

Hackathon Project

Project Title:

Gesture-Based Human-Computer Interaction System using OpenCV, MediaPipe and Plam's text-bision-001

Team Name:

KATABATTHINI SADHANA

Team Members:

Katabatthini Sadhana

Kopurotu Ramya

Gurram Sravani

Kotturi Vaibhav Sai

Kethipelly Harish Reddy

Objective:

The project aims to improve user-computer interaction by creating a gesture-based system that can recognize a variety of hand gestures with a webcam; it will provide a smooth, intuitive mouse control experience; it will incorporate a paint application that is fully controlled by gestures, allowing users to express their creativity in a more efficient and natural way; and, finally, it will offer a novel substitute for conventional input devices, improving accessibility and usability across various platforms.

Problem Statement:

The Gesture-Based Human-Computer Interaction System leverages real-time hand gesture recognition to enable users to interact with computers through intuitive hand movements. Utilizing computer vision techniques with OpenCV and MediaPipe, the system detects and interprets a variety of gestures, such as thumbs up, fist, open hand, and more. Integrated with a generative AI model, it provides descriptive narratives for recognized gestures, enhancing user experience. A user-friendly Streamlit interface facilitates easy interaction and visualization, making this system ideal for touchless control, interactive gaming, and assistive technologies.

Proposed System:

- The Gesture-Based HCI System uses multimodal input, such as voice control, and real-time gesture detection (via OpenCV and MediaPipe) to enable natural, hands-free interactions between users and their gadgets.
- Through hand gestures and vocal instructions, the system will allow users to do things like navigate the web, start programs, manage PowerPoint presentations, and engage with multimedia material.

Key Features:

- **Gesture Recognition:** Enables users to engage with the system through pinches, swipes, and other gestures by recognizing both basic and sophisticated hand movements.
- **Voice Control:** Uses voice instructions to operate programs like media players, PowerPoint, and web navigation.
- **Personalized Calibration:** Through a calibration phase, it adjusts to the user's distinct voice commands and movements.
- **Multimodal Interaction:** Easily initiates actions by combining speech and gesture instructions.
- **Accessibility:** Provides user-friendly features that guarantee usability for those with impairments.

Target Users:

- **General Consumers** – Individuals who are searching for a new, touchless method of connecting with their devices that makes everyday tasks easier and more entertaining, such as managing presentations, browsing, or controlling media, want a simple, hands-free method of interacting with their devices that makes daily life more convenient and entertaining.
- **Individuals with Disabilities** – This approach makes it simpler for persons with disabilities to interact with computers and other devices by providing accessibility and inclusion using gestures that can be tailored to each user's needs. A more intuitive and customized experience is provided by reducing the complexity of conventional input devices so that consumers may operate them with simple voice commands or memorable gestures.
- **Educational Institutions** – In order to boost student engagement in STEM (science, technology, engineering, and art) subjects, educational institutions can use the system to create interactive learning experiences that use touchless, gesture-based interactions. Students can interact more dynamically and hands-on with learning materials, and teachers can control presentations, interactive lessons, and classroom activities without touching any devices.

Expected Outcome:

A major increase in user experience through smooth, hands-free interactions that improve accessibility, particularly for older users and those with impairments, is one of the anticipated results of the Gesture-Based Human-Computer Interaction (HCI) System with voice control. The technology will improve productivity, facilitate cross-platform multitasking, and expedite processes by providing customized gesture and speech commands. It will boost productivity and engagement in professional and educational contexts, enabling users to take complete control of media and presentations and fostering dynamic learning environments. Furthermore, the technology would facilitate smart home integration, guaranteeing users can conveniently control gadgets and improving hygiene by minimizing physical contact. Through improved creative workflows, safer driving experiences, and a more inclusive interface for users in a variety of industries, the technology will eventually lead to wider adoption, higher user satisfaction, and environmental benefits by reducing the need for physical input devices.

