An Industrial Oriented Mini Project (CS755PC)

On

Virtual Palette Using Artificial Intelligence

Submitted

In partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

In

Computer Science and Data Science

By

Ms. M.Meghana

(21261A6734)

Under the guidance of

Ms. A.Swapna

Assistant Professor

Department of Computer Science and Engineering

Mahatma Gandhi Institute of Technology

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

Kokapet(V),Gandipet(M),Hyderabad.

Telangana- 500075.

2024-2025

MAHATMAGANDHIINSTITUTE OFTECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University

Hyderabad)GANDIPET, HYDERABAD – 500075, Telangana

CERTIFICATE

This isto certify that the project entitled – **"VIRTUAL PALETTE USING ARTIFICIAL INTELLIGENCE** "is being submitted by **M.MEGHANA** bearing RollNo. **21261A6734** in partial fulfillment of the requirements for the Industrial Oriented Mini Project (CS755PC) in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** is a record of bonafide work carried out by him/her under our guidance and supervision.The results of the investigations enclosed in this report have been verified and found satisfactory.

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
Ms. A.Swapna	**Dr.M. Rama Bai**
(Assistant Professor)	(Professor & HoD)
Department of ET	Department of ET

EXTERNAL EXAMINER

DECLARATION

This is to certify that the work reported in this project titled– **"VIRTUAL PALETTE USING ARTIFICIAL INTELLIGENCE"** is a record of work done by me in the Department of Emerging Technologies, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

M.MEGHANA (21261A6734)

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to**Dr. G. Chandra Mohan Reddy, Principal MGIT**, for providing the working facilities in college.

I wish to express my sincere thanks and gratitude to **Dr. M. Rama Bai, Professor and HoD**, Department of Emerging Technologies, MGIT, for all the timely support and valuable suggestions during the period of project.

I am extremely thankful to our Project Coordinators**. Mallela Srikanth (Assistant Professor)** and **Ms. Karuna Verma (Assistant Professor),** Department of Emerging Technologies,MGIT,major project coordinators for their encouragement and support throughout the project.

Iam extremely thankful land indebted to my internal guide **Ms.A.Swapna**, **Assistant Professor**, Department of Emerging Technologies, for her constant guidance, encouragement and moral support throughout the project.

Finally, I would also like to thank all the faculty and staff of ET Department who helped me directly or indirectly, for completing this project.

M.MEGHANA (21261A6734)

TABLE OF CONTENTS

LIST OF TABLES

ABSTRACT

Envision the ability to draw virtually by simply waving a pen in the air, with the corresponding strokes appearing on the screen in real time—achieved without the need for specialized hardware. In today's technological era, advancements in image processing and artificial intelligence have significantly impacted various fields, enhancing human capabilities with minimal assistance. 'Virtual Palette' is an innovative tool designed to facilitate digital painting through natural hand movements. This application employs hand tracing and finger movement recording to translate gestures into marks on the screen. By utilizing the computer or laptop camera, it leverages image processing to interpret hand movements, transforming the user into a skilled illustrator capable of selecting colors and drawing with precision. The system is powered by OpenCV and MediaPipe, ensuring accurate recognition of hand gestures, which is crucial for effective Human-Computer Interaction (HCI). 'Virtual Palette' is developed for a range of applications, including artistic expression, interactive experiences, educational purposes, and technological innovation demonstrations

1. INTRODUCTION

1.1 MOTIVATION

Digital painting tools are typically designed for experienced artists, presenting a steep learning curve for beginners and hobbyists. This creates a significant barrier for individuals who are interested in exploring digital art but are intimidated by complex software and the technical skills required. The problem lies in the lack of accessible, user-friendly tools that enable novice users to create visually compelling artwork without extensive training. The goal is to develop an AI-powered virtual palette that empowers users of all experience levels to create digital art. By integrating AI, the system will provide intelligent guidance, simplifying the creative process and making artistic tools more intuitive and accessible. This project aims to make digital painting enjoyable and achievable for everyone, regardless of their background or skill level. One of the most intriguing and difficult research areas in the world of AI is writing in the air. These were the issues that troubled us, and it was as a result that we came up with the project idea for the air canvas. Everyone wants to sketch or draw in the air with their hands.

1.2 PROBLEM STATEMENT

One of the most intriguing and difficult research areas in the world of AI is writing in the air. These were the issues that troubled us, and it was as a result that we came up with the project idea for the air canvas. Everyone wants to sketch or draw in the air with their hands.

Psychological and Ergonomic Considerations:

Psychological Impact: AI Virtual Painting should enhance users' sense of accomplishment and creative fulfillment, boosting self-confidence.

Ergonomic Optimization: Analyzing hand and arm movements to optimize tool placement and reduce physical strain for a comfortable user experience.

1.3 PROJECT O B J E C T I V E

The purpose it to create an interesting platform for writing, drawing and etc . We can also minimize the usage of hardware components, it also saves our time and it is a kind of fun activity to kids. Social development is not a people's will, but the needs of human life are the main driving force anyway. The same situation happens in art. In the present circumstances, digital art and traditional art are inclusive of the symbiotic state, so we need to systematically understand the basic knowledge of the form between digital art and traditional art. The traditional way includes pen and paper, chalk and board method of writing. The essential aim of digital art is of building hand gesture recognition system to write digitally. Here we aim to create an external hardware free virtual painter that is easy to use. This also helps dumb people to do the things so easily. Here we just need to use our hand to draw or write, we need to use our two fingers to select a color, one finger to draw and we will have multiple colors to select as required also there will be an eraser incase if you go wrong.

1.4 EXISTING SYSTEM

1.4.1 ADOBE COLOR:

Several existing systems use AI to create Virtual palette, especially for art, design and fashion. These tools leverage AI to analyze color harmony, trends, and even user preferences. Extract color themes from images Apply different color rules Integration with Adobe Creative Cloud for design workflows.

1.4.2. COLOR MIND:

General color palettes from scratch Learns color combinations from data sets like images and media provides recommendation based on user input. This chapter gives an overview of the software and hardware components required for our project.

2. SYSTEM REQUIREMENTS

2.1 SOFTWARE REQUIREMENTS

- Operating System: Windows 10/11
- Coding language: Python 3.10.7
- Development Environment: V S Code

2.2 HARDWARE REQUIREMENTS

- System: Intel i5 or above
- Storage: Sufficient Storage
- RAM: 16 to 64 GB

2.3 LIBRARIES

2.3.1 OPEN CV

Open CV, short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers under the OpenCV Foundation.
OpenCV allows you to perform various operations in the image.

- **Read the Image :** OpenCV helps you to read the image fro file or directly from camera to make it accessible for further processing.
- **Image Enhacncement :** You will be able to enhance image by adjusting the brightness , sharpness or contract of the image. This is helpful to visualize quality of the image.
- **Object detection:** As you can see in the below image object can also be detected by using Open CV , Bracelet , watch , patterns, faces can be detected. This can also include to recognize faces , shapes or even objects .
- **Image Filtering:** You can change image by applying various filters such as blurring or Sharpening.
- **Draw the Image:** OpenCV allows to draw text, lines and any shapes in the images.
- **Saving the Changed Images:** After processing , You can save images that are being modified for future analysis.

2.3.2 MEDIAPIPE

MediaPipe is a comprehensive framework designed to simplify the development of complex computer vision applications. It offers a wide range of pre-trained models and customizable components, allowing developers to integrate sophisticated perception capabilities into their projects with relative ease. Developed by the Google Research Brain Team, MediaPipe is built on TensorFlow, making it accessible to a broad audience of researchers, engineers, and developers

```
.pip install media Pipe
pip install open cv-python
```

2.4 MODULES

2.4.1 Hand Tracking:

Technology: Hand tracking involves recognizing and analyzing the movements and gestures of a user's hands.

Purpose: It enables the application to interpret specific gestures (such as drawing or selecting) and convert them into actions within the virtual painting interface.

Tools: MediaPipe by Google is a leading library used for hand tracking, offering real-time detection andanalysis of hand movements.

2.4.2 Gesture Recognition:

Technology: Gesture recognition is a branch of hand tracking focused on interpreting specific hand movements and gestures.

Purpose: It enables the application to perform various actions, such as tool switching, brush size adjustment, and color changes, based on the gestures made by the user.

Tools: Libraries and frameworks like Tensor Flow, Keras, and custom machine learning models are employed to recognize and classify gestures.

