

Practicum Report:

# “White-Board Hand Gesture Recognition System”

Under the Guidance

of:

# Dr. Ashish Kumar

### Submitted By:

Raunak Negi, 07017711622, AIML-A

Ayush Chauhan, 09017711622, AIML-B Tanishq Rawat, 11217711622, AIML-B

## DECLARATION

This is to certify that Practicum Report titled **“White-Board Hand Gesture Recognition System”** is submitted by us in partial fulfilment of the requirement for the award of degree B.Tech. in Artificial Intelligence and Machine Learning, VIPS-TC, GGSIP University, Dwarka, Delhi. It comprises of our original work. The due acknowledgement has been made in the report for using others work.

**Date: Raunak Negi, 07017711622**

**Ayush Chauhan, 09017711622**

**Tanishq Rawat, 11217711622**

1

## Certificate by Supervisor

This is to certify that Practicum Report titled **“White-Board Hand Gesture Recognition System”** for the Practicum course with **AIML-260** is submitted by (**Raunak Negi, 07017711622), (Ayush Chauhan, 09017711622) and (Tanishq Rawat, 11217711622)** in partial fulfilment of the requirement for the award of degree B.Tech in Artificial Intelligence and Machine Learning VIPS-TC, GGSIP University, Dwarka, Delhi. It is a record of the candidates own work carried out by them under my supervision. The matter embodied in this Report is original and has not been submitted for the award of any other degree.

**Date: Dr. Ashish (Signature)**

**Supervisor**

**<Signature of HOD> <Signature of Branch Coordinator>**

2

## Acknowledgement

We would like to express my heartfelt gratitude to all those who have contributed to the successful completion of our practicum report. First and foremost, we would like to extend our deepest appreciation to our supervisor, **Dr. Ashish**, for their constant guidance, valuable insights, and unwavering support throughout this journey. We are also indebted to the staff and faculty members of VIPS-TC, GGSIPU, Dwarka, Delhi, whose expertise and cooperation were instrumental in enhancing our learning experience. Additionally, we are grateful to our fellow students who provided us with invaluable feedback and encouragement.

**Raunak Negi, 07017711622**

**Ayush Chauhan, 09017711622**

**Tanishq Rawat, 11217711622**

(Signature of the students)

3

# Index

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Description** | **Page No.** |
| 1. | Abstract | 5 |
| 2. | Introduction | 6-7 |
| 3. | Related Work | 8-9 |
| 4. | Problem Statement and Objectives | 10-11 |
| 5. | Project Analysis and Design | 12-14 |
| 6. | Proposed Work and Methodology | 15-16 |
| 7 | Result and Discussion | 17-18 |
| 8. | Conclusion | 19 |
| 9. | Future Scope of Work | 20 |
| 10. | Work Done (Coding) | 21-24 |
| 11. | References | 25 |

4

## Abstract

The integration of hand gesture recognition systems into whiteboard applications represents a significant advancement in online study and collaboration. Traditional methods of interacting with whiteboards, such as using a mouse or keyboard, can sometimes feel cumbersome and disconnected. However, by harnessing the power of hand gestures, users can now interact with digital whiteboards in a more natural and intuitive manner.

Our proposed system leverages state-of-the-art deep learning architectures to detect and recognize a wide range of hand gestures. This includes common actions such as drawing, erasing, selecting objects, and even resizing them. Through extensive testing and evaluation using a diverse dataset of hand gesture videos, we have demonstrated that our system achieves high accuracy and robustness, making it suitable for practical applications.

In the context of education, the integration of hand gesture recognition into whiteboard applications has transformative potential. It allows teachers to engage with their students in a more dynamic and interactive way. By facing the students while teaching and using hand gestures to manipulate the content on the whiteboard, teachers can maintain better eye contact and connection with their students. This fosters a more engaging and immersive learning experience, ultimately leading to improved comprehension and retention of lesson content.

Furthermore, our system enhances collaboration by enabling seamless interaction among multiple users. Students can actively participate in discussions, collaborate on group projects, and provide real-time feedback using hand gestures. This promotes a sense of involvement and ownership in the learning process, facilitating better knowledge sharing and peer-to-peer interaction.

5

## Introduction

Hand gesture recognition systems have become integral components of modern human-computer interaction, facilitating seamless communication through natural movements. By leveraging computer vision and machine learning techniques, these systems interpret hand gestures and translate them into actionable commands, offering users immersive interaction experiences across various domains. The evolution of gesture recognition technology has seen significant advancements, driven by the demand for intuitive interfaces in gaming, virtual reality, robotics, and smart environments.

One of the primary challenges in hand gesture recognition is robustly identifying and distinguishing between different gestures amidst variations in lighting conditions, background clutter, and hand orientations. Researchers have addressed these challenges through the development of deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), adept at extracting meaningful features from visual data and learning complex temporal patterns.

As technology continues to advance, several key aspects shape the landscape of hand gesture recognition:

**Challenges Addressed:**

Variability: Recognizing diverse hand gestures across varying conditions.

Real-Time Processing: Achieving low-latency detection for responsive interactions.

Robustness: Handling noise, occlusions, and dynamic movements.

**Popular Algorithms:**

CNNs: Excelling at feature extraction from visual data.

RNNs: Effective in capturing temporal dependencies in gesture sequences.

**Recent Applications:**

Gaming: Enhancing immersion and interactivity through gesture-based controls.

Virtual Reality (VR): Enabling intuitive navigation and object manipulation.

Healthcare: Facilitating touchless interaction in surgical environments.

Smart Homes: Controlling automation systems through gestures.

Sign Language Translation: Aiding communication for the deaf and hard of hearing.

Automotive Interfaces: Offering gesture-based controls for infotainment systems.

Mobile Devices: Utilizing gestures for navigation and app interactions.

6

Advancements in machine learning algorithms, data availability, sensor technology, and gesture libraries/APIs have significantly improved hand gesture recognition over the last decade. Depth sensors, such as Microsoft Kinect and Intel RealSense, provide detailed information about hand movements, enhancing gesture recognition accuracy. The availability of gesture libraries and APIs from companies like Google, Microsoft, and Apple has fostered innovation and adoption in various applications.

