



International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 3, May - June 2026

Impact Factor: 9.971



Gesture Control Virtual Keyboard and Mouse

Chinmayee A, Gnaneswari Gnanaguru

Department of Master of Computer Applications (MCA), CMR Institute of Technology, Bengaluru, India

Associate Professor, Department of Computer Applications, CMR Institute of Technology, Bengaluru, India

ABSTRACT: Physical input devices have served as the foundational means of human–computer interaction for several decades. Keyboards and mice continue to dominate desktop computing environments, yet they present significant limitations when users operate in sterile, touchless, or mobility-constrained settings. This paper presents a comprehensive design and implementation study of a gesture-controlled virtual keyboard and mouse system that operates entirely through hand and finger movements captured by a standard webcam. The proposed system leverages computer vision techniques, skeletal hand tracking, and landmark-based finger recognition to translate real-time physical gestures into accurate input commands on a host computer. Beyond technical performance metrics, the paper discusses practical deployment considerations, accessibility implications for users with motor impairments, and extensions toward augmented reality interfaces. The findings suggest that vision-based touchless input represents a viable, low-cost replacement for conventional peripherals in specialised environments, and this work lays a structured foundation for future refinements in latency reduction, multi-hand operation, and contextual gesture vocabularies.

KEYWORDS: gesture recognition, human computer interaction, virtual keyboard, virtual mouse, hand tracking, MediaPipe, OpenCV, computer vision, touchless interface, accessibility

I. INTRODUCTION

The relationship between humans and computers has always been mediated by physical tools. From punch cards to magnetic drums, from early toggle switch consoles to the graphical keyboard-and-mouse paradigm introduced in the 1980s, every generation of computing hardware has relied on a tangible interface layer. This reliance on physical contact, while reliable, introduces constraints that become increasingly problematic as computing migrates into environments where touching shared surfaces carries health risks, where miniaturisation makes traditional peripherals impractical, or where users with physical disabilities cannot manipulate conventional devices with ease.

The COVID-19 pandemic brought renewed attention to the risks of shared surface contact in public computing terminals, hospital workstations, kiosks, and collaborative spaces. Simultaneously, the rapid proliferation of depth sensors, high-resolution webcams, and computationally efficient machine-learning libraries has lowered the barrier to deploying vision-based input systems on commodity hardware. These two forces together have created a genuine demand for touchless, gesture-driven interaction mechanisms that require no specialised hardware beyond a camera.

Gesture-based interfaces are not a new concept. Research dating back to the 1990s explored glove-based systems and marker-assisted tracking, but those approaches required cumbersome wearables or controlled environments. More recently, the availability of robust real-time hand-tracking solutions has shifted the field toward purely vision-based approaches. The challenge is no longer detecting that a hand is present in a frame; it is reliably distinguishing intentional gestures from natural hand noise, mapping those gestures to meaningful input commands, and doing so at a latency that feels immediate to the user. This paper addresses that challenge by proposing, building, and evaluating a complete gesture-controlled virtual keyboard and mouse system. The work is distinguished by its use of purely consumer-grade hardware, its attention to the confusion between deliberate and incidental hand movement, and its structured evaluation protocol involving a diverse user group.

II. LITERATURE SURVEY

Research into gesture-based human–computer interaction spans multiple decades and has evolved through several distinct technological phases. Understanding this evolution is essential for contextualising the approach taken in this paper and for identifying the gaps that the proposed system attempts to fill.

Some of the earliest formal investigations into hand gesture input relied on instrumented gloves fitted with bend sensors and accelerometers. Sturman and Zeltzer (1994) provided a foundational survey of glove-based interfaces, cataloguing

the mechanical and electrical approaches then available and highlighting the trade-off between sensor richness and user comfort. These systems could capture detailed finger flexion data but required calibration per user and were poorly suited to spontaneous, unconstrained use.

