Object Detection and Tracking: OpenCV offers pre-trained models and algorithms for object detection and tracking, such as Haar cascades, HOG (Histogram of Oriented Gradients), and deep learning-based models. These capabilities enable tasks like face detection, pedestrian detection, and object tracking.

3. Feature Detection and Description: OpenCV provides functions for detecting and extracting features from images, including keypoint detection, feature matching, and local feature descriptors like SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features).

4. Camera Calibration: OpenCV includes tools for camera calibration, which is essential for obtaining accurate measurements from images. It allows you to calibrate cameras, estimate camera parameters, and undistort images, which is useful in applications like computer vision-based measurement systems and 3D reconstruction.

5. Machine Learning Integration: OpenCV integrates with machine learning frameworks like TensorFlow and PyTorch. This integration enables you to use deep learning models trained on these frameworks and apply them to tasks like image classification, object detection, and semantic segmentation.

6. Graphical User Interface (GUI): OpenCV provides a simple GUI module that allows you to create interactive windows, display images and videos, and capture user input events. It can be useful for quickly prototyping computer vision applications or building simple user interfaces.

OpenCV is widely used in various domains, including computer vision research, robotics, augmented reality, surveillance systems, medical imaging, and more. Its extensive documentation, large community support, and continuous development make it a powerful tool for image and video processing tasks in Python and C++.

6.2.2 Process Module

MediaPipe is an open-source cross-platform framework developed by Google Research that enables the building of real-time multimedia processing pipelines. It provides a comprehensive set of pre-built and customizable components for tasks such as video processing, audio analysis, and machine learning inference. MediaPipe simplifies the development of complex multimedia applications by providing a framework for efficient data processing, integration of machine learning models, and real-time visualization.

Here are some key features and functionalities of MediaPipe:

1. Cross-Platform Compatibility: MediaPipe is designed to be cross-platform, supporting a wide range of platforms including desktop, mobile devices (Android and iOS), and embedded systems. It allows developers to create multimedia applications that can run efficiently across multiple platforms.

2. Modular and Extensible Pipeline Architecture: MediaPipe offers a modular pipeline architecture that allows developers to connect and configure pre-built components, such as video decoders, encoders, image processing algorithms, and machine learning models. This modular approach enables rapid prototyping, easy customization, and integration of various multimedia processing elements.

3. Pre-Built Components: MediaPipe provides a collection of pre-built components for common multimedia tasks, including video stabilization, face detection and tracking, pose estimation, hand tracking, object detection, and segmentation. These components are ready to use, saving development time and effort.

4. Machine Learning Integration: MediaPipe supports the integration of machine learning models into multimedia processing pipelines. It includes tools for running deep learning models, such as TensorFlow Lite, on diverse hardware platforms. This enables developers to incorporate tasks like object recognition, gesture recognition, and semantic segmentation into their applications.

5. Real-Time Performance: MediaPipe is optimized for real-time multimedia processing, leveraging hardware acceleration and efficient algorithms. It enables low-latency and high-performance processing, making it suitable for applications that require real-time feedback or interactive experiences.

6. Visualization and Debugging Tools: MediaPipe provides tools for visualizing and debugging the processing pipeline, allowing developers to understand and monitor the flow of data through various components. This helps in diagnosing issues, optimizing performance, and refining the pipeline design.

7. Community and Ecosystem: MediaPipe has an active developer community and a growing ecosystem of third-party components and applications. This community support provides resources, examples, and contributions that enhance the framework and expand its capabilities.

MediaPipe has been widely used in various applications, including augmented reality (AR), virtual reality (VR), robotics, video analytics, and interactive multimedia experiences. Its flexibility, modularity, and performance make it a powerful tool for developers working on real-time multimedia processing projects.

NumPy (Numerical Python) is a fundamental package for scientific computing in Python. It provides powerful multidimensional array objects, along with a collection of mathematical functions, tools for array manipulation, and numerical computing capabilities. NumPy is a core library for data processing and analysis in Python and is widely used in various scientific and numerical applications.

