HAND GESTURE CONTROLLED PPT

A Summer Internship Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

G. ANIHANT	21071A7207
R. NAVYA SREE	21071A7259
A. SAATWIK	21071A7261

Under the guidance of

Ms. KRISHNA PRIYA
Assistant Professor, Department of CSE - (CyS, DS) and AI&DS

DEPARTMENT OF CSE-(CyS, DS) and AI&DS

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090

DEPARTMENT OF CSE-(DS, CyS) and AI & DS

CERTIFICATE

This is to certify that the project report entitled **HAND GUESTURE-CONTROLLED PPT** is a bonafide work done under our supervision and is being submitted by G. Anihant (21071A7207), R. NavyaSree (21071A7259), A. Saatwik(21071A7261) in partial fulfillment for the award of the degree of Bachelor of Technology in Artificial Intelligence and Data Science, of the VNRVJIET, Hyderabad during the academic year 2023-2024. Certified further that to the best of our knowledge the work presented in this project has not been submitted to any other University or Institute for the award of any Degree or Diploma.

Ms. Krishna Priya
Assistant Professor,
Dept. of CSE-(CyS, DS) and AI &DS
VNR VJIET

Dr. M. Rajasekar
Professor and Head of the
Dept. of CSE-(CyS, DS) and AI &DS
VNR VJIET

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

An Autonomous Institute, NAAC Accredited with 'A++' Grade NBA Accredited for CE, EEE, ME, ECE, CSE, EIE, IT B. Tech Courses Approved by AICTE, New Delhi, Affiliated to JNTUH Recognized as "College with Potential for Excellence" by UGC ISO 9001:2015 Certified, QS I GUAGE Diamond Rated

Vignana Jyothi Nagar, Pragathi Nagar, Nizampet (S.O), Hyderabad – 500 090

DEPARTMENT OF CSE - (CyS, DS) and AI&DS

DECLARATION

We declare that the Summer Internship project work entitled **HAND GUESTURE CONTROLLED PPT** submitted in the department of CSE-(CyS, DS) and AI&DS, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering and Technology, Hyderabad, in partial fulfillment of the requirement for the award of the degree in **Bachelor of Technology** is a bonafide record of our own work carried out under the supervision of Ms Krishna Priya**,** Assistant Professor**, Department of CSE-(CyS, DS) and AI&DS, VNRVJIET**. Also, we declare that the matter embodied in this thesis has not been submitted by us in full or in any part thereof for the award of any degree of any other institution or university previously.

Place: Hyderabad

G. Anihant	**R. Navyasree**	**A. Saatwik**
(21071A7207)	(21071A7259)	(21071A7261)

ACKNOWLEDGEMENT

Firstly, we would like to express our immense gratitude towards our institution VNR Vignana Jyothi Institute of Engineering and Technology, which created a great platform to attain profound technical skills in the field of Information Technology, thereby fulfilling our most cherished goal.

We are very much thankful to our Principal, **Dr. Challa Dhanunjaya Naidu,** and our Head of Department, **Dr. M Rajasekar**, for extending their cooperation in doing this project within the stipulated time.

We extend our heartfelt thanks to our guide, Ms. Krishna Priya**,** and the project coordinators **Mr. Manmath Nath Das,** and **Ms. E Lalitha** for their enthusiastic guidance throughout the course of our project.

Last but not least, our appreciable obligation also goes to all the staff members of the Information Technology department and to our fellow classmates who directly or indirectly helped us.

G. Anihant	21071A7207
R. Navyasree	21071A7259
A. Saatwik	21071A7261

ABSTRACT

The Hand Gesture-Controlled PPT project represents a paradigm shift in presentation dynamics, seeking to transform the traditional approach to slide management through innovative gesture-based control. It stands as a testament to the fusion of advanced technologies like machine learning and computer vision, harnessing these tools to decode and respond to an extensive range of hand movements. By liberating presenters from conventional input devices, this system aims to elevate the presentation experience, enabling seamless slide navigation, dynamic content emphasis, and effortless animation triggers, all achieved through intuitive hand gestures.

At its core, this project embodies the spirit of inclusivity and accessibility, bridging gaps for individuals with disabilities or limited mobility by offering an independent means of delivering impactful presentations. Leveraging sophisticated gesture recognition libraries such as OpenCV and MediaPipe, the system not only revolutionizes slide control but also encourages personalized interactions, empowering users to define custom gestures for specific actions. Beyond mere functionality, this innovation redefines audience engagement, fostering a more interactive and immersive presentation environment that captivates and resonates with diverse audiences. In essence, the Hand Gesture-Controlled PPT project reimagines the art of presenting, harmonizing technology, accessibility, and enhanced engagement into a seamless and captivating experience.

1. Introduction **10**

2. **Literature Survey/ Existing System** **1**1

2.1 Feasibility Study 11

2.1.1 Organizational Feasibility 11

2.1.2 Economic Feasibility 11

2.1.3 Technical Feasibility 11

2.1.4 Behavioural Feasibility 11

2.2 Literature Review 12

2.3 Existing System 13

2.4 Drawbacks Of the Existing System 13

3. **Software Requirement Analysis** **14**

3.1 Introduction 14

3.1.1 Document Purpose 14

3.1.2 Definitions 14

3.2 System Architecture 15

3.3 Functional Requirements 16

3.4 System Analysis 16

3.5 Non-Functional Requirements 18

3.6 Software Requirement Specification 18

3.7 Software Requirements 19

3.8 Hardware Requirements 19

4. Software Design **20**

4.1 UML Diagrams 21

4.1.1 Use Case Diagram 21

4.1.2 Sequence Diagram 26

4.1.3 Activity Diagram 30

5. Proposed System **32**

5.1 Methodology 32

5.2 Functionalities 34

5.2.1 Slide Navigation 34

5.2.2 Emphasizing Content 34

5.2.3 Defining Custom Gestures 34

5.2.4 Providing Feedback 34

5.3 Advantages Of Proposed System 33

6. Coding/Implementation **35**

6.1 Dataset 35

6.2 Implementation 35

7. Testing **40**

7.1 Types Of Testing 40

7.1.1 Manual Testing 40

7.1.2 Automated Testing 40

7.2 Software Testing Methods 41

7.2.1 Black Box Testing 41

7.2.2 Gray Box Testing 41

7.2.3 White Box Testing 42

7.3 Testing Levels 43

7.3.1 Non-Functional Testing 43

7.3.1.1 Performance Testing 43

7.3.1.2 Stress Testing 43

7.3.1.3 Security Testing 43

7.3.1.4 Portability Testing 43

7.3.1.5 Usability Testing 44

7.3.2 Functional Testing 44

7.3.2.1 Integration Testing 44

7.3.2.2 Regression Testing 44

7.3.2.3 Unit Testing 44

7.3.2.4 Alpha Testing 45

7.3.2.5 Beta Testing 45

7.4 Test Cases 45

8. **Results** **46**

9. Conclusion and Future work **48**

10. References **49**

List of Tables

Table	Page No
Table 4.1.2.1 Sequence diagram Symbols	28
Table. 7.4.1 Test cases	45

List of Figures

Figure	Page No
Fig 3.2.1 System Architecture	15
Fig 3.4.1 Analysis of the system	16
Fig 4.1.1.1 Use Case diagram for the application	24
Fig 4.1.2.1 Sequence diagram for the system	29
Fig 4.1.3.1 Activity Diagram for the system	31
Fig 5.1.1 Methodology	32
Fig 5.1.2 Mediapipe Hand Landmark	33
Fig 7.2.1.1 Black Box Testing	41
Fig 7.2.1.2 Grey Box Testing	42
Fig 7.2.3.1 White Box Testing	42
Fig:8.1 Hand Gesture To move on to next slide	46
Fig:8.2 Hand Gesture for going back to previous slide	46
Fig 8.3: Getting A pointer on slide	47
Fig 8.4 Draw using That Pointer	47
Fig 8.5 Erase the Drawing on Slide	47

1. INTRODUCTION

In the realm of modern presentations, the Hand Gesture-Controlled PowerPoint (PPT) system stands at the forefront of innovation, offering a transformative approach to slide management and presenter-audience interactions. This groundbreaking project redefines the traditional paradigm of navigating presentations, revolutionizing the way presenters engage with their content through intuitive hand gestures. Departing from the limitations of conventional input devices, this system embodies the convergence of cutting-edge technologies—leveraging machine learning, computer vision, and gesture recognition—to empower presenters and elevate audience experiences.

The objective of this report is to delve into the conceptualization, development, and implications of the Hand Gesture-Controlled PPT system. By providing an overview of its technological underpinnings, integration with popular presentation software, emphasis on inclusivity and accessibility, and its potential to reshape the landscape of presentations, this report aims to offer a comprehensive understanding of this innovative project. From its motivation to its proposed enhancements and implications for the future of presentations, this report will navigate the key aspects and impacts of this pioneering system, shedding light on its significance in advancing presentation dynamics and accessibility in the digital era.

2. LITERATURE SURVEY/ EXISTING SYSTEM

2.1 FEASIBILITY STUDY

The feasibility study for the Hand Gesture-Controlled PPT system involves a comprehensive evaluation of technical, economic, and operational factors to ascertain the practicality and viability of implementing this innovative technology. Assessing hardware requirements, technological capabilities, cost-effectiveness, user acceptance, and the system's compatibility with existing presentation platforms will be integral in determining the feasibility of widespread adoption.

2.1.1 ORGANIZATIONAL FEASIBILITY

Assessing the organizational feasibility of the Hand Gesture-Controlled PPT system involves evaluating the readiness and capacity of the organization to adopt and implement this innovative technology. Factors such as the organization's technological infrastructure, expertise in integrating new systems, availability of resources, and the willingness of stakeholders to embrace this change will play a crucial role.

