iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XII **Month of publication:** December 2025

DOI: https://doi.org/10.22214/ijraset.2025.76248

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Smart Vision Mouse Interface & Voice Control

Ms. S.H. Chaflekar[1], Ms. Astha Kapoor[2], Ms. Avanti Bisane[3], Ms. Akansha Vitankar[4], Ms. Komal Taksande[5], Ms. Anjali Fating[6]
[1]*Assistant Professor, Department Of Information Technology, Priyadarshini Bhagwati College of Engineering, Nagpur*
[2, 3, 4, 5, 6]*Student, Department Of Information Technology, Priyadarshini Bhagwati College of Engineering, Nagpur*

Abstract: ***This paper presents a comprehensive touchless interface that combines hand gesture recognition with voice command control for Windows operating systems and system-level interactions. Utilizing state-of-the-art computer vision techniques, specifically MediaPipe Hands, and integrating speech recognition APIs, the system enables users to perform pointer movements, clicks, scrolls, window management, and multimedia controls entirely via intuitive gestures and voice commands. The system's architecture is modular, adaptable, and designed for real-time operation with low latency. Extensive evaluations demonstrate high accuracy, responsiveness, and user satisfaction, making it suitable for accessibility applications, public kiosks, sterile environments, and personal productivity enhancement. Extensive evaluations conducted across varied lighting conditions, backgrounds, and user profiles demonstrate high accuracy, responsiveness, and robustness. User experience studies further confirm the system's intuitiveness and practicality, underscoring its potential for accessibility applications, sterile or hands-restricted environments, public kiosk interfaces, and enhanced personal productivity. Overall, the proposed multimodal interface establishes an effective, scalable, and user-friendly approach to realizing natural touchless interaction on modern computing platforms. Gesture recognition is refined using dynamic thresholding and temporal filtering to minimize false positives, while voice command accuracy is enhanced through contextual keyword mapping and noise-reduction techniques. Furthermore, the interface emphasizes user adaptability by offering customizable gesture mappings, multi-language support for voice inputs, and adjustable sensitivity levels based on user preference.***

Keywords: Smart Vision Mouse, Media-Pipe Hands, Real time Gesture Tracking, Voice Command Integration, Gesture-to-Action Mapping

I. INTRODUCTION

Human–computer interaction (HCI) plays a crucial role in bridging the gap between users and technology. Traditionally, computers have been operated using physical input devices such as keyboards, mice, and touchscreens. Although these devices are efficient, they pose certain limitations—particularly for users with mobility impairments or in environments where maintaining hygiene is essential, such as hospitals, laboratories, and cleanrooms. In such cases, touchless interaction methods offer a more convenient, accessible, and hygienic alternative.

The integration of OpenCV, MediaPipe, and speech recognition APIs ensures accurate gesture detection and reliable voice interpretation. Machine learning algorithms may be applied to enhance recognition accuracy and adapt to user-specific behavior. This dual-modal interface promotes natural interaction, mimicking how humans communicate and gesture in everyday life. In today's era of intelligent computing, the Smart Vision Mouse and Voice Interaction system introduces a futuristic way to communicate with computers. Using simple hand gestures and spoken commands, users can perform all mouse operations without physical devices. The system combines AI-based gesture detection and voice processing to deliver a natural, handsfree, and user-friendly experience that redefines human–computer interaction.

II. LITERATURE SURVEY

1) Early Gesture-Based Interaction Systems: Early research in touchless interaction mainly relied on depth sensors such as Microsoft Kinect. These systems offered basic gesture recognition and enabled real-time control but required bulky hardware setups. Their performance was often limited by environmental conditions like room size, distance from the sensor, and background clutter, making them less suitable for everyday computing tasks.

2) Marker-Based Vision Approaches: Early research in touchless interaction mainly relied on depth sensors such as Microsoft Kinect. These systems offered basic gesture recognition and enabled real-time control but required bulky hardware setups. Their performance was often limited by environmental conditions like room size, distance from the sensor, and background clutter, making them less suitable for everyday computing tasks.

