Mr.P. Loganathan, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

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

Gesture Controlled Virtual Mouse Using Computer Vision

Mr.P. Loganathan[1], Beula C[2], Mathumitha M[3], Kavipriya K[4], Vishnupriya K[5]

[2,3,4,5]UG Student, Department of Computer Science and Engineering,
[1]Assistant Professor, Department of Information Technology, Tamilnadu College of Engineering, Coimbatore, India

Abstract: The rapid advancement of computer vision and machine learning has opened transformative avenues for Human-Computer Interaction (HCI). Traditional input devices such as the mouse and keyboard impose physical constraints that make them inaccessible to users with motor impairments and impractical in sterile, hazardous, or presentation contexts. This paper presents the design and implementation of a Gesture Controlled Virtual Mouse system that enables touchless, real-time cursor control through hand gestures captured by a standard RGB webcam. The proposed system leverages Google's MediaPipe Hands framework for 21-point three-dimensional hand landmark detection, OpenCV for video acquisition and visual feedback rendering, and PyAutoGUI for system-level mouse event execution. A lightweight, rule-based gesture classifier interprets finger configuration vectors to map specific hand postures to mouse actions including cursor movement, left- click, right-click, double-click, scroll, and drag operations. Coordinate smoothing via an exponential moving average filter mitigates cursor jitter, and a time-based cooldown mechanism prevents unintended repeated click events. Experimental evaluation conducted across three lighting conditions on 100 trials per gesture demonstrates an overall gesture recognition accuracy of 92.8%, with cursor movement achieving 96% accuracy. The system operates at 26 frames per second on commodity hardware, yielding an end-to-end processing latency of approximately 38 milliseconds. A System Usability Scale evaluation with 15 participants yielded a score of 76.3 out of 100, indicating good usability. The proposed system presents a cost-effective, hardware-independent alternative to conventional mouse input and holds significant potential for accessibility, healthcare, education, and industrial applications.

Keywords—Computer Vision, Hand Gesture Recognition, Virtual Mouse, MediaPipe, OpenCV, Human-Computer Interaction, PyAutoGUI, Real-Time Hand Tracking, Touchless Interface, Accessibility

I. INTRODUCTION

Human-Computer Interaction (HCI) has undergone profound transformation over recent decades. From the earliest command-line terminals to modern graphical user interfaces, each paradigm shift has aimed at reducing the cognitive and physical burden placed on users. The introduction of the computer mouse in the 1960s and its subsequent proliferation through the personal computing revolution represented a landmark advancement, enabling intuitive point-and-click navigation of graphical environments. However, the mouse remains fundamentally constrained by physical form: it demands a flat surface, a degree of fine motor dexterity, and direct contact, all of which constitute significant barriers in a range of real-world contexts.

Motor impairments, repetitive strain injuries, and neurological conditions such as Parkinson's disease and carpal tunnel syndrome affect millions of users globally, rendering conventional mouse operation difficult or impossible. Beyond the domain of accessibility, professionals in sterile medical environments, laboratory technicians, and industrial operators wearing protective equipment face practical impediments to conventional input device use. Presenters, educators, and tour guides also require hands-free interaction capabilities that conventional devices do not readily support. These unmet needs motivate the development of alternative, touchless input modalities.

Concurrent with these interaction challenges, the last decade has witnessed dramatic advances in computer

vision, embedded machine learning, and open-source development tooling. Modern webcams capable of capturing high- definition video at 30 frames per second are ubiquitous and inexpensive, shipping as standard equipment in virtually all laptops and as low-cost USB peripherals for desktop systems. Libraries such as OpenCV provide mature, cross-platform support for video acquisition and image processing, while frameworks such as Google's MediaPipe deliver pre-trained, production-grade machine learning inference pipelines capable of running efficiently on commodity CPU hardware without dedicated GPU acceleration.

This convergence of accessible hardware and powerful software creates a compelling opportunity to deliver a gesture-based virtual mouse that requires no specialised equipment beyond the standard webcam, making it practically deployable for the widest possible user population. The objective of this research is to design, implement, and evaluate a complete software-only Gesture Controlled Virtual Mouse system that captures live video from a standard RGB webcam, detects and tracks the user's hand in real time using landmark-based pose estimation, classifies hand postures into discrete gesture categories, and maps those classifications to corresponding system-level mouse actions, providing an intuitive and responsive touchless computing experience.

