GESTURE CONTROL VIRTUAL MOUSE USING OPENCV, ML MODELS AND GESTURE RECOGNITION LIBRARIES

A MAJOR PROJECT REPORT

SUBMITTED BY

DHRUV DHAR [RegNo: RA2111003011108]
SYED FAIZAN HAIDAR[RegNo:RA2111003011583]

Under the Guidance of

DR.D.SHINY IRENE

(Assistant Professor, Department of Computing Technology)

in partial fulfillment of the requirementsfor the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in (SPECIALIZATION NAME)

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY

KATTANKULATHUR- 603 203

OCTOBER 2024

Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course	: B. Tech / Computer Science and Engineering
Student Name	**: DHRUV DHAR and SYED FAIZAN HAIDAR**
Registration Number	: RA2111003011108 and RA2111003011583

Title of Work : GESTURE CONTROL VIRTUAL MOUSE USING OPENCV, ML MODELS AND GESTURE RECOGNITION LIBRARIES

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

DHRUV DHAR and SYED FAIZAN HAIDAR
RA2111003011108 and RA2111003011583

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP107L - Minor Project [18CSP108L- Internship] report titled "**GESTURE CONTROL VIRTUAL MOUSE USING OPENCV, ML MODELS AND GESTURE RECOGNITION LIBRARIES**" is the bonafide work of DHRUV DHAR [RA2111003011108] and SYED FAIZAN HAIDAR [RA2111003011583]**"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Selvin Paul peter

Panel Head
Assistant Professor,
Department of
Computing Technology

SIGNATURE

Dr. G. NIRANJANA

PROFESSOR &HEAD
DEPARTMENT OF
COMPUTATIONAL INTELLIGENCE

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM InstituteofScience and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. T. V. Gopal ,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Com puting, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to , **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C.Lakshmi,** Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. R. Annie Uthra**, Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our Project Coordinators, Dr. Selvin Paul peter Panel Head, Dr. Selvin Paul peter and Panel Members, Department of Computational Intelligence, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor, Dr. Kishore Anthuvan Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, Dr. Selvin Paul peter Department of Computational Intelligence, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his / her mentorship. He / She provided us with the freedom and support to explore the research topics of our interest. His / Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computational Intelligence, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

Authors

ABSTRACT

The integration of human-computer interaction (HCI) with machine learning has opened new avenues in the creation of hands-free control systems, including gesture-based virtual mice. This approach enhances user experiences by allowing interactions without physical contact with hardware, which is especially valuable in specific scenarios like public installations, healthcare environments, and situations where touchless interaction is preferred. Gesture control virtual mouse systems use computer vision, machine learning (ML) models, and specialized libraries to detect and interpret hand gestures, translating them into mouse actions such as clicking, scrolling, and dragging. This abstract provides an overview of the gesture-controlled virtual mouse project, explaining the underlying technologies, objectives, techniques, and potential applications.

The gesture-controlled virtual mouse utilizes OpenCV, a popular computer vision library, to capture and process images from a camera feed, usually from a webcam. OpenCV enables real-time image processing, making it suitable for capturing gestures and providing fast feedback. The primary objective of this project is to interpret a user's hand gestures as control signals for mouse actions. The system accomplishes this by analyzing hand shapes, finger movements, and hand positions, which are translated into actions that simulate a traditional mouse's functions. The proposed system aims to provide an accurate, low-latency, and user-friendly alternative to traditional input devices, using gestures alone for cursor movements and actions.

To build this system, a key aspect is detecting and recognizing hand gestures accurately. This is achieved by combining image processing techniques in OpenCV with ML models designed for gesture recognition. The image processing phase begins with capturing frames from the webcam and processing them to isolate hand regions. Techniques such as color segmentation, background subtraction, or skin detection are often applied to distinguish the hand from the background. Once the hand region is detected, features like contour and convex hull are extracted, which provide useful information about the shape and orientation of the hand and fingers.

Machine learning models are then employed to recognize different gestures by analyzing these features. Convolutional Neural Networks (CNNs) are commonly used because of their strength in handling image data and identifying complex patterns within images. CNNs are trained on a labeled dataset of hand gestures, allowing them to classify gestures accurately. The training data for these models typically consists of a large number of hand images in different poses, with variations in lighting, background, and angle to ensure the model's robustness. In some implementations, transfer learning can be used to improve model performance with fewer data or training time by leveraging pre-trained models on similar datasets.

In addition to CNNs, other ML techniques such as Support Vector Machines (SVM) or k-Nearest Neighbors (k-NN) can be used to classify gestures, depending on the project's complexity and resources. These models are trained to recognize key gestures such as an open palm for cursor

movement, a fist for mouse click, and specific finger positions for scrolling or dragging. Gesture recognition libraries, such as MediaPipe or TensorFlow.js, can also be integrated with OpenCV to streamline the gesture detection and classification processes, reducing the need for developing custom gesture recognition models from scratch.

Once gestures are detected and classified, the corresponding mouse action is simulated using operating system libraries or APIs, such as PyAutoGUI in Python. PyAutoGUI provides functions to control the cursor position, simulate mouse clicks, and perform scroll actions programmatically, making it suitable for the gesture-controlled virtual mouse application. For example, an open palm gesture may correspond to moving the cursor based on the hand's position in the frame, while a clenched fist could signal a left mouse click. The system continuously monitors the user's hand movements, updating the cursor position and actions in real-time.

To improve the system's performance, various optimizations can be implemented. For instance, filtering techniques such as Gaussian or median filtering can reduce noise in the image, making hand detection more accurate. Thresholding methods, like Otsu's thresholding, can further enhance segmentation by creating a clear distinction between the hand and background. Additionally, calibration techniques are essential to adapt the system to different lighting conditions and skin tones, improving the robustness of hand detection. Customizing the sensitivity of gesture recognition is another important consideration, as overly sensitive systems may trigger unintended actions, while less sensitive systems may miss intentional gestures.

The challenges in creating a gesture-controlled virtual mouse include handling variations in lighting, background clutter, and differences in hand appearance, such as skin color and size. To address these, the system can incorporate dynamic background subtraction techniques and adaptive skin detection algorithms that adjust to various environmental factors. Moreover, by training the ML models on diverse datasets, the system can achieve greater accuracy and generalization, performing well across different users and settings.

In testing the system, evaluation metrics such as accuracy, latency, and user satisfaction are used to assess its effectiveness. Accuracy refers to how reliably the system can recognize gestures and translate them into the intended mouse actions. Latency is crucial in real-time applications, as delays in gesture recognition or cursor movement can hinder user experience. User satisfaction is also measured through feedback from individuals interacting with the system, providing insights into usability and areas for improvement.