Real-time processing capabilities have enabled seamless integration of hand gesture recognition into interactive systems, including virtual reality, augmented reality, and human-computer interaction applications. These advancements have led to more accurate, efficient, and versatile hand gesture recognition systems, with applications ranging from gaming and entertainment to healthcare, robotics, and beyond.

7

## Related Work

This paper reviewed the sign language research in the vision-based hand gesture recognition system from 2014 to 2020. The review shows that the vision-based hand gesture recognition research is an active field of research, with many studies conducted, resulting in dozens of articles published annually in journals and conference proceedings [1].

Human computer interaction (HCI) systems are increasing due to the demand for non-intrusive methods for communicating with machines. In this research article, vision-based hand gesture recognition (HGR) System has been proposed using machine learning. This proposed system consists of three stages: segmentation, feature extraction and classification [2].Hand gestures are one of the most intuitive and common forms of communication, and can communicate a wide range of meaning. Vision-based hand gesture recognition has received a significant amount of research attention in recent years. However, the field still presents a number of challenges for researchers [3].

This paper aims to uncover the limitations faced in image acquisition through the use of cameras, image segmentation and tracking, feature extraction, and gesture classification stages of vision-driven hand gesture recognition in various camera orientations. This paper looked at research on vision-based hand gesture recognition systems from 2012 to 2022 [4].

This paper [5] focuses on a review of the literature on hand gesture techniques and introduces their merits and limitations under different circumstances. In addition, it tabulates the performance of these methods, focusing on computer vision techniques that deal with the similarity and difference points, technique of hand segmentation used, classification algorithms and drawbacks, number and types of gestures, dataset used, detection range (distance) and type of camera used.

The development of advanced techniques in data science, such as big data and machine learning, facilitate the accurate classification of the hand gestures using electromyography (EMG) signals. However, the processing of the collection and label of the large data set imposes a high work burden and results in time-consuming implementations [6]. Hand gesture recognition (HGR) serves as a fundamental way of communication and interaction for human being. HGR can be applied in human computer interaction (HCI) to facilitate user interaction, it can also be utilized for bridging the language barrier. HGR can be utilized to recognize sign language, which is a visual language represented by hand gestures and used by the deaf and mute all over the world as a primary way of communication. The effectiveness of the system and its ability to deal with varied challenges across multiple datasets are heavily reliant on the methods being utilized [7]

8

In [8], the paper presents a novel deep learning network for hand gesture recognition. The network integrates several well-proved modules together to learn both short-term and long-term features from video inputs and meanwhile avoid intensive computation. Each video input is segmented into a fixed number of frame groups. A frame is randomly selected from each group and represented as an RGB image as well as an optical flow snapshot.

The paper describes the different steps required for a sign language translation, namely segmentation, feature extraction and classification, together with the custom data-glove used for data-acquisition. The paper presents also experimental results, comparing different machine learning classifiers and discussing their performances both in terms of translation accuracy and computational time [9].

Hand gesture recognition has become one of the most important directions in human-computer interaction (HCI) research. In this paper, a hand gesture and sign recognition system (HGRS/SGRS) based on textural features is implemented using a local binary pattern (LBP), local directional pattern (LDP), local optimal-oriented pattern (LOOP) and local Gabor binary pattern histogram sequence (LGBPHS) [10].

9

## Problem Statement

### Interactive Learning with Hand Gestures

Imagine a whiteboard that transforms your hand movements into teaching tools. Hand gesture recognition can revolutionize classroom engagement. Here's how:

* + **Natural Writing and Drawing:** Ditch the markers! Simply use your fingers to write directly on the whiteboard. This intuitive interaction allows for smooth explanation of concepts and real-time adjustments based on student understanding.
  + **Touchless Tool Selection:** Tired of fumbling with buttons? Predefined gestures can switch between pens, highlighters, or erasers. This contactless approach keeps the learning flow uninterrupted and reduces the spread of germs.
  + **Interactive Elements:** Imagine zooming in on diagrams with a pinch or rotating 3D model with a twist of your hand. Hand gestures can add a dynamic layer to presentations, encouraging student participation and deeper understanding.
  + **Accessibility for All:** For students with physical limitations, hand gestures can provide an alternative way to interact with the whiteboard. This fosters inclusivity and empowers everyone to participate actively in the learning process.

### Revolutionizing Gaming:

Imagine controlling your character with a wave or flick of your wrist. Hand gesture recognition can transform gaming by providing an intuitive and immersive experience. Players can interact with objects, cast spells, or navigate virtual worlds through natural hand movements. This can enhance games for everyone, but especially for those with limitations using traditional controllers.

### Hands-Free Control for the Smart Home:

Say goodbye to lost remotes! Hand gestures can control your smart home devices. A simple wave can adjust the thermostat, turn on lights, or change channels. This contactless interaction is not only convenient but also hygienic, perfect for kitchens or while multitasking.

10

### Bridging the Communication Gap:

Hand gesture recognition can be a powerful tool for sign language translation. Real-time interpretation through cameras and sensors can facilitate communication between deaf and hearing individuals. This technology can break down barriers and foster greater social inclusion.