Technical Requirements:

- Programming Language: Python
- **Frontend:** Streamlit
- **Backend:** OpenCV, MediaPipe

Functional Requirements:

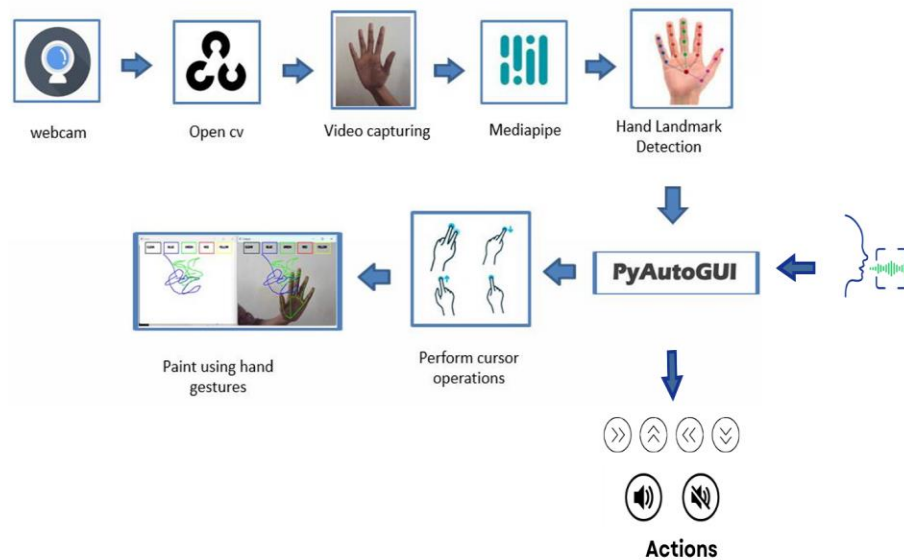
In order to achieve accurate gesture tracking and recognition, the system should use MediaPipe and OpenCV to capture and process real-time hand gesture data. This will enable actions like clicking, scrolling, and launching apps through user-friendly, customizable interfaces. Gesture setups should be movable to allow users to customize their experience and improve accuracy. Voice control should be integrated to complement gesture-based actions, allowing users to issue commands like opening applications, adjusting volume, or performing specific tasks hands-free. The system should be cross-platform compatible across Windows, Mac, and Linux, ensuring seamless gesture and voice-based control across multiple devices and platforms. Finally, voice feedback should be used to confirm actions or provide instructions, improving accessibility and user experience.

Constraints and Challenges:

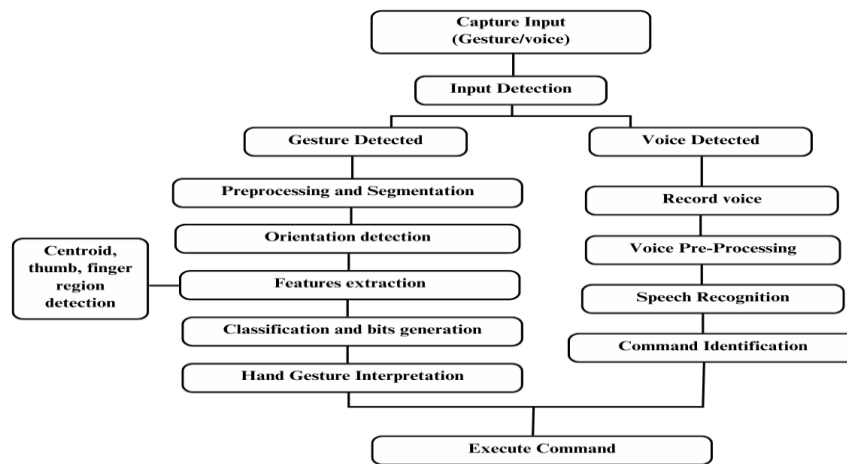
- **Real-Time Processing:** Reducing latency and guaranteeing fluid, in-the-moment gesture tracking to facilitate user engagement.
- **User Calibration:** Enabling quick and accurate system calibration to accommodate unique user preferences and gestures.