The system is designed with three key priorities. First, accessibility: the system must require no financial investment beyond what the user already possesses, deploying on hardware present in virtually every modern computing environment. Second, responsiveness: the end-to-end processing latency must be low enough for the interaction to feel natural and immediate, targeting operation at 25 frames per second or above. Third, robustness: the gesture recognition must maintain acceptable accuracy across realistic variations in user hand morphology, ambient lighting, background complexity, and camera-to-hand distance. These design constraints collectively define the research challenge addressed in this paper.

The remainder of this paper is organised as follows. Section II surveys related work in gesture recognition and virtual mouse systems. Section III describes the proposed system architecture. Section IV presents the methodology and algorithmic details. Section V details the implementation. Section VI reports experimental results and comparative evaluation. Section VII discusses advantages and limitations. Section VIII presents application domains. Section IX outlines future enhancement directions, and Section X concludes the paper.

III. LITERATURE SURVEY

The domain of vision-based gesture recognition for HCI has attracted sustained research attention since the early 1990s. This section reviews seminal and recent contributions, tracing the evolution from colour-glove approaches to modern landmark-based systems and identifying the limitations that the proposed work addresses.

A. Colour And Marker-Based Systems

Early gesture interfaces relied on instrumenting the user's hand with coloured markers or specialised gloves. Wang and Popovic [1] demonstrated that coloured gloves with per-finger colour coding could enable high-accuracy hand pose estimation using simple colour histogram matching. While achieving accuracy exceeding 90%, such systems require users to don specialised equipment, fundamentally compromising naturalness and practicality. Sturman and Zeltzer [2] provided a comprehensive survey of glove-based interfaces, cataloguing the trade-off between sensing fidelity and deployment friction that would motivate subsequent research toward bare-hand approaches.

B. Skin Colour Segmentation Approaches

A substantial body of work pursued hand detection through skin-colour segmentation, exploiting the distinctive chrominance signature of human skin in YCbCr or HSV colour spaces. Kakumanu et al. [3]

performed a comprehensive survey of skin detection methods, establishing that Gaussian mixture models trained on diverse skin-tone datasets achieved superior robustness to simple threshold-based methods. However, all skin-segmentation approaches share a fundamental vulnerability to illumination variation and background clutter containing skin-coloured regions, limiting their practical deployment reliability. Li et al. [4] proposed a virtual mouse based on skin segmentation combined with convex hull and convexity defect analysis for finger counting, achieving 85% accuracy but demonstrating significant degradation under non-uniform lighting.

C. Depth Sensor-Based Systems

The commercial introduction of structured-light depth sensors, notably the Microsoft Kinect in 2010, opened a new era of gesture recognition research. Suarez and Murphy [5] demonstrated a hand gesture recognition system using the Kinect that achieved gesture classification accuracy exceeding 96% across ten distinct gestures. The depth channel eliminates ambiguities inherent in RGB-only approaches and provides robust segmentation of the hand from the background regardless of skin tone or lighting. Keskin et al. [6] extended this line of work with a random decision forest classifier trained on depth images, achieving near-real-time performance. However, structured-light sensors carry significant cost and power consumption penalties, and their projection-based operation is incompatible with outdoor or high-ambient-light environments.

D. Cnn And Deep Learning Approaches

The deep learning revolution transformed gesture recognition as convolutional neural networks demonstrated dramatic accuracy improvements on image classification benchmarks. Molchanov et al. [7] applied 3D CNN architectures to video gesture recognition, achieving state-of- the-art performance on multiple benchmark datasets. Oyedotun and Khashman [8] demonstrated that deep neural networks could learn discriminative hand gesture features directly from raw image pixels, eliminating hand-engineered feature design. These approaches, while powerful, impose substantial computational requirements ill-suited to real-time inference on consumer CPU hardware without GPU acceleration, limiting their applicability in the target deployment context.

E. Mediapipe And Landmark-Based Approaches

Zhang et al. [9] introduced the MediaPipe Hands solution in 2020, presenting a two-stage machine learning pipeline consisting of a palm detection model followed by a hand landmark model that regresses 21 three-dimensional coordinates from a single RGB image. The pipeline achieves real-time performance on mobile CPUs, representing a significant engineering achievement. Lugaresi et al. [10] described the broader MediaPipe framework architecture, emphasising its production-readiness and cross-platform deployment capability. This foundational work enabled a wave of subsequent gesture interface research built on the MediaPipe Hands API.

