iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78615

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 14 Issue III Mar 2026- Available at www.ijraset.com

Human Activity Recognition

Akhilesh Nishad[1], Aditya Rai[2], Abhinav Tripathi[3], Mr. Satish Kumar[4]
[1, 2, 3, 4]*B.Tech 4th year students, Department of Computer Science and Engineering(Artificial Intelligence and Machine Learning), Buddha Institute of Technology, Gorakhpur, UP, India*
[5]*Asst. Prof., Department of Computer Science and Engineering, Buddha Institute of Technology, Gorakhpur, UP, India*

Abstract: ***This project presents a dual-mode system that integrates Human Activity Recognition (HAR) with Hand and Eye Gesture Control using Python. The system is designed to ensure mutual exclusivity: when HAR is active, gesture control pauses, and when gesture control is active, HAR pauses. This approach eliminates conflicts between simultaneous recognition tasks and provides a seamless user experience. The gesture control module leverages OpenCV and Mediapipe to detect hand movements (scroll, zoom, cursor control) and eye gestures (blink, gaze navigation). The activity recognition module classifies human activities such as Sitting, Standing, Walking, Talking, and Laughing using machine learning techniques. An integration layer manages switching between modules, ensuring real-time responsiveness and reliability. The proposed system enhances human-computer interaction by offering natural, intuitive control mechanisms. It is lightweight, cost-effective, and scalable, making it suitable for applications in desktop control, multimedia navigation, accessibility tools, and healthcare monitoring.***
Keyword: Human Activity Recognition (HAR), Gesture Control , OpenCV, Mediapipe, Real-time Interaction.

I. INTRODUCTION

A. Background and Motivation

In the modern era of computing, Human activity recognition and gesture control system has evolved far beyond the traditional keyboard and mouse. With the rapid advancement of artificial intelligence (AI), computer vision, and machine learning, researchers are increasingly exploring natural and intuitive ways for humans to interact with machines. Gesture recognition and activity recognition are two such domains that aim to bridge the gap between human intention and machine response.

Traditional input devices often limit accessibility and efficiency, particularly for users with physical disabilities or those engaged in multitasking environments. Gesture-based control systems, on the other hand, provide a touch-free, intuitive, and user-friendly interface. By leveraging computer vision frameworks such as OpenCV and MediaPipe, it becomes possible to detect hand movements, facial gestures, and other human activities in real time, thereby enabling seamless control of digital environments.

The motivation behind this project lies in creating a robust, scalable, and user-centric system that allows individuals to control their computer cursor, perform zoom in/out operations, open and close files, and adjust system volume — all through natural gestures such as hand movements and eye blinks. This not only enhances convenience but also contributes to accessibility, productivity, and innovation in HCI.

B. Problem Statement

Despite significant progress in gesture recognition technologies, existing systems often suffer from limitations such as:

- High latency in gesture detection.
- False triggers due to environmental noise or unintended movements.
- Limited gesture vocabulary, restricting the scope of user interaction.
- Complex hardware requirements, making systems less practical for everyday use.

This project aims to overcome these challenges by designing a Python-based solution using **OpenCV** for image processing and **MediaPipe** for efficient gesture tracking. The system is lightweight, requires only a standard webcam, and is capable of recognizing multiple gestures and activities with high accuracy.

C. Objectives of the Project

The primary objectives of the project are:

- To design and implement a gesture control system that enables cursor movement through hand tracking.
- To integrate eye-based activity recognition for zoom in/zoom out functionality.
- To allow file operations (open/close) using simple eye gestures such as blinking.

- To provide volume control through hand gestures, ensuring smooth multimedia interaction.
- To achieve real-time performance with minimal latency and high accuracy using Python, OpenCV, and MediaPipe.
- To ensure the system is scalable and extensible, allowing future integration of additional gestures and activities.

D. Significance and Applications

The proposed system has wide-ranging applications across multiple domains:

- Accessibility: Assisting individuals with mobility impairments by providing touch-free computer control.
- Productivity: Enabling professionals to multitask efficiently without relying on traditional input devices.
- Gaming and Entertainment: Offering immersive experiences through gesture-based controls.
- Education and Presentations: Allowing teachers and presenters to interact with digital content seamlessly.
- Healthcare and Rehabilitation: Supporting patients in physical therapy by monitoring and recognizing activities.

