

**VISION-BASED
GESTURE - CONTROLLED CURSOR
NAVIGATION SYSTEM**

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
Artificial Intelligence

By

**MOHAMMED HASHIM A (Reg.No - 40731063)
NEHAL AHMED A (Reg.No – 40731069)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE
CATEGORY -1 UNIVERSITY BY UGC**

**JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MOHAMMED HASHIM A(40731063)** and **NEHAL AHMED A(40731069)** who carried out the Project Phase-2 entitled “**Vision-Based Gesture-Controlled Cursor Navigation System**” under my supervision from December 2023 to April 2024.

Internal Guide
Mrs.D.SUDHA , M.E.,(Ph.D.)

Head of the Department
Dr. L. Lakshmanan M.E.,Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **NEHAL AHMED A(40731069)**, hereby declare that the Project Phase-2 Report entitled “**Vision-Based Gesture-Controlled Cursor Navigation System**” done by me under the guidance of **Mrs. D. SUDHA, M.E.,(Ph.D.)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering with Specialization in Artificial Intelligence**.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E.,Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.D.SUDHA , M.E.,(Ph.D.)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering with Specialization in Artificial Intelligence** who were helpful in many ways for the completion of the project.

ABSTRACT

Researchers worldwide are directing their efforts towards creating more interactive solutions. Among these endeavors, the conventional mouse stands as a stalwart device for facilitating computer interactions, available in wired, wireless, and Bluetooth variants. However, a paradigm shift is underway with the proposal of an innovative interactive computer system that operates seamlessly without the need for a physical mouse. This groundbreaking project harnesses cutting-edge machine learning and computer vision algorithms to recognize hand gestures, eliminating the necessity for any external hardware. The system's compatibility with Convolutional Neural Network (CNN) models implemented by Mediapipe further solidifies its efficacy. Notably, the implications extend far beyond mere convenience, offering a lifeline to individuals facing physical challenges, such as partial paralysis, who encounter difficulties operating traditional mice. Central to the proposed system is its reliance on computer vision technology, allowing users to seamlessly control mouse functionalities through intuitive hand gestures. By simply varying the number of fingers displayed, users can execute a myriad of actions, including clicking and dragging items, without physical contact with any hardware. Such a breakthrough promises to revolutionize accessibility and user experience across diverse demographics. Moreover, the system's streamlined design ensures practicality and ease of implementation. With a single computer serving as the hub, a camera assumes the role of the primary input device, capturing and interpreting hand gestures with precision. Leveraging Python and OpenCV, the system harnesses the power of these robust programming tools to deliver seamless functionality. In practice, the output from the camera is displayed directly on the system's interface, enabling real-time feedback and fine-tuning to meet the user's preferences. This intuitive approach not only enhances usability but also fosters a dynamic and engaging computing experience. The proposed gesture-controlled interactive computer system represents a significant leap forward in human-computer interaction technology. By harnessing the power of machine learning and computer vision, it transcends the limitations of traditional input devices, offering unparalleled accessibility and functionality. With its potential to empower users of all abilities, this innovation heralds a new era of inclusive and immersive computing.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Motivation of virtual mouse	2
	1.2 Significant impact	2
2	LITERATURE SURVEY	3
	2.1 Inferences from Literature Survey	4
	2.2 Open problems in Existing System	6
3	REQUIREMENT ANALYSIS	7
	3.1 Aim and Scope	7
	3.2 Feasibility Studies/Risk Analysis of the Project	8
	3.3 Software Requirements Specification Document	10
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 Selected Methodology or process model	11
	4.2 Architecture / Overall Design of Proposed System	12
	4.3 Project Management Plan	15
5	IMPLEMENTATION DETAILS	17
	5.1 Implementing Gesture Recognition	17
	5.2 Description of Software for Implementation and Testing plan of the Proposed Model/System	19
6	RESULTS AND DISCUSSIONS	22
	6.1 Overview	22
	6.2 Performance in various Environments	22
	6.3 Screenshot of Outputs	25
7	CONCLUSION	28
	7.1 Summary	28

7.2	Limitation	29
REFERENCES		30
APPENDIX		31
A. SOURCE CODE		31
B. SCREENSHOTS		37

LIST OF FIGURES

FIGURE NAME NO	F I G U R E	Page No.
		11
4.1	Block diagram for proposed system	13
4.2	Virtual Mouse Block Diagram	14
4.3	Flowchart for proposed system	15
4.4	Co-ordinates or landmark in hand	25
6.1	Cursor Movement	25
6.2	Left Click	26
6.3	Right Click	26
6.4	Double Click	27
6.5	Scroll Up	27
6.6	Scroll Down	37
B.1	Hand Tracking	37
B.2	Virtual Mouse	

CHAPTER 1

INTRODUCTION

Gesture Recognition has been very interest problem in computer vision community for a long time. Hand gestures are a facet of visual communication which will be conveyed through the middle of the palm, the finger position and therefore the shape constructed by the hand. Hand gestures are often classified into static and dynamic. As its name implies, the static gesture refers to the stable shape of the hand, whereas the dynamic gesture comprises a series of hand movements such as waving. There are a spread of hand movements within a gesture; for instance, a handshake varies from one person to a different and changes consistent with time and place. The main difference between posture and gesture is that posture focuses more on the form of the hand whereas gesture focuses on the hand movement. Computer technology has tremendously grown over the past decade and has become a necessary a part of everyday live. The primary accessory for Human Computer Interaction (HCI) is the mouse. The mouse isn't suitable for HCI in some real-world situations, like with Human Robot Interaction (HRI). There are many research on alternative methods to the pc mouse for HCI. The most natural and intuitive technique for HCI, that's a viable replacement for the pc mouse is with the utilization of hand gestures. While the majority of the people take these facilities for granted, people with physical impairments face many difficulties in properly using these devices. In particular, people with severe movement disabilities may have physical impairments which significantly limit their ability to control the fine motor. Therefore, they may not be able to type and communicate with a normal keyboard and mouse. In this situation, it is important to use effective assisted technologies to ensure accessibility for such people. Our vision was to develop a virtual mouse system that uses a camera to speak with the device during a more user-friendly way, as an alternate to employing a touch screen. In order to harness the full potential of a webcam, it can be used for vision-based CC, which would effectively track the hand gesture predict the gesture on basis of labels. Our vision became to broaden a virtual mouse system that makes use of an internet digicam to talk with the device in a more person-friendly way, as an alternative to using a touch display screen. If you want to harness the entire potential of a webcam, it may be used for vision based totally cc, which could efficiently track the hand gesture predict the gesture on basis of labels.

1.1 MOTIVATION OF VIRTUAL MOUSE

It is safe to predict that the Virtual the Mouse will soon take the place of the conventional physical in nature mouse in the not-too-distant future, as people strive to live in a world where every technological appliance can be operated and interacted with remotely without the need for any peripheral devices, such as remote controls, keyboards, etc. Not only does it offer ease, but it also saves money.

User-Friendly

Man-made consciousness has different applications in the present society. It is becoming fundamental for the present time since it can take care of complicated issues with an effective way in various ventures. Virtual Mouse doesn't need any of that since all that is needed is a camera to take pictures of the user's hand position and utilize those images to establish where the points should be.

Cost Effective

An AI virtual mouse, also known as a software-based mouse or an on-screen mouse, can be cost effective compared to a physical mouse for a few reasons: 1.No need for hardware: A virtual mouse does not require any additional hardware, such as a physical mouse, to function. 2. Accessibility: A virtual mouse can be used by individuals who may have difficulty using a physical mouse due to physical limitations, such as a disability. 3. Compatibility: A virtual mouse is typically compatible with most computer systems, regardless of the hardware or operating system being used. 4. Ease of use: A virtual mouse is typically easy to use and requires minimal training.

1.2 SIGNIFICANT IMPACT

The usage of a hardware computer mouse in conjunction with manual mouse inputs and mouse positioning is projected to be replaced by the Virtual Mouse program. With the use of gestures, every job can be completed with this program, making computer use easier. Furthermore, by simply displaying the right combination of colors to the webcam, the Virtual Mouse program enables persons with motor impairments to interact with the computer.

CHAPTER 2

LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers and automotive companies in a bid to create a solution, and all their solutions vary from analyzing various patterns.

Aashni Hariiaa et al. [1] proposed "Hand Gesture Recognition for Human Computer Interaction," which uses a background extraction and contours detection system to improve user-computer interaction. Human interaction can be accomplished through a variety of sensor modes such as gesture, speech, facial and body expressions. In their paper, Horatiu-Stefan et al. proposed a "Human Hand Gesture Based System for Mouse Cursor Control". This model was created using a blue colored hand pad, a webcam, and a computer. Specific operations were carried out based on hand pad postures. This model was created using Visual C++ 2008 and the OpenCV library.

Abhik Banerjee et al. [2] proposed a "Mouse Control Using a Web Camera Based on Color Detection" to control cursor movements and click events by detecting camera colour. Each colour represents a different cursor control, and clicking actions are performed by simultaneously detecting the colours. This method was created with the help of MATLAB software and the MATLAB image processing tool box.

Abhilash et al. [3], for controlling the mouse cursor and performing clicking operations proposed the "Virtual Mouse Using Hand Gesture" technique. They made a mask out of a red object in this case. This method was implemented using Python software, which included a variety of modules and functions.

Alisha Pradhana et al. [4] proposed the "Design of Intangible Interface for Mouseless Computer Handling Using Hand Gestures" to control the mouse cursor and click operations. This method, which employs a convex hull algorithm, was implemented using Microsoft virtual studio, a Microsoft integrated development environment. To detect the user's hand and reduce noise, a red glove is used.