F. Recent Virtual Mouse Implementations

Gupta et al. [11] implemented a virtual mouse using MediaPipe Hands and PyAutoGUI, demonstrating the feasibility of the approach but without systematic evaluation of recognition accuracy across lighting conditions. Reddy et al. [12] proposed an enhanced virtual mouse incorporating a custom CNN classifier on top of MediaPipe landmarks, achieving 95% accuracy but increasing processing time beyond real-time thresholds on standard hardware. Verma and Sharma [13] investigated the effect of hand-to-camera distance on MediaPipe landmark accuracy, finding acceptable performance in the 30 to 90 cm range relevant to desktop usage. Mistry et al. [14] explored touchless interface design for post-pandemic public kiosks, highlighting hygiene and usability considerations that motivate gesture-based systems in shared environments.

G. Accessibility-Focused Gesture Systems

Mahmoud et al. [15] developed a gesture-based assistive technology system targeting users with upper limb motor impairments, demonstrating that proximal joint gestures requiring less fine motor control than finger pinches could provide usable interaction for this demographic. Their work informs the gesture vocabulary design in the proposed system. Rekimoto [16] pioneered the concept of GestureWrist and GesturePad, lightweight sensing bands that detect wrist and forearm gestures, highlighting the broader design space of non-contact interaction modalities. Chen et al. [17] conducted a comparative usability study across voice, touch, and gesture interfaces for users with motor disabilities, finding that gesture-based systems achieved the highest task completion rates for specific manipulation tasks when gesture recognition accuracy exceeded 90%.

H. Summary Of Limitations

The literature review reveals four principal gaps motivating the present work. First, colour-glove and marker- based systems sacrifice the naturalness and spontaneity of interaction. Second, skin-segmentation systems lack the robustness required for practical deployment outside controlled laboratory conditions. Third, depth-sensor systems deliver high accuracy but impose prohibitive hardware cost and environmental constraints. Fourth, deep learning approaches achieve superior accuracy but exceed the real- time inference budget of commodity CPU hardware. The proposed system addresses all four limitations by combining the landmark-based accuracy of MediaPipe Hands with a computationally efficient rule-based classifier, delivering robust real-time performance on a standard webcam without specialised hardware or GPU acceleration.

IV. PROPOSED SYSTEM

The proposed Gesture Controlled Virtual Mouse is a software-only, vision-based system designed to replace conventional mouse input through natural hand gestures captured by an unmodified RGB webcam. The system operates entirely on the local host machine with no network dependency, cloud processing, or proprietary hardware requirement, ensuring both data privacy and broad deployability.

The system supports seven distinct interaction modalities: cursor movement driven by index fingertip position, left-click triggered by thumb-index pinch, right- click triggered by thumb-middle pinch, double-click triggered by simultaneous index and middle finger extension, scrolling driven by vertical index fingertip displacement, drag initiation on a full-fist gesture, and drag release on index extension recovery. This gesture vocabulary covers the complete functional scope of conventional mouse operation, enabling full desktop navigation without any keyboard or physical device interaction.

A. System Architecture Overview

The system architecture is structured as a four-layer processing pipeline. The Input Layer is responsible for video acquisition, implemented through OpenCV's VideoCapture interface, which provides a hardware-abstracted stream of BGR frames from any connected webcam at the configured resolution and frame rate. The Processing Layer encompasses frame preprocessing, MediaPipe Hands inference for landmark detection, finger state analysis, gesture classification, and coordinate computation. The Output Layer interfaces with the host operating system's mouse input subsystem through PyAutoGUI, translating classified gesture states into atomic mouse events. The Feedback Layer renders a real-time annotated view of the webcam feed, overlaying the detected hand skeleton, landmark indices, bounding box, and a gesture label string, providing immediate visual feedback that enables users to self-correct postures and accelerates the learning process.

B. Advantages Over Existing Systems

Compared to colour-glove systems, the proposed system requires no physical instrumentation of the user's hand, preserving interaction naturalness.