By combining gesture recognition with activity detection, the project contributes to the broader vision of **natural user interfaces (NUIs)**, where technology adapts to human behavior rather than the other way around.

E. Technical Framework

The system is implemented in **Python**, utilizing the following frameworks and libraries:

- OpenCV: For image acquisition, preprocessing, and computer vision operations.
- MediaPipe: For efficient hand tracking, palm detection, and facial landmark recognition.
- Custom Python Modules: For gesture-action mapping, debounce logic, and activity recognition.

The architecture follows a **modular design**, ensuring that each gesture (e.g., cursor movement, zoom, volume control) is handled by a dedicated module. This makes the system extensible and easy to maintain.

F. Research Context

Human activity recognition (HAR) has been a major research area in computer vision, with applications ranging from surveillance to healthcare. Gesture recognition, as a subset of HAR, focuses on interpreting human hand and facial movements. Previous studies have demonstrated the feasibility of gesture-based control systems, but many rely on specialized hardware such as depth sensors or motion controllers.

This project differentiates itself by using only a **standard webcam** and open-source libraries, making it cost-effective and widely accessible. The integration of both **gesture recognition** and **activity recognition** in a single system represents a step forward in creating unified, real-time HCI solutions

G. Expected Outcomes

The project is expected to deliver:

- A fully functional prototype capable of controlling cursor, zoom, file operations, and volume through gestures.
- A system that demonstrates high accuracy and low latency in real-time scenarios.
- A user-friendly interface with clear visual feedback for each recognized gesture.
- A scalable framework that can be extended to include additional activities such as “Talking” or “Laughing” recognition.

H. Societal Relevance and Accessibility

- One of the most compelling aspects of gesture-based control systems is their potential to improve digital accessibility. For individuals with physical disabilities or motor impairments, traditional input devices may pose significant challenges. A gesture-based system provides an alternative that is both inclusive and empowering.
- Beyond accessibility, such systems also enhance productivity in professional environments. Imagine a teacher controlling slides during a lecture without touching a keyboard, or a surgeon navigating digital records in an operating room without physical contact. These scenarios highlight the **social relevance** of the project and its potential to transform everyday interactions with technology.

I. Technical Challenges and Innovation:

Developing a reliable gesture control system is not without challenges. Some of the key technical hurdles include:

- Gesture ambiguity: Different users may perform the same gesture differently, requiring robust detection algorithms.

- Environmental variability: Lighting conditions, background noise, and camera quality can affect accuracy.
- Real-time performance: The system must process video streams quickly to avoid lag.
- False positives: Preventing unintended actions triggered by accidental gestures.

This project addresses these challenges by leveraging MediaPipe's efficient landmark detection and OpenCV's image processing capabilities. Python provides the flexibility to implement modular code structures, ensuring scalability and extensibility. The innovation lies in combining multiple modalities — hand gestures, eye blinks, and activity recognition — into a single, unified framework

J. Conclusion:

The proposed Human Activity Recognition and Gesture Control System represents a significant advancement in the field of human–computer interaction. By combining computer vision, machine learning, and intuitive gesture mapping, the project aims to create a practical, accessible, and innovative solution for everyday computing tasks. Its emphasis on modularity, scalability, and user-centric design ensures that the system can evolve with future technological developments, ultimately contributing to the broader vision of intelligent, adaptive, and natural user interfaces.

II. LITERATURE REVIEW AND FEASIBILITY STUDY

A. Evolution of Human Activity Recognition and Gesture Control:

Human Activity Recognition (HAR) has evolved significantly over the past few decades. Early systems relied on wearable sensors such as accelerometers and gyroscopes to capture motion data. These approaches provided accurate measurements but required users to wear additional devices, which limited practicality and accessibility. As computer vision advanced, researchers began exploring vision-based recognition, using cameras to detect and classify human movements. Techniques like background subtraction, optical flow, and contour detection allowed systems to track gestures without external sensors. However, these methods were highly sensitive to environmental conditions such as lighting variations and background noise, which reduced their reliability in real-world applications. The integration of machine learning algorithms marked a major turning point. Models such as Support Vector Machines (SVMs), Hidden Markov Models (HMMs), and K- Nearest Neighbors (k-NN) were applied to classify gestures and activities. While effective for small datasets, these algorithms struggled with complex, real-time recognition tasks. The emergence of deep learning further advanced the field.