- **Cross-Platform Compatibility:** Making sure the system runs reliably on several different operating systems, such as Linux, macOS, and Windows.
- **Scalability:** The ability of the system to accommodate an increasing number of users and gestures without sacrificing functionality.
- **Data security and privacy:** safeguarding user data, especially biometric data, and making sure privacy laws are followed.

System Architecture:








User Flow:



Project Planning (Agile Methodologies):

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & Library Integration	● High	6 hours	End of Day 1	Member 1	OpenCV, MediaPipe, Palm's text-bison-001 setup	Libraries integrated and working
Sprint 1	Gesture Recognition Model Setup	● High	4 hours	End of Day 1	Member 2	OpenCV, MediaPipe library initialized	Gesture recognition system working with webcam
Sprint 1	Gesture Action	● High	3 hours	End of Day 1	Member 3	Gesture recognition	Map gestures

	Mapping					ion model setup	to actions (scroll, click, etc.)
Sprint 1	Voice Command Integration - Basic Setup	● High	2 hours	End of Day 1	Member 4	Palm's text-bison-001	Basic voice command framework initialized (e.g., "Start", "Next", "Previous")
Sprint 2	Voice Control for PPT Presentation	● High	3 hours	Mid-Day 2	Member 4	Palm's text-bison-001 integration, Voice command setup	Implemented voice commands for PPT control (Next Slide, Previous Slide, Start, End)
Sprint 2	Text-to-Speech Integration	● High	2 hours	Mid-Day 2	Member 5	Palm's text-bison-001 integration	Text-to-speech for feedback on gestures and commands

Sprint 2	Voice Feedback for Gesture Recognition	 High	2 hours	Mid-Day 2	Member 2	Voice Control System	Voice feedback for recognized gestures (e.g., "You swiped left", "Action : Scroll")
Sprint 2	Error Handling & Debugging	 High	1.5 hours	Mid-Day 2	Member 5	Gesture recognition model	Debugging and fixing gesture detection errors
Sprint 2	Custom Gesture Calibration	 Medium	2 hours	Mid-Day 2	Member 2	Gesture recognition model setup	Allow users to calibrate custom gestures
Sprint 3	Accessibility Features for Voice and Gestures	 Medium	2 hours	Mid-Day 2	Member 4	Gesture recognition actions	Enable accessible gestures and voice commands for users with disabilities
Sprint 3	Testing & Debugging	 Medium	2 hours	Mid-Day 2	Member 1 & 5	All components working	Debug and ensure stable

							system operation
Sprint 3	Final Demo Preparation & Deployment	● Low	1 hour	End of Day 2	Entire Team	Working prototype	Ready-to-present, functional system

Sprint 1 – Setup & Initial Development (Day 1)

- (● **High Priority**) Set up environment & install dependencies (OpenCV, MediaPipe, Palm's text-bison-001). (Assigned to: Member 1)
- (● **High Priority**) Initialize gesture recognition model using MediaPipe and OpenCV. (Assigned to: Member 2)
- (● **High Priority**) Implement gesture-to-action mapping for functions like scroll, click, etc. (Assigned to: Member 3)
- (● **High Priority**) Set up voice command framework for controlling the system (basic setup for voice control like "Start," "Next Slide," "Previous Slide"). (Assigned to: Member 4)

Sprint 2 – Setup & Initial Development (Day 2)

- (● **High Priority**) Implement **voice control for PowerPoint presentation** (Next Slide, Previous Slide, Start Presentation, End Presentation, etc.). (Assigned to: Member 4)
- (● **High Priority**) Integrate **text-to-speech** for feedback on gestures and voice commands. (Assigned to: Member 4)
- (● **High Priority**) Debugging and error handling for gesture recognition, voice control, and action mapping. (Assigned to: Member 5)
- (● **Medium Priority**) Allow **custom gesture calibration** for user-specific gestures. (Assigned to: Member 2)

Sprint 3 – Setup & Initial Development (Day 2)

- (● **Medium Priority**) Add **accessibility features** for users with disabilities, such as simplified gestures and voice prompts. (Assigned to: Member 4)
- (● **Medium Priority**) Conduct **testing and finalize debugging** for smooth interaction between gesture control and voice control. (Assigned to: Member 1 & 5)