Compared to depth-sensor systems, it eliminates hardware cost entirely, relying solely on the standard webcam present in virtually all computing environments. Compared to skin-segmentation systems, MediaPipe's landmark detection demonstrates substantially superior robustness to illumination variation and background complexity. Compared to deep learning classifiers, the rule- based gesture recognition module operates with negligible CPU overhead, maintaining the processing budget required for real-time operation. Compared to prior MediaPipe-based virtual mouse implementations, the proposed system provides a more complete gesture vocabulary, a systematic accuracy evaluation, and a documented smoothing and cooldown architecture.

V. METHODOLOGY

A. Frame Acquisition And Preprocessing

The system initialises OpenCV's VideoCapture with device index 0, targeting a capture resolution of 640 by 480 pixels. Each frame retrieved from the capture buffer undergoes three preprocessing transformations before MediaPipe inference. First, the frame is horizontally flipped to produce a mirror-image view, aligning the on-screen hand movement with the user's natural proprioceptive expectation. Second, the frame is converted from OpenCV's default BGR colour ordering to the RGB ordering expected by the MediaPipe inference backend. Third, the frame's writability flag is set to False, disabling an unnecessary memory copy operation during inference and marginally improving throughput.

B. Hand Detection And Landmark Estimation

The MediaPipe Hands solution is initialised with static image mode disabled to enable the tracking pathway, which is computationally cheaper than re-running palm detection on every frame. Maximum number of hands is set to one to constrain processing overhead. Minimum detection confidence and minimum tracking confidence thresholds are both set empirically to 0.7 and 0.5 respectively, balancing false positive rejection with sensitivity to partially occluded hands. The solution returns, for each detected hand, a collection of 21 normalised landmark coordinates in the image coordinate system, where (0,0) is the top-left corner and (1,1) is the bottom-right. The z-coordinate is also provided, representing approximate depth relative to the wrist landmark.

C. Finger State Analysis

Finger extension state is determined independently for each of the five fingers using a landmark-comparison heuristic. For the four non-thumb fingers, a finger is classified as extended when the y-coordinate of its fingertip landmark is less than the y-coordinate of its proximal interphalangeal (PIP) joint by more than an empirically determined threshold of 0.02 normalised units. In the image coordinate system, smaller y-values correspond to higher positions in the frame, so this condition captures the geometric intuition that an extended finger's tip is above its base joint. The thumb uses an x-axis comparison instead due to its orthogonal axis of motion, classifying as extended when thumb tip x-coordinate is less than thumb metacarpal x-coordinate for the right hand, with the comparison inverted for the left hand.

D. Gesture Classification

Gesture classification is implemented as a rule-based finite state machine operating on the binary finger extension vector [thumb, index, middle, ring, pinky]. The mapping is implemented as an ordered conditional evaluation. Fist gesture (all fingers folded) triggers drag mode. Index-only extension maps to cursor movement. Simultaneous index and middle extension triggers double-click. Thumb-index pinch (index extended, thumb pinch distance below threshold) triggers left-click. Thumb-middle pinch triggers right-click. A state machine enforces a minimum hold duration of 150 milliseconds before a gesture transition is confirmed, preventing transient intermediate configurations during gesture transitions from generating spurious events.

E. Coordinate Mapping And Smoothing

Index fingertip coordinates in the normalised image space are mapped to screen pixel coordinates using linear interpolation via NumPy's interp function. A frame reduction boundary of 100 pixels on each edge of the 640 by 480 capture frame defines the active control zone. This boundary means that the user need only move their finger tip within a 440 by 280 pixel region to access the full extent of the screen, reducing the physical range of motion required and improving usability for users with limited mobility. The mapped coordinates are smoothed using an exponential moving average with alpha of 0.5, blending the current frame coordinate with the previous smoothed position, effectively low-pass filtering the cursor trajectory and eliminating perceptible jitter caused by natural hand tremor.

F. Click Cooldown And Scroll Mechanisms

A time-based cooldown mechanism prevents repeated click events from minor finger oscillations during sustained gesture holds. A Unix timestamp is recorded upon each click event registration, and subsequent clicks from the same gesture are suppressed until the cooldown interval of 300 milliseconds has elapsed. Scroll gesture detection tracks the signed vertical displacement of the index fingertip between consecutive frames when the scroll configuration is active. The scroll amount passed to PyAutoGUI's scroll function is proportional to the displacement magnitude, divided by a sensitivity divisor of 20, with the sign determining scroll direction.

