### A

Major Project On

**AIR BOARD USING HAND GESTURES**

(Submitted in partial fulfillment of the requirements for the award of the Degree)

# BACHELOR OF TECHNOLOGY

In

# INFORMATION TECHNOLOGY

BY

**VARSHA (227R1A1259****)**

**V.NITEESH (227R1A1262)**

**ABHINAV(237R5A1204)**

Under the Guidance of

## Dr.V.Malsoru

(Professor)

## DEPARTMENT OF INFORMATION TECHNOLOGY CMR TECHNICAL CAMPUS

**UGC AUTONOMOUS**

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH,Approved\_by\_AICTE,NewDelhi) Recognized Under Section 2(f) & 12(B) of the UGC Act.1956,Kandlakoya(V), Medchal Road, Hyderabad-501401.

**2023-2024**

# DEPARTMENT OF INFORMATION TECHNOLOGY



**CERTIFICATE**

This is to certify that the project entitled “ **AIR BOARD USING HAND GESTURES** “ being submitted by **VARSHA (227R1A1259), V.NITEESH (227R1A1262), ABHINAV (237R5A1204),** in partial fulfillment of the requirements for the award of the degree of B.Tech in Information Technology from Jawaharlal Nehru Technological University Hyderabad, is a record of the bonafide work carried out by them under our guidance and supervision during the year 2023-2024.

The results embodied in this project has not been submitted any other university or institute for the award of any degree or diploma

#### Dr.V.Malsoru Dr. A. Raji Reddy

**(Professor) DIRECTOR**

#### INTERNAL GUIDE

**Dr. B. Kavitha Rani EXTERNAL EXAMINAR HEAD OF THE DEPARTMENT**

**Submitted for viva voce Examination held on**

# ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any work depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express our profound gratitude and deep regard to our guide faculty, **Dr.V.Malsoru**, Professor for his exemplary guidance, monitoring and constant encouragement throughout the course of this Subject presentation. We take this opportunity to express our profound gratitude to our PRC coordinator, **Mr. MD.Sajid Pasha**, Assistant Professor. The blessing, help and guidance given by their time to time shall carry us a long way in the journey of line on which we are about to embark.

We also thankful to the Head of the Department **Dr. B. Kavitha Rani** for providing excellent Infrastructure and a nice atmosphere for completing this presentation successfully

We would like to express our sincere gratitude to **Dr. M. Ahmed Ali Baig**, Dean Administration, **Dr. D T V Dharmajee Rao,** Dean Academics, **Dr. Ashutosh Saxena**, Dean R&D for encouragement throughout the course of this presentation.

We obliged to our Director **Dr. A. Raji Reddy** for being cooperative throughout the course of this presentation.

We wish to express our sincere gratitude to the Management of **CMR Technical Campus**,Hyderabad,Sri **C. Gopal Reddy**, Honourable Chairman, Smt **C.Vasantha Latha**, Honourable Secretary, Sri **C. Abhinav Reddy**, Honourable Chief Executive Officer.The guidance and support received from all the members of **CMR TECHNICAL CAMPUS** who contributed and who are contributing to this presentation, was vital for the success of the presentation. We are grateful for their constant support and help. Finally, We would like to take this opportunity to thank our family for their constant encouragement without which this assignment would not be possible.We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this presentation.

#### VARSHA(207R1A1259)

#### V.NITEESH (227R1A1262)

**ABHINAV(237R5A120****3**)

# ABSTRACT

The advent of touchless interaction technologies has opened new avenues in human-computer interaction (HCI). This paper presents the design and implementation of an "Air Board" system, which leverages hand gestures to enable intuitive and immersive interaction with digital interfaces. The system utilizes advanced computer vision and machine learning algorithms to accurately detect and interpret hand movements, allowing users to draw, write, and manipulate objects in a virtual space without physical contact.

Key components of the system include a high-resolution camera for gesture recognition, a robust gesture classification algorithm trained on a diverse dataset, and an efficient real-time processing unit to ensure smooth and responsive interaction. The Air Board system is designed to be user-friendly and adaptable, with customizable gesture sets and sensitivity settings to cater to individual preferences and varying environmental conditions.

Experimental results demonstrate high accuracy in gesture recognition and low latency in response time, making the Air Board suitable for a wide range of applications, from educational tools and creative design platforms to remote collaboration and virtual reality environments. User studies indicate a positive reception, highlighting the system's potential to enhance productivity and creativity by providing a natural and engaging way to interact with digital content.

The paper concludes with a discussion of potential improvements, such as incorporating haptic feedback and expanding the gesture vocabulary, as well as future research directions to further refine and expand the capabilities of touchless interaction systems. The Air Board represents a significant step forward in the evolution of HCI, offering a glimpse into the future of seamless and intuitive digital interaction.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO** |
| Fig no 5.1 | System of architecture | 10 |
| Fig no 7.1 | SCREEN SHOT :using blue ink | 17 |
| Fig no 7.2 | SCREEN SHOT :using green ink | 17 |
| Fig no 7.3 | SCREEN SHOT :using red ink | 18 |
| Fig no 7.4 | SCREEN SHOT :using yellow ink | 18 |
| Fig no 7.5 | SCREEN SHOT :using clear button | 19 |

**ABSTRACT ------------------------------------------------------------------------------ I**

**LIST OF FIGURES ---------------------------------------------------------------------- II**

**1. INTRODUCTION ---------------------------------------------------------------------- 1**

1. INTRODUCTION --------------------------------------------------------------------- 1
2. OBJECTIVES -------------------------------------------------------------------------- 1
3. LIMITATIONS ------------------------------------------------------------------------ 2

**2. SYSTEM ANALYSIS ----------------------------------------------------------------- 3**

1. SYSTEM COMPONENTS ----------------------------------------------------------- 3
2. FUNCTIONALITY ------------------------------------------------------------------- 3
3. PERFORMANCE MATRICS ------------------------------------------------------- 4
4. USE CASE SCENARIOUS ---------------------------------------------------------- 4
5. CHALLENGES & LIMITATIONS ------------------------------------------------- 4

**3.SYSTEM STUDY ----------------------------------------------------------------------- 5**

1. SYSTEM OVERVIEW --------------------------------------------------------------- 5
2. COMPONENTS & ARCHITECTURE ---------------------------------------------- 5

**4.SYSTEM REQUIREMENTS SPECIFICATIONS -------------------------------- 6**

1. HARDWARE REQUIREMENTS ---------------------------------------------------- 6
2. SOFTWARE REQUIREMENTS ----------------------------------------------------- 7
3. PERFORMANCE REQUIREMENTS ----------------------------------------------- 7
4. USER REQUIREMENTS -------------------------------------------------------------- 7
5. SECURITY & PRIVACY -------------------------------------------------------------- 7

**5.SYSTEM DESIGN ------------------------------------------------------------------------ 8**

1. HIGH RESOLUTION CAMERA ----------------------------------------------------- 8
2. PROCESSING UNIT ------------------------------------------------------------------- 8
3. DISPLAY DEVICE --------------------------------------------------------------------- 8
4. OPERATING SYSTEM ---------------------------------------------------------------- 8
5. GESTURE RECOGNITION SOFTWARE ------------------------------------------ 8
6. USER INTERFACE SOFTWARE ---------------------------------------------------- 8
7. DATA FLOW ---------------------------------------------------------------------------- 8
8. USER INTERFACE DESIGN --------------------------------------------------------- 9
9. SECURITY MEASURES -------------------------------------------------------------- 9

