COMPUTER VISION BASED REAL-TIME HAND TRACKING SYSTEM FOR CONTACTLESS INTERACTION

A PROJECT REPORT

Submitted by

JAGADISH K

VAISHNAVANAMBI S V

SARAVANASETHU G

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER TECHNOLOGY

MIT CAMPUS

ANNA UNIVERSITY : CHENNAI 600 044

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 044

BONAFIDE CERTIFICATE

Certified that this project report, "**COMPUTER VISION BASED REAL-TIME HAND TRACKING SYSTEM FOR CONTACTLESS INTERACTION**" is the bonafide work of "**JAGADISH K (2019503018), VAISHNAVANAMBI S V (2019503058), and SARAVANASETHU G (2019503046)**" who carried out the project work under my supervision.

SIGNATURE
Dr. P. JAYASHREE
HEAD OF THE DEPARTMENT
Professor
Department of Computer Technology
Anna University, MIT Campus
Chromepet, Chennai- 600044

SIGNATURE
Dr. R. KATHIROLI
SUPERVISOR
Assistant Professor
Department of Computer Technology
Anna University, MIT Campus
Chromepet, Chennai- 600044

ABSTRACT

In the wake of the COVID-19 pandemic, contactless interaction has become essential to maintain hygiene and safety. Traditional methods of interacting with computers, such as mouse and keyboards, pose a potential risk of infection. In response to the increasing demand for contactless interaction, we propose a computer vision-based real-time hand tracking system with a gesture recognition module for creating virtual mouse and keyboard. The system tracks the hands in real-time using a hand detection module, followed by hand gesture recognition to control a computer virtually. Keyboard features, mouse cursor movement and click events are implemented to provide complete control of the computer through hand gestures. The proposed system aims to provide a cost-effective and efficient solution that can be easily integrated into existing devices or systems. Its significance lies in its potential to be widely adopted, particularly in industries where contactless interaction is critical, to mitigate the spread of infections. In addition to contactless interaction, our proposed method provides additional features for convenient and efficient device control, including the ability to control system volume and perform window management operations.

ACKNOWLEDGEMENT

We are highly indebted to our respected Dean, **Dr. J. PRAKASH,** and our respected Head of the Department **Dr. P. JAYASHREE** for providing sufficient facilities that contributed to the success of the project.

We express our deep sense of gratitude to our respected supervisor **Dr. R. KATHIROLI**, Assistant Professor, Department of Computer Technology, Anna University, MIT Campus for her guidance, vision, and constant encouragement throughout our project.

We thank our panel members **Dr. C. VALLIYAMMAI**, **Dr. P. PABITHA**, and **Dr. S. MUTHURAJKUMAR** for their valuable comments and their different views on our project.

Finally, we thank all the teaching and non-teaching members of the Department of Computer Technology, library staff, friends, family members, seniors, and everyone who has bestowed us with their knowledge directly or indirectly, intentionally or unknowingly.

JAGADISH K (2019503018)

VAISHNAVANAMBI S V (2019503058)

SARAVANASETHU G (2019503046)

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	ACKNOWLEDGEMENT	**iv**
	LIST OF TABLES	**viii**
	LIST OF FIGURES	**ix**
	LIST OF ABBREVIATIONS	**x**
1	**INTRODUCTION**	**1**
	1.1 OVERVIEW	1
	1.2 HUMAN-COMPUTER INTERACTION	1
	1.3 TOUCHLESS INTERACTION	2
	1.4 OBJECTIVES	3
	1.5 PROBLEM STATEMENT	3
	1.6 ORGANIZATION OF THESIS	3
2	**LITERATURE SURVEY**	**5**
	2.1 HAND TRACKING	5
	2.2 MOUSE	6
	2.3 KEYBOARD	8
	2.4 INFERENCE	9
3	**CONTACTLESS INTERACTION**	**11**
	3.1 PROPOSED WORKFLOW	11

CHAPTER NO.	TITLE	PAGE NO.
	3.2 ARCHITECTURE DIAGRAM OF HAND TRACKING SYSTEM	13
4	**ALGORITHM AND IMPLEMENTATION**	**15**
	4.1 HAND TRACKING	15
	4.2 VIRTUALIZING THE MOUSE	16
	4.3 VIRTUALIZING THE KEYBOARD	18
	4.4 HAND GESTURE RECOGNITION	20
	4.4.1 Volume Control	20
	4.5 TOOLS USED	21
	4.5.1 Hardware	21
	4.5.2 Software and Libraries Used	22
5	**RESULTS**	**23**
	5.1 HAND TRACKING	23
	5.2 VIRTUAL MOUSE	24
	5.3 VIRTUAL KEYBOARD	26
	5.4 HAND GESTURE RECOGNITION	27
	5.4.1 Volume Control	27
	5.4.2 Window Management Operations	29
6	**PERFORMANCE ANALYSIS**	31
	6.1 MOUSE ACTION PERFORMANCE	31
	6.2 KEYBOARD PERFORMANCE	32
	6.3 VOLUME CONTROL PERFORMANCE	33

CHAPTER NO.	TITLE	PAGE NO.
7	**CONCLUSION AND FUTURE PROSPECTS**	**34**
	7.1 CONCLUSION	34
	7.2 FUTURE PROSPECTS	34
	REFERENCES	**35**

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
6.1	Mouse Action Performance Testing Results	31
6.2	Keyboard Performance Testing Results	32
6.3	Volume Control Performance Testing Results	33

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	Architecture Diagram	14
5.1	Hand with 21 Landmarks	23
5.2	Hand Gestures	24
5.3	Moving Cursor	24
5.4	Left Click Action	25
5.5	Right Click Action	25
5.6	Virtual Keyboard	26
5.7	Corresponding Text in Notepad	26
5.8	Adjusting Volume Percentage	27
5.9	Volume Set	28
5.10	System Volume	28
5.11	Scroll Down	29
5.12	Scroll Up	29
5.13	Close Window	30
5.14	Minimize Window	30

LIST OF ABBREVIATIONS

BGR	Blue Green Red
CNN	Convolutional Neural Network
CSS	Cascading Style Sheet
FPS	Frames per Second
HCI	Human Computer Interaction
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ITR	Information Transfer Rate
OpenCV	Open-Source Computer Vision
RGB	Red Blue Green
VKB	Virtual Keyboard

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

With the rapid advancement of computer technology, the importance of Human-Computer Interaction (HCI) has grown significantly. HCI is a multidisciplinary field that focuses on the design of computer technology and the interaction between humans and computers. To enable productive interaction between users and computers, non-touch input systems that rely solely on cameras and eliminate the need for traditional devices such as keyboards and mouse have gained popularity.

One of the key components of human-computer interaction is vision-based hand gesture recognition technology. Gesture recognition enables users to interact with computers in a more natural way by leveraging their habitual hand movements. In today's world, virtual communication has become mainstream, and the COVID-19 pandemic has increased social isolation, which can have negative effects on mental health.

1.2 HUMAN COMPUTER INTERACTION

HCI is an interdisciplinary field that investigates the design, evaluation, and implementation of interactive computer systems for human use. It combines knowledge from computer science, psychology, design, and other fields to create interfaces that are both user-friendly and efficient. As more people rely on technology in their daily lives, HCI has grown in significance in recent years. It is crucial to make sure that these technologies are user-friendly, efficient, and satisfy their needs. To achieve this, user interface design is essential because it influences how users interact with technology and affects their overall experience. A variety of approaches and procedures are used in HCI research,

including as prototyping, usability testing, and cognitive modeling. These methods seek to comprehend how consumers engage with technology, pinpoint their requirements and needs, and create user interfaces that are specific to those demands.

1.3 TOUCHLESS INTERACTION

The concept of touchless interaction involves the use of various sensing technologies such as voice recognition, gesture recognition, eye tracking, and facial recognition. These technologies enable users to control devices and interfaces through natural body movements and verbal commands, without the need for physical buttons or touchscreens. Human factors and usability must be carefully considered while creating touchless interfaces. For instance, the system must be user-friendly and intuitive, as well as quick and precise enough to understand and carry out user commands. Additionally, during the design phase, privacy issues pertaining to the gathering and archiving of user data must be taken into consideration. One of the main advantages of this system is that it provides a contactless and hygienic interface, which is especially important in public settings such as hospitals, airports, and retail stores. Additionally, the system provides a more natural and intuitive way of interacting with devices and interfaces, which can improve user experience and accessibility. Touchless interaction using hand tracking represents a promising approach to human-computer interaction, providing a more convenient, hygienic, and efficient means of device control that could revolutionize the way we interact with technology. This technology offers several benefits, including more hygienic interaction with devices and a more accessible and user-friendly interface that can be used by a wider population, including those with physical disabilities. Moreover, it can be used in various settings, including public spaces, healthcare facilities, and manufacturing environments.

1.4 OBJECTIVES

- To create a hygienic contactless interaction between humans and computers.
- To develop a rapid hand-tracking system that can perform mouse and keyboard control.
- To incorporate window management operations by recognizing hand gestures for a user-friendly interaction.

