AirBoard: A Gesture-Controlled Virtual Keyboard Interface.

Prathmesh Balsurkar[1†], Pratik Jadhav[1†], Ashmeet Arora[1†], Vinay Jain[1†], Sonali Potdar[2†]

[1,2*]School of Computer Engineering & Technology,Dr. Vishwanath Karad MIT World Peace University, Pune, Maharashtra, 411038, India.

Contributing authors: prathmeshb.code@gmail.com; pratik1.jadhav@mitwpu.edu.in; ashmeet.arora@mitpwu.edu.in; vinay.jain1@mitwpu.edu.in; sonali.potdar@mitwpu.edu.in;
[†]These authors contributed equally to this work.

Abstract

In order to replace a real keyboard, this project suggests creating an AI-powered virtual keyboard that uses a webcam to recognize hand gestures. The system monitors and detects hand landmarks in real time, using MediaPipe and OpenCV. Keystrokes are detected by mapping fingertip positions to corresponding key areas on a virtual keyboard interface superimposed on the screen. When a user taps in the air aligned with the position of a key, the system recognizes a valid input and displays text in real time. With potential uses in contactless environments, assistive technologies for physically disabled people, and augmented/virtual reality systems, the proposed method aims to enhance touchless typing interfaces. This approach provides a novel, economical solution for human–computer interaction based on vision-based gesture recognition.

Keywords: Virtual Keyboard, Hand Gesture Recognition, OpenCV, MediaPipe, Human–Computer Interaction, Touchless Interface, Real-Time Hand Tracking, Augmented Reality (AR), Virtual Reality (VR), Assistive Technology

1 Introduction

Advances in AI and computer vision have made new ways of interacting with computers possible in recent years. One such novelty is touchless interfaces, without the need for physical input devices. This suggests a virtual keyboard (driven by AI), which, via a webcam, would instantly identify hand motions without the use of a traditional keyboard. Given the recent growth of AR and VR technologies, hand gesture recognition has become an exciting new technology field [1].

The proposed method contributes to better accessibility and ease, offering a novel, vision-based, and low-cost touchless typing solution. Contactless settings, assistive technologies for people suffering from physical impairments, and AR/VR systems are just a few of the domains in which it may find employment. Its applications range over several areas, including touchless settings, support technologies for people with physical impairments, and augmented or virtual reality (AR/VR) platforms. Gesture-based virtual drawing tools demonstrate the potential of computer-vision systems to offer natural, contactless interaction methods in modern HCI [6].

2 System Design / Proposed System

2.1 System Overview

The proposed system captures real-time video from a webcam, detects the user's hand, extracts key landmarks using MediaPipe, and maps fingertip positions to corresponding regions of a virtual keyboard rendered on the screen. A keystroke is registered when the index fingertip performs a "tap" gesture near a key region, after which the corresponding character is displayed in real time.

The overall processing pipeline includes:

1. Video capture using OpenCV,
2. Hand landmark detection using MediaPipe Hands,
3. Key region mapping and gesture detection,
4. Virtual keyboard rendering,
5. Text rendering and real-time output display.

2.2 Tools and Technologies Used

Table 1 summarizes the tools and technologies used in the system.

Table 1: Tools and Technologies Used

Component	**Description**
Programming Language	Python
Libraries / Frameworks	OpenCV, MediaPipe, NumPy
Hardware Requirements	Standard webcam, PC/Laptop with 4 GB RAM
Operating Environment	Windows / Linux
Development Tools	Jupyter Notebook / VS Code

OpenCV handles webcam streaming, frame preprocessing, and the rendering of the virtual keyboard and text overlays. MediaPipe Hands provides robust and real-time detection of 21 hand landmarks. NumPy is used for numerical operations and coordinate processing.

2.3 System Architecture

The architecture of the AirBoard system consists of several interconnected modules responsible for gesture detection and keyboard input generation:

- **Input Module:** Captures live video frames from the webcam.
- **Processing Module:** Converts frames to RGB, extracts hand landmarks using MediaPipe, and determines fingertip positions.
- **Gesture Recognition Module:** Detects tap gestures using the distance between the thumb tip (Landmark 4) and index fingertip (Landmark 8) or downward fingertip motion.
- **Virtual Keyboard Module:** Draws keyboard keys using OpenCV rectangles, updates visual states (hover/press).
- **Output Module:** Displays the typed text below the virtual keyboard in real time.