Convolutional Neural Networks (CNNs) excelled at extracting spatial features from images, while Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks captured temporal dependencies in video sequences. These models achieved state-of-the-art accuracy but required large datasets and high computational resources, making them less practical for lightweight, real-time systems.

B. Modern Frameworks and Research Gaps

Recent developments in open-source frameworks have transformed HAR and gesture recognition. OpenCV provides robust tools for image processing, object detection, and tracking, while MediaPipe offers efficient pipelines for real-time hand tracking, facial landmark detection, and pose estimation. These frameworks are lightweight, accessible, and optimized for real-time performance, making them ideal for practical applications that require only a standard webcam. Despite these advancements, gaps remain in existing research. Many systems focus exclusively on hand gestures or facial gestures, but few integrate multiple modalities into a single framework. Accessibility and low-cost implementation are also underexplored, as some solutions depend on specialized hardware like Kinect or Leap Motion. Furthermore, ensuring real-time performance with minimal latency remains a challenge. This project addresses these gaps by combining hand gesture recognition, eye-based activity recognition, and human activity detection into one unified Python-based system using OpenCV and MediaPipe. By integrating multiple modalities, the system enhances usability, accessibility, and scalability, contributing to the broader vision of natural user interfaces (NUIs).

C. Technical and Economic Feasibility

The project is technically feasible because it uses Python, OpenCV, and MediaPipe— all open-source tools optimized for real-time performance. The hardware requirements are minimal, needing only a standard webcam and a mid-range computer. The modular design ensures scalability, allowing new gestures and activities to be added easily. Challenges such as false triggers and environmental variability are addressed through debounce logic and robust gesture-action mapping. Economically, the system is highly viable since it avoids costly sensors or specialized devices, making it affordable for students, researchers, and professionals.

D. Operational and Social Feasibility

Operationally, the system is practical and user-friendly. The gestures chosen — hand movements for cursor control, eye blinks for zooming, and hand gestures for volume adjustment — are intuitive and easy to perform. Real-time detection ensures smooth operation with minimal lag, while modular error handling improves reliability.

Socially, the system has strong relevance. It enhances accessibility for individuals with physical disabilities, aligns with modern demand for touch-free interfaces, and promotes inclusivity in diverse environments such as healthcare, education, and entertainment. By offering a low-cost, intuitive, and scalable solution, the project demonstrates both immediate usability and long-term societal impact.

Beyond technical, economic, and social considerations, the project is also environmentally and ethically feasible. Since it relies on existing hardware and open-source software, it minimizes electronic waste and avoids the need for specialized devices that could increase environmental burden. Ethically, the system promotes inclusivity by enabling touch-free interaction for users with disabilities,

ensuring that technology is accessible to a wider population. It also respects privacy, as the system processes video input locally without requiring cloud-based storage or external servers, thereby reducing risks of data misuse. This balance of sustainability, inclusivity, and privacy makes the project ethically responsible and environmentally conscious. Since the project uses lightweight frameworks like OpenCV and MediaPipe, it consumes less computational power than deep learning models requiring GPUs.

III. METHODOLOGY / PLANNING OF WORK

A. Step-by-Step Approach to Achieve Project Objectives

The project is executed through a carefully structured sequence of steps, each designed to progressively move closer to the final objectives

1) .Requirement Analysis and Objective Definition: At the outset, the scope of the project is defined in detail. This involves identifying the specific gestures and activities that the system must recognize: cursor control through hand movements, zoom in/out via eye blinks, file operations, and volume adjustment. Each requirement is documented, ensuring clarity of purpose and alignment with user expectations. This stage also includes analyzing potential challenges such as lighting variations, false triggers, and latency issues, so that solutions can be planned in advance.

2) System Design and Architecture Planning: A modular architecture is designed to divide the system into independent components. Each functionality — cursor control, zoom, file operations, and volume adjustment — is treated as a separate module. This modularity ensures scalability, meaning new gestures or activities can be added later without disrupting the existing framework. The architecture also defines how OpenCV will handle image preprocessing, how MediaPipe will detect landmarks, and how Python modules will map gestures to actions.