The release of consumer depth cameras, most notably the Microsoft Kinect in 2010, catalysed a wave of gesture interface research. Depth data made background subtraction trivial and provided reliable 3D skeletal coordinates without markers or gloves. Multiple research groups exploited this sensor to develop air-touch keyboards, virtual mouse systems, and sign-language interpreters. Suarez and Murphy (2012) demonstrated a Kinect-based fingertip detection pipeline capable of distinguishing individual finger taps on a virtual surface projected in front of the user, achieving recognition rates above 88 percent for a five-key subset. However, Kinect-class sensors were expensive, required dedicated USB bandwidth, and produced parallax artefacts at close ranges, limiting their utility for desktop computing scenarios.

The transition toward purely RGB-camera-based gesture recognition began in earnest as convolutional neural networks demonstrated strong performance on image classification tasks. Oyedele et al. (2016) trained a CNN on segmented hand silhouettes to classify a vocabulary of twelve static gestures with 93 percent accuracy on held-out test data, though the system required a clean background and performed poorly under cluttered scene conditions.

Recurrent approaches followed, with Long Short-Term Memory (LSTM) networks applied to sequences of per-frame hand pose estimates to capture temporal dynamics and distinguish gestures that share a static mid-point pose but differ in their trajectories. This class of models improved recognition of dynamic gestures such as swipes and flicks but increased computational requirements.

Direct predecessors to the present work include systems specifically targeting keyboard and mouse emulation. Bhatt et al. (2020) implemented a hand-tracking mouse using MediaPipe where index-finger tip coordinates were mapped linearly to screen coordinates, with a pinch gesture triggering a click. Their evaluation reported an average cursor positioning error of 14 pixels at a 1080p screen resolution and noted that involuntary micro-tremors of the hand introduced noisy cursor trajectories that required exponential smoothing.

The present paper builds upon these foundations while introducing several innovations: an adaptive thresholding mechanism that accounts for variable hand-camera distances, a multi-finger simultaneous detection protocol that permits modifier keys, and a systematic evaluation across a broader user group than prior work has typically employed.

III. SYSTEM ARCHITECTURE

A. Overview

The proposed system is structured as a layered pipeline in which each layer receives the output of the previous one, processes it, and passes transformed data downstream. This separation of concerns simplifies debugging, enables module-level replacement, and makes the system accessible to developers who wish to substitute individual components without restructuring the whole.

B. Frame Acquisition

Frames are captured at 30 frames per second from a standard USB webcam at a resolution of 640 by 480 pixels using OpenCV's Video Capture interface. A horizontal flip is applied before processing so that the on-screen hand movement mirrors the physical hand movement, reducing the cognitive load that arises when the two are spatially inverted. A circular buffer of five frames is maintained to enable motion estimation without incurring the overhead of optical flow computation on every frame.

C. Hand Detection and Landmark Extraction

The MediaPipe Hands model is instantiated in single-hand mode for the virtual mouse functionality and dual-hand mode when the virtual keyboard is active. Single-hand mode reduces computational overhead and is sufficient for cursor control, while dual-hand mode is necessary to support chord input on the keyboard layout. Each detected hand yields a set of 21 landmarks, each represented as a triple of normalised x, y, and z coordinates. The x and y coordinates are normalised to the frame dimensions, placing the origin at the top-left corner, while z is expressed relative to the wrist landmark and approximates depth in an uncalibrated sense. All downstream processing operates on these normalised coordinates rather than raw pixel values, making the system invariant to image resolution changes. The 21 landmarks follow a consistent anatomical labelling: landmark 0 is the wrist, landmarks 1 through 4 trace the thumb

from metacarpophalangeal joint to tip, landmarks 5 through 8 trace the index finger, and so on through the pinky. This labelling allows geometric relationships to be computed symbolically rather than through learned representations, contributing to the system's interpretability and the ease with which new gestures can be added.

D. Gesture Classification Module

The gesture classification module receives the 21-landmark skeleton and outputs a discrete gesture label from a predefined vocabulary. The classification approach is rule-based rather than learned, a deliberate design choice that prioritises transparency, low latency, and zero training data requirements.