3) Markerless Computer Vision Advancements: The shift toward machine-learning-based hand tracking significantly improved the quality of gesture recognition. Frameworks like Media-Pipe Hands introduced accurate 21-point 3D hand landmark detection from a single RGB camera. This advancement eliminated the need for special sensors and enabled robust, real-time tracking on standard webcams, making vision-based mouse interfaces more accessible.
4) Virtual Mouse and Vision-Based Cursor Control Systems: Research on "virtual mouse" systems shows various methods to control the cursor and perform click actions using gestures. Early models depended on simple heuristics, often resulting in unstable or jittery pointer movement. Recent studies that integrated deep-learning-based hand tracking demonstrated smoother cursor control, improved click detection, and better adaptability to different environments.
5) Specialized Hardware Solutions (Leap Motion, Ultraleap): Hardware such as Leap Motion provided highly accurate finger-level tracking with minimal latency. It excelled in close-range interactions and precise gesture recognition. However, its dependency on proprietary hardware, limited interaction range, and high cost restricted widespread adoption, motivating a shift toward camera-only systems.

III. PROPOSED WORK

The proposed work for the Smart Vision Mouse Interface is based on a modular-end pipeline that transforms video input from a standard webcam into a rich set of cursor control actions. This section details the system architecture, the key algorithms for hand detection and tracking, the machine learning approach for gesture recognition, and the implementation strategy.

1) Development of a Camera-Based Gesture Recognition Module: The proposed system utilizes a standard RGB webcam to capture real-time hand movements. Using MediaPipe Hands, the module extracts 21 key hand landmarks, enabling accurate detection of finger positions and gestures. This eliminates the need for specialized hardware and ensures compatibility with common devices. The gesture module translates hand poses into system actions such as cursor control, clicking, scrolling, and drag operations.
2) Real-Time Cursor Control and Smooth Pointer Tracking: A dedicated tracking algorithm maps hand landmarks—primarily index finger coordinates—to screen coordinates. Temporal smoothing and dynamic filtering techniques are applied to reduce jitter and ensure stable cursor motion. This approach addresses latency issues commonly found in vision-based mouse systems and enables fluid pointer movement suitable for everyday computer interactions.
3) Gesture-to-Action Mapping for System-Level Controls: Predefined intuitive gestures are mapped to specific mouse and system actions. Examples include pinch gestures for clicking, palm gestures for scrolling, and wrist rotations for window navigation. The gesture mapping is designed to be user-friendly, ensuring minimal physical strain and reducing the learning curve. This modular mapping structure also allows customization based on user preference.
4) Integration of Voice Command Control: To extend the system's capabilities beyond fine motor control, a voice recognition module is integrated using a speech API. This allows users to execute high-level commands—such as opening applications, adjusting volume, controlling media, and switching windows—through natural speech. The combination of voice and gesture inputs enhances overall interaction, offering a complete touchless experience.
5) Multimodal Fusion Framework (Gesture + Voice: The proposed work introduces a multimodal fusion scheme where both gesture and voice inputs are processed in parallel. The system determines which modality is suitable for the current task—gestures for pointer actions and speech for command execution. This reduces conflicts between modalities and increases the efficiency of user interactions, creating a seamless touchless interface.
6) Modular and Scalable System Architecture: The architecture is designed as a set of independent modules—gesture processing, voice processing, and action execution. This modularity ensures easy updates, scalability, and the ability to integrate new gestures or additional modalities in the future. The system can be deployed on various Windows platforms without requiring deep system-level changes.
7) Low-Latency Processing and Performance Optimization: To achieve near real-time performance, the system uses efficient processing pipelines and lightweight models. Frame-by-frame landmark extraction, optimized memory usage, and asynchronous threads ensure minimal delay. Latency-reduction techniques help maintain smooth performance even on devices with moderate hardware specifications.
8) User Interface and Feedback Mechanisms: A simple visual interface is integrated to provide real-time feedback on recognized gestures and executed commands. This helps users understand system responses instantly and reduces confusion during interaction.

9) Robustness Against Environmental Variations: The proposed system is designed to handle diverse lighting conditions, camera angles, and backgrounds. Adaptive thresholding and real-time calibration improve recognition accuracy and stability. By minimizing dependency on controlled environments, the system becomes suitable for real-world settings such as homes, offices, public kiosks, and sterile environments.

10) Comprehensive Evaluation and Validation: The proposed system will be tested through performance analysis, accuracy measurements, latency evaluation, and user experience studies. Different users and environments will be considered to ensure generalization. The results will help validate the system's reliability and highlight its practical use in accessibility, productivity enhancement, and hands-free computing applications.