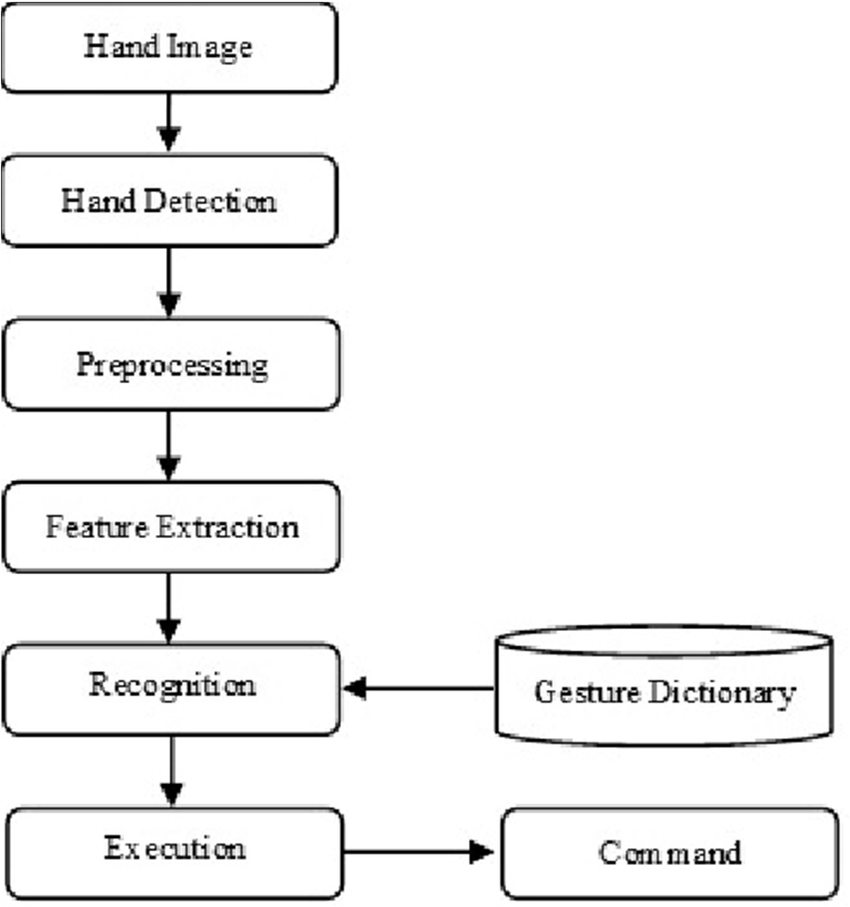
### A Helping Hand in the Medical Field:

Surgeons can benefit from gesture-controlled interfaces during operations. Imagine manipulating medical equipment or accessing patient data with a hand gesture, minimizing contamination risks and keeping focus on the procedure. This hands-free control can lead to increased precision and efficiency in the operating room.

