Tribhuvan University

Faculty of Humanities and Social Sciences

A Project Proposal

On

HAND GESTURE CONTROLLED VIRTUAL MOUSE

Submitted to:

Department of Computer Application

Patan Multiple Campus

Patan Dhoka, Lalitpur

In partial fulfilment of the requirements for the Bachelors in Computer Application

Submitted by:

Yojan Karki

Roll no: 39/076

8th Semester

Tribhuvan University

Faculty of Humanities and Social Sciences

Patan Multiple Campus

Patan Dhoka, Lalitpur

Bachelor in Computer Applications (BCA)

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project be prepared under my supervision by **Yojan Karki** entitled "**HAND GESTURE CONTROLLED VIRTUAL MOUSE"** in the Partial Fulfillment of requirement for the degree of Bachelor in Computer Application is recommended for that final evaluation.

……

Jagdish Bhatta

Project Supervisor

BCA Department

Patan Multiple Campus

Tribhuvan University

Faculty of Humanities and Social Sciences

Patan Multiple Campus

Patan Dhoka, Lalitpur

Bachelor in Computer Applications (BCA)

LETTER OF APPROVAL

This is certified that this project prepared by **Yojan Karki** entitled "**HAND GESTURE CONTROLLED VIRTUAL MOUSE**" in the Partial Fulfillment of requirement for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

…..... Jagdish Bhatta Supervisor BCA Department Patan Multiple Campus	...…..... Roshan Tandukar Program Coordinator Patan Multiple Campus Patan Dhoka, Lalitpur
…..... Internal Examiner	…..... External Examiner

ACKNOWLEDGMENT

The success of this project would not have been possible without the joint efforts of the college, the teachers, the supervisor and the faculty members, and I am immensely blessed to have got this all along the duration of my project. I would like to extend my profound gratitude to each one of them.

I am highly indebted to **Patan Multiple Campus** for constant guidance and supervision, as well as for providing all the necessary ICT infrastructure and friendly environment for the successful completion of the project. I am also appreciative of the efforts of BCA coordinator **Mr. Roshan Tandukar**, without his supporting role, the project would have been nowhere near completion.

I would like to express my gratitude to my project supervisor **Mr. Jagdish Bhatta** who took keen interest in my project and guided me throughout the project by providing all the necessary ideas, information and knowledge for developing a functional web application. Further thankful for his constant encouragement and guidance towards making this report standard as per the norms and values.

I am thankful and fortunate enough to get constant support from my seniors and every teaching staff of the BCA department which helped me successfully complete my project. I would also like to extend my regards to all the non-teaching staff of the BCA department for their timely support.

Last but not the least, I would like to thank my teachers, parents and colleagues who have been knowingly or unknowingly part of this project and lent support and views during the entire development time.

Yours sincerely,

Yojan Karki

ABSTRACT

The Hand Gesture Controlled Virtual Mouse project aims to provide a novel and intuitive interaction mechanism for controlling a computer's mouse cursor using hand gestures. Traditional computer input devices such as mice and touchpads often lack the naturalness and flexibility of human hand movements. This project seeks to address this limitation by leveraging computer vision and machine learning techniques to interpret hand gestures and translate them into mouse cursor movements. The system employs a camera to capture real-time video input of the user's hand. The captured video frames are then processed using computer vision algorithms to detect and track the user's hand within the frame. Various hand gesture recognition algorithms are applied to interpret the hand movements and map them to specific mouse cursor actions, such as movement, clicking, and scrolling.

Overall, this project showcases the potential of hand gesture recognition technology in revolutionizing human-computer interaction by providing a natural, hands-free, and intuitive control mechanism for computer users. It opens up new possibilities for enhancing accessibility and usability in various computing domains, ranging from gaming and virtual reality to productivity and accessibility applications.

Keywords: Media Pipe, Open CV, Virtual Mouse, Hand Gesture, Python3

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
BCA	Bachelors of Computer Application
BGR	Blue Green Red
CNN	Convolutional Neural Network
CV	Computer Vision
GUI	Graphical User Interface
HGCVM	Hand Gesture Controlled Virtual Mouse
IRJET	International Research Journal of Engineering and Technology
MATLAB	Matrix Laboratory
ML	Machine Learning
RGB	Red Green Blue

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

In our technology-driven world, computers have become essential in nearly every aspect of our lives. The rapid growth of computer technologies has led to remarkable advancements, making them indispensable tools capable of performing tasks beyond human capabilities. Traditionally, humans have interacted with computers through input devices like the mouse, a key tool for navigating graphical user interfaces (GUIs).

However, hardware mice and touchpads have limitations. They can be cumbersome, time-consuming for complex tasks, and prone to damage, leading to decreased productivity. Wireless mice improved convenience but introduced their own challenges. Speech recognition technology emerged as an alternative, allowing voice commands to control computers, but it can be slow due to the need for accurate interpretation. Eye-tracking technology also offered a new way to control the mouse cursor but faced issues like interference from contact lenses or long eyelashes.

Recognizing the need for more intuitive interaction methods, developers have explored human gesture recognition models. These often-required costly gloves and sensors, which limited their widespread adoption. Despite these challenges, advancements in artificial intelligence (AI) have opened new possibilities.

This technology uses a camera to track hand movements and perform mouse functions. Computer vision algorithms analyse the video feed to identify gestures like pointing, swiping, or clicking. Machine learning models, trained on extensive hand gesture datasets, interpret these gestures and translate them into corresponding mouse movements.

The Hand Gesture Controlled Virtual Mouse offers several benefits:

1. Enhanced accessibility for people with mobility impairments.
2. Elimination of the need for a physical mouse or touchpad.
3. A more immersive and engaging user experience.

This innovative approach to human-computer interaction addresses the limitations of existing methods and introduces a new paradigm in control mechanisms.

1.2 Problem Statement

Computer technologies are essential in many areas like education, business, and government. However, traditional input devices like mice and touchpads have problems that easy they are to use, their accessibility, and how portable they are. This is especially important in Nepal, where people and organizations with limited resources might find it hard to get and maintain these devices because of cost or availability.

Globally, people with mobility impairments also struggle with traditional input devices, and those in remote or underserved areas may have trouble obtaining them. In today's fast-paced and mobile world, there's a growing need for portable and east-to-use interaction methods. Carrying a physical mouse or touchpad is not always practical for people who work on the go or in unusual environments.

Therefore, an alternative solution is needed that provides a more accessible, portable, and intuitive way to interact with computers, both in Nepal and worldwide.

1.3 Objectives

The objectives of this project are:

- To enhance portability and convenience by eliminating the need for physical input devices like mice or touchpads.

1.4 Scope and Limitation

Every system has its own pros and cons. This system has following scope and limitations

1.4.1 Scope

The scope of this system includes:

- Creating software to recognize hand gestures and control the computer mouse.
- Focusing on predefined gestures like pointing, swiping, clicking, and scrolling.
- Ensuring real-time tracking and quick responses to hand movements.
- Aiming for compatibility across different computer systems and operating systems.
- Measuring performance through accuracy, responsiveness, and user satisfaction.
- Considering future improvements based on user feedback and needs.

1.4.2 Limitations

The limitations of this project include:

- Accuracy may be affected by lighting, background clutter, and different hand shapes or sizes.
- Users might need time to get used to the hand gesture system and learn the gestures.
- The system recognizes a limited number of gestures, which could restrict interaction.
- Performance depends on the quality of the camera used for tracking.
- Hand gesture control may not be ideal for people with severe motor impairments or limited hand mobility.
- Some users may prefer traditional input devices due to comfort or familiarity.

1.5 Development Methodology

The waterfall model is a possible development methodology for the system. The waterfall model is a linear and sequential approach to project management, where each phase of the project is completed before moving to the next one. The waterfall model for this system has the following phases:

- **Requirements:** Customer requirements for the system are gathered in this phase such as their preferred location and room type.
- **Design:** The architecture and structure of the system are designed in this phase, based on the requirements. Decisions are made about the tools, technologies, and methods to use for requirements. Decisions are made about the tools, technologies, and methods to use for developing the system.
- **Implementation:** The system is coded and tested in this phase, according to the design. The source code is written and unit testing is performed.
- **Maintenance:** This system is maintained and updated in this phase.

