Gesture-Based Smart Presentation Assistant

Team members:

1- **Rewaa Alaa El-Dein 320210297**
2- **Nouran Galal 320210264**
3- **Sama El-Qasaby 320210313**

Introduction and Objectives

The Smart Presentation Assistant is a Python-based tool designed to transform how presenters interact with PowerPoint or Google Slides by enabling hands-free control through hand gestures and securing access with facial recognition. Our goal was to create an intuitive system that enhances the presentation experience, making it more engaging and secure for users. The project was developed by a team of six, each contributing to specific components to meet both core and optional requirements.

The system's objectives are:

- **Presenter Verification**: Use facial recognition to ensure only authorized users can control the presentation.
- **Gesture Navigation**: Allow slide control (next, previous, start, end) using distinct hand gestures.
- **On-Screen Feedback**: Display presenter details, gesture status, and system state in real time.
- **Emotion-Based Pausing**: Pause the presentation if the presenter shows signs of confusion (optional feature).
- **Camera Calibration**: Correct webcam distortion to improve gesture detection accuracy.

This report outlines the tools we used, how we built the system, the challenges we faced, and our vision for future improvements.

Tools and Technologies

We selected a suite of reliable libraries to build the system, balancing performance and ease of use:

- **OpenCV**: Handles webcam input, image processing, and text overlays for the user interface.
- **MediaPipe**: Provides robust hand-tracking capabilities for gesture recognition.
- **DeepFace**: Powers facial recognition for presenter authentication and emotion detection.
- **NumPy**: Supports efficient numerical computations for image and landmark processing.
- **pyttsx3**: Enables voice feedback to confirm actions like slide changes.
- **pyautogui**: Simulates keyboard inputs to navigate presentation software.
- **python-pptx**: Allows loading PowerPoint files for potential direct integration.
- **System Utilities (OS, Sys, Platform, Time)**: Manage file paths, platform compatibility, and timing.

These tools were chosen for their proven effectiveness in computer vision and real-time applications, supported by strong community resources.

System Implementation

The project is organized into seven Python modules, each handling a distinct aspect of the system. Below, we describe how each component works and how they integrate to deliver the required functionality.

Camera Calibration

This module aims to correct webcam lens distortion to enhance gesture detection. Currently, it serves as a placeholder, passing frames without modification. We plan to implement chessboard-based calibration to fulfill this requirement fully.

Presenter Authentication

To ensure security, the system verifies the presenter's identity by comparing a webcam frame to a reference image using DeepFace's facial recognition model. The process involves saving a temporary frame, performing the comparison, and displaying the result on-screen with the presenter's name and status (green for verified, red for unverified). Authentication is limited to 10 attempts to avoid delays, with a fallback to proceed if verification fails repeatedly.

Gesture-Based Navigation

The system recognizes four hand gestures to control slides:

- **Previous**: A thumbs-up gesture.
- **Next**: A thumbs-down gesture.
- **Start**: A peace sign (index and middle fingers extended).
- **End**: An open palm with spread fingers.
  Using MediaPipe's hand-tracking, the system analyzes finger positions and thumb orientation to identify gestures. Detected gestures trigger slide changes, with voice feedback confirming actions (e.g., "Next slide"). The interface shows the gesture name and debug details, like the number of extended fingers, to aid troubleshooting.

Emotion Detection

An optional feature pauses the presentation if the presenter appears confused or upset. DeepFace analyzes facial expressions, grouping them into "happy" (happy, neutral, surprised) or "sad" (sad, angry, fearful). If a sad emotion is detected, the slideshow pauses, and a message appears on-screen. Happy emotions allow the presentation to continue, with status updates displayed in green.

User Interface

The interface overlays critical information on the webcam feed, including:

- Presenter name and authentication status.
- Current gesture (with a checkmark if detected).
- Emotion state (happy or paused due to sadness).
- Instructions for gesture controls.
- Debug data for gesture detection.
  Text is color-coded for clarity, and the layout is designed to be unobtrusive yet informative.

Presentation Control

Slide navigation is achieved by simulating keyboard inputs (e.g., right arrow for next slide, F5 to start). The system also includes support for loading PowerPoint files, laying the groundwork for direct integration. Gestures trigger these controls only when the presenter is authenticated and emotions permit.

Main Application

The main script ties all modules together, initializing the webcam, loading slide images (JPEGs for testing), and managing the application loop. It handles errors, such as missing files or webcam failures, and ensures smooth operation by coordinating authentication, gesture detection, emotion analysis, and interface updates.

Key Features

- Voice feedback for actions, enhancing accessibility.
- Debug information to monitor system performance.
- Robust error handling for file and hardware issues.
- A clean, color-coded interface for real-time feedback.

Challenges and Resolutions

We encountered several hurdles during development, each requiring creative solutions:

1. **Incomplete Camera Calibration**: Without a working calibration module, we relied on MediaPipe's robustness for gesture detection. We prioritized other features but noted calibration as a future task.
2. **Slow Facial Recognition**: DeepFace's processing caused occasional lag. We reduced overhead by skipping unnecessary face detection steps and capping authentication attempts.
3. **Voice Feedback Issues**: The voice engine failed to initialize on some systems. We added retry logic and a fallback to print messages to the console.
4. **Gesture Errors**: Subtle hand movements sometimes triggered wrong gestures. We fine-tuned detection thresholds and displayed debug data to identify issues.
5. **File Management**: Missing image files caused crashes. We implemented checks to verify file existence before processing.

These solutions ensured the system remained functional while meeting deadlines, though some areas, like calibration, require further work.

Future Enhancements

To improve the system, we propose:

- **Full Camera Calibration**: Implement distortion correction using a chessboard pattern.
- **Faster Recognition**: Explore lighter models, like MediaPipe Face Mesh, for authentication and emotion detection.
- **Custom Gestures**: Allow users to define their own gestures for flexibility.
- **Direct Integration**: Control slides via PowerPoint APIs or browser automation instead of keyboard simulation.
- **Multi-User Support**: Authenticate multiple presenters for collaborative presentations.
- **Advanced Emotions**: Detect a wider range of emotions for more nuanced pausing.

These improvements would make the system more robust and adaptable to diverse use cases.

Conclusion

The Smart Presentation Assistant delivers a functional, innovative solution for hands-free presentation control. It successfully implements presenter authentication, gesture navigation, emotion-based pausing, and a clear user interface, with voice feedback as a bonus feature. While challenges like camera calibration and performance were addressed with practical workarounds, the system meets its core objectives and provides a foundation for future enhancements. We believe it has the potential to redefine how presenters engage with their audiences.