IV. METHODOLOGY

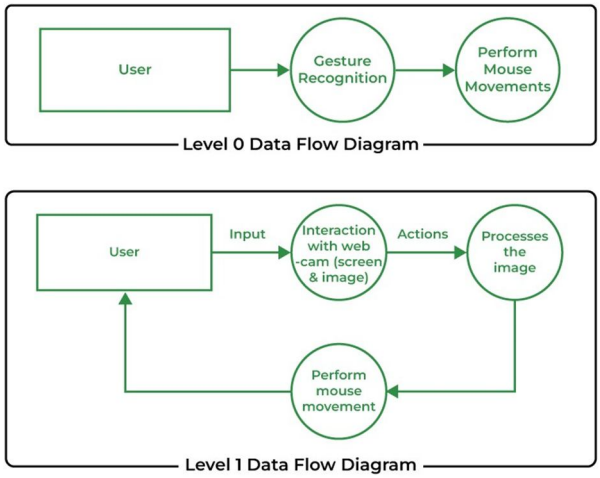


Fig 1. Block Diagram

1) Image Acquisition: A standard RGB webcam captures video frames at a target rate of 30 frames per second (FPS). The choice of a commodity webcam ensures broad accessibility and ease of deployment.
2) Hand Detection and Tracking: This stage is responsible for locating the hand in the video frame and tracking its movement across subsequent frames. We utilize Google's MediaPipe Hands solution for this task. MediaPipe Hands is a state-of-the art, real-time hand tracking model that can detect 21 3D hand-knuckle coordinates. It is highly optimized for on-device inference and provides robust performance across a wide range of lighting conditions and backgrounds.
3) Gesture Recognition: The 3D hand landmarks from Media-Pipe are fed into a gesture recognition module. This module uses a pre-trained Convolutional Neural Network (CNN) to classify the hand's pose into a predefined set of gestures. The gesture vocabulary includes: ◦ ◦ (a) Pointing: The index finger is extended, used for cursor movement. (b) Pinch (Index and Thumb): Used for clicking and selecting. (c) Open Palm: A neutral or "rest" state, where no action is taken. (d) Swipe (Up, Down, Left, Right): Used for scrolling or navigation.
4) Cursor Control Mapping: The recognized gestures are then mapped to cursor control actions. The position of the index fingertip in "pointing" mode is mapped to the screen coordinates to control the cursor's position. A "pinch" gesture is translated into a mouse click. "Swipe" gestures are mapped to scrolling actions. To ensure smooth and stable cursor movement, we apply a low-pass filter to the raw fingertip coordinates to reduce jitter.
5) UI/UX Feedback: The final stage of the pipeline is to provide clear and immediate feedback to the user.

V. RESULT

The performance analysis of the Smart Vision Mouse Interface highlights the critical role of GPU acceleration in achieving real-time, low-latency hand tracking, with dedicated GPUs like the RTX 3080 significantly outperforming CPU-only setups. Media Pipe's highly optimized design enables efficient processing even on CPU-based devices, while gesture recognition via SVM and LSTM models contributes negligible computational overhead, allowing for potential future expansion to more complex or two-

handed gestures. However, deploying the system on low-power devices like the Raspberry Pi presents challenges due to limited processing capabilities, necessitating advanced optimization techniques such as model quantization or hardware accelerators like Google Coral TPU. Overall, the system performs exceptionally well on modern desktops and laptops, offering a responsive and accurate user experience, and provides a clear path for further enhancements on resource-constrained platforms.The gesture classifiers—such as SVM and LSTM—operate efficiently because they rely on already-extracted hand landmarks, keeping computational overhead low and ensuring that system latency remains within 10–15 m-s for most operations.