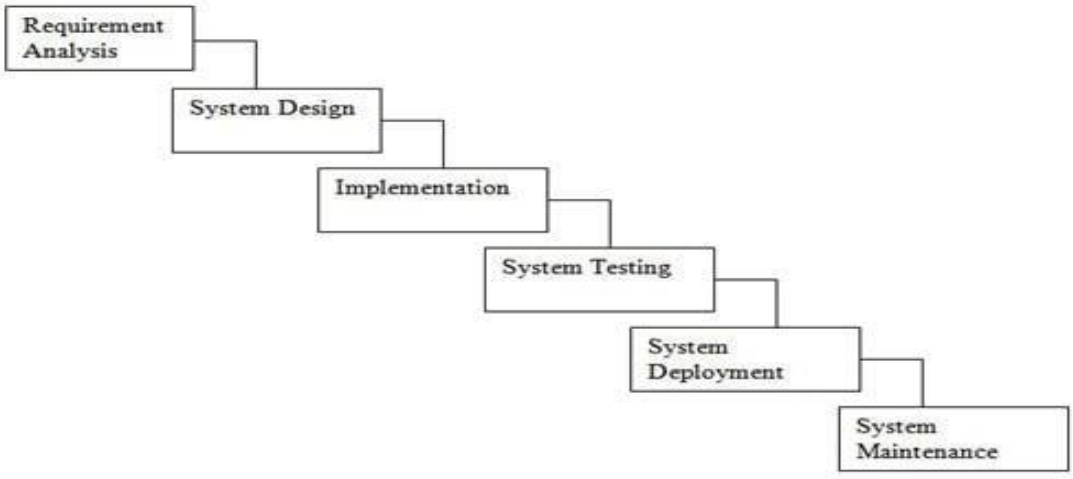

Figure 1.5.1: Waterfall model of the Hand Gesture Controlled Virtual Mouse

1.6 Report Organization

The report is prepared following the guidelines provided by the Faculty of Humanities and Social Science, Tribhuvan University. The report is separated into different chapters and each chapter consists of various sub chapters with its content. The preliminary section of the report consists of Title Page, Acknowledgment, Abstract, Table of Contents, List of abbreviations, List of figures and List of Tables. The main report is divided into 5 chapter, which include:

Chapter 1: Introduction

It includes the general overview of the system and the project. It includes the Introduction, Problem Statement, Objective, Scope and Limitations of the project etc.

Chapter 2: Background Study and Literature Review

It includes the study of the current scenario the system will be deployed into. It includes the Review of the similar projects, theories done by other researchers.

Chapter 3: System Analysis and Design

System Analysis and Design of the system using various charts and figures. Functional requirements defined using use cases and other techniques. Object Modelling: Object & Class Diagram, Dynamic Modelling: State & Sequence diagram, Process modelling: Activity Diagram. Similarly, System design diagrams like refinement of class and object, component as well as deployment diagram are drawn.

Chapter 4 Implementation and Testing

It includes the details of the different design and development tools used. The implementation details of the modules presented in the form of code snippets of functions, classes, it also includes the testing of the system with different test cases as per the requirement.

Chapter 5: Conclusion and Future Recommendations

It includes the summary of the system and the project. It also includes the possibilities the system can implement in the future.

The final part of the report consists of References and Appendices. The references are listed in accordance with the IEEE reference standards and the Appendices includes the screenshots of the system.

CHAPTER 2: BACKGROUND STUDY

2.1 Background Study

The world is rapidly advancing in technology, making computers and essential part of our daily lives. Interaction with computers has traditionally relied on input devices like mice and touchpads, which help users navigate graphical user interfaces (GUIs) and perform tasks. Early computer mice were wired, limiting movement and causing the inconvenience of tangled cables. As technology evolved, wireless mice were introduced, offering greater freedom of movement and enhancing ease of use.

With further advancements, speech recognition emerged as a promising input method, allowing users to control computers through voice commands, this innovation enabled hands-free operation, simplifying tasks like searching the web of transcribing text. However, using speech recognition for precise tasks like pointing and clicking proved inefficient and time-consuming. This led researchers to explore alternative methods of human-computer interaction, including eye tracking technology. Eye tracking uses cameras or sensors to monitor eye movements and translate them into cursor movement on the screen. While useful in fields like psychology and market research, eye tracking requires specific hardware and calibration, which can be cumbersome and less accurate for some users.

More recently, hand gesture recognition has gained attention as a natural and intuitive way to control computers. This technology uses computer vision and artificial intelligence to track and interpret hand movements, allowing users to interact with computers through gestures. Hand gesture recognition is especially beneficial for individuals with mobility impairments, as it removes the need for physical input devices. It also offers a more direct and intuitive way of interacting with computers, bridging the gap between the physical and digital worlds. Developing hand gesture-controlled virtual mouse systems involves using cameras to capture video input of the user's hand. Computer vision algorithms process this video feed, identifying and tracking hand movements in real-time. Machine learning models are trained to recognize specific gestures like pointing, swiping, clicking, and scrolling, translating them into corresponding actions on the computer screen.

This method of interaction mimics real-world hand movements, creating a seamless and engaging user experience. Hand gesture control has the potential to revolutionize user interfaces, making them more accessible and intuitive for both novice and expert users. In conclusion, integrating hand gesture recognition into virtual mouse systems offers a promising

new approach to human-computer interaction. It provides advantages such as improved accessibility, natural interaction, and greater portability. Beyond traditional computer use, this technology has significant potential in gaming, virtual reality, and augmented reality. However, challenges related to accuracy, user adaptation, hardware limitations, and compatibility must be addressed for successful implementation and widespread adoption of hand gesture-controlled virtual mouse systems.

2.2 Literature Review

The project "Hand Gesture Controlled Virtual Mouse" investigates using hand gestures to control computer mouse functions. This innovative approach to human-computer interaction has drawn significant attention from researchers and developers. By tracking and interpreting hand movements, users can interact with graphical user interfaces without physical input devices like mice or touchpads.

One of the seminal works in this field is the paper "Hand Gesture Recognition for Virtual Mouse Control" by Hindusthan College of Engineering and Technology. The authors propose a deep learning-based hand gesture recognition system to control mouse functions such as left click, right click, and scrolling without a physical mouse, thereby reducing the risk of COVID-19 spread by minimizing physical contact [1].

Another notable contribution is the study "Hand Gesture Controlled Virtual Mouse Using Artificial Intelligence" by the Bannari Amman Institute of Technology. This system uses a camera to capture hand images, which are processed by an AI algorithm to recognize gestures. The gestures are then translated into mouse movements on a virtual screen, providing an alternative interface for users with difficulty using traditional input devices [2].

Vantukala VishnuTeja Reddy's study on "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition" explores two methods: using colored caps and hand gesture detection. The system detects fingers using color identification and tracks hand gestures through contour detection and convex hull formation, enabling on-screen cursor control [3].

The International Research Journal of Engineering and Technology (IRJET) features a project that employs machine learning and computer vision algorithms for hand gesture recognition. This system uses CNN models implemented by Mediapipe and does not require additional

hardware, allowing users to control the cursor and perform actions like clicking and dragging with hand movements [4].

Hand gesture recognition (HGR) is a natural and intuitive way to interact with computers. The paper "Curvature of Perimeter" presents a new approach using fingertip detection for HGR. The system utilizes a webcam and computer vision algorithms developed with MATLAB toolboxes to implement a virtual mouse [5].

In conclusion, the reviewed literature highlights significant advancements in hand gesture controlled virtual mouse technology. Various approaches, including deep learning models, vision-based techniques, and assistive applications, have been explored to enhance accuracy, robustness, and accessibility. These studies provide valuable insights for developing the Hand Gesture Controlled Virtual Mouse project.

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 System Analysis

During the system analysis phase, the requirements and objectives of the Hand Gesture Controlled Virtual Mouse system were identified and defined. This involved understanding the needs of the users, and the specific context in which the system will be deployed.

3.1.1 Requirement Analysis

i. Functional Requirements

The functional requirements of hand gesture controlled virtual Mouse is:

- **Gesture Recognition:** The system should accurately recognize and classify hand gesture performed by the user, such as pointing, swiping, clicking, scrolling, and dragging.
- **Cursor Control:** The system should translate recognized hand gestures into corresponding mouse movements, allowing users to control the cursor on the computer screen accurately.

  This is important to note these functional requirements provide a general overview and can be further refined or customized based on specific project goals, user needs, and technological capabilities.