Fig. 1: End-to-End Processing Pipeline of the Vision-Based Virtual Keyboard System.

2.4 Hand Landmark Detection

- MediaPipe Hands essentially works by analyzing each frame and providing 21 landmark points for the traced hand. Out of these points, three are mainly used:
- Index finger tip (Landmark 8): The main pointer by which virtual keys are selected
- Thumb tip (Landmark 4): To recognize pinch-based taps, this is the finger used
- Wrist (Landmark 0): It acts as positional stability and an origin point for calibration
- Since the hand landmarks are relative to the width and height of the frame, the gestures can be correctly mapped.

2.5 Virtual Keyboard Rendering

- The virtual keyboard is constructed on the display with the help of OpenCV functions:
- A rectangle is used to symbolize every key; the size of each rectangle is constant
- The keys, being rectangles, are saved as a list or dictionary with their coordinates (x, y, w, h)

- The letters (A–Z, 0–9, special keys) are written with the help of the function cv2.putText()
- By using the semi-transparent overlay, the keyboard can be seen as if it is placed over the camera feed.

2.6 Gesture Recognition and Tap Detection

Two methods of recognizing tap are implemented:

1. Depth-Based Tap

- The tap is the result of the index fingertip moving downwards beyond the threshold.

2. Pinch-Based Tap

- The recognition of a tap is carried out if the distance between:
- Landmark 8 (Index Tip)
- Landmark 4: Thumb Tip
- is shorter than a certain threshold.

An interval method is employed for debouncing so as to prevent accidental multiple detections and make sure that a single keypress per gesture is performed. Studies have shown that pinch gestures can help in speeding up virtual keyboard input systems and also in reducing the delay time [4].

2.7 Mapping Gestures to Keyboard Keys

In each frame:

- The (x, y) coordinates of the index finger tip are used to check the location of all keys.
- If the finger tip is found inside a key area and a tap is recognized:
- The text buffer is updated with the new character.
- A user sees the key that has been pressed visually which is done through highlighting the pressed key.

2.8 Displaying Text Output

- The typed characters are shown either at the top or the bottom of the frame in real time.
- Similar to a typing effect, the display keeps on updating by itself.

2.9 Optional Enhancements

The developed system can perform the following functions:

- Auto-spacing
- Backspace key detection
- Integration with external applications for real typing simulation

2.10 Algorithm Workflow

The AirBoard gesture-controlled virtual keyboard system's vertical, image-based workflow is depicted in the figure. In addition to a feedback loop that permits continuous real-time interaction, it demonstrates the sequential flow from library initialization and camera frame capture to hand detection, virtual keyboard rendering, gesture recognition, keystroke registration, and output display.

The workflow:

1. Setting up the necessary libraries and modules for hand tracking and video processing
2. Live video frames are continuously captured by the camera
3. Key landmarks are extracted and the user's hand is detected
4. The video feed's rendering of the virtual keyboard interface
5. Identification of user gestures for key selection and interaction
6. Keystroke registration based on gestures detected
7. Displaying the generated text output in real time
8. Feedback loop facilitating smooth and ongoing communication

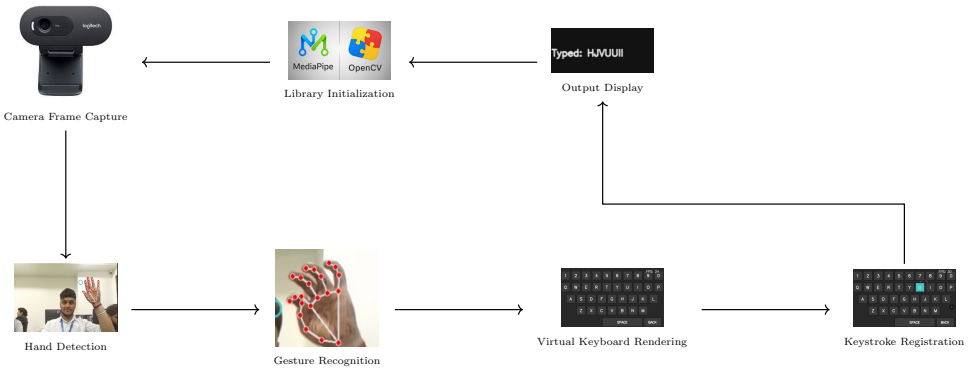

Fig. 2: Logical Two-Row Workflow of the Virtual Keyboard System

3 Implementation

The AirBoard implementation: the Gesture-Controlled Virtual Keyboard Interface has been broken down into various modules, which have been integrated to perform real-time gesture tracking and text input functionalities. The implementation language is Python, and OpenCV is used for video processing, while MediaPipe Hands is used for the extraction of hand landmarks.