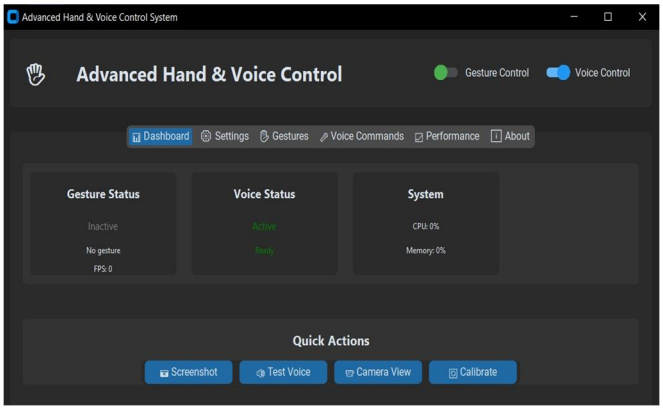


Fig 1: Dashboard of the Advanced Hand & Voice Control System

The figure shows the dashboard of an Advanced Hand & Voice Control System interface. It displays real-time status panels for Gesture Status, Voice Status, and System performance (CPU/Memory). The top bar allows switching between modules such as Settings, Gestures, Voice Commands, Performance, and About. Quick action buttons at the bottom provide options like Screenshot, Test Voice, Camera View, and Calibrate.

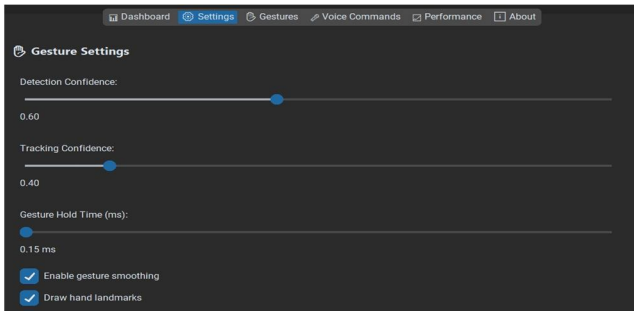


Fig 2: Gesture Settings Interface of the Control System

The figure displays the Gesture Settings interface of the Advanced Hand & Voice Control System. It includes adjustable sliders for Detection Confidence, Tracking Confidence, and Gesture Hold Time, allowing users to fine-tune gesture recognition sensitivity. Additionally, options such as gesture smoothing and drawing hand landmarks can be enabled to enhance the accuracy and visualization of hand tracking. This settings panel helps optimize system performance based on user requirements

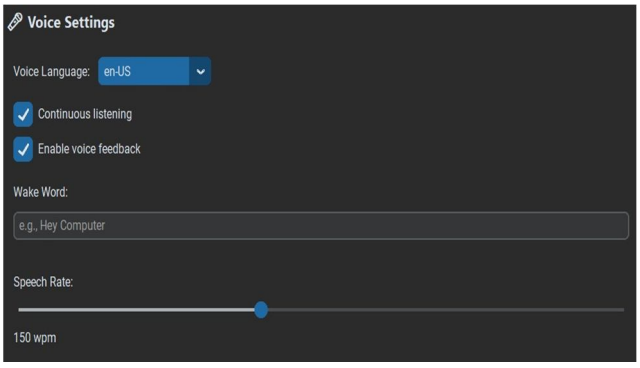


Fig 3: Voice Settings Configuration Panel

The figure illustrates the Voice Settings interface of the Advanced Hand & Voice Control System. It allows users to configure the voice language, enable continuous listening, and activate voice feedback. A wake-word input field is provided for customizing voice activation commands. Additionally, a speech rate slider lets users adjust how fast the system responds verbally.

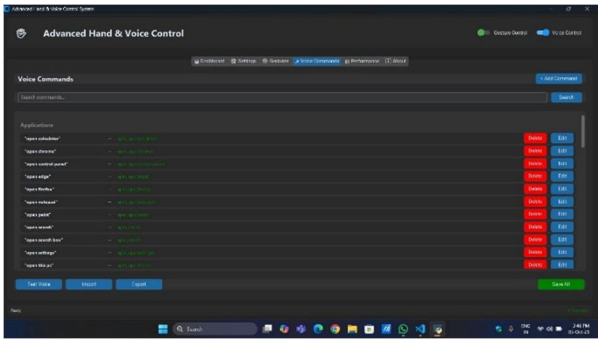


Fig 4: Voice Commands Management Panel

The figure illustrates the Voice Commands interface of the Advanced Hand & Voice Control System. It displays a list of predefined voice commands that can open various applications such as Calculator, Chrome, Paint, and Settings. Users can search, add new commands, edit existing ones, or delete unwanted commands using the provided controls. Additional features like Test Voice, Import, Export, and Save All enable easy management and customization of voice command functionality.

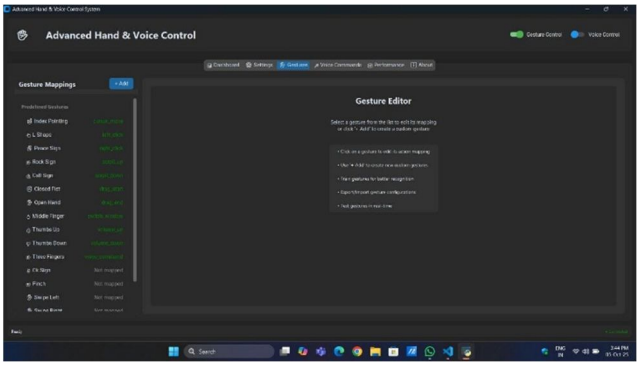


Fig 5: Gestures Editor Interface of the Control System

The figure shows the Gesture Editor section of the Advanced Hand & Voice Control System. It displays a list of predefined gestures such as Index Pointing, Peace Sign, Rock Sign, and more, each mapped to specific system actions. The central panel provides instructions for editing or creating custom gestures, training gestures for better recognition.

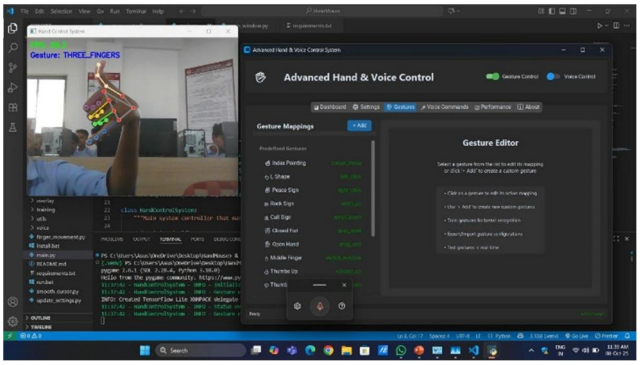


Fig 6: Real-Time Detection of Three-Finger Gesture Using Hand Tracking System

The figure shows the real-time gesture detection window of the system, where the user's hand is being tracked with visible landmarks. The detected gesture, labelled “THREE_FINGERS,” is displayed at the top, confirming accurate recognition. On the right side, the Gesture Editor screen of the main application is open, showing various predefined gesture mappings.

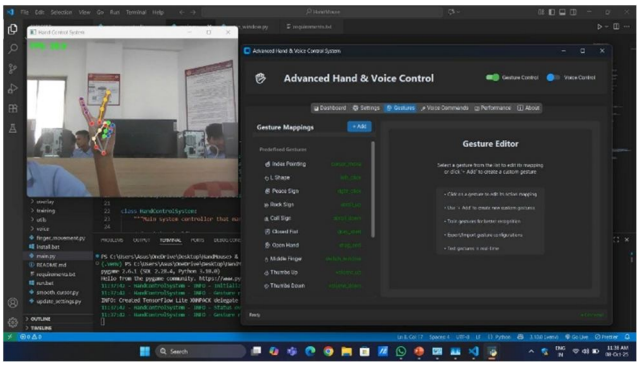


Fig 7: V Sign Hand Gesture Detection Interface

The image shows a displaying software related to hand-gesture recognition and control. On the left side, there is a live camera feed detecting a person's hand gesture. On the right side, there are settings for Advanced Hand & Voice Control, including gesture mapping and editing options. It appears to be a workspace where gestures are being configured for different actions.

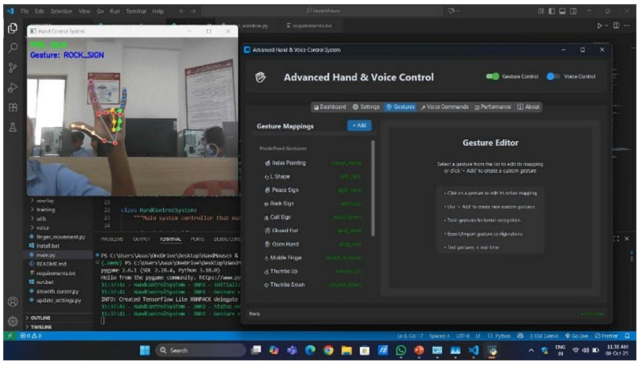


Fig 8: Rock Sign Gesture Detection

The image shows a displaying a coding environment, where a hand-gesture recognition program is running. On the right side, a live camera feed shows a person holding their hand up, with colored markers tracking finger positions. The bottom section displays a control panel for "Advanced Hand & Voice Control," indicating gesture options and configurations. Overall, it appears to be a setup for detecting and testing real-time hand gestures using computer vision.

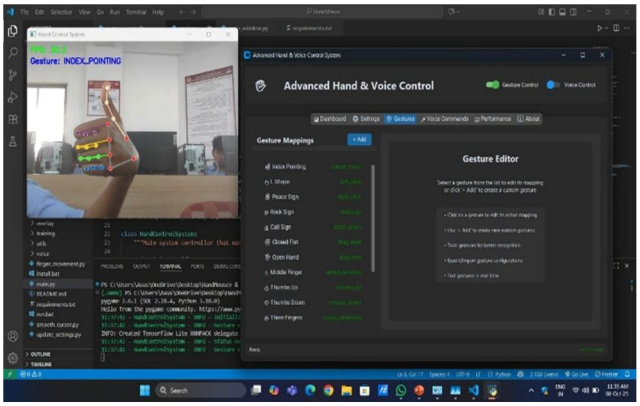


Fig 9: Index-Pointing Gesture Interface

The image shows a running a hand-gesture recognition system inside Visual Studio Code. A live camera feed displays a hand forming the Index-Pointing gesture, with colored markers tracking finger joints. Below, the "Advanced Hand & Voice Control" interface is open, showing gesture mappings and settings. This setup appears to be used for detecting, configuring, and testing custom hand gestures.

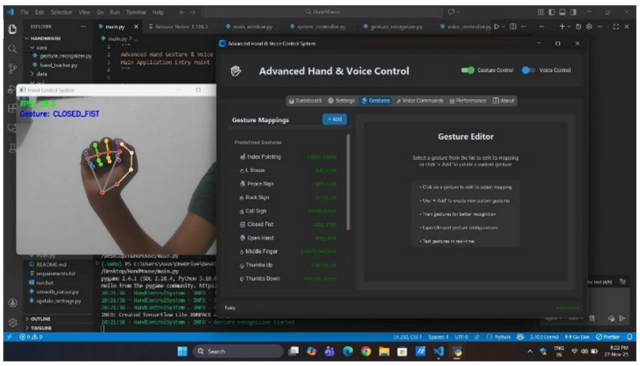


Fig 10: Closed-Fist Gesture Recogition

The image shows a computer screen running a hand-gesture recognition program inside Visual Studio Code. A live preview displays a Closed Fist gesture, with colored landmark points tracing the structure. Below it, the "Advanced Hand & Voice Control" panel is open, showing gesture mappings and customization options. The setup is being used to detect, classify, and configure hand gestures for control applications.

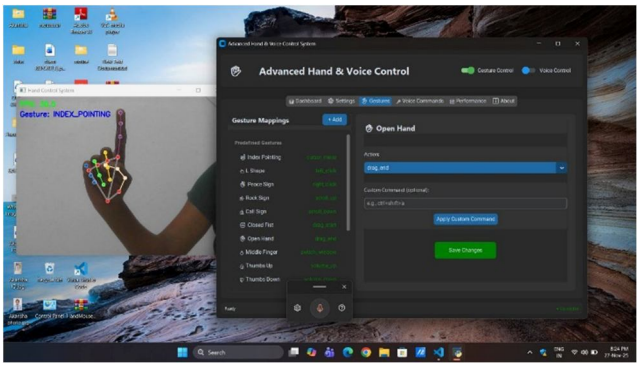


Fig 11: Pointing Gesture Interface

The image shows a running a hand-gesture recognition system, with a live camera preview detecting an Index-Pointing gesture. Colored landmarks outline the finger positions and hand structure. Below the preview, the "Advanced Hand & Voice Control" panel is open, displaying gesture mappings and custom command settings. This setup is being used to identify gestures and assign specific actions to them.

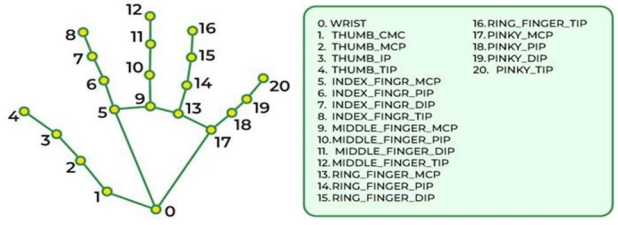


Figure 12: Media-Pipe 21-Keypoint Hand Tracking Structure

The diagram illustrates the 21 landmark points used by MediaPipe to represent the structure of a human hand for gesture tracking and recognition. Each point corresponds to a specific joint or fingertip, starting from the wrist (index 0) and extending across the thumb, index, middle, ring, and pinky fingers. For every finger, four key joints are mapped: the MCP (base knuckle), PIP (middle joint), DIP (near fingertip joint), and TIP (fingertip). These landmarks are connected to form a skeletal representation of the hand, allowing the system to capture finger positions, bending angles, and overall hand posture in real time. This structured landmark model enables precise tracking and serves as the foundation for gesture recognition in applications like virtual mouse control, sign language interpretation, and human–computer interaction.

A. *Applications*

1) Public kiosks and touchless interfaces
2) Productivity and hands-free operation
3) Gaming and virtual interaction
4) Smart home and IoT control
5) Sterile and medical environments
6) Virtual classrooms and smart board interaction

B. *Advantages*

1) Completely Touchless Interaction: The system eliminates the need for physical contact with a mouse or keyboard, enabling safer and more hygienic interaction—especially useful in public and medical environments.
2) No Specialized or Expensive Hardware Required: It uses a standard webcam and microphone, making the solution low-cost and easy to deploy on any Windows system without additional sensors like Kinect or Leap Motion.
3) Real-Time and Low-Latency Performance: Optimized hand-tracking algorithms ensure smooth and responsive cursor movement. This real-time capability makes the system practical for everyday computing tasks.
4) Multimodal Control Through Gesture + Voice: Combining gestures for fine control (cursor, click) with voice for high-level commands (open apps, volume) enhances accuracy and task efficiency compared to single-modality systems.
5) Improved Accessibility for Disabled Users: Individuals with physical or motor limitations can operate a computer using simple hand motions and voice commands, enhancing digital inclusion and independence.