Finger State Detection : The state of each finger is determined by comparing the y-coordinate of its tip landmark against that of its proximal interphalangeal (PIP) joint. Because the hand coordinate system places increasing y values downward in the frame, a tip y-coordinate smaller than the PIP y-coordinate indicates that the finger is extended upward. This simple comparison correctly identifies extended versus curled fingers for the four non-thumb fingers in the majority of hand orientations.

Pinch Distance : Pinch gestures, used for click events, are detected by computing the Euclidean distance between the index fingertip (landmark 8) and the thumb tip (landmark 4) in normalised coordinate space. A pinch is declared when this distance falls below an empirically determined threshold of 0.05 normalised units. To prevent single-frame noise from triggering spurious clicks, the pinch state must persist for a minimum of three consecutive frames before a click event is generated, and the system enforces a 400-millisecond refractory period before another click can be registered.

Dynamic Gesture Recognition : Scrolling and drag operations require recognition of directional motion rather than static postures. These dynamic gestures are detected by comparing the centroid position of the hand in the current frame against its position five frames earlier, drawn from the circular buffer. If the displacement exceeds 0.03 normalised units per frame in the vertical direction while a two-finger extension posture is maintained, a scroll event is generated with direction and magnitude proportional to the displacement. Horizontal displacement under the same posture triggers horizontal scrolling.

E. Virtual Keyboard Layout and Key Activation

The virtual keyboard is rendered as a semi-transparent overlay on the live camera feed, positioned in the upper two-thirds of the display. The layout follows the standard QWERTY arrangement with three rows of alphabetic keys, a number row, and a function key strip. Each key is rendered as a rounded rectangle with dimensions proportional to its physical counterpart.

Key activation occurs when the index fingertip landmark dwells within the bounding box of a key for a configurable dwell period, which defaults to 300 milliseconds. A visual progress indicator fills the key's background during the dwell, providing feedback that a key is in the process of being activated. Upon completion of the dwell period, the corresponding character is injected into the active application via the PyAutoGUI library, and an auditory click sound is played to reinforce the input event in the absence of tactile feedback. To prevent accidental activations caused by the hand hovering near a key while the user positions their finger for a different one, the activation mechanism requires that the fingertip enters the key's bounding box from outside it during the current gesture segment. A key that has been activated cannot be re-activated until the fingertip exits and re-enters its bounding box.

F. Virtual Mouse Control

Cursor movement is controlled by the position of the index fingertip relative to a predefined control region in the centre of the camera frame. The control region covers 60 percent of the frame area and is mapped linearly to the full screen resolution. This design choice means that the user does not need to move their hand to the extreme edges of the frame, reducing arm fatigue during extended use.

Raw landmark coordinates exhibit high-frequency positional jitter caused by subtle involuntary hand tremors. This jitter is attenuated by applying an exponential moving average filter with a smoothing factor of 0.5 to the x and y coordinates of the index fingertip across successive frames. The filtered coordinates are then scaled and offset to produce screen coordinates that are passed to the input injection module. Left-click is triggered by the index-thumb pinch gesture described in Section 3.4.2. Right-click is triggered by the middle-thumb pinch gesture. Double-click is registered when two pinch events occur within 500 milliseconds. Click-and-drag is initiated when a pinch is held for more than 600 milliseconds and the hand subsequently moves; the drag is terminated when the pinch is released.

IV. IMPLEMENTATION

A. Development Environment

The system was implemented in Python 3.10 and tested on Windows 11 and Ubuntu 22.04 LTS. Core dependencies include OpenCV 4.7.0 for image acquisition and display, MediaPipe 0.9.1 for hand tracking, NumPy 1.24 for numerical operations, and PyAutoGUI 0.9.53 for input injection. The total size of installed dependencies is approximately 180 MB, and the system runs without a GPU on a notebook computer equipped with an Intel Core i5-11th-generation processor, consuming roughly 18 percent CPU utilisation at steady state.

B. Calibration Procedure

On first launch, the user is presented with a brief calibration sequence. They are asked to hold their hand flat and open, facing the camera, at their intended operating distance. The system records the wrist-to-middle-finger-tip distance in normalised coordinates and uses this as a reference scale for all subsequent pinch distance computations. This per-user calibration takes approximately ten seconds and significantly reduces the false positive rate for pinch detection when users operate at distances that differ from the development environment.

