*A SOCIETAL RELATED (or) REAL TIME Project Report on*

## PRESENTATION HANDLER USING HAND GESTURE

*Submitted in partial fulfillment of the requirements for the award of the degree of*

## Bachelor of Technology(B.Tech -IInd year/IIsemester)

In

## Computer Science and Engineering

By

**B.Vijayendar Reddy** (23AG1A05D7) **E.Pranitha** (23AG1A05E4) **MD.Irshadabbas** (23AG1A05H0)

Under the Esteemed guidance of

B.R Srinivasa Rao

(Assistant Professor)



# Department of Computer Science and Engineering

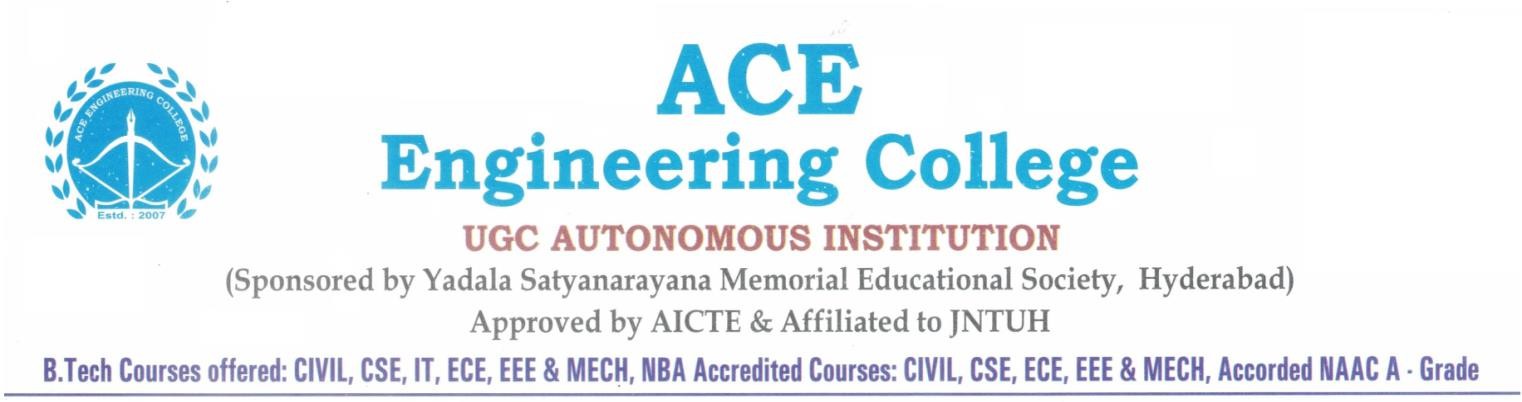
**ACE Engineering College**

**Ankushapur(V), Ghatkesar(M), Medchal Dist - 501 301**

***(An Autonomous Institution, Affiliated to JNTUH, Hyderabad)***

[www.aceec.ac.in](http://www.aceec.ac.in/)

**2024-2025**



# Department of Computer Science and Engineering

**CERTIFICATE**

This is to certify that the Societal Related (Or) Real Time project report entitled “**PRESENTATION HANDLER USING HAND GESTURE**” is a bonafide work done by B.Vijayendar Reddy,E.Pranitha ,MD.Irshadabbas bearing 23AG1A05D7,23AG1A05E4,23AG1A05H0 in partial fulfillment for the award of Degree of BACHELOR OF TECHNOLOGY in Computer Science and Enginnering from JNTUH University, Hyderabad during the academic year 2024- 2025. This record of bonafide work carried out by them under our guidance and supervision.

**Subject Coordinator Head of the Department**

**B.R.srinivasa Rao Dr. M.V.Vijaya Saradhi**

Assistant Professor Professor and Head Dept.

of CSE

**Project Coordinator**

**ACKNOWLEDGEMENT**

We would like to express my gratitude to all the people behind the screen who have helped me transform an idea into a real time application.

We would like to express my heart-felt gratitude to my parents without whom we would not have been privileged to achieve and fulfill my dreams.

A special thanks to our General Secretary, **Prof. Y. V. Gopala Krishna Murthy,** for having founded such an esteemed institution. Sincere thanks to our Joint Secretary **Mrs.**

**M. Padmavathi**, for support in doing project work. I am also grateful to our beloved principal, **Dr.K.S.RAO** for permitting us to carry out this project.

We profoundly thank **Dr.M.V.Vijaya saradhi**, Associate Professor and Head of the Department of Computer Science and Engineering , who has been an excellent guide and also a great source of inspiration to my work.

We extremely thank **Mr.P.Santhosh ,** Assistant Professors, Project coordinators, who helped us in all the way in fulfilling of all aspects in completion of our Mini-Project.

We are very thankful to my internal guide **Mr.B R Srinivasa rao** who has been an excellent and also given continuous support for the Completion of my project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, I would like to thank all the other staff members, both teaching and non- teaching, who have extended their timely help and eased my task.

### B.VijayendarReddy23AG1A05D7

### E.Pranitha-23AG1A05E4

### MD.Irshadabbas -23AG1A05H0

**PRESENTATION HANDLER USING HAND GESTURE**

# ABSTRACT

Looks into how hand motions can be used to improve presentation handling through computer vision (CV) techniques.Draws attention to the necessity of digital presentations having simple, hands-free control techniques.Creates and applies reliable CV algorithms for the recognition and interpretation of gestures in real time.Assesses how well users interact with the system, paying particular attention to user happiness, accuracy of gesture detection, and overall presentation control efficiency.Seek to provide insights into CV-based strategies for engaging,dynamic presenting environments.

In the contemporary digital age, seamless interaction between humans and technology is paramount. Gesture control presents an intuitive and efficient means to bridge this interaction gap. This abstract introduces a project focused on developing a Presentation Handler using Hand Gestures (PHHG).

The PHHG system leverages computer vision techniques and machine learning algorithms to interpret hand gestures and translate them into commands for controlling presentations. By utilizing a camera to capture hand movements in real-time, the system processes these gestures to navigate through slides, manipulate content, and interact with presentation elements.

Key components of the PHHG system include hand gesture recognition, feature extraction, and gesture-to-command mapping. Deep learning models, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are employed for robust gesture recognition and classification. Feature extraction techniques extract relevant information from the hand gesture data stream, facilitating accurate interpretation.

Moreover, the PHHG system offers adaptability and customization, allowing users to define and configure gesture-based commands according to their preferences. This flexibility enhances user experience by accommodating individual interaction styles and preferences.

The benefits of the PHHG system extend beyond convenience to inclusivity and accessibility. By eliminating the need for traditional input devices like keyboards or

remote controls, it empowers users with disabilities or mobility impairments to engage effectively in presentations. Additionally, the intuitive nature of gesture control enhances audience engagement and interaction during presentations, fostering a dynamic and immersive experience.

In conclusion, the Presentation Handler using Hand Gestures (PHHG) project represents a significant advancement in human-computer interaction, offering a novel approach to presentation control that is intuitive, adaptable, and inclusive. With its potential to revolutionize the way presentations are conducted, PHHG sets a new standard for seamless interaction between users and technology.

# CONTENT

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CHAPTER NAME** | **PAGE NO** |
| **1** | **INTRODUCTION** | **9** |
| **2** | **EXISTING SYSTEM** | **11** |
| **2.1** | **EXISTING SYSTEM DRAWBACKS** | **12** |
| **3** | **LITERATURE SURVEY** | **14** |
| **4** | **PROPOSED MODEL / SYSTEM** | **15** |
| **5** | **REQUIREMENTS GATHERING** | **17** |
| **5.1** | **Software Requirements &** | **17** |
| **5.2** | **Hardware Requirements**  **Functional Requirements &** | **18** |
| **5.3** | **Non-Functional Requirements**  **Data Collection** | **19** |
| **6** | **SYSTEM ANALYSIS & DESIGN** | **20** |
| **6.1** | **Module Description** | **21** |
| **6.2** | **System Architecture** | **22** |
| **6.3** | **UML Diagrams** | **23** |
| **7** | **IMPLEMENTATION** | **27** |
| **7.1** | **Algorithm Used** | **27** |
| **7.2** | **Sample Code** | **29** |
| **8** | **TESTING** | **33** |
| **9** | **RESULTS** | **34** |
| **10** | **CONCLUSION** | **38** |
|  | **FUTURE WORK** | **38** |
|  | **REFERENCES** | **39** |

**LIST OF ABBREVIATIONS**

**Abbreviation Full Form**

**HCI Human-ComputeInteraction.**

**CV Computer Vision**

**ML Machine Learning.**

**LSTM Long Short-Term Memory.**

## INTRODUCTION

Our hands are more flexible when it comes to the usage of our body and when it comes to movement, so they are involved greatly in our day-to-day activities. It is a natural occurrence that someone uses their hands immediately when they wake up or want to get something. Researchers have been going over ways of interacting with computers and other devices and the aim is trying to increase human-computer interaction (HCI) through a communication channel from the user to the device.

One of the ways we can communicate with a computer is with our “hands” making movements which are said to be hand gestures. The use of hand gestures as a means of interacting with technology has become increasingly popular and allows users to interact more naturally and intuitively and it has started entering the field of presentation . Hand gestures have been used as a form of major communication in presentations for ages. Hand gestures have been effectively used in presentations by renowned presenters and public figures across past times, including Martin Luther King Jr., Winston Churchill, and Steve Jobs, to captivate audiences and convey compelling messages.

This historical setting highlights the importance of hand gestures as an important part of effective communication. Hand gestures in the field of presentation enhance communication performance, especially for presenters who may experience stage fright. While slides are often seen as the focal point, effective hand gestures can captivate the audience and draw their attention into the presentation. It then aids in emphasizing essential points, drawing focus on significant details, and illustrating ideas or operations. Presenters can improve listener understanding and recollection of information by adopting gestures that correspond to the topic. Non-verbal indicators, such as hand gestures, are an important part of communicating. These aid in the general comprehension and understanding of the information. Presentations get better when presenters enhance their verbal communication, express emotions, and include richness in their presentations by integrating intentional and well-timed gestures.

In recent years, many eyes have been on various technologies to offer improved user interfaces that enable computer interactions as intuitive as human interactions. Furthermore, standard gadgets such as a keyboard and mouse cannot entirely fulfill human interaction needs in presentations. The study on controlling presentations using hand gestures is significant as it aims to enhance user experience, increase interactivity, improve accessibility, enable seamless integration, and drive innovation in presentation delivery. By developing a presentation package that allows presenters to control their presentations using hand gestures, the study seeks to provide a more intuitive and natural interaction method. This approach eliminates the need for physical devices or complex keyboard shortcuts, making the presentation process seamless and enjoyable. Incorporating hand gestures in presentations enhances interactivity, captures the audience's attention, and promotes engagement. The study also emphasizes the importance of seamless integration with popular presentation software and contributes to the advancement of interaction techniques and innovative presentation delivery methods .

## EXISTING SYSTEM

While the specific concept of an "HAND GESTURE" might not be widely commercialized, several projects, prototypes, and technologies have been developed that align closely with this idea.These implementations typically leverage gesture recognition, computer vision, and real-time interaction technologies.

#### Gesture-Based Presentation Controller using Computer Vision:

This system captures hand gestures using a camera and processes the visual information using computer vision models. By recognizing specific gestures, it interprets actions to move forward or backward through presentation slides. Presenters can navigate freely without being tethered to a keyboard or mouse.

#### Hand-Gesture-Controlled-Presentation (GitHub):

An interactive presentation control system that combines hand gestures and speechrecognition. It uses OpenCV and the cvzone library to enable slide navigation, annotation drawing, and zooming via hand gestures

#### Leap Motion Controller:

The Leap Motion Controller uses infrared cameras and LEDs to track hand movements, interpreting them as mouse movements. It allows users to control presentations without physical devices like a mouse or keyboard.

#### Myo Armband:

The Myo armband detects hand-arm movements using electro myograp sensors them into,translating mouse movements

## 2.1.EXISTING SYSTEM DRAWBACKS

**Accuracy and Precision:** Ensuring high accuracy in hand tracking and gesture recognition remains a significant challenge, particularly in varied lighting conditions and complex backgrounds.

**Latency:** Minimizing latency between hand movements and their representation on the digital canvas is crucial for providing a seamless user experience.

**Robustness:** Maintaining consistent performance across different users and environments is challenging due to variations in hand size, skin color, and background noise.

**Detection Range and Accuracy**:Existing systems often rely on cameras or sensors to detect hand gestures. The detection range can be limited, affecting the system’s responsiveness.

Accuracy is crucial for seamless control. If the system misinterprets gestures, it can disrupt the presentation flow12.

**Stability and Amplification:**Some systems amplify recognized gestures even when the presenter doesn’t intend to perform them. Ensuring stability is essential to prevent unintended actions2.

Stability issues can lead to frustrating experiences for presenters and distract the audience.

**Cost and Complexity:**Traditional gesture recognition systems may require specialized hardware, such as gloves or markers, which can increase costs.

AI-based systems offer a more cost-effective approach, but they still need robust training and testing3.

**Real-Time Performance**:For real-time performance, especially in large presentations, a reasonably high-end GPU is necessary.

Balancing performance and accuracy is crucial to maintain a smooth experience4.

**User Learning Curve**:Presenters need to learn and remember specific gestures for commands (e.g., next slide, previous slide).

Intuitive gestures are essential to minimize the learning curve.

## LITERATURE SURVEY

Controlling PowerPoint Slide Presentations Through Hand Gestures: This paper discusses vision-based HCI methods that utilize hand gestures for controlling presentations. It explores how users can complete activities without physically accessing typical input devices (mouse, keyboard, etc.) by using hand gestures2.

Gesture-Driven Automation for Dynamic Presentations: This research paper investigates the feasibility and effectiveness of utilizing hand gestures as a natural and intuitive interface for controlling digital presentations. It explores the development and implementation of an automated system for presentation control using hand gesture techniques3.

Virtual Control System for Presentations by Real-Time Hand Gesture Recognition: This paper proposes a virtual control system for presentations based on real-time hand gesture recognition.The system captures hand gestures using a webcam or built in camera,allowing presenters to control their slides naturally.

## PROPOSED MODEL/SYSTEM

#### Gesture Detection:

Use cameras or depth sensors to capture hand movements.

Employ computer vision techniques (such as convolutional neural networks) to recognize specific gestures (e.g., swipe left, swipe right, point, or pinch).

#### Gesture Mapping:

Map recognized gestures to presentation commands (e.g., next slide, previous slide, start/stop video).

Create an intuitive mapping to minimize the learning curve for presenters.

#### Real-Time Processing:

Process gesture data in real time.

Optimize for low latency to ensure responsiveness during presentations.

#### Presentation Software Integration:

Integrate with popular presentation software (e.g., PowerPoint, Google Slides).

Send commands (e.g., keyboard shortcuts) based on recognized gestures.

#### User Feedback and Calibration:

Provide visual feedback to presenters (e.g., highlighting recognized gestures).

Allow calibration for different hand sizes and user preferences.

#### Error Handling:

Handle false positives/negatives gracefully (e.g., accidental gestures).

Implement fallback mechanisms (e.g., backup keyboard controls).

#### Security and Privacy:

Ensure data privacy (no recording or storing of hand images).

Prevent unauthorized access to presentation controls.