3 . LITERATURE SURVEY

Table 3.1 Comparison of Literature survey

S. NO	TITLE	AUTHORS	YEAR	ALGORITHAM AND METHODOLOGY	ADVANTAGES	DISADVANTAGES
1	AI-based Virtual Palette for Digital Artists	R.Silva,A.Kumar	2023	Developed an AI tool based on image segmentation and color transfer techniques	Allowed artists to extract and apply colors from real-world images	Struggled with highly abstract or unconventional artistic styles
2	Deep learning Approaches for Color Scheme Optimization	K.Brown,T.Lee	2022	Used reinforcement learning to recommend ideal color combinations	Improved efficient and aesthetic accuracy of palette selection	Limited to pre – defined color combinations
3.	Smart Palette AI-Assisted Color Selection Tool	C.Liu,B.Garcia	2021	Used deep learning to predict and suggest color harmonies	Facilitated efficient workflow for graphics designers	Limited customization for niche artistic requirements
4	Virtual Color Palette Generation using AI	J.Smith,L.Doe	2020	Trained GANs to generate color palettes based on user input	Successfully generated color palettes based on user images or preferences	Limited scope for highly artistic or complex paletttes
5	Intelligent Color Palette Assistant for Designers	M.Taylor,S.Wang	2019	Combined CNN and KNN for color recognition and recommendation	Provided personalized color suggestions based on design need	Did not account for emotional and cultural aspects of color

4. SYSTE DESIGN

4.1 System Design

In this phase, system and software design documents are developed based on the requirements specification document. This process defines the overall system architecture. There are two types of design documents produced during this phase:

4.1.1 High-Level Design (HLD)

- Offer a brief description and name for each module.
- Summarize the functionality provided by each module.
- Explain the interface relationships and dependencies among the modules.
- Define the database tables and highlight their key elements.
- Present detailed architecture diagrams, including the technologies

4.1.2 Low-Level Design (LLD)

- The functional logic for each module in detail.
- Database table structures, including types and sizes.
- Full specification of interfaces.
- Solutions for all dependency issues.
- A catalog of error messages.

4.2 PROPOSED METHODOLOGY:

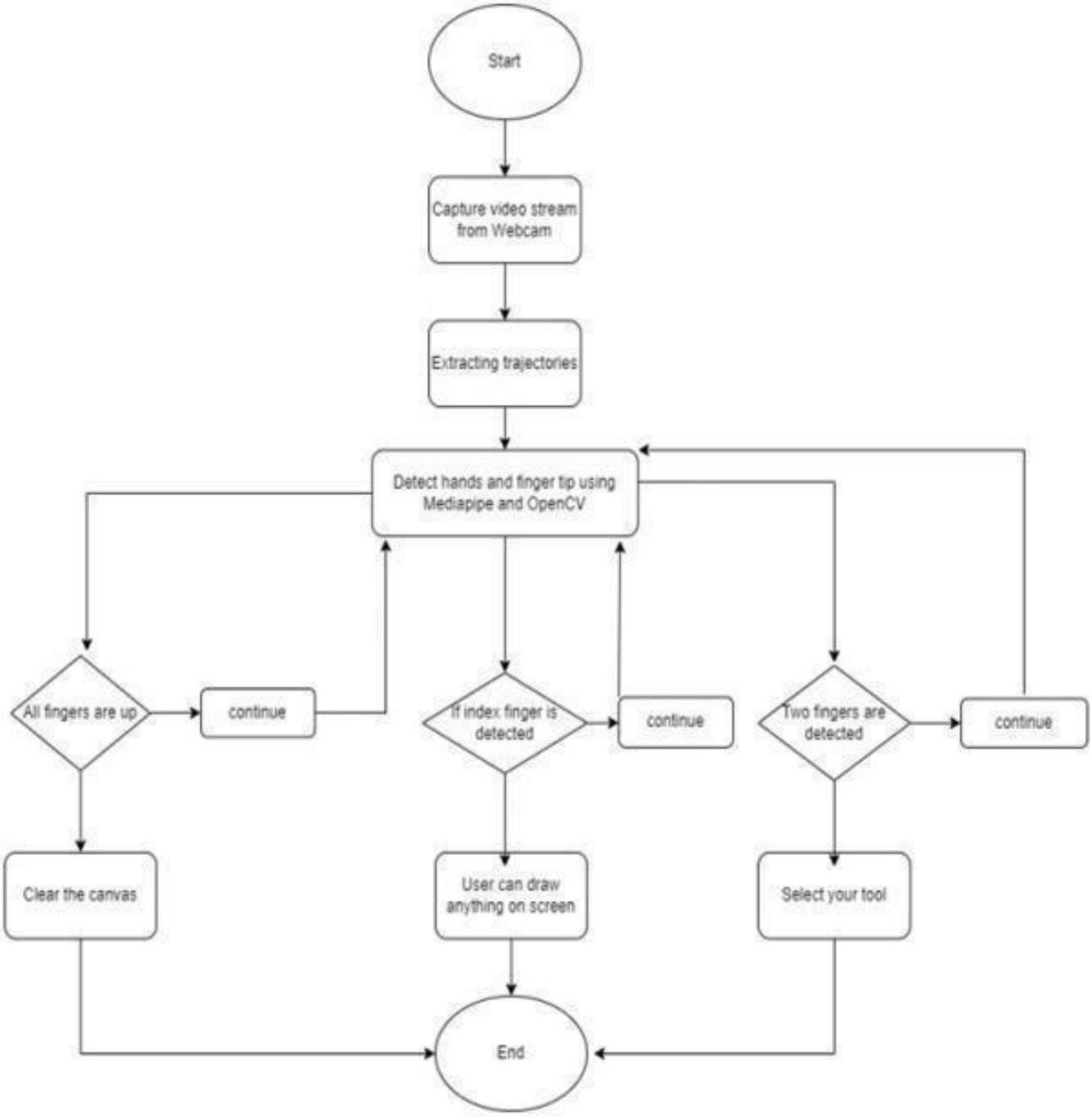

Figure-4.2.1 Proposed Methodology

To carry out the AI Virtual Painter project, you will need to integrate various elements such as computer vision, hand tracking, and interactive graphics. This process involves capturing hand movements, translating them into painting actions, and rendering these actions on a digital canvas. Here's a concise overview of the implementation steps:

4.2.1 SET UP DEVELOPMENT ENVIRONMENT:

Start by installing the essential software and libraries, including Python, MediaPipe, OpenCV, and any other necessary dependencies. Ensure that your environment is configured correctly to support these tools. Additionally, set up a virtual environment if needed to manage packages efficiently and avoid conflicts.

4.2.2 UNDERSTAND MEDIAPIPE AND OPENCV:

Get acquainted with the features of MediaPipe and OpenCV, paying close attention to the modules that handle hand tracking and computer vision. Dive into how these libraries manage image data and process hand gestures. This understanding is essential for effectively integrating their capabilities into your virtual painter project and ensuring seamless interaction between the components.

4.2.3 CAMERA SETUP:

Connect and configure the camera to capture video input, ensuring it has a clear and unobstructed view of the user's hand movements. Position the camera to achieve the best angle and lighting for accurate tracking. Additionally, verify that the camera's resolution and frame rate are set appropriately to capture smooth and detailed hand gestures.

4.2.4 INITIALIZE HAND TRACKING:

Employ Media Pipe to establish hand tracking by detecting and pinpointing the user's hands in the camera feed. This step involves configuring Media Pipe to accurately recognize and track hand gestures in real-time. Ensure that the tracking system is calibrated for precision to facilitate smooth and responsive interaction with the virtual painting applications.

4.2.5 EXTRACT HAND LANDMARKS:

Use the hand tracking data to identify and extract key points on the user's hand. These points will help track hand movements and gestures, which are essential for interacting with the virtual painting setup.

4.2.6 OPTIMIZATION:

Improve the code for better performance by addressing factors like frame rate, responsiveness, and overall efficiency. Optimize the system to ensure smooth operation and effective interaction with the virtual painting application.

4.2.7 TESTING:

Test the system to ensure hand tracking is accurate, drawing is smooth, and the user experience is satisfactory. Check how it performs in different situations to confirm it works as expected.

4.2.8 DRAW ON THE CANVAS:

Use the tracked hand landmarks to let users draw on the digital canvas. Implement controls that respond to hand movements or gestures to adjust the drawing tools.

4.2.9 ITERATE AND IMPROVE:

Collect feedback from users to identify areas for enhancement. Make iterative updates to address anyissues and refine the project based on user experiences. Continuously test and adjust the system to improve functionality, performance, and overall user satisfaction.