1.5 PROBLEM STATEMENT

In recent years, touch-based interfaces have become increasingly popular in the form of mobile phones, tablets, and touch-screen kiosks. These interfaces provide a more intuitive and user-friendly experience than traditional input devices such as a keyboard and mouse. However, the size of the touch-screen directly affects the price of the device, with larger touch-screens being more expensive than smaller ones. As a result, this can limit the accessibility of these devices to a wider population.

Moreover, the COVID-19 pandemic has brought to light the need for hygienic and contactless interaction with devices. Touch-based interfaces such as ATMs and vending machines have been identified as potential mediums for virus transmission, leading to an increased demand for contactless interfaces. This has highlighted the need for new and innovative methods of human-computer interaction that are both hygienic and convenient.

1.6 ORGANIZATION OF THESIS

Chapter 1 provides an introduction to the thesis, including stating the primary objectives, the motivation behind the derivation of the problem statement and the organizational structure of the thesis itself. Chapter 2 provides a comprehensive literature review on the topics of hand tracking, mouse and keyboard control. It discusses the different approaches and methodologies used

in previous research and identifies the strengths and limitations of each approach. The chapter also reviews the gaps identified in the existing literature that this thesis aims to address. Chapter 3 outlines the proposed system architecture for touchless interaction using hand tracking and gesture recognition. It discusses the different components of the system, including the hardware and software requirements, and provides a detailed description of the workflow involved in the system. The chapter also presents an architecture diagram that illustrates the different stages of the system and the interactions between them. Chapter 4 goes into detail about the implementation and algorithms used in the proposed system. It discusses the hand tracking module and its integration with the virtual mouse and virtual keyboard. The chapter also covers the implementation of volume control and window management operations using hand gesture recognition. Chapter 5 presents the results of virtual mouse, virtual keyboard, volume control, and window management operations using hand gesture recognition output screenshots. Chapter 6 analyzes the performance of the virtual mouse, virtual keyboard, and volume control functions. It includes tables that present the results of testing for each module, including the number of successful attempts and error rate. Chapter 7 summarizes the key findings of the thesis. It also discusses the future work that can be done to improve the proposed system.

CHAPTER 2

LITERATURE SURVEY

2.1 HAND TRACKING

Liuhao *et al.* [10] conducted a study comparing the performance of 2D CNN and 3D CNN for hand gesture recognition. They found that 3D CNN outperformed 2D CNN due to its ability to capture 3D spatial information. In their approach, a volumetric representation of the hand in 3D space was generated from a depth image in real-time. The study evaluated three datasets: MicroSoft Research Asia, New York University, and Institute of Computer Vision and Graphics Laboratory, and two different 3D CNN architectures were employed - a 3D Shallow Plain Network and a 3D Deep Dense Network - with a loss function. The results demonstrated that the 3D CNN-based method could better utilize the depth information and provide more accurate estimation than the 2D CNN-based method.

A new authentication method called Doodling Authentication was proposed by Abdelghafar *et al.* [1] which involves the utilization of IoT devices to enable interaction between humans and smart objects. In order to tackle the challenges associated with dynamic hand gestures in the air and authentication verification, the researchers employed a Light-weight Convolution Neural Network (CNN) and a Kalman filter. The CNN model was utilized to enhance the accuracy of hand gesture recognition, while the Kalman filter was employed to correct the zigzags generated during drawing in the air. This innovative approach offers a more intuitive and natural method of authentication, surpassing the convenience and security provided by traditional password-based techniques.

The development of a visual hand tracking system that enables users to create texts, drawings, and erase contents by using their finger as a pen was proposed by Isaiah *et al.* [6]. This system utilized OpenCV and Mediapipe and consisted of two major processes: Color Thresholding and Mediapipe-based

Finger Tracking. By accurately detecting and tracking the hand as a marker or pen, users were able to generate drawings and handwritten content based on their movements.

Kirti *et al*. [8] developed a system that utilizes a webcam to track objects and record video images. The recorded images are then analyzed to identify the monitored object, and the program uses the image to track the object's positions in subsequent phases. The tracked positions are then used for further processing, such as mouse movement, clicks, and keyboard features. The extracted data can be used for a variety of purposes, including interactive gaming and virtual reality applications.

Kriznar *et al.* [15] developed two separate applications that utilized hand tracking technology. The first application aimed to enhance student-teacher collaboration by allowing students from remote locations to draw shapes and lines in real-time, which were displayed on both the teacher's and the student's computer screens. The application utilized Node.js for server connectivity and Google's MediaPipe module for hand tracking. The second application was a video game designed for young children, which helped them learn basic shapes, colors, and logical reasoning through play. Hand tracking was achieved through the use of the handtrack.js library, and the primary application was developed using HTML/CSS and JavaScript.

2.2 MOUSE

A method for controlling video players using hand gestures and image processing techniques was presented by Chinnam *et al.* [3]. The hand gestures were captured using a camera or a TOF camera and underwent pre-processing, which consisted of two steps: segmentation and morphological sifting. The extracted hand region was further processed for feature extraction to obtain the relevant features required for gesture recognition. This method enabled the retrieval of improved pictures or data from the captured hand gestures.

Koushik *et al.* [9] proposed a gesture-based HCI system that enables hand tracking and cursor movement based on hand movements. The system comprises three main stages: Palm Detection, Landmark Detection, and Detection of Index Finger Tip. Unlike traditional methods, this system does not record or save any video from the webcam, thereby minimizing data storage requirements. The system has potential applications in various fields, such as Automated Teller Machines (ATMs), hospitals, gaming industry, and Augmented Reality.

A technology that eliminates the need for any additional hardware to perform mouse operations was developed by Manav *et al.* [11]. The system was implemented using Python with OpenCV and PyAutoGUI libraries. Images are captured and pre-processed for range-based skin detection, and the face is removed using the Haar Cascade Classifier for hand gesture recognition. Hand gestures are recognized using two CNN layers followed by max pooling, with datasets containing six classes of hand gestures. The proposed system can recognize and interpret hand gestures as mouse actions, making it applicable in various domains such as gaming, virtual reality, and assistive technology.

Changhyun *et al.* [2] proposed a novel hand-mouse interface that utilizes a Kinect sensor to extract the user's physical characteristics in real-time and navigate a virtual environment on a "virtual monitor". The interface employs the user's natural gestures and is considered a Natural User Interface (NUI). The user's left hand controls the cursor movement while the right hand is used for click operations. The coordinates on the virtual display are accurately mapped to the coordinates on the physical monitor, allowing for seamless navigation of the virtual environment.

Dubbaka *et al.* [4] presented an innovative approach to human-computer interaction using hand gestures to control the mouse pointer on the screen. The proposed system utilized object identification and image processing techniques to capture hand motions on a webcam and translate them into cursor movements on the computer screen. The system also incorporated the ability to recognize additional hand gestures and draw point-based lines to move the cursor more

precisely. This system aimed to provide a simpler and more natural way of interacting with computers without the need for additional hardware like a mouse.

In the development of a real-time gesture recognition system, a dataset of labeled images of people making different gestures was collected by Sairam *et al.* [14]. By training a machine learning model on this dataset and applying transfer learning techniques, they enhanced the accuracy of the model. Integration of the model into the system allowed for real-time identification of the most probable gesture. To further improve the system's accuracy, centroid motion tracking was employed, and GUI mouse control automation was implemented.

2.3 KEYBOARD

Jungpil *et al.* [7] proposed a new method for inputting Japanese hiragana and English characters using hand tapping gestures. The system utilizes a Kinect sensor to obtain skeleton data and images, and the OpenCV function uses the Suzuki85 technique to detect contours for P-hand detection.

Hubert *et al.* [5] developed a virtual keyboard using portable and non-invasive eye trackers for people who are unable to communicate. The effectiveness of the virtual keyboard was measured by the speed and Information Transfer Rate (ITR) at both the command and application levels. The study found that participants typed at a rate of 18.43 letters per minute with a mouse alone, 15.26 letters per minute with an eye tracker and switch, and 9.30 letters per minute with an eye tracker alone. The ITR of the system was 44.96 bits/min and 57.46 bits/min at the letter and command levels, respectively. The results show that transitioning between several modalities can have a significant impact on performance.

A method to improve typing speed by using hand gesture recognition and a virtual keyboard that focuses on vowels was proposed [12]. This method, developed by Su-Jin *et al.* involves the utilization of five new gestures for vowels, along with two gestures called "CLICK" and "READY". By employing this approach, the time required for typing vowels and navigating between layers of

the layered keyboard layout is reduced. According to simulation data, the suggested approach resulted in a 23.07% increase in typing speed.