2.1.2 ECONOMIC FEASIBILITY

The economic feasibility study for the Hand Gesture-Controlled PPT system assesses its cost-effectiveness and financial viability. It involves analyzing initial investments, potential savings or revenue, and overall return on investment. Factors such as development costs, hardware expenses, maintenance, and benefits like productivity enhancement and audience engagement determine its long-term financial sustainability.

2.1.3 TECHNICAL FEASIBILITY

The technical feasibility of the Hand Gesture-Controlled PPT system assesses its compatibility with existing technology, resource availability, and the feasibility of implementing necessary components like gesture recognition and machine learning. It considers hardware needs, software compatibility, and the system's accuracy in interpreting hand gestures for successful implementation.

2.1.4 BEHAVIORAL FEASIBILITY

Behavioral feasibility studies how well users adapt to the hand gesture-controlled system within the organization. It assesses user acceptance, training needs, and the system's integration into presentation styles, ensuring its compatibility with user habits for effective adoption.

2.2 LITERATURE REVIEW

Introduction

The introduction section contextualizes the Hand Gesture-Controlled PPT system within presentation technology, emphasizing its potential to revolutionize audience engagement and presenter interaction through intuitive gestures. It outlines the project's motivations and technological advancements, setting the stage for an in-depth exploration of existing literature and its relevance to human-computer interaction in presentations.

Effectiveness of Fully Automatic Agricultures:

The Hand Gesture-Controlled PPT system excels in revolutionizing presentations by enabling intuitive slide control through gestures. It enhances presenter autonomy, streamlines navigation, fosters audience engagement, and promotes inclusivity for individuals with disabilities. Integrating seamlessly with popular software and offering customizable gestures, it sets a new standard for accessible and engaging presentations.

Techniques Used in Fully Automatic Agricultures:

The Hand Gesture-Controlled PPT system harnesses sophisticated techniques like machine learning, computer vision, and gesture recognition libraries such as OpenCV and MediaPipe. Through machine learning algorithms, it deciphers and maps an extensive range of hand movements, enabling precise control over PowerPoint functionalities. Leveraging computer vision, the system accurately detects gestures, ensuring responsiveness and seamlessness in interpreting and executing user commands. Integration with gesture recognition libraries enhances its capability to recognize diverse gestures, empowering presenters with a more intuitive and interactive means of controlling their presentation.

2.3 EXISTING SYSTEM

The existing Hand Gesture-Controlled PPT systems typically rely on specific hardware components like depth cameras or motion sensors for accurate hand gesture tracking. These systems often employ pre-defined gesture patterns, such as swiping left or right to change slides or utilizing specific hand movements to activate designated actions within PowerPoint. However, limitations persist, including a restricted set of recognized gestures, primarily aimed at basic slide navigation, and reliance on integration with Microsoft PowerPoint, which confines the system's functionalities to a singular platform. Despite these constraints, these systems mark a significant step towards intuitive presentation control, albeit within defined boundaries of recognized gestures and software compatibility.

2.4 DRAWBACKS OF THE EXISTING SYSTEM

- **Limited Gesture Set:** Existing systems often have a restricted repertoire of recognized hand gestures, limiting the range of interaction and functionalities available to presenters.
- **Hardware Dependencies:** They rely on specific hardware components like depth cameras or motion sensors, making them less flexible in terms of accessibility and deployment.
- **Platform Dependency:** Most existing systems are tailored for integration with Microsoft PowerPoint, limiting their use to this specific software and restricting compatibility with other presentation platforms.
- **Lack of Customization:** Users have limited or no options to define or customize their own hand gestures for specific actions or commands, reducing the personalization of interactions.
- **Complexity in Recognition:** Some systems struggle with accurate and consistent gesture recognition, leading to occasional misinterpretation of gestures and consequent errors in controlling presentations.

3. SOFTWARE REQUIREMENT ANALYSIS

3.1 INTRODUCTION

The software requirement analysis delineates the prerequisites for developing Hand Gesture-Controlled PPT. It identifies the essential functionalities, features, and constraints for enhancing this technological endeavor.

3.1.1 DOCUMENT PURPOSE

This document outlines the specific software needs and objectives critical for the design, development, and implementation of Hand Gesture-Controlled PPT, facilitating a comprehensive understanding among stakeholders.

3.1.2 DEFINITIONS

- Hand Gesture Controlled PPT: The Hand Gesture-Controlled PPT system introduces a groundbreaking way for presenters to interact with slides using intuitive hand gestures. It promises dynamic, engaging presentations, departing from traditional input devices to enhance audience engagement and inclusivity. This innovation sets the stage for a transformative shift in how information is presented and received.

- Functionalities: The Hand Gesture-Controlled PPT system enables seamless slide navigation through intuitive gestures while offering advanced functionalities like emphasizing key points, triggering animations, and customizing interactions. It allows zooming, content highlighting, and media activation, fostering engaging and personalized presentations with adaptable gesture controls.

- Constraints: Constraints of the Hand Gesture-Controlled PPT system include limited gesture recognition, dependency on specific hardware, reliance on Microsoft PowerPoint integration, and restricted compatibility with other platforms. Limited customization options also hinder its versatility, yet ongoing advancements aim to address these constraints for a more universal and adaptable presentation solution.

3.2 SYSTEM ARCHITECTURE

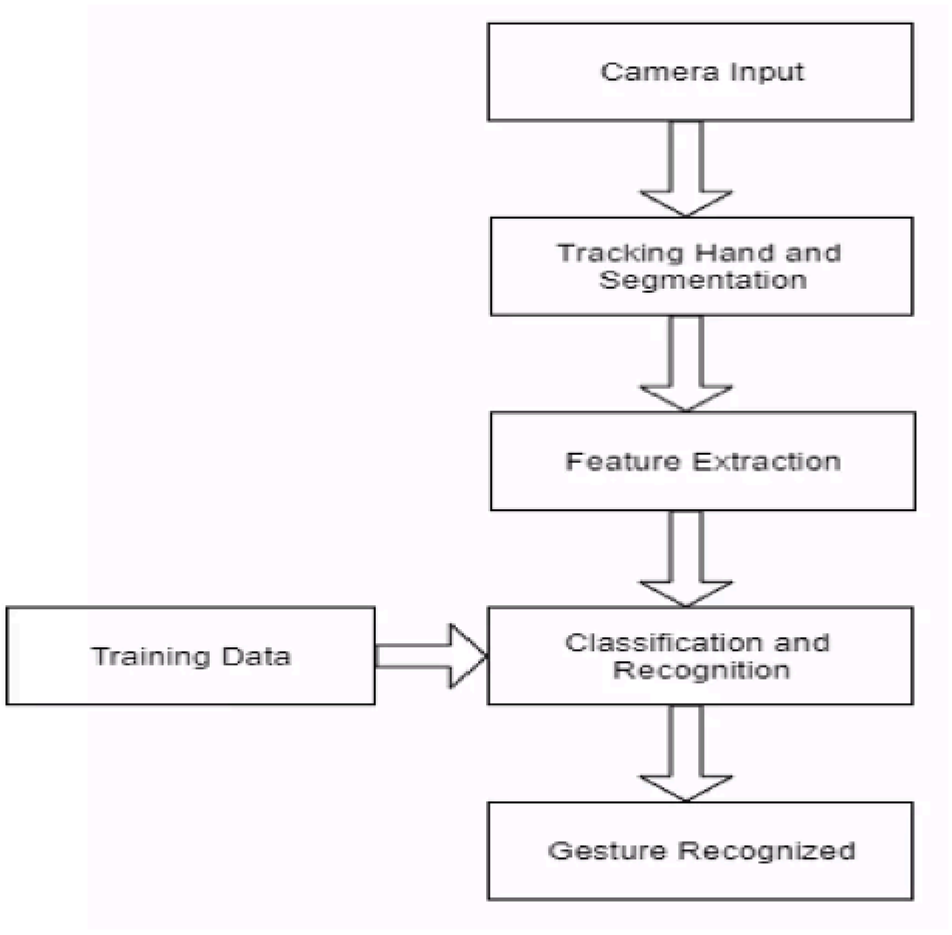


Fig 3.2.1: System Architecture

The Hand Gesture-Controlled PPT system comprises a multi-stage architecture. Initially, a camera input module captures live video feed, processed by a hand tracking and segmentation module, identifying and isolating the hand within frames. Extracting pertinent features, such as hand shape or key points, follows this. Training data is generated, associating specific gestures with extracted features. Utilizing machine learning algorithms, a classification model is trained for gesture recognition. During live operation, this model interprets real-time video feed, enabling the system to recognize and classify gestures, facilitating intuitive control over presentations through hand movements. This architecture combines computer vision, machine learning, and gesture recognition for seamless and dynamic interaction within presentation environments.