4.3 WORKFLOW OF THE SYSTEM:

To guarantee this, the interface is very straightforward and easier to understand by the human. Humans are capable of drawing in accordance with their choice what they actually want to draw without any problems or any type of difference. In the future, it is really very useful for making the kids learn things and draw in schools in an interactive and appropriate way or it gives the clear vision to kids to explore it and get it easily and faster. The frames are read, converted to the HSV color system, and made simple for color identification. Construct the canvas frame and affix the appropriate link buttons to it. Now, set the print bar values to locate the colored marker mask. The mask is preprocessed using phonological procedures like erosion and dilation. The final or end point is to draw the points that have been stored in a given array on the frames and canvas. An efficient way of communication that eliminates the necessity for writing and lowers the use of mobile devices and laptops. The input image is converted by the algorithm code into HSV space, a very good color system that is ideal for color tracking.

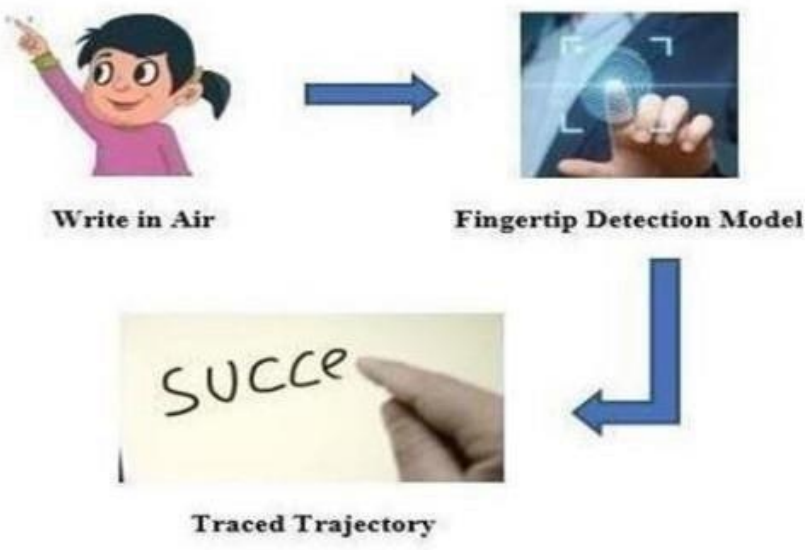

Figure 4.3.1 Work flow of System

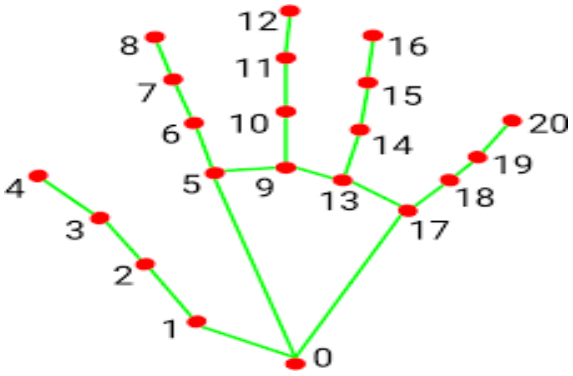

Figure 4.3.2 Mediapipe’s Hand Landmark Model

4.3.3 Gesture Recognition Model

It identifies specific gestures, such as the number of fingers extended, to switch between different modes (e.g., selection mode with two fingers and drawing mode with one finger). This functionality enhances user interaction by allowing intuitive control over the painting process.

4.3.4 Drawing Engine:

- The drawing engine is responsible for rendering the brush strokes on the canvas based on input from the hand tracking and gesture recognition modules.
- **Rendering Process**: It translates the user's hand movements into digital strokes, allowing users to create artwork on the screen. The engine must ensure that the rendering is smooth and responsive to maintain a natural drawing experience

4.3.5 Color and Brush Management:

- This module manages the selection of colors and brush types.
- **Palette Display**: It provides a user-friendly color palette from which users can choose colors for their artwork. Users can also select different brush types, allowing for customization of their drawing tools.

Figure 4.3.6 Header

4.3.7 Fingertip Detection:

The method, takes use of a fingertip. We think it should be possible for individuals to write without having to lift or carry anything around the stylus. Each frame's fingertip was identified using Deep Learning algorithms, which produced a set of coordinates.

4.4 UML Design:

Unified Modeling Language (UML) is a general-purpose visual modeling language designed to standardize the way systems are visualized. Unlike programming languages, UML is a toolfor creating diagramsthat represent a system's behavior and structure. It assists software engineers, business professionals, and system architects in modeling, designing, and analyzing systems. UML was officially adopted as a standard by the Object Management Group (OMG) in 1997 and has been managed by them since. The International Organization for Standardization (ISO) also recognized UML as an approved standard in 2005. Over the years, UML has undergone revisions and continues to be reviewed periodically.

Do we really need UML?

- Complex applications require collaboration and planning across multiple teams, necessitating a clear and concise communication method to ensure effective coordination.
- Since businessmen do not typically understand code, UML becomes crucial for conveying essential requirements, functionalities, and processes of the system to non-programmers.
- UML is closely associated with object-oriented design and analysis, utilizing elements and their relationships to create diagrams that represent the system's structure and behavior.

The Primary goals in the design of the UML are as follows:

- Offer users a ready-to-use, expressive visual modeling language that enables them to create and exchange meaningful models.
- Provide mechanisms for extending and specializing core concepts to accommodate various needs.
- Remain independent of specific programming languages and development processes.
- Establish a formal foundation for understanding the modeling language.
- Promote the expansion of the object-oriented (OO) tools market.
- Support advanced development concepts such as collaborations, frameworks, patterns, and components.

- Incorporate best practices in modeling.

4.4.1 Types of UML Diagrams:

Structural Diagrams:

Capture the static aspects or structure of a system with Structural Diagrams, which include Component Diagrams, Object Diagrams, Class Diagrams, and Deployment Diagrams.

Behavior Diagrams:

Capture the dynamic aspects or behavior of the system with Behavior Diagrams, which include Use Case Diagrams, State Diagrams, Activity Diagrams, and Interaction Diagrams.

The image below shows the hierarchy of diagrams according to UML

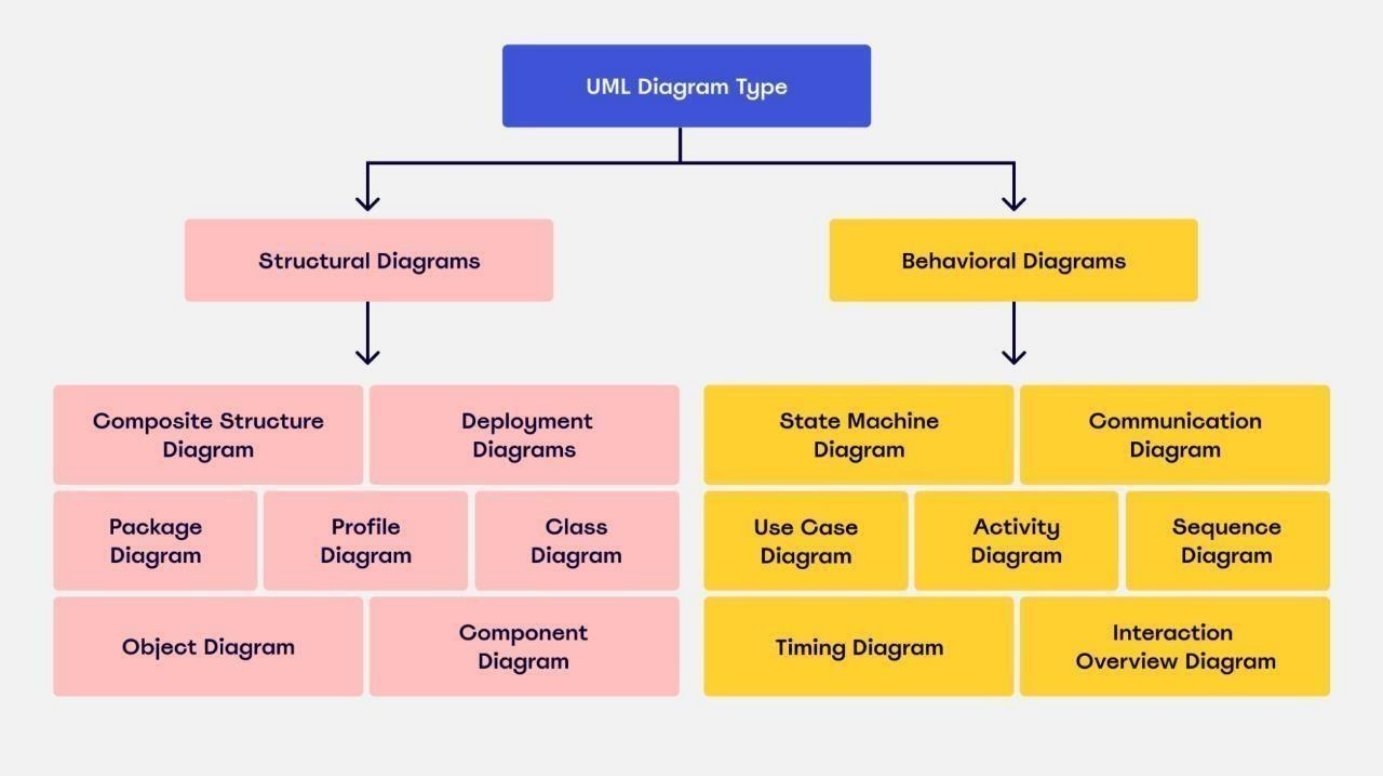

Figure-4.4.2 UML Hierarchy diagrams

4.5 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that outlines the system's structure by depicting its classes, along with their attributes, operations (or methods), and the relationships between them. It illustrates which classes hold specific information.