**6.IMPLEMENTATION ------------------------------------------------------------------- 11**

1. HARDWARE SETUP ------------------------------------------------------------------ 11
2. SOFTWARE DEVELOPMENT ------------------------------------------------------ 11
3. DEPLOYMENT ------------------------------------------------------------------------- 12

**SOURCE CODE ---------------------------------------------------------------------------- 13**

**7. SCREEN-SHOTS ------------------------------------------------------------------------ 17**

**8.SYSTEM TESTING ---------------------------------------------------------------------- 20**

1. UNIT TESTING ------------------------------------------------------------------------- 20
2. INTEGRATION TESTING ------------------------------------------------------------ 21
3. USER TESTING ------------------------------------------------------------------------- 21
4. PERFOMANCE TESTING ------------------------------------------------------------ 22
5. SECURITY TESTING ----------------------------------------------------------------- 22
6. REGRESSION TESTING -------------------------------------------------------------- 23

**9.CONCLUSION AND FUTURE ENHANCEMENT ------------------------------- 24**

**10.BIBIOGRAPHY ------------------------------------------------------------------------- 25**

10.1 REFERANCE --------------------------------------------------------------------------- 25

# 1. INTRODUCTION

## INTRODUCTION

The rapid advancement of technology has continually reshaped the way humans interact with computers and digital interfaces. From the early days of command-line interfaces to the more recent adoption of touchscreens, the evolution of human-computer interaction (HCI) has been driven by the pursuit of more intuitive, natural, and efficient methods of communication between users and machines. In this context, the development of touchless interaction technologies, particularly those based on hand gestures, represents a significant leap forward.

The "Air Board" is an innovative system designed to harness the power of hand gestures for seamless interaction with digital environments. By eliminating the need for physical contact, the Air Board offers a hygienic and effortless means of controlling and manipulating digital content. This technology is particularly relevant in the wake of global health concerns, where minimizing touch surfaces can contribute to safer public and personal computing experiences.

## OBJECTIVES

The main objective is to clarify the types of activities and corresponding risks.

Develop a system that allows users to interact with digital interfaces in a natural and intuitive manner using hand gestures.

Reduce the learning curve associated with using new technologies by mimicking familiar physical interactions.

Eliminate the need for physical contact with input devices, promoting hygiene and reducing the spread of germs, particularly in shared environments.

Implement advanced computer vision and machine learning algorithms to ensure high accuracy in gesture recognition.

1

## 1.3 LIMITATIONS

Environmental Conditions: The accuracy of gesture recognition can be affected by varying lighting conditions, background noise, and other environmental factors.

Occlusion and Overlapping: Accurate detection can be hindered when gestures involve overlapping hands or objects, causing misinterpretation of gestures.

Camera Quality: The system's performance heavily relies on the quality of the camera used. Low-resolution cameras or those with limited frame rates can degrade the accuracy and responsiveness.

Processing Power: Real-time processing of gestures requires significant computational resources, which might not be available on all devices, leading to potential delays and lag.

2

# 2.SYSTEM ANALYSIS

## 2.1 SYSTEM COMPONENTS

**High-Resolution Camera:** Captures hand movements and gestures. The quality and frame rate of the camera significantly impact the system's performance.

**Processing Unit:** Handles real-time processing of video feed and gesture recognition. High computational power is required for smooth and responsive interaction.

**Display Interface:** The output device (monitor, projector, VR headset) where the user interacts with the virtual content.

**Gesture Recognition Algorithm:** Utilizes machine learning models trained on extensive datasets to accurately identify hand gestures.

**Computer Vision Techniques:** Implemented to preprocess the video feed, detect hand positions, and track movements.

**User Interface (UI):** The virtual board interface where users can draw, write, and manipulate objects. Customizable to suit different applications

## 2.2 FUNCTIONALITY

The camera captures real-time video of the user's hand movements.

The computer vision system identifies and tracks the hand, extracting features such as shape, position, and motion.

The gesture recognition algorithm classifies the detected hand movements into predefined gestures.

**Drawing and Writing:** Users can draw or write on the virtual board using specific gestures.

**Object Manipulation:** Gestures can be used to select, move, resize, and rotate virtual objects.

**Command Execution:** Specific gestures can trigger commands, such as opening a menu, saving a file, or undoing an action.

3

.

## 2.3 PERFORMANCE MATRICS

The system's ability to correctly recognize and interpret gestures. High accuracy is crucial for reliable interaction.

Factors affecting accuracy include lighting conditions, background noise, and hand occlusion.

The latency between performing a gesture and the corresponding action appearing on the screen. Low latency ensures a seamless user experience.

Depends on the processing power and efficiency of the gesture recognition algorithm.

Evaluated through user studies and feedback. Measures how intuitive and comfortable the system is for various tasks.

Includes factors such as ease of use, learning curve, and ergonomic comfort

## 2.4 USE CASE SCENARIOS

Teachers can use the Air Board for interactive lessons, drawing diagrams, and annotating content in real-time without physical contact.

Designers and artists can sketch and manipulate digital art using hand gestures, providing a more natural and fluid creative process.

Facilitates virtual meetings and collaborative work by allowing participants to interact with shared digital content using gestures.

Enhances VR experiences by enabling natural hand interactions, increasing immersion, and reducing the need for physical controllers.

## 2.5 CHALLENGES AND LIMITATIONS

The system's performance can be significantly affected by changes in lighting and background conditions.

High-quality cameras and powerful processing units are necessary, which might increase the overall cost and accessibility of the system.

Prolonged use of hand gestures can lead to physical strain, limiting the duration for which the system can be comfortably used.

Capturing video of hand movements raises potential privacy issues, especially in sensitive environment

4

**3. SYSTEM STUDY**

# 

**3.1 System Overview**

The Air Board system is designed to facilitate touchless interaction with digital interfaces through the use of hand gestures. It integrates hardware and software components to detect, interpret, and respond to user gestures in real-time, providing a seamless and intuitive user experience..

#### 3.2 Components and Architecture

**Camera:** A high-resolution, high-frame-rate camera captures hand movements. The camera must be capable of accurately detecting hand positions and gestures in various lighting conditions.

**Processing Unit:** A powerful processor or GPU is required to handle the computational demands of real-time gesture recognition and interface rendering.

**Display Device:** The output can be displayed on monitors, projectors, or VR headsets, depending on the application.

**Gesture Recognition Software:** Utilizes computer vision and machine learning algorithms to identify and classify hand gestures. The software includes:

**Hand Detection Module:** Identifies the presence of hands in the camera feed.

**Feature Extraction Module:** Extracts relevant features (e.g., hand shape, position, motion) for gesture recognition.

**Classification Module:** Classifies the extracted features into predefined gestures using trained models.

5

# 4. SYSTEM REQUIREMENTS / SPECFICATIONS

The successful implementation and operation of the Air Board using hand gestures require specific hardware and software components. These components must meet certain specifications to ensure the system performs accurately and responsively. Below are the detailed system requirements and specifications.

#### 4.1 ****Hardware Requirements****

**Resolution:** Minimum 1080p (1920x1080 pixels) for clear and detailed capture of hand movements.

**Frame Rate:** At least 30 frames per second (fps) to ensure smooth tracking of fast hand gestures.

**Field of View:** Wide-angle lens (70-120 degrees) to cover a broad interaction area.

**Connectivity:** USB 3.0 or higher for fast data transfer to the processing unit.

**Processor:** Multi-core processor, such as Intel Core i7 or AMD Ryzen 7, for efficient handling of real-time data processing.

