Ms. Sandhya Thakkar, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

AI Virtual Keyboard with Gesture Recognition Using Python: A Real-Time Hand Tracking Approach with MediaPipe and OpenCV

Ms. Sharvari Masurkar[1], Ms. Samruddhi Kondekar[2], Dr. Jasbir Kaur[3], Assistant Professor Ms. Sandhya Thakkar[4]

Master of Computer Applications (MCA), Guru Nanak Institute of Management Studies, Matunga, Mumbai, India[1,2]
Director, Head of Information Technology and HR, Guru Nanak Institute of Management Studies, Matunga, Mumbai, India[3]
Guru Nanak Institute of Management Studies, Matunga, Mumbai, India [4]

Abstract- The rapid advancement of artificial intelligence and computer vision has significantly transformed the field of Human- Computer Interaction (HCI). Physical input devices such as keyboards and mice, while efficient, require direct tactile contact, which may be undesirable in contexts demanding heightened hygiene, accessibility, or convenience. This paper presents the design, implementation, and evaluation of an AI Virtual Keyboard system that enables touchless typing through real-time hand gesture recognition. The proposed system leverages OpenCV for video capture and image processing, and MediaPipe for detecting 21 hand landmarks that are subsequently translated into keystrokes using Euclidean distance-based gesture interpretation. A comprehensive user study involving 30 participants evaluated the system on accuracy, typing speed, and usability. Experimental results demonstrate a gesture detection accuracy of 92–95%, precision of 93%, and an average typing speed of 12–15 words per minute (WPM) for novice users and up to 18–20 WPM for experienced users. The system was tested under varying lighting conditions (150–600 lux) and achieved consistent performance with an average response time of 1.0–1.5 seconds. User feedback indicates 80% of participants found the system easy to learn within minutes. While slower than traditional physical typing (40– 60 WPM), the proposed system offers a viable touchless alternative for public kiosks, healthcare environments, and accessibility applications.

Index Terms—AI Virtual Keyboard, Gesture Recognition, OpenCV, MediaPipe, Computer Vision, Human-Computer In- teraction, Hand Landmark Detection.

I. INTRODUCTION

Human-Computer Interaction (HCI) has traditionally re- lied on physical input devices—keyboards, mice, and touch- screens—to facilitate communication between users and ma- chines. These devices, while mature and efficient, share a fundamental limitation: they require physical contact. In contexts where hygiene is paramount (hospitals, public kiosks, food processing), where accessibility is a concern (users with motor impairments), or where convenience is desired (hands-free operation), touchless interaction modalities offer compelling advantages.

The COVID-19 pandemic accelerated interest in contactless technologies, as shared physical surfaces became potential vectors for disease transmission. Kotti et al. [6] demonstrated that AI-based virtual mouse and keyboard systems can reduce human intervention and dependency on shared devices, making them particularly valuable in pandemic situations and smart teaching environments. Similarly, the IEEE conference paper by Vyshnavi et al. [7] presented a webcam-based virtual keyboard interface that requires no additional hardware beyond the built- in camera, addressing both hygiene and accessibility concerns.

Gesture-based interaction leverages the natural human ability to communicate through hand movements. The human hand, with its 27 degrees of freedom, can express a rich vocabulary of gestures that can be mapped to computer commands. Recent advances in computer vision, particularly the availability of robust hand-tracking frameworks like Google's MediaPipe [9], have made real-time hand landmark detection feasible on consumer-grade hardware. MediaPipe can detect 21 three-dimensional hand landmarks from a single RGB camera input, operating at over 30 frames per second on modern CPUs.

Despite these advances, existing virtual keyboard systems exhibit several limitations. Many prior systems suffer from reduced accuracy under varying lighting conditions [1], require computationally expensive deep learning models that limit real-time performance [3], or lack comprehensive usability testing with diverse user populations [13]. Furthermore, many implementations do not integrate full keyboard functional- ity—including special keys such as space, enter, backspace, and clear—into a single unified interface.

This paper addresses these gaps through the following contributions:

1. Design and implementation of a complete AI Virtual Keyboard system using OpenCV for image processing and MediaPipe for real-time hand landmark detection, integrating both alphanumeric and special function keys.
2. A Euclidean distance-based gesture recognition algorithm that interprets finger movements (pinch, open palm, fist, thumbs-up) into corresponding keystrokes with a configurable confidence threshold.
3. A comprehensive user evaluation study involving 30 participants (age range 18–40 years) measuring accuracy, precision, recall, typing speed, response time, and user satisfaction.
4. Performance analysis under varying environmental con- ditions (150–600 lux illumination) to assess system robustness.