Tae-Ho *et al.* [13] developed an ambidextrous Virtual Keyboard (VKB) layout by extending their previous one-hand VKB layout. To save time and effort in dataset preparation, they generated an automatic training dataset. A hand segmentation algorithm was employed, and a deep learning-based Gesture Recognition approach was applied to the VKB algorithm to overcome the limitations of a color-based GR method, which does not work well when the background color is similar to the skin color. They used YOLOv3 to train the model, and the proposed DL-based GR method achieved a mean average precision result of 95% and a speed of 41 FPS.

2.4 INFERENCE

Based on the literature surveys, it can be inferred that hand tracking systems using techniques like 3D CNN and OpenCV have proven to be effective in enabling intuitive interaction with computers without the need for external devices. These studies demonstrate the feasibility of developing a hand gesture based control system for your project. By utilizing similar techniques such as 3D CNN and OpenCV, you can accurately detect and track hand movements, allowing users to perform actions like cursor movement, clicks, and even keyboard features using hand gestures. These findings suggest that implementing a hand tracking system would provide users with a more natural and immersive way of interacting with their computers.

Hand gesture based mouse control systems have been successfully developed by eliminating the need for traditional mouse devices. These studies show that it is possible to replace the conventional mouse with hand gestures. By leveraging the techniques and methodologies outlined in the literature, a hand gesture based mouse control system can be developed. This would allow users to control cursor movement, perform clicks, and potentially other mouse-related

functions using hand gestures, providing a more intuitive and hands-free interaction experience with the PC.

Hand gesture recognition can be employed for inputting characters and controlling virtual keyboards, eliminating the need for physical keyboards. By utilizing hand tapping gestures or recognizing specific hand gestures associated with characters or commands, users can input text and control virtual keyboards without relying on external physical keyboards. This allows for a more flexible and convenient interaction method, especially in scenarios where physical keyboards may not be accessible or desired.

CHAPTER 3

CONTACTLESS INTERACTION

3.1 PROPOSED WORKFLOW

The primary objective is to develop a touchless interface that enables seamless interaction between humans and computers using hand tracking technology. To achieve this objective, the system incorporates advanced hand tracking techniques. It utilizes real-time video streams and employs algorithms to detect and track hands. By identifying and tracking 21 key hand landmarks, the system can accurately interpret hand movements and gestures. With the hand tracking capability in place, users can control the cursor on the screen without the need for a physical mouse. They can navigate through the interface, select objects, and perform various mouse actions by simply moving their hands in the air. To further enhance the system's functionality, virtual buttons are integrated into the interface. These buttons replicate the functions of a traditional keyboard, allowing users to input text and interact with applications that require keyboard input. By recognizing hand gestures, the system can facilitate window management tasks, enabling users to manipulate and control windows without any physical touch.
The system is composed of the following modules:

i. Hand Tracking
ii. Virtualizing the Mouse
iii. Virtualizing the Keyboard
iv. Hand Gesture Recognition

i. Hand Tracking

The system starts by capturing real-time video streaming frames and then converting them into RGB frames. These RGB frames are then processed to detect the position of the palm and identify 21 landmarks in the hand for each frame. This allows for accurate tracking of the hand

movements. The landmark positions are constantly updated in real-time to ensure accurate tracking of the hand movements.

ii. Virtualizing the Mouse

The hand landmarks identified in each frame are used to control the cursor and perform mouse actions. Specifically, tracing landmark 8, which corresponds to the tip of the index finger, is used to control the cursor. Clicking actions are performed with the help of landmark 12, which corresponds to the tip of the middle finger. This enables users to interact with the computer without the need for a physical mouse.

iii. Virtualizing the Keyboard

Virtual buttons are designed to emulate a QWERTY keyboard, with each button having a specific position, text label, and size. The user can control the keyboard by hovering their index over the corresponding button and pressing down with the help of middle fingertip. This activates the button press and sends the corresponding keystroke to the computer. In this way, the user can control the keyboard and input text without any physical contact with the computer.

iv. Hand Gesture Recognition

In addition to controlling the cursor and emulating keyboard functions, the system is designed to recognize hand gestures for performing various tasks. For example, gestures for closing and minimizing an application. Gestures for scrolling and controlling system volume are also recognized by the system. These gestures enable the user to perform various tasks without any physical contact with the device, providing a touchless interface.

3.2 ARCHITECTURE DIAGRAM OF HAND TRACKING SYSTEM

The system architecture for the touchless interaction project involves several components and stages. The process begins with real-time video streaming, which captures frames of the user's hand. These frames are then processed and the palm of the hand is identified within the image. Landmarks are then rendered over the detected hand to provide reference points for further actions.

The detected landmarks in the hand are utilized to control the cursor and perform mouse actions. The index finger tip is used to trace the cursor's movement on the screen, allowing the user to navigate and interact with different elements. The middle finger tip is utilized to perform click actions, enabling the user to select items or activate functionality.

To create a virtual keyboard, the system incorporates buttons designed in a similar layout to the QWERTY keyboard. The tips of the index and middle fingers are used to press the virtual buttons, and the button press is recognized using the landmarks associated with the hand gestures. This enables the user to input text and control the virtual keyboard without the need for a physical keyboard.

Additionally, the system includes hand gesture recognition for window management operations such as closing applications, minimizing applications, and scrolling through content. The hand gestures are detected using the landmarks and the OpenCV library, allowing the user to perform these actions intuitively using their hand movements. Furthermore, the system integrates a volume control module that enables the user to adjust the system's volume using hand gestures. This functionality adds convenience and flexibility to the touchless interaction experience.

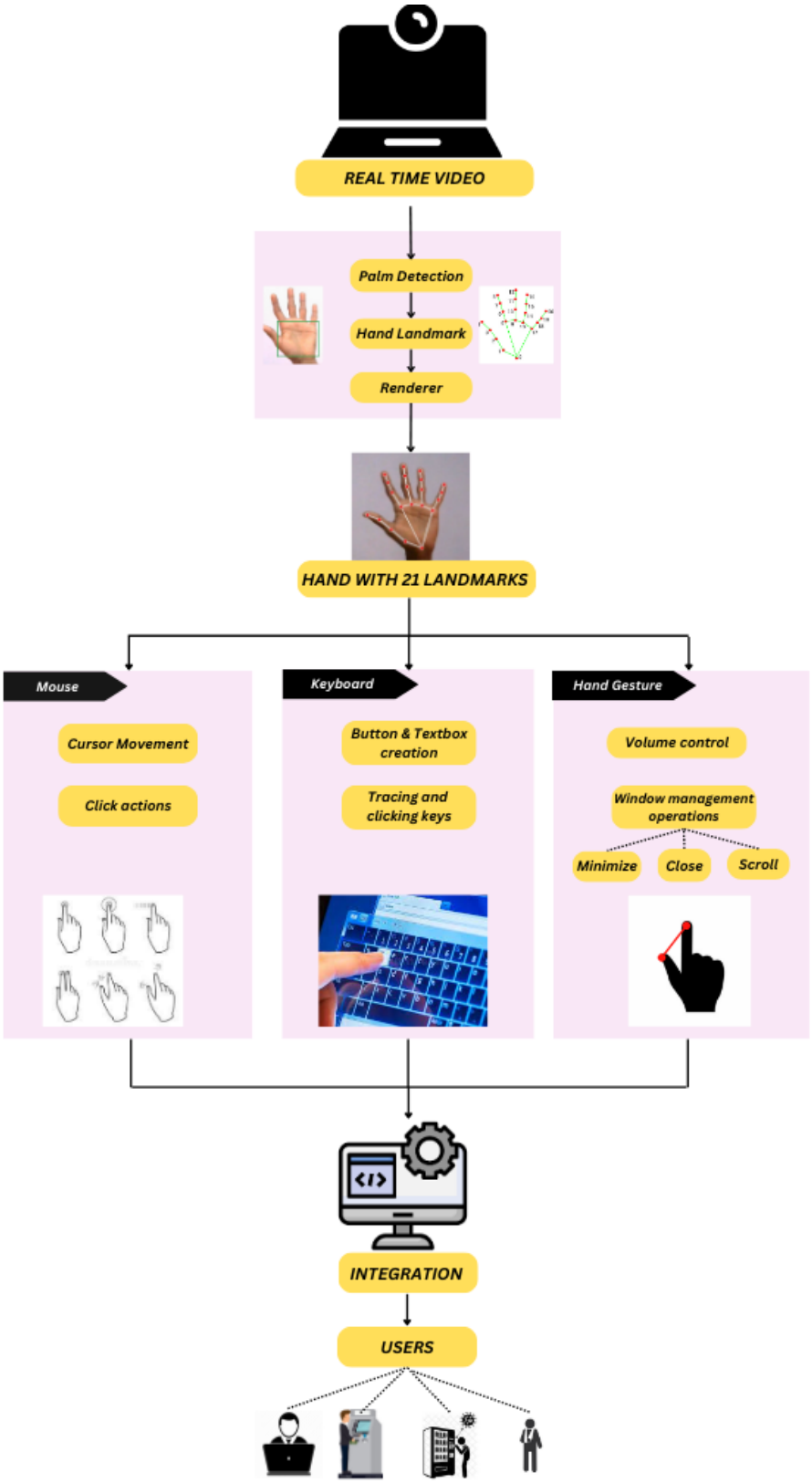

Figure 3.1 Architecture Diagram

Figure 3.1 illustrates the architectural flow, showcasing the different stages and interactions between the components involved in the system.