Here are some key features and functionalities of NumPy:

1. ndarray: NumPy's ndarray (n-dimensional array) is a highly optimized, homogeneous, and multidimensional container for storing and manipulating large arrays of homogeneous data. It allows efficient storage, manipulation, and computation of large datasets, making it an essential component for numerical computing tasks.

2. Mathematical Operations: NumPy provides a wide range of mathematical functions for performing computations on arrays. These include basic operations like element-wise addition, subtraction, multiplication, and division, as well as advanced mathematical functions like trigonometric functions, exponential functions, logarithmic functions, and more.

3. Array Manipulation: NumPy offers functions for manipulating arrays, such as reshaping, slicing, indexing, concatenation, splitting, and stacking. These operations allow you to extract and manipulate subsets of arrays, combine arrays, and transform the shape and dimensions of arrays.

4. Broadcasting: NumPy employs a concept called broadcasting, which allows for efficient element-wise operations between arrays of different shapes and sizes. This feature simplifies mathematical operations and avoids the need for explicit loops.

5. Linear Algebra: NumPy includes functions for linear algebra operations, such as matrix multiplication, matrix decomposition (e.g., LU, QR, and SVD), eigenvalues and eigenvectors, solving linear equations, and more. These capabilities make NumPy a powerful tool for linear algebra computations.

6. Random Number Generation: NumPy provides a robust random module for generating random numbers and random arrays with different probability distributions. It offers various functions to generate random samples, set seed values, and control the properties of the generated random numbers.

7. Integration with other Libraries: NumPy integrates well with other scientific computing libraries in Python, such as SciPy, Matplotlib, and pandas. It serves as the foundation for these libraries, providing efficient data structures and mathematical operations.

NumPy's efficient array operations and numerical computing capabilities make it an essential tool for scientific computing, data analysis, machine learning, and other numerical applications in Python. Its broad range of functions and extensive community support contribute to its popularity and wide adoption in the scientific and data science communities.

`Time` module is a built-in module in Python that provides functions for working with time-related operations. It allows you to manipulate time values, measure time intervals, and perform various time-related calculations.

6.2.3 Output Module

Autopy is a Python library that provides cross-platform automation functionalities, allowing you to automate various tasks on your computer. It provides a simple and easy-to-use interface for simulating keyboard and mouse input, capturing and manipulating screenshots, controlling the mouse cursor, and performing other automation actions. Autopy is primarily focused on providing basic automation capabilities rather than complex robotic process automation (RPA) tasks. Here are some key features and functionalities of Autopy:

1. Keyboard Input: Autopy allows you to simulate keyboard input by programmatically pressing and releasing keys, typing strings of text, and performing keyboard shortcuts. 2. Mouse Control: Autopy provides functions for moving the mouse cursor, clicking mouse buttons, and scrolling the mouse wheel programmatically. It also supports relative mouse movement and allows you to retrieve the current position of the mouse cursor. 3. Screen Capture: Autopy enables you to capture screenshots of your computer screen or specific regions of the screen. It can save the captured images to files or provide direct access to the pixel data for further manipulation.

4. Screen Pixel Manipulation: With Autopy, you can read and modify individual pixels on the screen. It allows you to retrieve the color of a specific pixel or change the color of pixels at specific coordinates.

5. Window Management: Autopy provides functions to interact with windows on your desktop, including activating windows, bringing them to the front, and resizing or moving them programmatically.

6. Global Event Listener: Autopy allows you to listen for global events, such as keyboard and mouse events, allowing you to trigger actions based on user input.

7. Cross-Platform Compatibility: Autopy is designed to be cross-platform and works on various operating systems, including Windows, macOS, and Linux.

Autopy can be useful in a range of automation tasks, such as automating repetitive tasks, creating macro scripts, performing GUI testing, automating game actions, and creating simple user interaction scenarios. However, for more complex automation tasks or business process automation, dedicated RPA frameworks or tools might be more appropriate.