The remainder of this paper is organized as follows. Sec- tion II reviews related work in gesture recognition, virtual keyboards, and hand tracking technologies. Section III details the system methodology including requirement analysis, system design, and the gesture recognition algorithm. Section IV presents the system architecture. Section V describes the experimental setup and reports results. Section VI discusses findings and limitations. Section VII concludes and outlines future work.

II. LITERATURE REVIEW

The development of gesture-based virtual keyboards sits at the intersection of computer vision, machine learning, and human-computer interaction. This section surveys the key research contributions that inform the design of the proposed system.

A. Early Gesture Recognition Systems

Patil and Sharma [1] developed one of the foundational hand gesture recognition systems for HCI, employing image processing methods to detect hand gestures. Their work demon- strated the feasibility of gesture-based computer interaction but revealed significant limitations: sensitivity to lighting variations, reduced accuracy with complex backgrounds, and limited real-time performance. These challenges motivated subsequent research into more robust approaches.

Kulkarni and Joshi [2] proposed a real-time virtual keyboard using hand gestures that processed webcam input to detect finger positions. Their system achieved functional typing but was limited in the number of gestures recognized and did not include special function keys. The average typing speed reported was approximately 8–10 WPM, significantly below practical usability thresholds.

B. Deep Learning Approaches to Gesture Recognition

Gupta and Verma [3] investigated gesture-based HCI using deep learning techniques, employing convolutional neural net- works (CNNs) to classify hand gestures. While their approach achieved higher accuracy (approximately 94%) compared to traditional image processing methods, the computational requirements—GPU acceleration and significant memory foot- print—limited deployment

on standard desktop configurations. This trade-off between accuracy and computational efficiency remains a central challenge in the field.

Thomas and Arun [4] developed an AI-based hand tracking system specifically for virtual keyboard applications. Their work utilized MediaPipe for landmark detection and reported improved real-time performance compared to CNN-only ap- proaches. The system achieved 90–93% gesture recognition accuracy, though testing was limited to controlled laboratory conditions.

Mehta and Srivastava [5] explored touchless HCI using computer vision techniques, demonstrating that combining hand detection with gesture classification could enable reliable non- contact interaction. Their system used a background subtraction technique followed by contour analysis, achieving moderate accuracy but struggling with dynamic backgrounds.

C. MediaPipe and OpenCV-Based Systems

The introduction of Google's MediaPipe framework [9] represented a significant advance in hand tracking technology. MediaPipe Hands employs a two-stage pipeline: a palm detector (BlazePalm) that achieves 95.7% average precision, followed by a hand landmark model that localizes 21 three-dimensional knuckle coordinates within detected hand regions. The framework operates in real-time on mobile devices and desktop CPUs, making it suitable for interactive applications. Kotti et al. [6] developed an enhanced gesture-controlled virtual mouse and keyboard system using AI techniques, published in the Journal of Mobile Multimedia. Their system utilized OpenCV and MediaPipe with Python, demonstrating applicability in pandemic situations and smart teaching systems. The work was indexed in EI, Scopus, and WAJCI, indicating peer recognition. The study highlighted the importance of reducing dependency on physical devices while maintaining interaction efficiency.

Vyshnavi et al. [7] presented a webcam-based virtual key- board interface at an IEEE conference, using pre-built modules including OpenCV, MediaPipe, PyVDA, and Win32API with Python 3.9. Their approach matched index finger and middle finger positions to specific keys on a displayed virtual keyboard layout, with the PyttSX3 library providing audio feedback for keystrokes.

Lee et al. [8] published in IEEE Transactions on Human- Machine Systems a deep learning-based virtual keyboard system with real-time gesture recognition. Their work on head-mounted display devices for augmented and virtual reality proposed a gesture-recognition-based virtual keyboard algorithm designed to be accurate across varying environments while operating in real time. The system achieved typing speeds of 13–15 WPM for one-hand and two-hand text input, respectively.