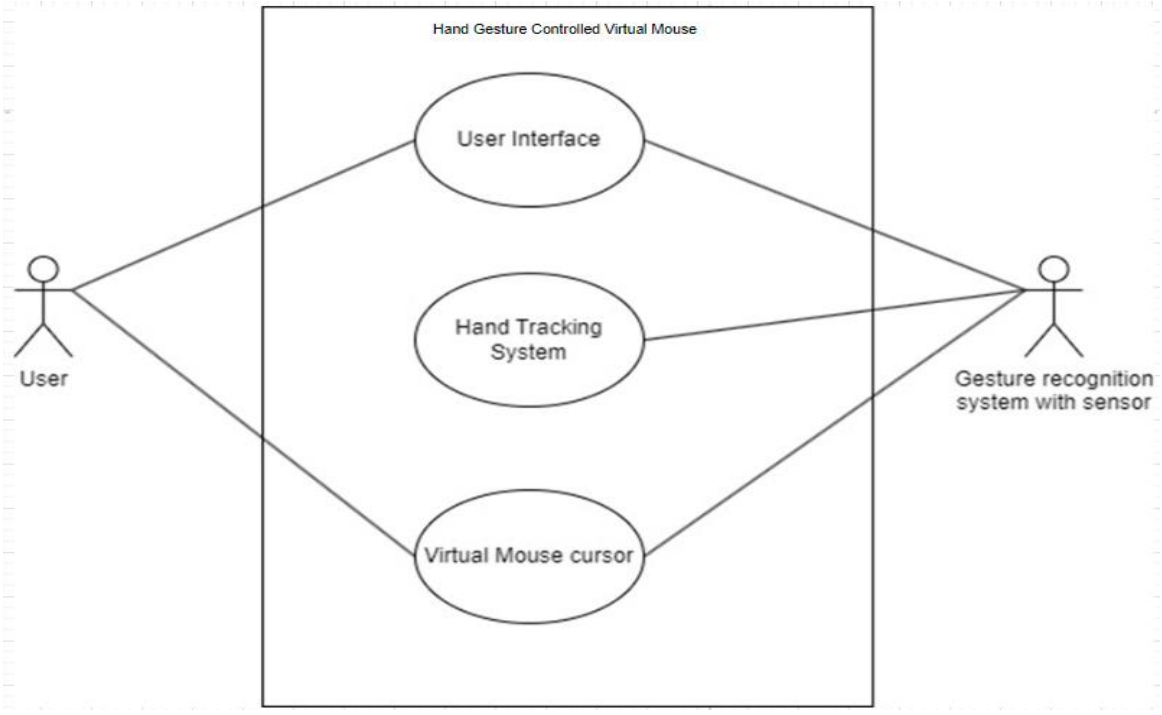

Figure 3.1.1: Use Case Diagram of the Hand Gesture Controlled Virtual Mouse

ii. Non-functional Requirements

Security: Implement robust security measures to protect user data captured through the webcam.

Availability and Reliability: Ensure the system is highly available with minimal downtime, providing consistent access to users.

Performance: Achieve low latency in recognizing and responding to hand gestures, ensuring real-time processing without noticeable delay.

3.1.2 Feasibility Analysis

i. **Technical feasibility:** The system is found to be technically feasible as no additional hardware or software on top of existing system is required to run the system, a simple laptop with Visual Studio Code was more than enough. The technical resources used in this project are Visual Studio Code, and python which are easy to use. No additional knowledge was required to run this software. Similarly, completing the project with Python on Visual Studio Code required no additional knowledge.

ii. **Operational feasibility:** The Hand Gesture Controlled Virtual Mouse is available on ease once it is released. It is simple to use and covers almost all factors that a hand gesture controlling system should have. With many features included in, it fulfills the criteria of the operational feasibility.

iii. **Economic feasibility:** The project is economically viable as resources required to successfully complete the project were available to download for free from the internet. Visual Studio Code is free to download and other python libraries were easily accessible in the internet. Since, there was no need to spend any financial resources on any of the additional hardware or software, the system is economically feasible.

3.1.3 Object Modeling (Class diagram)

The class diagram outlines a system for controlling a virtual mouse using hand gestures. The system includes four main components: Hand, Virtual Mouse, Gesture Recognizer, and Calibration.

1. **Hand**: Represents the user's hand tracked in 3D space. It has attributes like position, gesture, finger count, velocity, and acceleration. The Hand class includes methods to update its position, detect gestures, count fingers, and adjust velocity and acceleration.

2. **Virtual Mouse**: Represents the on-screen cursor. It has attributes for its position and button states (left and right click). The Virtual Mouse class includes methods to move the cursor and manage button clicks.

3. **Gesture Recognizer**: Tracks the hand, recognizes gestures, and maps those gestures to mouse actions. It manages attributes like tracked hands, gesture mapping, thresholds, and calibration data. It also includes methods for tracking, gesture recognition, action mapping, threshold adjustment, and calibration.

4. **Calibration**: Handles the process of optimizing hand tracking and gesture recognition. It has attributes related to calibration phases, data, and parameters. The Calibration class provides methods to start calibration, collect data, and fine-tune settings based on that data.

In summary, the class diagram shows how these entities work together to enable hand gesture control of a virtual mouse, with the Calibration component enhancing system accuracy.

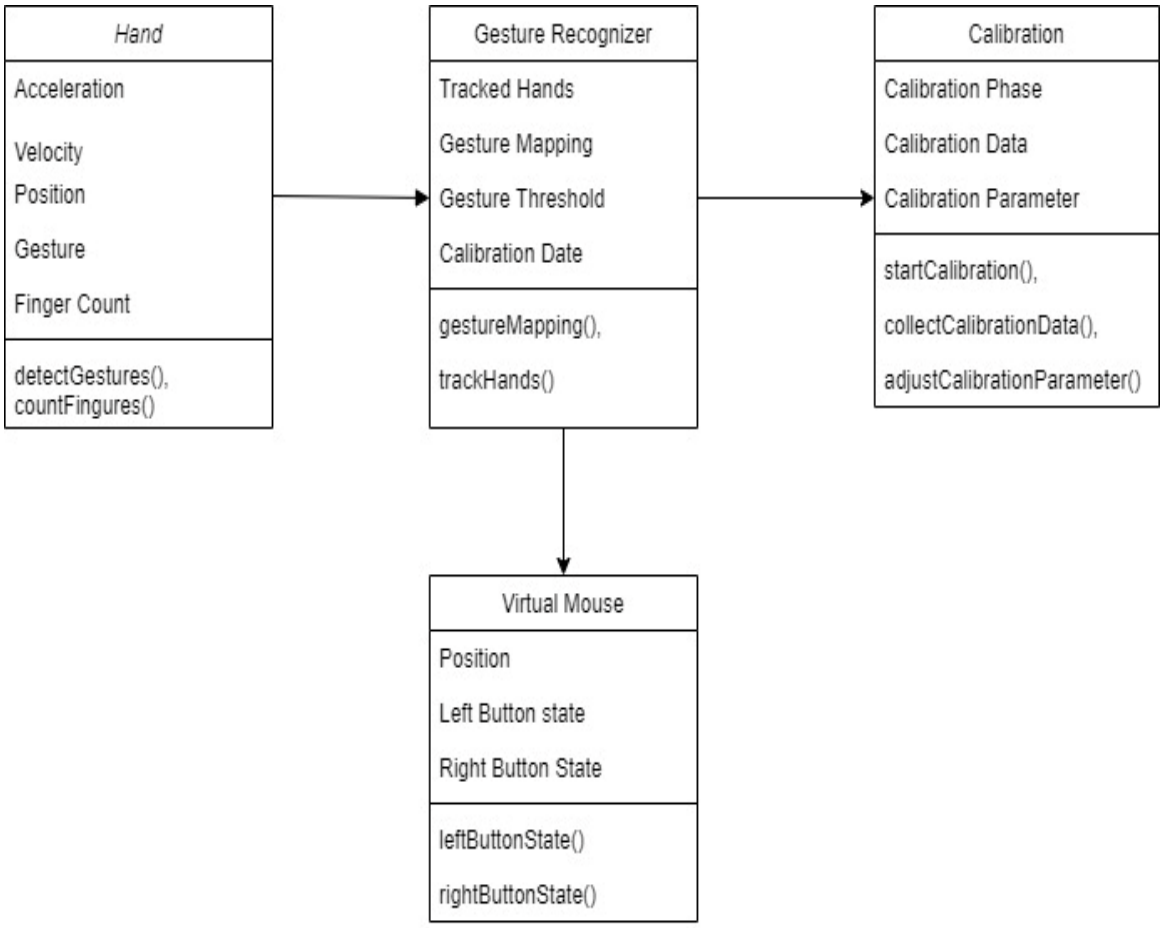

Figure 3. 1 Class Diagram of the Hand Gesture Controlled Virtual Mouse

3.1.4 Dynamic Modeling

In the dynamic modelling of the Hand Gesture Controlled Virtual Mouse, both a state diagram and a sequence diagram are created. The state diagram shows the different states of the application, while the sequence diagram details the order of processes during the application's execution.

3.1.4.1 State Diagram

The state diagram illustrates the various states and transitions involved in recognizing hand gestures and controlling the virtual mouse. It shows the flow of data and operations through stages like receiving input frames, detecting hands, extracting hand landmarks using a model like MediaPipe, recognizing gestures, and executing the corresponding actions.

- **Input Frames:** This state represents the video or image frames received for processing.
- **Hand Detection:** The system identifies and locates hands within the input frames.
- **Landmark Extraction:** Once a hand is detected, key points or landmarks on the hand are extracted using a model like MediaPipe.
- **Gesture Recognition:** The system analyses these landmarks to recognize the specific gesture being performed.
- **Action Execution:** After recognizing the gesture, the system performs the associated action, such as moving the cursor, clicking, or executing a command.

The state diagram visually represents this sequence of states and transitions in the gesture-controlled virtual mouse system.

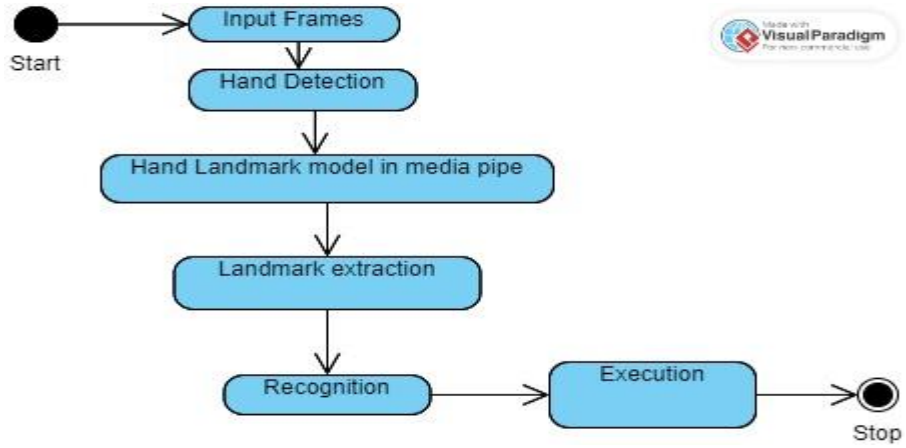

Figure 3. 2 State Diagram of the Hand Gesture Controlled Virtual Mouse

3.1.4.2 Sequence Diagram

In this sequence diagram, there are three key entities: Virtual Mouse, Gesture Recognizer, and Hand. The arrows show how messages flow between them.

Here’s how the process works:

- The Virtual Mouse starts the process by calling the trackHands() method in the Gesture Recognizer.
- The Gesture Recognizer tracks the hands, interacts with the hand tracking system, and then calls recognizeGesture().
- The Gesture Recognizer identifies the gesture and asks the Hand to update its position using the updatePosition() method.
- The Hand module updates its velocity by calling the updateVelocity(velocity) method.
- The Hand module counts the fingers by calling the countFingers() method.
- The Gesture Recognizer maps the recognized gesture to a mouse action using the mapGestureToAction(gesture) method.
- The Hand module updates the virtual mouse's position by calling the updatePosition(x, y) method.
- The Hand module updates the virtual mouse's left button state using the setLeftButtonState(state) method.
- The Hand module updates the virtual mouse's right button state using the setRightButtonState(state) method.

This sequence diagram outlines the simplified interactions between these entities in the Hand Gesture Controlled Virtual Mouse system.

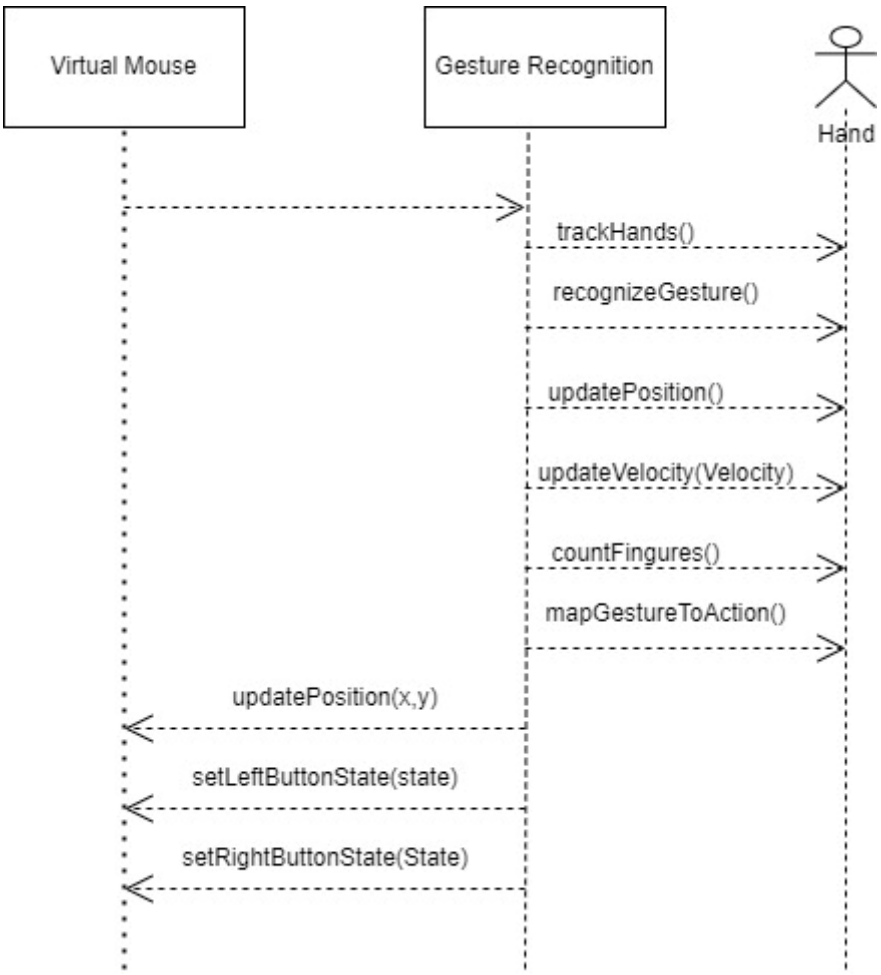

Figure 3. 3 Sequence diagram of the Hand Gesture Controlled Virtual Mouse

3.1.5 Process Modeling

In this activity diagram, the process begins by capturing an input frame from a camera or other device. The system then detects any hands in the frame using hand detection algorithms. Once a hand is detected, the system identifies key landmarks like finger and palm positions using a hand landmark model. These landmarks are then analysed to recognize the gesture the user is making, either by comparing with pre-defined patterns or using machine learning.

Based on the recognized gesture, the system performs the corresponding mouse action, such as moving the cursor, clicking, scrolling, or other mouse functions. The process then ends, and the system is ready to capture the next frame for continuous interaction.

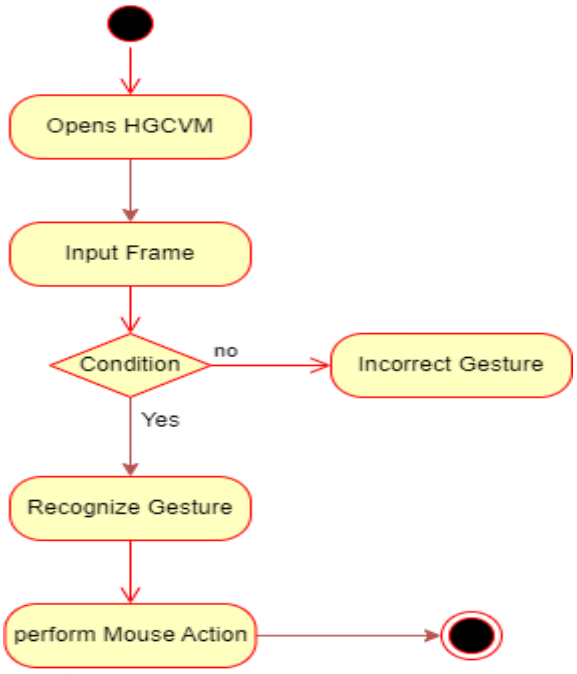

Figure 3. 4 Process modelling using Activity Diagram of the HGCVM

3.2 System Design

System design involves refining classes and objects, followed by creating component and deployment diagrams. The refinement diagram is similar to the class diagram but includes more details about data types, operations, and class relationships. The component diagram provides a simple overview of the components in the Hand Gesture Controlled Virtual Mouse. The deployment diagram shows how the system's modules are deployed.

3.2.1 Refinement of classes and object

The refined class and object diagram for the Hand Gesture Controlled Virtual Mouse adds detail to the system's structure and behaviour. The "Hand" class includes attributes like position, gesture, finger count, velocity, and acceleration, with methods for updating these attributes. The "Virtual Mouse" class manages the virtual mouse's position and button states, with methods for updating them. The "Gesture Recognizer" class tracks hand, recognizes gestures, maps them to actions, adjusts thresholds, and handles calibration. The "Calibration" class manages the calibration process, including phases, data collection, and parameter adjustments. This refinement provides a clearer understanding of how the system functions.

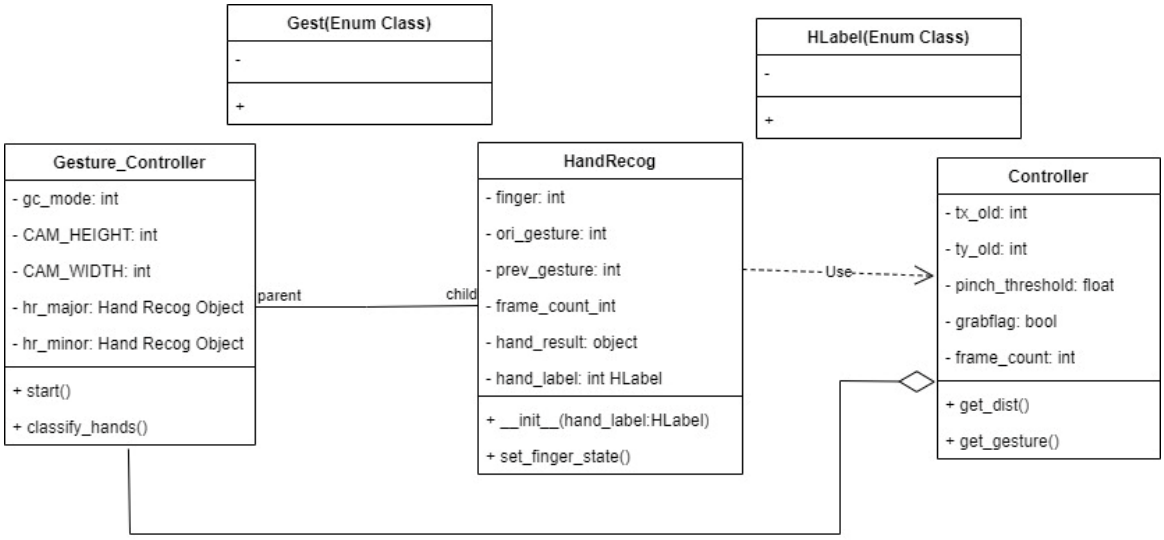

Figure 3. 5 Refinement of Classes and Object Diagram

3.2.2 Component Diagram

Here component diagram consists of five components, namely the major modules. The Input frames, Hand Detection, Recognition, Execution and Perform Mouse Action are included in the component diagram of the Hand Gesture Controlled Virtual Mouse.