CHAPTER 7

SYSTEM IMPLEMENTATION AND TESTING

System implementation and deployment is the part in which the system will be implemented and deployed into real life to be used. The system can only be implemented and deployed when the system design and analysis is completed. Some issues are occurring during the system implementation.

7.1 IMPLEMENTATION ISSUES AND CHALLENGES

Difficulties and challenges are always occurring in implementing something new to the existing system. There are several issues and challenges that have been identified in implementing the new Fingerprint based attendance management system. These difficulties and challenges include the lack of implementation time, stable network required, developer skills, and etc., The first implementation issue and challenge is lack of implementation time.

The time given to complete the whole project might not be enough and sometime may lead to negligence. In this situation, the system might not be able to meet the final requirement since the developer will need to complete the project in rush mode. So the application was developed for a particular institution not for all. Another implementation issue and challenge is the stable network required in implementing this system. Since the attendance obtained from every faculty will be update to the autopy which access the cursor directly through the internal manipulation of coordination, so, stable network is required to faster the processing.

Other than that, users will need to clean the camera on the laptop while they scan their finger. Last but not least, the developer skill also is an implementation issue and challenge since the developer never writes programs that work with hardware. So, time taken to learn how the software will work with the hardware is kind of time consuming which may slow down the whole project progress.

7.2 TESTING

Testing is the stage of implementation aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. After the system is developed, the process of system testing must be carried on in order to test if the system is free of bugs. If during the system testing, there are bugs or errors detected, the developer may need to correct and fix the bugs immediate

CHAPTER 8
SOURCE CODE

8.1 HandTracking.py

```
import cv2 # Can be installed using "pip install opencv-python"
import mediapipe as mp # Can be installed using "pip install mediapipe"
import time
import math
import numpy as np

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

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

def findHands(self, img, draw=True): # Finds all hands in a frame
imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
self.results = self.hands.process(imgRGB)

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

def findPosition(self, img, handNo=0, draw=True): # Fetches the position of hands 45
xList = []
yList = []
bbox = []
self.lmList = []
if self.results.multi_hand_landmarks:
myHand = self.results.multi_hand_landmarks[handNo]
for id, lm in enumerate(myHand.landmark):
h, w, c = img.shape

cx, cy = int(lm.x * w), int(lm.y * h)
xList.append(cx)
```

```
List.append(cy)
self.lmList.append([id, cx, cy])
if draw:
cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

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

if draw:
cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
(0, 255, 0), 2)
return self.lmList, bbox
def fingersUp(self): # Checks which fingers are up
fingers = []

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

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

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

# totalFingers = fingers.count(1)

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

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

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


def main():
pTime = 0
cTime = 0
cap = cv2.VideoCapture(1)
detector = handDetector()
while True:
success, img = cap.read()
img = detector.findHands(img)
lmList, bbox = detector.findPosition(img)
if len(lmList) != 0:
print(lmList[4])

cTime = time.time()
fps = 1 / (cTime - pTime)
pTime = cTime
cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3, (255, 0, 255), 3)
cv2.imshow("Image", img)
cv2.waitKey(1)
if __name__ == "__main__":
main()
```

8.2 VirtualMouse.py

```
import cv2
import numpy as np
import time
import HandTracking as ht
import autopy # Install using "pip install autopy"

### Variables Declaration
pTime = 0 # Used to calculate frame rate

width = 640 # Width of Camera
height = 480 # Height of Camera
frameR = 100 # Frame Rate
smoothening = 8 # Smoothening Factor
prev_x, prev_y = 0, 0 # Previous coordinates
curr_x, curr_y = 0, 0 # Current coordinates

cap = cv2.VideoCapture(0) # Getting video feed from the webcam
cap.set(3, width) # Adjusting size
cap.set(4, height)

detector = ht.handDetector(maxHands=1) # Detecting one hand at max screen_width,
screen_height = autopy.screen.size() # Getting the screen size while True:
success, img = cap.read()
img = detector.findHands(img) # Finding the hand
lmlist, bbox = detector.findPosition(img) # Getting position of hand

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

fingers = detector.fingersUp() # Checking if fingers are upwards
cv2.rectangle(img, (frameR, frameR), (width - frameR, height - frameR), (255, 0, 255), 2) #
Creating boundary box
if fingers[1] == 1 and fingers[2] == 0
: # If fore finger is up and middle finger is down
if fingers[1] == 1 and fingers[2] == 1: # If fore finger & middle finger both are up length, img,
lineInfo = detector.findDistance(8, 12, img)
```

```
if length < 40: # If both fingers are really close to each other
cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0), cv2.FILLED)
autopy.mouse.click() # Perform Click


cTime = time.time()
fps = 1/(cTime-pTime)
pTime = cTime
cv2.putText(img, str(int(fps)), (20, 50), cv2.FONT_HERSHEY_PLAIN, 3, (255, 0, 0), 3)
cv2.imshow("Image", img)
cv2.waitKey(1)
```