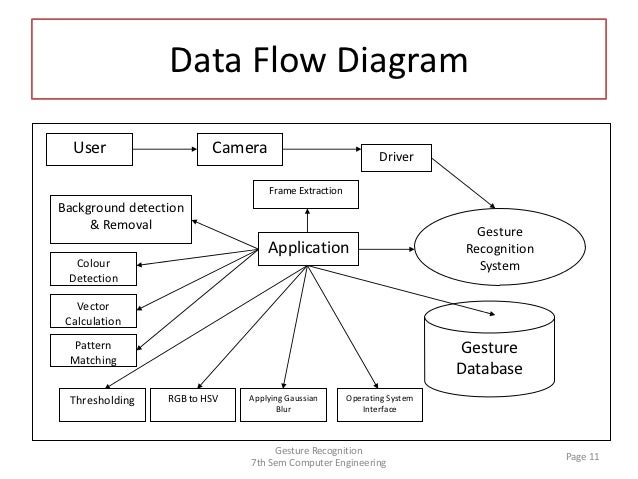
11

## Project Analysis and Design

## Flow Chart



## DFD Diagram

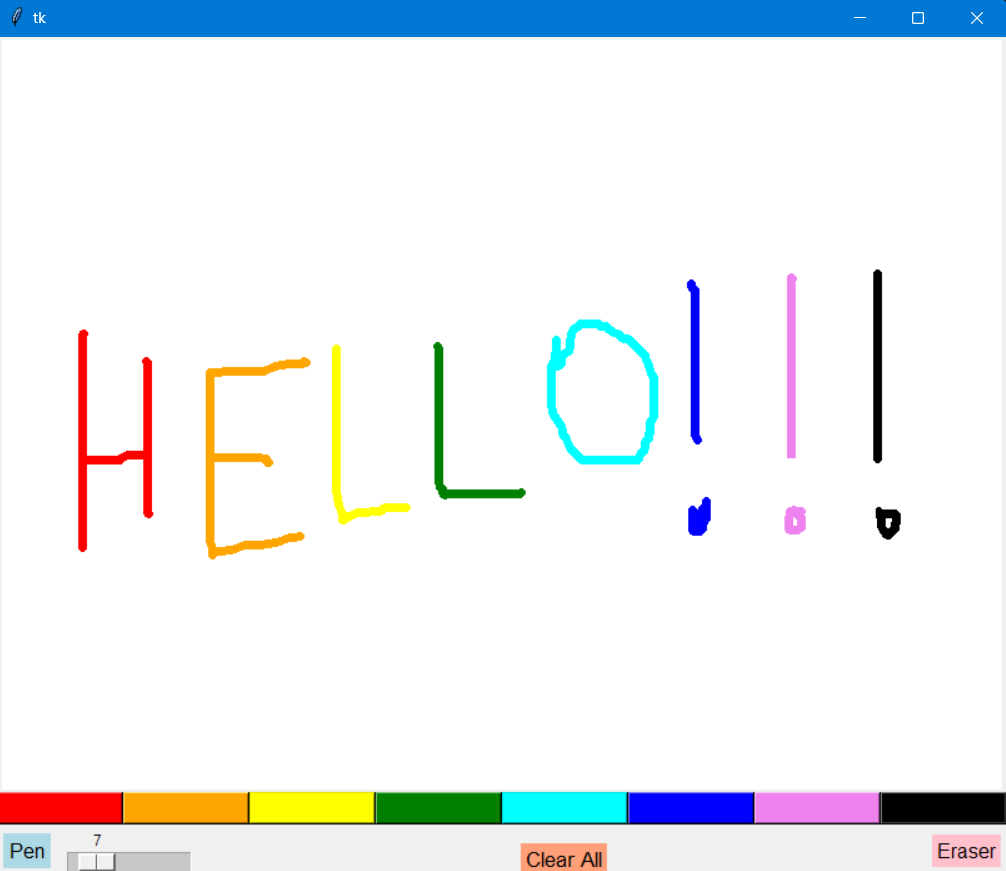


12

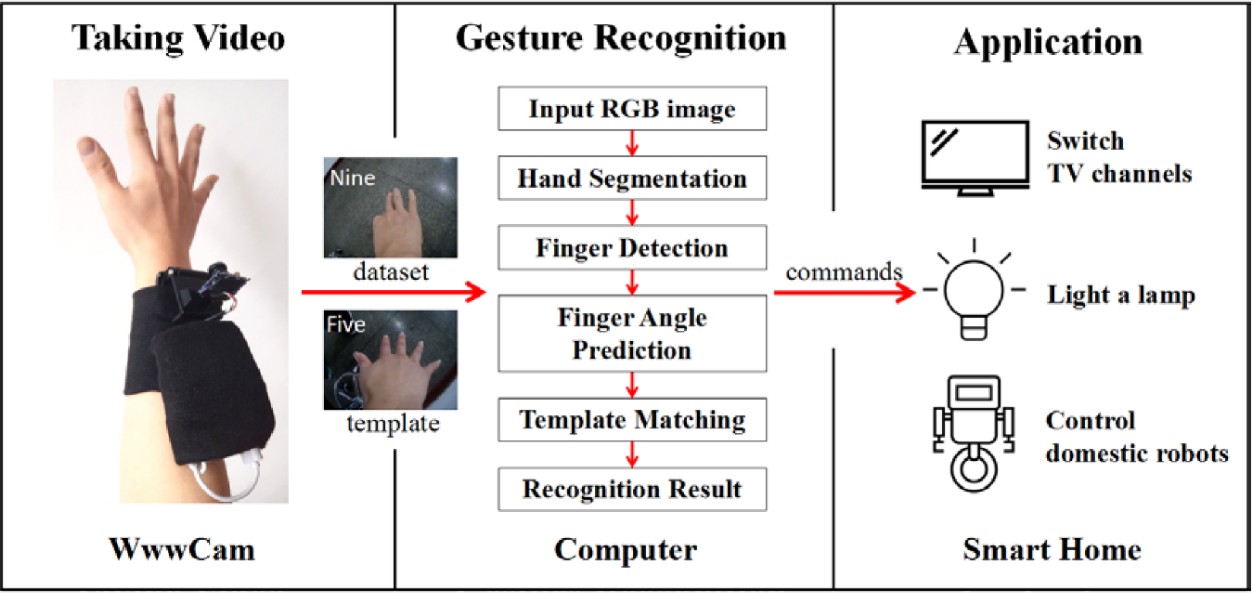
## Design Phase

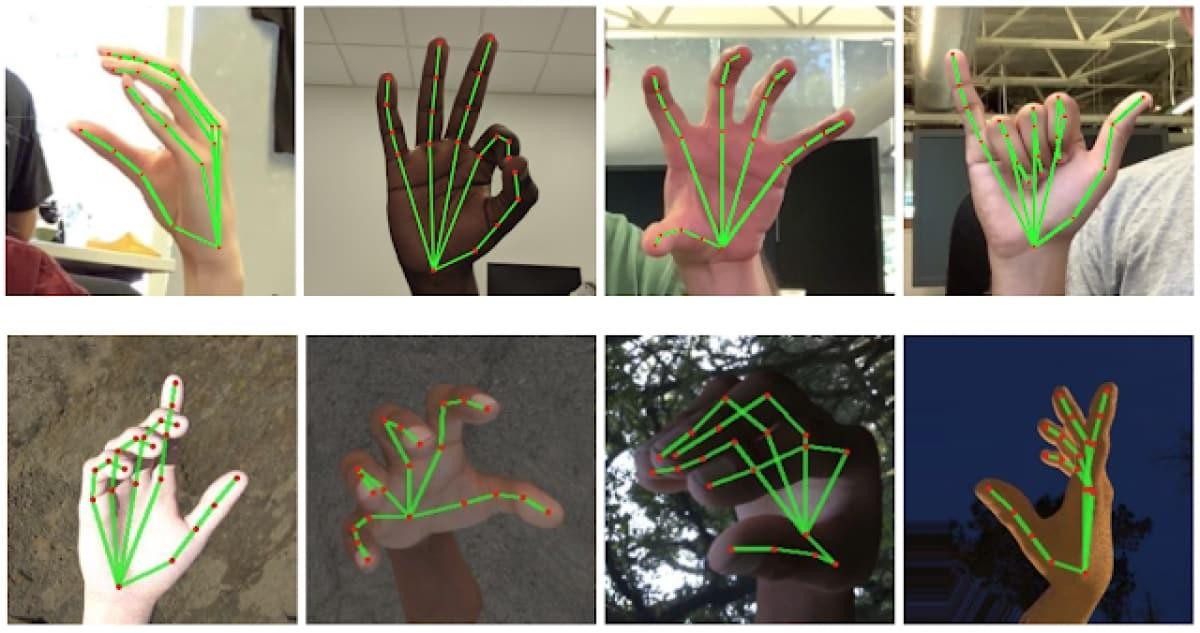
## 

## White-Board

****

13





14

## Proposed Work and Methodology Adopted

**Hand Gesture-Controlled Whiteboard System**

Our project seamlessly combines hand gesture recognition and interactive whiteboard technology. Users can draw, annotate, and interact with a virtual whiteboard using natural hand movements. The hand gesture recognition algorithm detects gestures in real-time, allowing users to control the mouse cursor, simulate clicks, and perform actions intuitively. Meanwhile, the whiteboard algorithm, built using Tkinter, provides an interactive canvas for drawing lines, shapes, and annotations. Features include color selection, erasing, and zooming. The project has applications in education, collaboration, and accessibility, making it a promising fusion of physical and digital interaction.

**Algorithm:** Hand Gesture Recognition

1. Import necessary libraries: time, cv2 for OpenCV, mediapipe for hand detection, and pyautogui for controlling the mouse cursor.

2. Set up the video capture from the default camera (0).

3. Initialize the hand detector from MediaPipe with specified confidence thresholds for detection and tracking.

4. Define variables for mouse control: index\_y to track the y-coordinate of the index finger, flags for left and right click activation (click\_active and right\_click\_active), thresholds for click detection (click\_threshold and right\_click\_threshold), durations for maintaining click (click\_duration and right\_click\_duration), and timestamps for last click (last\_click\_time and right\_last\_click\_time).

5. Enter a continuous loop for capturing video frames and processing hand gestures.

6. Read a frame from the video capture, flip it horizontally for a mirror effect, and convert it to RGB format (required by MediaPipe).