VI. IMPLEMENTATION

A. Technology Stack

The system is implemented in Python 3.9 or later, selected for its extensive scientific computing ecosystem, strong cross-platform support, and the availability of all required libraries as maintained open-source packages. OpenCV version 4.8 or later provides video capture, image preprocessing, and real-time annotation rendering. MediaPipe version 0.10 or later supplies the pre-trained hand landmark detection pipeline. PyAutoGUI version 0.9.54 or later provides cross-platform mouse control through the operating system's native accessibility APIs on Windows, macOS, and Linux. NumPy version 1.24 or later supports numerical operations including the coordinate interpolation and distance computation. The entire dependency set is declarable in a standard requirements.txt file and installable via pip in a single command, minimising deployment friction.

B. Hand Detection Module

The Hand Detection Module wraps the MediaPipe Hands API and exposes a single process frame method that accepts a raw RGB frame and returns the detected hand landmarks collection or None if no hand is detected. The module maintains the MediaPipe Hands instance across frames, enabling the internal Kalman-filter-based tracking pathway to operate between frames, significantly reducing the computational cost compared to running full palm detection on every frame. The 21 landmarks returned are indexed zero through twenty, with landmark zero representing the wrist, landmarks one through four the thumb chain from metacarpal to tip, and each subsequent group of four representing the metacarpal, proximal, intermediate, and distal joints of the index, middle, ring, and pinky fingers respectively.

C. Gesture Recognition Module

The Gesture Recognition Module accepts the 21-landmark array and returns a string gesture label from the defined vocabulary. It first computes the extension state vector by evaluating the finger state analysis heuristic for each finger. It then computes the Euclidean pinch distance between the thumb tip landmark and each of the index and middle finger tip landmarks in normalised image space. The classification rule evaluation is implemented as an ordered conditional chain, ensuring that more specific gestures such as pinch-based clicks take precedence over general movement gestures when both conditions would

match. The module also maintains a rolling gesture history buffer of length five frames, returning the majority class across the buffer to further smooth transient misclassifications.

D. Cursor Control Module

The Cursor Control Module translates the gesture label and current fingertip coordinates into operating system mouse events via PyAutoGUI. For the move gesture, PyAutoGUI's moveTo function is called with the smoothed screen coordinates and a duration parameter of zero to minimise latency. PyAutoGUI's fail-safe feature, which raises an exception if the cursor reaches screen corners, is disabled in the production configuration to prevent unintended interruptions. Click events are generated via the corresponding click, rightClick, and doubleClick functions, each wrapped with cooldown gate logic. Drag mode is implemented by calling PyAutoGUI's mouseDown on drag gesture detection and mouseUp on release, enabling click- and-drag operations necessary for window repositioning and text selection.

E. Visual Feedback Module

The Visual Feedback Module renders the annotated webcam feed in a dedicated OpenCV window updated on each processing loop iteration. MediaPipe's drawing utilities render the full 21-landmark hand skeleton with joint and connection annotations directly on the frame buffer, providing users with immediate confirmation of hand detection and landmark quality. The detected gesture label is rendered as white text with a black outline in the top-left corner of the frame using OpenCV's putText function, enabling users to verify gesture recognition in real time. The active control zone boundary rectangle is drawn as a coloured overlay, helping users maintain their hand within the interaction region. The displayed frame is horizontally consistent with the mirror preprocessing applied to the inference input, ensuring the on-screen annotation matches the user's visual expectation.

VII. RESULTS AND DISCUSSION

A. Experimental Setup

All experiments were conducted on a laptop computer equipped with an Intel Core i5-1135G7 processor, 8 gigabytes of DDR4 RAM, and the integrated 720p webcam. The operating system was Windows 11 with Python 3.10. No GPU was used during any experiment. Testing was performed in three lighting conditions: bright (approximately 600 lux artificial overhead lighting), dim (approximately 80 lux ambient window light), and mixed (approximately 200 lux with directional shadowing across the hand). Each gesture was tested across 100 trials per lighting condition, with trials randomised across five participants of diverse hand size and skin tone.