**GPU:** Dedicated graphics processing unit (e.g., NVIDIA GeForce GTX 1060 or higher) to accelerate computer vision and machine learning tasks.

**RAM:** Minimum 16 GB to support concurrent processing of multiple tasks.

**Storage:** Solid State Drive (SSD) with at least 256 GB for fast read/write speeds and storage of gesture data.

**Monitor/Projector:** Full HD resolution (1920x1080 pixels) or higher, with a refresh rate of at least 60 Hz.

**VR Headset (optional):** For immersive applications, a VR headset with at least 90 Hz refresh rate and 1080x1200 pixels per eye resolution.

6

* 1. **Software Requirements**

**Windows 10/11**, **macOS Catalina or later**, or **Linux (Ubuntu 18.04 or later)**.

**Computer Vision Library:** OpenCV or equivalent for hand detection and feature extraction.

**Machine Learning Framework:** TensorFlow, PyTorch, or equivalent for gesture classification.

**SDKs/APIs:** Integration with relevant SDKs (e.g., Microsoft Azure Kinect SDK) for camera and sensor data processing.

**Development Platform:** Unity, Unreal Engine, or similar for creating interactive and responsive UIs.

**Custom Tools:** Custom-built tools for drawing, writing, and object manipulation, compatible with the chosen development platform.

**4.3 Performance Requirements**

**Gesture Recognition Accuracy:** 95% or higher in optimal conditions.

**Environmental Robustness:** Maintain at least 85% accuracy in varying lighting conditions and with different background complexities.

**System Latency:** Less than 150 milliseconds from gesture performance to visual feedback on the display.

**4.4 User Experience Requirements**

**Ease of Setup:** Plug-and-play hardware setup with minimal configuration required.

**User Interface:** Intuitive and easy-to-navigate interface with clear visual feedback.

**Comfort:** Designed to minimize user fatigue with efficient gesture design and short interaction times.

**Adaptability:** Adjustable interaction area and gesture sensitivity to accommodate different user preferences and physical capabilities.

**4.5 Security and Privacy**

**Data Encryption:** Ensure all captured data is encrypted to protect user privacy.

**Local Processing:** Prefer local data processing over cloud-based to reduce privacy risks.

**User Authentication:** Implement authentication mechanisms to prevent unauthorized access.

**Permission Management:** Allow users to control what data is captured and how it is used.

By meeting these requirements and specifications, the Air Board using hand gestures can provide a high-performance, user-friendly, and secure touchless interaction system suitable for a wide range of applications.

7

# 5. SYSTEM DESIGN

**5.1 High-Resolution Camera:**

**Specifications:** 1080p resolution, 30 fps, wide-angle lens (70-120 degrees).

**Mounting:** Adjustable mount to position the camera for optimal coverage.

**Connectivity:** USB 3.0 or higher for data transfer to the processing unit.

**5.2 Processing Unit:**

**CPU:** Multi-core processor (e.g., Intel Core i7 or AMD Ryzen 7).

**GPU:** Dedicated GPU (e.g., NVIDIA GeForce GTX 1060 or higher).

**RAM:** Minimum 16 GB.

**Storage:** SSD with at least 256 GB

**5.3 Display Device:**

**Monitor/Projector:** Full HD resolution (1920x1080 pixels) or higher.

**Optional VR Headset:** For immersive applications, with at least 90 Hz refresh rate and 1080x1200 pixels per eye resolution.

**5.4 Operating System:**

**Supported OS:** Windows 10/11, macOS Catalina or later, Linux (Ubuntu 18.04 or later).

**5.5 Gesture Recognition Software:**

**Computer Vision Library:** OpenCV or equivalent for hand detection and tracking.

**Machine Learning Framework:** TensorFlow, PyTorch, or equivalent for gesture classification.

**5.6 User Interface Software:**

**Development Platform:** Unity, Unreal Engine, or similar for creating interactive UIs.

**Custom Tools:** Drawing, writing, and object manipulation tools integrated into the UI.

#### ****5.7 Data Flow****

#### The camera captures the user's hand movements in real-time and sends the video feed to the processing unit.

#### The preprocessing module extracts frames from the video feed, applying filters to enhance image quality and isolate hand regions.

#### 

#### 8

**5.8** **User Interface Design**

**Drawing Canvas:** Central area where users can draw, write, and manipulate objects.

**Tool Palette:** Sidebar with drawing tools (pen, brush, shapes), color options, and command buttons (save, undo, redo).

**Visual Cues:** Highlighting the detected hand position and displaying gesture recognition status.

**On-Screen Prompts:** Providing instructions and feedback based on user actions.

**Settings Menu:** Allows users to customize gesture sensitivity, calibration, and interface preferences.

**Gesture Editor:** Interface for defining and training custom gestures.

#### 5.9 ****Security Measures****

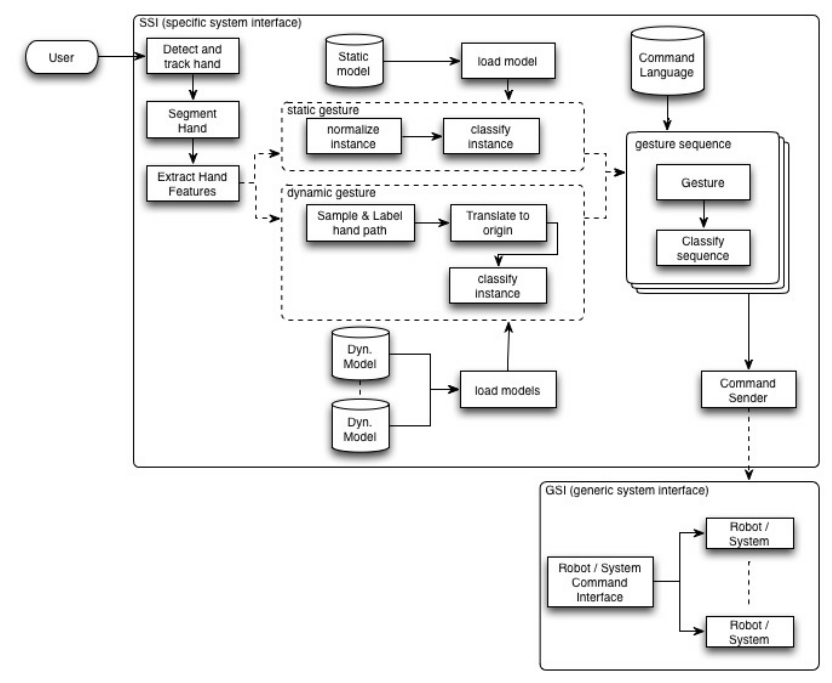
#### ****Encryption:**** Encrypt all captured data to protect user privacy.

#### ****Local Processing:**** Prefer local data processing to minimize privacy risks.

#### ****Authentication:**** Implement user authentication mechanisms to prevent unauthorized access.

#### ****Permissions:**** Allow users to control data capture settings and access levels.

9

****

**Fig 5.1 - SYSTEM ARCHITECTURE DIAGRAM**

10

# IMPLEMENTATION

The implementation of the Air Board using hand gestures involves the integration of hardware and software components to create a functional system. Below is a step-by-step guide covering setup, development, testing, and deployment.