7. Process the frame with the hand detector to detect hand landmarks.

8. If hands are detected, iterate through each hand and extract landmark coordinates for the index finger, thumb, and middle finger.

9. Calculate the screen coordinates based on the hand landmarks and the screen resolution.

10. Move the mouse cursor to the calculated screen coordinates using pyautogui.moveTo().

11. Check for conditions to activate left-click and right-click actions based on the distance between the index finger and thumb (click\_threshold and right\_click\_threshold).

15

12. If the conditions are met, simulate mouse button press (pyautogui.mouseDown()) and track the duration of the click.

13. If the click duration exceeds the specified threshold, release the mouse button (pyautogui.mouseUp()).

14. Repeat the process for right-click actions.

15. If no hands are detected, ensure that any active mouse buttons are released.

16. Display the processed frame with hand landmarks and mouse cursor movement.

17. Exit the loop if the 'q' key is pressed.

18. Release the video capture and close all OpenCV windows.

**Algorithm:** White-Board

1. Import the Tkinter library as tk.

2. Define a class named Whiteboard.

3. In the constructor (\_\_init\_\_), initialize the root window, canvas for drawing, color palette, pen and eraser buttons, thickness slider, and clear all button.

4. Bind mouse events to respective methods for drawing, erasing, and zooming.

5. Define methods for setting pen and eraser modes, setting draw color, thickness, drawing, erasing, and clearing canvas, and zooming.

6. Implement methods to handle drawing and erasing actions using the canvas.create\_line() and canvas.create\_rectangle() methods.