B. Gesture Recognition Accuracy

Table I presents the gesture recognition accuracy results across all lighting conditions. The overall system accuracy of 92.8% was achieved, with cursor movement performing best at 96% average accuracy. Double-click and drag gestures exhibited the lowest accuracy, attributable to the greater geometric similarity of their landmark configurations to adjacent gesture classes and the sensitivity of the two-finger extension and full-fist conditions to partial occlusion.

TABLE I. Gesture Recognition Accuracy (%)

Gesture	Bright	Dim	Mixed	Average
Cursor Move	98%	96%	94%	96.0%
Left Click	97%	93%	91%	93.7%
Right Click	96%	92%	90%	92.7%
Double Click	95%	91%	88%	91.3%
Scroll	96%	93%	92%	93.7%
Drag Mode	93%	89%	87%	89.7%
Overall	95.8%	92.3%	90.3%	92.8%

C. System Performance

The system achieved an average frame processing rate of 26 frames per second, corresponding to an end-to-end processing latency of approximately 38 milliseconds from frame capture to mouse event execution. Peak CPU utilisation on the test hardware was 42%, leaving substantial processing headroom for concurrent application workloads. Memory footprint was measured at approximately 180 megabytes resident set size, well within the constraints of the minimum 4 gigabyte RAM requirement. These performance metrics confirm that the system meets the responsiveness criterion for practical real-time interaction.

D. Usability Evaluation

Fifteen participants were recruited for a structured usability evaluation. Participants performed four standardised tasks: launching an application from the desktop, selecting and copying a block of text, navigating a webpage, and repositioning a window using drag. The System Usability Scale questionnaire was administered after task completion. The average SUS score of 76.3 out of 100 falls in the Good category according to the standard SUS interpretation scale, above the industry average of 68. Average task completion rate across all four tasks was 87%, with window drag achieving the lowest completion rate of 73% due to the precision demands of the drag gesture hold. Average learning time was 8 minutes, and average user satisfaction rating was 4.1 out of 5.0.

E. Comparative Performance Analysis

Table II presents a comparative analysis of the proposed system against three reference implementations representing the principal alternative approaches identified in the literature survey. The proposed system achieves accuracy comparable to depth-camera solutions while eliminating all hardware cost beyond the standard webcam, and substantially outperforms skin-segmentation approaches in accuracy while matching their portability and platform compatibility.

TABLE II. Comparative System Analysis

Parameter	Depth Camera	Skin Segmentation	Proposed System
Hardware Cost	High ($150+)	Nil	Nil
Avg. Accuracy	96%	74%	92.8%
Frame Rate	30+ FPS	20-25 FPS	26 FPS
Portability	Low	High	High
Platform Support	Win/Linux	All	All
Setup Complexity	High	Low	Low

VIII. ADVANTAGES AND LIMITATIONS

A. Advantages

The proposed system delivers several distinctive advantages. The absence of any specialised hardware dependency makes the system immediately deployable on any computing environment equipped with a standard webcam, eliminating both capital expenditure and procurement friction. The software-only architecture means the system can be installed, updated, and uninstalled like any standard application, without device driver management or firmware dependencies. Local processing ensures complete user privacy, as no video data is transmitted to any remote service.

The cross-platform implementation runs without modification on Windows, macOS, and Linux, broadening the addressable user population. The gesture vocabulary covers the complete functional scope of a conventional mouse, enabling full desktop workflow support without supplementary physical input devices. The real-time visual feedback overlay

significantly reduces the learning curve, with participants achieving proficient gesture control in under ten minutes in usability testing.

B. Limitations

The system exhibits sensitivity to extreme lighting conditions. Under low illuminance below 50 lux or in scenarios with direct backlighting creating silhouette effects, MediaPipe landmark detection confidence decreases, resulting in increased gesture misclassification rates. The single-hand processing constraint, while computationally motivated, limits the interaction vocabulary compared to dual-hand capable systems. Depth perception from a monocular RGB webcam remains approximate, introducing occasional false pinch detection triggers when the thumb and finger tips are spatially aligned in the image plane but separated in depth. The current rule-based classifier, while computationally efficient, is less flexible than trained machine learning classifiers and may require manual recalibration for users with atypical hand morphology. Processing demand scales with host CPU performance, and systems below the minimum specification of Intel Core i3 at 2.0 GHz may experience frame rate degradation below the comfortable interaction threshold.