CHAPTER 4

ALGORITHM AND IMPLEMENTATION

4.1 HAND TRACKING

The initial stage is to detect and track the landmarks of the hand. OpenCV is used to process the frames from the camera and display the results. The HandDetector class provides a convenient interface to perform hand-tracking tasks.

Functions in the HandDetector class:

i. init(): This function initializes the HandDetector object with various parameters such as mode, maxHands, modelComp, detectionCon, and trackingCon.
ii. findHands(): This function takes an image as input and returns the image with the detected hand landmarks drawn on it.
iii. findPosition(): This function takes an image and a hand number as input and returns the landmark positions for the specified hand.
iv. findDistance(): This function takes two landmark positions as input and computes the Euclidean distance between them. It also draws the points and the line between them on the image.

$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 \quad (4.1)$$

v. fingersUp(): This function takes the landmark positions for a hand as input and returns a list of binary values representing whether each finger is up or down.

The main() function initializes the video capture device and the HandDetector object. It processes the frames from the camera and displays the video feed with the hand landmarks and the number of fingers up. It also calculates the frames per second (fps) and displays it on the screen.

Algorithm 4.1 discusses the process of tracking hands in real-time video and identifies landmarks.

Algorithm 4.1: Hand tracking Algorithm

Input: Real-time video

Output: 21 landmarks rendered frame

```
Begin
    V => videocap();                    // Input
    F_BGR = read_frame(V);              // Frame
    I_RGB = convert_color(F_BGR);       // RGB img
    H = find_hand(I_RGB);
    if H > 1:
for each hand H_0 ∈ H:
draw_landmarks(H_0);     // Output
end for
End
```

4.2 VIRTUALIZING THE MOUSE

The system provides a touchless interface that allows users to control the mouse virtually and perform various actions without physical contact. The video frame is bounded by a rectangle to provide a visual cue to the user to keep their hand within the frame. The system tracks the user's hand movements and uses the position of the index finger to move the cursor. To provide a smoother movement, the cursor position is interpolated based on the current and previous positions.

$$current\_loc = prev\_loc + (x - prev\_loc) / smooth \qquad (4.2)$$

In equation (4.2), smooth is a parameter that controls the smoothness of the cursor movement. The variable 'x' refers to the coordinate of the target position in the reduced frame, and similarly, 'y' would represent the corresponding vertical

coordinate. The equation calculates the new current_loc by gradually updating its position towards the target coordinates (x, y) based on the previous location (prev_loc) and the smooth parameter.

The system also allows for different mouse actions to be performed using specific finger gestures. A left-click action is executed when the index and middle fingers are both up and close together. A double-click action is triggered by holding the index and middle fingers close together for 2 seconds. Similarly, a right-click action is performed when the middle and ring fingers are both up and close together. Additionally, scrolling up and down is achieved by raising the pinky finger and thumb finger respectively.

Algorithm 4.2: Mouse Virtualization Algorithm

Input: Real-time video

Output: Mouse control and click actions

```
Begin
    V => videocap();                    // Input
    F_BGR = read_frame(V);              // Frame
    I_lm = find_hands(F_BGR);           // Img after rendering
    L = find_position(I_lm);            // List of 21 landmarks coordinates
    Draw_rect(fr);                      // Frame reduction
    Let finger => Finger status list;
    if finger[1]==0 and finger[2]==0:
        moveMouse(x, y);
    if finger[1]==1 and finger[2]==1:
        mouse.click();
    if finger[2]==1 and finger[3]==1:
        mouse.rightclick();
    if finger[0]==1 and finger[1]==0:
        scrolldown();
```

```
        if finger[1]==0 and finger[4]==1:
                scrollup();
        if finger==[1, 1, 0, 0, 1]:
                closeWindow();
        if finger==[0, 1, 0, 0, 1]:
                minimizeWindow();
    End
```

Algorithm 4.2 discusses the Mouse Virtualization Algorithm, which enables touchless mouse control through hand tracking and gesture recognition. It describes how the algorithm allows users to move the cursor, perform clicks, and scroll up and down using specific finger gestures. This algorithm provides a seamless touchless interface for intuitive mouse control.

4.3 VIRTUALIZING THE KEYBOARD

The keyboard can be virtualized and operated using hand landmarks captured by a webcam. An object of the HandDetector class from the HandTrackingModule is created, which is used to detect hands and hand landmarks in the video frames captured by the webcam. Button class is used to represent the buttons on the virtual keyboard. The drawButton() function is used to draw the buttons on the screen having properties such as position, text, and size for each button, and the buttonList list is used to store instances of the Button class. If a hand landmark is within the bounding box of a button, the button will be highlighted by drawing a rectangle around it and changes its color to gray. If the user performs a "click" gesture by bringing their index finger close to their middle finger, the corresponding button will be pressed. Some special buttons, such as the "Backspace" and "Enter" buttons are also included, which are used to simulate the corresponding keys on a keyboard. The "Reset" button is used to clear the text that has been entered so far.

Algorithm 4.3: Keyboard Virtualization Algorithm

Input: Real-time video

Output: Keyboard input

```
Begin
    V => videocap();                    // Input
    F_BGR = read_frame(V);              // Frame
    Button();                           // Create buttons of diff pos,text,size
    I_lm = find_hands(F_BGR);           // Img after rendering
    L = find_position(I_lm);            // List of 21 landmarks coordinates;
    Draw_button();
    if landmark[8] within boundary:
        Highlight the button
        l= find_distance(8,12);     // Dist b/w landmark 8 and 12
        text = “”;
        if l<40:
            if char== “Backspace”
                text= text[:-1];
            else if char== “Space”
                text+=“ ”;
            else if char== “Reset”
                text= “”;
            else
                text.append(char);
            pressKey();
End
```

Algorithm 4.3 discusses the Keyboard Virtualization Algorithm, which virtualizes and operates a keyboard using hand landmarks captured by a webcam. The algorithm detects hands and hand landmarks, creates virtual buttons for the

keyboard. The algorithm captures keyboard input based on hand movements, providing a touchless interface for typing and controlling the virtual keyboard.

4.4 HAND GESTURE RECOGNITION

4.4.1 Volume Control

The system's volume can be adjusted through a hand gesture recognition system that utilizes the HandTrackingModule to detect and track the user's hand and compute the distance between two fingers. This distance is then converted into a percentage to regulate the volume level. A polynomial function is employed to determine the distance between the hand and the camera, which helps to identify if the hand is at the appropriate distance for precise volume control.

$$y = Ax^2 + Bx + C \qquad (4.3)$$

The distance from the camera (y) can be determined by utilizing the known x value, which represents the distance between landmark 5 and landmark 17 in equation (4.3).

$$volPercent = smooth * round(volPercent / smooth) \qquad (4.4)$$

In equation 4.4, smooth parameter controls the smoothness of the volume adjustment. It determines the granularity of the volume levels. A higher value of smooth will result in larger steps between volume adjustments, while a lower value will provide finer steps. ‘volPercent’ represents the volume percentage, indicating the desired volume level for the system.

A green rectangle is displayed on the screen to indicate the maximum and minimum volume levels, with a vertical bar inside the rectangle showing the current volume level. The length or height of the vertical bar may be proportional to the volume level, providing a more precise visual representation.

Algorithm 4.4: Volume Control Algorithm

Input: Real-time video

Output: Volume adjustment

```
Begin
    V => videocap();                    // Input
    F_BGR = read_frame(V);              //Frame
    I_lm = find_hands(F_BGR);           // Img after rendering
    L = find_position(I_lm);            // list of 21 landmarks coordinates
    Dt = find_dist(I_lm);               // Distance from camera
    if 40 < Dt < 65:
        Let D_48 => distance b/w landmark 4 and 8;
        vp = interpolate(D_48, [minVol, maxVol]);
        if finger[4] == 0:
            setVolume(vp);      // Output
End
```

Algorithm 4.4 discusses the process of touchless volume control using hand gesture recognition. It tracks the user's hand, calculates the distance between fingers, and adjusts the volume based on the hand gesture. This provides a convenient and intuitive way to control the system's audio output without physical contact.

4.5 TOOLS USED

4.5.1 Hardware

A computer or laptop with a modern processor (e.g., Intel Core i5 or equivalent) and a minimum of 8GB of RAM is recommended. Additionally, a webcam or camera is required for capturing live video input.

4.5.2 Software and Libraries Used

Python: Python is a high-level programming language known for its simplicity and readability, making it beginner-friendly and widely used for various applications.

PyCharm: PyCharm is a popular Integrated Development Environment (IDE) specifically designed for Python. It offers advanced coding features, debugging tools, and project management capabilities to enhance productivity for Python developers.