3.3 FUNCTIONAL REQUIREMENTS

- Gesture Recognition
- Real-time Tracking
- Customizable Gestures
- Platform Compatibility
- Accessibility Features

3.4 SYSTEM ANALYSIS

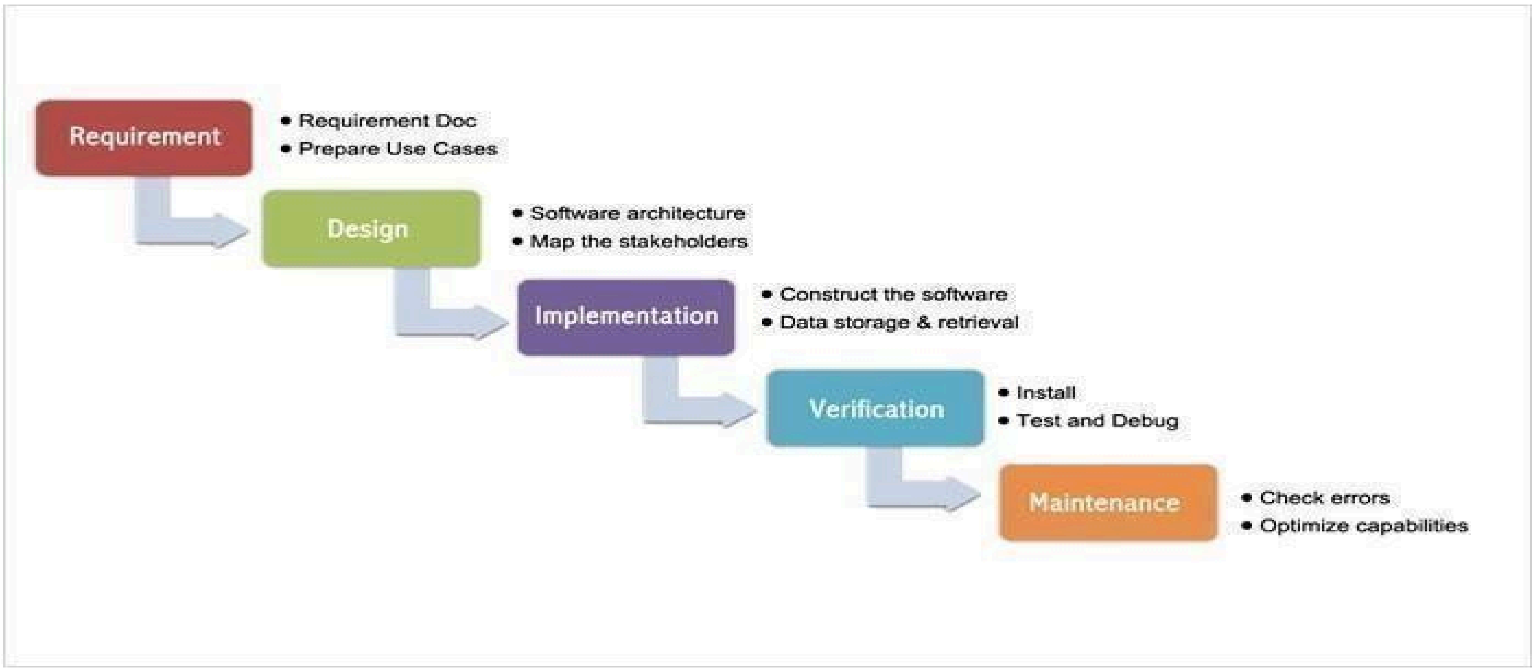


Fig:3.4.1 Analysis of the system

It was the first Process Model to be introduced. A linear sequential life cycle model is another name for it. It's quite simple to use and understand. Phases do not overlap in this paradigm, and each phase must be finished before the next one begins. The first SDLC approach used during software development was the waterfall model. The model shows that the development of software is linear and is a sequential process. Only after one phase of the development is completed, we can go to the next phase. In this waterfall paradigm, the phases do not overlap.

The steps in the waterfall model are explained below

1. **Requirements:** This phase involves eliciting and documenting stakeholder needs and system functionalities for the Hand Gesture-Controlled PPT system. It includes discussions, interviews, and surveys to understand user expectations, feature specifications, and technical constraints.

2. **Analysis:** Analyzing collected requirements to determine feasibility, constraints, and potential challenges. It involves refining requirements, prioritizing functionalities, and establishing the system's scope, ensuring alignment with project objectives.

3. **Design:** Creating a blueprint for the Hand Gesture-Controlled PPT system based on analyzed requirements. This phase includes architectural design, interface design, and database design. It focuses on defining system components, interactions, and data flow.

4. **Implementation:** Translating the system design into actual code and software components. This phase involves writing and integrating the necessary algorithms, functionalities, and interfaces to develop the Hand Gesture-Controlled PPT system as per design specifications.

5. **Testing:** Evaluating the developed system to ensure it meets specified requirements. It encompasses unit testing, integration testing, and system testing, aiming to identify and rectify defects, ensuring functionality, usability, and reliability.

6. **Maintenance:** Post-implementation phase focusing on system upkeep, enhancements, and bug fixes. It involves continuous monitoring, user feedback incorporation, and periodic updates to ensure the Hand Gesture-Controlled PPT system's effectiveness and adaptability to evolving needs.

3.5 NON-FUNCTIONAL REQUIREMENTS

1. **Performance:** Ensure real-time responsiveness and minimal latency in recognizing hand gestures, maintaining a seamless interaction with the presentation software.

2. **Accuracy:** Achieve high accuracy in gesture recognition to avoid misinterpretation and ensure precise control over presentation commands.

3. **Usability:** Design an intuitive and user-friendly interface, ensuring ease of use for presenters of varying technical abilities.

4. **Reliability:** Ensure consistent and reliable gesture recognition across different environmental conditions (e.g., varying lighting, backgrounds) to maintain system effectiveness.

5. **Security:** Implement robust security measures to protect user data and prevent unauthorized access or manipulation of the system.

6. **Compatibility:** Ensure compatibility with diverse devices, operating systems, and presentation software to maximize accessibility and usability across various platforms.

3.6 SOFTWARE REQUIREMENT SPECIFICATIONS

OpenCV:

OpenCV is a robust open-source computer vision library, offering tools for real-time image processing, object detection, and gesture recognition. Supporting multiple programming languages like Python and C++, it simplifies complex tasks in computer vision for applications in robotics, augmented reality, and facial recognition. Its continuous updates and active community support make it a leading choice for developers creating innovative solutions, including gesture-controlled systems like the Hand Gesture-Controlled PPT.

MediaPipe:

MediaPipe is a versatile open-source framework developed by Google, focusing on building cross-platform machine learning solutions for perceptual computing. It offers a wide range of tools and ready-to-use modules for tasks like hand tracking, pose estimation, face detection, and gesture recognition. With its easy integration and pre-built models, MediaPipe simplifies the development of complex machine learning and computer vision applications, making it particularly valuable for creating gesture-controlled systems like the Hand Gesture-Controlled PPT.

3.7 SOFTWARE REQUIREMENTS

Software	:	PyCharm
Operating System	:	Windows
Technology	:	OpenCV, MediaPipe, Computer Vision, Machine Learning

3.8 HARDWARE REQUIREMENTS

- Laptop with clear camera
- Minimum 4GB Ram Laptop
- Processor: Intel Core i5/i7 or AMD Ryzen
- Internet Connection

4. SOFTWARE DESIGN

4.1 UML DIAGRAMS

The Device Architecture Manual describes the application requirements, operating state, application and subsystem functionality, documents and repository setup, input locations, yield types, human-machine interfaces, management reasoning, and external interfaces. The Unified Modeling Language (UML) assists software developers in expressing an analysis model through documents that contain a plethora of syntactic and semantic instructions. A UML context is defined as five distinct viewpoints that present the system from a particularly different point of view.

The components are similar to modules that can be combined in a variety of ways to create a complete UML diagram. As a result, comprehension of the various diagrams is essential for utilizing the knowledge in real-world systems. The best method to understand any complex system is to draw diagrams or images of it. These designs have a bigger influence on our understanding. Looking around, we can see that info-graphics are not a new concept, but they are frequently utilized in a variety of businesses in various ways.

User Model View
The perspective refers to the system from the clients' point of view. The exam's depiction depicts a situation of utilization from the perspective of end-clients. The user view provides a window into the system from the perspective of the user, with the system's operation defined in light of the user and what the user wants from it.

Structural model view
This layout represents the details and functionality of the device. This software design maps out the static structures. This view includes activity diagrams, sequence diagrams and state machine diagrams

Behavioral Model View
It refers to the social dynamics as framework components, delineating the assortment cooperation between various auxiliary components depicted in the client model and basic model view. UML Behavioral Diagrams illustrate time-dependent aspects of a system and communicate the system's dynamics and how they interact. Behavioral diagrams include interaction diagrams, use case diagrams, activity diagrams and state–chart diagrams.

Implementation Model View
The essential and actions as frame pieces are discussed in this when they are to be

manufactured. This is also referred to as the implementation view. It uses the UML Component diagram to describe system components. One of the UML diagrams used to illustrate the development view is the Package diagram.

Environmental Model View
The systemic and functional component of the world where the program is to be introduced was expressed within this. The diagram in the environmental view explains the software model's after-deployment behavior. This diagram typically explains user interactions and the effects of software on the system. The following diagrams are included in the environmental model: Diagram of deployment.

The UML model is made up of two separate domains:

- Demonstration of UML Analysis, with a focus on the client model and auxiliary model perspectives on the framework.
- UML configuration presenting, which focuses on demonstrations, usage, and natural model perspectives.

4.1.1 USE CASE DIAGRAM

The objective of a use case diagram is to portray the dynamic nature of a system. However, because the aim of the other four pictures is the same, this description is too broad to characterize the purpose. We'll look into a specific purpose that distinguishes it from the other four diagrams.
The needs of a system, including various factors, are collected using use case diagrams. Most of these specifications are design specifications. As a result, use cases are constructed and actors are identified when examining a system to gather its functions.

Use case diagrams are made to represent the outside view once the primary task is completed.
In conclusion, use case diagrams are useful for the following purposes:

- Used to collect a system's requirements.
- Used to get a bird's-eye view of a system.
- Determine various factors that are influencing the system.
- Display the interaction of the requirements as actors.

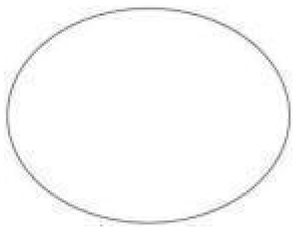
Use case diagrams are used to analyze a system's high-level requirements. The functionality of a system is recorded in use cases when the requirements are examined. Use cases can be defined as "system functionalities written in a logical order." The actors are the second pillar of use cases that is important. Any entities that interact with the system are referred to as actors.

The following factors should be kept in mind when constructing a use case diagram.

- As a use case, functionalities will be represented.
- Actors.
- Relationships between use cases and actors.

Use Cases
A use case is a written depiction of how visitors will execute tasks on your website. From the standpoint of a user, it defines how a system responds to a request. Each use case is characterized by a sequence of basic actions that start with the user's goal and finish when that goal is achieved.