3) Data Acquisition and Preprocessing: Input data is captured using a standard webcam. Preprocessing is critical to ensure accuracy and robustness. Frames are resized to reduce computational load, converted into suitable color spaces (e.g., RGB or grayscale), and filtered to remove noise. These steps prepare the raw video feed for reliable gesture detection. By standardizing input, the system becomes more resilient to environmental variations such as lighting changes or background clutter

4) Gesture and Activity Detection: MediaPipe is employed to detect hand landmarks, palm positions, and facial features. For example, fingertip positions are tracked to interpret cursor movement, while eye landmarks are analyzed to detect blinks for zoom operations. Activity recognition modules are integrated to identify states such as “eye open,” “eye closed,” or “hand raised.” This step transforms raw video data into meaningful signals that can be mapped to user actions.

5) Gesture-Action Mapping and Control Logic: A dictionary of gestures and their corresponding actions is created. For instance, hand movement maps to cursor control, eye blinks map to zoom functions, and hand gestures map to volume control. Debounce logic is implemented to prevent false triggers, ensuring that accidental movements do not cause unintended actions. This mapping is the core of the system, translating human gestures into computer commands.

6) Integration of Modules: All modules are integrated into a unified Python framework. Integration ensures that multiple gestures can be recognized simultaneously without conflict. Synchronization between gesture detection and system response is optimized to achieve real-time performance**.**

7) Testing and Validation: The system undergoes rigorous testing in diverse environments. Accuracy, latency, and reliability are measured under different lighting conditions and backgrounds. Iterative debugging and refinement are performed to improve performance. This stage ensures that the system is robust and ready for real-world use..

B. Development Strategy and Implementation Plan:

The development strategy emphasizes modularity, scalability, and user-centric design, ensuring that the system is practical, reliable, and future-ready.

1) Incremental Development: The system is built incrementally, beginning with basic gesture detection and gradually adding functionalities. This approach allows continuous testing and refinement at each stage, reducing the risk of errors in the final system.

2) Optimization for Real-Time Performance: Latency is a critical factor in gesture control. The implementation plan includes optimizing frame processing and gesture detection pipelines. Lightweight algorithms and efficient data structures are used to ensure smooth performance.

3) Error Handling and Reliability: Debounce logic, thresholding, and fallback mechanisms are incorporated to minimize false triggers. This ensures that the system remains reliable even in variable environmental conditions

4) Scalability and Extensibility: The modular design supports future expansion. New gestures (such as swipe for slide navigation) or additional activity recognition (like “Talking” or “Laughing”) can be integrated seamlessly. This ensures that the system remains relevant as user needs evolve.

5) User-Centric Testing and Iteration: Usability testing is conducted with diverse users to evaluate intuitiveness and practicality. Feedback is analyzed to refine gesture mappings and improve user experience. This iterative process ensures that the system meets real-world expectations.

6) Documentation and Presentation: Detailed documentation of the methodology, algorithms, and system architecture is prepared. .

IV. FACILITIES REQUIRED FOR PROPOSED WORK

The successful development of the Human Activity Recognition and Gesture Control System requires a combination of software tools and hardware resources. Since the project is designed to be cost-effective and accessible, the facilities are chosen to minimize expenses while maximizing efficiency and scalability

A. Software Requirements

1) Python Programming Language

Python serves as the backbone of the project. It is chosen for its simplicity,
readability, and extensive ecosystem of libraries. Python’s modular nature allows the project to be divided into independent components, such as gesture detection, activity recognition, and action mapping. Its support for scientific computing and machine learning makes it ideal for implementing computer vision tasks.

2) OpenCV (Open Source Computer Vision Library)

OpenCV is a powerful library for image processing and computer vision. It is used for acquiring video frames from the webcam, preprocessing them (resizing, color conversion, noise reduction), and performing operations such as contour detection and landmark visualization. OpenCV ensures that the raw video feed is transformed into clean, usable data for gesture recognition**.**

3) MediaPipe Framework

MediaPipe, developed by Google, provides efficient pipelines for real-time hand tracking, facial landmark detection, and pose estimation. It is lightweight and optimized for performance, making it suitable for real-time gesture recognition. In this project, MediaPipe is used to detect hand landmarks for cursor control and eye landmarks for zooming and file operations

4) NumPy and Pandas

NumPy supports numerical computations required for image processing and gesture analysis. Pandas is used to organize and analyze gesture data during testing and evaluation. Together, they provide the mathematical foundation for reliable recognition.