OpenCV: OpenCV is a computer vision library that provides various functions for image processing, object detection, and tracking. It is used in this project for capturing video frames, displaying images, drawing annotations, and performing basic image operations.

Mediapipe: Mediapipe is a framework developed by Google that offers a wide range of pre-built solutions for various computer vision and machine learning tasks. In this project, the Mediapipe library is utilized for hand detection and tracking, enabling the identification of hand landmarks and gestures.

PyAutoGUI: PyAutoGUI is a cross-platform Python module that enables programmatically controlling the mouse. It provides functions to move the mouse cursor, simulate mouse clicks, and perform keyboard input simulation. In this project, PyAutoGUI is used to control the mouse movements and clicks based on the hand landmarks detected.

pynput.keyboard: The pynput.keyboard module is a Python library that allows controlling and monitoring keyboard inputs. It provides functionalities to simulate key presses, releases, and combinations. In this project, the pynput.keyboard module is used to simulate keyboard inputs based on the hand landmarks detected.

CHAPTER 5

RESULTS

5.1 HAND TRACKING

The frames obtained from the camera are processed by the hand detection algorithm, which locates and identifies the presence of hands in each frame. Once the hands are detected, the system further analyzes the hand regions and extracts 21 hand landmarks. In Figure 5.1, a visual representation is provided to illustrate the 21 hand landmarks. The figure showcases the detected hand with labeled landmarks, enabling users to understand the specific points that have been identified. The system calculates the Frames per Second (fps) to provide an indication of the real-time performance. The fps value represents the number of frames that can be processed or displayed in one second. It serves as a measure of how smoothly the system is running and indicates the speed at which the camera frames are being processed and analyzed. This information is then displayed on the screen, allowing users to monitor the system's performance and ensure that it meets their desired requirements.

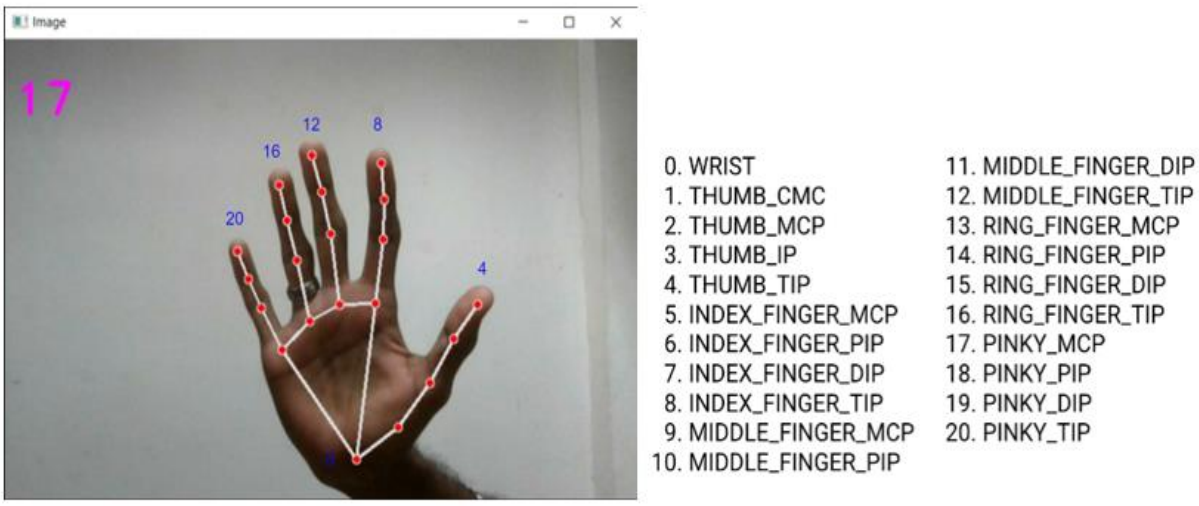

Figure 5.1 Hand with 21 Landmarks

5.2 VIRTUAL MOUSE

The virtual mouse system incorporates intuitive hand gestures that allow users to control the mouse cursor without the need for physical devices. Figure 5.2 visually represents these hand gestures, providing a clear illustration of how users can manipulate the virtual mouse using their hands.

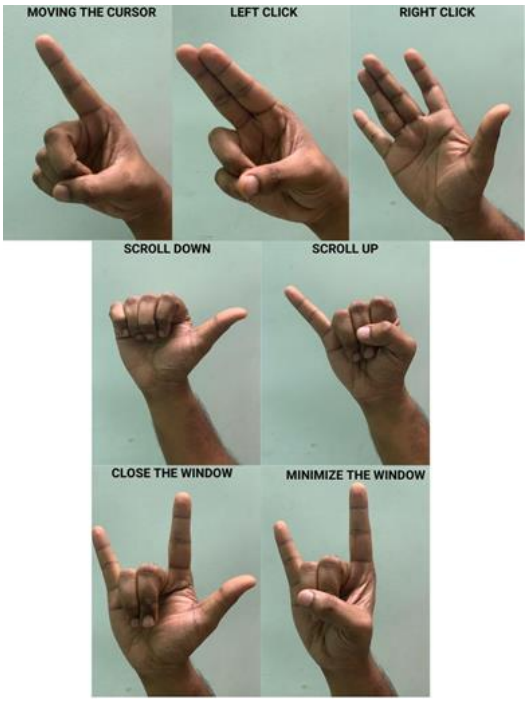

Figure 5.2 Hand Gestures

The index fingertip is used to move the cursor on the screen, and a blue rectangle is created as shown in Figure 5.3 to ensure that all corners of the screen can be reached.

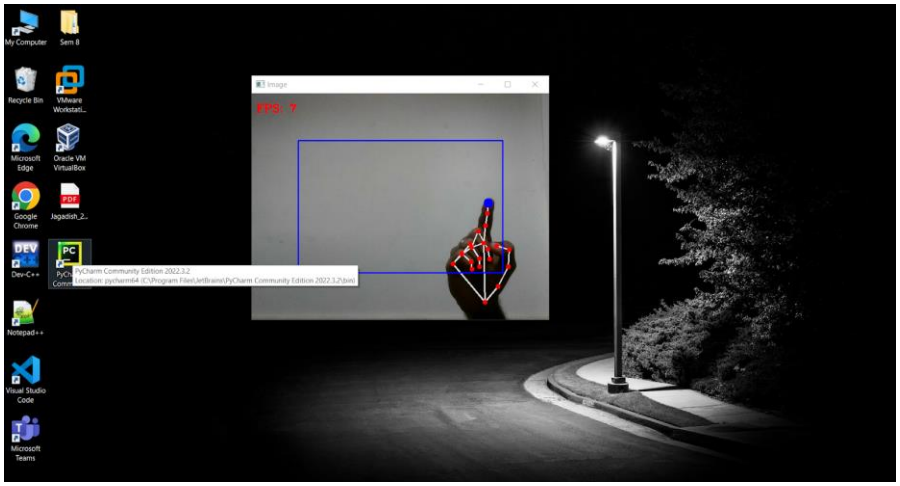

Figure 5.3 Moving Cursor

After reducing the frame and obtaining the region range, it is necessary to interpolate the range to fit the screen size accurately. This interpolation process ensures that the virtual mouse system can accurately translate the user's hand movements into corresponding cursor movements on the screen. Once the interpolation is complete, users can perform various mouse actions using specific hand gestures. Left-clicking is performed when the index and middle fingertips are close together, as shown in Figure 5.4. To initiate a double-click action, the user must hold their index and middle fingertips close together for 2 seconds.

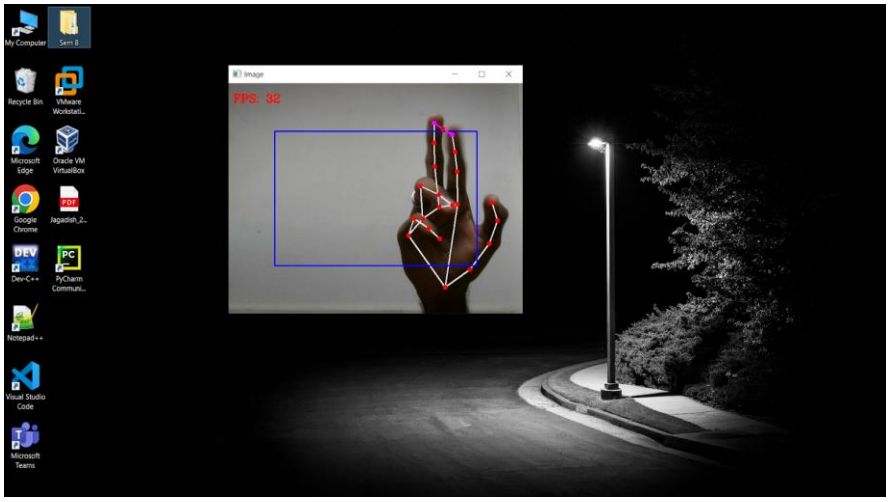

Figure 5.4 Left Click Action

A right-click action is performed when the middle and ring fingers are both up and close together, as shown in Figure 5.5.