IX. APPLICATIONS

The Gesture Controlled Virtual Mouse addresses a broad range of real-world application domains. In healthcare settings, the system enables touchless interaction with medical imaging software, electronic health record systems, and diagnostic displays in operating theatres and intensive care units where physical contact with input devices introduces contamination risk. The software-only deployment eliminates the need for dedicated gesture input hardware within sterile field boundaries.

In accessibility technology, the system provides an alternative input modality for individuals with conditions including repetitive strain injury, carpal tunnel syndrome, essential tremor, and upper limb amputations. The configurable pinch threshold and smoothing parameters allow adaptation to individual motor capability profiles, and the proximal arm movement required for cursor control distributes the physical demand away from the fine motor structures most commonly affected by such conditions.

In educational environments, the system enables touchless presentation control for educators who need to advance slides, control video playback, and annotate content while maintaining eye contact and spatial positioning relative to their audience. In industrial automation contexts, factory floor operators and laboratory technicians wearing protective equipment can interact with supervisory control interfaces without removing gloves. In public kiosk and digital signage applications, touchless interaction reduces surface contamination transmission risk and provides a more hygienic user experience in high-traffic environments.

The system also holds potential in immersive media and casual gaming contexts, where gesture-based input can provide an engaging alternative interaction mode. The 38- millisecond end-to-end latency is suitable for casual applications, and projected latency reductions through algorithmic optimisation would extend applicability to more latency-sensitive interactive experiences.

X. FUTURE ENHANCEMENTS

Several enhancement directions have been identified based on system evaluation findings and usability participant feedback. In the near term, dual-hand gesture support represents the highest-priority enhancement, enabling two- handed interaction vocabulary including pinch-to-zoom, rotate, and multi-point selection gestures. Dual-hand support is architecturally feasible within the current framework by setting max num hands to two in the MediaPipe initialisation, with gesture classification extended to reason across both detected hand landmark arrays simultaneously.

Adaptive illumination preprocessing represents a second near-term enhancement target. Applying contrast-limited adaptive histogram equalisation

(CLAHE) to each captured frame prior to MediaPipe inference is expected to extend robust operation to illuminance conditions below 50 lux, addressing the principal environmental limitation identified in testing. A gesture hold-duration state machine that requires a minimum confirmed hold time of 200 milliseconds before transition would eliminate the transient misclassification errors that currently occur during gesture change sequences.

In the medium term, replacing the rule-based classifier with a lightweight convolutional neural network trained on a demographically diverse, multi-illumination dataset of hand landmark sequences is projected to increase average recognition accuracy to 97% or above while maintaining real- time inference speed. Training data collection will be designed to ensure representation across the full range of human skin tones, hand sizes, and lighting environments to prevent performance disparities. A graphical configuration dashboard would enable non-technical users to adjust gesture sensitivity parameters, smoothing alpha, and cooldown intervals without code modification.

Long-term research directions include the integration of sign language alphabet recognition to support text input for Deaf and Hard-of-Hearing users, extending the system's accessibility value proposition. Porting the processing pipeline to Android and iOS smartphone platforms would enable gesture mouse operation using the built-in front- facing camera of a mobile device, eliminating even the standard webcam hardware dependency. Edge deployment on low-power embedded hardware such as the Raspberry Pi 4 would enable gesture-controlled kiosk applications at very low system cost. Federated learning approaches for continuous accuracy improvement across a deployed user base without centralising video data represent a privacy- preserving path to ongoing model refinement.

XI. CONCLUSION

This paper has presented the design, implementation, and experimental evaluation of a Gesture Controlled Virtual Mouse system that delivers touchless, real-time cursor control through hand gestures captured by a standard RGB webcam. The system integrates the MediaPipe Hands landmark detection framework with a rule-based gesture classifier, exponential coordinate smoothing, and time-based click cooldown logic, producing a complete functional alternative to the conventional computer mouse requiring no specialised hardware beyond equipment already present in virtually all modern computing environments.

Experimental evaluation demonstrated an overall gesture recognition accuracy of 92.8% across seven gesture classes and three lighting conditions on commodity hardware, with real-time frame processing performance of 26 frames per second and an end-to-end latency of 38 milliseconds. A structured usability evaluation with 15 participants yielded a System Usability Scale score of 76.3, classified as Good, with an average task completion rate of 87% and a mean learning time of 8 minutes. Comparative analysis confirmed that the proposed system achieves accuracy substantially superior to skin-segmentation approaches and comparable to depth- sensor systems while eliminating hardware cost and matching the portability and platform compatibility of software-only alternatives.