CHAPTER 9
SCREENSHOTS

INPUT MODULE

Getting the User Interaction through web camera

Fig no: 9.1.1 Hardware used by the project

PROCESS MODULE

Identify the hand and the fingers and mark the box around them.

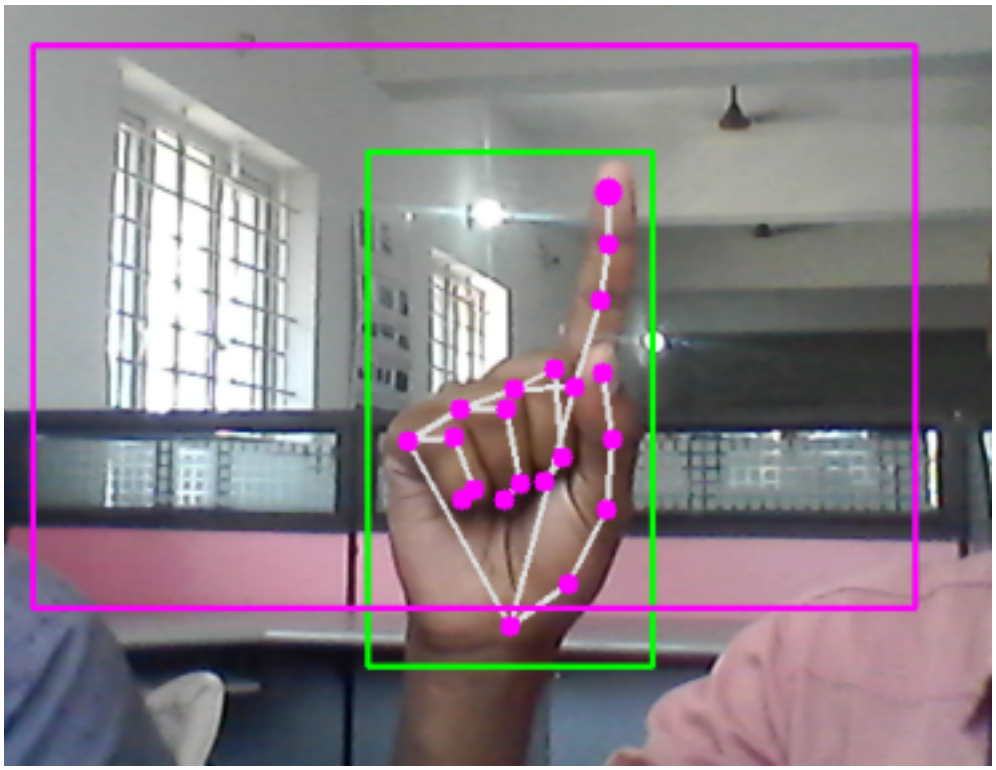


Fig no: 9.1.2 Detecting the user hand

OUTPUT MODULE

Moving the cursor based on movement of index finger.