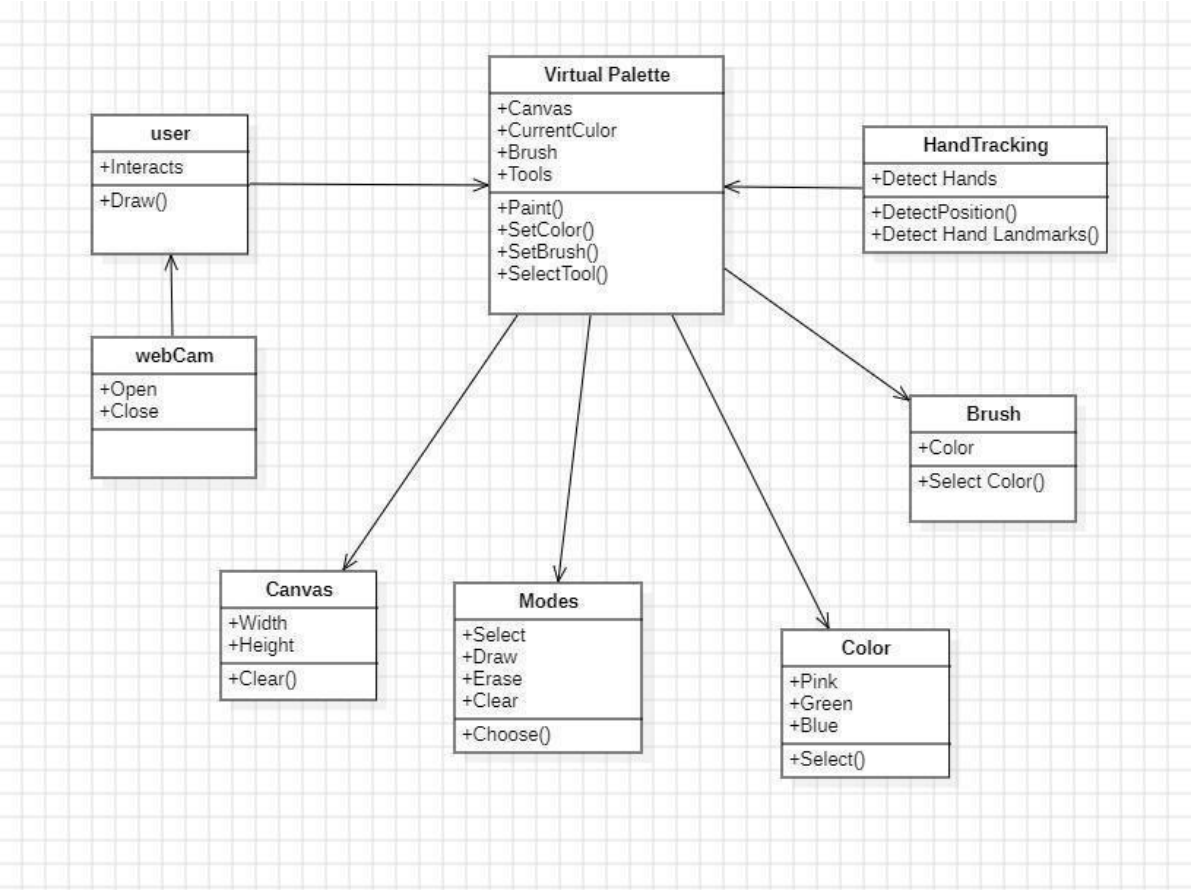

Figure-4.5.1 Class Diagram

4.6 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram derived from use-case analysis. Its purpose is to provide a graphical overview of the system's functionality, illustrating how actors interact with the system through their goals (represented as use cases) and any dependencies between these use cases. The primary goal of a use case diagram is to demonstrate which system functions are performed for each actor and to depict the roles of the actors within the system.

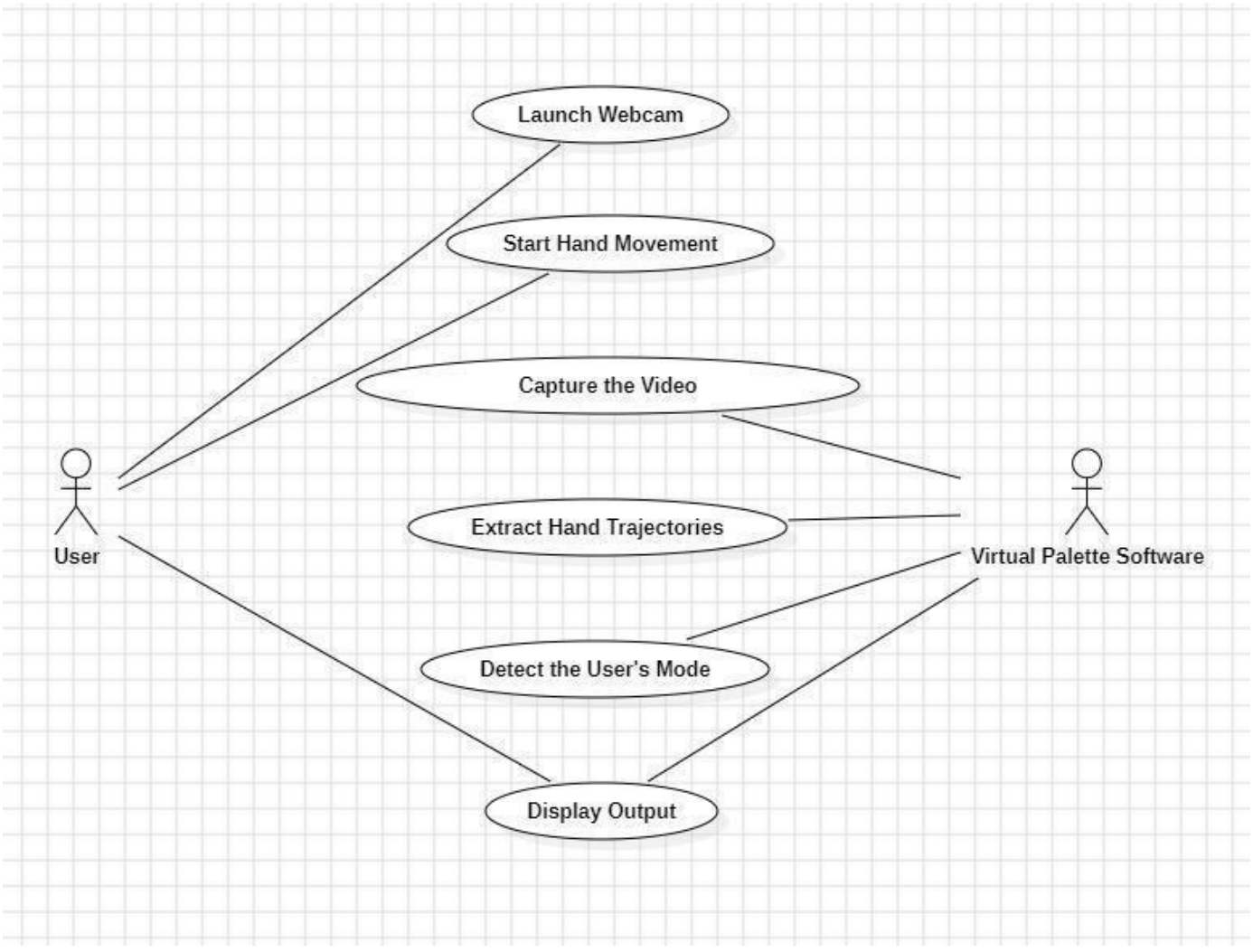

Figure- 4.6.1 Use Case Diagram

4.7 COMPONENT DIAGRAM:

A component diagram, or UML component diagram, illustrates the organization and connections of the physical components within a system. These diagrams are often used to model implementation details and verify that all aspects of the system's required functions are addressed by the planned development.