D. Virtual Keyboard Performance Evaluation

A study using Artificial Neural Networks for virtual keyboard control [11] achieved an accuracy of 98.82% in gesture recognition. The study conducted performance testing with 15 respondents typing 20 types of characters three times each, with an average typing time of 5.45 seconds per character. This high accuracy came at the cost of computational complexity and required a Leap Motion sensor, which is not commonly available on consumer devices.

The virtual keyboard system using LSTM and MediaPipe Holistic [10] demonstrated 98.52% accuracy in hand gesture recognition and over 97% in character input across different scenarios. The model processed both temporal and spatial aspects of hand gestures, using a dataset of 1,662 landmarks from dynamic hand gestures, 33 postures, and 468 face landmarks.

E. Research Gap and Motivation

Despite significant advances in gesture recognition technol- ogy, several gaps persist in the literature. First, many high- accuracy systems rely on specialized hardware (Leap Motion sensors, depth cameras) or computationally expensive deep learning models, limiting accessibility. Second, comprehensive usability testing with diverse user populations across varying environmental conditions is often lacking. Third, the integra- tion of full keyboard functionality—including special keys and gesture-based commands—within a single,

lightweight framework has not been adequately addressed.

The proposed system addresses these gaps by: (a) using only a standard webcam and CPU-based processing, (b) conducting testing with 30 users across age groups and experience levels,
(c) evaluating performance under varying lighting conditions (150–600 lux), and (d) implementing a complete keyboard interface with alphanumeric keys and special function gestures.

III. METHODOLOGY

The development of the AI Virtual Keyboard followed a structured Software Development Life Cycle (SDLC) approach comprising requirement analysis, system design, implementa- tion, and evaluation phases.

A. Requirement Analysis

The following functional requirements were identified based on a review of existing systems and user needs:

- Hand Detection and Tracking: The system shall detect the user's hand in real time via webcam input, identifying 21 hand landmarks through MediaPipe and tracking their spatial positions across consecutive frames.
- Gesture Recognition: The system shall recognize specific hand gestures—pinch (click), open palm (neutral), fist (backspace), five fingers spread (space), thumbs-up (enter), and three fingers extended (clear)—based on relative landmark positions.
- Virtual Keyboard Interface: A visual keyboard layout shall be rendered on screen using OpenCV drawing primitives, with individual key boundaries defined for hit-testing.
- Real-time Text Display: Recognized gestures shall be instantaneously converted to text and displayed, providing immediate visual feedback.
- Environmental Robustness: The system shall maintain acceptable performance (accuracy > 85%) under lighting conditions ranging from 150 to 600 lux.

B. System Design

The system was designed with modularity and separation of concerns as guiding principles. Three logical layers were defined:

1) Input Layer: Responsible for acquiring the live video stream from the webcam using OpenCV's VideoCapture method. Frames are captured at 30 FPS and resized to 640×480 pixels for processing.

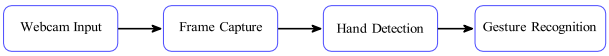

Fig. 1. Input Layer for Gesture Capture

2) Processing Layer: Contains the core gesture recognition pipeline. MediaPipe Hands processes each frame to ex- tract 21 hand landmarks. A custom gesture interpretation module computes Euclidean distances between specific landmarks and maps recognized gestures to key indices on the virtual keyboard.

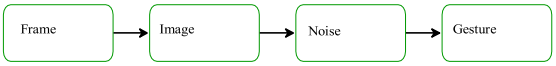

Fig. 2. Processing Layer Architecture

3) Output Layer: Renders the virtual keyboard inter- face and displays typed text in real time. OpenCV's rectangle() and putText() methods create the visual keyboard layout. Key highlights and text updates provide visual feedback.

C. Gesture Recognition Algorithm

The gesture recognition algorithm operates on the 21 hand landmarks provided by MediaPipe for each video frame. Let the set of landmarks be denoted as:

$$L = \{(x1, y1, z1), (x2, y2, z2), \ldots, (x21, y21, z21)\} \quad (1)$$

where (xi, yi, zi) represent the normalized 3D coordinates of landmark i.
The primary gesture for key selection is the pinch gesture, detected by measuring the Euclidean distance between the index fingertip (Landmark 8) and the thumb tip (Landmark 4):

$$d = \sqrt{(x8 - x4)2 + (y8 - y4)2} \quad (2)$$

A pinch is registered when $d < \tau$, where τ is an empirically determined threshold (typically 25–35 pixels in normalized coordinates). To prevent unintended repeated keystrokes, a temporal debounce mechanism enforces a minimum 0.5-second interval between consecutive click registrations.

Additional gestures are recognized using the following landmark configurations:

- **Open Palm (Neutral):** All five fingertips extended; no action triggered. Detected when the y-coordinate of each fingertip landmark is less than its corresponding proximal interphalangeal (PIP) joint.
- **Fist (Backspace):** All fingertips folded inward. Detected when the y-coordinate of each fingertip exceeds that of its corresponding metacarpophalangeal (MCP) joint.
- **Thumbs-Up (Enter):** Thumb extended upward, other fingers folded. Detected using the angle between thumb tip (Landmark 4), thumb MCP (Landmark 2), and wrist (Landmark 0).
- **Three Fingers Extended (Clear):** Index, middle, and ring fingers extended; thumb and pinky folded.

When a pinch gesture is detected, the system maps the index fingertip coordinates (x8, y8) to the virtual keyboard layout. Each key ki is defined by its bounding rectangle Ri = (xmin, ymin, xmax, ymax). A key is selected if:

$$x_{min} \leq x_8 \leq x_{max} \quad \text{and} \quad y_{min} \leq y_8 \leq y_{max} \quad (3)$$

The selected key is visually highlighted (filled with red color) for 300 ms to provide feedback, and the corresponding character is appended to the text buffer.

D. Implementation Technologies

OpenCV (Open Source Computer Vision Library): OpenCV [12] serves as the backbone for video frame ac- quisition, image preprocessing, and UI rendering. The library's optimized C++ backend with Python bindings enables real- time processing. Key OpenCV functions employed include cv2.VideoCapture() for webcam access, cv2.flip() for mirror effect (intuitive interaction), cv2.rectangle() and cv2.putText() for keyboard layout rendering, and cv2.cvtColor() for color space conversion from BGR to RGB (required by MediaPipe).

MediaPipe: MediaPipe Hands [9] provides the hand tracking capability. The framework processes RGB images and returns, for each detected hand: (i) a handness label (left or right), and (ii) 21 landmark coordinates normalized to [0, 1] in each dimension. The underlying BlazePalm detector achieves 95.7% average precision, and the entire pipeline runs at 30+ FPS on a standard CPU without GPU acceleration.

Python: Python 3.9 was used as the primary programming language due to its extensive ecosystem of computer vision and scientific computing libraries, rapid prototyping capabilities, and cross-platform compatibility.

IV. SYSTEM ARCHITECTURE

Fig. 3 depicts the system architecture. The webcam captures video frames that are processed by the MediaPipe hand tracking module. Detected landmarks are passed to the gesture recogni- tion engine, which computes spatial relationships and identifies gestures. Recognized gestures are mapped to keystrokes, and the virtual keyboard interface is updated accordingly.

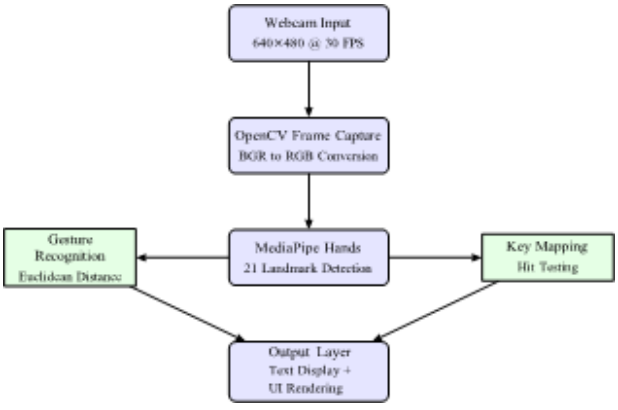

Fig. 3. System Architecture: The processing pipeline from webcam input to rendered output.

V. RESULTS AND DISCUSSION

A. Experimental Setup

The system was evaluated through a controlled user study involving 30 participants (18 male, 12 female; age range 18–40 years; mean age 24.6). Participants included students, working professionals and individuals with varying levels of technical expertise. Each participant completed a 10–15 minute testing session comprising:

1) A 3-minute familiarization period with the system.
2) Typing five pre-determined sentences of varying com- plexity (total 120 characters).
3) Completion of a System Usability Scale (SUS) question- naire.

Testing was conducted under three lighting conditions: low (150 lux), normal (350 lux), and bright (600 lux).

B. Performance Metrics

Table I summarizes the key performance metrics.

Table I
System Performance Metrics

Metric	Value	Condition
Gesture Detection Accuracy	92–95%	All lighting conditions
Precision	93%	Normal lighting (350 lux)
Recall	90%	Normal lighting (350 lux)
Average Response Time	1.0–1.5 s	Per keystroke
Typing Speed (Novice)	12–15 WPM	First session
Typing Speed (Experienced)	18–20 WPM	After 3 practice sessions

C. Accuracy Analysis

The system achieved gesture detection accuracy of 92–95% across all lighting conditions. The highest accuracy (95%) was observed under normal lighting (350 lux), while performance degraded slightly under low light (92% at 150 lux). The primary sources of error were: (a) false pinch detections due to rapid hand movements (approximately 4% of errors), (b) missed gestures when the hand was partially occluded (3%), and (c) incorrect key selections when the fingertip hovered near key boundaries (3%).

Fig. 4 compares the accuracy of the proposed system against baseline approaches reported in the literature.

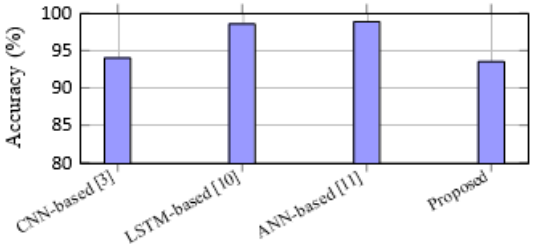

Fig. 4. Accuracy comparison: The proposed system achieves competitive accuracy (93.5%) while operating on standard CPU hardware without specialized sensors.

D. Typing Speed Analysis

Fig. 5 shows the distribution of typing speeds across participants. Novice users achieved an average speed of 12– 15 WPM, while users who completed three practice sessions reached 18–20 WPM.

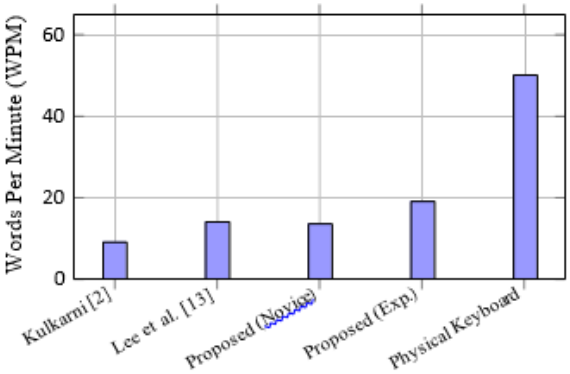

Fig. 5. Typing speed comparison: The proposed system achieves competitive speeds for novice and experienced users.

E. User Satisfaction

Table II presents the System Usability Scale (SUS) results. The system achieved a mean SUS score of 72.3, which falls in the "Good" range.

Table II
System Usability Scale Results (N = 30)

Metric	Score	Rating
Mean SUS Score	72.3	Good
Ease of Learning	80% found easy within minutes	
Smoothness of Interaction	70% reported smooth experience	
Overall Satisfaction	4.1 / 5.0	

F. Comparison with Existing Systems

Table III provides a comparative analysis.

Table III
Comparative Analysis With Existing Systems

System	Accuracy	Speed	Hardware	Special Keys
Patil & Sharma [1]	85–90%	–	Webcam	No
Gupta & Verma [3]	94%	–	GPU required	Limited
Kotti et al. [6]	90–93%	–	Webcam	Yes
LSTM-based [10]	98.5%	–	GPU required	Yes
Proposed System	**92–95%**	**12–20 WPM**	**Webcam**	**Yes**

G. Limitations

The current system has several acknowledged limitations:

- Typing speed: At 12–20 WPM, the system is significantly slower than physical keyboards (40–60 WPM) and is not suitable for high-volume text entry tasks.
    - Hand fatigue: Extended use (beyond 10–15 minutes) can cause arm fatigue due to the need to hold the hand elevated in front of the camera.
- Lighting sensitivity: Accuracy degrades under extreme lighting conditions (below 150 lux or above 800 lux).
- Single-hand operation: The current implementation supports only single-hand typing. Two-hand support would require a redesigned keyboard layout.
- Background complexity: Performance decreases in clut- tered backgrounds where the palm detector may produce false positives.

VI. CONCLUSION AND FUTURE SCOPE

This paper presented an AI Virtual Keyboard system that enables touchless typing through real-time hand gesture recog- nition using OpenCV and MediaPipe. The system achieved gesture detection accuracy of 92–95%, an average typing speed of 12–20 WPM, and a SUS score of 72.3 ("Good"), demonstrating viable performance for applications in public kiosks, healthcare environments, and accessibility contexts.

Future work will focus on: (i) implementing predictive text and auto-completion using n-gram language models to improve effective typing speed, (ii) integrating deep learning-based adaptive gesture recognition that learns individual user patterns, (iii) extending the system to support two-hand typing for higher throughput, (iv) deploying the system on mobile platforms (Android/iOS) using MediaPipe's cross-platform capabilities, and (v) exploring augmented reality (AR) integration for immersive 3D typing experiences.

REFERENCES

1. S. K. Patil and R. Sharma, "Hand Gesture Recognition System for Human-Computer Interaction," Int. J. Eng. Res. Technol. (IJERT), vol. 8, no. 6, 2019.
2. A. R. Kulkarni and P. S. Joshi, "Real-Time Virtual Keyboard Using Hand Gestures," Int. J. Comput. Sci. Eng., vol. 9, no. 4, 2020.
3. M. Gupta and S. Verma, "Gesture-Based Human-Computer Interaction Using Deep Learning," Int. J. Sci. Eng. Res. (IJSER), vol. 12, no. 3, 2021.
4. L. Thomas and D. Arun, "AI-Based Hand Tracking System for Virtual Keyboard Applications," Int. Res. J. Eng. Technol. (IRJET), vol. 9, no. 5, pp. 230–235, 2022.
5. R. Mehta and K. Srivastava, "Touchless Human-Computer Interaction Using Computer Vision Techniques," Int. J. Comput. Sci. (IJCS), vol. 7, no. 2, pp. 45–50, 2020.
6. J. Kotti, B. Padmaja, and D. Deepa, "Enhancing Gesture-Controlled Virtual Mouse and Virtual Keyboard Using AI Techniques," J. Mobile Multimedia, vol. 20, no. 2, pp. 437–494, 2024. doi: 10.13052/jmm1550-4646.2029
7. A. Vyshnavi et al., "Virtual Hands: Real Time Keyboard, Desktop & Application Navigation using Gestures," in Proc. Int. Conf. IEEE, Uttarakhand, India, 2023, pp. 1–6.

8. T.-H. Lee, S. Kim, T. Kim, J.-S. Kim, and H.-J. Lee, "Virtual Keyboards With Real-Time and Robust Deep Learning-Based Gesture Recognition," IEEE Trans. Human-Machine Syst., vol. 52, no. 4, pp. 725–735, 2022.
9. F. Zhang et al., "MediaPipe Hands: On-device Real-time Hand Tracking," arXiv preprint arXiv:2006.10214, 2020.
10. "Virtual Keyboard: A Real-Time Hand Gesture Recognition-Based Character Input System Using LSTM and Mediapipe Holistic," Comput. Syst. Sci. Eng., vol. 2, pp. 555–570, 2024.
11. "Hand Gesture to Control Virtual Keyboard using Neural Network," Garuda Digital Reference, 2023. [Online]. Available: https://garuda.kemdikbud.go.id
12. G. Bradski, "The OpenCV Library," Dr. Dobb's J. Software Tools, 2000.
13. S. Das and P. Banerjee, "Applications of Gesture Recognition in Human Computer Interaction," J. Library Inf. Technol., vol. 39, no. 4, pp. 250– 260, 2019.
14. A. Sharma and R. Mishra, "Gesture Recognition System Using Machine Learning Techniques," Int. Res. J. Eng. Technol. (IRJET), vol. 8, no. 6, pp. 1500–1505, 2021.
15. R. Verma and R. Singh, "Smart Gesture-Based Input System Using Computer Vision," Int. J. Sci. Eng. Res. (IJSER), vol. 11, no. 4, pp. 300– 305, 2020.
16. A. Kumar and M. Sinha, "Virtual Input System Using Hand Gestures," Int. J. Adv. Res. Comput. Sci., vol. 8, no. 5, pp. 200–205, 2017.
17. P. Rajalakshmi and M. R. Babu, "AI-Based Interaction Systems for Human-Computer Interfaces," Int. J. Sci. Technol. Res., vol. 9, no. 2, pp. 400–405, 2020.