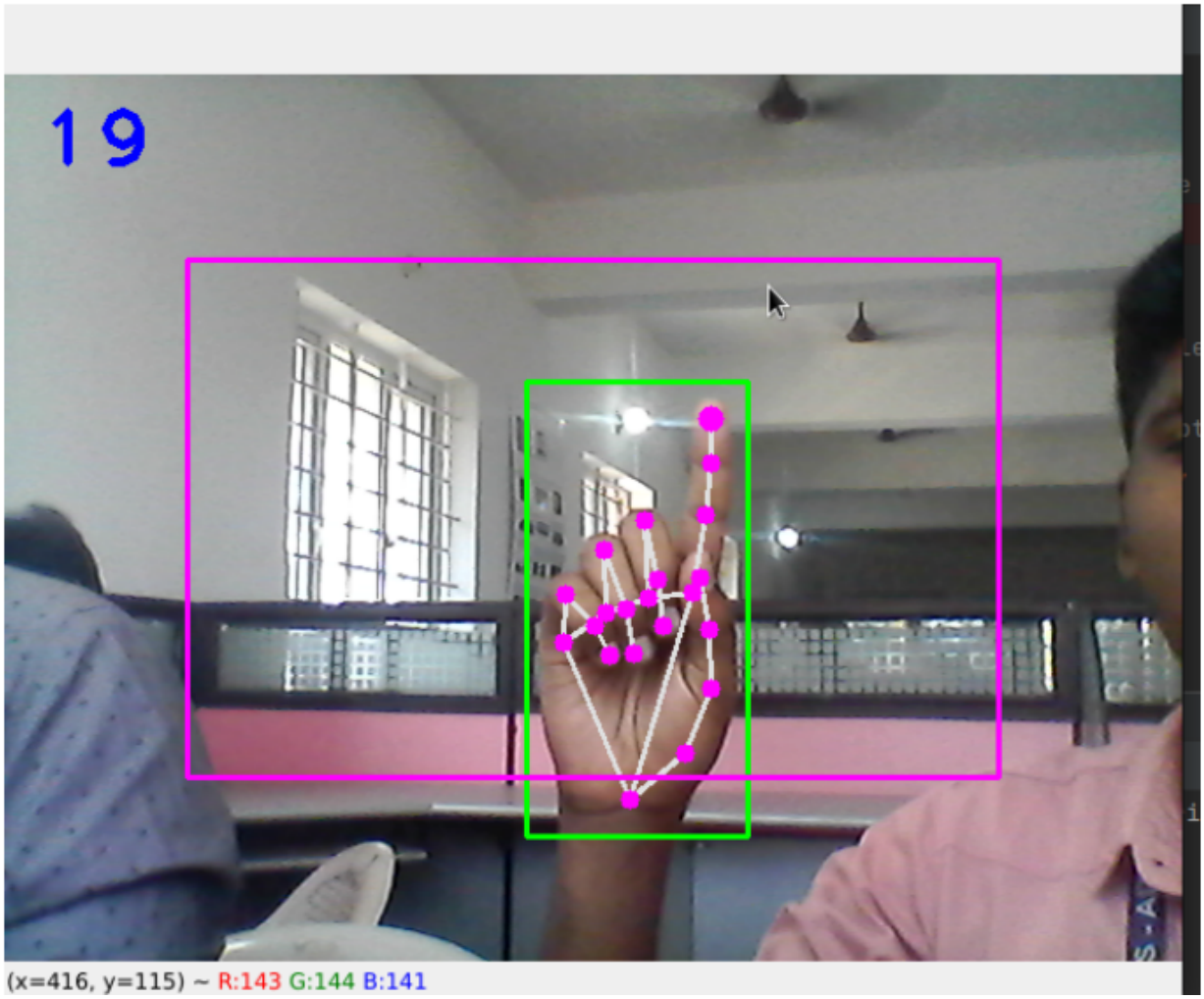