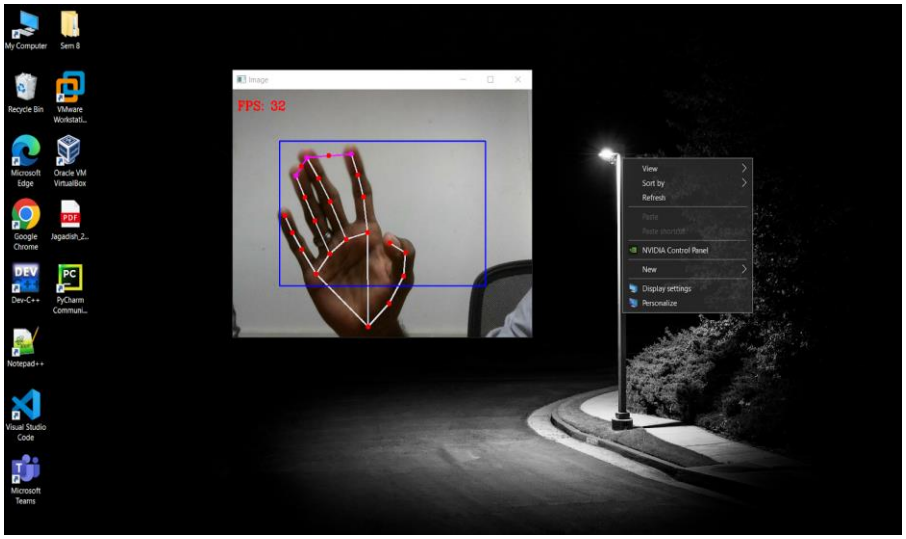

Figure 5.5 Right Click Action

5.3 VIRTUAL KEYBOARD

The virtual keyboard system utilizes the index fingertip as the primary tool for navigating and selecting characters. When interacting with the virtual keyboard, users can easily move their index finger across the virtual keys to find the desired character. The system employs a unique method to trigger the selection of a specific key. By bringing the index and middle finger tips in close proximity to each other, users activate the key associated with the currently selected character as shown in Figure 5.6.

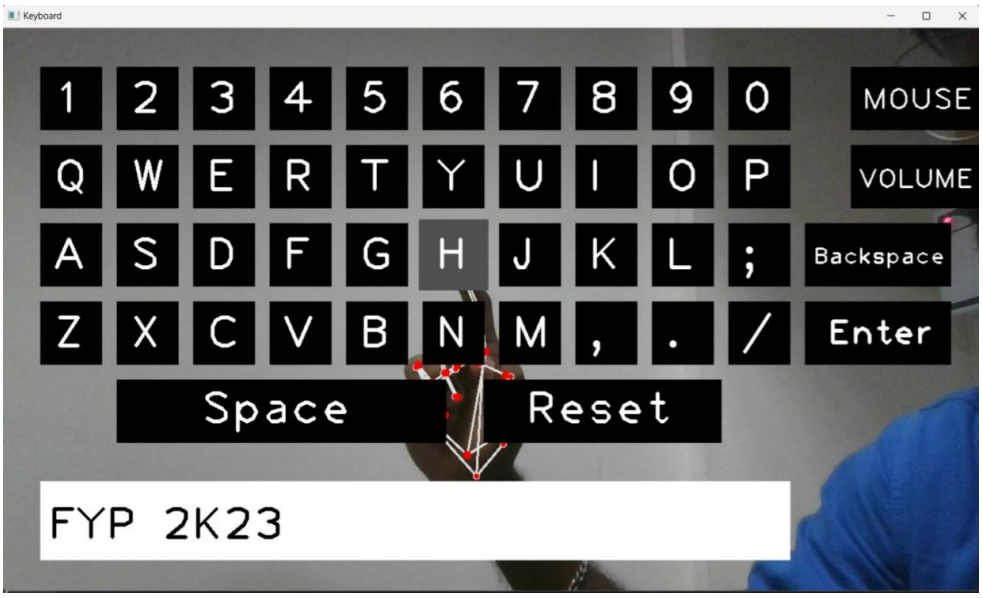

Figure 5.6 Virtual Keyboard

Figure 5.7 serves as a visual illustration of the feedback process, showcasing how the characters are displayed in the notepad as users interact with the virtual keyboard.

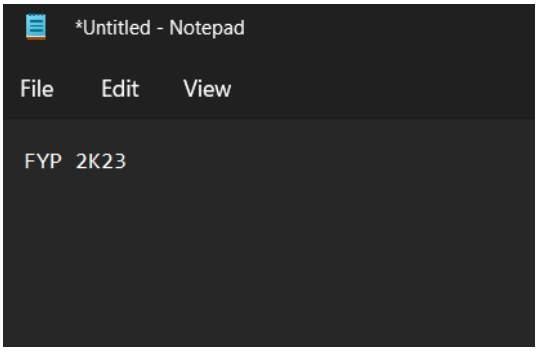

Figure 5.7 Corresponding Text in Notepad

5.4 HAND GESTURE RECOGNITION

5.4.1 Volume Control

In order to enable volume adjustment through hand gestures, the user is instructed to position their hand at a specific distance from the camera, specifically between 45 and 65 centimeters. This distance is chosen to ensure optimal tracking and recognition of hand movements.

The hand gesture used for volume control involves the index and thumb fingertips. Figure 5.8 visually illustrates this gesture. By bringing the index finger and thumb together, the user can increase or decrease the volume level. The distance between these fingertips is used to determine the magnitude of the volume adjustment.

This hand gesture-based volume control mechanism offers a convenient and intuitive way for users to modify the system's audio output. By utilizing natural hand movements, users can seamlessly adjust the volume without the need for physical buttons or external devices.

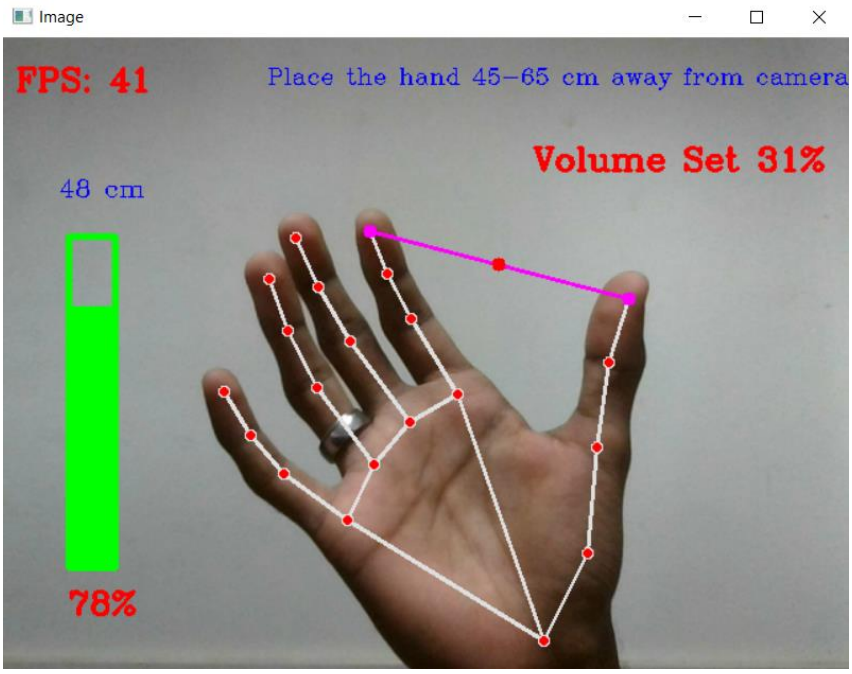

Figure 5.8 Adjusting Volume Percentage

To ensure precise and accurate volume control, the system incorporates an additional hand gesture involving the pinky finger, as depicted in Figure 5.9. This gesture allows users to fine-tune the volume settings with increased precision. To set the system volume, the user is required to close their pinky finger. This hand gesture serves as a trigger for the volume adjustment functionality. When the pinky finger is closed, the system detects the gesture and initiates the process of setting the volume.

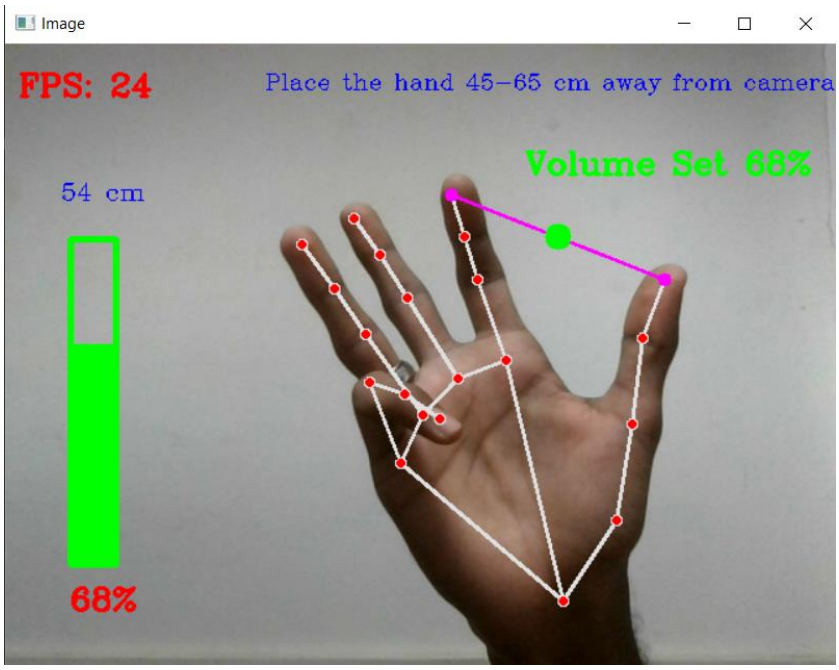

Figure 5.9 Volume Set

The current audio percentage, indicating the volume level, is visually presented on the screen in a graphical representation, as depicted in Figure 5.10.

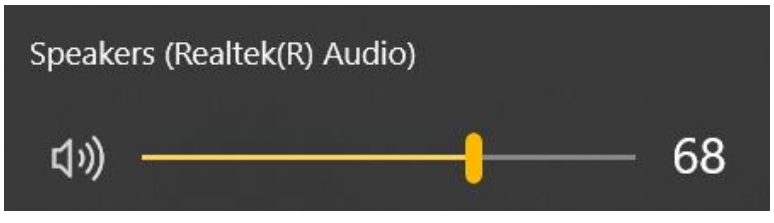

Figure 5.10 System Volume

5.4.2 Window Management Operations

The virtual mouse system incorporates various features for managing windows, such as scrolling, closing, and minimizing, as these actions are commonly performed by users when utilizing a mouse for navigation and control. One of the essential functionalities provided by the virtual mouse system is scrolling, which allows users to navigate through the content of a window or document. To enable scrolling down, the system utilizes a gesture where the thumb finger is raised, as illustrated in Figure 5.11.

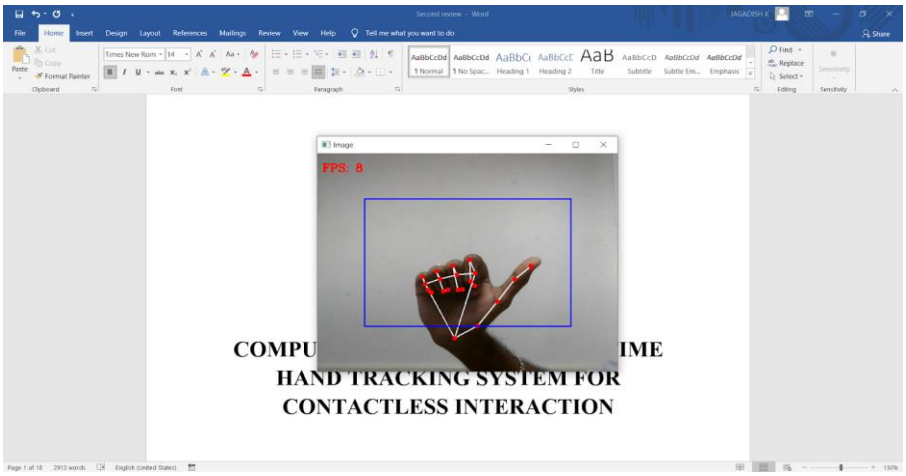

Figure 5.11 Scroll Down

Scrolling up is achieved through a distinct gesture where the pinky finger is lifted, as depicted in Figure 5.12.

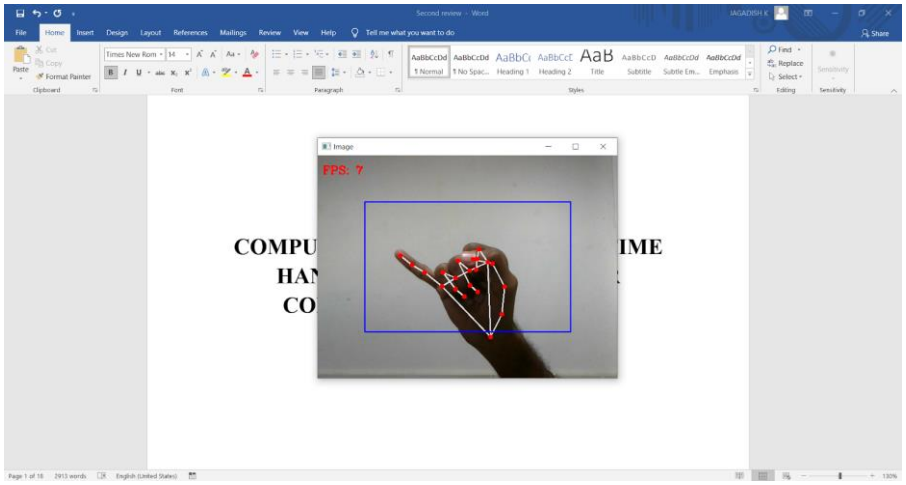

Figure 5.12 Scroll Up

To initiate the window-closing action, the user can raise their thumb, index, and pinky fingers together in a simultaneous movement. This gesture is illustrated in Figure 5.13, providing a visual representation of the hand position during the interaction.

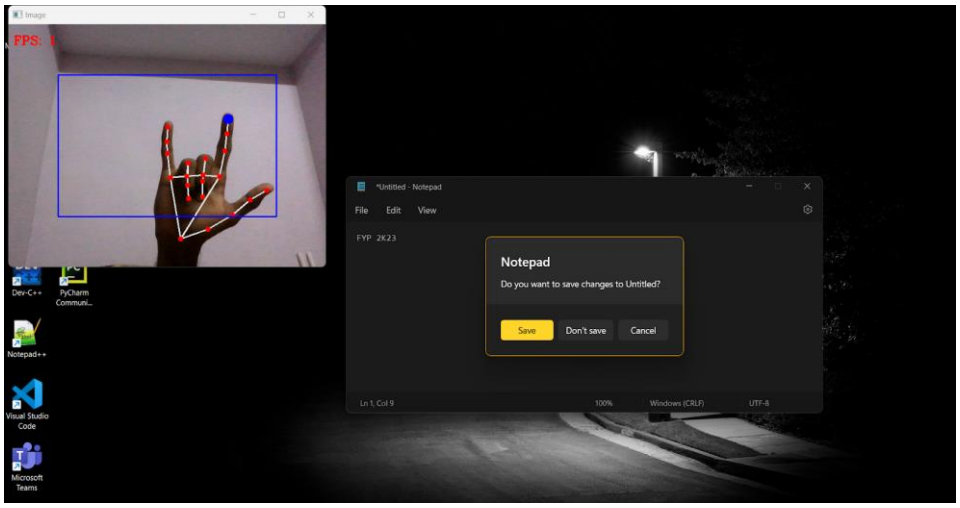

Figure 5.13 Close Window

Minimizing the current window is achieved through a distinct gesture where the user raises their index and pinky fingers together. Figure 5.14 illustrates this gesture, visually representing the hand position during the interaction.

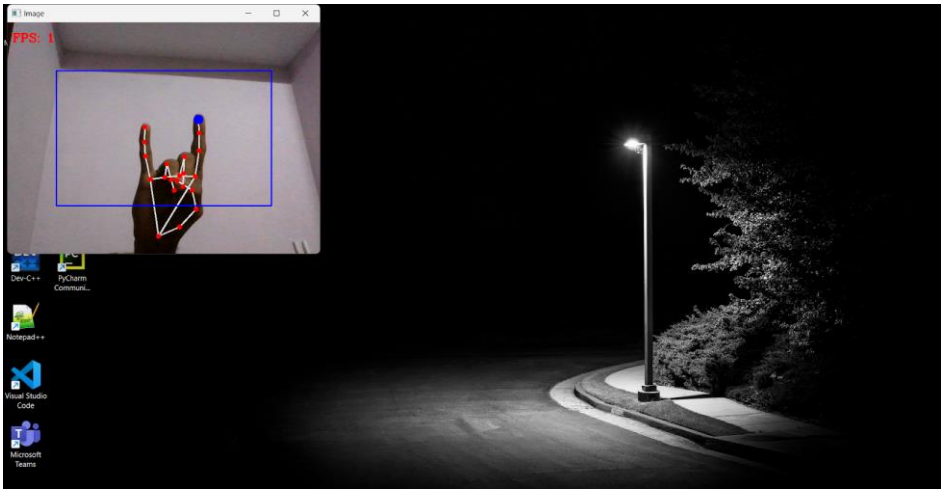

Figure 5.14 Minimize Window

CHAPTER 6

PERFORMANCE ANALYSIS

6.1 MOUSE ACTION PERFORMANCE

The virtual mouse system underwent a comprehensive evaluation to assess its performance across various functions commonly associated with mouse interactions. The objective of these tests was to measure the system's effectiveness in executing tasks such as left click, right click, double click, scroll up and down, as well as closing and minimizing windows. Each test case represented a specific task, and the success rates achieved in each case, based on 30 attempts, were recorded and summarized in Table 6.1.

Table 6.1 Mouse Action Performance Testing Results

Test Case	Task	Success Rate (%)
1	Left click on a small icon	90%
2	Right click on a specific target	100%
3	Right-click and select an option from the context menu	92%
4	Double click on a specific target	92%
5	Double-click on a file to open it	100%
6	Scroll up and down in a document	100%
7	Close a window	100%
8	Minimize a window	100%

By conducting multiple trials for each test case, a comprehensive overview of the system's performance was obtained. The success rates presented in Table 6.1 provide valuable insights into the system's proficiency in carrying out each task.

6.2 KEYBOARD PERFORMANCE

Table 6.2 presents the comprehensive results of the performance testing conducted to assess the virtual keyboard's typing speed and accuracy. This testing aimed to evaluate the system's effectiveness in various typing tasks. The table provides detailed information on the test cases, corresponding tasks, time taken to complete each task in seconds, and success rates in percentage. The accuracy was measured by counting the number of correct keystrokes, while the success rate represents the percentage of successful attempts out of 30 trials for each task. The testing encompassed a range of typing tasks, including typing simple and complex sentences, long paragraphs, passwords, numerical values, and random combinations of letters and numbers.

Table 6.2 Keyboard Performance Testing Results

Test Case	Task	Time taken (s)	Success Rate (%)
1	Type a simple sentence ("The sky is blue")	30	100%
2	Type a complex sentence with punctuation ("Hi, hope you are fine.")	50	95%
3	Type a long paragraph with multiple sentences	120	90%
4	Type a password with at least 8 characters	20	100%
5	Type a numerical value with decimal points (e.g. 2.58)	5	100%
6	Type a series of random letters and numbers	15	100%

6.3 VOLUME CONTROL PERFORMANCE

Table 6.3 presents the results of the performance testing conducted to evaluate the virtual volume control function. This testing focused on assessing the effectiveness of the virtual volume control in various tasks related to audio management. The table provides an overview of the test cases, corresponding tasks, and the success rates achieved for each task. The primary objective of the testing was to measure the system's ability to accurately and reliably control the volume in a virtual environment. The virtual volume control function was evaluated across different tasks, including adjusting the volume, decreasing the volume, setting the volume using the pinky finger gesture, muting the volume, and setting the maximum volume level. The success rates presented in Table 6.3 represent the percentage of successful attempts out of the 30 trials conducted for each task. The high success rates achieved for each task, ranging from 92% to 100%, demonstrate the effectiveness of the virtual volume control function. Factors such as user familiarity, hand positioning, and environmental conditions can influence the actual success rates. These results assure users that the system is capable of accurately and reliably managing audio levels, contributing to an enhanced and immersive virtual audio experience.

Table 6.3 Volume Control Performance Testing Results

Test Case	Task	Success Rate (%)
1	Adjusting volume	100%
2	Decrease volume	100%
3	Setting volume using pinky finger	92%
4	Mute volume	94%
5	Setting maximum volume	100%

CHAPTER 7

CONCLUSION AND FUTURE PROSPECTS

7.1 CONCLUSION

The virtual mouse and keyboard system developed in this project has demonstrated outstanding performance in terms of efficiency, accuracy, and usability. Through extensive testing, it successfully executed various mouse actions, window management tasks, and typing functions with high success rates and minimal errors. The system's ability to accurately interpret user inputs and provide seamless interaction highlights its potential for practical use. Furthermore, the inclusion of the virtual volume control function showcased its effectiveness in adjusting audio levels using intuitive gestures. The integration of contactless interaction capabilities in the system contributes to a safer and more hygienic user experience, particularly in environments where cleanliness and germ prevention are crucial. Overall, the virtual mouse and keyboard system, with its exceptional efficiency, accuracy, and contactless interaction capabilities, represents a significant advancement in user interface technology. Its potential to enhance accessibility, user experience, and hygiene makes it a valuable asset for both individuals and organizations seeking efficient and safe digital interactions.

7.2 FUTURE PROSPECTS

The system could be further optimized to improve its accuracy and speed, and new features could be added to enhance its functionality. For instance, support for additional hand gestures or the integration of voice commands could be explored. Additionally, the system could be adapted for use with virtual reality or augmented reality environments, enabling new and immersive ways of interacting with digital content.

REFERENCES

[1] Abdelghafar, R.E., and Shawkat, K.G., (2021), ‘On-Air Hand-Drawn Doodles for IoT Devices Authentication During COVID-19’, IEEE Access, Vol. 9, pp. 161723 - 161744.

[2] Changhyun, J., Oh, J.K., and Dongkyoo, S., (2017), ‘Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction’, IEEE Access, Vol. 5, pp. 25181 - 25188.

[3] Chinnam, D.S.N., Chukka, U.S.R., Brumancia, E., Indira, K., Anandhi, T., and Ajitha, P., (2020), ‘Finger Recognition and Gesture based Virtual Keyboard’, International Conference on Communication and Electronics Systems, Vol. 1, pp. 1321 - 1324.

[4] Dubbaka, M.S.R., Srilekha, K., and Rishika, K.T.M., (2022), ‘Virtual Mouse Using Hand Gesture’, International Conference on Knowledge Engineering and Communication Systems, Vol. 1, pp. 340 - 345.

[5] Hubert, C., (2016), ‘A Multimodal Gaze-Controlled Virtual Keyboard’, IEEE Transactions on Human-Machine Systems, Vol. 46, No. 4, pp. 601 - 606.

[6] Isaiah, J.T., and Melvin, C., (2021), ‘Vision-Based Hand Tracking System Development for Non-Face-to-Face Interaction’, IEEE International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management, Vol. 1, pp. 1 - 6.

[7] Jungpil, S., and Cheol, M.K., (2017), ‘Non-Touch Character Input System Based on Hand Tapping Gestures Using Kinect Sensor’, IEEE Access, Vol. 5, pp. 10496 - 10505.

[8] Kirti, A., and Anuja, A., (2022), ‘An Approach to Control the PC with Hand Gesture Recognition using Computer Vision Technique’, International Conference on Computing for Sustainable Global Development, Vol. 1, pp. 760 - 764.

[9] Koushik, R., and Mohammed, A.H.A., (2022), ‘Real Time Hand Gesture Based User Friendly Human Computer Interaction System’, 3rd International Conference on Innovations in Science, Engineering and Technology, Vol. 1, pp. 260 - 265.

[10] Liuhao, G., Hui, L., Junsong, Y., and Daniel, T., (2019), ‘Real-Time 3D Hand Pose Estimation with 3D Convolutional Neural Networks’, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 41, No. 4, pp. 956 - 970.

[11] Manav, R., Madhur, R., Neha, L., and Radha, S., (2021), ‘Hand Gesture Recognition Based Virtual Mouse Events’, International Conference for Emerging Technology, Vol. 1, pp. 1 - 4.

[12] Su, J.P., Tae, H.L., Viduranga, M., Tae, S.K., and Hyuk, J.L., (2023), ‘Fast Virtual Keyboard Typing Using Vowel Hand Gesture Recognition’, International Conference on Electronics, Information, and Communication, Vol. 1, pp. 1 - 4.

[13] Tae, H.L., Sunwoong, K., Taehyun, K., Jin, S.K., and Hyuk, J.L, (2022), ‘Virtual Keyboards With Real-Time and Robust Deep Learning-Based Gesture Recognition’, IEEE Transactions on Human-Machine Systems, Vol. 52, No. 4, pp. 725 - 735.

[14] Sairam, U., Dharani, K.R.G., and Sai, C.K., (2022), ‘Virtual Mouse using Machine Learning and GUI Automation’, 8th International Conference on Advanced Computing and Communication Systems, Vol. 1, pp. 1112 - 1117.

[15] Kriznar, V., Leskovsek, M., and Batagelj, B., (2021), ‘Use of Computer Vision Based Hand Tracking in Educational Environments’, International Convention on Information, Communication and Electronic Technology, Vol. 1, pp. 804 - 809.

PAPER NAME

Final_FT.pdf

AUTHOR

JAGADISH K

WORD COUNT

8255 Words

CHARACTER COUNT

45307 Characters

PAGE COUNT

47 Pages

FILE SIZE

1020.4KB

SUBMISSION DATE

May 31, 2023 3:50 PM GMT+5:30

REPORT DATE

May 31, 2023 3:51 PM GMT+5:30

16% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 9% Internet database
- 8% Publications database
- Crossref database
- Crossref Posted Content database
- 11% Submitted Works database