- (● Low Priority) **Final demo preparation and deployment** of the functional system, showcasing voice and gesture control features. (Assigned to: Entire Team)

Voice Control Features for PPT:

- **Voice Command Integration:**
 - Use **speech recognition** to handle voice commands for controlling the PowerPoint presentation (e.g., **Google Speech-to-Text, Microsoft Azure Speech**).
 - Key commands include:
 - **zoom in:**
 - **zoom out:**
 - **screenshot:**
 - **scroll up:**
 - **scroll down:**
 - **scroll left:**
 - **scroll right:**
 - **volume up:**
 - **volume down:**
 - **mute:**
 - **brightness up:**
 - **brightness down:**
 - **Self:**
 - **previous window:**
 - **maximize:**
 - **close window:**
- **Voice Feedback for Gesture Recognition:**
 - Provide real-time **voice feedback** when a gesture is recognized.
 - Example feedback: "You performed a left swipe" or "Action: Scroll up."
- **Testing & Debugging Voice Control:**
 - Test that voice commands are properly recognized, accurately triggering PPT actions.
 - Ensure seamless integration with gesture recognition for a hands-free experience.
- **Accessibility:**
 - Ensure that voice control also functions for users with disabilities by allowing voice commands to control gestures, simplifying the interaction.

Technology Stack Used:

- **Frontend:** Streamlit
- **Backend:** OpenCV, MediaPipe
- **AI Model:** Palm's Text-Bison-001
- **Programming Language:** Python
- **Computer Vision Library:** OpenCV (for real-time gesture recognition)
- **Gesture Recognition Library:** MediaPipe (for hand tracking and gesture detection)

Development Process:

- **Integrate OpenCV and MediaPipe** for real-time hand gesture recognition, enabling the system to detect both simple and complex gestures.
- **Map gestures** to specific system actions like scrolling, clicking, and media control for seamless interactions.
- **Integrate Palm's Text-Bison-001** to generate descriptive stories or feedback based on detected gestures, providing users with auditory and visual cues.
- **Enhance system performance** by optimizing gesture recognition for various lighting conditions and reducing lag for smooth real-time interactions.
- **Develop a user-friendly interface** that displays real-time feedback on recognized gestures and executed actions.
- **Implement voice control** for hands-free operation, enabling voice commands like "Play," "Pause," and "Next Slide" for enhanced control.
- **Combine gesture recognition and voice control** for flexible, multimodal user interaction, allowing users to switch between both methods seamlessly.
- **Include accessibility features**, such as customizable gestures and voice prompts, to ensure usability for people with disabilities.
- **Ensure cross-platform compatibility**, making the system work across Windows, Mac, and Linux, while testing on various devices.
- **Test and debug the system**, followed by creating detailed user documentation and setup guides to ensure smooth deployment and operation.

Challenges & Fixed:

1. Gesture Recognition Accuracy

- **Challenge:** Inconsistent or inaccurate gesture detection due to varying hand movements, lighting conditions, or webcam quality.
- **Fix:** Optimize the **gesture recognition algorithm** by fine-tuning the model and implementing real-time feedback mechanisms. Ensure that the system is tested under different environmental conditions and lighting, and provide an option for **gesture calibration** to improve accuracy.

2. Performance Lag

- **Challenge:** Delays between gesture input and system response, causing a lag in real-time interactions.
- **Fix:** **Optimize the processing pipeline** by reducing unnecessary computations and utilizing **GPU acceleration** for faster image processing. Use multi-threading to handle gesture recognition, voice input, and UI updates concurrently.

3. Voice Command Recognition Errors

- **Challenge:** Voice commands may be misunderstood or not detected accurately, especially in noisy environments.
- **Fix:** Implement a **noise-cancellation** algorithm to filter background noise and improve the accuracy of voice input. Additionally, **train the speech recognition system** to recognize specific phrases and commands with high accuracy.

4. Cross-Platform Compatibility Issues

- **Challenge:** The system may not work consistently across different operating systems (Windows, Mac, Linux), causing inconsistencies in performance.
- **Fix:** Use **cross-platform frameworks** such as **Python with OpenCV and MediaPipe**, and test on multiple platforms to identify and resolve OS-specific issues. Ensure that dependencies are properly configured for each system.

5. Accessibility and Customization

- **Challenge:** Ensuring that the system is accessible to users with disabilities, such as those with limited motor skills or visual impairments.
- **Fix:** Provide **customizable gesture setups**, including adjustable speed and sensitivity for gesture recognition. Integrate **voice feedback** for users with visual impairments and **haptic feedback** (if applicable) for users with hearing impairments.

6. Synchronization Between Gesture and Voice Controls

- **Challenge:** Confusion or miscommunication when both gesture and voice controls are used simultaneously, leading to conflicting actions.
- **Fix:** Implement a **clear system state indicator** that tells the user whether the system is currently waiting for a gesture or a voice command. Allow **priority settings** to handle simultaneous inputs (e.g., gestures may take precedence over voice commands).

7. Hardware Limitations

- **Challenge:** Different devices (webcams, microphones, PCs) may have varying performance capabilities, affecting gesture recognition and voice control.
- **Fix:** Conduct **hardware testing** and **optimization**, ensuring that the system works on devices with lower specs. Allow for **adjustable settings** in the software, such as resolution and frame rate, to ensure compatibility with older or less powerful hardware.

8. Gesture Fatigue

- **Challenge:** Users may experience fatigue or discomfort when using gestures for extended periods.
- **Fix:** Design the system to recognize **simple, minimal gestures** for prolonged use. Additionally, incorporate **voice commands** as an alternative to gestures, providing users with a more relaxed way of interacting with the system.

9. Error Handling and Debugging

- **Challenge:** Bugs or unhandled exceptions that disrupt the system's functionality, causing crashes or incorrect responses.
- **Fix:** Implement comprehensive **error-handling routines** and **logging** to capture issues and resolve them quickly. Perform **unit and integration testing** to ensure stability across all features before deployment.

10. User Learning Curve

- **Challenge:** Users may have difficulty learning how to use gesture-based and voice-controlled systems effectively.
- **Fix:** Create **user-friendly tutorials** and **interactive onboarding** to guide users in setting up gestures and voice commands. Offer customizable settings that allow users to define gestures and voice commands that feel natural to them.

Test Cases for Gesture-Based HCI System

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Recognize hand gesture for "scroll up"	System should scroll up the page in the active application.	✅ Passed	Tester 1
TC-002	Functional Testing	Recognize hand gesture	System should zoom	✅ Passed	Tester 2

		for "pinch-to-zoom"	in or out in the active application.		
TC-003	Functional Testing	Voice command "Next slide" for presentation control	System should switch to the next slide in the presentation.	✅ Passed	Tester 1
TC-004	Performance Testing	Hand gesture recognition delay should be under 200ms	Gesture recognition should be near real-time, with minimal delay.	⚠ Needs Optimization	Tester 3
TC-005	Bug Fixes & Improvements	Incorrect gesture for "fist" mapped to "open hand" action.	The system should correctly map a closed fist to a "close" action.	✅ Fixed	Developer
TC-006	Final Validation	Test voice and gesture recognition on Windows, Mac, and Linux	The system should recognize gestures and voice commands on all platforms.	✅ Passed	Tester 2
TC-007	Final Validation	Test UI responsiveness across mobile and desktop devices.	UI should work seamlessly on both mobile and desktop.	❌ Failed - UI broken on mobile	Tester 1
TC-008	Deployment Testing	Host the app using Streamlit Sharing	The app should be accessible online through the shared link.	🚀 Deployed	DevOps
TC-009	Functional Testing	Test gesture calibration functionality for custom gestures.	Users should be able to calibrate gestures for better accuracy.	✅ Passed	Tester 3

TC-010	Performance Testing	System performance with 10 simultaneous hand gestures tracked	System should not crash and handle multiple gestures without lag.	⚠ Needs Optimization	Tester 2
--------	---------------------	---	---	----------------------	----------