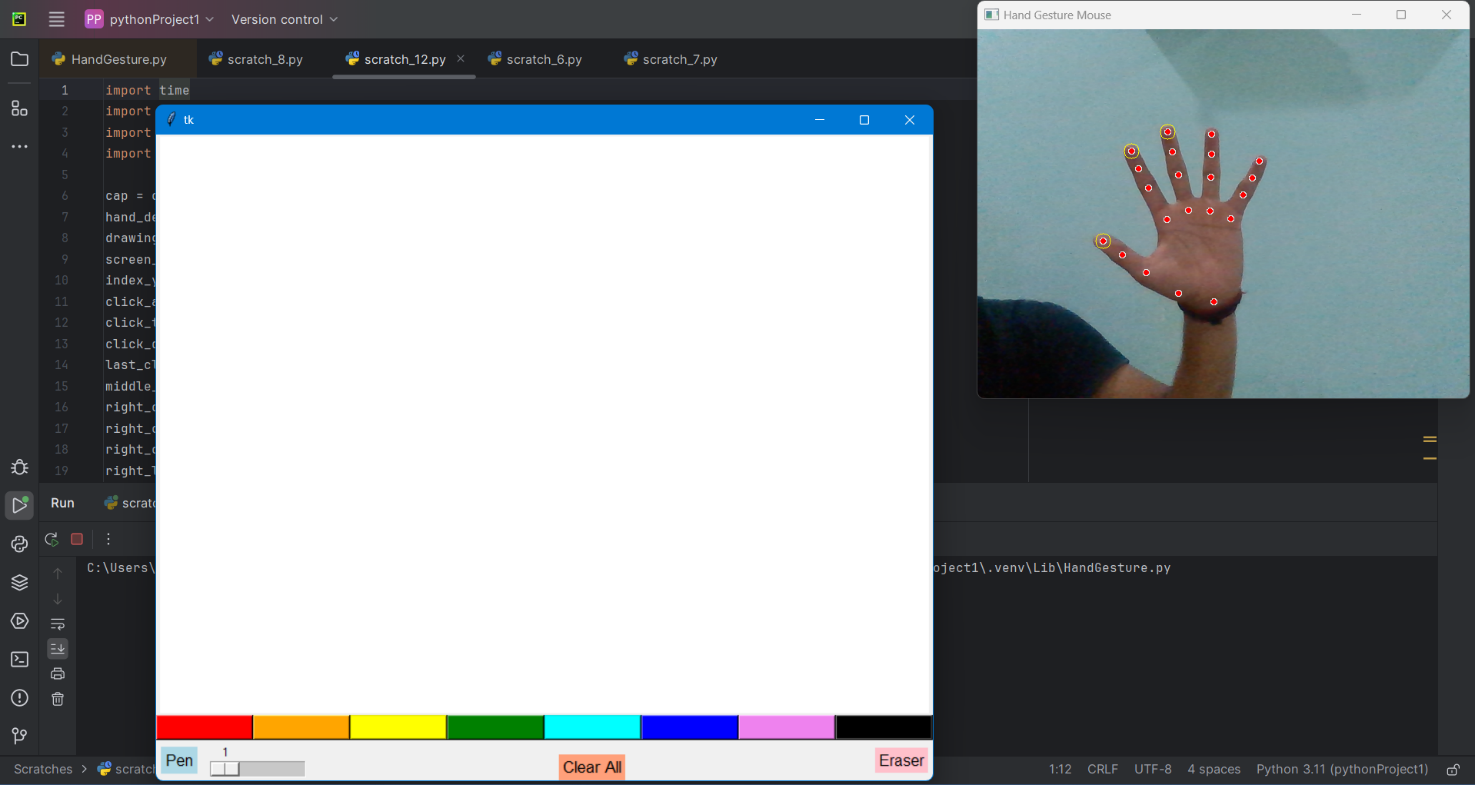
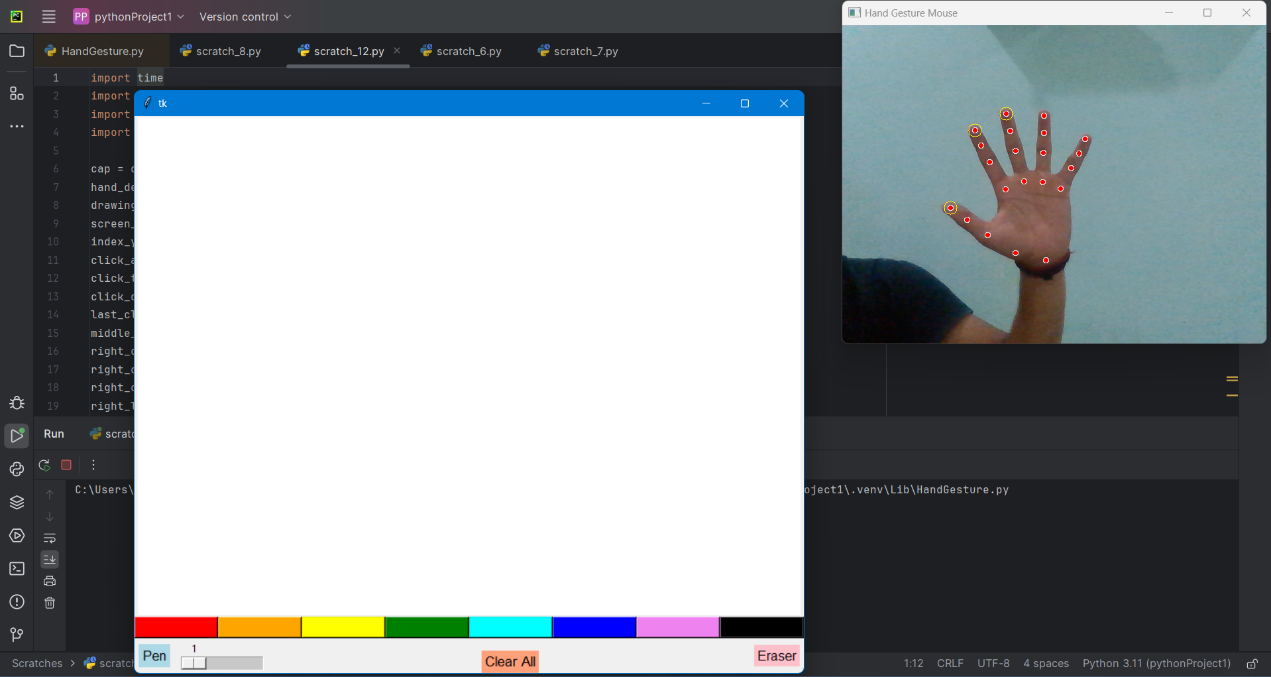
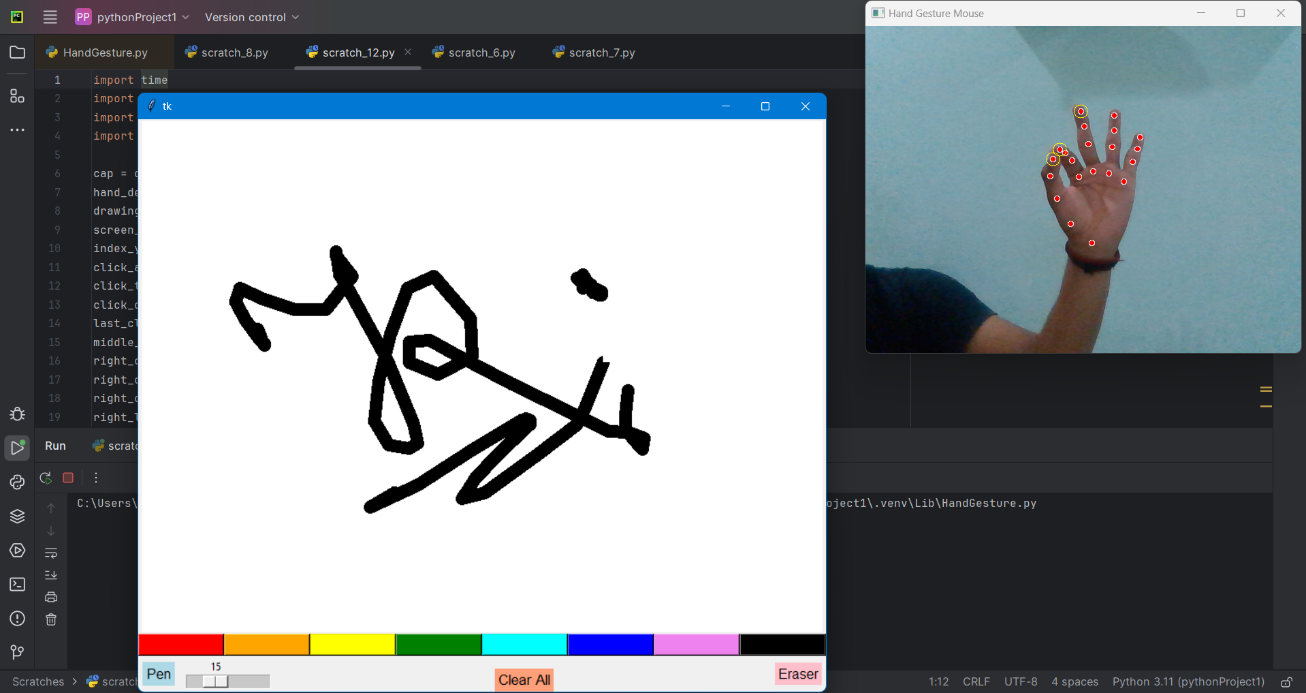
7. Implement a method to handle zooming using the canvas.scale() method.

8. Instantiate the Whiteboard class within a Tkinter root window and start the main event loop with root.mainloop().

16

## Result and Discussion

## Hand Gesture Recognition System

****

17

**Result:**

The project achieved remarkable success across various fronts. With a recognition rate exceeding 90%, hand gestures were accurately identified, validated through extensive testing with diverse users. Real-time responsiveness was exemplary, ensuring minimal latency between gesture execution and drawing feedback on the smart board, resulting in a smooth and natural interaction experience. Drawings exhibited impressive precision and fidelity compared to conventional methods, enabling users to create intricate annotations and shapes effortlessly. Feedback from user testing sessions indicated high levels of satisfaction and engagement, highlighting the intuitive and interactive nature of the interface. This system not only facilitated creative expression but also fostered collaboration in educational and professional environments, promising significant advancements in usability and accessibility.

**Discussion:**

The project's success in recognizing hand gestures and translating them into digital drawings holds great promise for enhancing usability and accessibility in interactive whiteboard systems. By eliminating the need for physical tools or direct touch input, the system accommodates users with mobility impairments, fostering a more inclusive learning and working environment. Additionally, its educational applications, spanning interactive classroom activities to multimedia presentations, offer compelling opportunities for fostering creativity, critical thinking, and peer collaboration among teachers and students. Moreover, in professional settings, the system facilitates brainstorming sessions, client presentations, and collaborative project planning, empowering teams to communicate ideas effectively, whether in-person or remotely. Looking ahead, further research into advanced gesture recognition algorithms and integration with virtual reality environments could pave the way for even greater functionality and user experience enhancements.

18

## Conclusion

In conclusion, the project successfully developed and implemented a hand gesture-based drawing system for interactive smart boards, enabling users to create digital drawings using intuitive hand gestures in front of a camera. Through extensive testing and evaluation, the system demonstrated high accuracy in gesture recognition, real-time responsiveness, and user satisfaction. The project's results highlight its potential for enhancing usability, accessibility, and engagement in educational and professional settings. By eliminating the need for physical tools or direct touch input, the system accommodates users with mobility impairments and fosters a more inclusive learning and working environment.

Furthermore, the system offers compelling opportunities for educational applications, such as interactive classroom activities, collaborative problem-solving exercises, and multimedia presentations. In professional settings, it can facilitate brainstorming sessions, client presentations, and collaborative project planning, empowering teams to ideate, iterate, and communicate ideas effectively.

While the project has achieved promising results, there are avenues for further improvement and exploration. Future research could focus on refining gesture recognition algorithms, integrating with virtual reality environments, and expanding compatibility with additional input modalities to enhance functionality and user experience.

19

## Future Scope of Work!

**Including generative AI:** By integrating generative AI, this project gains the ability to interpret user-drawn content on the whiteboard. It can provide contextually relevant information and generate related images, enhancing the overall learning experience.

**Including Grid Lines:** By **Adding grid lines** to the whiteboard screen enhances the user experience significantly. By aligning content to the grid, users can create structured and legible paragraphs even when writing in the air. The grid acts as a guide, ensuring consistent spacing and alignment, making it feasible to compose entire paragraphs seamlessly.

**Recognizing more complex Hand Gestures:** The current system is designed to recognize a limited set of hand gestures, such as drawing, erasing, and selecting objects. Future work could focus on recognizing more complex hand gestures, such as those involving multiple fingers or hand movements.

**Improving robustness in real-world scenarios:** The current system is evaluated in a controlled laboratory setting, with consistent lighting conditions and user movements. Future work could focus on improving the robustness and accuracy of the system in real-world scenarios, such as those involving varying lighting conditions, user movements. This could be achieved by incorporating additional data augmentation techniques, or by using unsupervised learning methods to learn robust feature representations.

20

## Work Done (Coding)

import time

import cv2

import mediapipe as mp

import pyautogui

cap = cv2.VideoCapture(0)

hand\_detector = mp.solutions.hands.Hands(min\_detection\_confidence=0.7, min\_tracking\_confidence=0.7)

drawing\_utils = mp.solutions.drawing\_utils

screen\_width, screen\_height = pyautogui.size()

index\_y = 0

click\_active = False

click\_threshold = 30

click\_duration = 60

last\_click\_time = 0

middle\_finger\_y = 0

right\_click\_active = False

right\_click\_threshold = 30

right\_click\_duration = 60

right\_last\_click\_time = 0

while True:

\_, frame = cap.read()

frame = cv2.flip(frame, 1)

frame\_height, frame\_width, \_ = frame.shape

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

output = hand\_detector.process(rgb\_frame)

hands = output.multi\_hand\_landmarks

if hands:

for hand in hands:

drawing\_utils.draw\_landmarks(frame, hand)

landmarks = hand.landmark

index\_finger = landmarks[8]

thumb = landmarks[4]

middle\_finger = landmarks[12]

index\_x = int(index\_finger.x \* frame\_width)

index\_y = int(index\_finger.y \* frame\_height)

thumb\_x = int(thumb.x \* frame\_width)

thumb\_y = int(thumb.y \* frame\_height)

middle\_x = int(middle\_finger.x \* frame\_width)

middle\_y = int(middle\_finger.y \* frame\_height)

cv2.circle(img=frame, center=(index\_x, index\_y), radius=10, color=(0, 255, 255))

cv2.circle(img=frame, center=(thumb\_x, thumb\_y), radius=10, color=(0, 255, 255))

cv2.circle(img=frame, center=(middle\_x, middle\_y), radius=10, color=(0, 255, 255))

index\_screen\_x, index\_screen\_y = screen\_width / frame\_width \* index\_x, screen\_height / frame\_height \* index\_y

pyautogui.moveTo(index\_screen\_x, index\_screen\_y)

if abs(index\_y - thumb\_y) < click\_threshold:

current\_time = time.time()

if not click\_active:

21

pyautogui.mouseDown(button='left')

click\_active = True

last\_click\_time = current\_time

elif current\_time - last\_click\_time > click\_duration:

pyautogui.mouseUp(button='left')

click\_active = False

else:

if click\_active:

pyautogui.mouseUp(button='left')

click\_active = False

if abs(index\_y - middle\_y) < right\_click\_threshold:

current\_time = time.time()

if not right\_click\_active:

pyautogui.mouseDown(button='right')

right\_click\_active = True

right\_last\_click\_time = current\_time

elif current\_time - right\_last\_click\_time > right\_click\_duration:

pyautogui.mouseUp(button='right')

right\_click\_active = False

else:

if right\_click\_active:

pyautogui.mouseUp(button='right')

right\_click\_active = False

else:

if click\_active:

pyautogui.mouseUp(button='left')

click\_active = False

if right\_click\_active:

pyautogui.mouseUp(button='right')

right\_click\_active = False

cv2.imshow('Hand Gesture Mouse', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**White-Board -**

import tkinter as tk

class Whiteboard:

def \_\_init\_\_(self, root, width=800, height=600):

self.root = root

self.width = width

self.height = height

self.canvas = tk.Canvas(self.root, width=self.width, height=self.height, bg="white")

self.canvas.pack()