Chiung Hsieh et al. [5] proposed "A Real Time Hand Gesture Recognition System Using Motion History Image" to control the mouse cursor. The proposed method employs an adaptive skin colour detection model to reduce misclassifications. To develop these methodologies, they used a C++ software platform with the image processing library open cv installed.

Pooja Kumari et al. [6] proposed "Cursor Control Using Hand Gestures" for controlling a mouse with camera captured hand gestures. The camera acts as a sensor in this method, capturing and recognising colour tips attached to the hand. Because it requires the user to have colour tips on his hand in order to control the mouse, this method is also known as the marker-based approach method. To implement this methodology, they used the MATLAB environment, the MATLAB Image Processing Tool box, and the OpenCV library.

2.1 INFERENCES FROM LITREATURE SURVEY

Based on the literature survey on various methods for controlling the mouse cursor using hand gestures, we can infer:

Diverse Implementation Platforms

Researchers have used a variety of programming languages and software platforms to develop their mouse cursor control systems. These include C++, Java, MATLAB, C programming, Python, and Microsoft Visual Studio. The choice of platform depends on factors like the application's requirements and the ease of implementation in a specific environment.

Image Processing Libraries

OpenCV and MATLAB Image Processing Toolbox are commonly used libraries for image processing and computer vision tasks. These libraries play a crucial role in various aspects of these systems, such as color detection, skin detection, and image manipulation.

Color Detection

Several methods rely on color detection and the use of colored markers, gloves, or objects to control the mouse cursor. These color-based markers act as reference points for the system to interpret hand gestures and movements.

Gesture Recognition Techniques

Gesture recognition is a common theme in many of these approaches. Some methods use skin color detection, while others employ techniques like background subtraction, HSV color models, and contour detection. These techniques help in recognizing hand gestures and translating them into mouse movements or clicks.

Hardware Components

Some approaches use external cameras or webcams to capture hand movements and gestures, while others use specialized hardware like data gloves and colored hand pads. These hardware components enhance the accuracy and efficiency of gesture recognition.

Programming Languages

The choice of programming language varies, with C++, Java, Python, and others being used. The choice often depends on factors like the specific requirements of the application, the availability of libraries, and the familiarity of the researchers with a particular language.

Machine Learning

Machine learning is employed in some methods, such as the "Gesture Recognition Using Data Glove," where data from a novel data glove is collected and processed using advanced machine learning techniques.

Efficiency and Reliability

Several methods focus on improving the efficiency and reliability of cursor control through hand gestures. This includes the use of different colored caps, gloves, or markers to enhance recognition and reduce noise.

Human-Computer Interaction

Many of these methods aim to enhance human-computer interaction through gesture-based control. They offer potential applications in fields such as gaming, accessibility, and virtual environments.

Sensor Modes

Hand gestures are just one of the sensor modes used for human-computer interaction. Some approaches explore other modes like speech, facial expressions, and body gestures to provide a more comprehensive interaction experience.

This literature survey highlights the diversity of methods and techniques used in the field of mouse cursor control through hand gestures. Researchers employ various software platforms, image processing libraries, hardware components, and gesture recognition techniques to develop systems with the aim of improving the efficiency and reliability of human-computer interaction. These approaches have potential applications in a wide range of domains, from gaming to accessibility and beyond.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The existing system proposed a very high demanding cost and resources. Due to which there were not many user who could afford it.

Also the environment used in the project was not friendly due to which not all user were not able to handle it.

The existing system is not very known as it has many drawbacks to look into the project.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 AIM AND SCOPE

Aim

This project aims to create a user-friendly, vision-based virtual mouse that utilizes hand gestures for computer interaction, effectively eliminating the need for a physical mouse. By leveraging computer vision and machine learning techniques, particularly focusing on convolutional Neural Networks (CNN) models compatible with MediaPipe, the aim is to develop a seamless and intuitive interface for users to control their computers.

Scope

The primary objective is to implement robust hand gesture recognition using computer vision and machine learning algorithms. This entails detecting and interpreting hand movements captured by a webcam, translating these gestures into corresponding mouse actions such as cursor movement, left/right clicks, and dragging. The system will also include features for user calibration and feedback, ensuring a smooth and personalized interaction experience.

Technical Specifications:

The development will be carried out using Python along with the OpenCV library, a popular choice for computer vision applications. The system will require a standard computer equipped with a webcam, serving as the primary input device. By harnessing the capabilities of these tools, the project aims to deliver a reliable and accessible solution for users with diverse needs.

A key focus of this project is to prioritize ease of use for all users, particularly individuals with physical limitations that may hinder their ability to use a traditional mouse. By offering an alternative input method based on hand gestures, the virtual mouse aims to enhance accessibility and inclusivity in computer interaction.

Certain aspects are deliberately excluded from the project scope to maintain focus and feasibility. This includes integration with external devices beyond the webcam, as well as features such as voice command integration and advanced functionalities like scrolling or multi-finger gestures. While these may be considered for future expansion, the current scope prioritizes the development of core functionalities for the virtual mouse application.

3.2 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

There are lots of risks for a project involving the development of a mouse cursor control system using hand gestures and various other techniques involves identifying potential risks and assessing their impact and likelihood.

Technical Risks

- Software Bugs and Glitches: The project may face technical challenges related to software development, leading to bugs, glitches, or unexpected behavior.
- Hardware Compatibility: Compatibility issues with different cameras, sensors, or hardware components could arise.
- Performance Issues: The system may not meet performance expectations, resulting in lag or inaccuracies in cursor control.
- Limited Gesture Recognition: The system might struggle to accurately recognize complex or nuanced hand gestures.

Data and Model Risks

- Data Quality: Insufficient or noisy training data could lead to subpar gesture recognition models.
- Model Overfitting: Overfitting of machine learning models to training data could result in poor generalization to real-world use.

Security and Privacy Risks

- Data Privacy: The system may capture sensitive data unintentionally, posing privacy concerns.
- Security Vulnerabilities: Security vulnerabilities in the software could expose users to hacking or data breaches.

Usability and User Experience Risks

- User Training: Users may find it challenging to learn and use the gesture-based interface effectively.
- User Fatigue: Prolonged use of gestures may cause user fatigue or discomfort.

Environmental Risks

- Lighting Conditions: Variations in lighting conditions may affect the performance of color-based recognition methods.
- Hardware Failures: The failure of cameras, sensors, or hardware components could disrupt system functionality.

Cost and Resource Risks

- Budget Overruns: The project might exceed its allocated budget due to unexpected costs or scope changes.
- Resource Constraints: Insufficient resources, such as time, personnel, or hardware, may impede progress.

Regulatory and Compliance Risks

- Legal Compliance: The project may need to adhere to legal regulations related to data collection, privacy, and accessibility.
- Accessibility Compliance: Ensuring that the system is accessible to users with disabilities could be a regulatory requirement.

Market and Adoption Risks

- Competitive Landscape: The market may already have competing products or technologies, affecting adoption rates.
- User Adoption: Users may resist adopting gesture-based control if it's unfamiliar or less convenient than existing input methods.

External Dependencies

- Third-Party Services: Relying on third-party services or libraries could introduce risks related to their availability and compatibility.

Project Management Risks

- Scope Creep: Expanding the project's scope beyond the initial plan may lead to delays and resource constraints.
- Communication and Collaboration: Ineffective communication among team members can lead to misunderstandings and coordination challenges.

To mitigate these risks, we should consider the following strategies:

- Rigorous testing and quality assurance processes to identify and address technical issues.
- Careful data collection and preprocessing to ensure data quality.
- Regular software updates and security audits to address vulnerabilities.
- Extensive user testing and feedback to improve usability.
- Robust project management practices to monitor scope and resource allocation.
- Compliance with relevant legal and regulatory requirements.
- Thorough market research to understand the competitive landscape and user needs.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

These are the software requirements needed to run the program. All these requirements are essential for the proper functioning of the project:

- Python (version 3.6 – 3.8.5)
- OpenCV
- pycaw
- PyAutoGUI
- AutoPy
- MediaPipe
- NumPy

These software requirements provide a foundation for developing the project, ensuring that it meets functional, performance, and usability expectations. Additional, more detailed requirements may be necessary based on the project's specific goals and use cases.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

This project promotes an approach for the human computer interaction, where cursor motion can be managed by the usage of a real-time digital camera. it's far an alternative to the present day methods including manual input of buttons or converting the positions of a physical pc mouse. As an alternative, it utilizes a digital camera and computer imaginative and prescient technology to control numerous mouse activities and is able to acting each assignment that the bodily pc mouse can. We'll first use mediapipe to recognize the hand and the hand key factors. Mediapipe returns a total of 21 key points for each detected hand. Palm detection model and hand landmark version are utilized by mediapipe to discover hand. First palm is detected as it's far an easy procedure with recognize at hand landmark.

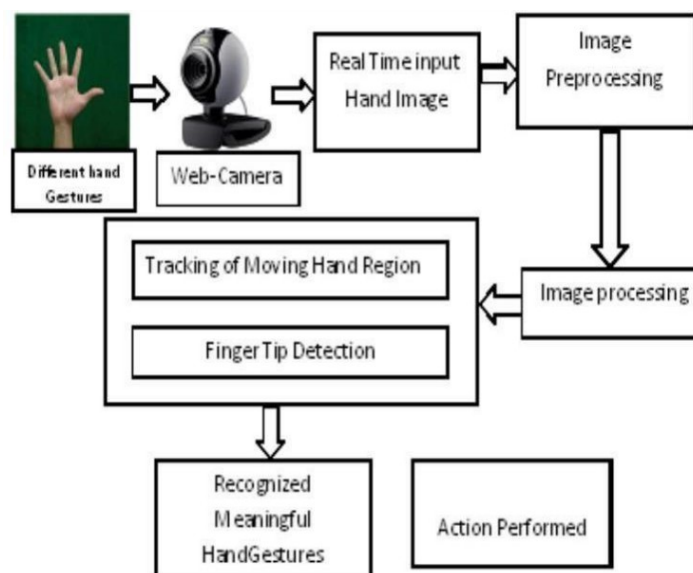


Fig 4.1: Block diagram for proposed system

Fig-4.1 depicts a functional block diagram of the proposed system, which demonstrates how the system operates. We must hold our hand up to the webcam. The webcam starts the video and captures the frames. The input image was subjected to pre-processing. The function of the image pre-primary processor is to standardize the image. The process of scaling and pre-processing an image to have

similar heights and widths is known as standardization. To improve the quality of the standard image, it is now processed using an image processing technique. The camera moves its hand after image processing, and the finger tips are detected using MediaPipe and openCV. After recognizing the hand and finger tips, it begins to draw. There are hand landmarks and a box around the hand on the screen. On the window pc, draw a rectangular box to hold the mouse. It will determine which of your fingers is up and which is down. Based on the finger detections, the mouse action is performed, and the program returns to the frames to perform the next operation. This is how the entire system operates.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

Step 1: Start

Step 2: Start the webcam video capture and initialize the system.

Step 3: Frame capture with a webcam.

Step 4: Using Media Pipe and OpenCV, detect hands and hand tips and draw hand landmarks and a box around the hand.

Step 5: Draw a rectangle around the computer window area where we'll be using the mouse.

Step 6: Determine which finger is raised.

Step 6.1: The gesture is neutral if all five fingers are up, and the next step is taken. Step 6.2: The cursor moves to step 2 if both the middle and index fingers are raised. Step 6.3: A double click is performed when both index and middle fingers are joined side by side, and step 2 is performed.

Step 6.4: If both index and middle fingers are down, perform a left click and proceed to step 2.

Step 6.5: If the middle finger is down and the index finger is up, the right click is performed and the process proceeds to step 2.

Step 6.6: Volume up and down are accomplished by joining the thumb and index fingers and moving them up and down.

Step 7: To exit, press the EXIT key.

System Design

There are two main steps in the process of color recognition: the calibration phase and the recognition phase. In the calibration phase, which will be utilized later in the recognition phase, the system will be able to identify the Hue Saturation Values of the colors selected by the users. It will save the parameters and settings into text documents for later use. The system will begin to take frames during the recognition phase and look for color input based on the values that have been stored during the calibration process phase. The following figure depicts the stages of the virtual mouse:

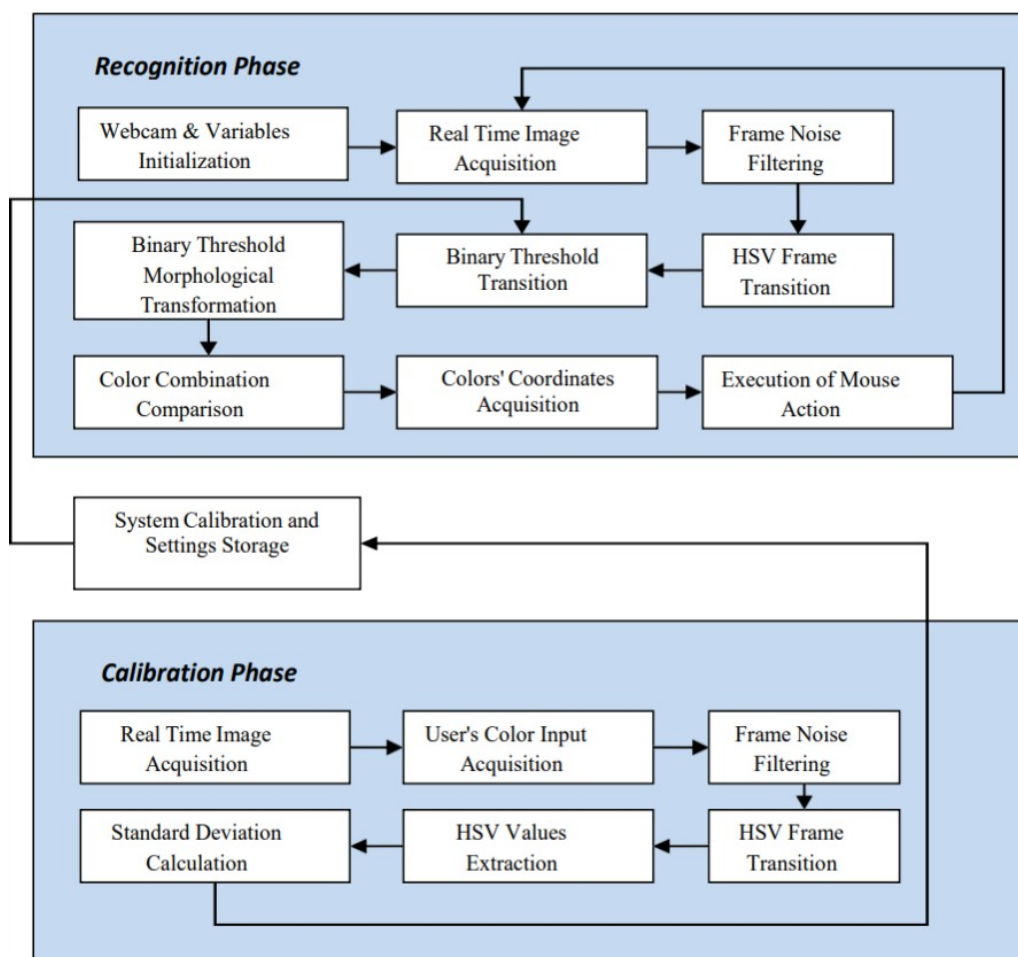


Fig 4.2: Virtual Mouse Block Diagram

The preceding algorithm shows how to simulate virtual mouse control with hand gestures step by step. This makes it easier to swap a physical mouse for a virtual mouse. This will aid in the conversion of input hand gestures into mouse clicking functions. The preceding procedure, i.e. the algorithm, is depicted diagrammatically in flow chart Fig-4.3.

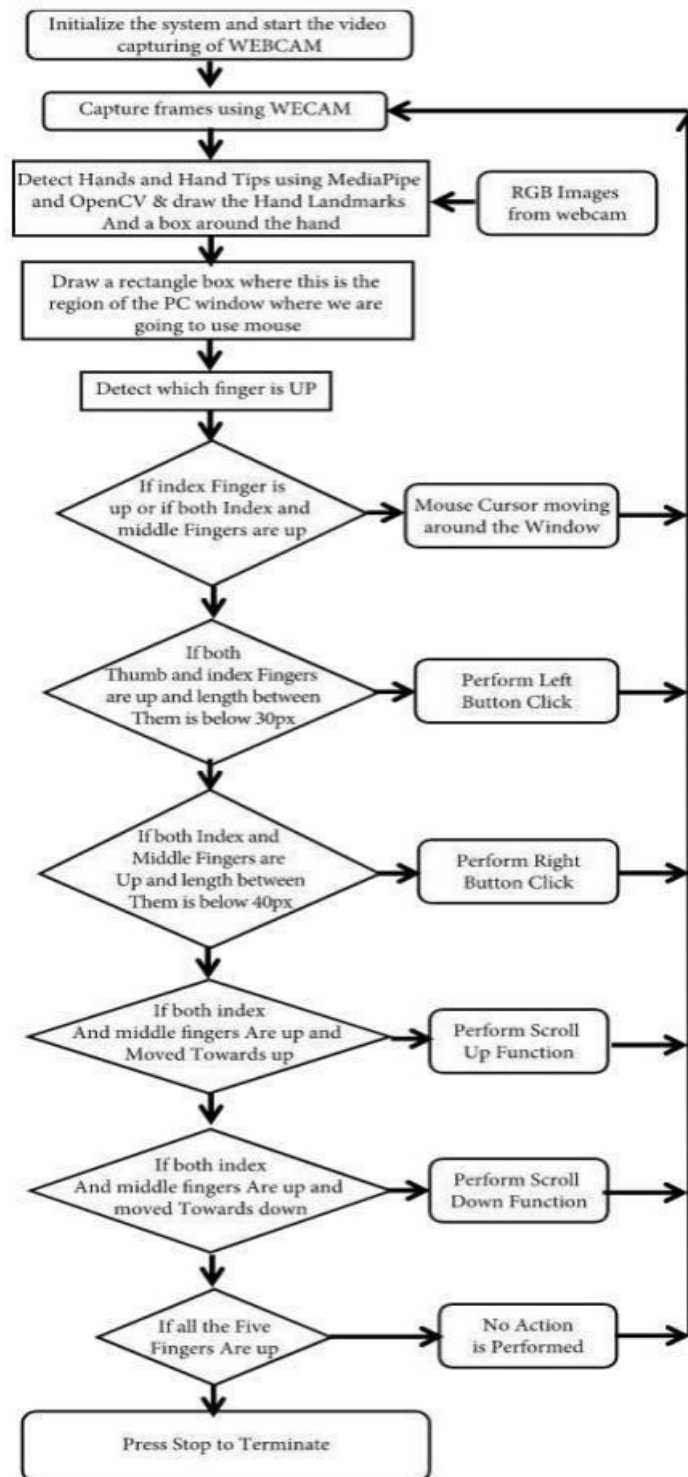


Fig 4.3: Flowchart for proposed system

The AI Virtual Mouse System uses a camera. The proposed AI virtual mouse system is based on frames captured by a laptop or PC's webcam. The video capture object is created using the Python computer vision library OpenCV, as shown in Fig-4.1, and the web camera begins capturing video. The web camera

captures images, which are then sent to the AI virtual system. The video is recorded and processed. The AI virtual mouse system uses a webcam to capture each frame until the program is terminated.

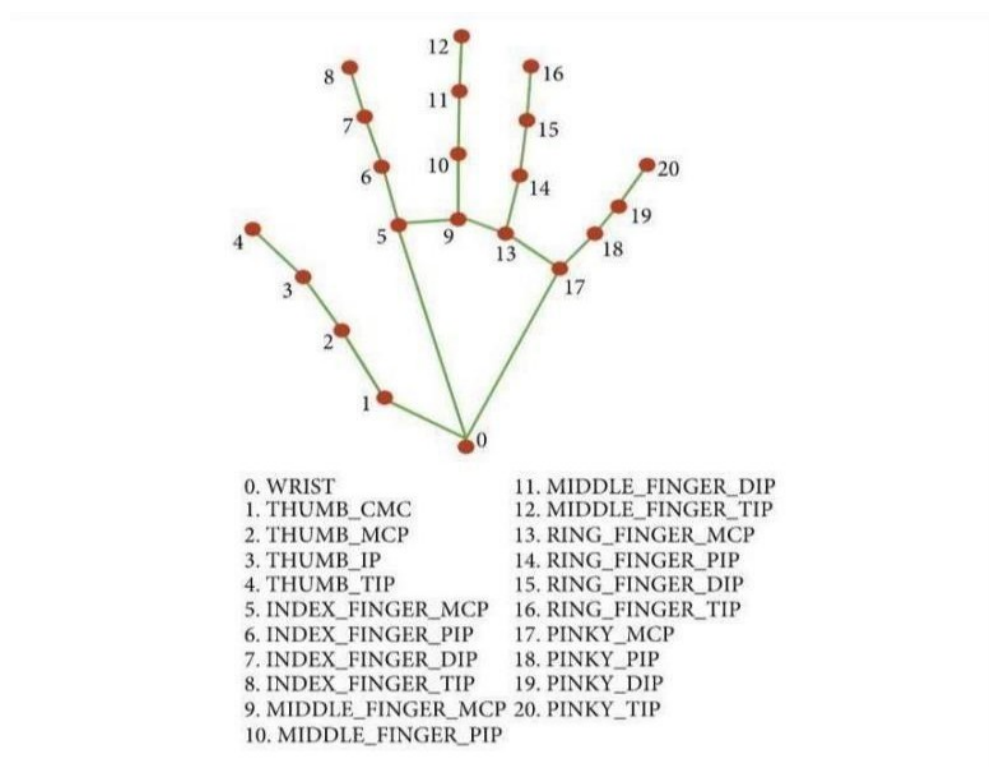


Fig 4.4 Co-ordinates or landmark in hand

Detecting which finger is up and carrying out mouse functions. At this point, we use the tip Id of the respective finger discovered using the MediaPipe and the respective co-ordinates of the up fingers, as shown in Fig-4.4, to determine which finger is up, and then we perform the appropriate mouse function. Mouse functions based on hand gestures and hand tip detection using computer vision.

4.3 PROJECT MANAGEMENT PLAN

Data Acquisition

The system begins by capturing data through a camera or depth-sensing sensor. This sensor records visual information about the user's hand or body movements in front of the camera.

Image Processing

The captured images or depth data are processed using image processing and computer vision algorithms. These algorithms analyze the data to identify and track relevant features, such as the user's hand or specific marker.

Gesture Recognition

Once the system identifies the user's hand or gestures, it uses gesture recognition algorithms to interpret these movements. These algorithms compare the observed gestures to predefined gesture templates or models to determine the user's intention.

Cursor Control

After recognizing the gestures, the system translates them into cursor movements on the computer screen. It determines the direction and speed of the cursor based on the detected gestures, allowing the user to navigate the screen.

User Feedback

The system often provides visual or audio feedback to inform the user of successful gesture recognition and cursor movement. This feedback helps users understand their interactions with the system.

Calibration

To ensure accurate gesture recognition and cursor control, the system may require initial calibration. This step helps align the system with the user's environment and setup.

Continuous Interaction

The system operates in real-time, continuously tracking the user's movements and responding to gestures as they occur. This creates a responsive and natural user interface.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 IMPLEMENTING GESTURE RECOGNITION

Within the expansive project report detailing the design and development of a revolutionary gesture-controlled virtual mouse system, a pivotal section demands an exhaustive exploration: the intricate implementation of the gesture recognition algorithm. This critical component serves as the bedrock of the system's functionality, enabling seamless interpretation and response to user hand gestures in real-time.

Algorithmic Framework and Core Functionality

A comprehensive elucidation of the algorithm's underlying framework and core functionality is indispensable. This entails a detailed examination of how the algorithm processes input data from the camera feed, discerns nuanced hand gestures, and orchestrates their translation into precise computer commands, constituting a fundamental aspect of the virtual mouse interface's operation.

Sophisticated Data Preprocessing Strategies

In-depth scrutiny of the sophisticated data preprocessing strategies deployed to enhance the quality and interpretability of input data is imperative. This encompasses a nuanced exploration of advanced techniques such as noise reduction, image normalization, and feature extraction, pivotal for optimizing gesture recognition accuracy and robustness.

Intricate Machine Learning Model Architecture and Training Methodology

A granular exploration of the intricate architecture and training methodology of the machine learning model employed for gesture recognition is warranted. This entails a comprehensive breakdown of the model's structural components, training processes, hyperparameter tuning methodologies, and optimization strategies, offering profound insights into its efficacy and generalizability.

Seamless Integration with Cutting-edge Computer Vision Libraries

A meticulous examination of the seamless integration of the gesture recognition algorithm with state-of-the-art computer vision libraries such as OpenCV and Mediapipe is essential. This involves elucidating the intricacies of customizations, adaptations, and augmentations necessitated to align the algorithm with the specific requirements and constraints of the virtual mouse system.

Elaborate Gesture Mapping and Command Translation Mechanisms

A detailed exposition of the elaborate gesture mapping and command translation mechanisms is imperative for a comprehensive understanding of system functionality. This entails a meticulous delineation of the methodology employed to discern, categorize, and translate diverse hand gestures into actionable commands, encompassing nuanced functionalities such as cursor movement, clicking, dragging, and gesture-based shortcuts.

Robust Real-time Processing Efficiency and Performance Optimization Techniques

A thorough assessment of the algorithm's robust real-time processing efficiency and performance optimization techniques is indispensable. This encompasses a detailed analysis of computational complexity, frame processing speed, resource utilization, parallelization strategies, and hardware acceleration techniques, elucidating the mechanisms employed to ensure fluid and responsive interaction with the virtual mouse interface under varying computational loads and constraints.

Rigorous Testing, Validation, and Performance Benchmarking

An exhaustive scrutiny of the algorithm's performance through rigorous testing, validation, and performance benchmarking procedures is paramount. This involves a comprehensive evaluation of recognition accuracy, false positive/negative rates, latency, robustness to environmental variability, and scalability, substantiating the algorithm's reliability, accuracy, and generalizability across diverse usage scenarios and user demographics.

Forward-looking Insights into Future Directions and Enhancements

A visionary exploration of prospective directions and enhancements for the gesture recognition algorithm is crucial for charting the system's evolution and future trajectory. This encompasses the conceptualization and envisagement of novel methodologies for incorporating additional gestures, enhancing computational efficiency, exploring alternative machine learning paradigms, leveraging multimodal inputs, and fostering seamless integration with emerging technologies, paving the way for continuous innovation and improvement.

By embarking on a comprehensive elucidation of the gesture recognition algorithm's multifaceted implementation, the project report not only showcases the technical prowess underlying the gesture-controlled virtual mouse system but also fosters a profound and nuanced understanding of its intricacies, capabilities, and potential for revolutionizing human-computer interaction paradigms.

5.2 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

Description of Software

The software for the proposed Vision-Based Gesture-Controlled Cursor Navigation System is designed to simulate mouse actions using hand gestures captured by a computer's webcam.

Video Capture Component

This component uses the OpenCV library to access and capture video frames from the webcam. It continuously streams video input.

Hand Detection Component

Using the MediaPipe library, the software identifies and tracks the user's hand(s) within the captured frames. It also detects the positions of the fingertips.

Mouse Control Component

The software emulates mouse actions by controlling the cursor's position and simulating clicks. It can also handle volume control by monitoring the thumb and index finger's proximity.

User Interface

A simple graphical user interface (GUI) might be included to display the video feed from the webcam, and it could provide feedback about the recognized gestures and the region of the computer screen where the mouse control is active.

Exit Mechanism

The software should include a way for the user to exit the system, which is specified as pressing an "EXIT" key.

Testing Plan

Testing is a crucial phase to ensure the functionality and reliability of the proposed virtual mouse control system.

Unit Testing

Test each software component independently to ensure they work as expected. For instance, test video capture, hand detection, gesture recognition, and mouse control separately.

Integration Testing

Combine the software components and verify that they work together seamlessly. Ensure that hand detection and gesture recognition trigger the correct mouse actions.

User Interaction

Testing Test the software with real users to assess the effectiveness and ease of use. Evaluate the system's responsiveness to different hand gestures, including neutral, left-click, right-click, double-click, and volume control.

Boundary Testing

Test the software under various conditions, such as different lighting conditions, hand sizes, and backgrounds. Verify that the system can handle a wide range of hand positions and orientations.

Error Handling and Robustness Testing

Introduce unexpected scenarios, such as hands partially in the frame or gestures that are not well-defined, and ensure the software handles these gracefully without crashing.

Performance Testing

Measure the software's performance, including the frame rate for gesture recognition and cursor movement. Test the system's resource usage (CPU and memory) to ensure it doesn't overload the computer.

Usability Testing

Gather feedback from potential users to assess the software's user-friendliness and identify any areas for improvement.

Compatibility Testing

Ensure that the software works on different operating systems and with various webcam hardware.

Security and Privacy Testing

Ensure that the software does not inadvertently capture or store sensitive information. Verify that the system respects privacy and data protection regulations.

Documentation and Training

Develop comprehensive documentation and training materials to help users understand how to use the system effectively.

Accessibility Testing

Ensure the software can be used by individuals with different levels of dexterity.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 OVERVIEW

Using a virtual mouse, the system's webcam is used for tracking hand gestures, and hand gesture recognition enables users to control the mouse with the help of hand gestures. Gesture recognition is accomplished using computer vision techniques. To collect data from a live video, OpenCV includes a package called video capture. This project can be applied in a variety of real-time applications, such as computer cursor control and smart televisions running Android, among others. The work is so straightforward that it minimizes the use of external hardware in such a way that finger motion in front of a camera will cause the necessary operation on the screen, even though there are tools like mouse and laser remotes for the same purpose. This project demonstrated a real-time hand tracking system that uses markers and a commodity computer with inexpensive cameras. When a calibrated pair of cameras is looking down at the hands with the palms facing downward, the system can specifically track the positions of the index finger and middle finger tips.

6.2 PERFORMANCE IN VARIOUS ENVIRONMENTS

Evaluations of virtual-mouse systems in the scientific community are still a little rudimentary. A cross method evaluation is challenging since there is a dearth of available academic research and open-access datasets. The performance provided by the virtual mouse's fingertip detection will be covered first in this section.

The performance will then be shown under various lighting, backdrop, and distance-tracking settings. The experimental findings for selecting the primary person to direct the mouse pointer and fingertip tracking with numerous persons are then shown. Finally, we evaluate how well our approach compares to earlier virtual-mouse research. The suggested method is based on detecting the amount

of target colors (area of interest) that activates the mouse action based on the gesture generated. Initially, a picture is taken with the hand positioned toward the front of the sensor.

The user then chooses the color cap that will be monitored during gesture creation in order to carry out different mouse actions. When taking a picture of the hand, the color of the cap must be chosen from a variety. The notion of enhancing interaction between humans and computers using machine vision is presented in the suggested all virtual mouse system.

Cross-validation for the AI simulated mouse system can be problematic due to the low 29 amount of datasets available. Hand movements and fingertip identification have been verified in a variety of lighting situations, as well as at varied distances from a webcam for tracking and recognition of hand motions and finger tips. In our project, we can perform various operations by using hand gesture-based cursor movement which depend on the distance between two fingers. Such that

1. we can fix our settings for brightness and sound quality based on the movement of the cursor.
2. On the other hand, we can open a file or folder by using the hand movement-based cursor.

Moving Environment

When the project runs, the cursor is already in moving mode. Based on the user's hand movements, the cursor will move to perform brightness increasing and decreasing operations. When the distance between two fingers is the greatest, the required one is required, and the cursor is said to be in moving mode. Artificial intelligence virtual mouse system camera.

The frames that have been recorded by a laptop's or PC's camera serve as the foundation for the suggested AI virtual mouse system. The webcam will start recording video when the video collection object is built using the programming language's machine vision package OpenCV. The webcam records and sends footage through the AI Virtual Mouse system.

Accuracy: The percentage of accurately classified data samples over all the data is known as accuracy. Accuracy can be calculated by the following equation.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{TN} + \text{FN})$$

Open File Environment

In our program, we already specify a minimum distance between two fingers. When the user fulfills this, it will mean the cursor is now in clickable mode. Now the user can open a file or folder. The computer is programmed to activate the left mouse button if the thumb finger with point ID= 0 and the pointer finger with tip ID =1 are both up and the gap among these two fingers is less than 30 px. The AI virtual mouse performed exceptionally well in terms of precision. The suggested model is unique in that it can execute most mouse tasks and mouse cursor movement using fingertip detection, and it is also useful in operating the PC in the virtual mode like a hardware mouse.

We have tested our AI Virtual Mouse many times and got a good accuracy which is shown in the following table:

Hand Gesture	Time	Accuracy
Cursor Move	99/100	99%
Left Click	95/100	95%
Right Click	95/100	95%
Double Click	90/100	90%
Scroll Up	92/100	92%
Scroll Down	92/100	92%

6.3 SCREENSHOT OF OUTPUTS

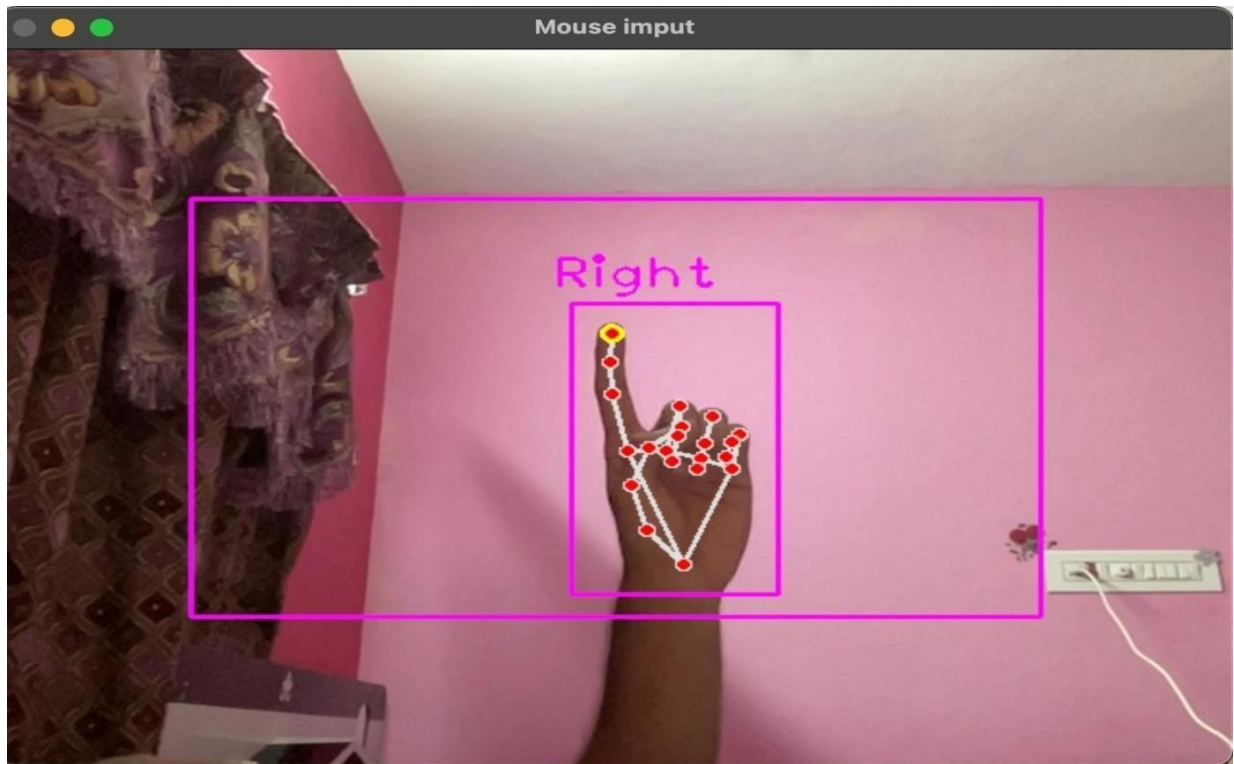


Fig 6.1: Cursor Movement

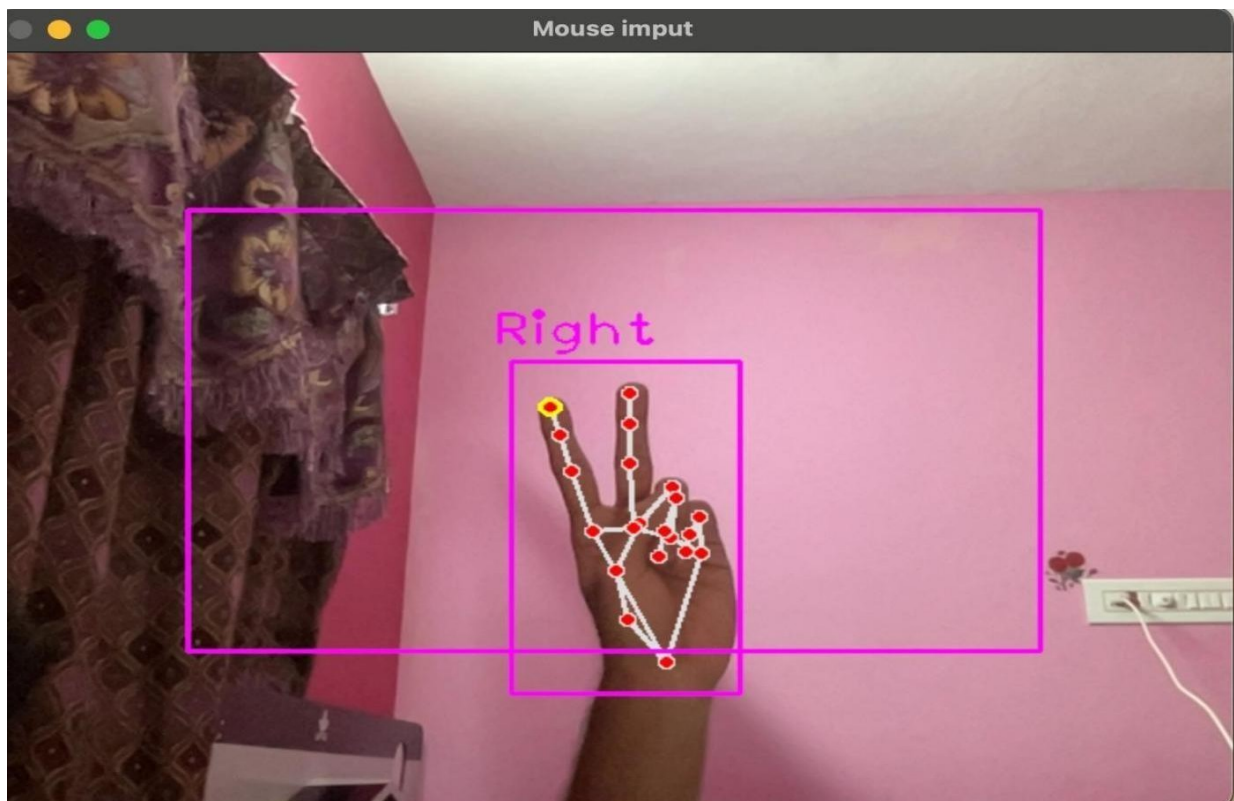


Fig 6.2: Left Click

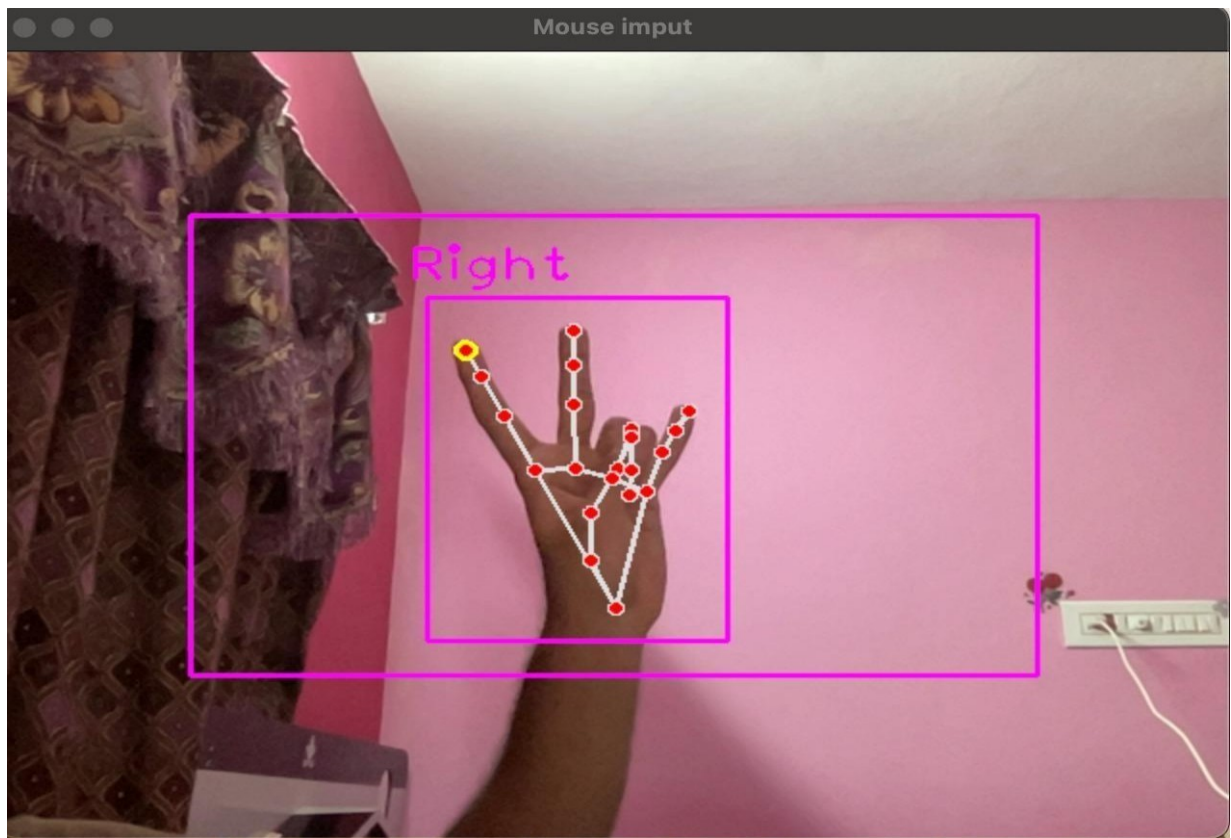


Fig 6.3: Right Click

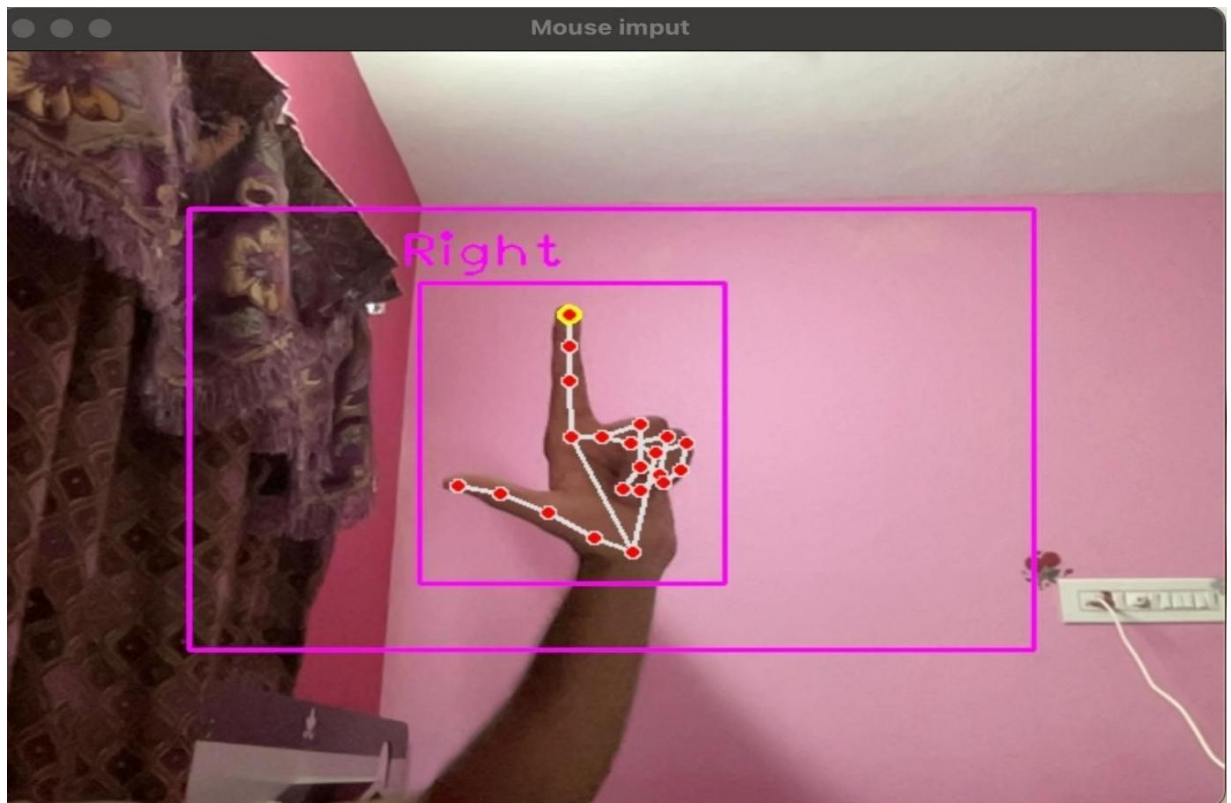


Fig 6.4: Double Click

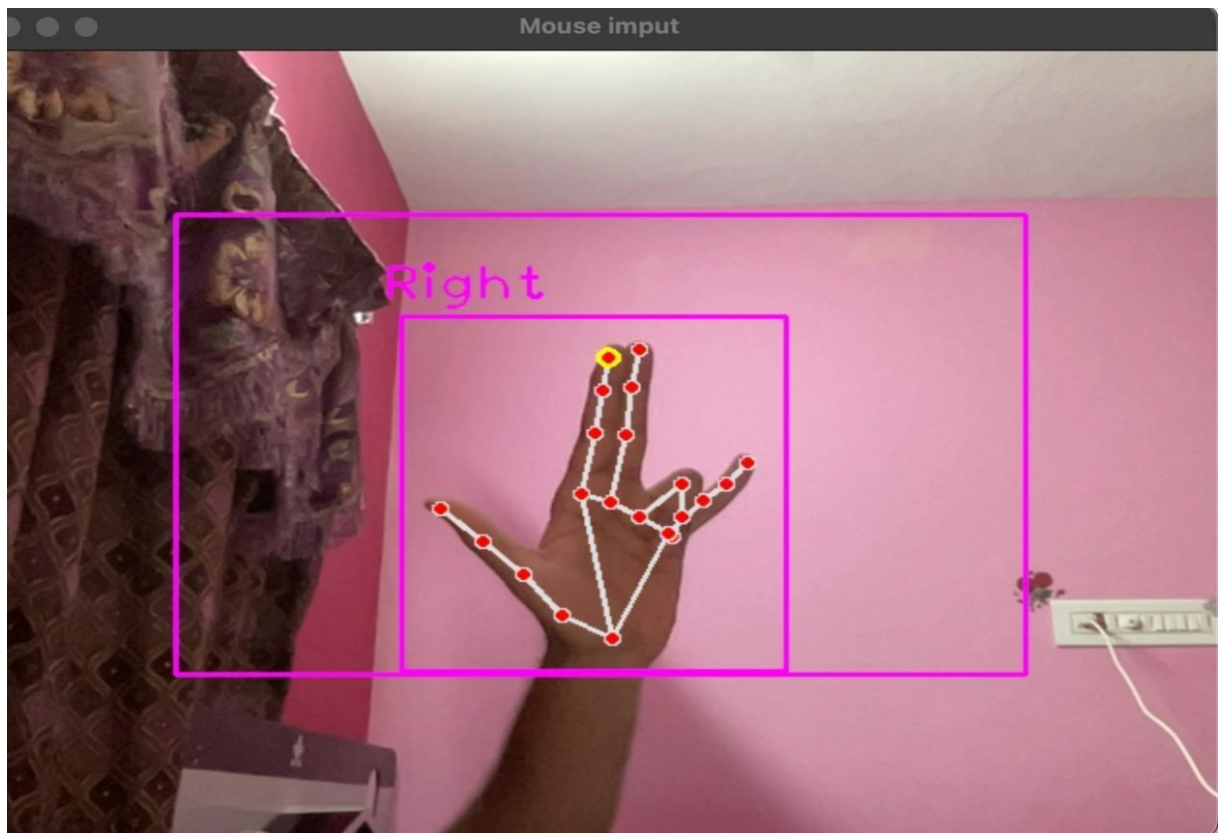


Fig 6.5: Scroll Up

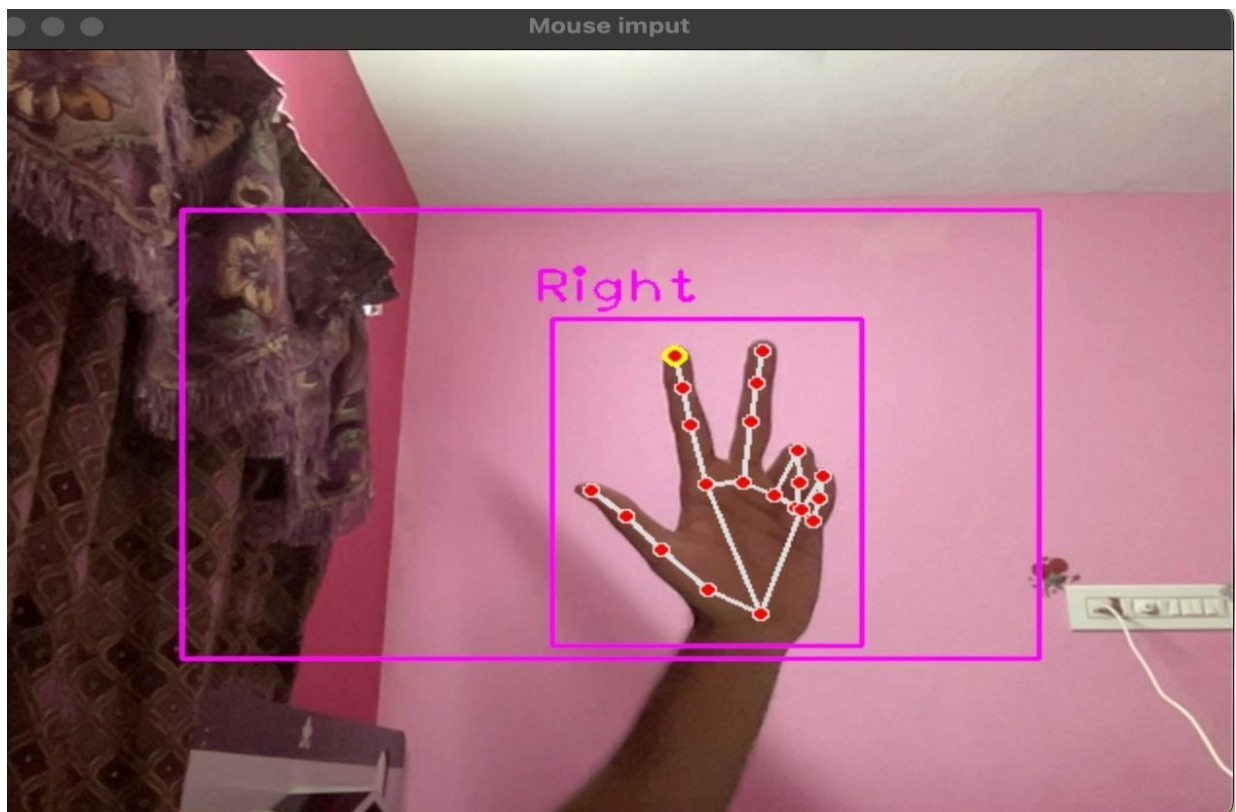


Fig 6.6: Scroll Down

CHAPTER 7

CONCLUSION

The main objective of the AI virtual mouse system is to control the mouse cursor functions by using the hand gestures instead of using a physical mouse. AI virtual mouse system has performed very well and has a greater accuracy compared to the existing models and also the model overcomes most of the limitations of the existing systems. Since the proposed model has greater accuracy, the AI virtual mouse can be used for real-world applications, and also, it can be used to reduce the spread of COVID-19, since the proposed mouse system can be used virtually using hand gestures without using the traditional physical mouse. The model has some limitations such as small decrease in accuracy in right click mouse function and some difficulties in clicking and dragging to select the text. Hence, we will work next to overcome these limitations by improving the fingertip detection algorithm to produce more accurate results.

7.1 SUMMARY

In the Human-Computer Interfaces (HCI) field, where every mouse movement may be done with a fast of your fingertips anywhere, it should come as no surprise that the real mouse will also be overtaken by an immersive non-physical mouse and without regard to the environment, at any moment. In order to replace the common physical mouse without sacrificing precision and efficiency, this project had to design a color recognition program.

This program can recognize color movements and combinations and translate them into functional mouse actions. A few strategies had to be used because accuracy and efficiency are crucial factors in making the application as helpful as a real-world mouse. The primary objective of this method is to lessen and maintain the sensitivity of the cursor by averaging the values of the colors responsible for managing cursor motions based on a set of coordinates movements, as even a small movement could cause unintended cursor movements. In addition, a number of color combinations were developed with the relation of distance computations between the two colors in the combination because a difference in distance can result in a difference in the way the mouse behaves.

This implementation's goal is to make it easier to control the application with minimal trouble. As a result, accurate mouse function triggering can be achieved with little trial-and-error. Moreover, calibrations phase was included to promote effective and versatile color tracking. This enables people to select their preferred colors for various mouse functions as long as the chosen colors don't identical or comparable RGB hues (e.g. blue and sky-blue).

Responsive validations were additionally developed, which essentially enables the software to save various sets of HSV levels across various angles to be utilized during the initialization step. In regards to efficiency and lifestyle, modern technology has made significant progress in improving society's quality of life, as opposed to the other side around.

Hence, cultures must not mix while hesitantly adopting outdated technologies. The latest one is accepting revisions at the IA(HONS) Information System Engineering Department of the Institute of Information and Communication Technology (Perak Campus), UTAR 40. Instead, it is advised that individuals accept modifications to lead a lifestyle that is more effective and productive.

7.2 LIMITATION

There are number of ongoing issues in this research that could impede the outcomes of color recognition. The environment aspect when the recognition phase is taking place is one of the issues. The recognition procedure is very sensitive to brightness levels since extreme intensity or blackness may make it impossible to see the targeted colors in the acquired frames.

In addition, distance is another issue that could have an impact on outcomes of color identification. As the current detecting zone can only allow displays of color within limited radius. Any displays of colors beyond this limitation will be viewed as noise and filtered out.

REFERENCES

- [1] Aashni Hariaa , Archanasri Subramaniana , Nivedhitha Asokkumara , Shristi Poddara and Jyothi S Nayak “Hand Gesture Recognition for Human Computer Interaction” Proc.ICACC(International Conference on Advances in Computing & Communications), 2017 August,pp367–374.
- [2] Abhik Banerjee, Abhirup Ghosh, Koustuvmoni Bharadwaj,” Mouse Control using a Web Camera based on Color Detection”,IJCTT,vol.9, Mar 2019.
- [3] Abhilash S , Lisho Thomas, Naveen Wilson, and Chaithanya “VIRTUAL MOUSE USING HAND GESTURE” Proc. International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056 p-ISSN: 2395-0072,Apr-2018, Volume: 05 Issue:04.
- [4] Alisha Pradhana , B.B.V.L. Deepak “Design of Intangible Interface for Mouseless Computer Handling using Hand Gestures” ICCCV(International Conference on Communication, Computing and Virtualization), 2016, oi: 10.1016/j.procs.2016.03.037.
- [5] Chiung Hsieh, D.-Hua Liou, & D. Lee, , “A real time hand gesture recognition system using motion history image,” Proc. IEEE Int'l Conf. Signal Processing Systems (ICSPS), 2.10.1109/ICSPS.2010.5555462, 2020.
- [6] Pooja Kumari ,Ghaziabad Saurabh Singh, Ghaziabad Vinay and Kr. Pasi “Cursor Control using Hand Gestures” International Journal of Computer Applications (0975 –8887),2016