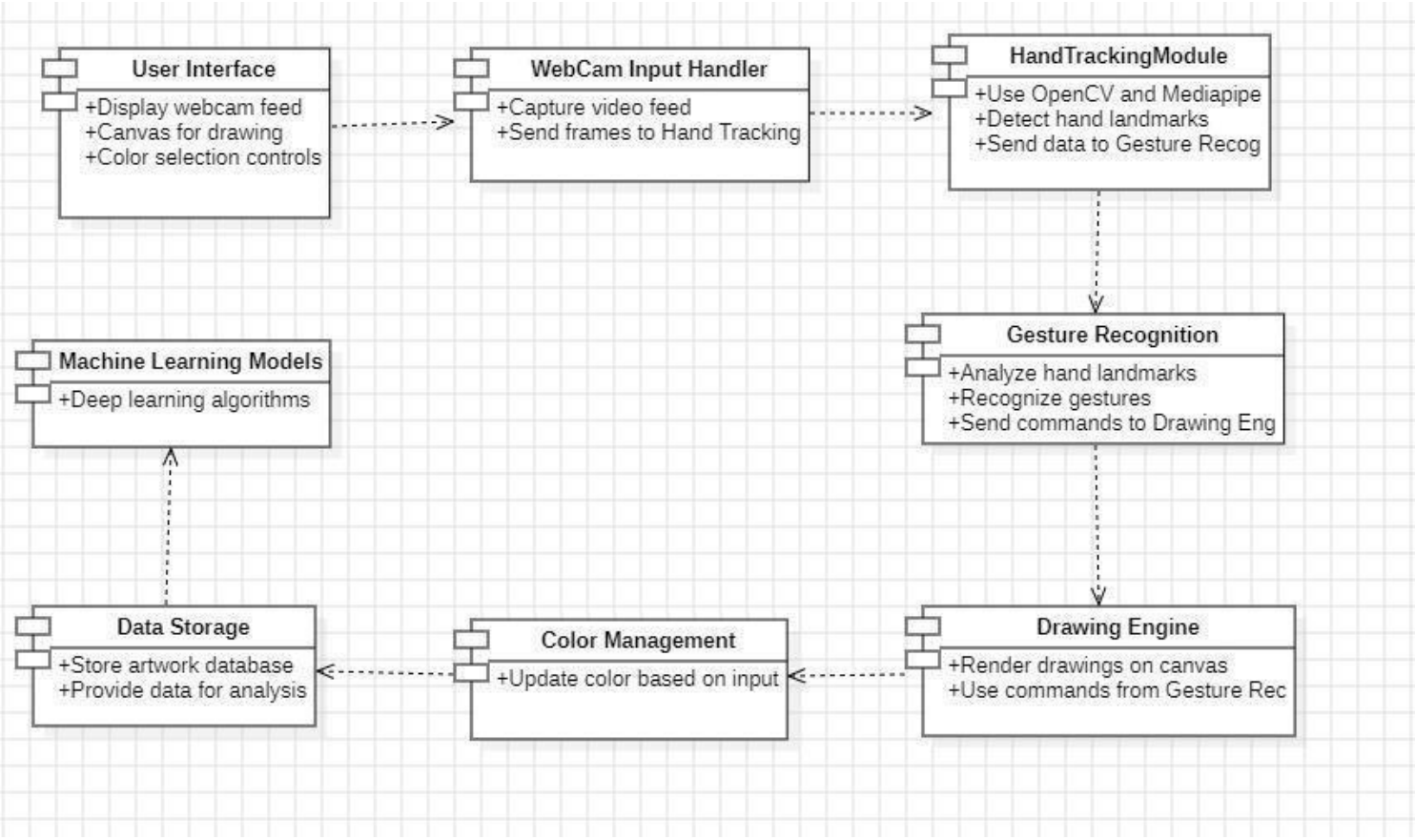

Figure-4.7.1 Component Diagram

4.8 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is an interaction diagram that depicts how processes interact with each other and the order in which these interactions occur. It is a type of Message Sequence Chart and is sometimes referred to as an event diagram, event scenario, or timing diagram.

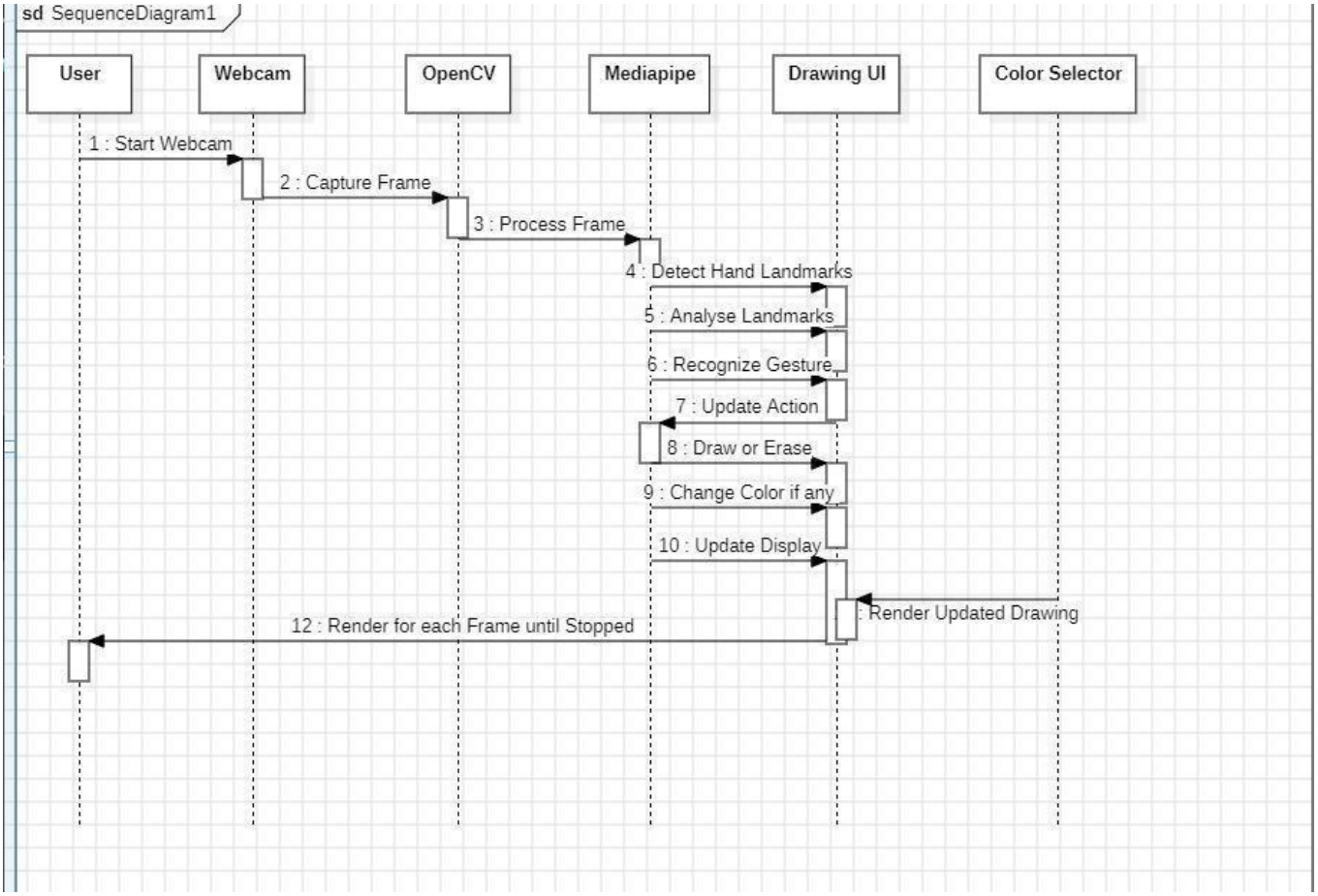

Figure-4.8.1 Sequence Diagram

4.9 ACTIVITY DIAGRAM:

In UML, an activity diagram is used to display the sequence of activities. Activity diagrams show the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity.

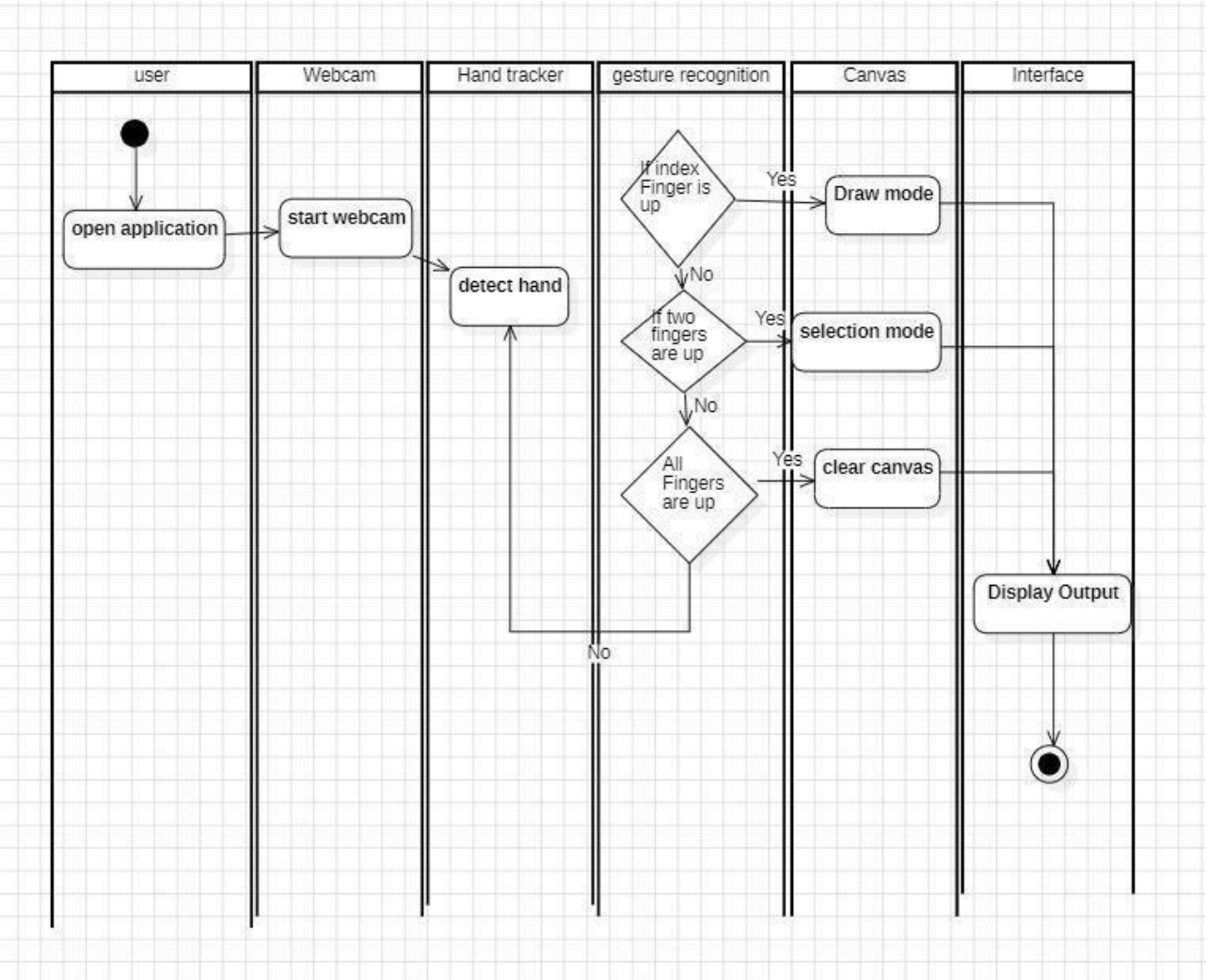

Figure-4.9.1 Activity Diagram

4.10 TECHNOLOGY DESCRIPTION

Here's an overview of the technologies commonly used in an AI-powered virtual painter application:

4.10.1 Computer Vision:

Technology: Computer Vision is an AI field that enables machines to analyze and understand visual information from the world.

Purpose: For a virtual painter application, computer vision tracks hand movements and gestures using the camera. It detects the hand's position, orientation, and movement to translate these into drawing actions on the digital canvas.

Tools: Popular tools for this include OpenCV (OpenSource Computer Vision Library) and MediaPipe.

4.10.2 Performance Optimization:

Technology: Performance optimization involves techniques to ensure the application runssmoothly and efficiently.

Purpose: To maintain high frame rates, reduce latency, and manage system resources effectively.

Tools: Profiling tools, performance monitoring, and optimization strategies specific to the development environment are used to enhance application performance.

5. TESTING AND RESULTS:

5.1 TESTING:

5.1 .1 TESTING DEFINITION:

The primary goal of testing is to identify errors. Testing involves systematically uncovering potential faults or weaknesses in a product. It allows for verifying the functionality of individual components, sub-assemblies, assemblies, and the complete product. Testing exercises the software to ensure it meets its specified requirements and user expectations, and to verify that it does not fail in unacceptable ways. Various testing methods exist, each designed to address specific testing needs.

5.1.2 UNIT TESTING

Unit testing is typically done during the common coding and testing phases of the software development lifecycle. However, coding and unit testing often follow separate stages. The goal of unit testing is to verify the functionality of each component or "unit" of the software to ensure that each piece of code works well independently. These tests are important for early detection and correction of errors, increasing reliability and control. Ensuring that each unit works as expected before integrating with the rest of the system is an important step in the testing process.

- **Test strategy and approach**

The primary objective of testing is to detect errors by systematically identifying potential faults or weaknesses in a product. It involves verifying the functionality of individual components, sub- assemblies, assemblies, and the entire product.

- **Test objectives**

- All field entries must work properly.
- Pages must be activated fromthe identified link.
- The entryscreen, messages and responses must not be delay

5.1.3 INTEGRATION TESTING

Integration testing examines the interactions between different modules to confirm they work together smoothly. Its purpose is to ensure that various components or software applications— whether they are parts of a single system or multiple systems within a company— communicate andfunction correctly without issues.

Test Results: All the test cases mentioned above were successfully passed with no defects encountered.

Acceptance Testing

User Acceptance Testing is a crucial phase of any project that requires substantial involvement from the end users. It ensures that the system meets the functional requirements and expectations..

Test Results: All the test cases mentioned above were successfully passed with no defects encountered.Grey box testing involves access to internal code when designing test cases. This type of testing is carried out by individuals who have expertise in both coding and testing.

Outcomes Possible:

Pass: The test case successfully validates the expected behavior of the application, indicating that specific functionality works as intended.

Fail: The test case fails to validate the expected behavior, indicating a defect or issue in the application. This outcome requires further investigation and fixing the identified problem.

Error: An error occurs during the test execution due to unexpected system behavior or exceptions. This could indicate a bug or potential issue that needs to be addressed.

Blocked: The test case cannot be executed due to external dependencies or environmental constraints. This outcome indicates that the test case is blocked and cannot be validated at the moment.

Skipped: The test case is intentionally skipped from execution, typically due to low priority or specific conditions that prevent it from being applicable at the current stage of testing.

TEST CASES:

Test case Id	**Test Case Description**	**Input**	**Expected Output**
1	Verify Hand detection	User places their hand in front of the webcam.	The system detects and highlights the hand in real-time.
2	Verify Landmark Detection	User performs various hand positions.	The system identifies and displays 21 landmarks on the hand.
3	Verify Drawing Gesture	User raises their index finger.	The system enters drawing mode, allowing drawing on the canvas.
4	Verify Selection Gesture	User raises their index and middle fingers.	The system switches to selection mode.
5	Verify Color Change	User selects different colors.	The strokes reflect the selected color.

Table-5.1.4 Test Cases

`

Performance Test Cases:

Test case Id	Test Case Description	Input	Expected Output
1	Verify Smooth Drawing Performance	User draws continuously at a fast pace for 5 minutes.	The application maintains a consistent frame rate (>30 FPS) with minimal latency.
2	Verify Performance with High-Resolution Canvas	User creates a drawing on a 4K canvas (3840x2160) for 3 minutes.	The application maintains a consistent frame rate (>20 FPS) with minimal performance degradation.

Table-5.1.5 Performance Testing

5.2 RESULTS

- Once the program is executed, the following results are observable:

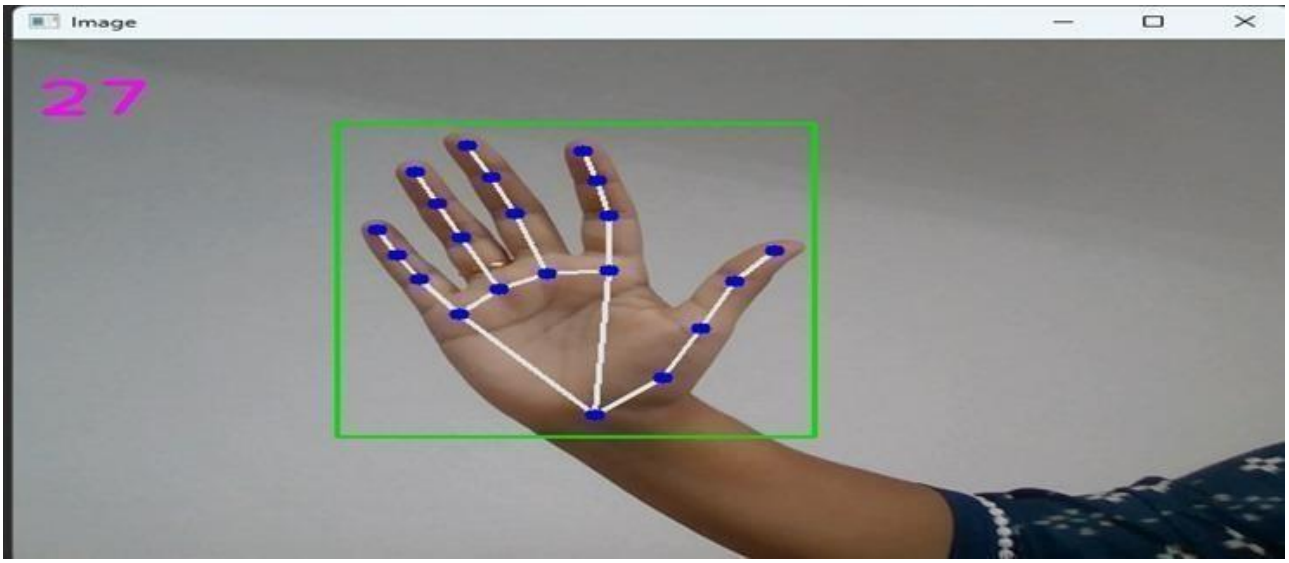

Figure-5.2.1 Hand Tracking

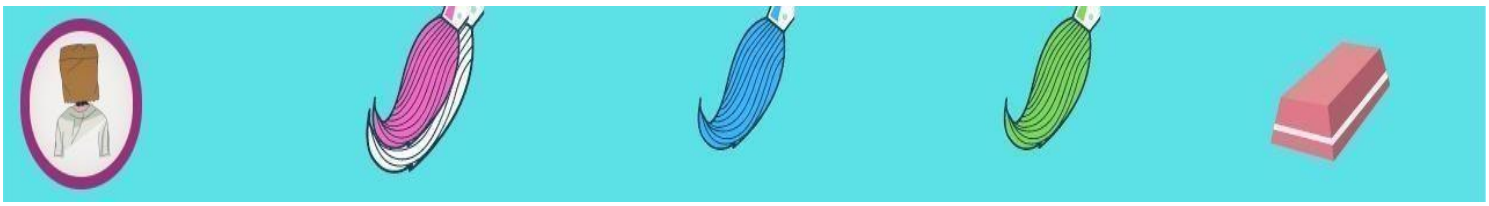

Figure 5.2.2 Header

`

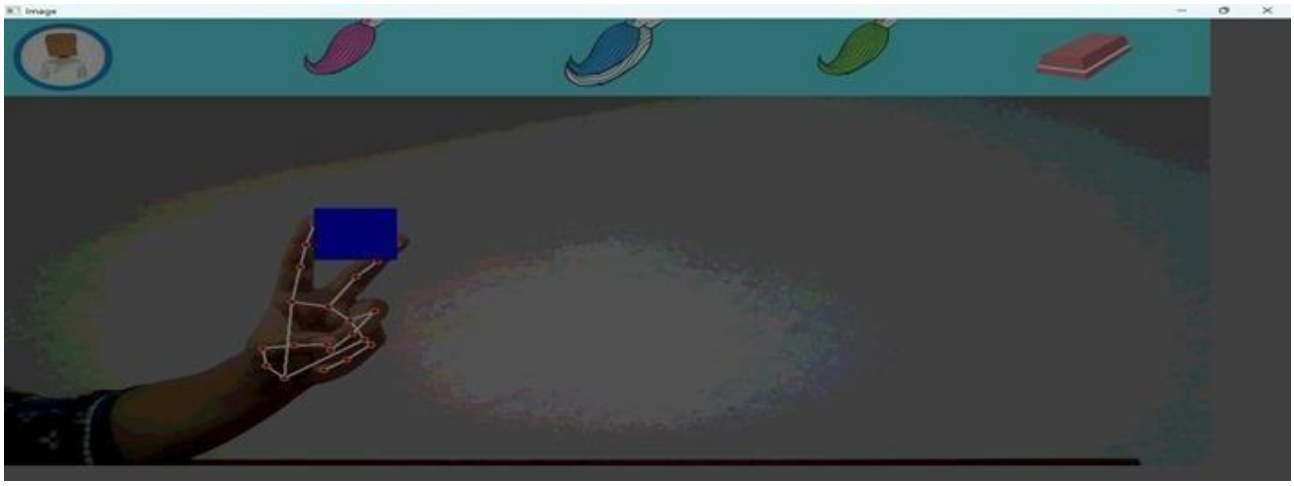

Figure 5.2.3 Selection Mode

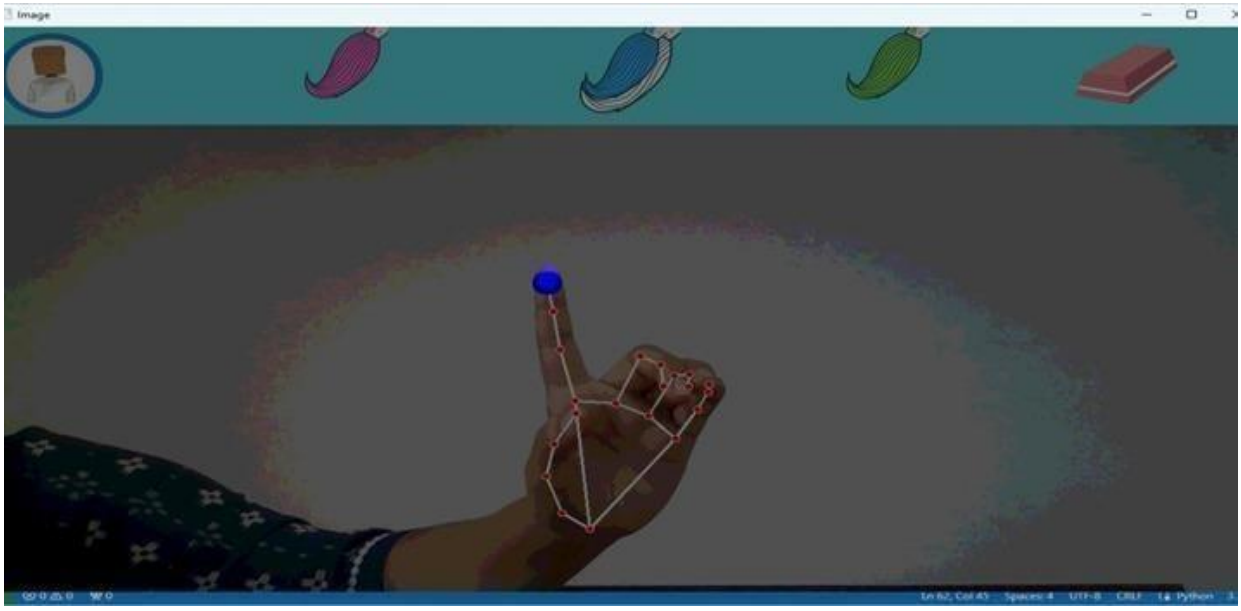

Figure-5.2.4 Drawing Mode

`

Figure- 5.2.5 Canvas Cleared

Figure 5.2.6 Realtime Rendering

6. CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In summary, the "Virtual Palette using AI" project combines AI technology with artistic expression, providing users with powerful tools for creating and personalizing digital art. This concept allows the individual to experience a unique and reciprocal world where they one might be able to sketch. This framework has the potential to disrupt traditional writing and teaching methods. Its key advantage lies in the control it offers users, particularly in fields requiring clear definitions, such as virtual art It is versatile, requiring minimal writing skills, and allows users to quickly convey their ideas. Users can easily draw and make selections with simple gestures, such as using one finger for drawing and two fingers for selection.

6.2 FUTURE WORK

The "Virtual Palette using AI" project has several exciting future prospects. Expanding the variety of artistic styles to include a broader range of cultural and historical techniques could offer users more creative options. Adding real-time collaboration features would allow multiple users to co-create art, enhancing the collective creative experience. Future enhancements might include additional features to assist professionals, consolidating tools for drawing, editing, and sharing into a single platform. Advanced customization could enable users to design and fine-tune their own artistic styles with AI- generated suggestions. Incorporating Augmented Reality (AR) would allow users to see their art in physical spaces, adding a new dimension to their creations. Enhancing AI models to produce more realistic and diverse artwork is crucial, alongside developing personalized art generation based on individual preferences. Expanding the system to support mobile devices and other platforms would improve accessibility, while adding educational features could offer valuable insights into art styles and AI technology.

7. BIBLIOGRAPHY:

- Xiaoyan Wang, Yi Sun, "Artificial intelligence application of virtual reality technology in digital media art creation", 2021 2nd International Conference on Information Science and Education (ICISE-IE).

- Akash Kumar Choudhary, Bharat Phogat, Saurabh uday Saoji, Bharati Vidyapeeth, " Basic Paint Window Application via Webcam Using OpenCV and Numpy in Python", August 2021 Journal of Interdisiplinary Cycle Research XIII(VII):1680-1686

- Koushik Roy, Md. Akiful Hoque Akif," Real Time Hand Gesture Based User Friendly Human Computer Interaction System", February 2022 International Conference on Innovations in Science, Engineering and Technology (ICISET).

- Shaurya Gulati, Ashish Kumar Rastogi, Mayank Virmani, Rahul Jana, "Paint / Writing Application through WebCam using MediaPipe and OpenCV", February , 2022 2nd International Conference on InnovativePracticesinTechnologyand Management(ICIPTM)

- Vaishnav Chunduru, Mrinalkanti Roy, Dasari Romit N. S, Rajeevlochana G. Chittawadigi," Hand Tracking in 3D Space using MediaPipe and PnP Method for Intuitive Control of Virtual Globe", September- October 2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC).

- Laura Raya, José Jesús García-Rueda ,Daniel López-Fernánde , Jesús Mayor," Virtual Reality Application for Fostering Interest in Art", IEEE Computer Graphics and Applications (Volume: 41, Issue: 2, March-April 2021).

- Beibei Shu, Bjørn Solvang," Architecture for task-dependent human-robot collaboration", January, 2021 IEEE/SICE International Symposium on System Integration (SII).

8.APPENDIX:

VirtualPalette.py

```
import cv2
import numpy as
np import os
import HandTrackingModule as htm
brushThickness = 15
eraserThickness = 100
folderPath = "Header"
myList = os.listdir(folderPath)
overLayList = [cv2.imread(f'{folderPath}/{imPath}') for imPath in
myList] header = overLayList[0]

drawColor = (225, 0, 225)

cap = cv2.
VideoCapture(0) cap.set(3,
1280)
cap.set(4, 720)

detector = htm.handDetector(detectionCon=0.85)
xp, yp = 0, 0

imgCanvas = np.zeros((720, 1280, 3),

np.uint8) while True:
    success, img = cap.read()
    img = cv2.flip(img, 1)

    # 2. Find Hand Landmarks
    img =
    detector.findHands(img)
    lmList, bbox = detector.findPosition(img, draw=False)
```

```
if len(lmList) != 0:
    x1, y1 = lmList[8][1:]
    x2, y2 = lmList[12][1:]

    fingers = detector.fingersUp()

    # Clear canvas if all fingers are
    up if all(fingers):
        imgCanvas.fill(0) # Efficient way to clear the canvas
        cv2.putText(img, "Canvas Cleared", (200, 200),
                    cv2. FONT_HERSHEY_SIMPLEX, 3, (0, 0, 255), 5)
    # Drawing mode: Index finger is
    up elif fingers[1] and not
    fingers[2]:
        cv2.circle(img, (x1, y1), 15, drawColor, cv2.
        FILLED) if xp == 0 and yp == 0:
            xp, yp = x1, y1

        if drawColor == (0, 0, 0):
            cv2.line(img, (xp, yp), (x1, y1), drawColor,
            eraserThickness) cv2.line(imgCanvas, (xp, yp), (x1, y1),
                    drawColor, eraserThickness)
        else:
            cv2.line(img, (xp, yp), (x1, y1), drawColor,
            brushThickness) cv2.line(imgCanvas, (xp, yp), (x1, y1),
                    drawColor, brushThickness)

        xp, yp = x1, y1
    # Selection mode: Two fingers are
    up elif fingers[1] and fingers[2]:
        xp, yp = 0, 0
        if y1 < 125:
            if 250 < x1 < 450:
                header = overLayList[0]
                drawColor = (225, 0,
                225)
            elif 550 < x1 < 750:
                header =
                overLayList[1]
```

```
            drawColor = (225, 0, 0)
        elif 800 < x1 < 950:
            header =
            overLayList[2]
            drawColor = (225, 225, 0)
        elif 1050 < x1 < 1200:
            header =
            overLayList[3]
            drawColor = (0, 0, 0)
    cv2.rectangle(img, (x1, y1 - 25), (x2, y2 +
                25), drawColor, cv2. FILLED)



imgGray = cv2.cvtColor(imgCanvas, cv2. COLOR_BGR2GRAY)
_, imgInv = cv2.threshold(imgGray, 50, 225, cv2.
THRESH_BINARY_INV) imgInv = cv2.cvtColor(imgInv, cv2.
COLOR_GRAY2BGR)
img = cv2.bitwise_and(img, imgInv)
img = cv2.bitwise_or(img, imgCanvas)

img[0:125, 0:1280] = header
img = cv2.addWeighted(img, 0.5, imgCanvas, 0.5, 0)

cv2.imshow("Image", img)
cv2.waitKey(1)
```

HandTrackingModule.py

```
import cv2
#cv2: This is OpenCV, a library used for computer vision tasks like image processing and
video capture.

import mediapipe as mp
#mediapipe: A library by Google for real-time hand and face detection using machine
learning.
import time
#time: Provides time-related functions, used here to measure the frame
rate. import math
#math: Provides mathematical functions; here, it's used for calculating distances.


class handDetector:
    def _init_(self, mode=False, maxHands=2, detectionCon=0.5,
        trackCon=0.5): self.mode = mode
        self.maxHands = maxHands
        self.detectionCon =
        float(detectionCon) self.trackCon =
        float(trackCon)

        self.mpHands = mp.solutions.hands
         self.hands = self.mpHands.Hands(
            static_image_mode=self.mode,
            max_num_hands=self.maxHands,
            min_detection_confidence=self.detectionCon,
            min_tracking_confidence=self.trackCon)

        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2. COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        if self.results.multi_hand_landmarks:
```

```
        for handLms in self.results.multi_hand_landmarks:
            if draw:
                self.mpDraw.draw_landmarks(img, handLms,
                                           self.mpHands.HAND_CONNECTIONS)
    return img

def findPosition(self, img, handNo=0,
    draw=True): xList = []
    yList = []
    bbox = []
    self.lmList = []
    if self.results.multi_hand_landmarks:
        myHand = self.results.multi_hand_landmarks[handNo]
        for id, lm in enumerate(myHand.landmark):
            h, w, c = img.shape
            cx, cy = int(lm.x * w), int(lm.y *
            h) xList.append(cx)
            yList.append(cy)
            self.lmList.append([id, cx, cy])
            if draw:
                cv2.circle(img, (cx, cy), 5, (255, 0, 0), cv2.
        FILLED) xmin, xmax = min(xList), max(xList)
        ymin, ymax = min(yList),
        max(yList) bbox = xmin, ymin,
        xmax, ymax
        if draw:
            cv2.rectangle(img, (bbox[0] - 20, bbox[1] - 20),
                          (bbox[2] + 20, bbox[3] + 20), (0, 225, 0), 2)
    return self.lmList, bbox

def fingersUp(self):
    fingers = []
    if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] -
        1][1]: fingers.append(1)
    else:
        fingers.append(0)
    for id in range(1, 5):
        if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
```

```
            fingers.append(1)
        else:
            fingers.append(0)
    return fingers

def findDistance(self, p1, p2, img, draw=True):
    x1, y1 = self.lmList[p1][1], self.lmList[p1][2]
    x2, y2 = self.lmList[p2][1], self.lmList[p2][2]
    cx, cy = (x1 + x2) // 2, (y1 + y2) // 2
    if draw:
        cv2.circle(img, (x1, y1), 15, (225, 0, 225), cv2. FILLED)
        cv2.circle(img, (x2, y2), 15, (225, 0, 225), cv2. FILLED)
        cv2.line(img, (x1, y1), (x2, y2), (225, 0, 225), 3)
        cv2.circle(img, (cx, cy), 15, (225, 0, 225), cv2.
    FILLED) length = math.hypot(x2 - x1, y2 - y1)
    return length, img, [x1, y1, x2, y2, cx, cy]


def main():
    pTime = 0
    cTime = 0
    cap = cv2.
    VideoCapture(0) detector
    = handDetector() while
    True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])
        cTime = time.time()
        fps = 1 / (cTime -
        pTime) pTime = cTime
        cv2.putText(img, str(int(fps)), (10, 70), cv2. FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)
        cv2.imshow("Image", img)
        cv2.waitKey(1)

if _name_ == "_main_":
    main()
```