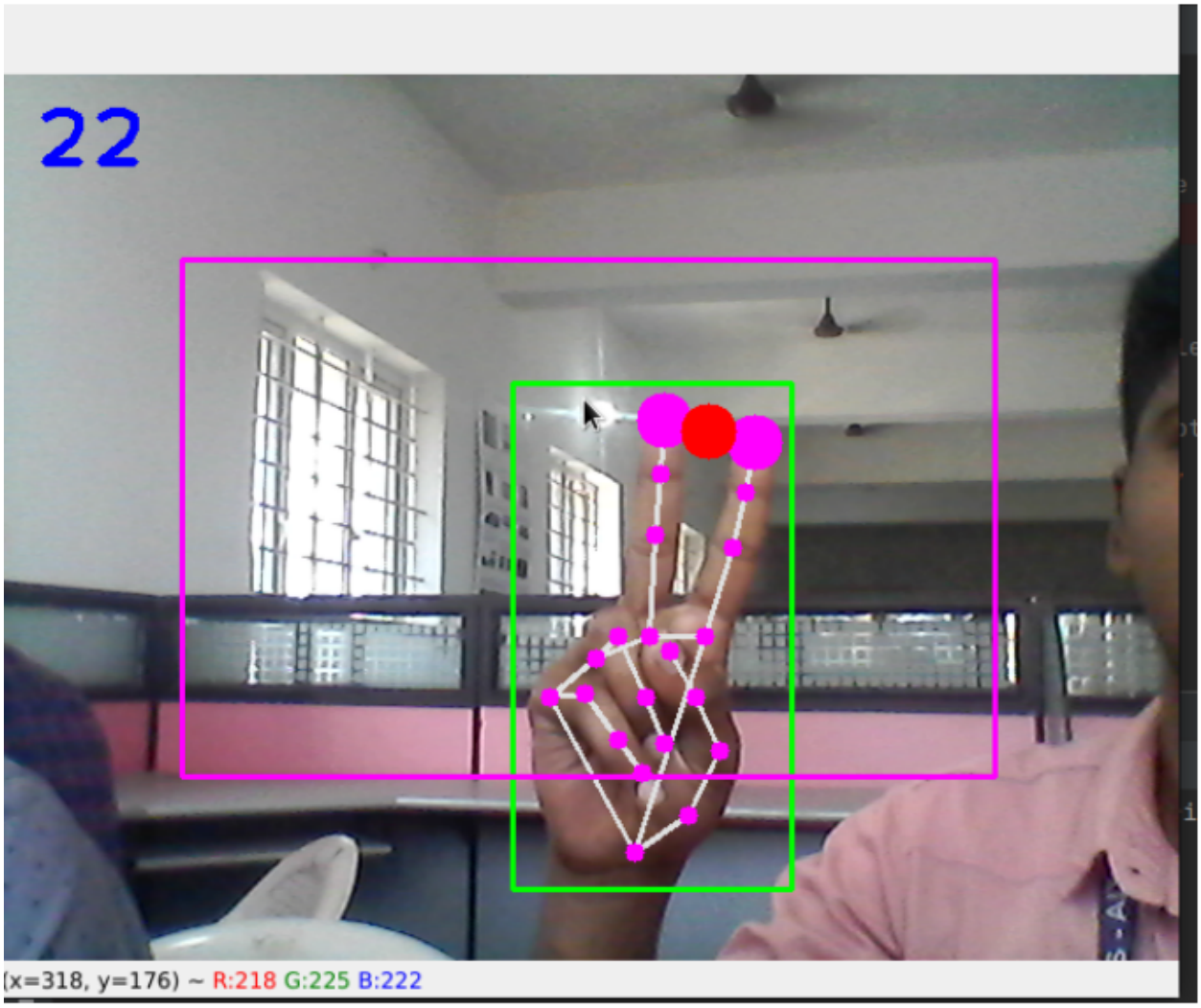