## REQUIREMENTS GATHERING

### Software Requirements & Hardware Requirements:

#### SOFTWARE REQUIREMENTS:

Operating system : WINDOWS 8 Ultimate Language used : Python

IDE used 2023 : Pycharm

#### HARDWARE REQUIREMENTS:

System : Intel I3 2.2 GHZ RAM : 4GB

### Functional Requirements & Non-Functional Requirements:

**Functional Requirements:**

* + 1. User Interface Requirements

Main Slide: A digital area where the user’s can highlight.

* + 1. Gesture Recognition

Hand Detection: The system should accurately detect the user's hand. Gesture Identification: Distinguish between different gestures.

* + 1. Drawing Mechanics

Smooth Lines: Ensure lines are smooth and continuous.

* + 1. Security and Privacy

Data Privacy: Ensure user data (especially video feed) is processed

securely .

Permission Management: Clear user permissions for accessing camera.

### Non-Functional Requirements:

* Performance: Ensure quick response to user gestures and smooth Handling experience.
* Reliability: Maintain high availability and recover gracefully from errors.
* Usability: Design an intuitive interface accessible to all users.
* Compatibility: Support various platforms and devices, ensuring seamless operation.

### Data Collection:

1. Install Required Libraries:

Make sure you have the necessary libraries installed

1. Set Up Webcam Input:

Use OpenCV to capture video from your webcam.

1. Hand Detection:

Use color detection, contour detection, or a machine learning model to detect the hand. For simplicity, let's use color detection.

1. Track Hand Movement:

Track the detected hand movement to simulate drawing on a canvas.

1. Draw on the Canvas:

Translate the tracked movements to draw on a virtual canvas.

1. **SYSTEM ANALYSIS & DESIGN**

#### Gesture-Based Presentation Controller using Computer Vision:

Traditional methods like using a mouse, keyboard, or remote clicker can restrict a presenter’s movement during a presentation.

A gesture-based system allows users to navigate presentation slides using simple hand gestures.

The system captures hand gestures using a camera and processes the visual information with a computer vision model.

By recognizing specific gestures, the system can control presentation slides (e.g., moving forward or backward).

This approach frees presenters from being tethered to a keyboard or clicker.

#### Research Papers on Gesture-Driven Presentation Control:

Researchers have explored using hand gestures as a natural and intuitive interface for controlling presentations.

These systems utilize webcams or built-in cameras to capture hand gestures, providing a more dynamic and engaging presentation experience.

* 1. **MODULE DESCRIPTION**

The system captures the user’s hand gestures using a camera.A computer vision model processes this visual information and recognizes specific hand gestures.Based on the detected gesture, the system executes corresponding commands to control the presentation slides (e.g., moving to the next or previous slide).Presenters can move freely without being tethered to a keyboard, mouse, or presentation clicker.

#### To build such a system:

Collect and label a dataset of hand gesture images (e.g., for “next” and “previous” commands).

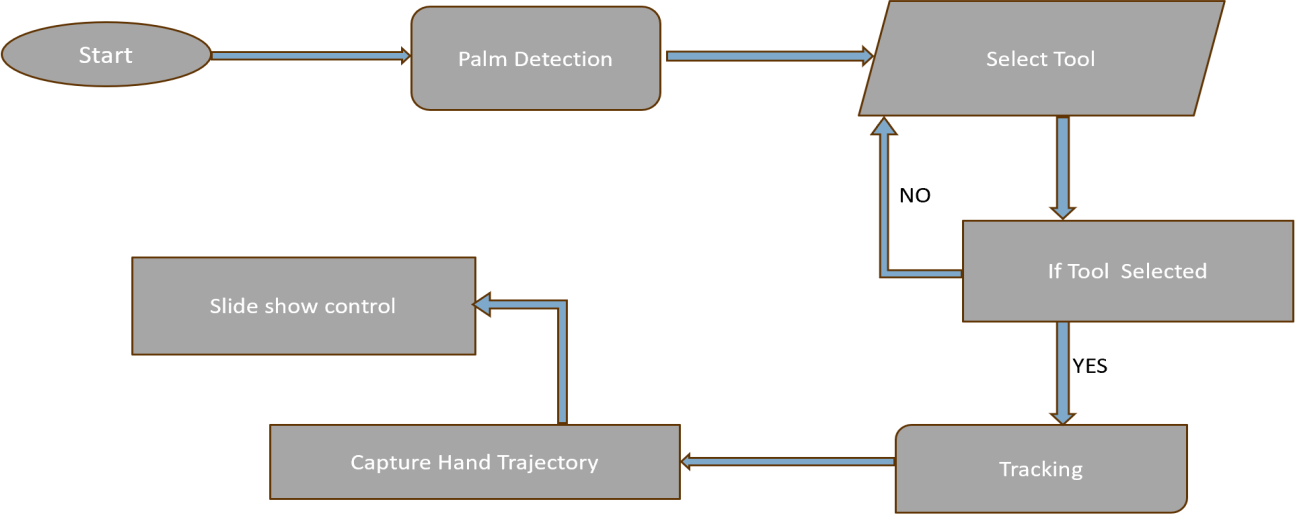
Train a model using computer vision techniques to recognize these gestures. Deploy the model to interpret real-time hand gestures during presentations.

#### Examples of open-source projects related to this topic:

GestureFlow: Allows control of presentation slides through recognized hand gestures captured via a webcam.

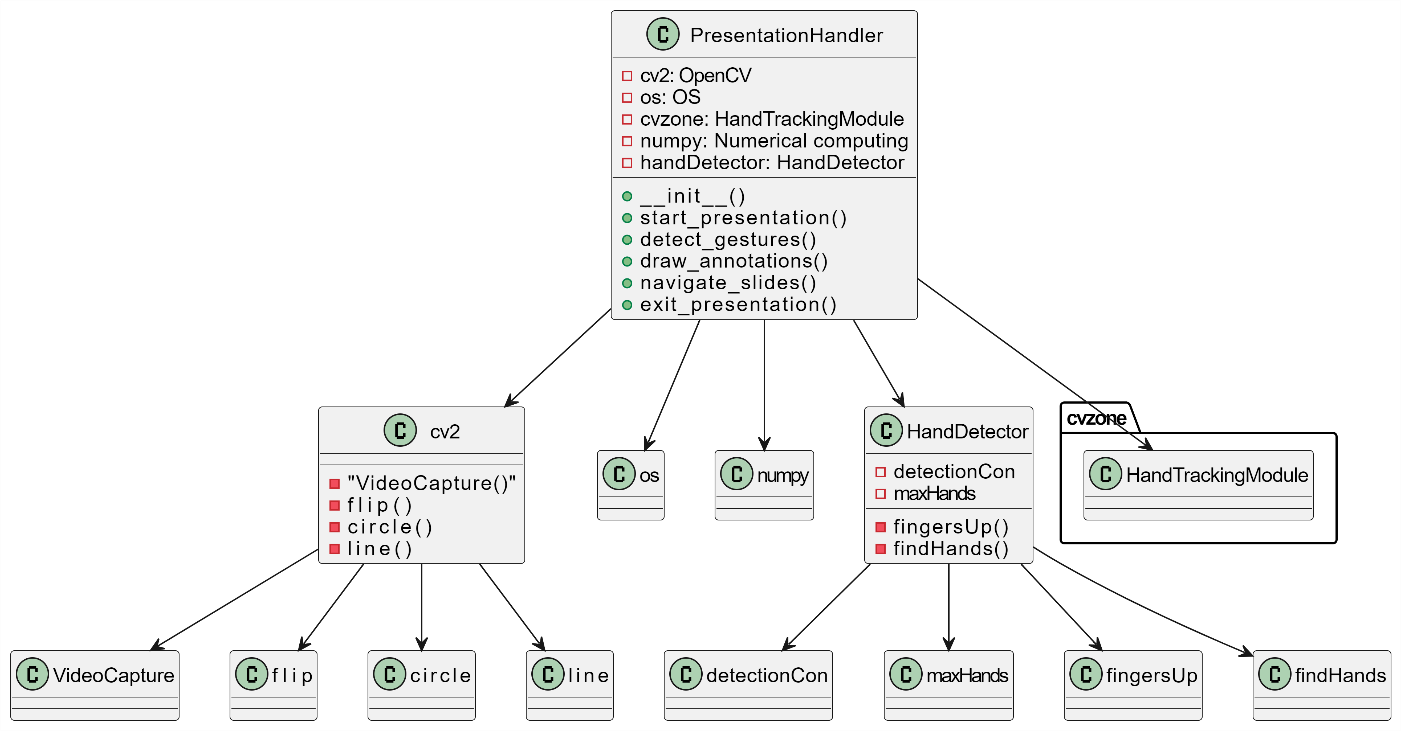
PPT-Control-Using-Hand-gesture: Controls PowerPoint presentations using hand gestures.

### System Architecture

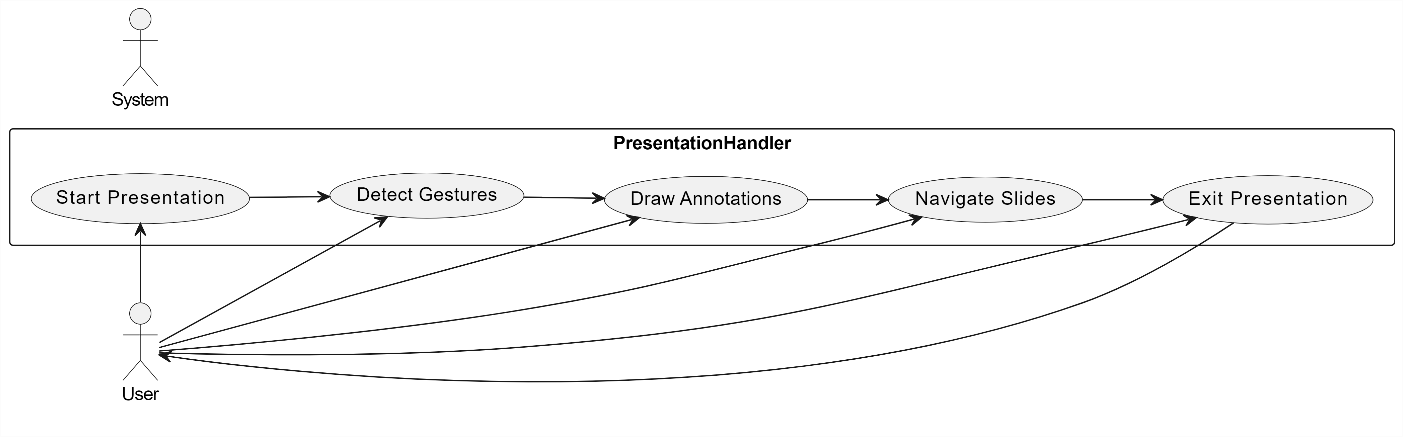
****

* 1. **UML Diagrams**

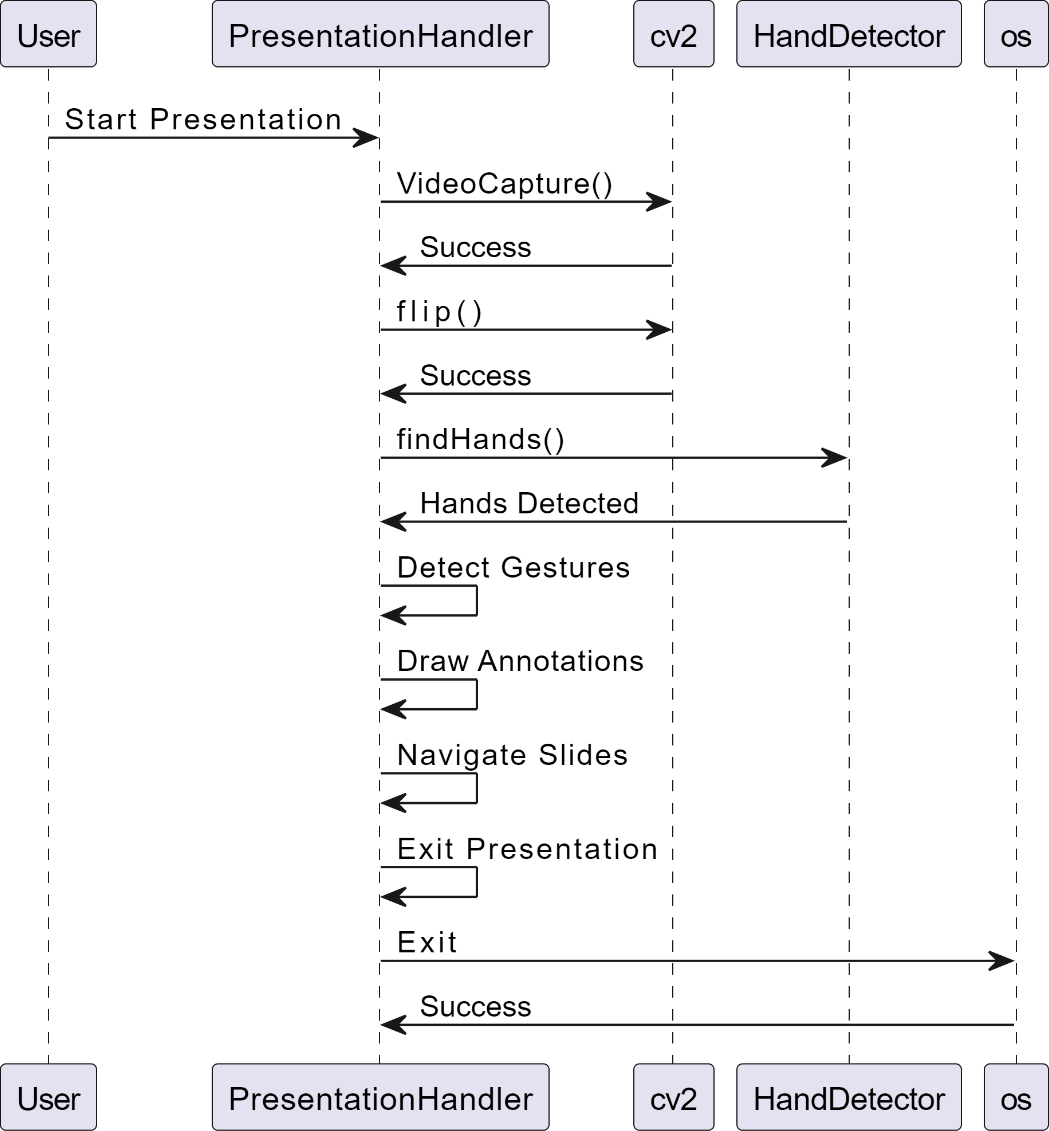
Class diagram:



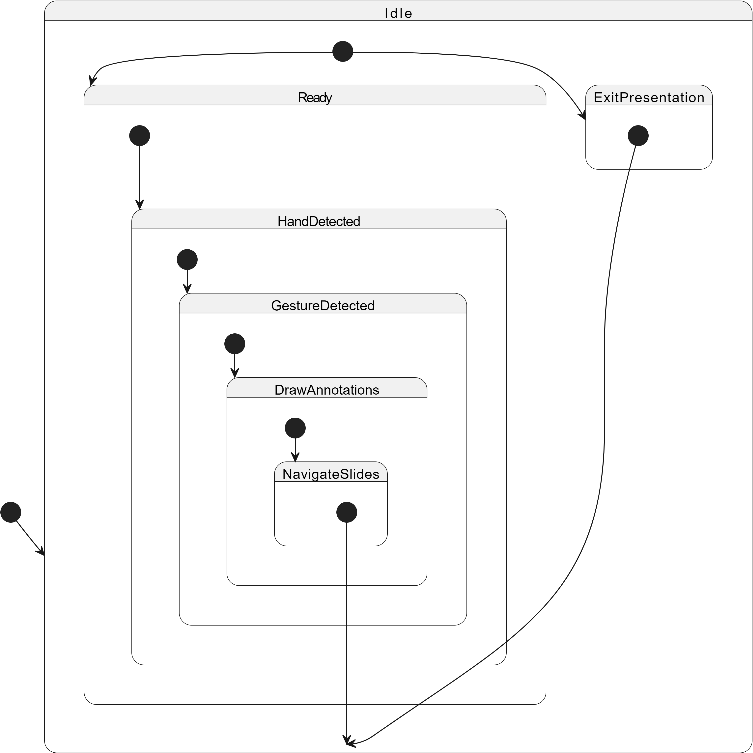
Use case diagram:



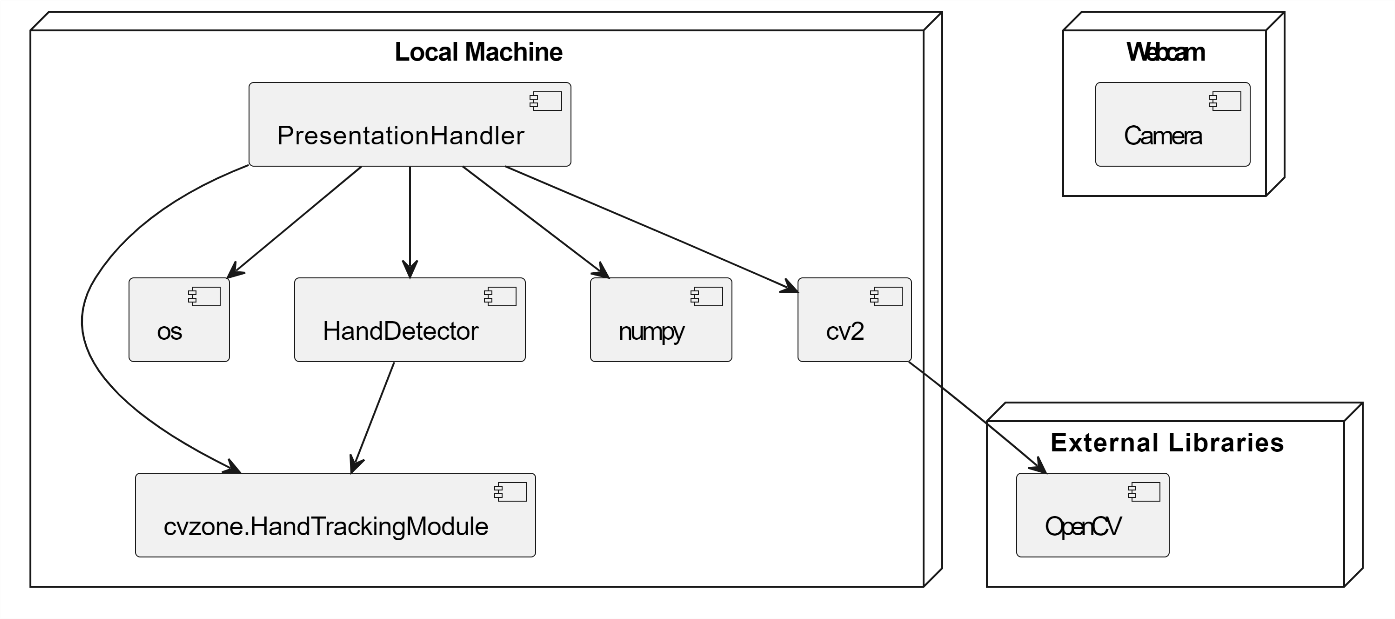
Interaction diagram:



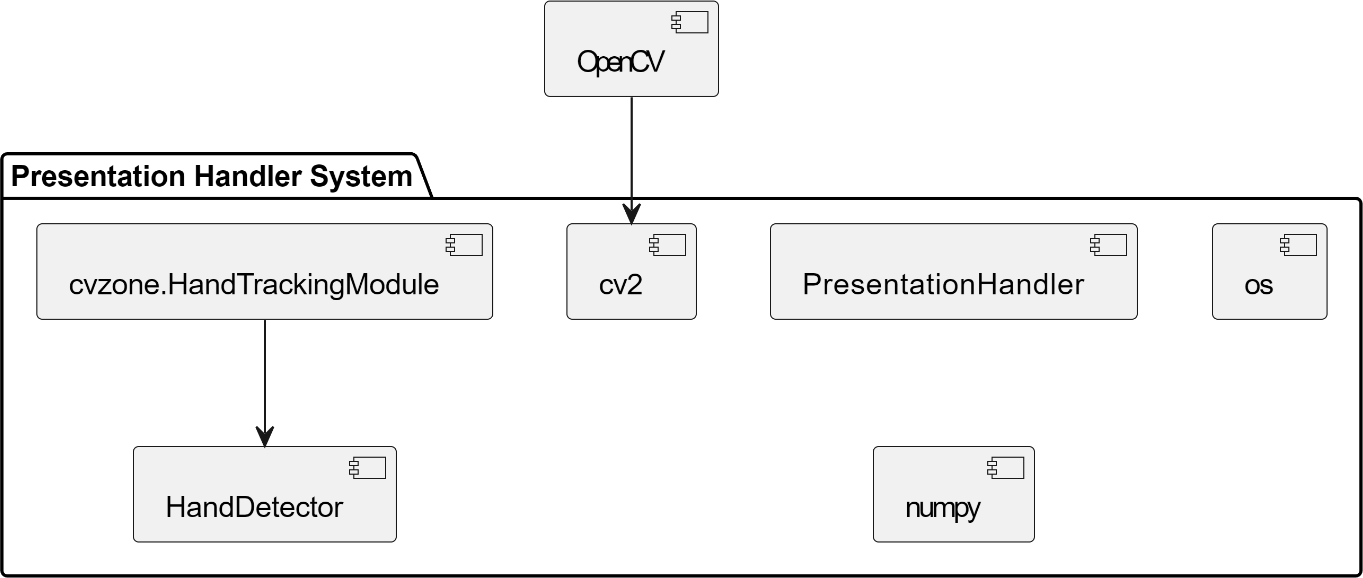
state chart diagram:



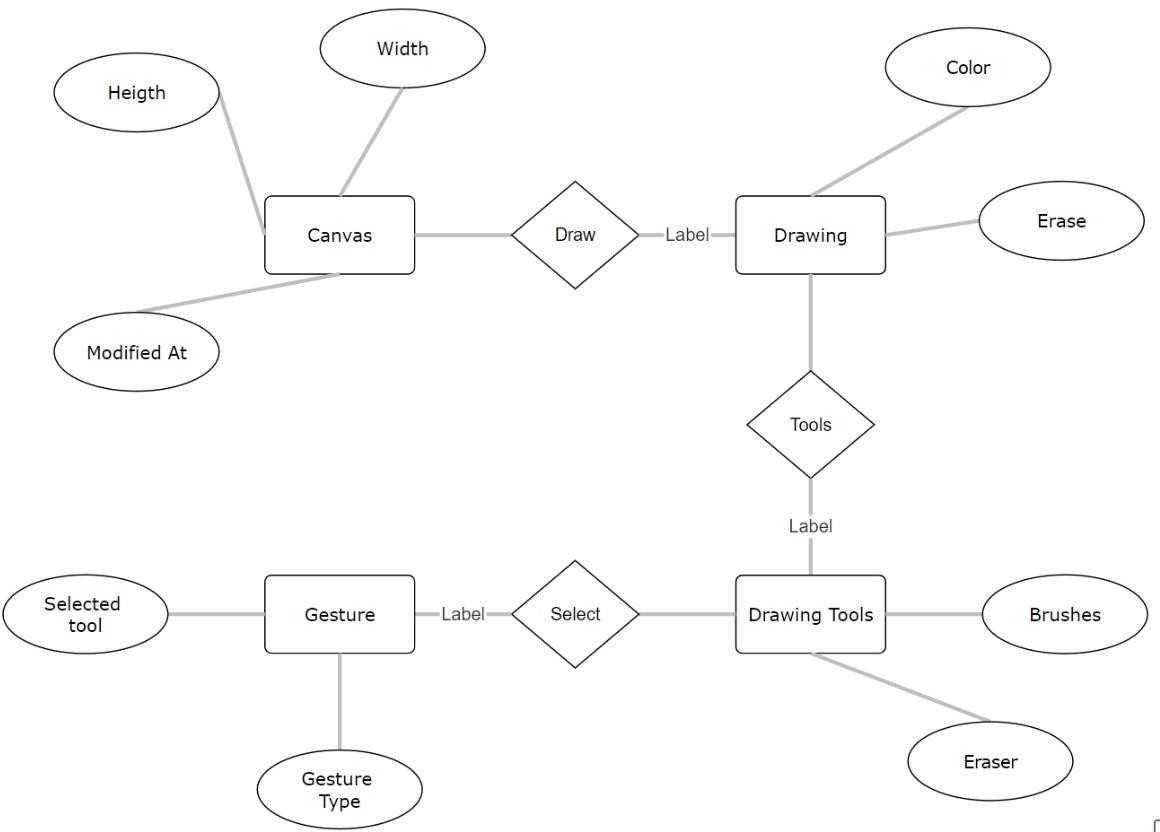
Deployment diagram:



Component diagram:



**E-R Diagram:**

****

1. **IMPLEMENTATION**
   1. **ALGORITHM DESIGN:**

In the context of computer science and software engineering, "algorithm design" refers to the process of creating algorithms to solve specific problems or perform tasks efficiently and effectively. This involves:

#### Understanding the Problem:

Clearly defining the problem that needs to be solved, including input and output specifications.

#### Developing a Strategy:

Determining the overall approach or strategy to solve the problem. This could involve choosing an algorithmic paradigm such as divide and conquer, dynamic programming, greedy algorithms, or backtracking.

#### Designing the Algorithm:

Writing the step-by-step procedure that transforms the input into the desired output. This includes specifying the sequence of operations, data structures to be used, and how to handle different cases and conditions.

#### Analyzing the Algorithm:

Evaluating the algorithm's efficiency in terms of time complexity (how the runtime increases with input size) and space complexity (how the memory usage increases with input size).

#### Implementing the Algorithm:

Writing the actual code to execute the algorithm in a programming language.

#### Testing and Debugging:

Ensuring the algorithm works correctly and efficiently through thorough testing and debugging.

When someone refers to an "algorithm design heading," they are likely referring to a section or title in a document, lecture, or course syllabus that focuses on the principles, methods, and practices involved in designing algorithms. This heading would typically precede detailed discussions, methodologies, and examples of algorithm design**.**

### Sample Code:

import cv2 import os

from cvzone.HandTrackingModule import HandDetector import numpy as np

#variables

width, height = 1740, 770 folderpath = "presentation" #camera setup

cap = cv2.VideoCapture(0) cap.set(3, width)

cap.set(4, height) #get the list of images

pathImages = sorted(os.listdir(folderpath), key=len) # print(pathImages)

#variables for images imgnumber = 0

hs, ws=int(120\*1), int(213\*1) gestureThreshold = 300 buttonpressed=False buttoncounter=0 buttondelay=30 annotations=[[]] annotationNumber=0 annotationStart=False

#Hand detector

detector = HandDetector(detectionCon=0.8,maxHands=1) while True:

#Import Images success, img=cap.read() img = cv2.flip(img,1)

pathFullImage=os.path.join(folderpath,pathImages[imgnumber]) imgcurrent=cv2.imread(pathFullImage) hands,img=detector.findHands(img,flipType=False) cv2.line(img,(0,gestureThreshold),(width,gestureThreshold),(0,255,0),10) if hands and buttonpressed is False:

hand = hands[0] fingers=detector.fingersUp(hand) cx,cy=hand['center'] lmList=hand['lmList']

#constrain easy drawing xVal=int(np.interp(lmList[8][0],[width//2,width],[0,width])) yVal=int(np.interp(lmList[8][1],[150,height-150],[0,height])) indexFinger=xVal,yVal

print(fingers)

if cy<=gestureThreshold: #gesture 1

if fingers==[0,0,0,0,0]: print("left") annotationStart=False if imgnumber>0:

buttonpressed=True annotations=[[]] annotationNumber=0 imgnumber-=1

#gesture 2

if fingers==[1,0,0,0,1]: annotationStart=False print("right")

if imgnumber<(len(pathImages)-1): buttonpressed=True annotations=[[]] annotationNumber=0

imgnumber+=1 #gesture 3 -show pointer if fingers==[1,1,1,0,0]:

cv2.circle(imgcurrent,indexFinger,9,(205,92,92),cv2.FILLED) annotationStart=False

#gesture 4 -Draw

if fingers==[1,1,0,0,0]:

if annotationStart is False: annotationStart=True annotationNumber+=1 annotations.append([])

cv2.circle(imgcurrent,indexFinger,9,(205,92,92),cv2.FILLED) annotations[annotationNumber].append(indexFinger)

else:

annotationStart=False #gesture 5 to undo

if fingers==[1,1,1,1,0]: if annotations:

if annotationNumber>=0: annotations.pop(-1) annotationNumber-=1 buttonpressed=True

else:

annotationStart=False

if buttonpressed: buttoncounter+=1

if buttoncounter>buttondelay: buttoncounter=0 buttonpressed=False

for i in range(len(annotations)):

for j in range(len(annotations[i])):

if j!=0:

cv2.line(imgcurrent,annotations[i][j-1],annotations[i][j],(205,92,92),5) #adding webcam image on the screen

imgsmall = cv2.resize(img,(ws,hs)) h,w,\_=imgcurrent.shape imgcurrent[0:hs,0:ws]=imgsmall cv2.imshow("Image",img) cv2.imshow("slides",imgcurrent) key = cv2.waitKey(1)

if key==ord('q'): break

1. **TESTING**

The Air Canvas project underwent a comprehensive testing process to ensure its functionality, usability, and performance. The testing objectives were to verify that the application initialized correctly, accurately recognized air gestures, facilitated seamless canvas interaction, and provided an intuitive user interface. The testing environment comprised a high-performance system equipped with a depth-sensing camera, running on Windows 10 with development tools such as Python, OpenCV, and MediaPipe, alongside testing tools like Selenium and PyTest.

Functional testing included verifying the initialization process, where the application was expected to start without errors and display the initial interface. Gesture recognition was tested by performing predefined air gestures, ensuring the system correctly interpreted and executed these gestures. Canvas interaction was assessed by drawing and erasing shapes using air gestures, with the expectation that the canvas would accurately reflect these actions.

Usability testing focused on evaluating the user interface's ease of use and intuitiveness. Testers navigated through the application’s features and performed common tasks like selecting tools and saving work, ensuring the interface was user- friendly. Performance testing measured the application’s responsiveness to rapid gesture sequences, expecting prompt responses. Additionally, resource usage was monitored to ensure efficient use of CPU and memory without causing system slowdowns.

The testing was executed over a scheduled period, with designated testers performing each test case and recording the results. The results included a summary of the total test cases, along with the number of passed, failed, and blocked cases. Detailed results and bug reports were documented, providing descriptions, steps to reproduce, severity, and status of each identified issue.

In conclusion, the overall assessment of the Air Canvas project indicated robust performance and usability, with recommendations provided for potential improvements. The appendices included test data, screenshots, and log files, offering comprehensive documentation of the testing process and outcomes.

1. **RESULTS**

**HAND GESTURES:**

****

Figure 1:Drawing Point



Figure 2:Undo



Figure 3:Next slide



Figure 4:Highlighting Point



Figure 5:Previous slide

## OUTPUTS:

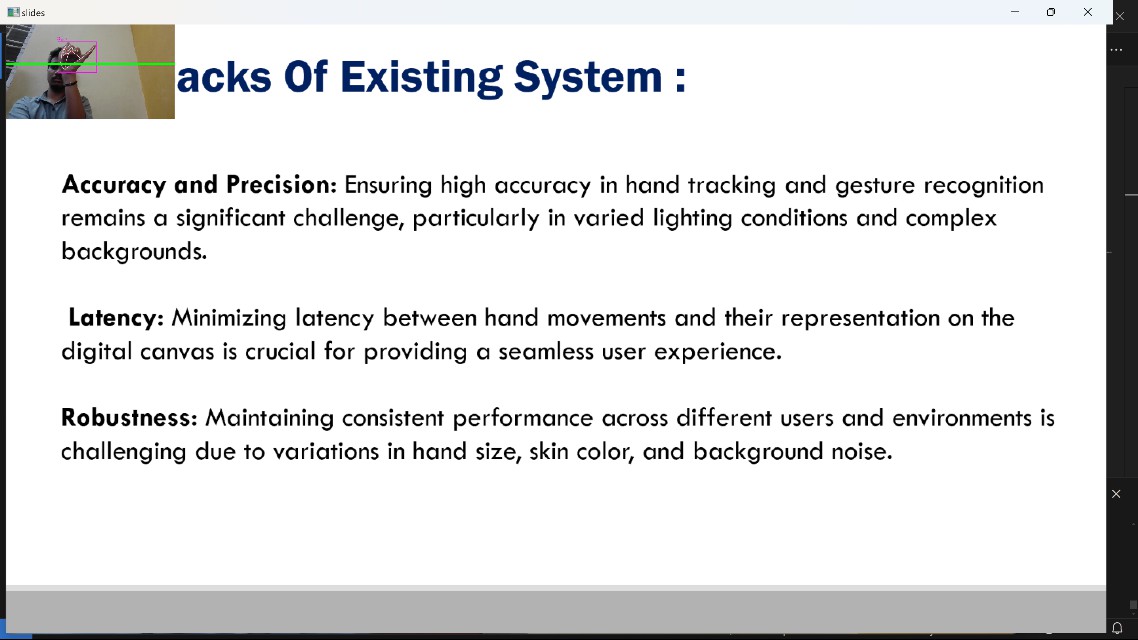
****

Figure 1:Previous slide

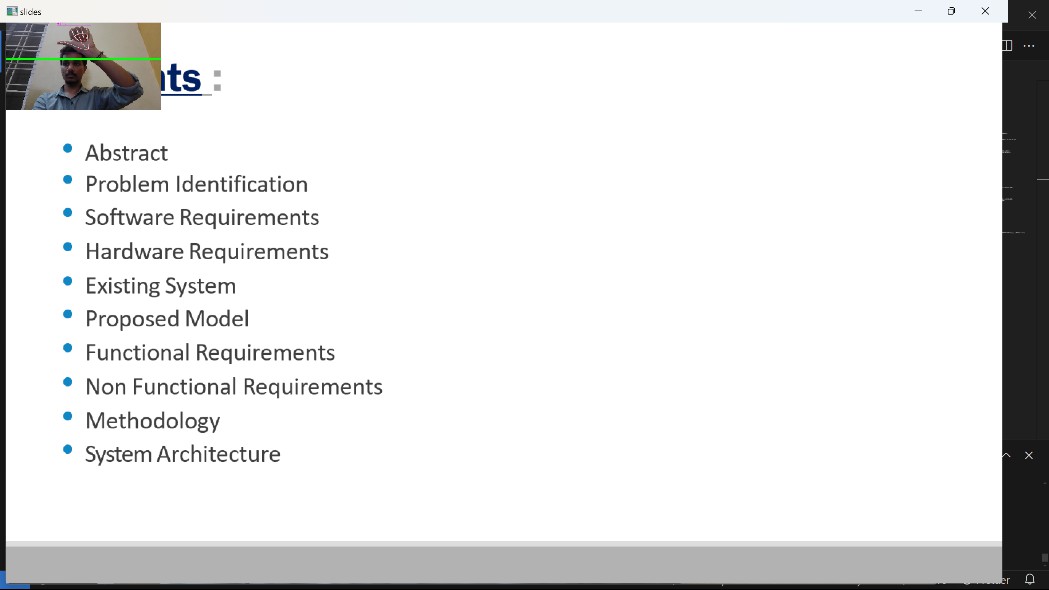


Figure 2:Next slide

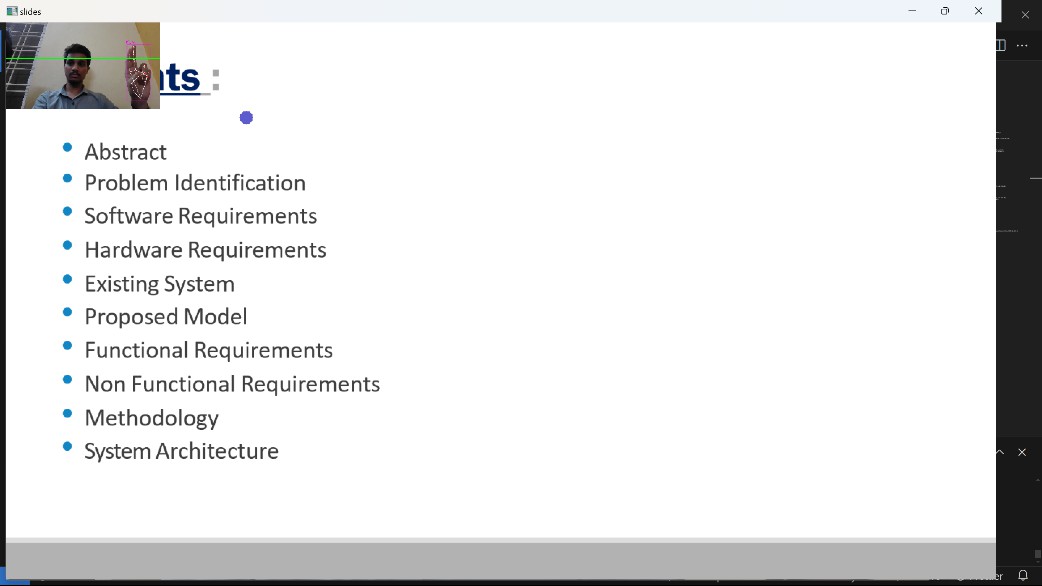


Figure 3:Highlighting Point

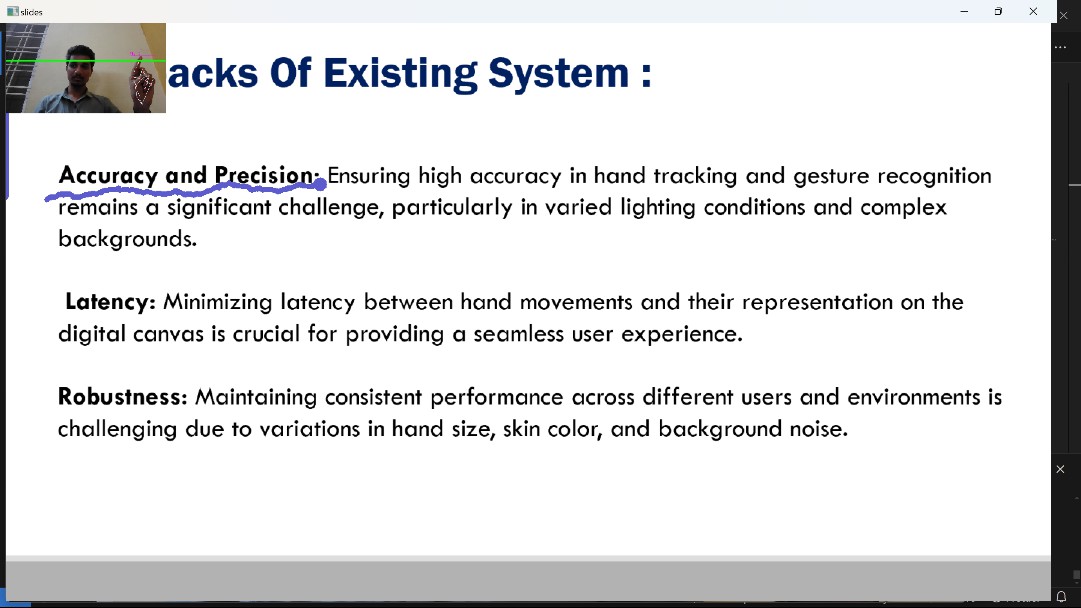


Figure 4:Drawing Point

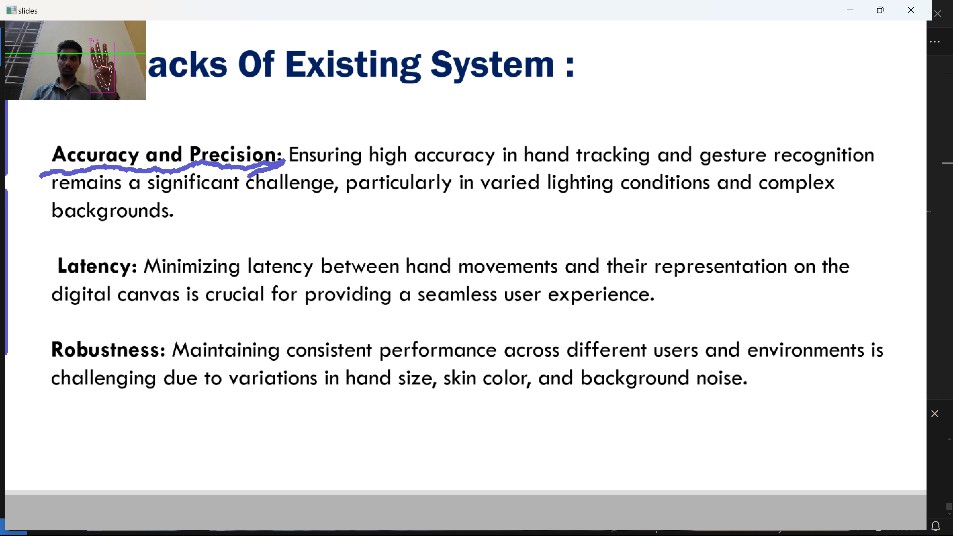


Figure 5:Undo

1. **CONCLUSION**

Assesses how well users interact with the system, paying particular attention to user happiness, accuracy of gesture detection, and overall presentation control efficiency.Seek to provide insights into CV-based strategies for engaging,dynamic presenting environments.

#### FUTURE WORK

Given more time to work on this project, we would improve hand contour recognition, explore our original Air Canvas goals, and try to understand the multicore module.

To enhance hand gesture tracking, we would have to delve more into OpenCV. There are many different methods of contour analysis, but in this particular algorithm, it may be worthwhile to take a look at the color histogram used to create the contours in question. Furthermore, we could experiment with different interpolation methods. PyGame includes a line drawing method (pygame.draw.line()) that could prove useful in producing smoother, cleaner lines. On the same vein, implementing a variety of brush shapes, textures, and even an eraser would make Air Canvas more robust as a drawing program. Allowing the user to save their final work or watch their drawing process as a playback animation could also be unique features that resemble real creativity software. Perhaps there would even be a way to connect Air Canvas to actual digital drawing programs such as Adobe Photoshop, Clip Studio Paint, or GIMP! Finally, we could make significant strides by figuring out how multicore processing works with in-order information processing.

**REFERENCES**

* Finger Detection” by

Izane: [https://github.com/lzane/Fingers-Detection-using-](https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python) [OpenCV-and-Python](https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python)

* “Finger Detection and Tracking” by Amesh Prakash Pandey: [https://dev.to/amarlearning/finger-detection-and-](https://dev.to/amarlearning/finger-detection-and-tracking-using-opencv-and-python-586m) [tracking-using-opencv-and-python-586m](https://dev.to/amarlearning/finger-detection-and-tracking-using-opencv-and-python-586m)
* OpenCV

contours: [https://docs.opencv.org/3.4.2/d4/d73/tutorial\_py\_](https://docs.opencv.org/3.4.2/d4/d73/tutorial_py_contours_begin.html) [contours\_begin.html](https://docs.opencv.org/3.4.2/d4/d73/tutorial_py_contours_begin.html)

* “AutoTurret”: [https://courses.ece.cornell.edu/ece5990/ECE](https://courses.ece.cornell.edu/ece5990/ECE5725_Spring2018_Projects/fy57_xz522_AutoTurret/index.html) [5725\_Spring2018\_Projects/fy57\_xz522\_AutoTurret/index.](https://courses.ece.cornell.edu/ece5990/ECE5725_Spring2018_Projects/fy57_xz522_AutoTurret/index.html) [html](https://courses.ece.cornell.edu/ece5990/ECE5725_Spring2018_Projects/fy57_xz522_AutoTurret/index.html)
* DrawinginPyGame: [https://www.pygame.org/docs/ref/dra](https://www.pygame.org/docs/ref/draw.html) [w.html](https://www.pygame.org/docs/ref/draw.html)
* “Guide to Set Up Pi Camera & Some OpenCV Image Processing Basics” by Xitang

Zhao: [https://blackboard.cornell.edu/bbcswebdav/pid-](https://blackboard.cornell.edu/bbcswebdav/pid-4087974-dt-content-rid-24070270_1/courses/12153_2019SP/12153_2019SP_ImportedContent_20190408043030/Pi_camera_and_OpenCV_v1%281%29.pdf) [4087974-dt-content-rid-](https://blackboard.cornell.edu/bbcswebdav/pid-4087974-dt-content-rid-24070270_1/courses/12153_2019SP/12153_2019SP_ImportedContent_20190408043030/Pi_camera_and_OpenCV_v1%281%29.pdf) [24070270\_1/courses/12153\_2019SP/12153\_2019SP\_Impo](https://blackboard.cornell.edu/bbcswebdav/pid-4087974-dt-content-rid-24070270_1/courses/12153_2019SP/12153_2019SP_ImportedContent_20190408043030/Pi_camera_and_OpenCV_v1%281%29.pdf) [rtedContent\_20190408043030/Pi\_camera\_and\_OpenCV\_v](https://blackboard.cornell.edu/bbcswebdav/pid-4087974-dt-content-rid-24070270_1/courses/12153_2019SP/12153_2019SP_ImportedContent_20190408043030/Pi_camera_and_OpenCV_v1%281%29.pdf) [1%281%29.pdf](https://blackboard.cornell.edu/bbcswebdav/pid-4087974-dt-content-rid-24070270_1/courses/12153_2019SP/12153_2019SP_ImportedContent_20190408043030/Pi_camera_and_OpenCV_v1%281%29.pdf)