5) IDE / Code Editor (PyCharm, VS Code)

A robust development environment is required for coding, debugging, and testing. Jupyter Notebook is particularly useful for iterative development and visualization, while PyCharm or VS Code provide advanced debugging features.

6) Operating System (Windows/Linux):

The system is compatible with both Windows and Linux platforms. Windows is chosen for ease of use and compatibility with academic environments, while Linux provides flexibility for advanced users

B. Hardware Requirements

1) Standard Webcam

A webcam is the primary input device for capturing real-time video. It provides the raw data for gesture and activity recognition. Since the system is designed to be accessible, even a basic built-in laptop webcam is sufficient

2) Personal Computer / Laptop

The computer serves as the processing unit for running Python scripts and executing gesture-action mappings. Minimum recommended specifications include:

- Processor: Intel i5 or equivalent
- RAM: 8 GB (to handle real-time video processing smoothly)
- Storage: 256 GB or more
- Graphics: Integrated GPU sufficient; dedicated GPU optional for faster These specifications ensure that the system runs efficiently without requiring

3) Stable Power Supply and Internet Connection:

A reliable power supply is essential for uninterrupted testing and development. Internet connectivity is required for downloading libraries, frameworks, and updates, though the system itself runs locally without needing continuous internet access.

4) Optional External Devices:

- External Monitor: Useful for better visualization during testing and debugging.
- Speakers/Headphones: Required for testing volume control functionality.
- Mouse/Keyboard (for fallback): While the system is gesture-based, traditional input devices may be used during debugging or in case of system errors.

V. BENEFITS AND IMPACT OF THE PROJECT

A. Technological Benefits

The project introduces a novel approach to human–computer interaction by integrating gesture recognition and human activity recognition into a single framework. Unlike traditional input devices such as keyboards and mice, this system enables touch-free control of digital environments. The use of Python, OpenCV, and MediaPipe ensures that the system is lightweight, efficient, and capable of real-time performance. The modular design allows for easy scalability, meaning new gestures and activities can be added without disrupting the existing framework. This makes the project a valuable contribution to the field of computer vision and artificial intelligence, offering a foundation for future innovations in natural user interfaces.

B. Productivity and Efficiency

In professional environments, the system can significantly improve productivity. For example, teachers can control slides during lectures without touching a keyboard, surgeons can navigate digital records in operating rooms without physical contact, and office workers can multitask more efficiently. By reducing dependency on traditional devices, the system streamlines workflows and enhances efficiency. The ability to perform tasks such as cursor control, zooming, file operations, and volume adjustment through natural gestures reduces time and effort, making digital interaction more intuitive and seamless.