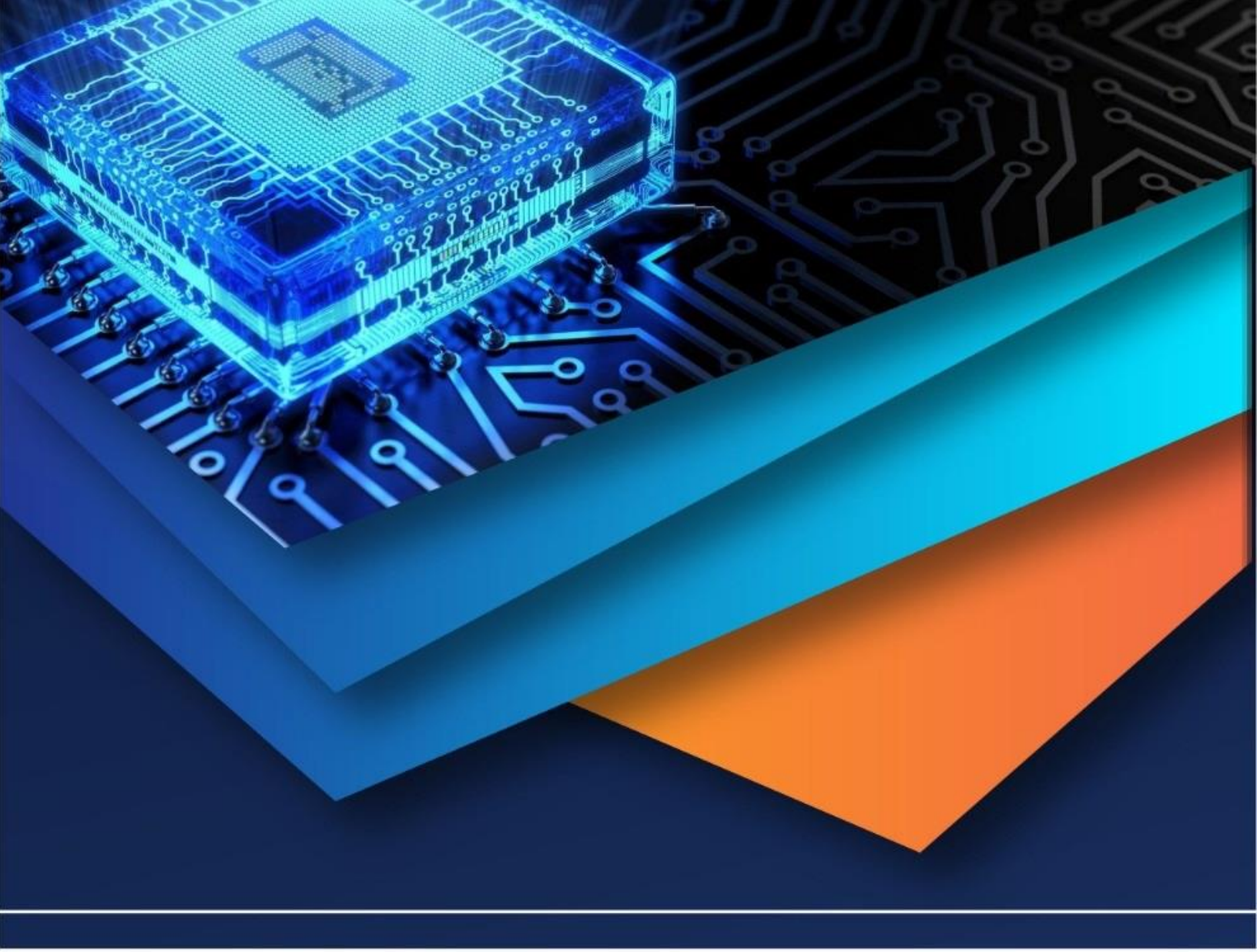
VI. CONCLUSION

In conclusion, this research successfully presents a robust and real-time touchless interaction system that combines hand gesture recognition with voice command integration, offering an intuitive and contact-free interface for Windows-based environments. The modular design of the system ensures flexibility for future enhancements, such as support for additional gestures, multi-hand interaction, or cross-platform compatibility. Experimental results validate the system's high accuracy, low latency, and user-friendly performance, demonstrating its potential for real-world deployment. Beyond convenience, this touchless interface holds significant value in domains such as assistive technology for users with mobility impairments, sterile or hygienic computing in healthcare settings, and immersive control in AR/VR environments. Overall, the research establishes a strong foundation for next-generation human–computer interaction systems that prioritize accessibility, efficiency, and user comfort.

The system's successful integration of gesture tracking and voice commands showcases the potential of multimodal interaction frameworks to transform everyday computing highly suitable for environments where hands-free control is essential.

REFERENCES

[1] On the Feasibility of Real-Time 3D Hand Tracking using Edge GPGPU Acceleration. (2018). arxiv. https://arxiv.org/pdf/1804.11256
[2] Accurate, Robust, and Flexible Real-time Hand Tracking. (2015). Microsoft Research. http://www.cs.toronto.edu/~jtaylor/papers/CHI2015-HandTracking.pdf
[3] Continuous hand gesture recognition: Benchmarks and methods. (2025). Computer Vision and marcoCVIU2025.pdf Image Understanding. https://www.eecs.ucf.edu/~jjl/pubs/
[4] Fast-Tracking Hand Gesture Recognition AI Applications with Pretrained Models from NGC. (2025). NVIDIA.https://developer.nvidia.com/blog/fast-tracking-hand-gesture recognition-ai-applications-with-pretrained-models-from-ngc/
[5] Robust Hand Gesture Recognition Using HOG Features and Machine Learning. (2024). Sohag Journal of Science. article_346414_d94ceb0c1ef700044dda35299a48d760.pdf Institute https://sjsci.journals.ekb.eg/
[6] Using Haar Cascade for Object Detection. Machine Learning Mastery. https:// www.machinelearningmastery.com/using-haar-cascade-for-object-detection/
[7] Hand Gesture Recognition with Two Stage Approach Using Transfer Learning. (2023). arXiv. https://arxiv.org/abs/2309.11610
[8] Gesture-Based Interfaces: Designing for Touchless UX. (2023). Medium. https:// medium.com/@marketingtd64/gesture-based-interfaces-designing-for-touchless ux809a8b131705



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY