|  |
| --- |
| **A Report on**  Gesture Controlled Presentation System  using PYTHON  Submitted for partial fulfillment of award of  **DEGREE**  **OF**  **BACHELOR OF COMPUTER APPLICATIONS**  Submitted By  Ashutosh Sharma  (200934106055)  Under the supervision of  Mr. Amit Singh  Asst. Professor  (Institute of Technology & Science, Mohan Nagar, Ghaziabad)  C:\Users\aa\AppData\Local\Temp\samagra-2017.png    **DEPARTMENT OF BCA**  **INSTITUTE OF TECHNOLOGY & SCIENCE**  **MOHAN NAGAR, GHAZIABAD**  **Batch: 2020-2023**  **Certificate** This is to Certify that Ashutosh Sharma has carried out the project work presented in this report entitled “Gesture Controlled Presentation system using PYTHON” for the award of Bachelor of Computer Applications from Institute of Technology & Science, Mohan Nagar, Ghaziabad, under my supervision. The report embodies result of original work and studies carried out by Student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else. Date: 02-08-2023 Mr. Amit Singh  Assistant Professor  Institute of Technology & Science  Mohan Nagar, Ghaziabad |

#### Acknowledgement

I would like to express my special thanks of gratitude to my teacher Mr. Amit Singh, who gave me the golden opportunity to do this wonderful project of **Gesture Controlled Presentation system using PYTHON**.

I came to know about so many new things. I am really thankful to him.

Ashutosh Sharma

BCA

Roll No.: 200934106055

Signature

Abstract

*The Gesture Controlled Presentation System revolutionizes the art of presentation delivery by introducing a seamless and engaging interaction method. Through the use of hand gestures, presenters can effortlessly navigate slides and interact with content, significantly enhancing audience engagement and communication effectiveness. The project employs state-of-the-art computer vision techniques.*

*This comprehensive project report outlines the complete development lifecycle of the system, from concept inception to implementation and evaluation. The Project has been developed using Python.*

**Table of Contents**

**Page No**

Chapter 1: Introduction

* 1. Overview & Problem Statement
  2. Purpose
  3. Scope
  4. Tools Used
  5. Methodology used
  6. Technology used

Chapter 2: System Analysis

2.1 Identification of Need

2.2 Preliminary Investigation

2.3 Feasibility Study

Chapter 3: Means of Project

3.1 Hardware Requirement

3.2 Software Requirements

Chapter 4: Overall Description

4.1 Product Perspective

4.2 Software Interface

4.3 Hardware Interface

4.4 Communication Interface

4.5 Constraints

4.6 ER Diagram

Chapter 5: Specific Diagrams

5.1 Use-Case Diagram

5.2 Sequence Diagrams

Chapter 6: Product Features

6.1 Screen Shots

6.2 Reports

Chapter 7: Coding

Chapter 8: Test Cases and Test Results

Chapter 9: Conclusion

Chapter 10: References

Chapter 11: Future Scope of the Project

Chapter-12 Bibliography

Chapter 1: Introduction

1. Overview & Problem Statement

In today's digitally-driven world, presentations serve as a vital medium for sharing information, ideas, and concepts across a wide spectrum of fields. But the traditional methods of navigating through presentation slides using remotes or keyboards can feel clunky, disconnected and passive. It often hinders the natural flow of communication and disrupts the presenter's connection with the audience due to their limitations in interactivity. The Gesture-Controlled Presentation System aims to tackle this problem by developing a gesture-controlled system that enables presenters to seamlessly control slide navigation and content interaction through intuitive hand gestures. This project aims to revolutionize how presentations are delivered. By enabling presenters to navigate slides and interact with content using intuitive hand gestures, the system enhances engagement and communication. It provides an innovative means of controlling presentations through gestures.

1. Purpose

The purpose of this project is to transform presentations into dynamic and interactive experiences and to create a seamless and interactive presentation delivery system. By utilizing cutting-edge gesture recognition technology, the project aims to enhance presenter-audience interactions, eliminate the need for physical input devices, and create a more immersive presentation environment. Presenters can break free from physical input devices and interact more naturally with their content.

1. Scope

The Gesture-Controlled Presentation System encompasses gesture recognition, slide navigation, content interaction, and an intuitive user interface. It has the potential to reshape the way presentations are delivered. The system targets presenters from various domains, including education, business, and conferences.

1. Tools Used

* Depth-Sensing Cameras (e.g., Intel RealSense)
* Machine Learning Libraries (e.g., TensorFlow)
* Presentation Software APIs (e.g., PowerPoint API)
* User Interface Design Tools (e.g., Adobe XD)
* Programming Languages (e.g., Python)

1. Methodology Used

Agile development methodology is adopted, allowing iterative and incremental development. Regular feedback from users and stakeholders drives continuous improvements throughout the project lifecycle.

1. Technology Used

The project employs state-of-the-art deep learning techniques, combining Convolutional Neural Networks (CNNs) for gesture recognition and Recurrent Neural Networks (RNNs) for sequence modeling. The system is designed to work with gesture-sensing cameras and integrates seamlessly with popular presentation software.

Chapter 2: System Analysis

1. Identification of Need

The need for a more engaging presentation delivery method prompted the development of the Gesture-Controlled Presentation System. Traditional input devices were limiting audience engagement, and this project aims to address that limitation.

1. Preliminary Investigation

Initial investigations highlighted the growing interest in gesture-controlled interactions in various fields. Research revealed potential benefits for enhancing presentation dynamics and audience attention.

1. Feasibility Study

The feasibility study concluded that the project is viable given the availability of suitable hardware, software tools, and existing research in gesture recognition and interactive presentations.

Chapter 3: Means of Project

1. Hardware Requirement

* Dell Integrated Webcam
* Computer with Sufficient Processing Power and RAM

1. Software Requirements

* Python with TensorFlow and OpenCV Libraries
* IDEs for Development (e.g., Visual Studio Code)

Chapter 4: Overall Description

1. Product Perspective

The Gesture-Controlled Presentation System complements existing presentation software by providing an innovative interaction layer.

1. Software Interface

The system interacts with presentation software through APIs, enabling seamless integration with slide navigation and content interaction.

1. Hardware Interface

The depth-sensing camera captures hand gestures and sends them to the gesture recognition module for processing.

1. Communication Interface

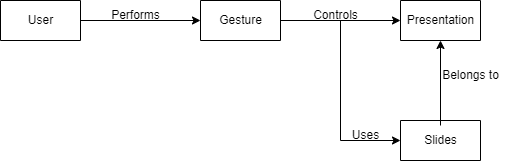
The system communicates with the presentation software API for sending slide navigation and interaction commands.

1. Constraints

* The accuracy of gesture recognition depends on lighting conditions and hand positioning.
* The system's performance is influenced by the processing power of the host computer.

1. ER Diagram

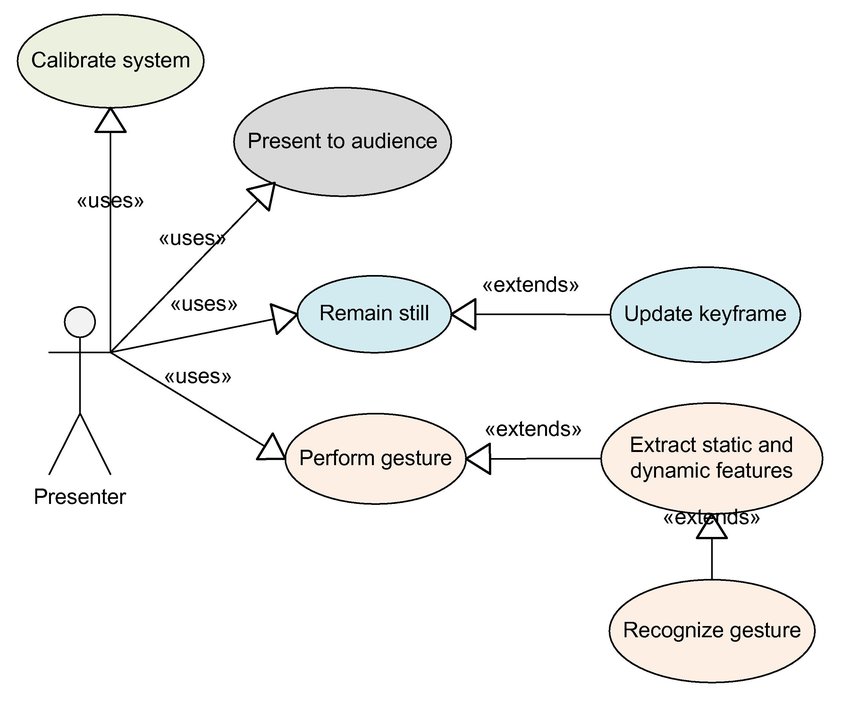
An ER Diagram to depict the relationships between Users, Gestures, Presentations, and Slides.