Fig no : 9.1.3 Tracking the finger

OUTPUT MODULE

Perform the clicking operation in order to select or open a folder or software.



CHAPTER 10
CONCLUSION

10.1 CONCLUSION

In conclusion, a virtual mouse is a software-based solution that allows users to simulate mouse movements and actions without the need for a physical mouse. It provides an alternative input method for controlling the cursor and interacting with graphical user interfaces (GUIs) using software-based controls.

Python can be used to develop a virtual mouse by leveraging various libraries and modules. The `pyautogui` library, for example, provides functions for controlling the mouse cursor, clicking mouse buttons, and performing other mouse-related actions programmatically. By utilizing Python's capabilities, developers can create custom virtual mouse implementations that suit their specific needs.

Some advantages of using a virtual mouse include:

1. Accessibility: Virtual mice can be beneficial for individuals with physical disabilities or limitations that make it difficult to use a physical mouse. They offer alternative input methods and can be customized to accommodate different accessibility needs.
2. Remote Access: Virtual mice can be utilized in remote access scenarios, where users control a computer or device from a remote location. This allows for remote control and operation without the need for a physical mouse.
3. Automation and Scripting: Virtual mice can be incorporated into automation workflows and scripting tasks. Python's scripting capabilities and libraries like `pyautogui` enable the creation of automated mouse movements and actions, streamlining repetitive tasks.
4. Gaming and Virtual Environments: Virtual mice can be employed in gaming or virtual reality applications. They allow users to control in-game actions or interact with virtual environments using software-based mouse controls.

It is important to note that virtual mice may not provide the same level of precision or tactile feedback as physical mice. Additionally, they may require additional configuration or calibration to match individual user preferences or system requirements.

Overall, virtual mice offer flexibility, accessibility, and automation capabilities, making them useful in various contexts such as accessibility, remote access, automation, and gaming. Python's extensive libraries and modules further enhance the development and customization of virtual mouse solutions.

.

10.2 SCOPE FOR THE FUTURE ENHANCEMENT

In conclusion, a virtual mouse is a software-based solution that allows users to simulate mouse movements and actions without the need for a physical mouse. It provides an alternative input method for controlling the cursor and interacting with graphical user interfaces (GUIs) using software-based controls.

Python can be used to develop a virtual mouse by leveraging various libraries and modules. The `pyautogui` library, for example, provides functions for controlling the mouse cursor, clicking mouse buttons, and performing other mouse-related actions programmatically. By utilizing Python's capabilities, developers can create custom virtual mouse implementations that suit their specific needs.

Some advantages of using a virtual mouse include:

1. Accessibility: Virtual mice can be beneficial for individuals with physical disabilities or limitations that make it difficult to use a physical mouse. They offer alternative input methods and can be customized to accommodate different accessibility needs.
2. Remote Access: Virtual mice can be utilized in remote access scenarios, where users control a computer or device from a remote location. This allows for remote control and operation without the need for a physical mouse.
3. Automation and Scripting: Virtual mice can be incorporated into automation workflows and scripting tasks. Python's scripting capabilities and libraries like `pyautogui` enable the creation of automated mouse movements and actions, streamlining repetitive tasks.
4. Gaming and Virtual Environments: Virtual mice can be employed in gaming or virtual reality applications. They allow users to control in-game actions or interact with virtual environments using software-based mouse controls.

It is important to note that virtual mice may not provide the same level of precision or tactile feedback as physical mice. Additionally, they may require additional configuration or calibration to match individual user preferences or system requirements.

CHAPTER 11
REFERENCES

REFERENCES

1. “Fingertip-writing alphanumeric Character Recognition for Vision-based Human Computer Interaction” by Cheng-Yuan Shih, Chien-Cheng Lee, and Bor-ShennJeng,in International Conference on Broadband, Wireless Computing, Communication and Applications 978-0-7695-4,2010

2. Hemanta Saik, Abhik Banerjee, Koustuvmoni Bharadwaj, Abhirup Ghosh, “Mouse Control using a Web Camera based on Colour Detection”in International Journal of Computer Trends and Technology (IJCTT) – volume 9.

3. “Virtual Mouse Implementation using Color Pointer Detection” by D.S. Suresh and I.V. Bhavana.

4. “A virtual mouse interface with a two layered Bayesian network” by S.W. Lee, D. Kang, S. Huh, M.C. Roh.

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