#### 6.1 ****Hardware Setup** :**

#### ****Selection:**** Choose a camera with at least 1080p resolution and 30 fps capability.

#### ****Mounting:**** Position the camera in a location that covers the intended interaction area (0.5 to 2 meters from the user).

#### ****Connection:**** Connect the camera to the processing unit via USB 3.0 or higher.

#### ****Specification:**** Ensure the processing unit meets the required specifications (e.g., multi-core CPU, dedicated GPU, 16 GB RAM, SSD).

#### ****Setup:**** Install necessary drivers for the camera and ensure the processing unit is configured with the appropriate operating system.

#### ****Connection:**** Connect the display device (monitor/projector/VR headset) to the processing unit.

#### ****Configuration:**** Adjust display settings to match the resolution and refresh rate requirements.

#### 6.2 ****Software Development****

#### ****IDE:**** Set up an Integrated Development Environment (IDE) such as Visual Studio, PyCharm, or Unity Editor.

#### ****Libraries and Frameworks:**** Install necessary libraries and frameworks:

#### Extract features such as hand shape, position, and motion vectors.

#### Use these features to create a dataset for training the gesture recognition model.

#### Train a machine learning model (e.g., Convolutional Neural Network) on the dataset.

#### Implement the trained model to classify real-time hand gestures.

#### Develop a digital canvas in Unity or Unreal Engine where users can draw and write.

#### Implement tools such as pen, brush, and shapes with customizable options.

#### Enable gestures for selecting, moving, resizing, and rotating objects on the canvas.

#### Map specific gestures to system commands (e.g., open menu, save file, undo action).

#### Develop a calibration tool to adjust the system for different users and environments.

#### Implement a customization panel where users can define and train custom gestures.

#### 11

#### 6.3 ****Deployment****

* Set up the system in the intended environment (e.g., classroom, office, studio).
* Ensure all components are securely connected and configured.
* Provide training sessions for users to familiarize them with the system.
* Offer documentation and tutorials on using the Air Board and customizing gestures.
* Regularly update the software to improve performance and add new features.
* Perform periodic calibration and maintenance of hardware components to ensure optimal performance.

# 

# 

# 12

# SOURCE CODE :

# # All the imports go here

# import cv2

# import numpy as np

# import mediapipe as mp

# from collections import deque

# # Giving different arrays to handle colour points of different colour

# bpoints = [deque(maxlen=1024)]

# gpoints = [deque(maxlen=1024)]

# rpoints = [deque(maxlen=1024)]

# ypoints = [deque(maxlen=1024)]

# # These indexes will be used to mark the points in particular arrays of specific colour

# blue\_index = 0

# green\_index = 0

# red\_index = 0

# yellow\_index = 0

# #The kernel to be used for dilation purpose

# kernel = np.ones((5,5),np.uint8)

# colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (0, 255, 255)]

# colorIndex = 0

# # Here is code for Canvas setup

# paintWindow = np.zeros((471,636,3)) + 255

# paintWindow = cv2.rectangle(paintWindow, (40,1), (140,65), (0,0,0), 2)

# paintWindow = cv2.rectangle(paintWindow, (160,1), (255,65), (255,0,0), 2)

# paintWindow = cv2.rectangle(paintWindow, (275,1), (370,65), (0,255,0), 2)

# paintWindow = cv2.rectangle(paintWindow, (390,1), (485,65), (0,0,255), 2)

# paintWindow = cv2.rectangle(paintWindow, (505,1), (600,65), (0,255,255), 2)

# cv2.putText(paintWindow, "CLEAR", (49, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(paintWindow, "BLUE", (185, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(paintWindow, "GREEN", (298, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(paintWindow, "RED", (420, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(paintWindow, "YELLOW", (520, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2,

# cv2.LINE\_AA)

# cv2.namedWindow('Paint', cv2.WINDOW\_AUTOSIZE

# 13

# # initialize mediapipe

# mpHands = mp.solutions.hands

# hands = mpHands.Hands(max\_num\_hands=1, min\_detection\_confidence=0.7)

# mpDraw = mp.solutions.drawing\_utils

# # Initialize the webcam

# cap = cv2.VideoCapture(0)

# ret = True

# while ret:

# # Read each frame from the webcam

# ret, frame = cap.read()

# x, y, c = frame.shape

# # Flip the frame vertically

# frame = cv2.flip(frame, 1)

# #hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)

# framergb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# frame = cv2.rectangle(frame, (40,1), (140,65), (0,0,0), 2)

# frame = cv2.rectangle(frame, (160,1), (255,65), (255,0,0), 2)

# frame = cv2.rectangle(frame, (275,1), (370,65), (0,255,0), 2)

# frame = cv2.rectangle(frame, (390,1), (485,65), (0,0,255), 2)

# frame = cv2.rectangle(frame, (505,1), (600,65), (0,255,255), 2)

# cv2.putText(frame, "CLEAR", (49, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(frame, "BLUE", (185, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(frame, "GREEN", (298, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(frame, "RED", (420, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# cv2.putText(frame, "YELLOW", (520, 33), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE\_AA)

# #frame = cv2.cvtColor(hsv, cv2.COLOR\_HSV2BGR)

# # Get hand landmark prediction

# result = hands.process(framergb)

# # post process the result

# if result.multi\_hand\_landmarks:

# landmarks = []

# for handslms in result.multi\_hand\_landmarks:

# for lm in handslms.landmark:

# # # print(id, lm)

# # print(lm.x)

# # print(lm.y)

# lmx = int(lm.x \* 640)

# lmy = int(lm.y \* 480)

# landmarks.append([lmx, lmy])

# # Drawing landmarks on frames

# 

# 14

# mpDraw.draw\_landmarks(frame, handslms, mpHands.HAND\_CONNECTIONS)

# fore\_finger = (landmarks[8][0],landmarks[8][1])

# center = fore\_finger

# thumb = (landmarks[4][0],landmarks[4][1])

# cv2.circle(frame, center, 3, (0,255,0),-1)

# print(center[1]-thumb[1])

# if (thumb[1]-center[1]<30):

# bpoints.append(deque(maxlen=512))

# blue\_index += 1

# gpoints.append(deque(maxlen=512))

# green\_index += 1

# rpoints.append(deque(maxlen=512))

# red\_index += 1

# ypoints.append(deque(maxlen=512))

# yellow\_index += 1

# elif center[1] <= 65:

# if 40 <= center[0] <= 140: # Clear Button

# bpoints = [deque(maxlen=512)]

# gpoints = [deque(maxlen=512)]

# rpoints = [deque(maxlen=512)]

# ypoints = [deque(maxlen=512)]

# blue\_index = 0

# green\_index = 0

# red\_index = 0

# yellow\_index = 0

# paintWindow[67:,:,:] = 255

# elif 160 <= center[0] <= 255:

# colorIndex = 0 # Blue

# elif 275 <= center[0] <= 370:

# colorIndex = 1 # Green

# elif 390 <= center[0] <= 485:

# colorIndex = 2 # Red

# elif 505 <= center[0] <= 600:

# colorIndex = 3 # Yellow

# else :

# if colorIndex == 0:

# bpoints[blue\_index].appendleft(center)

# elif colorIndex == 1:

# gpoints[green\_index].appendleft(center)

# elif colorIndex == 2:

# rpoints[red\_index].appendleft(center)

# elif colorIndex == 3:

# 

# 15

# ypoints[yellow\_index].appendleft(center)

# # Append the next deques when nothing is detected to avois messing up

# else:

# bpoints.append(deque(maxlen=512))

# blue\_index += 1

# gpoints.append(deque(maxlen=512))

# green\_index += 1

# rpoints.append(deque(maxlen=512))

# red\_index += 1

# ypoints.append(deque(maxlen=512))

# yellow\_index += 1

# # Draw lines of all the colors on the canvas and frame

# points = [bpoints, gpoints, rpoints, ypoints]

# # for j in range(len(points[0])):

# # for k in range(1, len(points[0][j])):

# # if points[0][j][k - 1] is None or points[0][j][k] is None:

# # continue

# # cv2.line(paintWindow, points[0][j][k - 1], points[0][j][k], colors[0], 2)

# for i in range(len(points)):

# for j in range(len(points[i])):

# for k in range(1, len(points[i][j])):

# if points[i][j][k - 1] is None or points[i][j][k] is None:

# continue

# cv2.line(frame, points[i][j][k - 1], points[i][j][k], colors[i], 2)

# cv2.line(paintWindow, points[i][j][k - 1], points[i][j][k], colors[i], 2)

# cv2.imshow("Output", frame)

# cv2.imshow("Paint", paintWindow)

# if cv2.waitKey(1) == ord('q'):

# break

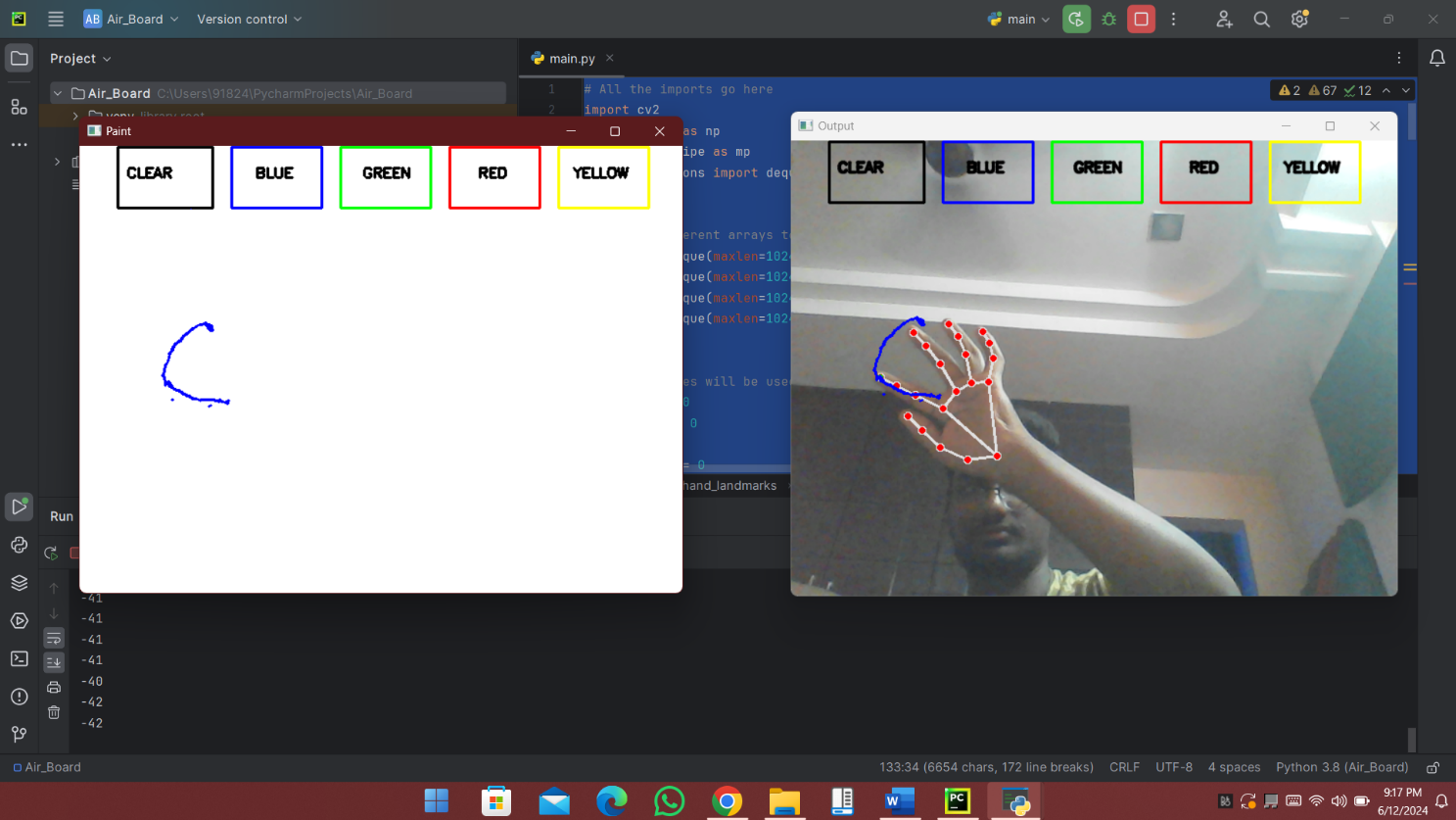
# # release the webcam and destroy all active windows

# cap.release()

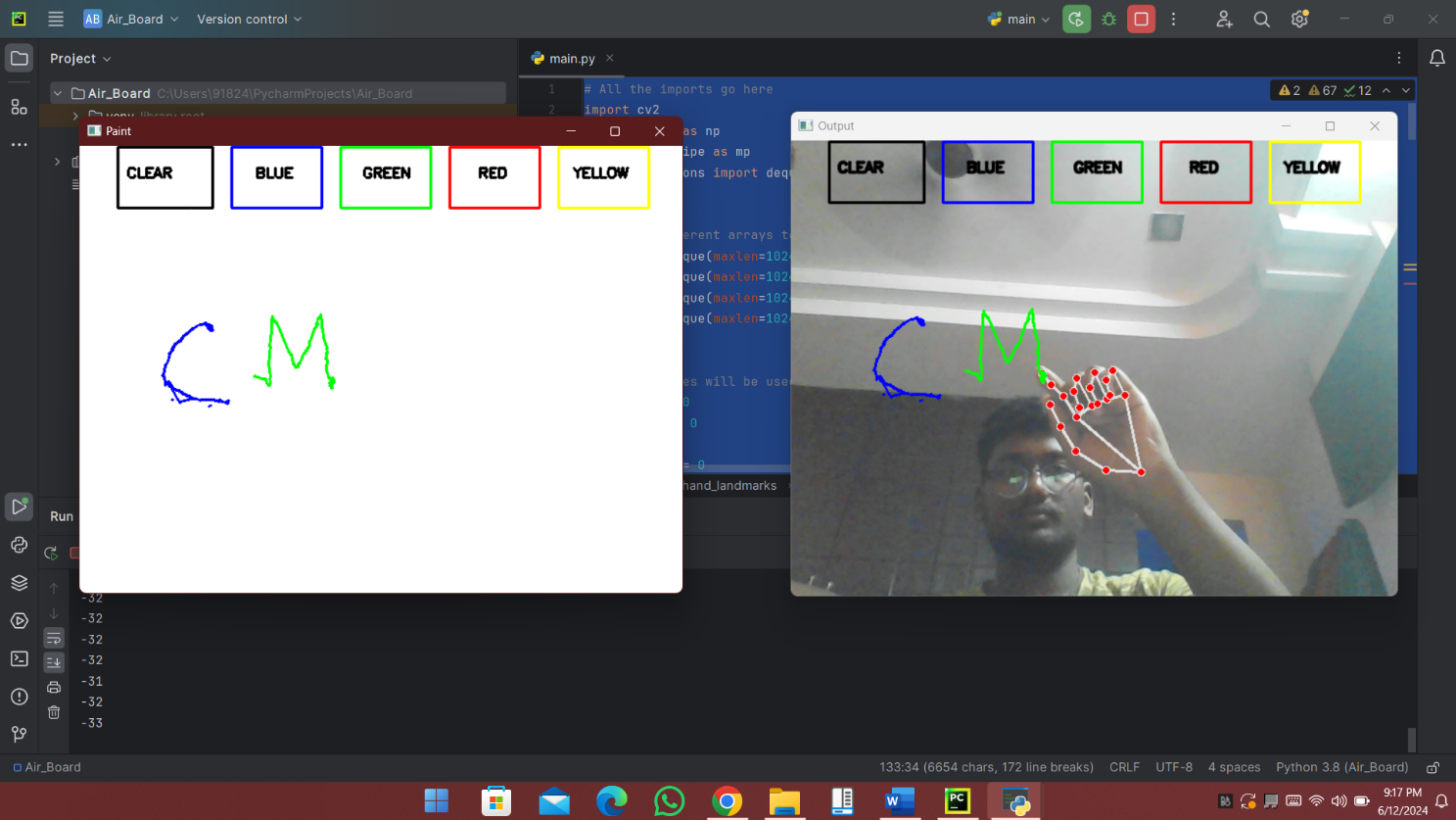
# cv2.destroyAllWindows()

16

# SCREENSHOTS

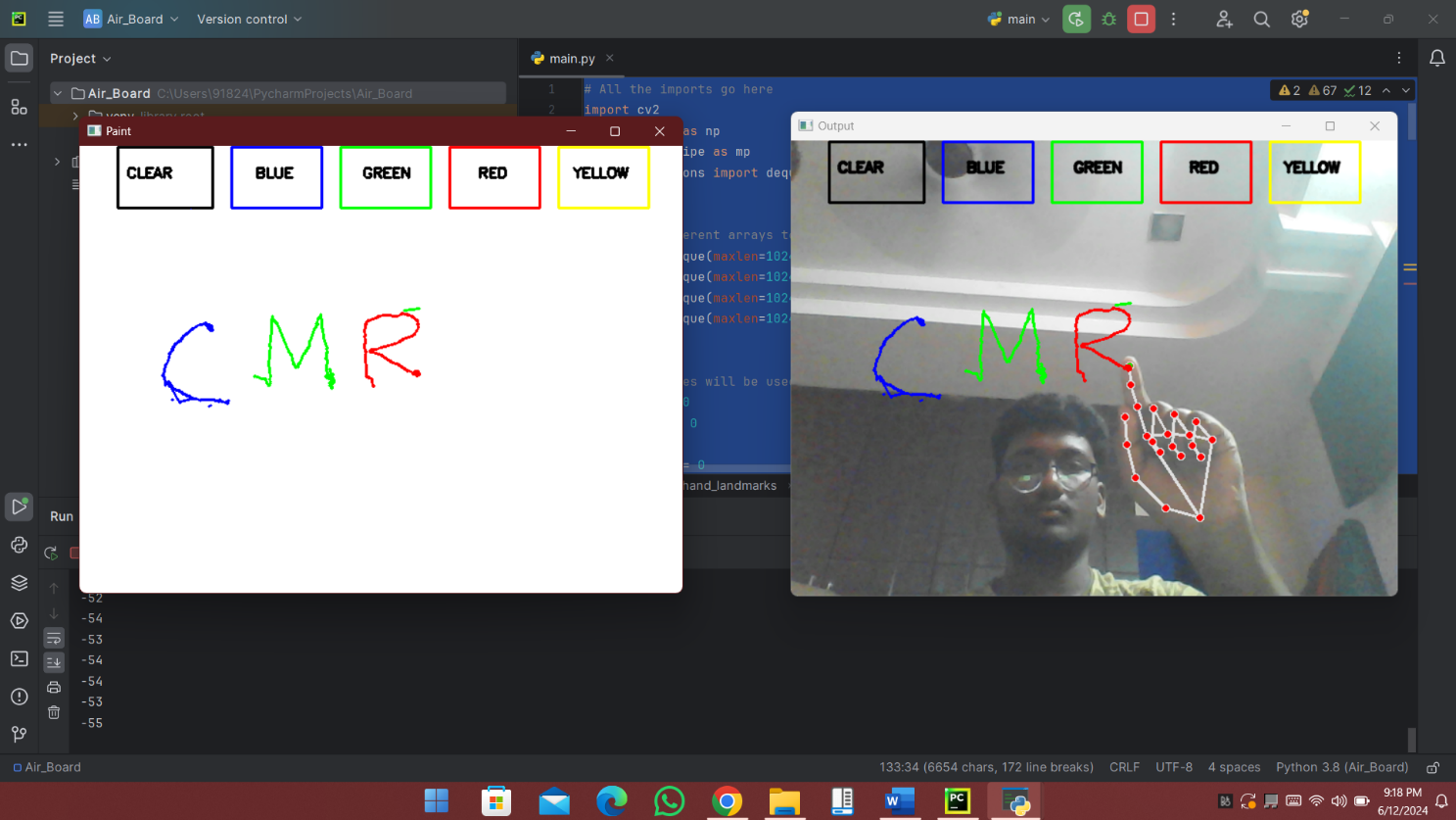


**Fig 7.1 SCREEN SHOT :** USING BLUE INK

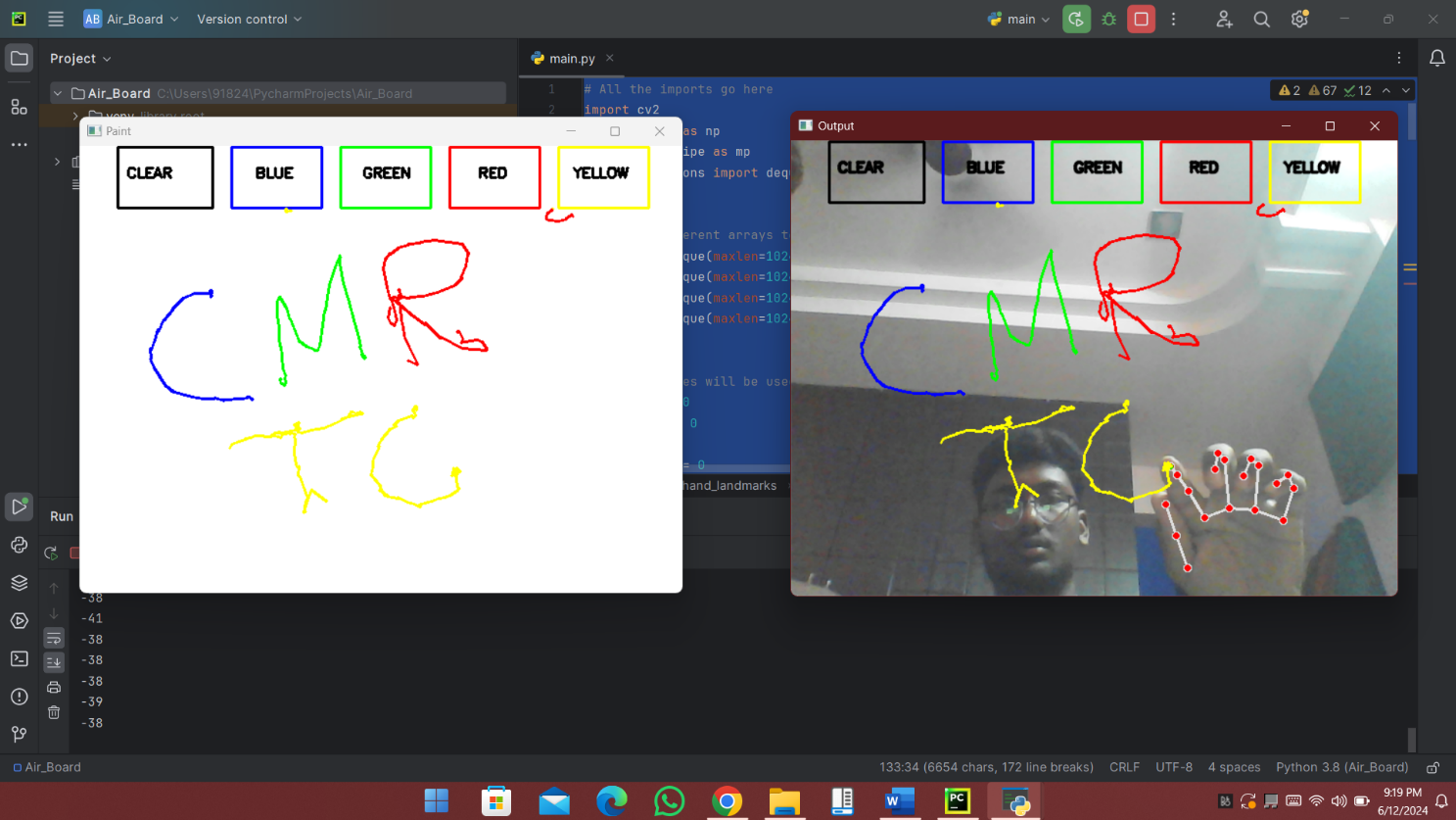


**Fig 7.2 SCREEN SHOT :** USING GREEN INK

17

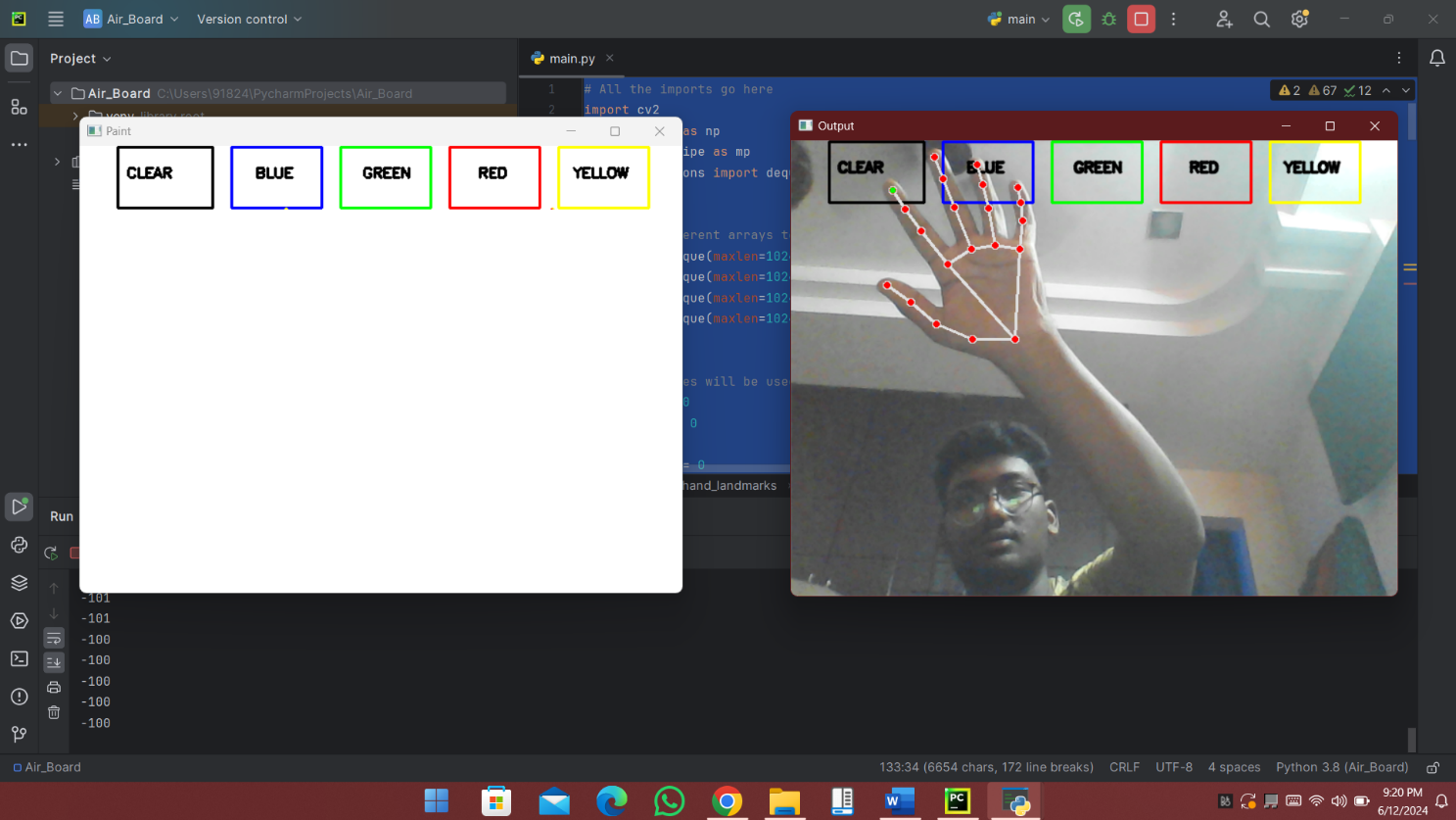


**Fig 7.3 SCREEN SHOT :** USING RED INK



**Fig 7.4 SCREEN SHOT :** USING YELLOW INK

18



**Fig 7.5 SCREEN SHOT :** USING CLEAR BUTTON

19

# SYSTEM TESTING

System testing for the Air Board using hand gestures involves various testing stages to ensure that all components work together seamlessly and meet the required performance, accuracy, and usability standards. Here is a comprehensive guide to the system testing process:

#### 8.1 ****Unit Testing****

Unit testing focuses on verifying the functionality of individual components in isolation. Each module should be tested to ensure it performs as expected.

**8.1.1 Hand Detection Module:**

* **Objective:** Ensure the module can accurately detect hands in different lighting conditions and backgrounds.
* **Test Cases:**
  + Test with hands in various positions and orientations.
  + Test in low, medium, and high lighting conditions.
  + Test with different backgrounds (plain, patterned, dynamic).