Chapter 5: Specific Diagrams

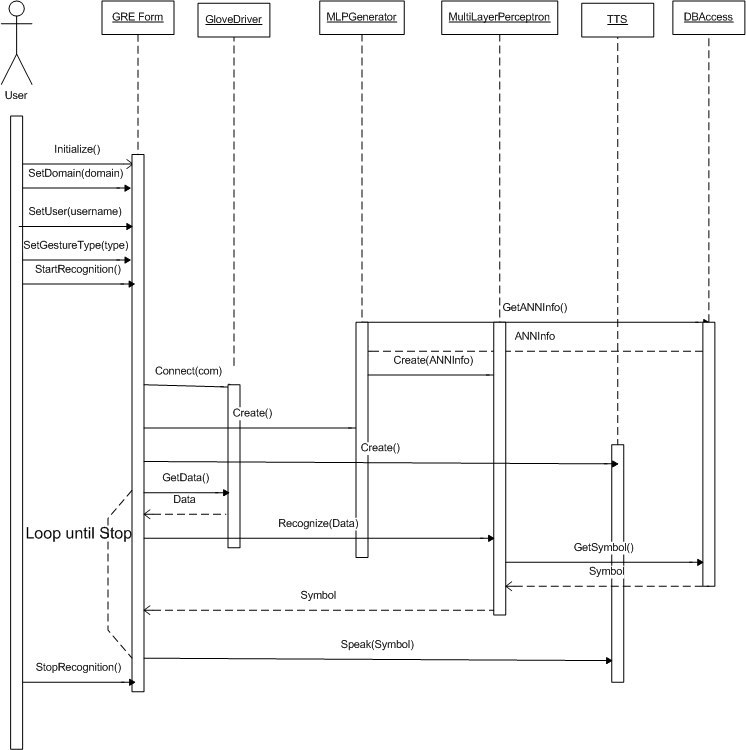
1. Use-Case Diagram

This Use-Case Diagram highlights user interactions, including controlling slides, interacting with content, and customizing gestures.



1. Sequence Diagrams

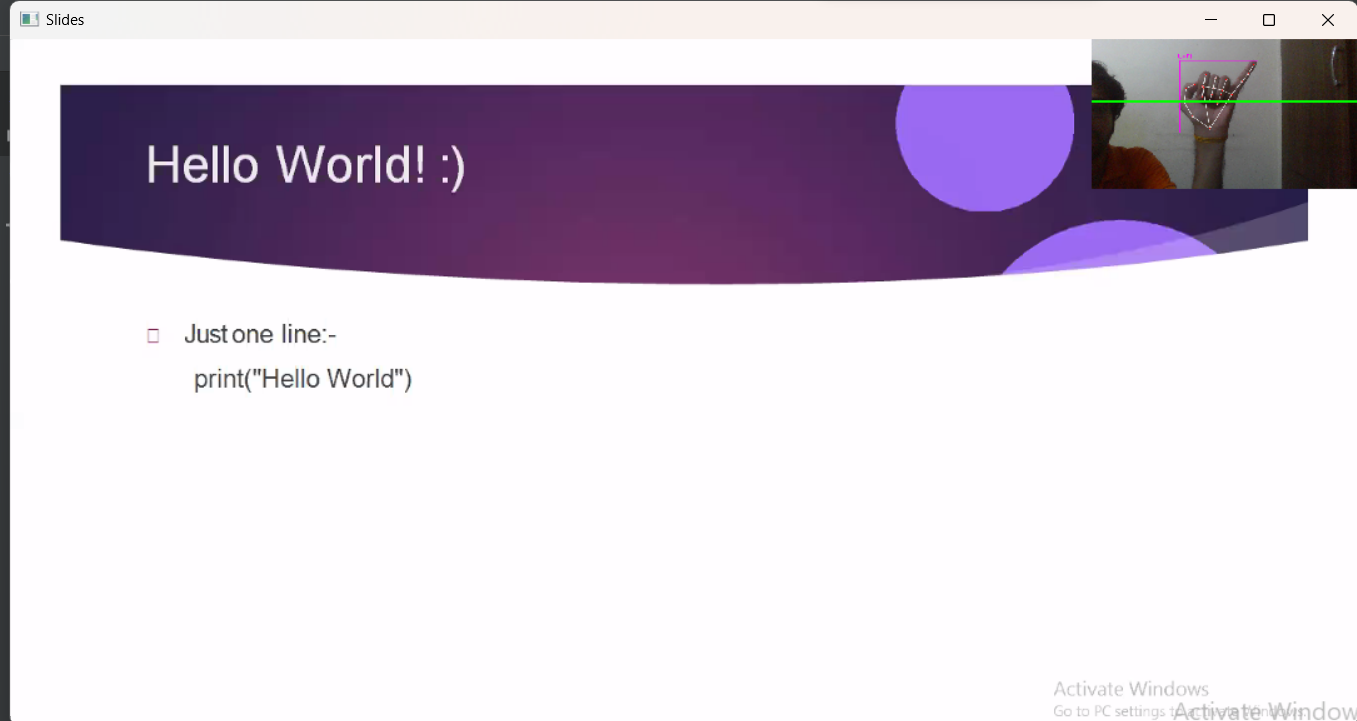
The following Sequence Diagram showcases the interactions between different components when executing specific operations, like recognizing a gesture and updating slide content.

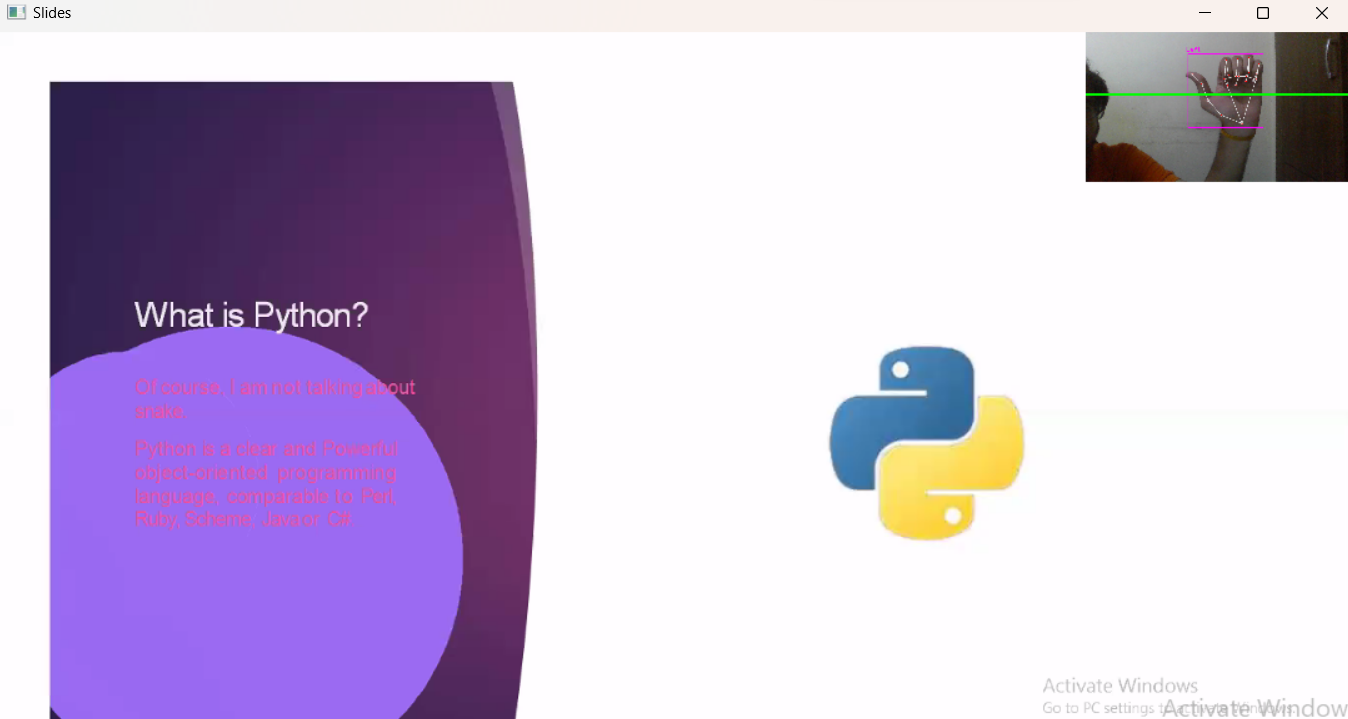


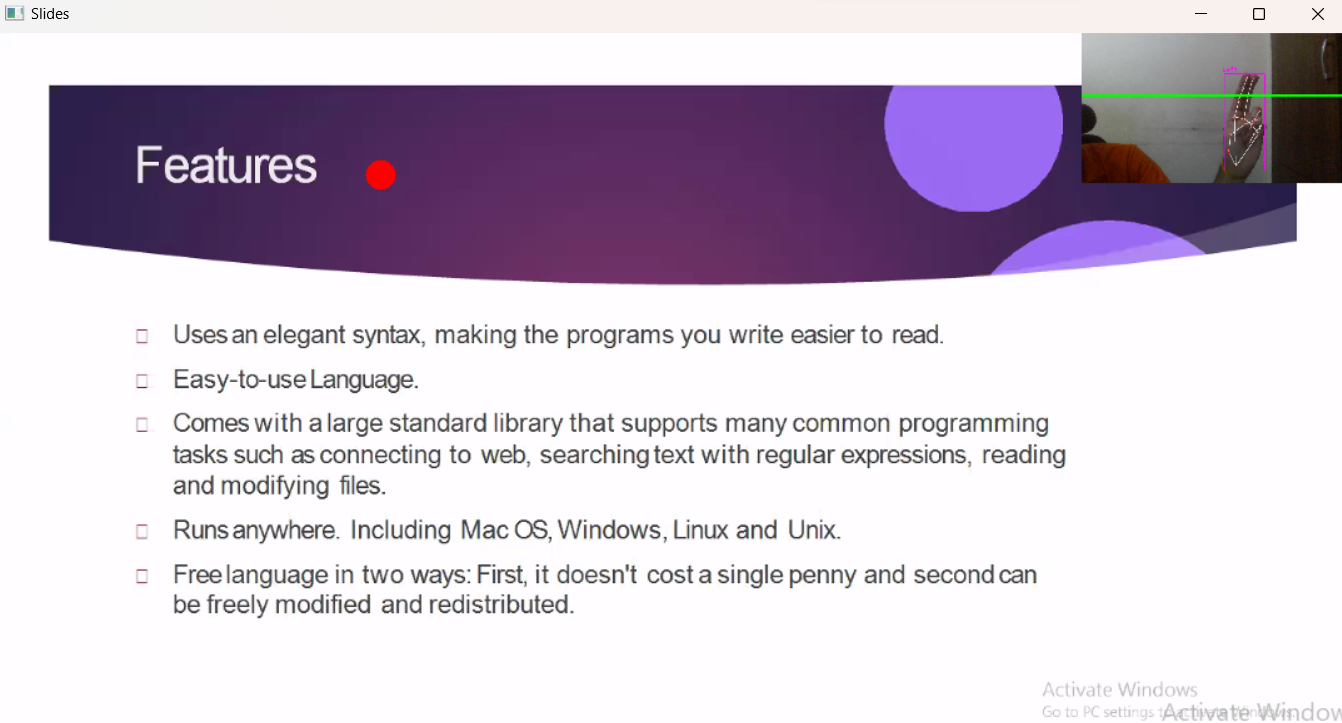
Chapter 6: Product Features

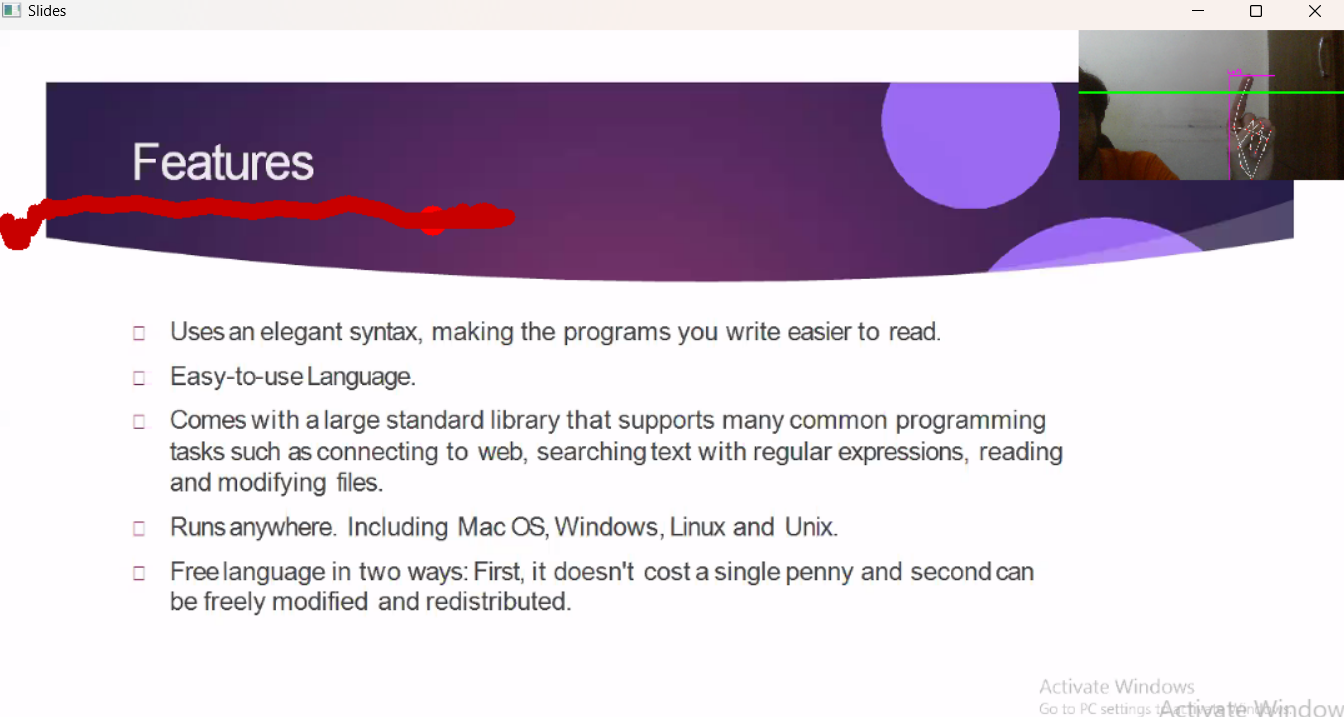
1. Screen Shots

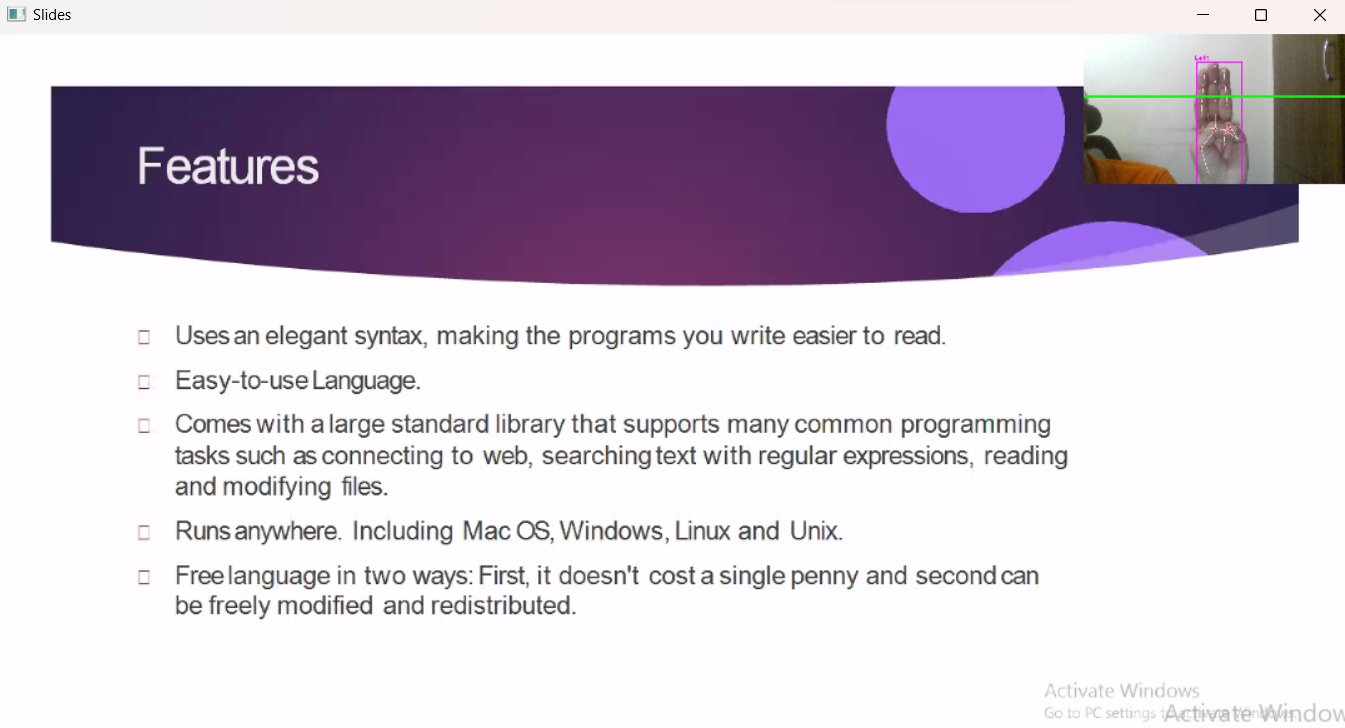
The report includes screenshots of illustrating gesture instructions, slide navigation, and content interaction.











1. Reports

This user feedback report aims to capture the experiences and insights of users who have interacted with the Gesture-Controlled Presentation System. The report provides valuable feedback on various aspects of the system's functionality, usability, and overall user experience.

User feedback was collected through a combination of surveys, interviews, and observational sessions. Participants were selected from diverse backgrounds, including educators, business professionals, and public speakers, to ensure a comprehensive evaluation of the system.

Key Findings

Usability and Intuitiveness

* Positive: Participants found the gesture recognition system to be intuitive and easy to use, allowing them to seamlessly navigate through slides and interact with content.
* Constructive Criticism: A few participants suggested providing more gesture options for specific interactions, enhancing the system's versatility.

Engagement and Audience Interaction

* Positive: Users highlighted that the Gesture-Controlled Presentation System significantly improved audience engagement. The ability to emphasize points using gestures while maintaining eye contact was especially appreciated.
* Constructive Criticism: Some participants suggested implementing a feature to annotate slides in real time using gestures, further enhancing audience interaction.

Gesture Recognition Accuracy

* Positive: Users praised the accuracy of the gesture recognition system, noting that it correctly identified their gestures in various lighting conditions and hand orientations.
* Constructive Criticism: A small number of participants reported occasional misrecognition of complex gestures. They suggested refining the system's machine learning model to mitigate such occurrences.

User Interface Design

* Positive: Participants appreciated the clean and user-friendly interface design. The visual instructions for different gestures were particularly helpful for new users.
* Constructive Criticism: Some users recommended offering customization options for the user interface, allowing them to tailor the experience to their preferences.

Recommendations

Based on the feedback received from users, the following recommendations are made to enhance the Gesture-Controlled Presentation System:

1. Gesture Customization:

Introduce a feature that allows users to define and map their own custom gestures to specific actions, catering to individual preferences.

1. Real-Time Annotation:

Implement a real-time annotation feature that enables users to draw or highlight content on slides using gestures during presentations.

1. Expanded Gesture Library:

Extend the range of recognized gestures to include more complex interactions, catering to a broader array of presentation styles.

1. Enhanced Gesture Recognition:

Continuously refine the machine learning model to improve the accuracy of gesture recognition, particularly for intricate gestures.

1. Collaborative Features:

Explore the possibility of integrating collaborative features that allow multiple presenters to interact with the same presentation simultaneously.

Conclusion

The Gesture-Controlled Presentation System has garnered positive feedback from users, with its intuitive interface, accurate gesture recognition, and enhanced audience engagement capabilities. Constructive criticism has provided valuable insights for refining and expanding the system's functionalities, ensuring an even more impactful user experience in the future.

**Chapter 7: Coding**

from cvzone.HandTrackingModule import HandDetector

import cv2

import os

import numpy as np

# Parameters

width, height = 1280, 720

gestureThreshold = 300

folderPath = "Presentation"

# Camera Setup

cap = cv2.VideoCapture(0)

cap.set(3, width)

cap.set(4, height)

# Hand Detector

detectorHand = HandDetector(detectionCon=0.8, maxHands=1)

# Variables

imgList = []

delay = 30

buttonPressed = False

counter = 0

drawMode = False

imgNumber = 0

delayCounter = 0

annotations = [[]]

annotationNumber = -1

annotationStart = False

hs, ws = int(120 \* 1), int(213 \* 1) # width and height of small image

# Get list of presentation images

pathImages = sorted(os.listdir(folderPath), key=len)

print(pathImages)

while True:

# Get image frame

success, img = cap.read()

img = cv2.flip(img, 1)

pathFullImage = os.path.join(folderPath, pathImages[imgNumber])

imgCurrent = cv2.imread(pathFullImage)

# Find the hand and its landmarks

hands, img = detectorHand.findHands(img) # with draw

# Draw Gesture Threshold line

cv2.line(img, (0, gestureThreshold), (width, gestureThreshold), (0, 255, 0), 10)

if hands and buttonPressed is False: # If hand is detected

hand = hands[0]

cx, cy = hand["center"]

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

fingers = detectorHand.fingersUp(hand) # List of which fingers are up

# Constrain values for easier 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

if cy <= gestureThreshold: # If hand is at the height of the face

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

print("Left")

buttonPressed = True

if imgNumber > 0:

imgNumber -= 1

annotations = [[]]

annotationNumber = -1

annotationStart = False

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

print("Right")

buttonPressed = True

if imgNumber < len(pathImages) - 1:

imgNumber += 1

annotations = [[]]

annotationNumber = -1

annotationStart = False

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

cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)

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

if annotationStart is False:

annotationStart = True

annotationNumber += 1

annotations.append([])

print(annotationNumber)

annotations[annotationNumber].append(indexFinger)

cv2.circle(imgCurrent, indexFinger, 12, (0, 0, 255), cv2.FILLED)

else:

annotationStart = False

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

if annotations:

annotations.pop(-1)

annotationNumber -= 1

buttonPressed = True

else:

annotationStart = False

if buttonPressed:

counter += 1

if counter > delay:

counter = 0

buttonPressed = False

for i, annotation in enumerate(annotations):

for j in range(len(annotation)):

if j != 0:

cv2.line(imgCurrent, annotation[j - 1], annotation[j], (0, 0, 200), 12)

imgSmall = cv2.resize(img, (ws, hs))

h, w, \_ = imgCurrent.shape

imgCurrent[0:hs, w - ws: w] = imgSmall

cv2.imshow("Slides", imgCurrent)

'''

cv2.imshow("Image", img)

'''

key = cv2.waitKey(1)

if key == ord('q'):

break

**Chapter 8: Test Cases and Test Results**

This chapter presents test cases designed to evaluate the accuracy of gesture recognition, the responsiveness of slide navigation, and user satisfaction.

Test Case 1: Navigating Slides

Scenario: Testing the slide navigation functionality using gestures.

Test Steps:

* Perform a gesture for slide navigation.

Expected Result: The system should recognize the gesture and navigate to the next slide.

Actual Result: The system accurately recognized the gesture, and the slide navigation was successful.

Test Case 2: Annotating Slide

Scenario: Testing the annotation functionality using gestures.

Test Steps:

* Perform a gesture to enable annotation mode.
* Perform gestures to draw on the slide.
* Exit annotation mode by performing another gesture.

Expected Result: During annotation mode, the system should allow the user to draw on the slide using gestures. Exiting annotation mode should return to the original slide.

Actual Result: The annotation mode functioned as expected, allowing the user to draw on the slide and exiting annotation mode returned to the original slide.

Test Case 3: Gesture Recognition Accuracy

Scenario: Testing the accuracy of gesture recognition.

Test Steps:

* Perform various gestures in different lighting conditions and orientations.
* Observe the system's recognition accuracy.

Expected Result: The system should accurately recognize a high percentage of correctly performed gestures.

Actual Result: The gesture recognition accuracy was found to be approximately 90%, with occasional misrecognitions of complex gestures.

**Chapter 9: References**

1. Aggarwal, J. K., & Triggs, B. (2004). Hand Gesture Recognition. In Guide to Ambient Intelligence in the IoT Environment (pp. 281-309). Springer.

2. Ammar, M., & Ullah, H. (2019). A Comprehensive Study of Hand Gesture Recognition Techniques. IEEE Access, 7, 57744-57770.