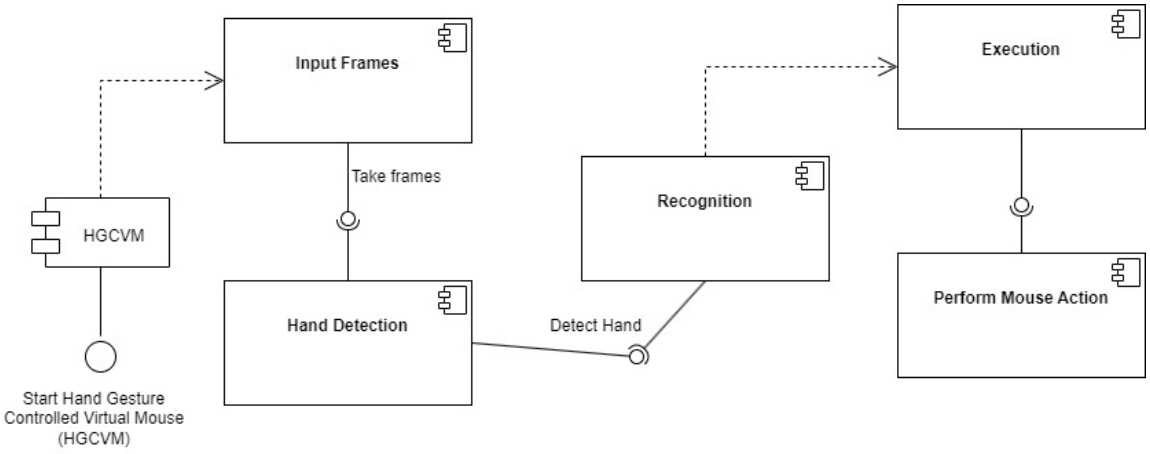

Figure 3. 6 Component Diagram of the ***Hand Gesture Controlled Virtual Mouse***

3.2.3 Deployment Diagram

The deployment diagram shows how different parts of the system are distributed across various devices. It highlights the interaction between the user's device, the hand tracking system, and the gesture recognition system. There are three main nodes in the diagram:

- **User Device**: The device the user uses to interact with the Hand Gesture Controlled Virtual Mouse system.

- **Gesture Recognition System**: The system that recognizes hand gestures.
- **Hand Tracking System**: The component, like a camera or sensor, that captures the user's hand movements.

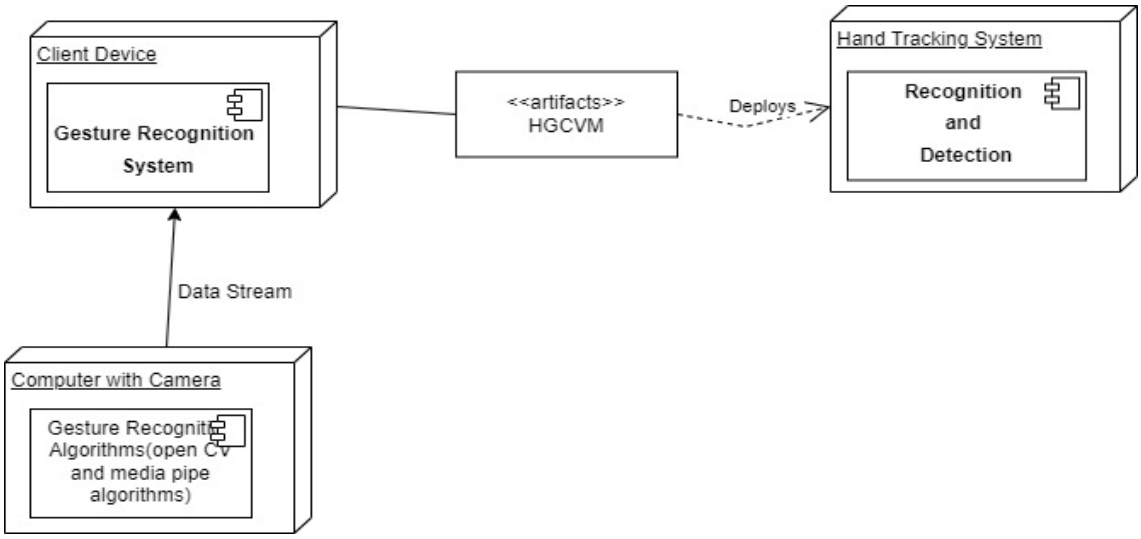

Figure 3. 7 Deployment Diagram of the Hand Gesture Controlled Virtual Mouse

3.3 Algorithm Details

In the Hand Gesture Controlled Virtual Mouse system, Convolutional Neural Networks (CNNs) are used to recognize complex hand gestures by processing video frames in real-time. The CNN identifies specific landmarks on the hand, allowing for accurate gesture interpretation and smooth control of the mouse cursor on the screen.

How the System Works:

1. **Hand Detection:**
    - **Data Preparation:** A large dataset of images with and without hands is collected. These images are annotated to highlight the areas where hands are present (called Regions of Interest or ROIs).
    - **Training:** A CNN model, such as an SSD or Faster R-CNN, is trained to detect hands within video frames and draw bounding boxes around them.
    - **Real-time Detection:** During real-time use, the CNN processes each video frame to identify and outline any hands present.
2. **Landmark Localization:**
    - **Data Collection:** A separate dataset is used to mark specific points on the hand, like fingertips and knuckles.

- **Training:** Another CNN model is trained to locate these landmark points within the detected hand region.
- **Real-time Localization:** After detecting a hand, the model predicts the exact positions of the landmarks, which represent the hand's pose.

Algorithm Steps:

1. Collect and prepare a dataset of hand gesture images.
2. Split the dataset into training, validation, and test sets.
3. Design the CNN architecture with convolutional and fully connected layers.
4. Compile the model with suitable loss and optimization functions.
5. Train the CNN on the training data, using data augmentation to improve performance.
6. Validate the model on the validation set.
7. Test the model on a separate test set for real-world performance.
8. Deploy the trained CNN into the virtual mouse system.
9. Continuously capture and process hand gestures in real-time.
10. Use the CNN to recognize gestures and control the virtual mouse.
11. Keep the system running to ensure continuous gesture recognition and mouse control.

This system uses CNNs effectively to detect hands, locate landmarks, and interpret gestures, allowing users to control a virtual mouse intuitively and in real-time.

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Implementation

4.1.1 Tools Used

Programming Language:

- **Python**: Python was used to develop the "Hand Gesture Controlled Virtual Mouse" system. Its flexibility, wide range of libraries, and user-friendly nature made it the perfect choice for this complex project.

Libraries and Frameworks:

- **MediaPipe**: MediaPipe, developed by Google, is an open-source framework designed to create real-time computer vision applications. It includes tools for object detection, pose estimation, hand tracking, and more.

Use of MediaPipe in the Project:

- **Integration**: MediaPipe was integrated into the project's code, with the necessary dependencies set up to use its features.
- **Hand Detection**: MediaPipe's hand detection module was used to identify and track hands in real-time video or image streams using machine learning algorithms.
- **Landmark Extraction**: After detecting hands, MediaPipe extracted specific hand landmarks, like finger joints and the palm center, to track hand movements.
- **Gesture Recognition**: The system used the hand landmarks to recognize and interpret different gestures, enabling it to understand user commands.
- **Interaction with Virtual Mouse**: Recognized gestures were translated into mouse actions, such as moving the cursor or clicking, allowing users to control the virtual mouse with their hand gestures.

Overall, MediaPipe was essential for real-time hand detection, landmark extraction, and gesture recognition, enabling intuitive control of the virtual mouse through hand movements.

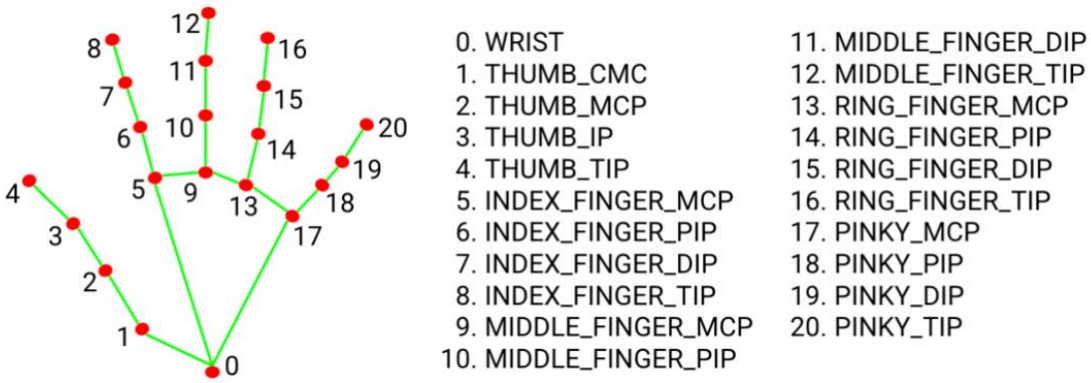

Figure 4. 1 Hand Coordinates of Landmarks

OpenCV: In the Hand Gesture Controlled Virtual Mouse project, OpenCV was used for various computer vision tasks.

1. **Hand Detection**: OpenCV's algorithms identified and located hands in video or images using techniques like background subtraction, motion detection, and skin color segmentation.
2. **Hand Landmark Extraction**: After detecting hands, OpenCV extracted key landmarks, such as finger joints and the palm center.
3. **Gesture Recognition**: Custom algorithms or machine learning models in OpenCV analyzed these landmarks to recognize different hand gestures.
4. **Image Processing**: OpenCV's image processing and filtering techniques enhanced the quality of the input, improving hand detection and gesture recognition accuracy.
5. **Mouse Control**: OpenCV translated hand gestures into mouse movements and actions, enabling users to control the virtual mouse with their hands.

Overall, OpenCV provided essential tools for hand detection, landmark extraction, gesture recognition, and image processing, making hand gesture control of the virtual mouse accurate and intuitive.

PyAutoGUI: PyAutoGUI controlled the virtual mouse cursor and simulated mouse actions by specifying coordinates, allowing precise cursor movements that matched the recognized hand gestures. This was crucial for translating hand gestures into virtual mouse actions.

Various other Python libraries were also used for additional functionalities.

4.1.2 Implementation Details of Modules (Description of Procedures/functions)

Different modules of this system are described as below:

a. Hand Detection and Landmark Extraction:

The code for hand detection and landmark extraction was implemented within the GestureController class. The classify_hands() method was utilized to assign hand landmarks to the major and minor hands based on the classification obtained from MediaPipe. Within the start() method, video frames were captured and processed using MediaPipe to obtain hand landmarks. The handmajor and handminor objects of the HandRecog class were utilized for finger state detection, gesture recognition, and gesture handling. The detected gestures were then handled accordingly.

Code:

```
class HandDetection:
    def __init__(self):
        self.hands = mp_hands.Hands(
            max_num_hands=2, min_detection_confidence=0.5,
min_tracking_confidence=0.5
        )
    def detect_hands(self, image):
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = self.hands.process(image_rgb)
        return results
    def draw_landmarks(self, image, hand_landmarks):
        mp_drawing.draw_landmarks(image, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
```

Listing 4. 1 Listing of Hand Detection and Landmark Extraction

b. Gesture Recognition:

Implement custom algorithms or use machine learning models to recognize and interpret hand gestures based on the extracted landmarks. The HandRecog class encompasses several methods related to gesture recognition. These methods include update_hand_result, get_signed_dist, get_dist, get_dz, set_finger_state, get_gesture, and handle_controls. These methods are responsible for processing the hand landmarks obtained from MediaPipe and mapping them to

recognizable gestures based on the defined enumeration (Gest). The get_gesture method returns the recognized gesture based on the hand landmarks.

Code:

```
# Gesture Encodings
class Gest(IntEnum):
    # Enumerations for different gestures
# Multi-handedness Labels
class HLabel(IntEnum):
    # Enumerations for minor and major hands
class HandRecog:
    # Class for converting Mediapipe landmarks to recognizable gestures
class Controller:
    # Class for executing commands based on detected gestures
class GestureController:
    # Class as the entry point of the program for gesture control
```

Listing 4. 2 Listing of Gesture Recognition module

c. Virtual Mouse Control:

Use PyAutoGUI library to control the virtual mouse cursor. It enables controlling the computer mouse using hand gestures. It utilizes the Mediapipe library for hand tracking and gesture recognition. The module includes classes such as HandRecog for converting Mediapipe landmarks to recognizable gestures, Controller for executing commands based on detected gestures, and GestureController as the main entry point for the module. The code implements functionality for recognizing gestures like fist, palm, V-gesture, pinch gestures, and more. It allows controlling mouse movement, left and right-click actions, scrolling, and adjusting system brightness and volume based on specific gestures detected from hand movements.

```
class Controller:
        def getpinchylv(hand_result):
        """returns distance between starting pinch y coord and current hand position
y coord."""
        dist = round((Controller.pinchstartycoord - hand_result.landmark[8].y) * 10,
1)
        return dist
 # ... (other methods and attributes)
class GestureController:
     def __init__(self):         """Initializes attributes."""
    def classify_hands(results):    """Classifies the detected hands into major and
minor hands."""
```

Listing 4. 3 Listing of Virtual Mouse control module

d. Continual Tracking and Execution:

Continuously track hand gestures in real-time for seamless control of the virtual mouse.

The get_signed_dist and get_dist methods in the HandRecog class are used to calculate the distances between two specified landmark points on a hand. These methods take a point parameter, which consists of two elements representing the indices of the landmark points.

```
def __init__(self):
        """Initilaizes attributes."""
        GestureController.gc_mode = 1
        GestureController.cap = cv2.VideoCapture(0)
        GestureController.CAM_HEIGHT                                    =
GestureController.cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
        GestureController.CAM_WIDTH                                     =
GestureController.cap.get(cv2.CAP_PROP_FRAME_WIDTH)
        def classify_hands(results):
        left, right = None, None
        try:
           handedness_dict = MessageToDict(results.multi_handedness[0])
           if handedness_dict['classification'][0]['label'] == 'Right':
```

```
            right = results.multi_hand_landmarks[0]
        else:
            left = results.multi_hand_landmarks[0]
    except:
        pass
    try:
        handedness_dict = MessageToDict(results.multi_handedness[1])
        if handedness_dict['classification'][0]['label'] == 'Right':
            right = results.multi_hand_landmarks[1]
        else:
            left = results.multi_hand_landmarks[1]
    except:
        pass
            if GestureController.dom_hand == True:
        GestureController.hr_major = right
        GestureController.hr_minor = left
    else:
        GestureController.hr_major = left
        GestureController.hr_minor = right
```

Listing 4. 4 Listing of Continual Tracking and Execution Module

4.2 Testing

Test Case: Hand Detection

Table 4. 1 Test Case Hand detections

Input	**Expected Output**	**Result**
Showing Neutral gesture	Curser should be stable	Pass
Showing Two finger gesture	Curser should move	Pass
Extract Hands	Hand coordinates showing	Pass
Show back side of hand	No detection	Pass

Test Case: Test Gesture Recognition functionality

Table 4. 2 test case on Gesture functionality of HGCVM

Description	Input (Simulated Hand Landmarks)	Expected Result	Output	Pass/Fail
Test recognition of a Palm gesture	Hand landmarks resembling a palm hand gesture	Gesture recognized as "PALM"	Gesture recognized: PALM	Pass
Test recognition of a Fist gesture	Hand landmarks resembling a fist hand gesture	Gesture recognized as "FIST"	Gesture recognized: FIST	Pass
Test recognition of a Two Finger Closed gesture	Hand landmarks resembling two fingers being closed	Gesture recognized as "TWO_FINGER_CLOSED"	Gesture recognized: TWO_FINGER_CLOSED	Pass
Test recognition of a V Gesture	Hand landmarks resembling a V hand gesture	Gesture recognized as "V_GEST"	Gesture recognized: V_GEST	Pass
Test recognition of a Pinch Major gesture	Hand landmarks resembling a major pinch gesture	Gesture recognized as "PINCH_MAJOR"	Gesture recognized: PINCH_MAJOR	Pass

4.3 Result Analysis

The effectiveness of the system was confirmed by conducting unit tests. The outcomes demonstrated that the project accomplished its objectives, although there is potential for enhancing the system's features.

4.3.1 Evaluating Accuracy

Accuracy Evaluation:

Gesture Recognition Accuracy: The accuracy of the gesture recognition heavily depends on the underlying hand tracking and gesture recognition algorithms that are mp.solutions.hands and mp.solutions.drawing_utils provided by the Mediapipe library. If the hand tracking is accurate and the landmarks are detected reliably, the accuracy of the system can be high. However, the accuracy might decrease in challenging scenarios, such as varying lighting conditions, different hand orientations, and occlusions.

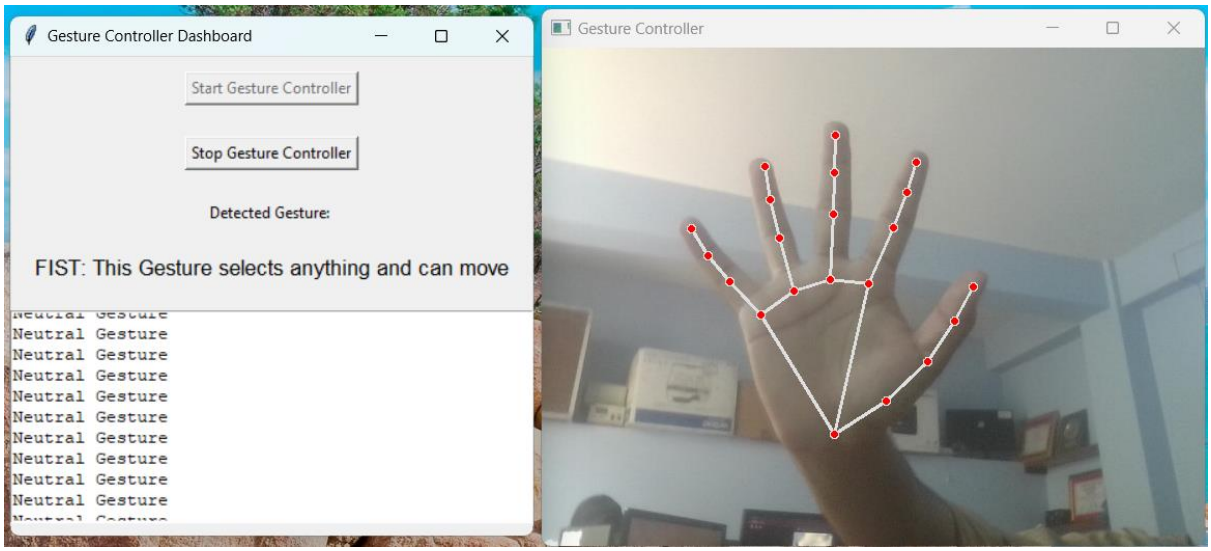

Figure 4. 2 Neutral Gesture showing in HGCVM

The above figure shows HGCVM is on neutral position as, the current used landmarks is used as a neutral gesture for controlling the virtual mouse.

Noise Handling: The HandRecog class attempts to handle noise by calculating ratios and distances between landmarks. This can help mitigate fluctuations caused by small hand movements. However, the effectiveness of noise reduction depends on how well these calculations model actual hand gestures and how sensitive the system is to changes.

Cursor Movement Accuracy: The accuracy of cursor movement depends on how well the hand positions are translated to screen coordinates. Minor inaccuracies in hand tracking can result in imprecise cursor movement.

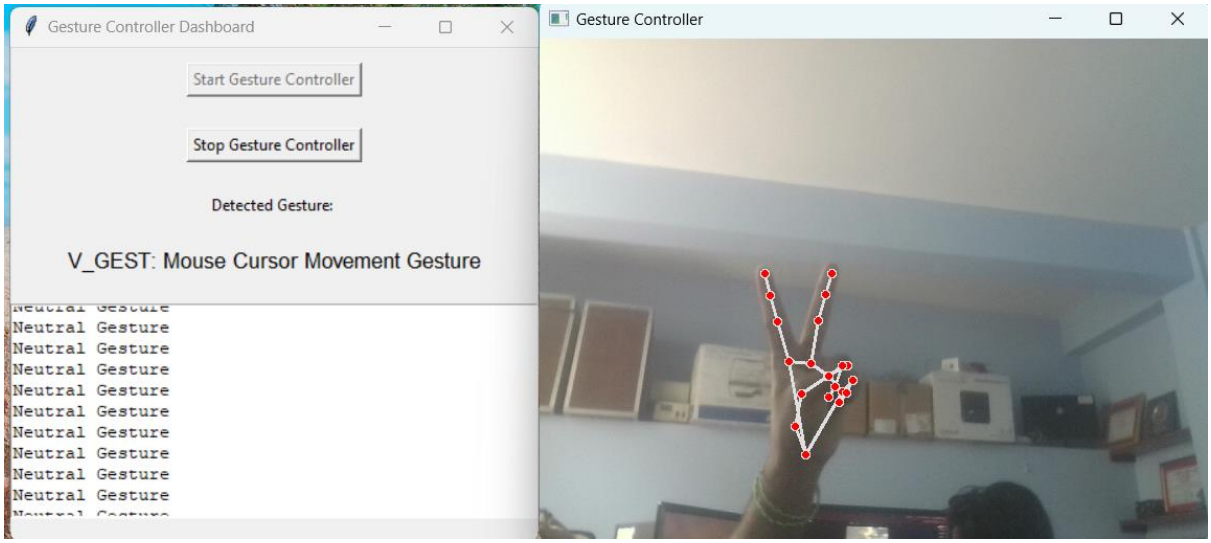

Figure 4. 3 Mouse Gesture as movement on the cursor in HGCVM

In the above picture, moving hand gesture is shown. It is a gesture shown as in a mouse's cursor is in moving position with index finger placed in like in left click and middle finger in right click. Moving these fingers without folding any of these two fingers keep moving the cursor in the screen.

Interaction Execution: The Controller class handles various interactions, such as clicking, scrolling, and system control. The accuracy of these interactions depends on the implementation and how well they map to user expectations.

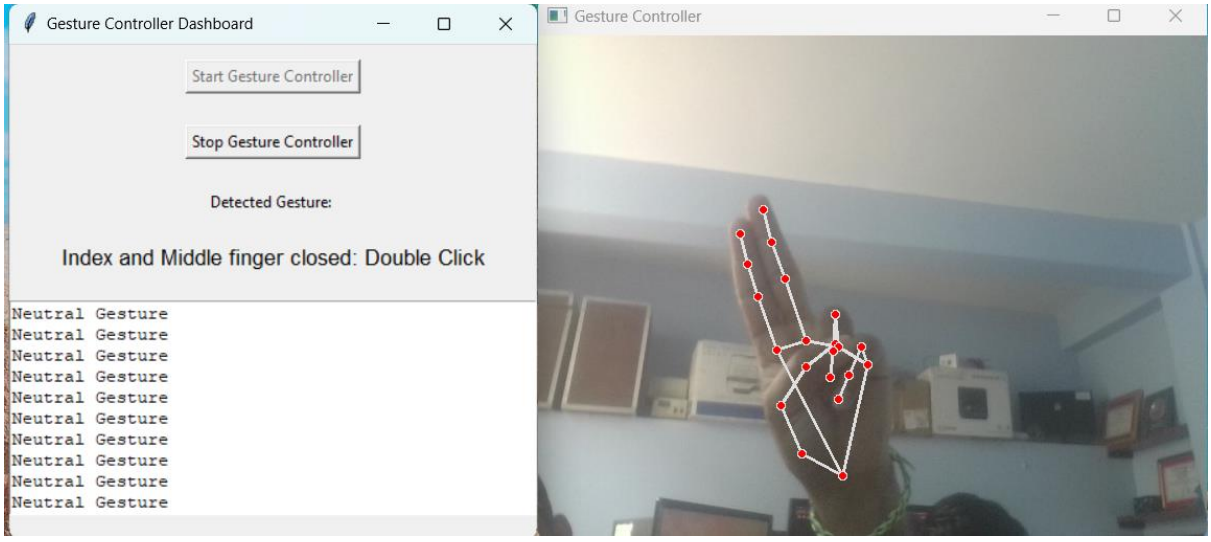

Figure 4. 4 Double Click operation in HGCVM

Above picture shows double click operation done joining the index and middle finger together. It is same gesture as clicking left button of mouse twice. As seen on the picture above the text "text" is selected as the two fingers are joined.

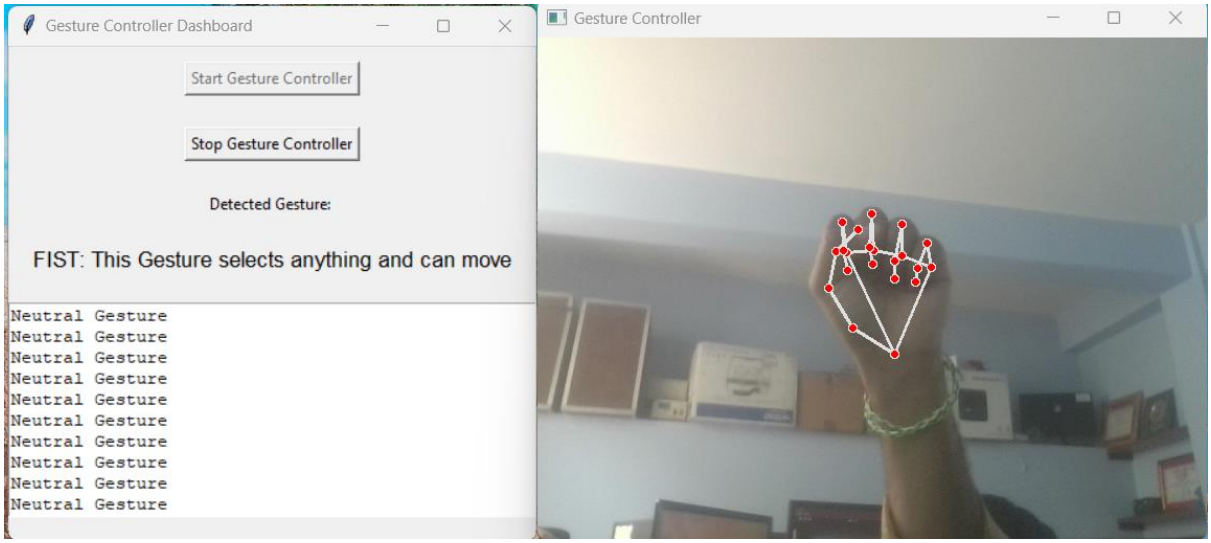

Figure 4. 5 Double Click and Scroll Operation in HGCVM

The figure above demonstrates select and scroll operation as the fist is made and scrolled. It is same as double clicking the mouse and scrolling to the bottom or some desired user location.