**8.1.2 Feature Extraction Module:**

* **Objective:** Verify that the module accurately extracts relevant features (hand shape, position, motion).
* **Test Cases:**
  + Extract features from different hand shapes and sizes.
  + Test the accuracy of position tracking.
  + Verify the consistency of motion detection.

**8.1.3 Gesture Classification Module:**

* **Objective:** Confirm that the machine learning model correctly classifies gestures.
* **Test Cases:**
  + Test with predefined gestures to check classification accuracy.
  + Test with custom gestures added by users.
  + Evaluate the model's performance with gestures performed at different speeds.

**8.1.4 User Interface (UI) Components:**

* **Objective:** Ensure the UI elements (drawing tools, object manipulation, command execution) function correctly.
* **Test Cases:**
  + Test each drawing tool (pen, brush, shapes).
  + Verify object manipulation functionalities (select, move, resize, rotate).
  + Check the execution of commands (open menu, save, undo).

20

#### 8.2 ****Integration Testing****

Integration testing ensures that the individual components work together as a cohesive system.

**8.2.1 Data Flow:**

* **Objective:** Verify the seamless flow of data from the camera to the processing unit and then to the display.
* **Test Cases:**
  + Test real-time hand movement capture and processing.
  + Ensure that the recognized gestures are correctly mapped to UI actions.

**8.2.2 System Performance:**

* **Objective:** Check the overall system performance, including responsiveness and latency.
* **Test Cases:**
  + Measure the time lag between performing a gesture and the corresponding action on the display.
  + Test the system's performance under different workloads (single user, multiple users).

**8.2.3 Calibration and Customization:**

* **Objective:** Validate the calibration and customization functionalities.
* **Test Cases:**
  + Test the calibration process to adjust for different users and environments.
  + Verify that users can successfully define and use custom gestures.

#### 8.3 ****User Testing****

User testing involves real users interacting with the system to provide feedback on usability, accuracy, and overall experience.

**8.3.1 Usability Testing:**

* **Objective:** Assess the ease of use and intuitiveness of the system.
* **Test Cases:**
  + Observe users performing various tasks (drawing, writing, object manipulation).
  + Collect feedback on the user interface and interaction design.

**8.3.2 Accuracy Testing:**

* **Objective:** Evaluate the accuracy of gesture recognition across different users and conditions.
* **Test Cases:**
  + Test with users of different ages, hand sizes, and dexterity levels.
  + Test in different environmental conditions (lighting, background noise).

21

**8.3.3 Satisfaction Testing:**

* **Objective:** Measure user satisfaction and identify areas for improvement.
* **Test Cases:**
  + Conduct surveys and interviews to gather user feedback.
  + Analyze feedback to identify common issues and suggestions.

#### 8.4 ****Performance Testing****

Performance testing focuses on the system's ability to handle different loads and its overall responsiveness.

**8.4.1 Latency Testing:**

* **Objective:** Ensure low latency for real-time interaction.
* **Test Cases:**
  + Measure the time from gesture performance to visual feedback.
  + Test under varying network conditions (if applicable).

**8.4.2 Stress Testing:**

* **Objective:** Assess the system's performance under high usage scenarios.
* **Test Cases:**
  + Test with multiple users simultaneously.
  + Evaluate the system's performance with prolonged use.

#### 8.5 ****Security Testing****

Security testing ensures that the system protects user data and prevents unauthorized access.

**8.5.1 Data Privacy:**

* **Objective:** Verify that user data is securely handled and stored.
* **Test Cases:**
  + Test data encryption mechanisms.
  + Ensure that data is processed locally or securely transmitted.

**8.5.2 Access Control:**

* **Objective:** Confirm that only authorized users can access the system.
* **Test Cases:**
  + Test user authentication mechanisms.
  + Verify permission management settings.

#### 22

#### 8.6 ****Regression Testing****

Regression testing ensures that new updates or changes do not negatively affect existing functionalities.

**8.6.1 Test Existing Features:**

* **Objective:** Verify that all previously tested features still work correctly after updates.
* **Test Cases:**
  + Re-run unit, integration, and user tests on the updated system.
  + Check for any new issues introduced by changes.

**8.6.2 Automated Testing:**

* **Objective:** Implement automated tests to streamline regression testing.
* **Test Cases:**
  + Develop automated test scripts for repetitive tasks.
  + Schedule regular automated test runs.

23

# CONCLUSION AND FUTURE ENHANCEMENT

# Conclusion :

The development and implementation of the Air Board using hand gestures represent a significant advancement in human-computer interaction, offering users a touchless and intuitive way to interact with digital interfaces. Through the integration of hardware components such as high-resolution cameras and powerful processing units, along with sophisticated software algorithms for gesture recognition and user interface design, the Air Board provides a seamless and immersive experience for a variety of applications.

During the system analysis, design, implementation, and testing phases, various challenges were addressed, including ensuring accurate gesture recognition in diverse environmental conditions, optimizing system performance to minimize latency, and addressing security and privacy concerns related to user data. Through rigorous testing, including unit testing, integration testing, user testing, and performance testing, the Air Board system has been validated for its reliability, accuracy, usability, and performance.

### **Future Enhancements :**

While the current implementation of the Air Board using hand gestures provides a solid foundation for touchless interaction, there are several avenues for future enhancements and improvements:

**Advanced Gesture Recognition:** Develop more robust algorithms for gesture recognition, capable of identifying complex and nuanced hand movements with higher accuracy.

**Haptic Feedback Integration:** Integrate haptic feedback mechanisms to provide tactile sensations, enhancing the user experience and providing sensory feedback during interactions.

**Expanded Gesture Vocabulary:** Increase the range of recognized gestures to support a wider variety of interactions and commands, allowing for more versatile use cases.

**Improved User Customization:** Enhance customization options to allow users to define and train custom gestures, tailoring the system to their specific preferences and needs.

By continuously innovating and iterating on the Air Board system, incorporating user feedback, and leveraging advancements in technology, the Air Board can evolve into a versatile and indispensable tool for a wide range of applications, including education, creative design, collaboration, and more.

24

# 10. BIBIOGRAPHY

# 10.1 REFERANCE :

* **"Gesture Recognition: A Survey"** - Li, Y., & Chen, X. (2012).

IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(6), 609-22. [Link](https://ieeexplore.ieee.org/document/6096697)

* **"Hand Gesture Recognition: A Survey"** - Pavithra, M. K., & Prema, K. V. (2014).

International Journal of Advanced Research in Computer and Communication Engineering, 3(10), 7849-7853. Link

* **"Real-time Hand Gesture Recognition Using Depth Sensors: A Review"** - Tayyab, M., & Chung, T. Y. (2019).

Multimedia Tools and Applications, 78(24), 35259-35285. [Link](https://link.springer.com/article/10.1007/s11042-019-07971-8)

* **"A Survey on Hand Gesture Recognition Techniques, Applications, and Challenges"** - Siddiqui, M. K., & Ali, S. (2018).

Journal of King Saud University - Computer and Information Sciences, 30(4), 430-448. [Link](https://www.sciencedirect.com/science/article/pii/S1319157817300578)

* **"Hand Gesture Recognition: A Comprehensive Review"** - Sharma, N., & Singh, P. K. (2020).

International Journal of Computer Science and Information Security, 18(5), 124-133. [Link](https://www.researchgate.net/publication/342784606_Hand_Gesture_Recognition_A_Comprehensive_Review)

25