3.1 System Setup and Initialization

- The steps of initial setup include starting the app:

- The camera is set up with the help of OpenCV's VideoCapture()
- MediaPipe Hands Module is set up for 21-point tracking
- Programming constructs to store keyboard key positions and text output buffer are set up
- The video for each frame is captured in BGR format, converted to RGB, and then fed to the MediaPipe model.

3.2 Hand Landmark Detection

- MediaPipe Hands does that by taking each frame for input and giving back 21 landmark coordinate points for the hand that it found. Key landmarks used:
- Index finger tip (Landmark 8): The main pointer that helps in selecting virtual keys
- Thumb tip (Landmark 4): Used to detect tap actions based on pinching
- Wrist (Landmark 0): Gives the positional stability and calibration reference
- The landmarks are relative to the frame width and height; hence, accurate gesture mapping can be done.

3.3 Virtual Keyboard Rendering

- OpenCV functions are used to draw the virtual keyboard on the display:
- Every key is shown as a rectangle; each of them has a fixed width and height.
- The keys are saved in a list or dictionary with coordinates (x, y, w, h)
- Labels (A–Z, 0–9, special keys) are drawn with the help of cv2.putText()
- The semi-transparent overlay allows the keyboard to be visually placed on top of the camera feed.

3.4 Gesture Recognition and Tap Detection

There are two methods for tap recognition that have been put in place:

3.4.1 Depth-Based Tap

- If the index fingertip is moved downwards beyond a certain threshold, a tap is thereby indicated.

3.4.2 Pinch-Based Tap

- Recognition of a tap if the distance between:
- Landmark 8 (Index Tip)
- Landmark 4: Thumb Tip
- is less than a predetermined threshold.

This debouncing method is implemented to prevent the occurrences of the accidental multiple detections and thus ensure that every gesture results in only one keypress. The use of pinch gestures has been shown to improve virtual keyboard input systems' performance and reduce the input delay [4].

3.5 Mapping Gestures to Keyboard Keys

In every frame:

- Index finger tip coordinates (x, y) are used to find the key that encloses these points.
- If the point locating the finger tip is inside a key region and a tap is detected:
- The corresponding character is added to the text buffer.
- The pressed key can be seen visually highlighted by the user for feedback.

3.6 Displaying Text Output

- The characters typed can be seen either in the top or bottom part of the frame and that happens in real-time.
- The display works independently from the main program and is therefore always up to date just like a typing effect.

3.7 Optional Enhancements

Such a system can provide:

- Auto-spacing
- Backspace key detection
- Real typing simulation through external application integration

4 Results and Discussion

This section presents the experimental results obtained from the implementation of the proposed *AirBoard: Gesture-Based Virtual Keyboard Interface* and discusses its performance, usability, and limitations. The system was evaluated in a real-time environment using a standard webcam, without any external sensors or wearable devices.

4.1 Gesture Detection and Tracking Performance

The module for recognizing gestural activities through the use of MediaPipe Hand Tracking and the OpenCV library recognized the hand gesture landmarks effectively. This allowed the tracking of 21 keypoints on the hands per frame, which helped the program to calculate the position of the fingers effectively. The program maintained an average frame rate of 24-30 FPS.

The pinch gesture was detected reliably based on the Euclidean distance between the thumb tip and index fingertip. The gesture functioned as the main input interface for key selection. The reliability of the tracking system allowed the performance of the gesture recognition to remain unaffected despite an average speed of the hand and background noises.

4.2 Virtual Keyboard Interaction Results

The virtual keyboard that functions with a gesture was able to allow characters to be typed by hovering over a character and performing a pinch gesture on it, leading to

the selection of the character. Experimental sessions gave evidence that users could input characters with reasonable accuracy after a little practice.

As seen in Figure 3, key typing statistics collected in the evaluation showed successful interaction. In an evaluation test situation, the system logged 59 keys in 7 interaction sessions with an average of 8.4 keys per session. The average duration per session was about 2.45 minutes, establishing the capability of continuous interaction through mid-air gestures.

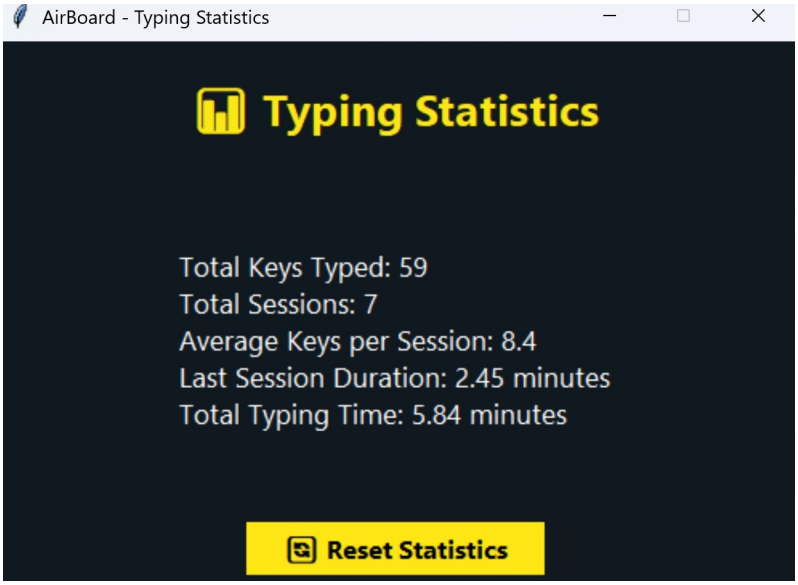

Fig. 3: Typing statistics captured during AirBoard usage, showing key press count, session duration, and interaction frequency.

4.3 AirDraw Mode Evaluation

Through the use of the tracked index finger, one could freely draw sketches within the air in the AirDraw mode. Through the conversion of finger tip locations to sketches within the virtual canvas, finger trajectory mapping was achieved. All keyboard commands concerning the functions of sketching, erasing, clearing the canvas, and exiting the AirDraw mode were also accomplished successfully.

The results in Fig 4. indicate the ease of interacting with the digital content using AirDraw without physical contact, but the slight jitter experienced during the rapid movements can be attributed to the shaking of hands and the low resolution of the camera. Smoothing filters can further enhance the drawing skills.

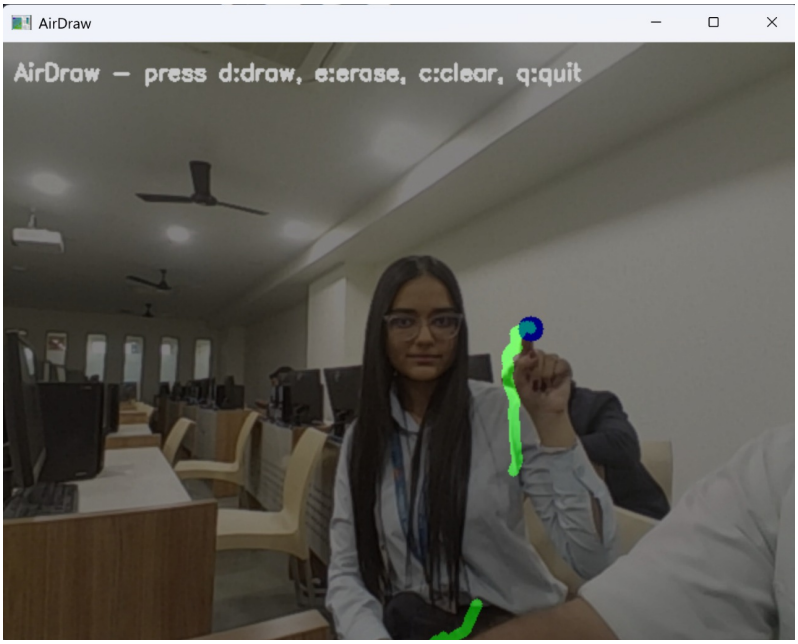

Fig. 4: AirDraw mode demonstrating real-time finger trajectory mapping on a virtual canvas using index finger tracking.

4.4 System Usability and Practical Implications

The system proves that the use of gesture input can be an optimal alternative to the conventional input devices. Additionally, the lack of the need for further hardware implies that the system is budget-friendly and simple to implement. The system may find applications in assistive technologies, virtual reality interfaces, smart classrooms, and public systems requiring hygiene.

Although the system has shown encouraging outcomes, it is vulnerable to lighting conditions and camera orientations. The system was found to slow down in low light conditions and in cases of finger obstruction. The system has also been found to cause user fatigue, considering the continuous gesture performed in mid-air.

4.5 Discussion

On the whole, the experimental results confirm the effectiveness of the proposed AirBoard system for realizing real-time gesture typing and drawing. The coupling of computer vision technology and human-computer interaction technology makes it possible for the system to input data in an intuitive way. In the future, improving the accuracy of gesture classification, adaptive calibration, and the extension of the system to multi-language keyboards and complex gesture commands can be considered.

5 Applications

The proposed AirBoard: Gesture-Controlled Virtual Keyboard Interface has several key applications across multiple fields. The system will give creative solutions for increased accessibility, hygiene, and productivity by touchless interaction using AI-driven hand gesture recognition in personal and professional scenarios. The following are the major areas of applications:

5.1 Contactless and Hygienic Interfaces

The use of contactless technology has now assumed much importance in the post-COVID era. AirBoard is a touch-free input device that can be applied in public settings such as airports, kiosks, ATMs, and medical centers, which pose a hygienic concern due to shared use of input devices. With AirBoard, individuals can now use the keyboard by merely typing or controlling the device with the aid of hand gestures, thus reducing the risk of contamination and ensuring a hygienic workplace.

5.2 Assistive Technology for Physically Disabled Individuals

This can be an important system in the field of assistive technology, as it offers a different form of communication for people with motor disabilities or restricted mobility. Users who cannot work with the conventional keyboard or mouse for interacting with computers can send information with simple movements of their hands, which are recorded by a webcam. This creates diversity in the digital context by making it more accessible and independent.

5.3 AR and VR Environments

Traditional input devices, such as keyboards and mouse, are usually inappropriate for most AR/VR applications. AirBoard can be used as a virtual keyboard or a control interface, while users can use gestures to input text or instructions in a realistic manner. Integrating such technology into interactive simulations or augmented/virtual reality headsets would lead to a better user experience, enhancing natural and intuitive human–computer interaction.

5.4 Smart Classrooms and Interactive Learning

AirBoard has the ability to provide teachers and students with a hands-free, interactive interface in smart learning environments. Teachers can control presentations, take notes, or interact with digital information shown in a projector or smart board classroom using gesture-based typing or commands without any physical input device, thus making the classroom more hygienic and entertaining.

5.5 Industrial and Workplace Automation

In industrial or manufacturing environments, workers often function in conditions that preclude practical use of physical keyboards or touchscreens because of gloves,

grime, or machinery. AirBoard can facilitate gesture-based control systems where command entry or production monitoring is possible without the need for direct contact; this increases both convenience and safety levels in risky or limited-access working environments.

5.6 Healthcare and Medical Applications

AirBoard can be utilized in sterile environments, such as operating theatres or diagnostic labs, by medical staff to communicate with a digital device. Using this, for example, surgeons might record data through gestures or read out medical imagery information without having to remove their gloves or violate the no-touch rule. In this respect, the tool serves very usefully to improve efficiency while maintaining hygiene during procedures.

5.7 Smart Homes and IoT Devices

AirBoard can also be extended to work as a gesture-controlled input interface for smart home devices in the IoT era. Users can navigate smart displays, control appliances, and input commands through air motions. This allows hands-free interaction by the user to perform everyday tasks with increased convenience of home automation.

5.8 Gaming and Entertainment

Gesture-based interfaces are finding an increased application in gaming. By extending the virtual keyboard capability of AirBoard to gesture-controlled games or entertainment systems, users can have a more captivating and immersive experience. In gaming situations, one would be able to input messages or interact with on-screen objects without the use of conventional technology.

6 Limitations and Future Work

The AirBoard: Gesture-Controlled Virtual Keyboard Interface has enabled contactless text input, but there are still some practical and technical restrictions. The main limitations found during implementation and assessment are covered in this section, along with possible future research directions to improve system scalability, accuracy, and usability.

6.1 Limitations

6.1.1 Lighting Sensitivity

Consistent lighting is necessary for the system to function properly. Because of shadow or glare interference, hand detection accuracy may suffer in low-light or extremely bright conditions. The system's accuracy is highly dependent on lighting conditions, which may cause inconsistent gesture detection in dim or overly bright environments [19].

6.1.2 Camera Position and Quality

Changes in webcam placement, frame rate, and resolution can have an impact on the accuracy of detection. Only when the camera is oriented squarely in front of the user's hand does optimal performance occur.

6.1.3 Typing Speed and Responsiveness

Currently, gesture recognition technology can only type at a slower rate than a traditional keyboard. Studies show that users do not reliably intersect the keyboard plane during mid-air typing, indicating that simple geometric tap detection is insufficient [7].

6.1.4 Limited Gesture Support

Only tap-based typing motions and a limited number of control keys (Space, Enter, Backspace, and Clear) are supported under the current model. There is currently no support for sophisticated gestures like swipes or multi-finger commands.

6.1.5 Environmental Restrictions

Hand-tracking models may become confused by cluttered or dynamic backgrounds; the system performs best against basic backgrounds.

6.2 Future Work

6.2.1 Integration of Depth Sensing

Incorporating depth cameras or artificial intelligence (AI)-based depth estimation can greatly increase the accuracy of tap detection and decrease false key activations under different lighting conditions.

6.2.2 Machine Learning-Driven Gesture Recognition

ML models (CNN/LSTM) can reduce jitter-related misclassifications and enable more reliable recognition of complex gestures by substituting threshold-based logic.

6.2.3 Personalized and Adaptive Keyboard Layouts

Upcoming models will be able to dynamically adjust the size or placement of keys in response to the hand movements of the user, improving typing comfort and lowering fatigue during mid-air.

6.2.4 Multi-Hand and Extended Controls

The virtual keyboard would become more functional and more akin to real-world typing experiences if it were extended to recognize two-handed input, multi-finger gestures, and modifier keys (Shift, Ctrl, Alt).

7 Conclusion

The AirBoard: Gesture-Controlled Virtual Keyboard Interface is a perfect example of how AI and computer vision can allow human-computer interaction without any physical contact. This system detects finger gestures very accurately and converts them into text input by using MediaPipe for real-time hand tracking and OpenCV for visual rendering. Being a very useful application in contactless environments, assistive technologies, and AR/VR systems, it offers a great accuracy level and responsiveness. The addition of a predictive text functionality that is presently being developed makes the AirBoard one gesture away from intelligent, touchless interfaces. The typing productivity and user experience are being enhanced to an even higher level by the device.

References

[1] D. A. Reddy, V. E. Jyothi, J. K. Viswanath, N. S. Chowdary, D. Swapna, and S. Sindhura, "A Pattern Recognition Model: Hand Gestures Recognition using Convolutional Neural Networks," in *Proc. 2023 Int. Conf. on Sustainable Computing and Smart Systems (ICSCSS)*, Coimbatore, India, 2023, pp. 460–465, doi:10.1109/ICSCSS57650.2023.10169433.

[2] N. F. Yıldıran, Ü. Meteriz-Yildiran, and D. Mohaisen, "AiRType: An Air-tapping Keyboard for Augmented Reality Environments," in *Proc. 2022 IEEE Conf. on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, Christchurch, New Zealand, 2022, pp. 676–677, doi:10.1109/VRW55335.2022.00189.

[3] Y.-M. Li, T.-H. Lee, J.-S. Kim, and H.-J. Lee, "CNN-Based Real-time Hand and Fingertip Recognition for the Design of a Virtual Keyboard," in *Proc. 2021 ITC-CSCC*, Jeju, Korea, 2021, pp. 1–3, doi:10.1109/ITC-CSCC52171.2021.9501471.

[4] D. Kim *et al.*, "Enhancing AR/VR Experiences with a Pinch-Gesture Multi-Layer Virtual Keyboard System," in *Proc. ICEIC 2025*, Osaka, Japan, 2025, pp. 1–3, doi:10.1109/ICEIC64972.2025.10879707.

[5] J. Kotti, B. Padmaja, and D. Deepa, "Enhancing Gesture-Controlled Virtual Mouse and Virtual Keyboard Using AI Techniques," *Journal of Mobile Multimedia*, vol. 20, no. 2, pp. 473–493, 2024, doi:10.13052/jmm1550-4646.2029.

[6] S. Lee and W. Choi, "Eyes on your Typing: Snooping Finger Motions on Virtual Keyboards," in *Proc. 2025 IEEE Symposium on Security and Privacy (SP)*, San Francisco, USA, 2025, pp. 4340–4355.

[7] X. Yi, C. Liang, H. Chen, J. Song, C. Yu, and Y. Shil, "From 2D to 3D: Facilitating Single-Finger Mid-Air Typing on Virtual Keyboards with Probabilistic Touch Modeling," in *Proc. 2022 VRW*, Christchurch, New Zealand, 2022, pp. 694–695, doi:10.1109/VRW55335.2022.00198.

[8] J. Shen, K. Khaldi, E. Zhou, H. B. Surale, and A. Karlson, "Gesture2Text: A Generalizable Decoder for Word-Gesture Keyboards in XR," *IEEE Trans. Visualization and Computer Graphics*, vol. 30, no. 11, pp. 7118–7128, 2024, doi:10.1109/TVCG.2024.3456198.

[9] A. P. S. Yadav *et al.*, "Gesture-Based Virtual Keyboard: Enhancing Human-Computer Interaction Through Hand Tracking," in *Proc. CYBERCOM 2024*, Dehradun, India, 2024, pp. 756–760, doi:10.1109/CYBERCOM63683.2024.10803180.

[10] A. Enkhbat, T. K. Shih, T. Thaipisutikul, N. L. Hakim, and W. Aditya, "HandKey: An Efficient Hand Typing Recognition using CNN for Virtual Keyboard," in *Proc. InCIT 2020*, Chonburi, Thailand, 2020, pp. 315–319, doi:10.1109/InCIT50588.2020.9310783.

[11] M. Miwa, K. Honda, and M. Sato, "Image Defocus Analysis for Finger Detection on A Virtual Keyboard," in *Proc. ICPR 2021*, Milan, Italy, 2021, pp. 24–30, doi:10.1109/ICPR48806.2021.9412219.

[12] C. Topal, B. Benligiray, and C. Akinlar, "On the efficiency issues of virtual keyboard design," in *Proc. VECIMS 2012*, Tianjin, China, 2012, pp. 38–42, doi:10.1109/VECIMS.2012.6273205.

[13] J. K. Mathew *et al.*, "Seamless Interaction through Gesture Recognition: Integrating Virtual Canvas, Keyboard, Calculator And Mouse," in *Proc. TEECCON 2024*, Bangalore, India, 2024, pp. 123–128.

[14] J. Shen, J. Dudley, and P. O. Kristensson, "Simulating Realistic Human Motion Trajectories of Mid-Air Gesture Typing," in *Proc. ISMAR 2021*, Bari, Italy, 2021, pp. 393–402, doi:10.1109/ISMAR52148.2021.00056.

[15] F. Kern, F. Niebling, and M. E. Latoschik, "Text Input for Non-Stationary XR Workspaces," *IEEE TVCG*, vol. 29, no. 5, pp. 2658–2669, 2023, doi:10.1109/TVCG.2023.3247098.

[16] B. Kabulov, N. Tashpulatova, and M. Akbarova, "Virtual Keyboard and Fingers," in *Proc. ICISCT 2019*, Tashkent, Uzbekistan, 2019, pp. 1–3, doi:10.1109/ICISCT47635.2019.9011894.

[17] T.-H. Lee, S. Kim, T. Kim, J.-S. Kim, and H.-J. Lee, "Virtual Keyboards With Real-Time Deep Learning-Based Gesture Recognition," *IEEE Trans. Human-Machine Systems*, vol. 52, no. 4, pp. 725–735, 2022, doi:10.1109/THMS.2022.3165165.

[18] R. N. Phursule, G. Y. Kakade, A. Koul, and S. Bhasin, "Virtual Mouse and Gesture Based Keyboard," in *Proc. ICCUBEA 2023*, Pune, India, 2023, pp. 1–4, doi:10.1109/ICCUBEA58933.2023.10392123.

[19] D. Bisht, A. Pal, and S. Banerjea, "VKM: A Virtual Keyboard and Mouse Solution Towards a Lightweight Computing System," in *Proc. DCOSS-IoT 2024*, Abu Dhabi, UAE, 2024, pp. 254–258, doi:10.1109/DCOSS-IoT61029.2024.00046.