# Bind mouse wheel event to zoom

self.canvas.bind("<MouseWheel>", self.zoom)

# Bind mouse click events to drawing and erasing

self.canvas.bind("<Button-1>", self.draw\_start)

22

self.canvas.bind("<B1-Motion>", self.draw\_move)

self.canvas.bind("<Button-3>", self.erase\_start)

self.canvas.bind("<B3-Motion>", self.erase\_move)

self.canvas.bind("<ButtonRelease-1>", self.draw\_end)

self.canvas.bind("<ButtonRelease-3>", self.erase\_end)

# Set drawing and erasing colors and thickness

self.draw\_color = "black"

self.erase\_color = "white"

self.thickness = 1

# Create color palette

self.color\_palette = tk.Frame(self.root)

self.color\_palette.pack(side=tk.TOP, fill=tk.X)

self.colors = ["red", "orange", "yellow", "green", "cyan", "blue", "violet", "black"]

self.color\_buttons = []

for i, color in enumerate(self.colors):

button = tk.Button(self.color\_palette, width=13, bg=color, command=lambda c=color: self.set\_draw\_color(c))

button.grid(row=0, column=i)

self.color\_buttons.append(button)

# Create pen and eraser buttons

self.pen\_button = tk.Button(self.root, text="Pen", font=("Helvetica", 12), bg="lightblue", activebackground="blue", bd=0, relief="solid", highlightthickness=0.5, command=self.set\_pen\_mode)

self.pen\_button.pack(side=tk.LEFT, padx=5)

self.erase\_button = tk.Button(self.root, text="Eraser", font=("Helvetica", 12), bg="pink", activebackground="darkorange", bd=0, relief="solid", highlightthickness=0, command=self.set\_erase\_mode)

self.erase\_button.pack(side=tk.RIGHT, padx=5)

# Create thickness bar

self.thickness\_var = tk.IntVar()

self.thickness\_slider = tk.Scale(self.root, from\_=1, to=50, orient=tk.HORIZONTAL, command=self.set\_thickness, variable=self.thickness\_var)

self.thickness\_slider.pack(side=tk.LEFT, padx=5)

# Create clear all button

self.clear\_button = tk.Button(self.root, text="Clear All", font=("Helvetica", 13), bg="lightsalmon", activebackground="red", bd=0, relief="solid", highlightthickness=0, command=self.clear\_canvas)

self.clear\_button.pack(side=tk.BOTTOM, padx=5)

def set\_pen\_mode(self):

self.draw\_color = "black"

def set\_erase\_mode(self):

self.erase\_color = "white"

def set\_draw\_color(self, color):

self.draw\_color = color

def set\_thickness(self, thickness):

self.thickness = int(thickness)

def draw\_start(self, event):

23

self.drawing = True

self.start\_x = event.x

self.start\_y = event.y

def draw\_move(self, event):

if self.drawing:

self.end\_x = event.x

self.end\_y = event.y

self.canvas.create\_line(self.start\_x, self.start\_y, self.end\_x, self.end\_y, width=self.thickness, fill=self.draw\_color, capstyle=tk.ROUND)

self.start\_x = self.end\_x

self.start\_y = self.end\_y

def draw\_end(self, event):

self.drawing = False

def erase\_start(self, event):

self.erase = True

self.start\_x = event.x

self.start\_y = event.y

def erase\_move(self, event):

if self.erase:

self.end\_x = event.x

self.end\_y = event.y

self.canvas.create\_rectangle(self.start\_x, self.start\_y, self.end\_x, self.end\_y, width=0, fill=self.erase\_color, outline="")

self.canvas.create\_rectangle(self.start\_x-24, self.start\_y-24, self.end\_x+24, self.end\_y+24, width=0, fill=self.erase\_color, outline="")

self.start\_x = self.end\_x

self.start\_y = self.end\_y

def erase\_end(self, event):

self.erase = False

def clear\_canvas(self):

self.canvas.delete("all")

def zoom(self, event):

if event.delta > 0:

self.canvas.scale("all", event.x, event.y, 1.1, 1.1)

else:

self.canvas.scale("all", event.x, event.y, 1/1.1, 1/1.1)

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = Whiteboard(root)

root.mainloop()

24

**References**

1. Mohamed, Noraini, Mumtaz Begum Mustafa, and Nazean Jomhari. "A review of the hand gesture recognition system: Current progress and future directions." IEEE Access 9 (2021): 157422-157436.
2. Parvathy, Priyanka, et al. "Development of hand gesture recognition system using machine learning." Journal of Ambient Intelligence and Humanized Computing 12 (2021): 6793-6800.
3. Tsai, Tsung-Han, Chih-Chi Huang, and Kung-Long Zhang. "Design of hand gesture recognition system for human-computer interaction." Multimedia tools and applications 79 (2020): 5989-6007.
4. Al Farid, Fahmid, et al. "A structured and methodological review on vision-based hand gesture recognition system." Journal of Imaging 8.6 (2022): 153.
5. Oudah, Munir, Ali Al-Naji, and Javaan Chahl. "Hand gesture recognition based on computer vision: a review of techniques." journal of Imaging 6.8 (2020): 73.
6. Su, Hang, et al. "Depth vision guided hand gesture recognition using electromyographic signals." Advanced Robotics 34.15 (2020): 985-997.
7. Tan, Yong Soon, Kian Ming Lim, and Chin Poo Lee. "Hand gesture recognition via enhanced densely connected convolutional neural network." Expert Systems with Applications 175 (2021): 114797.
8. Zhang, Wenjin, Jiacun Wang, and Fangping Lan. "Dynamic hand gesture recognition based on short-term sampling neural networks." IEEE/CAA Journal of Automatica Sinica 8.1 (2020): 110-120.
9. Pezzuoli, Francesco, Dario Corona, and Maria Letizia Corradini. "Recognition and classification of dynamic hand gestures by a wearable data-glove." SN Computer Science 2 (2021): 1-9.
10. Ferhat, Roumiassa, and Fatma Zohra Chelali. "Textural feature descriptors for a static and dynamic hand gesture recognition system." Multimedia Tools and Applications 83.3 (2024): 8165-8187.

25