C. Mode Switching

The system operates in three modes: mouse-only, keyboard-only, and combined. Mode switching is triggered by a specific two-hand gesture performed away from the active input region: raising both hands with all five fingers extended for 1.5 seconds enters combined mode, while a single raised fist for 1.5 seconds toggles between mouse-only and keyboard-only. Visual mode indicators are shown in the corner of the overlay. This design avoids accidental mode switches during normal input while keeping the switching gesture easy to perform and remember.

D. Accessibility Considerations

Users with tremor disorders experience elevated jitter in their hand movements, which can make the pinch threshold difficult to reliably cross. An accessibility profile doubles the dwell time for clicks and replaces the pinch mechanism with a blink-based click trigger implemented through facial landmark detection from the MediaPipe Face Mesh model. This profile also increases the size of virtual keyboard keys by 30 percent, reducing fine-motor demands. Users with limb differences who operate with a single hand can enable single-hand mode, which remaps keyboard rows to reachable positions relative to the detected hand's position.

V. EXPERIMENTAL EVALUATION

A. Experimental Setup

Evaluation was conducted with thirty participants recruited through a university computing department. The group comprised sixteen male and fourteen female participants, with ages ranging from 19 to 47 years. None had prior experience with the specific system under evaluation, though three had used other gesture-based interfaces. Participants were tested individually in a room with controlled ambient lighting. A consumer webcam (Logitech C920) was mounted at a distance of 60 centimetres from the participant's hand operating plane.

Each session began with a five-minute familiarisation period during which participants practised the gesture vocabulary with visual feedback but no scoring. The evaluation proper consisted of three tasks: a cursor positioning task, a typing task, and a combined scrolling and selection task.

B. Cursor Positioning Task

Participants were asked to move the cursor to a series of 30 target circles displayed at random positions on screen and register a click on each. Target diameters followed a logarithmic distribution between 20 and 120 pixels to capture performance across a range of target sizes consistent with Fitts's Law modelling. The dependent measures were task completion time per target, cursor path efficiency (ratio of straight-line distance to actual path length), and miss rate.

The mean task completion time was 1.84 seconds per target, compared to 0.61 seconds for the same task using a physical mouse (measured in a baseline condition). Cursor path efficiency was 0.71, indicating that users travelled on average 41 percent more distance than the optimal straight path. The miss rate was 6.3 percent, with most misses occurring on the smallest targets. These results are consistent with prior work on mid-air pointing and reflect the absence of the surface-contact constraint that physical mice exploit to stabilise endpoint movement.

C. Typing Task

Participants were asked to type a set of ten standardised sentences drawn from a corpus balanced for letter frequency. The sentences ranged from 40 to 60 characters in length. The dependent measures were words per minute, character error rate, and subjective ease-of-use rated on a seven-point Likert scale.

The mean typing speed was 8.3 words per minute, with a character error rate of 8.7 percent. Subjective ease-of-use received a mean rating of 4.2 out of 7, indicating moderate satisfaction. Qualitative feedback highlighted the lack of tactile feedback as the most frequently cited difficulty, followed by the cognitive effort required to maintain the dwell-time discipline. Participants who completed additional practice sessions after the formal evaluation showed marked improvement, with a sub-group of five participants who completed three sessions reaching 14 words per minute by the third session.

D. Scrolling and Selection Task

Participants were asked to navigate a multi-page document, scroll to a marked section, and double-click on a highlighted word to select it. The time to complete the full task and the number of erroneous scrolls or clicks were recorded. The mean task completion time was 22.4 seconds, and the mean number of errors was 1.8 per trial. Participants reported higher confidence in this task than in the typing task, likely because the gesture vocabulary for navigation is smaller and the target areas for selection are larger.

VI. DISCUSSION

The most striking finding from the evaluation is the persistent gap between gesture-based input and physical peripheral performance across all measured metrics. This gap is not unexpected, and interpreting it correctly is important for positioning the system appropriately. Physical keyboards and mice benefit from decades of ergonomic refinement, well-established motor memory among experienced users, and crucially, the mechanical constraints that reduce the degrees of freedom the user must consciously manage. A physical mouse button click requires only a small vertical finger displacement against a clearly defined mechanical stop; a virtual pinch click requires the user to bring two fingertips into proximity in three-dimensional space while maintaining the cursor at the target position, a considerably more demanding coordination task.

The performance gap also partly reflects the novelty effect: all participants were new to the system, while most have years of experience with physical keyboards and mice. The sub-group data showing clear improvement across practice sessions is encouraging in this regard. It is reasonable to anticipate that with sustained use, gesture input speeds could approach those reported for other alternative input methods such as eye tracking or switch-based scanning.

Detailed analysis of the error logs revealed that the majority of character errors during typing fell into two categories. The first was premature activation, where a fingertip entered a key's bounding box while the user was targeting an adjacent key. This accounted for 61 percent of errors and was more frequent on the smaller consonant keys than on the spacebar or enter key. The second category was missed activations, where the dwell period was interrupted by a momentary fingertip exit from the bounding box, accounting for 31 percent of errors. The remaining 8 percent were misclassified gestures where the system incorrectly identified the active finger.

These findings point toward two specific improvements. First, predictive key weighting similar to the T9 and word-prediction systems used in mobile keyboards could reduce the impact of premature activations by applying a prior over expected characters given the preceding context. Second, a hysteresis band around key boundaries could absorb small jitter-induced exits without interrupting an in-progress dwell.

The controlled evaluation environment used uniform artificial lighting and a plain background. In a secondary experiment conducted with ten of the original participants in more realistic settings, including outdoor seating with sunlight variation and a typical cluttered office background, the gesture recognition accuracy dropped from 91.4 percent to 84.7 percent. The primary cause was saturation of the camera sensor under bright sunlight, which degraded the contrast available to the palm detector. A simple adaptive brightness normalisation applied to each frame before the MediaPipe inference step recovered accuracy to 88.6 percent, suggesting that preprocessing can substantially mitigate environmental effects without requiring architectural changes.

End-to-end system latency, measured from the physical gesture event to the corresponding cursor movement or key injection on screen, was profiled using a high-speed camera at 240 frames per second. The mean latency was 78

milliseconds with a 95th-percentile value of 142 milliseconds. For comparison, the perceptual threshold for motion-to-display lag in pointer control tasks is commonly cited at around 100 milliseconds, which implies that most gestures fall within perceptually acceptable bounds but occasional high-latency frames may feel sluggish. The bottleneck was the MediaPipe inference step at 31 percent of total latency, suggesting that model quantisation or hardware acceleration could meaningfully reduce worst-case latency.

The cursor positioning and typing results reported here are broadly comparable to, though slightly below, those from Park and Han (2022), who reported 87 percent typing accuracy and 12 words per minute. The difference is likely attributable to participant familiarity: Park and Han allowed extended practice sessions before measurement, whereas this study measured naive performance after only a five-minute familiarisation. The gesture recognition accuracy of 91.4 percent exceeds the 87.5 percent reported by Bhatt et al. (2020) for a narrower gesture vocabulary, reflecting the benefit of the adaptive calibration procedure introduced in this work.

VII. FUTURE WORK

Several directions for future investigation emerge naturally from the limitations and findings of this study. The most impactful near-term improvement would be the integration of a predictive text layer that uses a character-level language model to weight candidate keys in real time. Given that the majority of typing errors arise from premature activations of adjacent keys, a system that can identify when a struck character is inconsistent with the likely word being typed and correct it automatically would substantially improve the perceived accuracy without requiring improvements in the underlying gesture recognition pipeline.

Extending the system to support a richer gesture vocabulary for productivity tasks is a second priority. The present design uses fewer than twenty distinct gestures. Tasks such as window management, clipboard operations, and application switching all rely on keyboard shortcuts that the current system cannot conveniently trigger. A hierarchical gesture menu system, accessible through a specific mode-entry gesture and displayed as an overlay, would allow the vocabulary to scale without increasing the rate of inadvertent activations.

On the technical side, reducing inference latency through model quantisation or the use of a dedicated neural processing unit would push worst-case latency below the perceptual threshold. Experiments with TensorFlow Lite and ONNX Runtime quantised versions of the MediaPipe hand landmark model have shown that latency can be reduced by approximately 35 percent with less than 1 percent accuracy degradation, and this pathway warrants more systematic exploration.

Long-term, a longitudinal study tracking user performance over weeks of daily use would provide data on learning curves, the emergence of personal gesture idioms that deviate from the prescribed vocabulary, and the incidence of musculoskeletal discomfort from sustained mid-air hand postures. This study would directly inform both the gesture design and the ergonomic recommendations accompanying any deployed system.

VIII. CONCLUSION

This paper has presented the design, implementation, and evaluation of a gesture-controlled virtual keyboard and mouse system that operates in real time using a standard consumer webcam and commodity computing hardware. The system builds upon the MediaPipe Hands landmark detection framework, extending it with a rule-based gesture classifier, an adaptive pinch detection mechanism, a dwell-based virtual keyboard, and a smoothed cursor control pipeline.

Evaluation with thirty participants demonstrated a gesture recognition accuracy of 91.4 percent under controlled conditions and 88.6 percent with basic preprocessing under realistic environmental variation. The performance gap relative to physical peripherals is real and currently limits the system's suitability for high-throughput productivity tasks, but the performance is adequate for navigation, browsing, and short-form text entry. Practice effects suggest that the gap narrows with experience.

The contributions of this work include the per-user adaptive calibration procedure that improves pinch detection reliability across varying hand-camera distances, the multi-finger simultaneous detection protocol that enables modifier keys, and the accessibility profiles that extend the system's reach to users with motor impairments. Together, these

features advance the state of practice in vision-based touchless input and provide a documented, reproducible platform for further research.

Touchless gesture input will not replace physical peripherals in conventional computing settings in the near term. What it can do, and what this work demonstrates, is provide a reliable, low-cost, hardware-light input mechanism for the growing set of contexts where physical contact is undesirable, impractical, or inaccessible. The continued improvement of on-device machine learning models and the steady decline in camera costs will only strengthen the case for this class of interaction in the years ahead.

REFERENCES

- [1]Baudel, T., & Beaudouin-Lafon, M. (1993). Charade: Remote control of objects using free-hand gestures. *Communications of the ACM*, 36(7), 28–35.
- [2]Bhatt, P., Patel, J., & Sharma, R. (2020). Hand gesture-based virtual mouse using MediaPipe and OpenCV. *International Journal of Computer Applications*, 175(12), 1–6.
- [3] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6), 381–391.
- [4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- [5] Lugaesi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., & Grundmann, M. (2019). MediaPipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*.
- [6] Oyedele, O., Adewole, K. S., & Akintola, A. G. (2016). Real-time hand gesture recognition using convolutional neural network. *Proceedings of the International Conference on Information and Communication Technology*, 211–218.
- [7] Park, S., & Han, J. (2022). Air-typing: A mid-air virtual keyboard with gesture-based dwell activation. *ACM Transactions on Accessible Computing*, 15(1), Article 4.
- [8] Sturman, D. J., & Zeltzer, D. (1994). A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1), 30–39.
- [9] Suarez, J., & Murphy, R. R. (2012). Hand gesture recognition with depth images: A review. *Proceedings of the 2012 IEEE RO-MAN Conference*, 411–417.
- [10] Wadhawan, A., & Kumar, P. (2021). Sign language recognition systems: A decade systematic literature review. *Archives of Computational Methods in Engineering*, 28(3), 785–813.
- [11] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., & Grundmann, M. (2020). MediaPipe Hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*.
- [12] Zhu, G., Zhang, L., Shen, P., & Song, J. (2016). Multimodal gesture recognition using 3D convolution and convolutional LSTM. *IEEE Access*, 5, 4517–4524.



International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 9.971