System Control: The system control actions, such as adjusting brightness and volume, might work accurately if the corresponding libraries (screen_brightness_control and pycaw) are reliable and responsive.

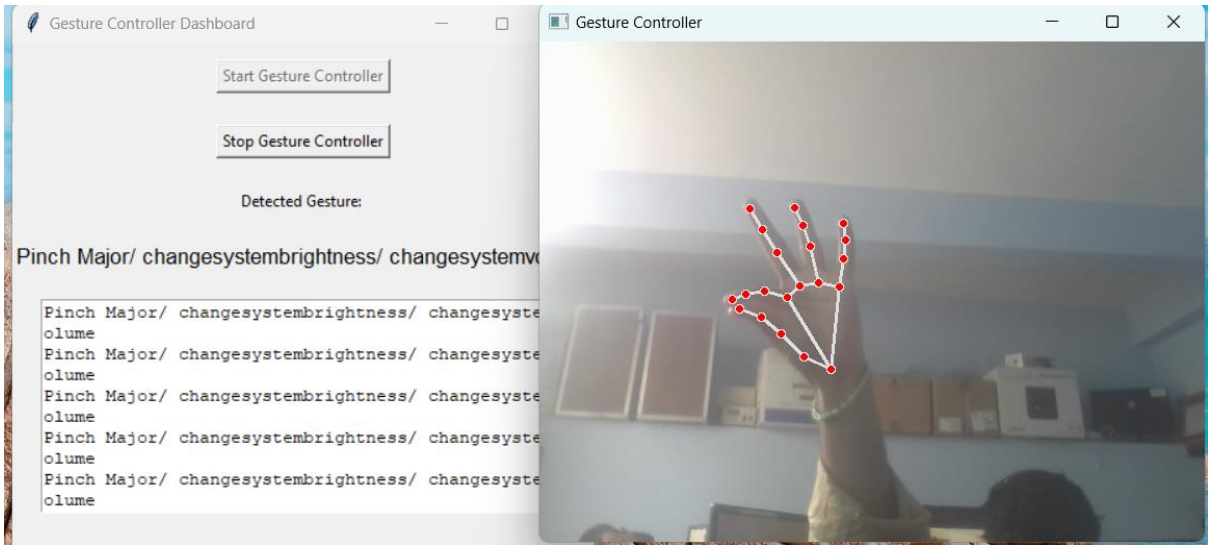

Figure 4. 6 Volume Control with Pinch gesture in HGCVM

The picture above demonstrates how system controls is done using this HGCVM as this pinch gesture is used the volume can be increased by moving the hand to right and decreased by moving the hand to left while still in the pinch gesture. The same process can be done to the brightness setting.

4.3.2 Experimental Result and Accuracy Calculation

Cross comparison of the testing of the AI virtual mouse system is difficult because only limited numbers of datasets are available. The hand gestures detection have been tested in various illumination conditions and also been tested with different distances from the webcam for tracking of the hand gesture and hand detection. An experimental test has been conducted to summarize the results shown in Table 4.3. The test was performed 25 times by 4 persons resulting in 600 gestures with manual labelling, and this test has been made in different light conditions and at different distances from the screen, and each person tested the AI virtual mouse system 10 times in normal light conditions, 5 times in faint light conditions, 5 times in close distance from the webcam, and 5 times in long distance from the webcam, and the experimental results are tabulated in Table 4.3.

Table 4.3 also suggests that the system has an average accuracy of 99%. Further details can be seen in the table itself which is tabulated below:

Table 4. 3 Experimental Results

Hand gesture	Mouse function performed	Success	Failure	Accuracy (%)
Tip ID 1 or both tip IDs 1 and 2 are up	**Mouse movement**	100	0	100
Tip IDs 0 and 1 are up and the distance between the fingers is <30	**Left button click**	99	1	99
Tip IDs 1 and 2 are up and the distance between the fingers is <40	**Right button clicks**	95	5	95
Tip IDs 1 and 2 are up and the distance between the fingers is >40 and both fingers are moved up the page	**Scroll up**	100	0	100
Tip IDs 1 and 2 are up and the distance between the fingers is >40 and both	**Scroll down**	100	0	100

fingers are moved down the page				
All five tip IDs 0, 1, 2, 3, and 4 are up	**No action performed**	100	0	100
Result		594	6	99
*Finger tip ID for respective fingers: tip Id 0: thumb finger; tip Id 1: index finger; tip Id 2: middle finger; tip Id 3: ring finger; tip Id 4: little finger. The finger tip information are given in Figure 4.1				

CHAPTER: 5 CONCLUSION AND FUTURE RECOMMENDATIONS

5.1. Lesson Learnt / Outcome

During the development of the Hand Gesture Controlled Virtual Mouse project, several important lessons were learned, leading to significant outcomes. One key lesson was the importance of strong hand detection, which is crucial for the system to work well, especially in different lighting conditions. Balancing accuracy with quick response times was also essential to ensure the system responded promptly to user gestures while maintaining precise tracking.

The project emphasized the value of iterative development and user testing. Regular feedback helped identify areas for improvement, leading to ongoing enhancements in performance and usability. Effective error handling was also critical, allowing the system to manage situations like no hand detection or multiple hands being detected smoothly.

The project successfully created a system that allowed users to control a virtual mouse with hand gestures. The system accurately detected hand movements, recognized gestures, and translated them into corresponding actions, all in real time. It also proved adaptable across different users, accommodating various hand sizes and gestures through calibration.

Overall, the project enhanced the user experience by enabling intuitive interaction with digital content using hand gestures, without the need for physical devices. The knowledge gained from this project lays the groundwork for future applications, including in gaming, virtual reality, augmented reality, and other interactive interfaces. The project demonstrated the potential of hand gesture control as a viable alternative input method, opening up new possibilities in human-computer interaction.

5.2. Conclusion

In conclusion, the Hand Gesture Controlled Virtual Mouse project successfully showed that hand gestures can be an effective way to control a virtual mouse. By combining technologies like hand detection, gesture recognition, and virtual mouse control, the project created an intuitive and immersive way for users to interact with digital content. Key lessons included the need for strong hand detection, balancing accuracy with responsiveness, and the value of iterative development with user testing.

The project delivered a system that accurately translates hand movements into virtual mouse actions, offering real-time responsiveness and adaptability to different hand sizes and gestures. This personalized user experience highlights the potential of hand gesture control as a new input method. The success of this project also opens the door for further exploration in areas like gaming, virtual reality, and interactive digital interfaces.

Overall, the Hand Gesture Controlled Virtual Mouse project achieved its goals, demonstrating the potential of gesture-based control systems and advancing the fields of computer vision, machine learning, and user interface design.

5.3. Future Recommendations

To improve the gesture control project, several enhancements can be made. Adding advanced gesture recognition techniques would allow the system to recognize more complex hand movements. Users could customize gestures to suit their preferences, and dynamic gesture adjustments could boost accuracy in different conditions. Providing visual feedback on recognized gestures would also enhance the user experience.

Other improvements include introducing user profiles for personalized controls, integrating multi-modal inputs like voice commands or facial recognition, and strengthening error handling for a more robust system. Additionally, developing a mobile application, contributing to open-source, and addressing security and privacy concerns would further enhance the project.

Key Recommendations:

- Implement Advanced Gesture Recognition

- Gesture Feedback and Visualizations

- User Profiles and Settings

- Multi-Modal Control

- Error Handling and Robustness

- Mobile Application Integration

- Open-Source Contribution

- Security and Privacy Considerations

REFERENCES

[1] Hindusthan College of Engineering and Technology, "Hand Gesture Recognition for Virtual Mouse Control."

[2] Bannari Amman Institute of Technology, "Hand Gesture Controlled Virtual Mouse Using Artificial Intelligence."

[3] V. V. Reddy, "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition."

[4] International Research Journal of Engineering and Technology, "Virtual Mouse Control Using Hand Gesture Recognition."

[5] "Curvature of Perimeter: A New Approach for Hand Gesture Recognition."