Graphical Representation
Use cases are represented by an oval shape.



The following is a more precise analysis of a use case:

- A pattern of behavior displayed by the system.
- A series of related transactions performed by an actor as well as the system.
- Delivering something useful to the actor.

You can utilize use cases to document system requirements, connect with top users and domain experts, and test the system. Looking at the actors and defining what they can accomplish with the system is the greatest way to uncover use cases.

Flow of events
A sequence of times can be thought of as a collection of interactions (or opportunities) carried out by the system. They provide daily point-by-point details, published in terms of what the framework can do rather than whether the framework performs the task.

- When and how the employment case begins and ends.
- Interactions between the use case and the actor.
- Information required by the employment case.
- The employment case's normal sequence of events.
- A different or exceptional flow.

Construction of Use case

The behavior of the framework is graphically illustrated in use-case outlines. These graphs show how the framework is utilized at a high level, when seen through the perspective of an untouchable (actor). A utilization case graph can depict all or some of a framework's work instances.

A use-case diagram may include the following elements:

- Actors.
- Use cases.

Relationships in use cases

Active relationships, also known as behavioral relationships, are a type of interaction that is frequently shown in use case diagrams. The four main types of behavioral relationships are inclusion, communication, generalization and extension.

1) Communicates

The behavioral relationship communicates connects an actor to a use case. Remember that the purpose of the use case is to provide some sort of benefit to the system's actor. As a result, it is critical to document these interactions between actors and use cases. A line with no arrowheads connects an actor to a use case.

2) Includes

The includes relationship (also known as the uses relationship) describes a situation in which a use case contains behavior that is shared by multiple use cases. To put it another way, the common use case is included in the other use cases. The included relationship is indicated by a dotted arrow pointing to the common use case.

3) Extends

The extended connection describes when one use case contains behavior that allows a new use case to handle a variant or exception to the basic use case. A distinct use case handles exceptions to the basic use case. The arrow connects the basic and extended use cases.

4) Generalizes

The generalized relationship indicates that one thing is more prevalent than another.
This link could be between two actors or between two use cases. The arrow points to a "thing" in UML that is more general than another "thing."

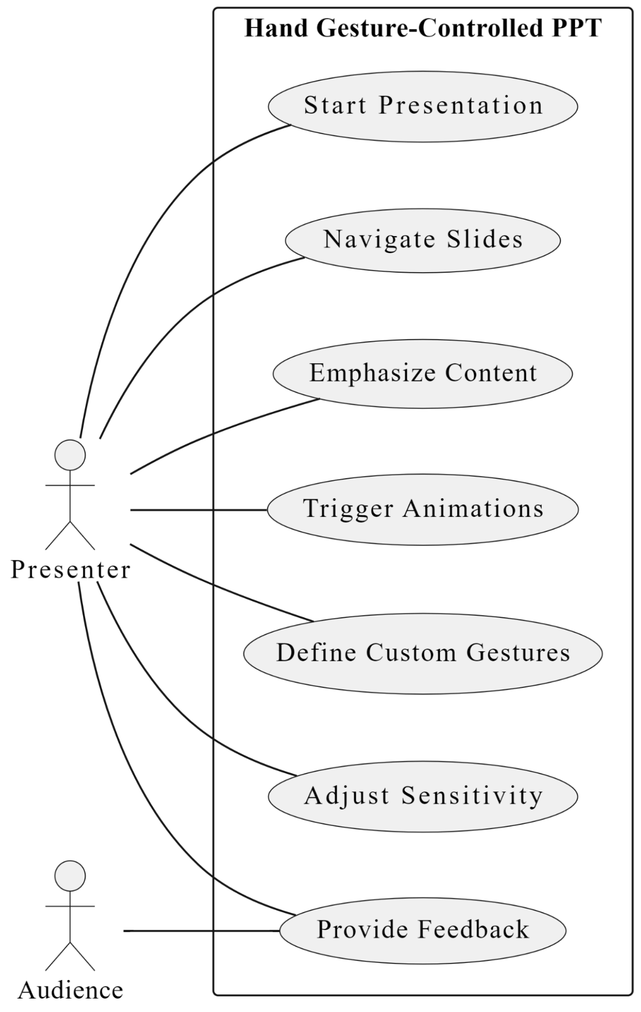


Fig 4.1.1.1: Use Case diagram for the application

:

Actors:

Presenter (P): Represents the user or presenter interacting with the Hand Gesture-Controlled PPT system.

Audience (A): Represents the audience or observers of the presentation.

Use Cases:

- Within the "Hand Gesture-Controlled PPT" box:
- Start Presentation
- Navigate Slides
- Emphasize Content

- Trigger Animations
- Define Custom Gestures
- Adjust Sensitivity
- Provide Feedback

Connections:

Actor-Use Case Connections:

- The "Presenter" actor (P) is associated with all the use cases within the system.
- The "Audience" actor (A) is linked to the "Provide Feedback" use case, indicating their interaction.

Diagram Structure:

- **Layout:** The diagram is organized in a left-to-right direction.

- **System Box:** The box "Hand Gesture-Controlled PPT" encapsulates all the system's use cases and functionalities.

4.1.2 SEQUENCE DIAGRAM

Because it illustrates how a group of items interact with one another, a sequence diagram is a form of interaction diagram. These diagrams are used by software engineers and businesspeople to comprehend the requirements for a new system or to document a current process. Sequence diagrams are sometimes known as event diagrams or event scenarios. Sequence diagrams can be useful as a reference for businesses and other organizations. Make the diagram to show:

- Describe the specifics of a UML use case.
- Create a model of the logic of a complex procedure, function, or operation.
- Examine how objects and components interact with one another in order to complete a process.
- Plan and comprehend the specific functionality of a current or future scenario. The following scenarios lend themselves well to the use of a sequence diagram:

A usage scenario is a diagram that shows how your technology might be utilized in the future. It's an excellent approach to make sure you've thought through every possible system usage situation.

Method logic:
A UML sequence diagram can be used to study the logic of any function, method, or complex process, just as it can be used to examine the rationale of a use case. If you view a service to be a high-level method utilized by several customers, a sequence diagram is a fantastic approach to map out service logic.

Object:
An object has a state, a lead, and a personality. The structure and direction of objects that are, for all intents and purposes, indistinguishable are depicted in their fundamental class. Each object in a diagram represents a specific instance of a class. An order case is an object that is not named.

Message:
A message is the exchange of information between two articles that causes an event to occur. A message transmits information from the source point of control convergence to the objective point of control convergence.

Link:
An existing association between two objects, including class, implying that there is an association between their opposing classes. If an object associates with itself, use the image's hover adjustment.

Lifeline:
It reflects the passage of time as it goes downward. The events that occur consecutively to an object during the monitored process are depicted by this dashed vertical line. A designated rectangle shape or an actor symbol could be the starting point for a lifeline.

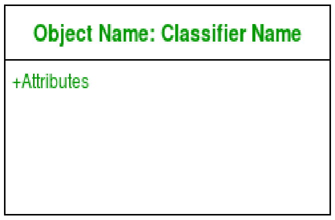
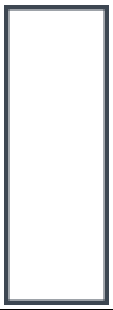
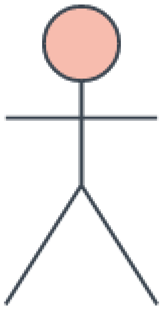
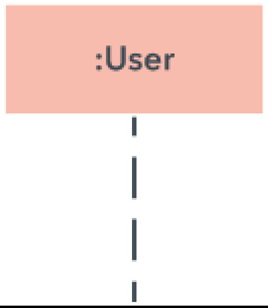
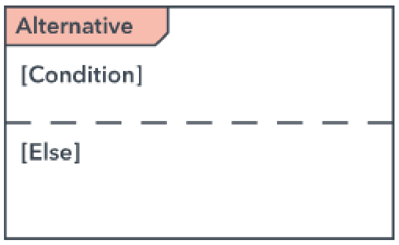
Actor:
Entities that interact with the system or are external to it are shown.

Synchronous message:
This is represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to queries before proceeding. Both the call and the response should be depicted in the diagram.

Asynchronous message:
A solid line with a lined arrowhead is used to represent this. Asynchronous messages do not necessitate a response before the sender can proceed. The diagram should only include the call.

Delete message:
An X follows a solid line with a solid arrowhead. This message has the effect of causing an object to be destroyed.

Sequence diagram components:

Name	Description	Symbol
Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.	
The activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.	
Actor symbol	Shows entities that interact with or are external to the system.	
Lifeline symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labelled rectangle shape or an actor symbol.	
Alternative symbol	Symbolises a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labelled rectangle shape with a dashed line inside.	

Messag e symbol	This symbol is used when a sender needs to send a message.	⟶
Reply messag e symbol	Represented by a dashed line with a lined arrowhead, these messages are replies to calls.	⇠

Table 4.1.2.1: Sequence Diagrams Symbols

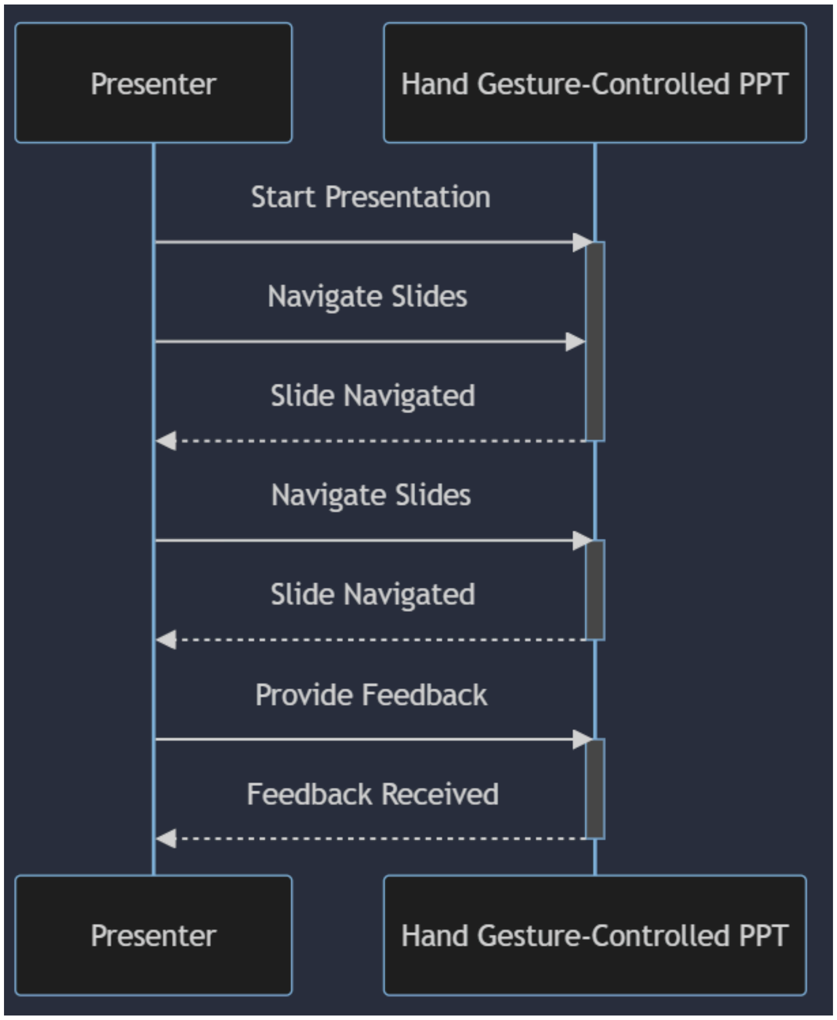


Fig 4.1.2.1: Sequence Diagram for the application

The sequence diagram for the Hand Gesture-Controlled PPT system intricately depicts the Presenter's interactions with the system, showcasing the seamless flow of slide navigation through hand gestures. Starting with the initiation of the presentation, the diagram illustrates the iterative process of navigating slides, where the system promptly acknowledges each slide change.

4.1.3 ACTIVITY DIAGRAM

An activity diagram is a flowchart that displays the movement of information from one action to the next. A system operation can be used to describe the activity.

From one operation to the next, the control flow is guided. In nature, this flow might be sequential, branching, or concurrent. By employing numerous parts such as join, fork and so on, activity diagrams cope with all sorts of flow control.

Activity diagrams provide the same basic functions as the other four diagrams. It captures the dynamic behavior of the system. The other four diagrams depict message flow from one item to the next, whereas the activity diagram depict. A specific system operation is referred to as an activity. It doesn't show any communication flow from one activity to the next. The phrases activity diagrams and flowcharts are often used interchangeably. Although the diagrams resemble flowcharts, they are not.

Notations

Initial point or start point

A small, filled circle, followed by an arrow, represents the beginning action state or starting point for any activity diagram. Make sure the start point of an activity diagram with swimlanes is in the top left corner of the first column.

Activity or Action state

An action state is a representation of an object's non-interruptible action. You can make an action state in SmartDraw by sketching a rectangle with rounded corners.

Action flow

Transitions from one action state to another are depicted by action flows, also known as edges and routes. An arrowed line is commonly used to depict them.

Decisions and branching

A diamond signifies a multiple-choice decision. Place a diamond between the two activities when one requires a decision before moving on to the next. A condition or guard expression should be used to label the outgoing alternates. One of the paths can also be labeled "else."

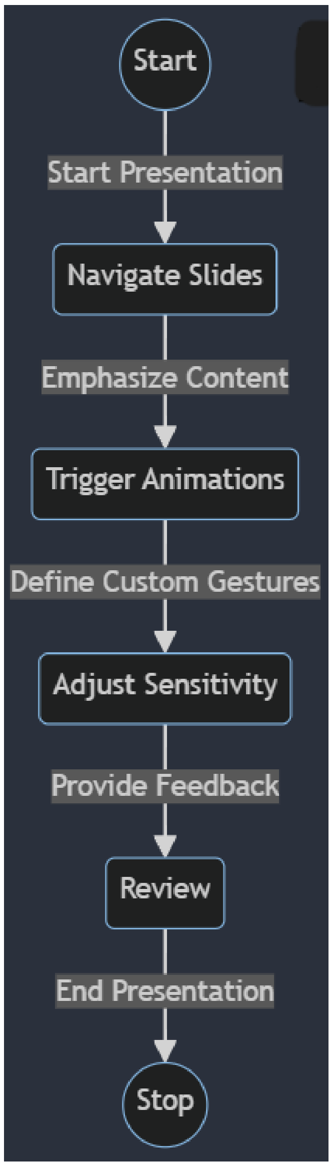


Fig 4.1.3.1: Activity Diagram for the system

5. PROPOSED SYSTEM

5.1 METHODOLOGY

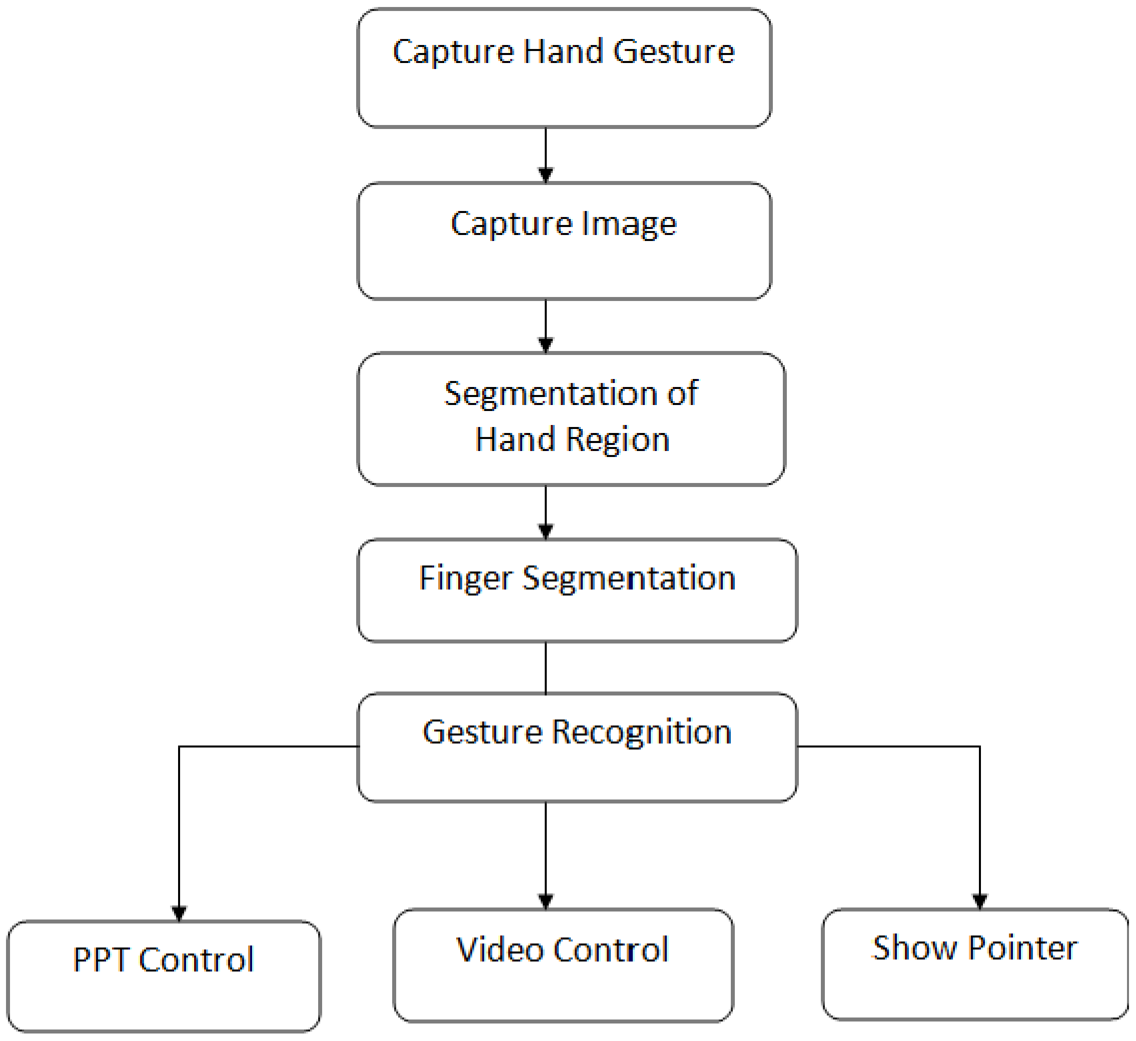


Fig. 5.1.1 Methodology

Data Set:

The "MediaPipe Hands" dataset, comprising 40,000+ annotated hand images, is curated for training hand tracking and gesture recognition models within MediaPipe. It offers diverse hand poses, lighting, and angles, annotated with precise landmark coordinates. These annotations aid in training machine learning models for accurate real-time hand movement detection. This dataset's variability ensures robust model development for interpreting various gestures in real-world scenarios.

Hand Landmark:

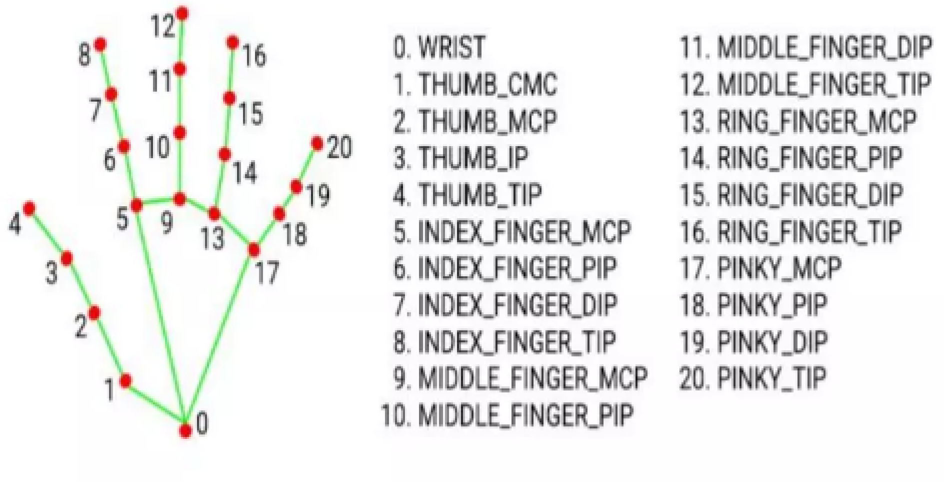


Fig 5.1.2: Mediapipe Hand Landmark

Accomplishes accurate crucial point clustering of 21 main points with only a 3D touch coordinates that is done within the identified hand areas and immediately generates the coordinates predictor that is a representation of hand landmarks within MediaPipe. Every touch of a landmark had already coordinate is constituted of x, y, and z in which x and y have been adjusted to [0.0, 1.0] besides image width and length, while z portrayal the complexity of landmark. A depth of the ancestral landmark, which is located at the wrist. The value decreases the more away the landmark is from the camera.

Model:

A machine learning model is defined as a mathematical representation of the output of th*e training process.* Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.

Train and Test:

Machine Learning is one of the booming technologies across the world that enables computers/machines to turn a huge amount of data into predictions. However, these predictions highly depend on the quality of the data, and if we are not using the right data for our model, then it will not generate the expected result. In machine learning projects, we generally divide the original dataset into training data and test data. We train our model over a subset of the original dataset, i.e., the training dataset, and then evaluate whether it can generalize well to the new or unseen dataset or test set. Therefore, train and test datasets are the two key concepts of machine learning, where the training dataset is used to fit the model, and the test dataset is used to evaluate the model.

5.2 FUNCTIONALITIES

5.2.1 Slide Navigation: Enable users to switch between slides using specific hand gestures, such as swiping left or right for backward or forward movement.

5.2.2 Emphasizing Content: Allow users to highlight or emphasize specific content on a slide using gestures to zoom in, underline, or enlarge certain elements.

5.2.3 Defining Custom Gestures: Offer the capability to customize and assign unique hand gestures for specific actions, allowing users to personalize control.

5.2.4 Providing Feedback: Include mechanisms for users to provide feedback, facilitating system improvement and customization based on user experience.

5.3 ADVANTAGES OF PROPOSED SYSTEM:

- **Intuitive Interaction:** Enables presenters to control presentations seamlessly through natural hand gestures, eliminating the need for traditional input devices and enhancing ease of use.

- **Enhanced Engagement:** Offers advanced functionalities beyond basic slide navigation, allowing for dynamic content emphasis, animation triggers, and personalized interactions, fostering greater audience engagement.

- **Accessibility and Inclusivity:** Empowers individuals with disabilities or mobility impairments to deliver presentations independently, promoting inclusivity and accessibility in public speaking scenarios.

- **Customization:** Allows users to define and customize their own hand gestures for specific actions, offering a personalized and adaptable interaction experience.

- **Compatibility and Integration:** Designed to integrate with various presentation platforms beyond Microsoft PowerPoint, extending its compatibility to other software like Google Slides or Keynote, enhancing its versatility.

- **Technological Advancements:** Leverages sophisticated technologies like machine learning, computer vision, and gesture recognition libraries (e.g., OpenCV and MediaPipe) for accurate gesture interpretation and responsiveness.

6. CODING AND IMPLEMENTATIONS

6.1 DATASET:

"MediaPipe Hands" dataset, which contains over 40,000 annotated hand images. This dataset is specifically designed for training and validating hand tracking and gesture recognition models within the MediaPipe framework. The dataset comprises a vast collection of hand images captured from various angles, lighting conditions, and hand poses. Each image is annotated with corresponding hand landmarks or keypoints, providing precise coordinates for different parts of the hand, such as fingertips, knuckles, and palm edges. These annotations enable the training of machine learning models to accurately detect and track hand movements in real-time. The dataset's diversity in hand poses and variations helps in building robust models capable of handling different gestures and orientations commonly encountered in real-world scenarios.

6.2 IMPLEMENTATION:

The code imports necessary Python modules for computer vision and hand tracking. `os` handles system interactions, `cv2` enables image processing, and `HandDetector` from `cvzone.HandTrackingModule` supports hand detection. Additionally, `numpy` (imported as `np`) offers numerical computation capabilities essential for various tasks, including computer vision.

Code:

```
import os
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np

#variables
width,height=1280,720
folderPath="PPT SLIDES"

#Camera Setup
cap=cv2.VideoCapture(0,cv2.CAP_DSHOW)
cap.set(cv2.CAP_PROP_FRAME_WIDTH,1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT,720)
cap.set(3,width)
cap.set(4,height)


# get the list of presentation images
pathImages=sorted(os.listdir(folderPath),key=len) #key=len means sorting
```

```
according to length
print(pathImages) # ['Slide1.PNG', 'Slide2.PNG', 'Slide3.PNG', 'Slide4.PNG',
'Slide5.PNG', 'Slide6.PNG', 'Slide7.PNG', 'Slide8.PNG', 'Slide9.PNG',
'Slide10.PNG']

# variables
imgNumber= 0 # for going front and back slides
heightsmall , widthsmall= int(120*1),int(213*1) # dividing
width,height=1280,720 by 6 each, we get 120,213 height,width such that it is
the height,width for the webcam on the slide on top write(not accurately)
gestureThreshold=300 # if values is above 300 then detect the hand
buttonPressed=False #used to slow down the movement of the slides slowly
insteadly of rapidly changing
buttonCounter=0
buttonDelay=30
draw= [[]]
drawNumber=-1
drawStart=False

#Hand Detector
detector= HandDetector(detectionCon=0.8,maxHands=1) # detectionCon=0.8
implies if 80% confidence that it is hand then work



while True:
   #import images
   success,img=cap.read()
   img=cv2.flip(img,1) # flip is used that if right hand moves towards left in
real life then in webcam it must move towards left only not right , 1 ->
horizontal flipping,0-> vertical flipping
   pathFullImg=os.path.join(folderPath,pathImages[imgNumber]) #
pathFullImg shows us the current slide imgNumber is pointing from
pathImages var which is sorted
   imgCurrent= cv2.imread(pathFullImg)
   print(drawNumber)

   # Find the hand and its landmarks
   hands,img=detector.findHands(img) # flipType=False is kept such that if
right hand is on the webcam then right must be displayed on the webcam not
left after flipping is used
   # Draw Gesture Threshold line
   cv2.line(img,(0,gestureThreshold),(width,gestureThreshold),(0,255,0),5)
```

```
    if hands and buttonPressed is False: # if hand is detected in the webcam
        hand=hands[0] # 0 means only 1 hand is detected as maxHands=1
        fingers=detector.fingersUp(hand) # List of which fingers are up
        cx,cy=hand["center"]
        #print(fingers)

        lmList=hand["lmList"] # List of 21 Landmark points

        #constrain values for easier drawing
        #indexFinger=lmList[8][0],lmList[8][1]  # 8-> indexfinger number ,0,1 are
the two points on the fingers
        xVal=int(np.interp(lmList[8][0],[width//2, width],[0,width])) # converting
[width//2, wslide]->[0,width] such that the pointer willl be at the right half of the
webcam for easy usage
        yVal =int(np.interp(lmList[8][1],[150,height-150],[0,height]))
        indexFinger=xVal,yVal


        if cy<=gestureThreshold:  # if hand is at the height of the face or above the
gesture line

            # Gesture 1 -left
            if fingers==[1,0,0,0,0]:
                print('left')
                buttonPressed = True  #
                if imgNumber>0:
                    draw = [[]]       #
                    drawNumber = -1  # the draw,drawNumber,drawStart is declared
again such that if we draw in current slide, then me move to next slide the
drawing will get erased
                    drawStart = False
                    imgNumber = imgNumber-1

            # Gesture 2 -right
            if fingers == [0, 0, 0, 0, 1]:
                print('right')
                buttonPressed = True
                if imgNumber <len(pathImages)-1:
                    draw = [[]]
                    drawNumber = -1
                    drawStart = False
                    imgNumber = imgNumber +1
        # Gesture 3 - Show Pointer
        if fingers==[0,1,1,0,0]:
            cv2.circle(imgCurrent,indexFinger,12,(0,0,255),cv2.FILLED)

        # Gesture 4 - Drawing Pointer
        if fingers==[0,1,0,0,0]:
            if drawStart is False:
                drawStart=True
                drawNumber=drawNumber+1
                draw.append([])

            print(drawNumber)
            draw[drawNumber].append(indexFinger)
            cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)

        else:
            drawStart=False

        # Gesture 5 - UNDO
        if fingers == [0, 1, 1, 1, 0]:
            if draw:
                    draw.pop(-1)
                    drawNumber = drawNumber - 1
                    buttonPressed=True
    else: # if no hand
        drawStart=False


    # Button Pressed Iterations
    if buttonPressed:
        buttonCounter=buttonCounter+1
        if buttonCounter>buttonDelay:
            buttonCounter=0
            buttonPressed=False

    for i in range(len(draw)):
        for j in range(len(draw[i])):
            if j!=0:
                cv2.line(imgCurrent,draw[i][j-1],draw[i][j],(0,0,200),12,)
# from draw[i-1]-> to draw[i] the line must be drawn

    #Adding Webcam image on the slides
    imgSmall=cv2.resize(img,(widthsmall,heightsmall))
hslide,wslide,channel=imgCurrent.shape # gives height,width of
slides
imgCurrent[0:heightsmall,wslide-widthsmall:wslide]=imgSmall #
putting the webcam on the slide in the top right conner

cv2.imshow("window",img)
cv2.imshow("Slides", imgCurrent)

key=cv2.waitKey(1)
if key==ord('q'):
  break
```

7. TESTING

In machine learning, testing is mainly used to validate raw data and check the ML model's performance. The main objectives of testing machine learning models are:

- Quality Assurance
- Detect bugs and flaws

Once your machine learning model is built (with your training data), you need unseen data test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

Testing data has two main criteria. It should:

- Represent the actual dataset
- Be large enough to generate meaningful predictions

7.1 TYPES OF TESTING

7.1.1 MANUAL TESTING

Manual Testing is a type of software testing in which test cases are executed manually by a tester without using any automated tools. The purpose of Manual Testing is to identify the bugs, issues, and defects in the software application. Manual software testing is the most primitive technique of all testing types and it helps to find critical bugs in the software application.

Any new application must be manually tested before its testing can be automated. Manual Software Testing requires more effort but is necessary to check automation feasibility. Manual Testing concepts does not require knowledge of any testing tool. One of the Software Testing Fundamental is "**100% Automation is not possible "**. This makes Manual Testing imperative.

7.1.2 AUTOMATED TESTING

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

The automation testing software can also enter test data into the System Under Test, compare expected and actual results and generate detailed test reports. Software Test Automation demands considerable investments of money and resources.

7.2 SOFTWARE TESTING METHODS

7.2.1 BLACK BOX TESTING

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer. In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

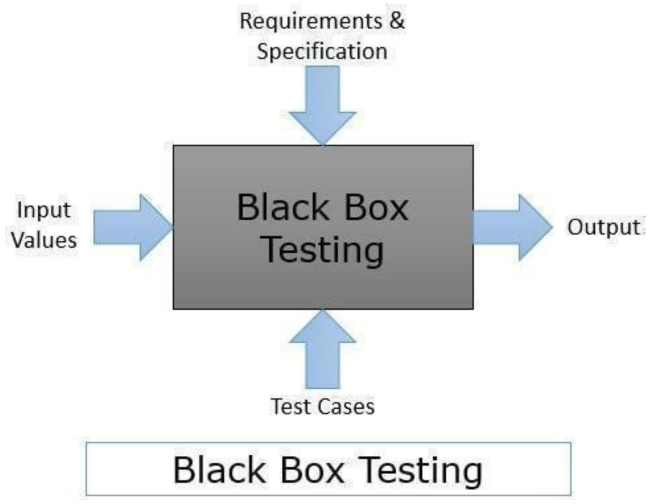


Fig. 7.2.1.1 Black Box Testing

7.2.2 GRAY BOX TESTING

Gray box testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a **combination of black box and white box testing** because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

Gray box testing commonly identifies context-specific errors that belong to web systems. For example; while testing, if tester encounters any defect, then he makes changes in code to resolve the defect and then test it again in real time. It concentrates on all the layers of any complex software system to increase testing coverage. It gives the ability to test both presentation layer as well as internal coding structure. It is primarily used in integration testing and penetration testing.

This testing technique is a combination of Black box testing and White box testing. In Black box testing, the tester does not have any knowledge about the code. They have information for what will be the output for the given input. In White box testing, the tester has complete knowledge about the code. Grey box testers have knowledge of the code, but not completely.

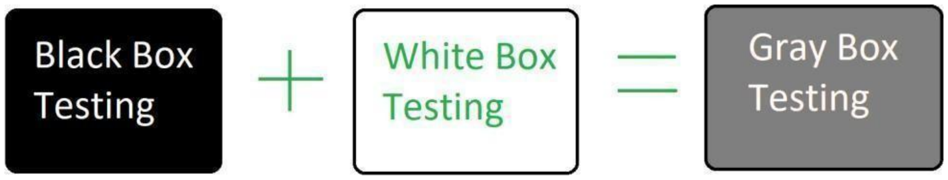


Fig. 7.2.2.1 Grey Box Testing

7.2.3 WHITE BOX TESTING

White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems. White box testing is an essential part of automated build processes in a modern Continuous Integration/Continuous Delivery (CI/CD) development pipeline. White box testing is often referenced in the context of Static Application Security Testing (SAST), an approach that checks source code or binaries automatically and provides feedback on bugs and possible vulnerabilities. White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

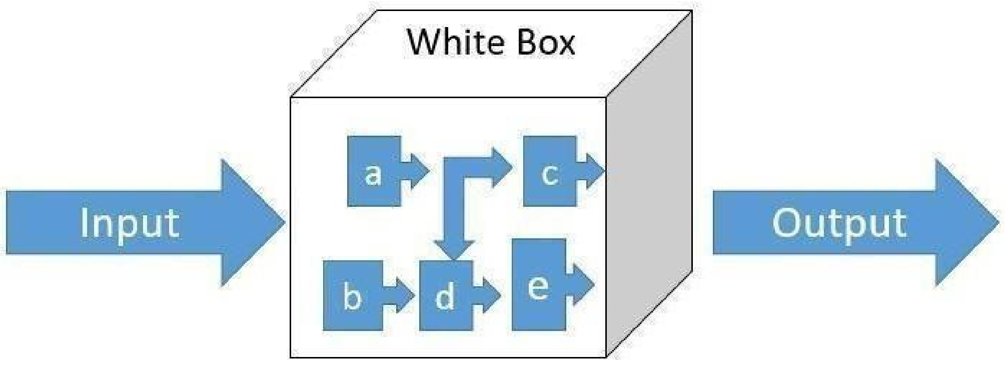


Fig. 7.2.3.1 White Box Testing

7.3 TESTING LEVELS

7.3.1 NON-FUNCTIONAL TESTING

Non-functional testing is a type of software testing to test non-functional parameters such as reliability, load test, performance and accountability of the software. The primary purpose of non-functional testing is to test the reading speed of the software system as per non-functional parameters. The parameters of non-functional testing are never tested before the functional testing. Non-functional testing is also very important as functional testing because it plays a crucial role in customer satisfaction.

7.3.1.1 PERFORMANCE TESTING

Performance testing is a form of software testing that focuses on how a system running the system performs under a particular load. This is not about finding software bugs or defects. Different performance testing types measure according to benchmarks and standards. Performance testing gives developers the diagnostic information they need to eliminate bottlenecks.

7.3.1.2 STRESS TESTING

Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.

7.3.1.3 SECURITY TESTING

Security Testing is a type of Software Testing that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders. It ensures that the software system and application are free from any threats or risks that can cause a loss. Security testing of any system is focused on finding all possible loopholes and weaknesses of the system which might result in the loss of information or repute of the organization.

7.3.1.4 PORTABILITY TESTING

Portability Testing is one of Software Testing which is carried out to determine the degree of ease or difficulty to which a software application can be effectively and efficiently transferred from one hardware, software or environment to another one. The results of portability testing are measurements of how easily the software component or application will be integrated into the environment and then these results will be compared to the non-functional requirement of portability of the software system.

7.3.1.5 USABILITY TESTING

Usability Testing, also known as User Experience (UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software applications to expose usability defects. Usability testing mainly focuses on the user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives. This testing is recommended during the initial design phase of SDLC, which gives more visibility on the expectations of the users.

7.3.2 FUNCTIONAL TESTING

It is a type of software testing which is used to verify the functionality of the software application, whether the function is working according to the requirement specification. In functional testing, each function is tested by giving the value, determining the output, and verifying the actual output with the expected value. Functional testing performed as black-box testing which is presented to confirm that the functionality of an application or system behaves as we are expecting. It is done to verify the functionality of the application. Functional testing is also called black-box testing.

7.3.2.1 INTEGRATION TESTING

Integration testing is done to test the modules/components when integrated to verify that they work as expected i.e. to test the modules which are working fine individually and do not have issues when integrated. The main function or goal of this testing is to test the interfaces between the units/modules. The individual modules are first tested in isolation. Once the modules are unit tested, they are integrated one by one, till all the modules are integrated, to check the combinational behavior, and validate whether the requirements are implemented correctly.

7.3.2.2 REGRESSION TESTING

Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but a full or partial selection of already executed test cases that are re-executed to ensure existing functionalities work fine. This testing is done to ensure that new code changes do not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

7.3.2.3 UNIT TESTING

Unit Testing is a software testing technique by means of which individual units of software.
i.e. a group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure.

7.3.2.4 ALPHA TESTING

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the **user acceptance testing**. This is referred to as alpha testing only because it is done early on, near the end of the development of the software. Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.

7.3.2.5 BETA TESTING

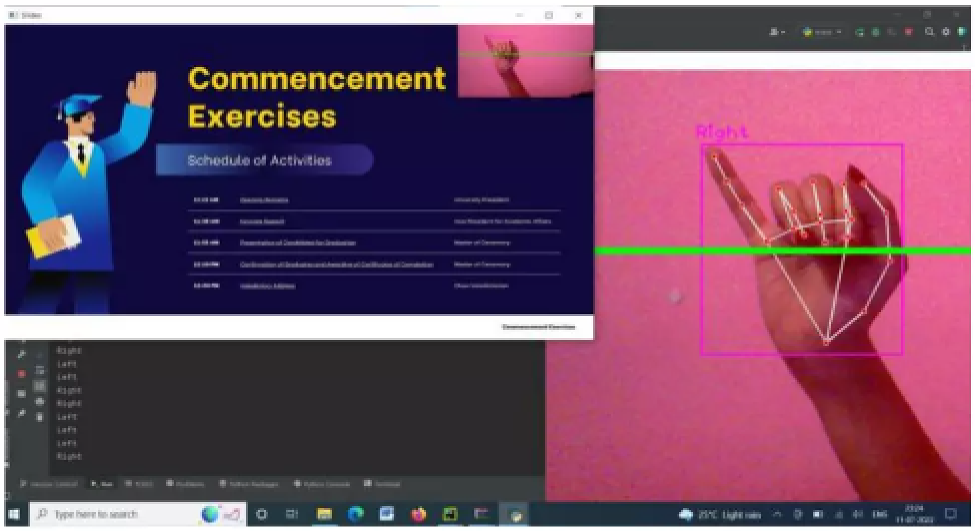
Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of **User Acceptance Testing**. A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks, and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers. One of the major advantages of beta testing is direct feedback from customers.

7.4 TEST CASES

Sl.no	**Testcase**	**Expected Result**	**Actual Result**	**Pass/ Fail**
1.	Navigate slides forward	Slide moves to the next in the sequence	Move on to the next slide	PASS
2.	Navigate slides backward	Slide moves to the previous in the sequence	Move on to the previous slide	PASS
3.	Showing Pointer on the Slide	Displaying the pointer on the slide	Getting a Pointer on Slide	PASS
4.	Draw Using Pointer on slide	Sketch on a slide with a pointer	Drawing on Slide	PASS
5.	Erase the Drawing on slide	Delete the drawing from the slide	Drawing Erased	PASS

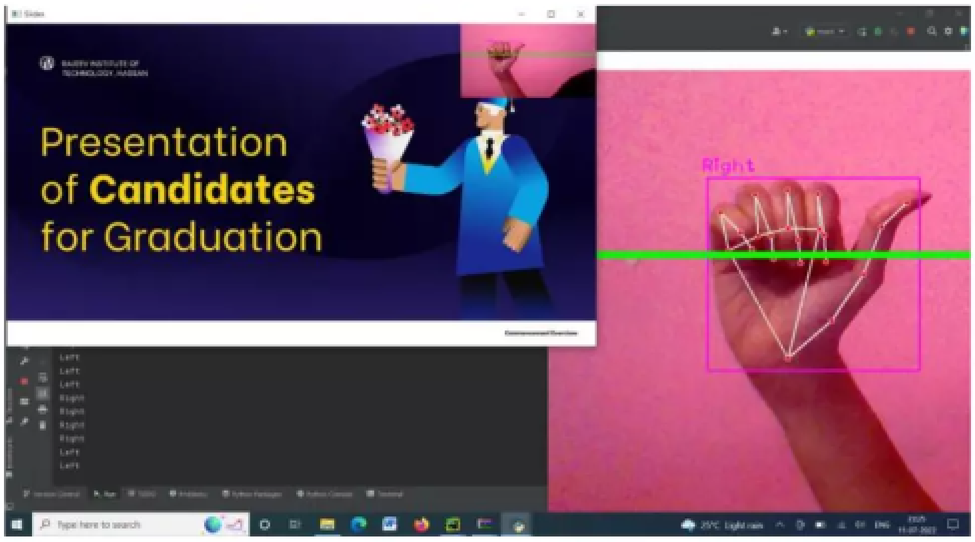
Table 7.4.1: Testcases

8. RESULTS



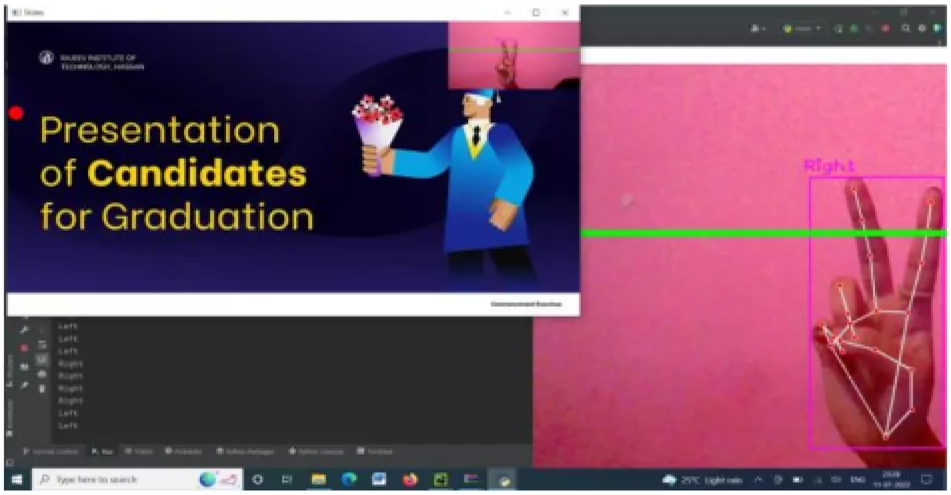
```
Right
Left
Left
Right
Right
Left
Left
Left
Right
```

Fig:8.1 Hand Gesture To move on to next slide



```
Left
Left
Left
Right
Right
Right
Right
Left
Left
```

Fig:8.2 Hand Gesture for going back to previous slide



```
Left
Left
Left
Right
Right
Right
Right
Left
Left
```

Fig 8.3 :Getting A pointer on slide

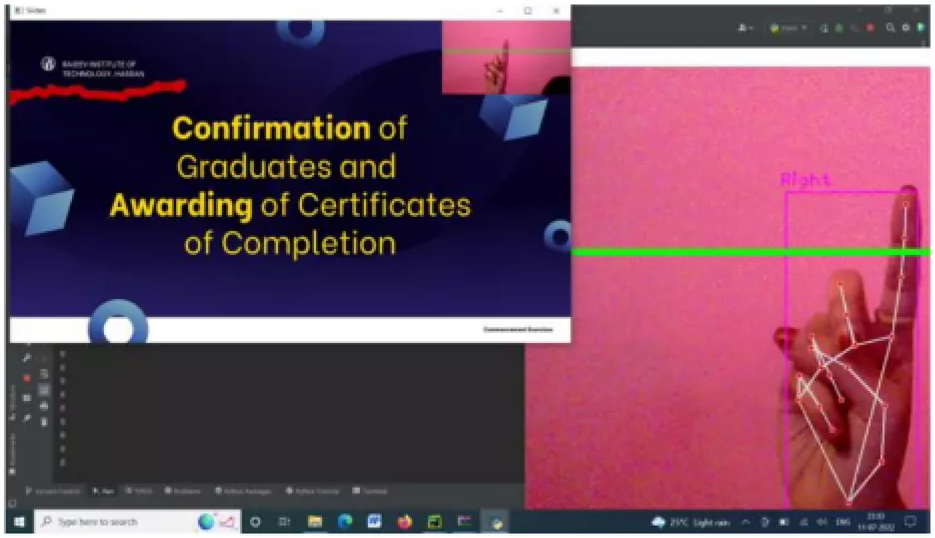

Confirmation of Graduates and Awarding of Certificates of Completion
Right

Fig 8.4: Draw using That Pointer

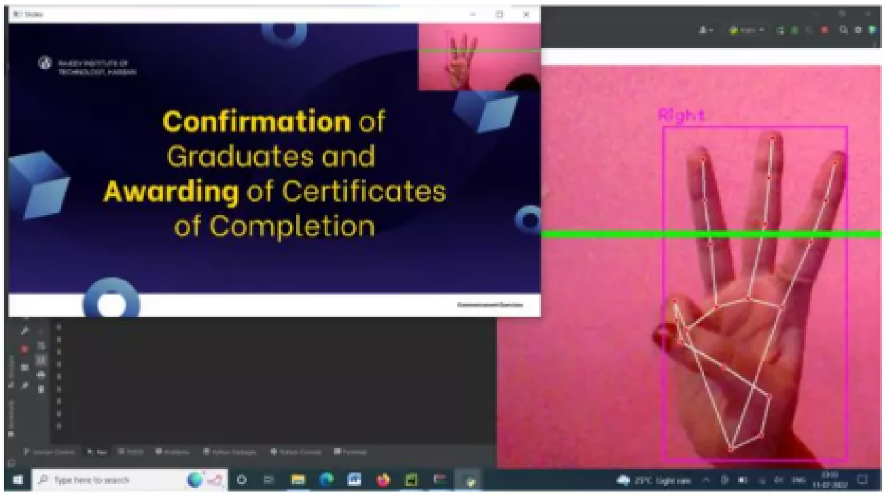

Confirmation of Graduates and Awarding of Certificates of Completion
Right

Fig 8.5: Erase the Drawing on Slide

9. CONCLUSION AND FURTHER WORK

- Technological Leap: The Hand Gesture-Controlled PPT system marks a significant advancement in presentation dynamics, offering an intuitive and interactive means for presenters to engage with their content.

- Empowered Presenters: Integration of machine learning, computer vision, and gesture recognition empowers presenters with dynamic functionalities, enhancing audience engagement and inclusivity.

- Personalization and Accessibility: The system's potential for personalized interactions, integration with various platforms, and accessibility for individuals with disabilities signifies a transformative shift in presentation dynamics.

- Gesture Recognition Refinement: Enhance accuracy and diversity in recognized gestures to broaden interaction possibilities and ensure more precise control.

- Compatibility Enhancement: Refine system compatibility to seamlessly integrate with a wider array of presentation software for increased versatility.

- Hardware Optimization: Explore optimized hardware solutions to enhance accessibility and usability, potentially minimizing reliance on specific devices.

- Real-Time Collaboration Features: Research and implement real-time collaboration features to facilitate interactive presentations involving multiple presenters or participants.

- User-Centric Refinement: Conduct user studies to gather feedback and insights, refining the system's usability and tailoring it to meet diverse presenter needs more effectively.

10. REFERENCES

1. https://www.slideshare.net/irjetjournal/smart-presentation-control-by-hand-gestures-using-computer-vision-and-googles-mediapipe
2. https://ieeexplore.ieee.org/document/8282654
3. https://www.slideserve.com/edwardmoore/hand-gesture-recognition-powerpoint-ppt-presentation
4. https://ieeexplore.ieee.org/document/10125869
5. https://learningmsp430.wordpress.com/2014/11/16/gesture-control-for-powerpoint-presentation/