Applications of gesture-controlled virtual mouse systems are vast, spanning multiple industries. In healthcare, touchless interaction can help maintain hygiene standards, allowing healthcare professionals to interact with medical imaging or patient data without physical contact. In public installations, gesture control can offer an intuitive interface for kiosks and information systems, minimizing the need for users to touch screens frequently handled by others. The technology is also beneficial in accessibility solutions for individuals with mobility impairments, as gesture control provides an alternative to traditional input devices.

TABLE OF CONTENTS

INTRODUCTION

Intelligent machines' appearance and continued technical developments have allowed the creation of more and more compact and complex technologies that are vital for our everyday lives. While we go through a digital world that is becoming more and more digital there is a great demand for new and more user-friendly user interfaces. Machine learning, computer vision, and gesture recognition technologies are the arms of the "Gesture-Based Virtual Mouse" project, whose concept is aimed at enabling a new approach for how people approach computers. The idea is to design a gesture-controlled interface that will enable the user to walk around the virtual space by doing simple hand gestures which are recorded by a webcam. In comparison to common input devices such as mice or touchpads, this technology is much more accessible and natural because it gives the user an easy way to communicate with computers, which is especially beneficial for those who have mobility impairments or physical disabilities.

Computer vision is an important part of this project as it brings about the system's capacity to understand and interpret the hand gestures in real-time.

Through complex algorithms, the webcam sees hand actions drawn from one image to the other where a gesture can be detected and materialized into an activity measured by motion recognition techniques. Thus, users could execute various tasks such as cursor movement, clicking, scrolling, and dragging with hand gestures, which greatly boost the user ability and accessibility.

The addition of machine learning algorithms also provides the power for the system to perceive and adapt to users' maneuvers contributed over the period of use, bringing about machine training. Extracting vast datasets of hand movements, the system can increase its reliability and response time and thereby perform successful programming through natural conversation with computers.

Furthermore, machine learning algorithms give a chance to a device to recognize various gestures and assign related commands, thus making the interface an all-around and user-friendly.

Gesture recognition algorithms are the main element in this innovative project, as they with the accurate identification and classification of hand gestures in real-time.

The computer is the camera-mining data scientist who gives it the leverage to analyze video inputs and observe specific patterns and movements, hence, the user is able to communicate ideas.

However, continuous refining and optimization of gesture recognition algorithms will be the main source in providing seamless and input-based human interface which will lead to making computing more inclusive for every user regardless of age and disability.

Moreover, the significance of user feedback and user acceptance handling are highlighted in the project to guarantee the fact that the gesture-controlled interface is well- accepted and suited to the various user groups. Providing data points to the project team, who can then use them for identification purposes, is the best way to identify the weak points in the system and work on specific development stages together with users. This type of management not only brings about the most functional interface but also creates a situation where users of all levels of ability and technical competence feel included and empowered.

In the first place, in personal computing, this project of a Gesture-Controlled Virtual Mouse is intended to cause a big splash in utilities and fields of work. For example, in hospital areas, where conventional input devices might not be necessary or clean, gesture control will allow healthcare workers to use no-touch procedures to search for and edit medical file systems.

Also in the educational environment, gesture-based interfaces can promote interactive learning that allows students to explore digital content in a more immersive and intuitive way. By analyzing these multiple kinds of applications, the project intends to be an all-away stick-it-in with technology, hence, get the highest optimization of its own and impact most students.

1.1 Objectives

This report aims to explore the foundational concepts and applications of gesture recognition. It aims to investigate these technologies through a closer look at the digital models used to analyze human gestures, e.g. deep learning and pattern recognition, and determine the technology's suitability for virtual mouse use. This objective glazed is a comprehensive review of the existing methods, including the penetration and restrictions of gesture recognition techniques.
The purpose of this project is to invent a system of virtual mouse, by which the traditional functions of a mouse are executed through different hand gestures. The main work includes specifying a list of gestures (such as swipe-move or point-click) and designing an interface which allows the users to navigate, scroll, and select options in a manner of user-friendliness. The objective is to guarantee a highly intuitive and sensitive virtual mouse system that makes possible to give accurate gesture-based commands which imitate the functions of the physical mouse.

Thus, it is important to identify and discuss challenges in implementing gesture recognition systems. Factors involved include how gesture variability, environmental interference which includes lighting, and requiring adaptable calibration may affect overall system performance. Future advancement strategies such as strengthening machine learning algorithms or how to improve the gesture tracking method might be proposed to support continual development in gesture control systems.

1.2 Scope

The scope of this project reaches the very fringe of technological advancement envisioning a world whereby human-computer interaction surpassed traditional input methods. By using the state-of-the-art technology that includes computer vision and machine learning Gesture recognition is a new paradigm shift in the design of user interfaces, and this project pioneers the concept aspires to frictionless, intuitive computing so that users can easily navigation of computer environments through natural hand movements.

Additionally, this project, through vast and extensive research, development, and innovation, became successful at the same time, striving to create the basis for the next generation of human-computer interaction.

The vision it goes beyond mere function to include a deeper understanding of user behavior and a customised and adaptive interface as per the taste of the users. They should be able to predict and respond to end individual needs with the purpose of this ambitious goal in mind, this project addresses the interface design from a holistic perspective so as to not only relate with technical aspects related to interface design but at the same time consider their implications at broader and deeper socio-cultural levels and their consequences when technology would be adopted.

This has been achieved by instigating collaboration and facilitating dialogue among technologists and designers, psychologists, along with end-users, ensuring that the resulting interfaces are not only technically viable but also socially responsible, along with ethical sound.

Moreover, this project considers scalability and interoperability in its dream of ubiquitous computing. The development of open standards and protocols helps the project to easily connect with other existing systems and platforms, accelerating the gestural interface's acceptance across the wide spectra of applications and industries.

Ultimately, it is this project's success dependent not only on technology but innovation but also the potential to inspire and empower the individuals to adopt new means of interacting with technology. Through education, outreach, and advocacy work, the project hopes to create a culture that supports innovation and experimentation as well as encouraging users to explore the full gesture-controlled interfaces' potential while making contributions to their continuing evolution and refinement.

1.3 Real-World Applications of Gesture-Controlled Virtual Mouse

Gesture-controlled virtual mouse considerably contribute to interaction innovations in many sectors by allowing the hands-free operation, especially ideal for clean, accessible, and immersion-intensive environments. The system interprets hand motions as commands, making it a contactless alternative to traditional input devices.

The gesture-controlled mouse are highly beneficial in healthcare settings. They improved workflow efficiency, reduced the risk of contamination, and made them more accessible.

Preventing contamination in the sterile environment of the operating room is of paramount importance. Surgeons, during a procedure, require access to patient records, medical images, or even 3D anatomical models. Gesture controls will allow surgeons to control digital information without touching it-touching the field with elbows, zooming into scans, or rotating models-to keep the field sterile and enhance workflow.

This feature, therefore has in the gaming industry resulted in an interesting and interactive game. The aspect of gestures also made games realistic in their expression.

The alternative to the conventional gaming controllers is gesture-based controls, which make the game accessible to people with physical limitations. The technology allows the use of hand gestures in interaction with games, thus making gaming more accessible.

Business, educational, and public speaking presentations using gesture-controlled mouse facilitate contactless presentation management in a manner that the audiences are drawn to such a presentation. The use of gestures provides presenters the ability to navigate through slides, zoom into charts, highlight particular points, and control any multimedia material with the assistance of only hand gestures instead of needing an actual hardware device.

Such a presentation setup lets presenters attend to their audiences and presents a rather technical touch towards the presentations. Gesture controls enable teachers to make their lessons interactive by navigating through slides, drawing attention to certain parts, and annotating the content. For remote learning, this means virtual instruction can be made more engaging and will not lose the focus of the students.

At conferences, the presenter can navigate through the presentation materials hands-free. This way, they can move around freely on stage and respond more naturally to the audience, thus enhancing delivery and audience interaction.

Gesture-controlled virtual mouse allow people who have physical disabilities to computerize and control home appliances, etc. The usage of gesture controls by people suffering from reduced mobility allows access to the web, reading emails, and making applications using software without using a real mouse. In a way, they become more independent.

SPRINT PLANNING AND EXECUTION

2.1 Sprint 1

2.1.1 Sprint Goal with User Stories of Sprint 1

The Sprint Goal defines high-level objectives guiding the team on what is being implemented at the sprint. At our project, it has created an initial prototype so a person can use gestures over a hand to guide around this virtual mouse to execute elementary features necessary for such proofs of concept as valid demos of user interactions with your application.

User Stories

Story 1: First Gesture Recognition Setup

Objective: This story is to let the system recognize the hand gestures that are to be used as inputs to the virtual mouse.

User Requirement: Users wish to interact with the virtual mouse using natural hand movements.

Expected Result:The system is expected to identify three simple gestures, namely, open hand, closed fist, and pointing, with an accuracy of 80%.

Importance: This is a basic step in order to ensure that all other functionalities are based on proper gesture recognition.

Story 2: Gesture to Move Cursor

Objective: The purpose of this user story is to enable the virtual mouse cursor to move with the user's hand movement.

User Requirement: Users need to experience it smoothly, wherein the movement of hands directly corresponds to cursor movement on the screen.

Expected Result: Ideally, cursor should have been responsive to the maximum level with very minimal response time delay such that if possible less than 100 milliseconds where the user does not perceive that there's a lagging or buffering.

It would be imperative to keep such a usability story that enables smooth usability of a cursor.

Story 3: Detection of Single Click

Objective: Provide the possibility of usability clicking on the mouse through gestures.

User Requirement: It should allow users to click items on the screen by using it without actually pressing a mouse.

The system should accept a closed fist as one click and must not allow multiple accidental clicks.

It allows interaction with this virtual mouse, thus making it more useful.

2.1.2 Functional Document

2.1.2.1. Introduction

This document is intended to present the functional requirements of the Gesture Control Virtual Mouse project. This will allow users to control virtual mice through hand gestures with an enhanced user experience and interactive interface with digital devices.

2.1.2.2. Scope

The project will present designing a minimal viable product for hand gesture recognition in terms of providing functionalities including simulating a mouse movement and clicking of it. Subsequent builds would encompass more than that, but such a note does not refer to all features included during the first iteration sprint.

Users

- Project Management team
- Developers
- Testers of Quality Assurance
- UX/UI Design
- Key Stakeholders

2.1.2.3. Functional Requirements

User Stories

User Story 1: Configuration of Initial Gesture Recognition

For the system, one will require the following function which can identify at least three basic hand gestures by this system: open, close fist, and the finger pointing.

Acceptance criteria :Gesture recognition must be more than 80%.

After gesture recognition, the system must give instantaneous feedback.

User Story 2: Gesture-Based Moving the Mouse Cursor

Requirements: Based on the hand movement, the virtual cursor must be moved.

Acceptance Criteria: The motion of the cursor must resemble hand motion in real time but should have a time delay less than 100 ms.

It should be in motion proportionate to and according to the position of the hand of the user.

User Story 3: Detection of Single Click

Requirement: Simulate single mouse click based on closed fist as a hand gesture.

Acceptance Criteria: The registered closed fist gesture should allow for only a single click without a spurious double click Feedback about registered click to be returned.

2.1.3 Architecture Document

2.1.3.1. Introduction

The objective of this architecture document is to explain the system's architectural design of a Gesture Control Virtual Mouse. These details will encompass the composing components and their interactions along with the technology utilized in building these systems.

This architectural document outlines the architecture design for the first sprint of a Gesture Control Virtual Mouse product, which are the major components intended to identify hand gestures so as to replicate mouse interaction.

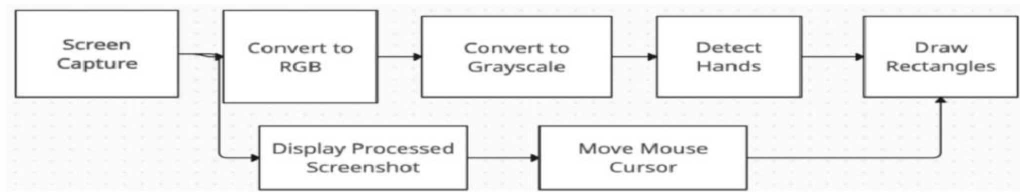
Users

- System Architects
- Developers
- Quality Assurance Engineers
- Project Managers
- Technical Stakeholders

2.1.3.2. System Overview

The Gesture Control Virtual Mouse system controls a virtual mouse using gestures of hands. It will be a series of sub-components, interlinked through each other and work seamlessly to capture the user inputs, process gestures, and control the virtual mouse cursor.

2.1.3.3. System Architecture Diagram



2.1.3.4. Module Description

- Input Module

Description: Capture live video feed of the camera and capture real-time video for hand-gesture detection.

Technologies:

Camera: Any general webcam or even an infrared camera.

Libraries: OpenCV is used to process images, as well as capture the video stream.

- Gesture Recognition Module

Description: Analyze video feeds of the captured input video feeds to identify precise hand-gesture recognitions.

Technologies:

Machine Learning Framework: TensorFlow or PyTorch to train and make inference of the gesture recognition model.

Algorithms: CNN for hand position and gesture classification.

- Output Module

Description: The module that controls the virtual cursor and simulates mouse events as a result of the recognized gestures.

Technologies:

Libraries: Use PyAutoGUI or its equivalent to create simulated mouse events.

Programming Language: Python or JavaScript.

- Calibration Module

Description: It should allow the users to calibrate the gesture recognition system to their environment for higher accuracy.

Technologies:

User Interface: It should select a GUI framework for calibration such as Tkinter for Python or Electron for JavaScript

Settings Storage: It should use either local storage or a configuration file for saving user preference.

- User Interface Module

Description: It should provide a graphic interface by which the users interact with the system, as well as get their feedback.

Technologies:

Frameworks: It should select Tkinter if using Python, Electron for JavaScript or similar

Design: UI is simple showing virtual mouse and gesture feedback.

- Data Flow

Description Data Flow

User Input : Capture of video input of hand of the user through the camera.

Gesture Processing: In a video stream, Recognition module of gesture carries out its processing in order to check whether there is any presence of gesture or not.

Cursor Control : Output Module takes the recognized gestures then interprets them in actions of the cursor and control the movement of the cursors.

Feedback: At identified gestures and resultant movements through visual feedback, through the help of User Interface Module.

Calibration: The settings within the Calibration Module could be modified by the users, affecting how gestures would be recognized and how these are interpreted by the system.

2.1.3.5. System Interaction

- External Interfaces

Camera Interface: the system will interact with the webcam or infrared camera as it records video.

User Input Devices: the system will respond to hand gesture that simulate classical mouse moves.

- Communication Protocols

Data Transfer: Local communications between modules through function calls or shared memory.

2.1.3.6. Security

Private Data: Processing Video Data in real-time without any storage of user video

User Permission: Requesting any permission required to access camera and other resources, etc.

2.1.3.7 Scalability

The architecture should be extensible to support future gestures and functionalities, such as multi-finger gestures, voice commands, and other applications.

2.1.3.8 Functional Test Cases

➢ User Story 1: Basic Gesture Recognition Setup

- Test Case 1.1: Test Basic Gesture Recognition
- Description: Validate the system accepts basic gestures Open Hand, Closed Fist, Pointing.
- Preconditions: The application has been launched and camera feed is on.
- Steps:

1. Perfomed "Open Hand" gesture in front of the camera.
2. Observed the display for the recognized gesture.
3. Performed "Closed Fist" gesture.
4. Observed the display for the recognized gesture.
5. Performed "Pointing" gesture.
6. Observed the display for the recognized gesture.

- Expected Output: The system must display "Open Hand," "Closed Fist," and "Pointing" for each of the gestures without any error.

➢ User Story 2: Move Mouse by Gesture

- Test Case 2.1: Verify Move Mouse using Hand Position
- Description: Verify that the cursor moves with hand position.
- Preconditions: Application Opened, Gesture detection shall be active.

- Steps

1. Position your hand at center of camera feed
2. On the left side of the camera feed
3. Verify movement of cursor on screen
4. Position your hand on right side of camera feed
5. Verify movement of cursor on screen.

- Expected Result: The cursor shall move in a smooth curve in the direction of hand movement.

➢ User Story 3: Single Click Recognition

- Test Case 3.1: Single Click Recognition Test
- Explanation: The system shall detect a click when the "Closed Fist" gesture has been performed.
- Precondition: The application has started, gesture recognition is ON.
- Steps:

1. Perform "Closed Fist" gesture
2. Observe the application whether a click action has been detected.
3. Expected Result The application opens a context menu or selects an item as if a mouse click has occurred.

2.1.3.9 Daily Call Progress

Date: 01 August 2024

Activities or components of the project discussed**:** Discussion for Zeroth review.

Date: 07 August 2024

Activities or components of the project discussed: Discussion to de done for the project after zeroth view.

Date: 14 August 2024

Activities or components of the project discussed: Collected data for the for the project
from different sources on the internet.

Date: 23 August 2024

Activities or components of the project discussed: Started working on backend code and adding data set in the code.

2.2 Sprint 2

2.2.1 Sprint Goal with User Stories of Sprint 2

The aim of Sprint 2 is to implement sophisticated functionalities based on gestures, to enhance the effectiveness and speed of gesture recognition and gestures control and to foster the development of user worry free interface by way of improving responses and advanced personalization features. By that time of the sprint, the users should be able to perform any of the important actions of a mouse i.e. right clicks, double taps, drag and drops, scrolling with ease and assurance.

User stories:

User Story 1: As a user, I want to use a gesture ables me to right click without using a mouse in order to interact with applications' context menus.

Acceptance Criteria: Combination of two fingers drawn or some other designated gesture is treated as a mouse right click.

Right-click action is performed by the virtual mouse 0.5 - 1 sec after the conduct of the action.

Users are informed by animation around cursor or change of color of cursor to show that right-click action is acknowledged.

Tasks:

Detect a two-finger dimensions tap inward.

Control this gesture to enable right mouse click action.

Incorporate visual indicator wellbeing of gesture acknowledgement to users.

Evaluate the precision of detection for various hand orientations and illumination levels.

User Story 2: As a user, I want to use a gesture that corresponds to a double click to open a file or folder and to select more than one object at a time.

Acceptance Criteria: A quick double tap motion is accepted as a double click by the system.

The system reliably distinguishes between a single tap and a double tap.

There is some form of feedback like a non-invasive cursor rotation which indicates that a double click has been executed.

Tasks:

Enable the system to recognize double tap gestures as double click actions.

Modify the timing to enhance the performance of single and double taps.

Check for the accuracy of the recognition and reduce the chances of false recognition.

Fine-tune the sensitivity as per the user feedback for improved user experience.

User Story 3: As a user, I would like to use gestures to drag and drop objects, so that I can arrange objects without having to use a mouse.

Acceptance Criteria: A drag can be initiated by holding down a certain gesture, for example a pinch.

Gesturing and holding the gesture enables one to smoothly drag the object in one's hand.

The object is released in another position upon the opening of one's hand.

The interface gives some guidance on how to perform drag-and-drop operations.

Tasks:

Create pinch or similar gesture to indicate drag initiation.

Synchronize the hand movement to corresponding every movement of the animated object within the screen.

Integrate some feedback, for instance, the cursor turning a different color, to indicate that dragging mode has been activated.

Evaluate how continuous the motion of the feature is working, in particular, for any jerky linear movement of the fingers or hand or when rapid gestures are made.

2.2.2 Functional Document

2.2.2.1 Introduction

In the Gesture-Controlled Virtual Mouse project developed cursor control and mouse handling is carried out through hand gestures, captured using a webcam. This system enables users to operate a computer without touching the mouse or keyboard, thus improving the level of interaction with the device and providing a new dimension of experience.

2.2.2.2 Scope

This document describes the features that will be implemented in the second sprint of the project, concerning gesture control for basic operations such as right-click, double-click, drag and drop, scroll and their speed and feedback control. It contains system requirements, user interface design and interaction scenarios.

2.2.2.3 Functional requirements

- .Right-click gesture

Function: Allows the user to perform a right-click action using a specific gesture.

Description: The system will recognize the two-finger tap (or any other predefined gesture) as a right-click operation enabling the user to display context menus.

User Interaction:

User draws in the right hand within the cameras' view and performs the right-click gesture.

The system detects the gesture and invokes a right click on the display where the cursor is positioned.

Visual Feedback: On a successful recognition of the right-click gesture, the die icon changes or a short animation is played to signify the action.

- Double Click Gesture

Function: Enables the user perform a double click action so as to open files, folders or highlight more than one item.

Description: A double tap gesture will also be interpreted as a double click gesture.

User Interaction:

The user points to a target with the cursor, and then executes the double-tap gesture.

The system understands the gesture and performs a double click where the cursor is.

Visual Feedback: The crosshair might rapidly enlarge or blink, suggesting a double click operation.

- Gesture of Drag and Drop

Purpose: Enables one to drag and move any object placed on the screen by making a hopefully swift motion with the given hand.

History: Drag and drop function is set into motion using a pinch gesture where the pinch is retained to shift an object and released to drop it.

User Interaction:

Users make a pinch gesture to enter drag mode.

Dragging an item along is done with the help of re pinching and moving the hand.

A pin drop gesture ends the drag operation and drops the item in the position held.

Visual Feedback: In drag mode, a ‘grab’ icon is shown on the cursor, and the icon returns to normal after the release.

2.2.3 Architecture Document

2.2.3.1 Introduction

The present document provides the design of the proposed structure for the project Gesture Controlled Virtual Mouse, which is concerned with controlling the cursor movements and mouse actions such as right and double click, drag and drop, scroll among others using hand gestures. The concerned architecture is planned to be modular, flexible and easily extendable with regards to enhancement and accommodating more gestures.

2.2.3.2 Interaction with the System

Gesture Controlled Virtual Mouse makes use of a web cam to obtain a video feed, processes the feed for a person’s hand and gestures and applies the gestures for the virtual mouse. The system is created with python as the main programming language with libraries such as OpenCV for images manipulation, Mediapipe for hand tracking and other files for gestures recognition and controlling the virtual cursor respectively.

2.2.3.3 Architectural Design

- System Components
  - Camera Interface

Purpose: Provides the system with the ability to recognize the gestures by streaming the video in real time.

Description: It interfaces with the hardware such as a webcam to acquire images in turns before the images are processed.

Tools/Technologies: OpenCV as it provides a function to capture the images in a sequence.

  - Image Processing Module

Purpose: Identifying and segmenting hand gestures from the video frame by frame.

Description: Turn the images into black and white images before applying some filters and thresholding to increase the precision achieved, and then the images are set for recognition of the hand gestures.

Tools/Technologies: OpenCV, image filters, contour detection.

  - Hand Tracking and Gesture Recognition Module

Purpose: Hand landmarks are tracked, gestures are recognized and converted to commands.

Description: Mediapipe is used to find hand landmarks such as finger tips and finger joints. Hand gestures are recognized from landmark position along with the movement coordinates.

Tools/Technologies: Mediapipe, writing own algorithms for squeezing, tics and stricks, drag and drop, and reader scrolling gestures.

  - Mapping of Gesture to the Action

Purpose: This program links inventorized gestures with their associated mouse operations like moving the pointer, clicking, scrolling, and droppings.

Description: A logic layer which transforms the results of gesture recognition into appropriate commands of a virtual mouse.

Instruments/Methods: Python, custom algorithms on gestures consequences mapping.

- Virtual Cursor Controller

Purpose: The user moves the on-screen cursor with his or her hand and performs actions as click, scroll, and drag and drop.

Description: This part gets commands from Gesture-To-Action Mapping system to perform on-screen movements.

Instruments/Methods: PyAutoGUI or another library for mouse emulation.

- User Feedback and Interface Module

Purpose: Offers presentation on the gestures recognized and those not recognized by the system, interacting with the user in real time.

Description: There are visual cues such as changing the color of the tip of the mouse pointer and animating the mouse pointer for gestures recognized, and tooltips for gestures not recognized.

Instruments/Methods: Overlay graphics (using OpenCV), PyAutoGUI for cursor feedback.

- The section includes the adjustable settings of the system

Purpose: This section can be used to change the parameters of the system such as speed and sensitive level of the cursor and gestures respectively.

General Description: A window that enables an adjustment of the cursor speed and terms of feedback is provided.

Tools/Technologies: Programming Language used is Python for developing the settings panel, the configurations will be stored in the configuration file.

- Data Flow and Interactions

The data flow design in the system can be divided into these major steps:

- Camera Capture:
  The camera captures a real-time video and onwards distributes to the Image Processing Module.
- Image Processing:

This module processes every intercepted frame (filtering, color smoothing, etc) and forwards enhanced frames to the Hand Tracking and Gesture Recognition Module.

- Hand Tracking and Gesture Recognition:

Hand patterns are made using landmarks which will be matched with specific gestures using the Mediapipe tool. If a gesture is detected, it moves onto Gestures to Action Mapping.

- Gestures to Actions Mapping:

Every gesture once recognized maps to a computer mouse action, such as right-click, scroll, etc, and these are fed to the Virtual Cursor Controller.

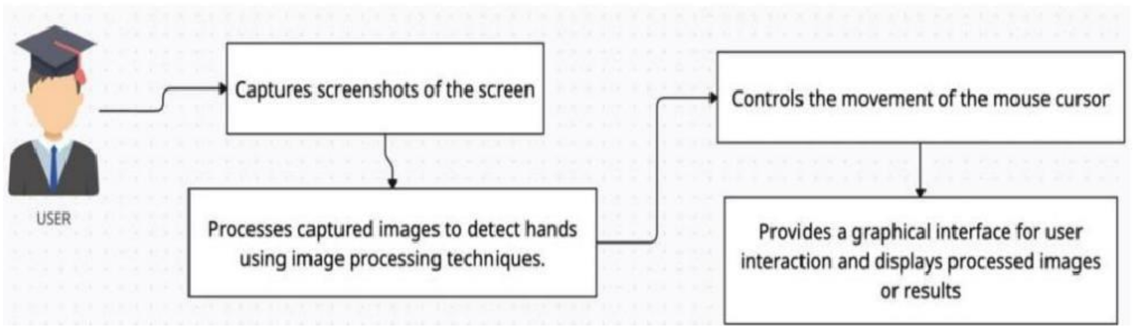
- Virtual Cursor Controller:

This performs the command that has been mapped, for instance, the cursor goes to a certain position, clicks, or scrolls and also provides details concerning the User Feedback and Interface Module.

- User Feedback:

A feedback visual is given to the user for gestures that are performed. For those that are not performed, a feedback visual is given in the form of an error message or a guide, then feedback is required and the process goes back to Step 1.

- **System Architecture Diagram**


USER
Captures screenshots of the screen
Processes captured images to detect hands using image processing techniques.
Controls the movement of the mouse cursor
Provides a graphical interface for user interaction and displays processed images or results

- Performance and Responsiveness

In order to achieve the desired level of real-time responsiveness image processing and gesture recognition should be designed and implemented with low latency in mind (target response within 0.5-1 seconds). Use optimum hand detection and hand tracking algorithms in order to ensure that the system can withstand some performance degradation, without incurring any usability issues.

- Scalability

The flexible architecture allows gesture expansion in the system by new gestures or modifications of currently existing ones.

- Usability

Feedback is present and easily understandable for assumed and not assumed gestures.

Adjustable settings help the user in switching the cursor or gesture sensitivity hence making the system user friendly.

- Security and Privacy

Considering that the system relies on real time video provision automatically no video data will be kept or sent out, which protects the users.

The application does not save or share images of the users contained in the video stream.

- Technology Knit

Programming Language used: Python Nichols Z.

- Libraries and Frameworks:

OpenCV: For capture of the image, processing activities and providing visual feedback.

Mediapipe: Contained hand tracking and recognition of gestures.

PyAutoGUI: For actions of mouse control and interaction with the screen.

Data Storage: Configuration data is kept in either JSON or YAML files.

Testing Tools: Unit testing through Pytest or any other similar testing tools and visual tests to check for accuracy and responses to gesture inputs

2.2.3 Functional Test Cases

Test Case 1.1: Right Click Gesture Recognition

Purpose: To ensure that the right click gesture is recognized and processed by the system correctly.

Preconditions: The software is active and user's hand is within sight.

Instructions:

Move the pointer of the device towards appropriate target which requires a right click (e.g., desktop).

Make the right click gesture (e.g. two finger touch).

Expected Result: The system identifies the gesture and right-clicks, followed by the display of a contextual menu.

Test Case 1.2 Right Click Gesture with Confirmation

Purpose: To ascertain that a right-click motion is accompanied by a change in display of the system.

Preconditions: The application is active and the active cursor is at a position with the likelihood of a right-click action occurring.

Instructions:

Make the right click gesture.

Expected Result: The system executes the right-click command, along with a visual cue (e.g., an icon blinking) appearing for a short time to indicate the action.

Test Case 2.1: Double Click Gesture Test

Objective: Confirm that the system performs an accurate recognition of double click gesture.

Preconditions: The application is opened and a user's palm is visible.

Steps:

1. Hover over an object (for instance, a drawer or a document) using a cursor.

2. Attempt the double-click motion (two taps quickly made) using the finger of one hand.

Expected Result:The system identifies the double click actions performed by the user and either the clicked item's window opens or the item is selected.

Test Case 2.2: Double-click Gesture with Feedback

Objective: Verify that there is a visual feedback presented for double click actions.

Preconditions: The application is open and the pointer is resting on the active element.

Steps:

1. Execute the double click gesture.

Expected Result: The system executes double-click and there is a small lag in the pointer's position before it changes to the click 'ok' pointer.

2. Touch Gesture Test Cases G- Drag-Drop action test cases

Test Case 3.1: Recognition of a drag gesture during object manipulation

Aim: To ensure that the drag gesture is comprehended and executed successfully.

Preconditions: The application is open and the hand of the user is available for tracking.

Approach:

Hover the pointer over any object that can be moved (for instance, a file icon).

Stretch the fingers on the output screen in order to perform the drag move.

Expected Result: The object is raised, and the cursor indicates the ‘grab’ position.

Test Case 3.2: Dragging and Dropping an object

Objective: Ensure the system adheres to the definition of drag-and-drop.

Preconditions: The user has already grasped the item by executing the drag gesture.

Instructions:

Tilt the hand to wheel the item in hand on the flat surface of the screen.

Take out the fingers in the remove gesture in order to release the item in the indicated area.

Expected Outcome: The item is successfully relocated, and the system provides appropriate visual indications for the action performed.

Test Case 3.3: Drag Gesture with Feedback

Objective: To test if dragging any object displays any visual response.

Preconditions: The application is opened and an object to be dragged is picked.

Steps for carrying out the test case

Initiate the drag action by using pinch.

Expected Result: The cursor indicates that ‘holding’ is in progress with an icon or indicator, which suggests something is being dragged in the process.

2.2.6 Daily Call Progress

Date: 02 September 2024

Activities or components of the project discussed: Editing in the projects for new mechanism in the project.

Date: 09 September 2024

Activities or components of the project discussed: Worked on ppt and technical term in the code.

Date: 17 September 2024

Activities or components of the project discussed: Started Research Paper and worked on the code and ppt.

Date: 24 September 2024

Activities or components of the project discussed: Worked After the review on the comments that we got.

2.3 Sprint 3

2.3.1 Sprint Goal with User Stories of Sprint 3

The objective of Sprint 3 is to focus on enhancing user interaction by fine-tuning gesture sensitivity, responsiveness and feedback systems whilst introducing new elements for better usability and flexibility. Among the functionalities are designing user calibration of acceptable gesture sensitivity, enhancing error correction and improving operating efficiency in challenging light or complex background scenes.

User Stories

User Story.1: As the end user I wish to modify the sensitivity of the gesture controls. In order to optimize the response range of the virtual mouse as per the user and situation.

Acceptance Criteria: Panel in the settings section enables the users to adjust the sensitivity for several gestures; pinch to drag or swipe to scroll for example.

Such modification of sensitivity affects immediately the sensitivity in gesture recognition and cursor movements.

Sensitivity settings remain unchanged even in different user log in sessions.

User Story 3.2: Enhancements in Low-Light Conditions and Background Handling.

I want the accuracy of gesture recognition capabilities to not be compromised even in low-light environments or visually cluttered surroundings.Thus, ensuring that the virtual mouse is effective and usable in different environments.

Acceptance Criteria:

The system behaves in a similar manner even in dimly lit environments (e.g. a room with low light).

There is an increase in the accuracy of gesture recognition with enhancements of the background.

Users are given visual information that informs them of conditions that may affect gesture recognition.

User Story 3.3: Prioritize the Constraint Handling Mechanism for Gestures that cannot be Identified.

I would appreciate it if the system gave some kind of information when the recognition of gestures failed and/or the gesture was interpreted incorrectly.

So that it becomes clear what happened and how to fix it.

Acceptance Criteria:

Error messages or icons appear when gestures are not recognized.

Available via extended tool tips or guides illustrating the performance of the gestures.

Feedback is given right away, with hardly any waiting time.

2.3.2 Functional Document

The primary goal is to enhance the operability and flexibility of a virtual mouse by improving gesture sensitivity, feedback systems, and including calibration options.

Specifications:

Gesture Sensitivity Calibration: Adjust sensitivity according to users' preferences.

Low-Light ' Background Handling' – Definition: Able to recognize gestures in different light conditions in different environmental setup.

Error Feedback – Definition: Feedbacks and help are provided within the system after a failed attempt at a recognized gesture.

Enhanced Feedback – Definition: Visual feedback indicating that the gesture has been successful (e.g. flashing cursor, changing of color)

Cursor Speed Adjustment – Definition: Users can change speed of cursor for their own convenience.

In-App Help – Definition: Tutorials and explanations of different gestures provided within the application.

Modular Code Structure – Definition: Code is separated to different files and classes to improve readability.

Performance: Targets should include recognition accuracy > 95% and feedback delay < 50 ms.

2.3.3 Architecture Document

1. Theoretical Overview

The present paper proposes the architectural design of the Gesture-Controlled Virtual Mouse focusing on the modular gesture recognition components, user interaction and feedback. The architecture is aimed at improving usability, adaptability and maintainability.

2. System Overview

The Gesture-Controlled Virtual Mouse is the system which allows the user control of the computer screen interface by just using hands. The system conducts video input in real time, recognizes hand gestures and performs corresponding actions of the cursor on the screen. The architecture has been created to support user calibration in terms of sensitivity, error correction and processing and feedback.

1. Component Descriptions

Settings Panel : Customize gesture sensitivity and cursor speed.

Help Section : Tutotrials and documentation for gestures.

Module of Gesture Recognition :

Functional - real-time detection of gestures with indices using Mediapipe and opencv.

Input - live camera.

Output - gestures perceived like swipe, pinch and etc.

Calibration Module :

Functionality: Adjusts sensitivity based on user input.

Input - settings defined by the user.

Output – thresholds for recognition are changed.

Feedback Module :

Functionality - Animate feedback whether the gesture has been successful or not.

Input - gestures intaken and mistakes commited.

Output – Feedback is provided by the indicators like cursor change.

Error Handling Module :

Functionality : Messages given for gestures that are not recognized.

Input : the data from gesture recognition.

Output: error messages or tool tips.

2. Transactions and Information Flow

User Involvement: Users change settings and execute actions.

Processing Flow: The gesture recognition engine takes in the feed and recognizes the gestures, errors and error correction is also implemented.

Feedback Loop: There are veritable changes in the visuals that take place as one interacts, thus enhancing user interactivity.

3. Other Important Factors

Performance : Set the latency to be less than 50 ms.

Scalability : Development is made in such a way that it is possible to add new features easy.

Maintainability : The use of modularity in coding allows for the easy revisits on updates or debugging.

2.3.5 Functional Test Cases

TC-01: Check if Settings Panel is Launched in Correct Manner

Step 1: Start up the application, Gesture-controlled virtual mouse.

Step 2: Look for the settings option in the main menu.

Step 3: Press the button entitled “Settings”.

Expected Result: A screen with settings is shown.

TC-02: Check if Choose your Sensitivity is Customizable

Step 1: Access the Settings Panel.

Step 2: Find the gesture sensitivity slider.

Step 3: Move the slider to 5.

Step 4: As a result, click the “Apply” button to save the changes.

Expected Result: The gesture sensitivity is set at 5.

TC-03: Check if Cursor Speed is Customizable

Step 1: Access the Settings Panel.

Step 2: Find in the section where cursor speed can be adjusted.

Step 3: Increase the cursor speed to 10.

Step 4: As a result, click on the “Apply” button to save the changes.

Expected Result: The cursor speed is 10.

2.3.6 Daily Call Progress

Date: 01 October 2024

Activities or components of the project discussed: Worked After the review on the comments that we got.

Date: 09 October 2024

Activities or components of the project discussed: Checked Plagiarism for the project and worked on it to reduce it.

Date: 17 October 2024

Activities or components of the project discussed: Publication of the research Paper and participated in the conference.

Date: 24 October 2024

Activities or components of the project discussed: Final report preparation and worked on report.

Date: 01 November 2024

Activities or components of the project discussed: Working on Report and completing it for submitting.

RESULTS AND DISCUSSIONS

3.1 Project Outcomes

CONCLUSIONS & FUTURE ENHANCEMENT

The main objective of the system of the artificial intelligence virtual cursor is to replace a physical mouse by manipulating the functions of a body mouse via hand movements. It may employ a web camera or a built-in camera that recognizes hand gestures, processes their frame and performs the required mouse movements. To summarize, the developed artificial intelligence virtual mouse system outperformed previous systems in terms of accuracy and quality, and it largely resolved those issues. Thanks to the evolvement of the proposed system which is more precise and can work electronically with hand gestures instead of the usual physical mouse, the intelligent virtual cursor can also be put into operational use.

The usage of control by virtual mouse and hand gestures can be foreseen with a great desire as far as the future is concerned. Evolution in gesture detection in future will enable the users to perform more effective interactions such as zooming, sliding and twiddling with the fingers which are very complex. This natural interface will be taken to a higher level with the integration of voice commands which will allow a more natural form of interaction with computers. Furthermore, rather than addressing a single demand, multi modal interaction will also let the individuals combine within the same action the hand and the screen or the keyboard and will allow the designing of spaces that will be used for work and will be productive. In this way, it is justifiable to aim at enhancing the updating of the existing accessibility features as well as the development of those features that will promote the use of computers by disabled persons adaptive interfaces. The communication, however, can be integrated through 3D hand tracking technology, which supports all other formats besides the conventional PC and makes it easy to actively communicate in virtual and augmented reality. Performing tasks around the smart home could be as simple as moving one's hand, or even interacting with objects in a different dimension. There are three main approaches you can apply in designing an effective virtual mouse that will revolutionize man-machine interface, and these are focusing on continuous action to enhance smooth cursor motion.

APPENDIX

B. SAMPLE CODING

```
import cv2
import mediapipe as mp import
pyautogui
import time

# Initialize MediaPipe hands module mp_hands
= mp.solutions.hands mp_drawing =
mp.solutions.drawing_utils

# Set up the webcam
cap = cv2.VideoCapture(0)

# Initialize the hands model
with mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.5) as hands:
    while cap.isOpened():  success,
        image = cap.read() if not
        success:
            print("Ignoring empty camera frame.") continue

        # Flip the image horizontally for a later selfie-view display
        image = cv2.flip(image, 1)

        # Convert the BGR image to RGB
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # Process the image and find hands results
        = hands.process(image_rgb)

        # Draw hand landmarks on the image if
        results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(image, hand_landmarks,
mp_hands.HAND_CONNECTIONS)

                # Get landmark positions for gesture recognition
                thumb_tip_y =
hand_landmarks.landmark[mp_hands.HandLandmark.THUMB_TIP].y index_tip_y =
hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].y middle_tip_y =
hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_TIP].y

                # Determine the gesture based on landmark positions
                if thumb_tip_y < index_tip_y and middle_tip_y < index_tip_y:  # Thumb up
gesture
                    cv2.putText(image, "Gesture: Left Click", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
                    pyautogui.click()  # Perform left click
                    time.sleep(1)  # Prevent multiple clicks in quick succession

                elif index_tip_y < middle_tip_y:  # Peace sign gesture
                    cv2.putText(image, "Gesture: Right Click", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
                    pyautogui.click(button='right')  # Perform right click time.sleep(1)  #
                    Prevent multiple clicks in quick succession


                elif thumb_tip_y > index_tip_y and middle_tip_y > index_tip_y:  # Fist
gesture
                    cv2.putText(image, "Gesture: Scroll", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
                    pyautogui.scroll(-10)  # Scroll down; use positive value for scrolling up
                    time.sleep(1)  # Prevent multiple scrolls in quick succession

        # Display the resulting frame
        cv2.imshow('Hand Gesture Recognition', image)

        if cv2.waitKey(5) & 0xFF == 27:  # Press 'ESC' to exit break

# Release the webcam and destroy windows cap.release()
cv2.destroyAllWindows()
```

IMPLEMENTATION

1. Importing Libraries

```
import cv2
import mediapipe as mp import
pyautogui
import time
```

- OpenCV (cv2): Used for handling the webcam feed and displaying frames with annotations.
- MediaPipe (mp): A framework that provides solutions for hand tracking and gesture recognition.
- PyAutoGUI (pyautogui): Automates mouse actions like clicks and scrolling.
- time: Used to add delays between gestures to avoid multiple triggers in quick succession.

2. Initializing MediaPipe Hands Module and Drawing Utilities

```
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
```

- mp_hands: This initializes the MediaPipe hand tracking module, which can detect and track hands in real-time.
- mp_drawing: Provides drawing utilities to visualize the hand landmarks detected by MediaPipe.

3. Setting Up the Webcam

```
cap = cv2.VideoCapture(0)
```

- cap: Captures video feed from the default webcam (0indicates the default camera). This feed will be processed frame by frame.

4. Initializing and Running the Hand Tracking Model

```
with mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.5) as hands:
    while cap.isOpened():
```

- mp_hands.Hands(): This initializes the hand tracking model.
  - max_num_hands=1: Limits the tracking to a single hand.
  - min_detection_confidence=0.5: The minimum confidence score to detect a hand.

The whileloop processes each frame as long as the camera is open.

5. Reading and Flipping the Image

```
success,   image   =   cap.read()   if   not
success:
        print("Ignoring empty camera frame.") continue

image = cv2.flip(image, 1)
```

- cap.read(): Captures a frame from the webcam.
- cv2.flip(image, 1): Mirrors the image for a natural selfie view so that hand movements match real-time movements.

6. Converting to RGB and Processing with MediaPipe

```
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) results =
hands.process(image_rgb)
```

- cv2.cvtColor: Converts the color format from BGR (default in OpenCV) to RGB, required by MediaPipe.
- hands.process(image_rgb): Processes the frame to detect hand landmarks. The results object contains the detection details.

7. Drawing Hand Landmarks and Identifying Gestures

```
if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks: mp_drawing.draw_landmarks(image,
        hand_landmarks,
mp_hands.HAND_CONNECTIONS)
```

- results.multi_hand_landmarks: If hands are detected, multi_hand_landmarkscontains details about each hand's landmarks.
- mp_drawing.draw_landmarks: Draws landmarks on the hand and connects them to represent finger and palm positions.

8. Gesture Recognition Logic

Landmark positions are used to define three basic gestures and map them to mouse actions.

```
# Get landmark positions for gesture recognition thumb_tip_y =
hand_landmarks.landmark[mp_hands.HandLandmark.THUMB_TIP].y index_tip_y =
hand_landmarks.landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].y middle_tip_y =
hand_landmarks.landmark[mp_hands.HandLandmark.MIDDLE_FINGER_TIP].y
```

- Retrieves y-coordinates of specific landmarks on the thumb, index, and middle fingers, used to detect gestures based on their relative positions.

8.1 Thumb-Up Gesture (Left Click)

```
if thumb_tip_y < index_tip_y and middle_tip_y < index_tip_y: cv2.putText(image, "Gesture: Left
        Click", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        pyautogui.click()
        time.sleep(1)
```

- Condition: Both thumb and middle finger tips are above the index finger tip.
- Action: Displays "Left Click" on the frame and performs a left mouse click using pyautogui.click().
- time.sleep(1): Adds a 1-second delay to prevent multiple clicks.

8.2 Peace Sign Gesture (Right Click)

```
elif index_tip_y < middle_tip_y:
        cv2.putText(image, "Gesture: Right Click", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        pyautogui.click(button='right') time.sleep(1)
```

- Condition: The index finger tip is above the middle finger tip, indicating a "peace sign."
  - Action: Displays "Right Click" and performs a right mouse click using pyautogui.click(button='right').

8.3 Fist Gesture (Scroll Down)

```
elif thumb_tip_y > index_tip_y and middle_tip_y > index_tip_y: cv2.putText(image,
    "Gesture: Scroll", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    pyautogui.scroll(-10) time.sleep(1)
```

- Condition: Thumb and middle finger tips are below the index finger tip, forming a "fist."
- Action: Displays "Scroll" and scrolls down using pyautogui.scroll(-10). Positive values would scroll up.

9. Displaying the Output and Exit Condition

```
cv2.imshow('Hand Gesture Recognition', image)
if cv2.waitKey(5) & 0xFF == 27:          # Press 'ESC' to exit
    break
```

- cv2.imshow: Shows the processed video feed with annotations.
- cv2.waitKey(5) & 0xFF == 27: Checks if the 'ESC' key is pressed to break the loop and end the program.

10. Releasing Resources

```
cap.release()
cv2.destroyAllWindows()
```

- Releases the webcam and closes any OpenCV windows to free up resources after the loop ends.

C. PLAGIARISM REPORT