APPENDIX

A. SOURCE CODE

```
import cv2
import numpy as np
import HandTrackingModule as htm
import time
import autopsy

#####
wCam, hCam = 640, 480
frameR = 100 # Frame Reduction
smoothing = 7
#####

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

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

while True:
    # 1. Find hand Landmarks
    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bbox = detector.findPosition(img)
    # 2. Get the tip of the index and middle fingers
    if len(lmList) != 0:
        x1, y1 = lmList[8][1:]
```



```

x2, y2 = lmList[12][1:]
# print(x1, y1, x2, y2)

# 3. Check which fingers are up
fingers = detector.fingersUp()
# print(fingers)
cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),
(255, 0, 255), 2)
# 4. Only Index Finger : Moving Mode
if fingers[1] == 1 and fingers[2] == 0:
    # 5. Convert Coordinates
    x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
    y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))
    # 6. Smoothen Values
    clocX = plocX + (x3 - plocX) / smoothening
    clocY = plocY + (y3 - plocY) / smoothening

    # 7. Move Mouse
    autopy.mouse.move(wScr - clocX, clocY)
    cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
    plocX, plocY = clocX, clocY

# 8. Both Index and middle fingers are up : Clicking Mode
if fingers[1] == 1 and fingers[2] == 1:
    # 9. Find distance between fingers
    length, img, lineInfo = detector.findDistance(8, 12, img)
    print(length)
    # 10. Click mouse if distance short
    if length < 40:
        cv2.circle(img, (lineInfo[4], lineInfo[5]),
        15, (0, 255, 0), cv2.FILLED)
        autopy.mouse.click()

# 11. Frame Rate

```

```

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)
# 12. Display
cv2.imshow("Image", img)
cv2.waitKey(1)
[5:56 pm, 07/04/2024] Nihal: import cv2
import mediapipe as mp
import time
import math
import numpy as np

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

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

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

        if self.results.multi_hand_landmarks:

```

```

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

    return img

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

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

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

    return self.lmList, bbox

```

```

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

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

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

    # totalFingers = fingers.count(1)

    return fingers

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

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

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

```

```

def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(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

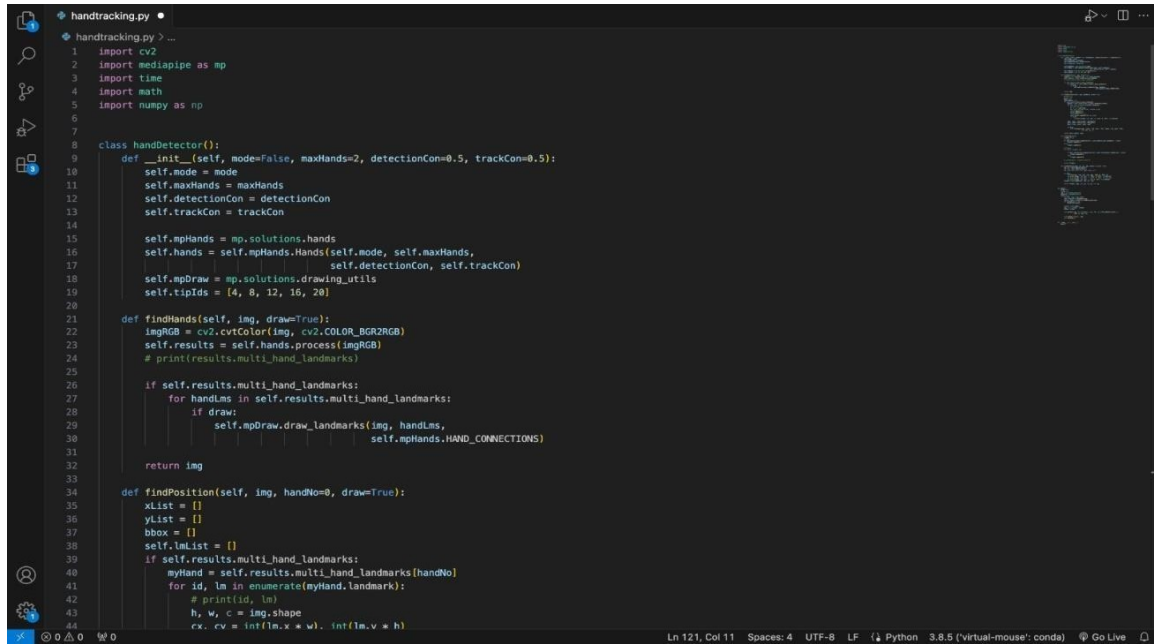
        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

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

if __name__ == "__main__":
    main()

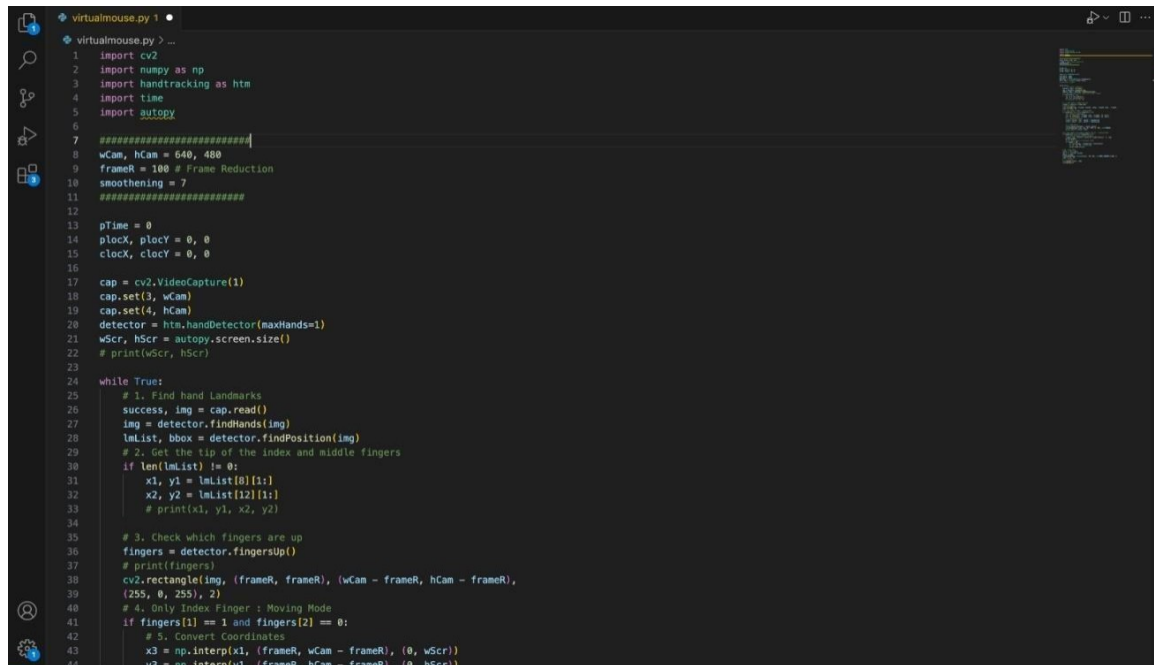
```

B. SCREENSHOTS



```
1 import cv2
2 import mediapipe as mp
3 import time
4 import math
5 import numpy as np
6
7 class HandDetector():
8     def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
9         self.mode = mode
10        self.maxHands = maxHands
11        self.detectionCon = detectionCon
12        self.trackCon = trackCon
13
14        self.mpHands = mp.solutions.hands
15        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
16                                       self.detectionCon, self.trackCon)
17        self.mpDraw = mp.solutions.drawing_utils
18        self.tipIds = [4, 8, 12, 16, 20]
19
20    def findHands(self, img, draw=True):
21        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
22        self.results = self.hands.process(imgRGB)
23        # print(results.multi_hand_landmarks)
24
25        if self.results.multi_hand_landmarks:
26            for handLms in self.results.multi_hand_landmarks:
27                if draw:
28                    self.mpDraw.draw_landmarks(img, handLms,
29                                              self.mpHands.HAND_CONNECTIONS)
30
31        return img
32
33    def findPosition(self, img, handNo=0, draw=True):
34        xList = []
35        yList = []
36        bbox = []
37        self.lmList = []
38        if self.results.multi_hand_landmarks:
39            myHand = self.results.multi_hand_landmarks[handNo]
40            for id, lm in enumerate(myHand.landmark):
41                # print(id, lm)
42                h, w, c = img.shape
43                cx, cy = int(lm.x * w), int(lm.y * h)
```

Fig. B.1: Hand Tracking



```
1 import cv2
2 import numpy as np
3 import handtracking as htm
4 import time
5 import autopy
6
7 #####
8 wCam, hCam = 640, 480
9 frameR = 100 # Frame Reduction
10 smoothening = 7
11 #####
12
13 pTime = 0
14 plocX, plocY = 0, 0
15 clocX, clocY = 0, 0
16
17 cap = cv2.VideoCapture(1)
18 cap.set(3, wCam)
19 cap.set(4, hCam)
20 detector = htm.HandDetector(maxHands=1)
21 wScr, hScr = autopy.screen.size()
22 # print(wScr, hScr)
23
24 while True:
25     # 1. Find hand Landmarks
26     success, img = cap.read()
27     img = detector.findHands(img)
28     lmList, bbox = detector.findPosition(img)
29     # 2. Get the tip of the index and middle fingers
30     if len(lmList) != 0:
31         x1, y1 = lmList[8][1:]
32         x2, y2 = lmList[12][1:]
33         # print(x1, y1, x2, y2)
34
35     # 3. Check which fingers are up
36     fingers = detector.fingersUp()
37     # print(fingers)
38     cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),
39                  (255, 0, 255), 2)
40     # 4. Only Index Finger : Moving Mode
41     if fingers[1] == 1 and fingers[2] == 0:
42         # 5. Convert Coordinates
43         x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
44         y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))
```

Fig. B.2: Virtual Mouse