The system's software-only architecture, cross-platform compatibility, complete local data processing, and low-cost deployment profile position it as a practical and immediately deployable solution for accessibility, healthcare, education, industrial, and public kiosk applications. Future enhancements targeting dual-hand support, adaptive illumination preprocessing, and machine learning-based gesture classification are expected to further improve accuracy and expand the interaction vocabulary, reinforcing the system's utility as a next-generation, inclusive human- computer interaction platform.

REFERENCES

1. S. Zhang, A. Ge, and L. Meng, "Adaptive hand gesture recognition system using deep convolutional neural networks for human-robot

interaction," IEEE Transactions on Human- Machine Systems, vol. 53, no. 2, pp. 245–257, Apr. 2023.
2. A. Gupta, R. Sharma, and P. Kumar, "Real-time gesture-based virtual mouse using MediaPipe and PyAutoGUI," International Journal of Computer Science and Engineering, vol. 11, no. 4, pp. 89–97, Dec. 2022.
3. S. Zhang, X. Zheng, C. Y. Liang, and H. Li, "MediaPipe hands: On-device real-time hand tracking," arXiv preprint arXiv:2006.10214, pp. 1–5, Jun. 2020.
4. R. K. Reddy, V. S. Rao, and K. Prasad, "Enhanced virtual mouse system using CNN-based gesture classifier on MediaPipe landmarks," in Proc. IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2023, pp. 1–6.
5. P. Verma and A. Sharma, "Distance-dependent accuracy analysis of MediaPipe hand landmark detection for gesture interfaces," in Proc. IEEE International Conference on Signal Processing and Communication (ICSC), Noida, India, 2022, pp. 287–292.
6. A. Mahmoud, H. Farag, and M. El-Habrouk, "Gesture-based assistive technology for users with upper limb motor impairments using RGB camera," IEEE Access, vol. 10, pp. 45612–45625, 2022.
7. C. Chen, L. Wang, and Y. Zhang, "Comparative usability study of voice, touch, and gesture interfaces for users with motor disabilities," ACM Transactions on Accessible Computing, vol. 15, no. 3, pp. 1–28, Sep. 2022.
8. P. Mistry, A. Nair, and S. Desai, "Touchless interactive kiosk systems for post-pandemic public environments using computer vision," Computers in Human Behavior, vol. 128, pp. 107–120, Mar. 2022.
9. M. Li, Y. Tang, and W. Chen, "Virtual mouse control using hand gesture recognition with convex hull and defect analysis," Journal of Visual Communication and Image Representation, vol. 84, pp. 103–115, Jul. 2022.
10. P. Molchanov, X. Yang, S. Gupta, K. Kim, and J. Kautz, "Online detection and classification of dynamic hand gestures with recurrent 3D convolutional neural networks," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 4207– 4215.
11. C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A framework for perceiving and processing reality," in Proc. Third Workshop on Computer Vision for AR/VR at IEEE/CVF Computer Vision and Pattern Recognition Conference, 2019, pp. 1–4.
12. O. K. Oyedotun and A. Khashman, "Deep learning in vision- based static hand gesture recognition," Neural Computing and Applications, vol. 28, no. 12, pp. 3941–3951, Dec. 2017.
13. P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," Pattern Recognition, vol. 40, no. 3, pp. 1106–1122, Mar. 2007.
14. C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun, "Real time hand pose estimation using depth sensors," in Consumer Depth Cameras for Computer Vision, Springer, London, 2013, pp. 119–137.
15. J. Suarez and R. Murphy, "Hand gesture recognition with depth images: A review," in Proc. IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, Paris, France, 2012, pp. 411–417.
16. J. Rekimoto, "GestureWrist and GesturePad: Unobtrusive wearable interaction devices," in Proc. Fifth International Symposium on Wearable Computers (ISWC), Zurich, Switzerland, 2001, pp. 21–27.
17. D. J. Sturman and D. Zeltzer, "A survey of glove-based input," IEEE Computer Graphics and Applications, vol. 14, no. 1, pp. 30–39, Jan. 1994.