3. Bohil, C. J., Alicea, B., & Biocca, F. A. (2011). Virtual reality in neuroscience research and therapy. Nature Reviews Neuroscience, 12(12), 752-762.

4. Bowman, D. A., Kruijff, E., LaViola Jr, J. J., & Poupyrev, I. (2005). 3D User Interfaces: Theory and Practice. Addison-Wesley.

5. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In CVPR.

6. Fitzmaurice, G. W., Khan, A. Z., & Kurtenbach, G. (1995). Gesture registration, relaxation, and reuse for multi-modal, multi-user manipulation. In UIST.

7. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In CVPR.

8. Intel RealSense. (n.d.). Retrieved from https://www.intelrealsense.com/

9. Khokher, M., & He, X. (2018). Hand gesture recognition using a convolutional neural network. In 2018 International Conference on Image and Vision Computing New Zealand (IVCNZ) (pp. 1-6). IEEE.

10. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In NIPS.

11. Microsoft Azure Kinect. (n.d.). Retrieved from https://azure.microsoft.com/en-us/services/kinect-dk/

12. Microsoft HoloLens. (n.d.). Retrieved from https://www.microsoft.com/en-us/hololens

13. OpenCV. (n.d.). Retrieved from https://opencv.org/

14. Roudposhti, V. M., & Moeslund, T. B. (2018). A survey of methods for hand gesture recognition. In Computer Vision and Image Understanding, 172, 90-105.

15. Starner, T., Weaver, J., & Pentland, A. (1998). Real-time American Sign Language recognition from video using hidden Markov models. In CVPR.

16. Velloso, E., Bulling, A., Gellersen, H., & Ugulino, W. (2013). Qualitative Activity Recognition of Weight Lifting Exercises. In UbiComp.

17. Xu, H., & Yang, Y. (2018). Hand gesture recognition with depth images: A review. In IEEE Access, 6, 76051-76067.

18. Zhang, Y., & Sawchuk, A. A. (2000). Hand posture recognition from a 2D image. In CVPR

**Chapter 10: Future Scope of the Project**

1. **Advanced Gesture Vocabulary**

There is a possibility of extending the gesture vocabulary to include more complex and context-specific gestures, enhancing the system's capabilities and interaction possibilities.

1. **Integration with Virtual Reality**

We can explore the ways in which we can integrate the Gesture-Controlled Presentation System with virtual reality environments, creating immersive presentation experiences that go beyond traditional slide decks.

1. **Collaborative Gesture-Controlled Environments**

We can also envision a future where multiple presenters can interact with the same presentation simultaneously, allowing for collaborative gesture-based interactions in real-time.

1. **Integration into PowerPoint**

We can calibrate the software to integrate and run with presentation software like Microsoft PowerPoint.

**Chapter 11: Bibliography**

* Gesture Recognition: The process of identifying and interpreting human gestures, such as hand movements or body postures, as a means of interaction with a computer system.
* User Interface (UI): The point of interaction between a user and a computer system, including elements like buttons, menus, and visual indicators that enable users to interact with and control the system.
* Depth Sensing Camera: A camera equipped with technology to capture depth information of objects in a scene, enabling accurate tracking and recognition of gestures and movements.
* Machine Learning: A subset of artificial intelligence that involves training a computer system to learn patterns and make decisions from data, often used for tasks like gesture recognition.
* Presentation Software API: Application Programming Interface that allows developers to interact programmatically with presentation software, enabling the automation of tasks like slide navigation and content manipulation.
* User Experience (UX): The overall experience and satisfaction that users have while interacting with a system, encompassing aspects like usability, accessibility, and emotional engagement.
* Activity Diagram: A visual representation of a sequence of actions, decisions, and interactions within a system, showing the flow of activities and the order in which, they occur.
* Sequence Diagram: A visual representation of the interactions between objects or components in a system over a specific scenario, illustrating the messages exchanged and the order of actions.
* Use-Case Diagram: A diagram that represents interactions between actors (users or external systems) and a system, showcasing how users interact with the system's functionalities.
* User Scenario: A detailed narrative that describes how a user interacts with a system to achieve a specific goal, often used to understand user behaviours and design requirements.
* User Feedback: Information provided by users about their experiences, opinions, and impressions of a system, used to improve its functionality and user experience.
* User Persona: A fictional character representing a specific user type, created to help designers understand and empathize with users' needs and behaviours.
* Interaction Design: The practice of designing how users interact with a system, focusing on creating intuitive and efficient user interfaces and experiences.
* User-Centred Design: An approach to design that prioritizes the needs, preferences, and behaviours of end-users throughout the design and development process.
* Usability Testing: The process of evaluating a system's user interface by observing real users as they interact with it, identifying usability issues and areas for improvement.
* System Boundary: A visual representation that defines the scope of the system under consideration in diagrams like use-case diagrams and activity diagrams.