C. Societal Impact

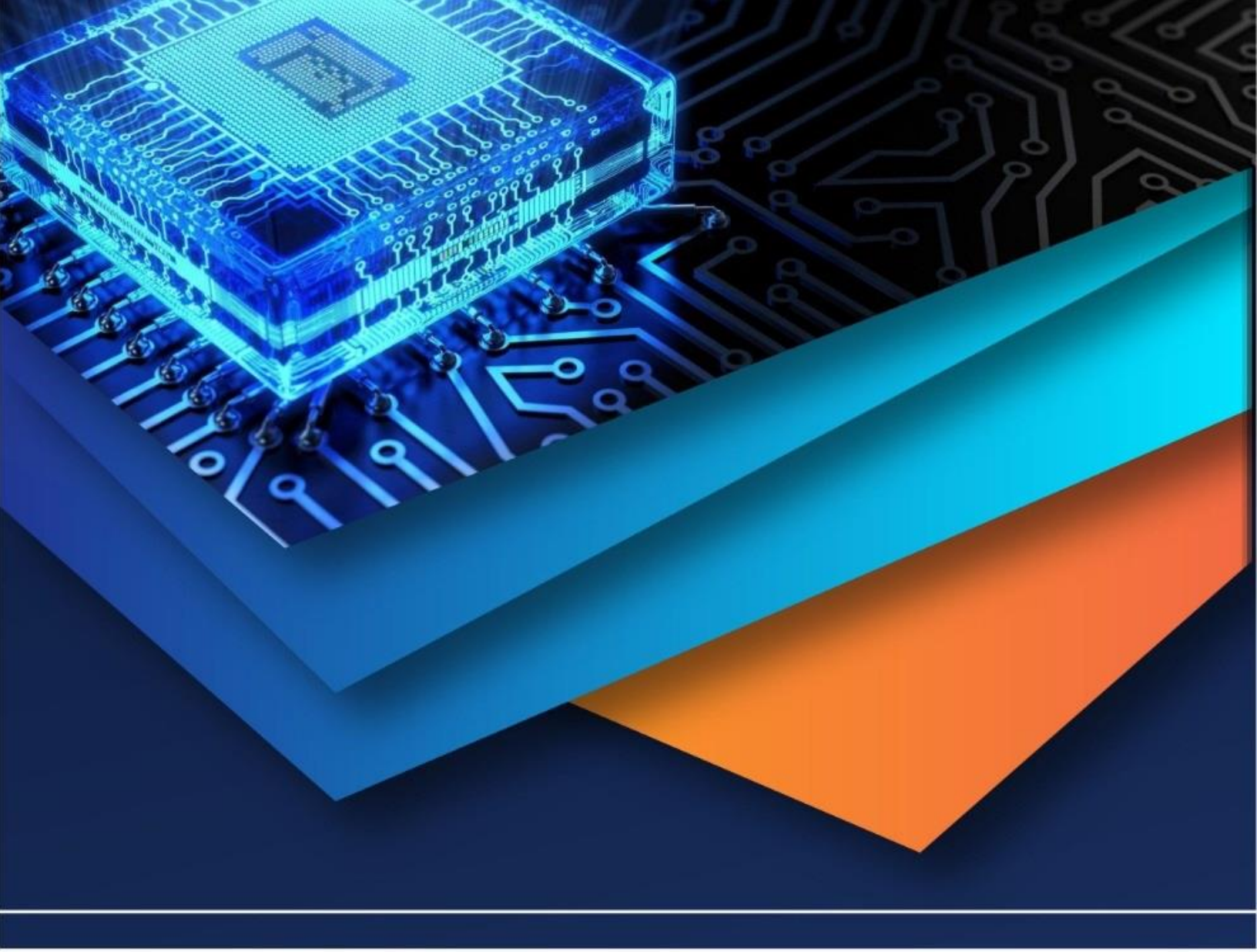
The societal impact of the project is broad and far-reaching. By making technology more intuitive, accessible, and inclusive, the system empowers individuals and communities. It aligns with global trends toward **human-centric computing**, where machines adapt to human behavior rather than requiring humans to adapt to machines. The project demonstrates how innovation can be both practical and socially responsible, contributing to a future where technology enhances everyday life in meaningful ways.

D. Academic and Research Impact

From an academic perspective, the project contributes to ongoing research in human activity recognition and gesture control. It provides a practical demonstration of how open-source frameworks can be leveraged to create real-time, scalable systems.

REFERENCES

[1] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.→ A foundational book on machine learning methods, often cited in gesture and activity recognition research.
[2] Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 77(2), 257–286. → Classic reference for sequence modeling, relevant to activity recognition.
[3] Zhang, Z. (2012). Microsoft Kinect Sensor and Its Effect. IEEE Multimedia, 19(2), 4–10. → Discusses early vision-based gesture recognition systems using Kinect.
[4] Pavlovic, V. I., Sharma, R., & Huang, T. S. (1997). Visual Interpretation of Hand Gestures for Human–Computer Interaction: A Review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 677–695. → A widely cited survey on gesture recognition.
[5] Google Developers. (2020). MediaPipe: A Framework for Building Perception Pipelines. Retrieved from https://developers.google.com/mediapipe → Official documentation for MediaPipe, the framework used in your project
[6] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools. → The original paper introducing OpenCV, a key tool in your project.
[7] Lara, O. D., & Labrador, M. A. (2013). A Survey on Human Activity Recognition Using Wearable Sensors. IEEE Communications Surveys & Tutorials, 15(3), 1192– 1209. → A comprehensive survey on HAR methods, useful for your literature review.
[8] Chen, M., et al. (2012). Human Activity Recognition Using Smartphones. Sensors, 12(2), 1157–1175. → Shows practical applications of HAR with mobile devices.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY