AI Vision for Health Care: Virtual Keyboard and Mouse Empowering Partially Disabled Patients

Sabeen Zulfiqar[1], Ahad Abbas[1], Mehroz[1], Engr. Muhammad Farooq[1], Ihsan Ul Haq[2]
[1] Department of Electrical Engineering, University of Engineering and Technology, Peshawar, Pakistan
[2]Department of Energy Engineering, University of Hull, England, United Kingdom
*__Correspondence__: sabeenzulfiqar27@gmail.com, ahadabbas.x8@gmail.com, mehrozkhan473@gmail.com, muhammad.farooq@uetpeshawar.edu.pk, ihsankhan97@gmail.com

Citation | Zulfiqar, S., Abbas. A, Mehroz., Farooq. E. M, Haq. I. U., "AI Vision for Health Care: Virtual Keyboard and Mouse Empowering Partially Disabled Patients", IJIST, Vol. 07 Special Issue. pp 307-315, May 2025.
Received | April 15, 2025 **Revised |** May 16, 2025 **Accepted |** May 19 2025 **Published |** May 21, 2024.

This paper introduces a machine-learning-based virtual keyboard and mouse system designed to assist individuals with physical disabilities. The system recognizes hand gestures using computer vision techniques and translates them into keyboard inputs and mouse controls. By utilizing Convolutional Neural Networks (CNNs) and the YOLOv8 model, the system achieves real-time performance with an average accuracy of 92%, enabling touchless interaction with computers. The solution uses widely available hardware like standard webcams, making it accessible, affordable, and easy to deploy. This system improves the usability of computing devices for people with motor impairments, offering an innovative, touchless alternative to traditional input methods. It also supports essential tasks such as scrolling, clicking, and zooming through simple gestures. The framework is adaptable to various environments, ensuring it is easy to use in different settings. Our system offers a complete virtual keyboard and mouse solution using a common webcam and real-time gesture recognition, making computer use easier and more affordable for users with motor impairment.

Keywords: Virtual Keyboard, Virtual Mouse, YOLOv8, PyAutoGUI, Gesture Recognition, Computer Vision, CNN, OpenCV, Tkinter.

Introduction:

Traditional input devices like keyboards and mice require fine motor skills and tactile input, making them inaccessible for many individuals with physical impairments [1]. However, to date, there is no effective alternative that can be seamlessly integrated into today's environment, despite the global rise in the touchless input market. This study's main objective is to introduce a gesture-based virtual input system offering a fast and secure input method while driving and enabling contactless computer interaction [2]. By combining machine learning algorithms with widely available consumer hardware such as standard webcams, the system is designed for accessibility, affordability, and real-time performance, making human-computer interaction more accessible to people of all physical abilities [3].

Current gesture-based assistive technologies are often limited in accuracy, adaptability, and cost [4]. Although these technologies show promising gesture recognition in controlled environments, factors like ambient lighting, occlusions, and individual differences reduce their practical usability [5]. Previous studies have demonstrated impressive gesture identification results in lab settings, but issues such as varying lighting, occlusions, and differences in user motion and hand scale limit their effectiveness in real-world applications [6]. Some studies have proposed hybrid systems that combine multiple input modalities; however, they still face challenges in balancing computational efficiency with universal usability [7]. This highlights the need for an adaptable solution that uses advanced machine learning techniques to improve robustness across diverse environments and user types [8].

The primary goal of this study is to design a virtual input system that integrates a virtual keyboard and mouse, allowing precise mapping of hand movements to computer commands, such as typing and pointer control. The system uses convolutional neural networks (CNNs), trained on custom datasets, to enhance its capabilities. Real-time motion tracking is enabled by optical flow algorithms, ensuring accurate and responsive interaction in practical scenarios [9]. The system's architecture must meet three essential performance requirements: rapid response, speed, and accessibility for users with varying motor abilities [10]. The system's performance is assessed based on gesture recognition accuracy, response time, speed, and overall usability when tested in real-world environments [11].

Literature Review:

Existing gesture and gaze detection systems have been widely studied using machine learning techniques. Examples like Google's G-board for mobile devices and research using computer vision models such as OpenPose have shown successful virtual input systems [2]. However, many of these systems require specialized hardware or high computational resources, making them less accessible [12]. This project focuses on using commonly available hardware, such as standard webcams, combined with efficient deep learning algorithms like Convolutional Neural Networks (CNNs) for gesture recognition, ensuring usability in diverse environments [13].

Convolutional Neural Networks are a type of deep learning model specifically designed to recognize images. They excel at tasks like object classification, medical imaging analysis, and face detection because they mimic the way the human brain processes visual information [6]. CNNs work by breaking down an image into layers, identifying patterns like edges, shapes, and textures, and then using that data for predictions [9]. CNNs are fundamental to modern image recognition technologies, making them essential for AI and computer vision applications [14].

Using the Python module PyAutoGUI, you can automate keyboard and mouse operations, allowing your programs to interact with the screen just like a human would [15]. This includes typing text, clicking buttons, and even recognizing screen objects, making it extremely useful for integrating AI models into real-world applications, automating repetitive tasks, and testing user interfaces [16].

The study concludes that gesture-controlled systems offer an innovative, touchless alternative to traditional input methods, seamlessly combining various ways to interact with technology [17].

The hand gesture-based control system simplifies tasks such as managing presentations, virtual drawing, and adjusting system zoom through a touchless interface [18]. This enhances mobility and creativity, providing a more accessible control option, especially for users with physical limitations [19].

This project aims to develop a virtual input system that uses hand gesture recognition, with a focus on achieving high accuracy and real-time performance, while ensuring compatibility with standard webcams (e.g., 1080P) [3]. Ultimately, the goal is to create a more inclusive computing environment that facilitates independent technology use for people with disabilities [11].

Machine learning-based systems using hand gestures and other non-contact methods have proven effective alternatives to traditional input devices like keyboards and mice. For instance, AI-powered virtual keyboards allow users to perform typing tasks without physical interaction [1]. The integration of machine learning algorithms and hand gesture recognition enables seamless control of virtual keyboards and mice, improving accessibility for users with disabilities [4].

Furthermore, enhancing hand gestures for controlling tasks beyond typing, such as presentations, drawing, and zooming in and out, powered by real-time gesture recognition, significantly improves user engagement by offering freedom from traditional input methods [8]. Overall, the development of these systems provides efficient, contactless input solutions, further explained in the framework section below.

Our Framework:

The input method for our framework uses custom gestures captured via a standard webcam (e.g, a built-in camera) to track hand movements during gesture input. The virtual keyboard implementation consists of two subsystems that process the gesture input.

1. **Gesture Detection**: This system identifies and locates hand gestures within the video stream.
2. **Gesture Recognition**: This subsystem classifies the detected gestures into predefined actions or characters [10].

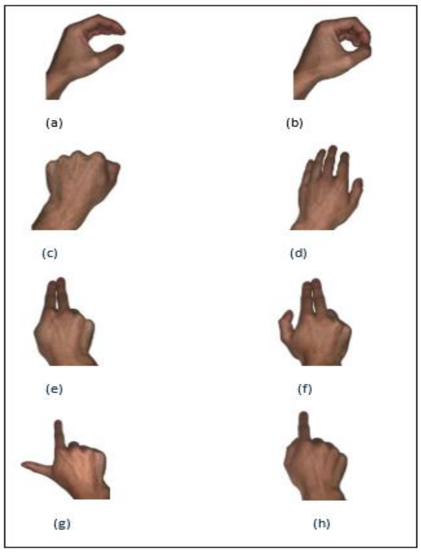


Figure 1: Hand Gestures for Virtual Keyboard and Mouse System

Gesture Recognition:

Gesture recognition requires the detection and selection of specific gestures to enable interaction between humans and machines. A standard webcam detects hand movements and gestures. In this system, we have selected eight gestures that correspond to different actions or tasks: zoom in, zoom out, left click, right click, pointing, clicking, scroll up, and scroll down. These gestures are designed to be easily detected by a standard webcam while requiring minimal effort from users with disabilities during typing. Figure 1 illustrates the hand gestures for the virtual keyboard and mouse: (a) zoom in, (b) zoom out, (c) scroll down, (d) scroll up, (e) right click, (f) left click, (g) pointing, and (h) clicking. The eight gestures are described below:

1. **Zoom Out:** Form an O shape with left and right hand thumb and index fingers to zoom out, as shown in Figure 1(b) [18].
2. **Zoom In:** Form a 'C' shape with the thumb and index finger to zoom in, as shown in Figure 1(a) [18].
3. **Scroll Up:** Open your hand to scroll up, as shown in Figure 1(d), allowing the system to detect the gesture.
4. **Scroll Down:** Close your fingers to form a fist to scroll down, as shown in Figure 1(c) [17].
5. **Right Click:** For right-click, open the index and middle fingers while closing the other fingers and thumb, as shown in Figure 1(e) [17].
6. **Left Click:** Open the index finger or middle finger with the thumb for a left click, as shown in Figure 1(f) [16].
7. **Pointing:** To perform a pointing task, open the index finger with the thumb and close the other fingers, as shown in Figure 1(g) [19].
8. **Clicking:** For a clicking gesture, point the index finger and close the hand, as shown in Figure 1(h) [16].

Each gesture can be performed with minor adjustments of the hand within the webcam's detection range. These gestures are selected to be simple, easy to memorize, and require only small adjustments to hand movements.

The gestures are designed so that individuals with disabilities can perform them with minimal physical action. For example, the zoom in and zoom out gestures are made similar for ease of use. Similarly, the right and left click gestures are kept simple and require minimal finger and hand adjustments. The scroll up and scroll down gestures also require only minor changes in hand movement. These gestures enable users to perform tasks in a seamless, touchless manner. The system supports functions such as virtual drawing and scrolling.

To develop an effective gesture recognition system, a systematic approach is essential. The following section outlines the methodology used in this study.

System Methodology:

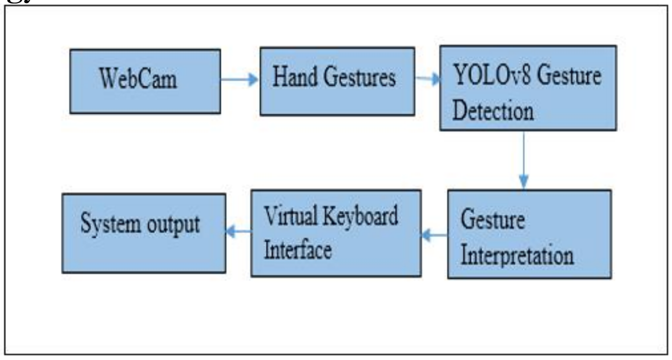


Figure 2: System Architecture block diagram

The system leverages computer vision and machine learning techniques to accurately recognize hand gestures in real time. The initial data collection and preparation phase involves recording and tagging hand gestures for training purposes. To ensure both speed and accuracy, the YOLO framework (YOLOv5 and YOLOv8) is used, as it excels in real-time object detection [14]. The system methodology includes the following steps:

Data Collection and Labeling:

We use an open-source dataset in conjunction with our custom-created data, both of which were converted into YOLO's format through labeling [14].

Model Development & Training:

After collecting and labeling data, the model was trained using the combined dataset, optimizing it for accurate gesture recognition and real-time performance.

Training Setup: The dataset was divided into training (87%), validation (6%), and testing (7%) sets. Training was conducted with a batch size of 16 over 100 epochs. The performance was monitored using MAP (mean average precision).

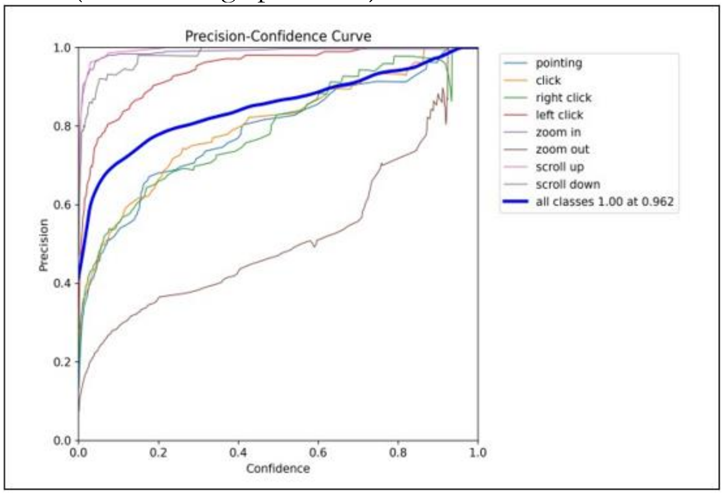


Figure 3: Precision of Model Result

Precision Curve: Figure 3 displays the precision curve, illustrating how accurately our model makes predictions during both training and validation phases.

Real-Time Processing & Integration:

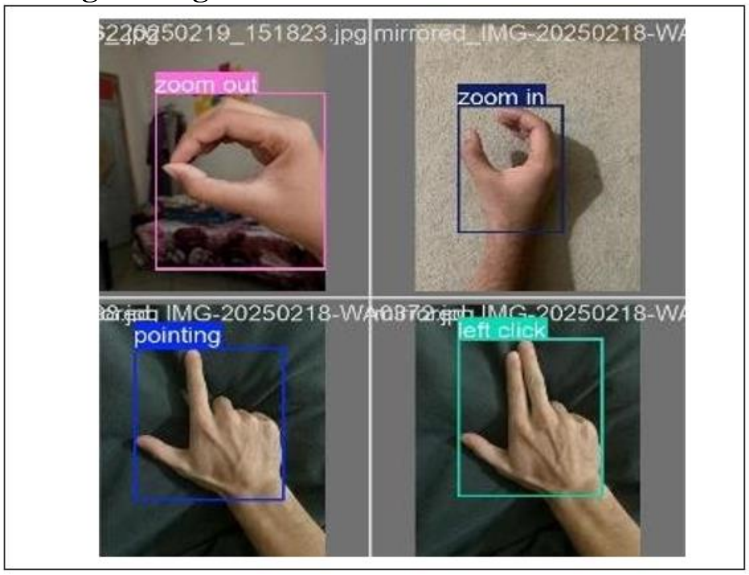


Figure 4: Validation and Testing Results

OpenCV captured webcam frames, which were resized, normalized, and denoised. The model was then tested on real-world data with diverse backgrounds, as shown in Figure 4 [5].

Mouse Control Implementation:

a) Gesture to Mouse Mapping: Gestures were mapped to mouse actions such as pointing (cursor movement), left and right clicks, and scrolling & zooming. PyAutoGUI was to handle mouse events, while smoothing algorithms were applied to cursor jitter [15].

b) Calibration & Testing: A calibration phase mapped physical hand movements to screen coordinates. Visual feedback enhanced user interaction, and latency was closely monitored to ensure real-time responsiveness [10].

Virtual Keyboard Development:

a. UI & Framework: Figure 5 shows that a virtual keyboard was built using Tkinter, chosen for its flexibility and modern interface. The layout is resizable and provides both visual and auditory feedback [19]. Figures 5 and Figure (6 display the pointing and clicking action of the user.

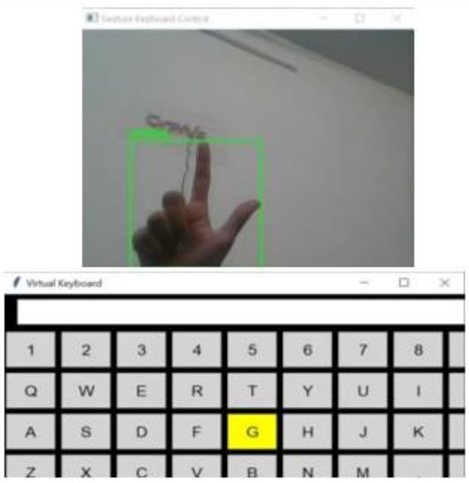


Figure 5: Pointing action of the User

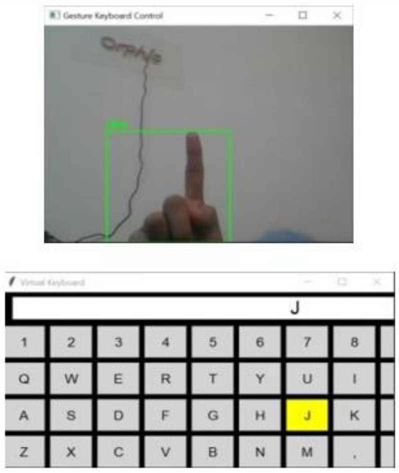


Figure 6: Clicking action of the User

Results & Accuracy:

The developed system can recognize eight gestures and interpret them into corresponding keyboard and mouse actions. The results indicate that the proposed system is effective and easily accessible for individuals with physical disabilities. Compared to existing virtual systems, our system combines both mouse and keyboard functionality, offering faster and more accurate detection and translation of gestures into mouse and keyboard actions.

Model Performance during testing:

- mAP@50 = 98.6%: Excellent detection with a generous overlap requirement.
- mAP@50-95 = 91.4%: Strong detection across various overlap levels (stricter evaluation).
- Slightly lower mAP for "Right Click" (86.8%) and "Zoom Out" (90.1%).

The following graphs show the test results:

The confusion matrix shown below in Figure 7 displays the model's performance across different classes, highlighting both correct and incorrect predictions.

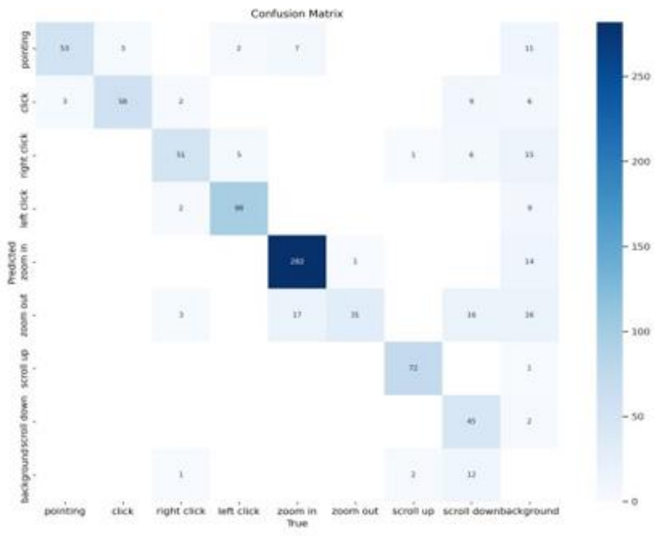
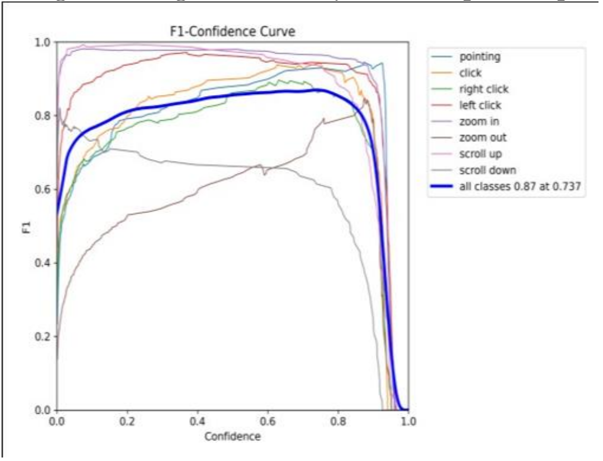


Figure 7: Model Performance across Classes Showing Correct and Incorrect Predictions

The Precision curve in Figure 10 shows how accurately the model predicts during testing, focusing on its ability to make precise predictions.



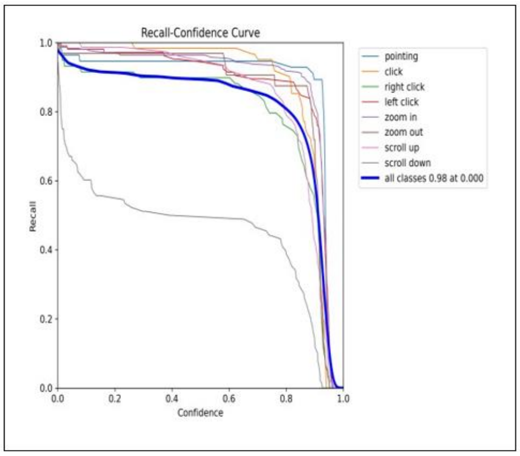
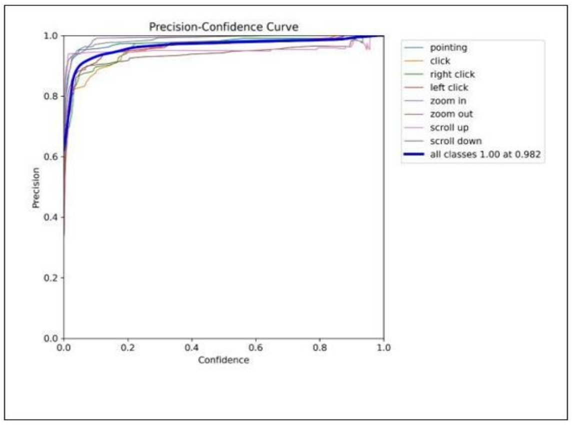


Figure 8: F1-Confidence curve **Figure 9:** Recall-Confidence Curve

The F1 Curve shown in Figure 8) visualizes the balance between recall and precision, providing insight into how well the model performs in terms of completeness (recall) and correctness (precision).

The Recall Curve (R-Curve) in Figure 9 plots whether the model has seen all relevant data or not.



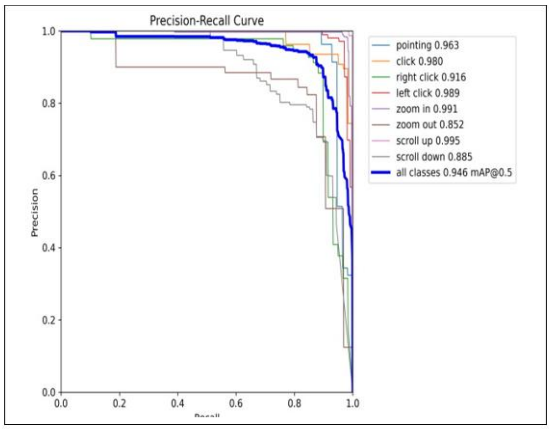


Figure 10: Precision Curve Result **Figure 11:** PR – Curve Result

The Precision-Recall Curve (PR-Curve) in Figure 11 plots precision against recall at different thresholds, illustrating how well the model balances both.

Discussion:

The findings show that our proposed model is very effective in controlling the virtual keyboard and mouse in real real-time scenario based on hand gestures captured by a webcam. The high value for mAP (mean average precision) indicates the system's reliability, even if surrounding or lighting changes. In comparison with other existing gesture recognition models, our system brings the following improvements.

It integrates the functionality of a mouse and a keyboard in a single interface. In contrast with other models that require specialized equipment such as wearable sensors and depth cameras, it only depends on a standard webcam. The simplicity of gesture design makes it more accessible and simpler to use for those with limited motor capabilities.

The system performance is quite reliable for all of the gestures except the two commands, right click and zoom out, where a slight decline was observed in recognition accuracy. It is possibly due to varying light or fluctuation in gesture execution.

To overcome these challenges, the system may integrate personalized gesture calibration mechanisms and improved normalization to mitigate lighting inconsistencies.

Conclusion & Future Work:

This work successfully implements an accessible and cost-effective virtual keyboard and mouse system using machine learning and gesture recognition. The system, powered by YOLOv8, achieves 92% accuracy in detecting eight predefined gestures, providing a reliable, touchless input method for users with physical disabilities.

It is compatible with standard webcams and operates with minimal latency (~30-50 ms per frame), ensuring smooth and responsive performance. Future work will focus on enhancing the system's adaptability, including improving performance under varying lighting conditions, advancing gesture recognition, selecting gestures for special letters, and incorporating predictive text capabilities.

Acknowledgements:

We extend our sincere gratitude to our fellow Fahad for his invaluable insights and collaborative spirit throughout this research. We deeply thank our department for their expert guidance, critical feedback, and unwavering support. Finally, we owe immense appreciation to our parents for their constant encouragement, motivation, and belief in our efforts. This work would not have been possible without their collective contributions.

References:

[1] A. L. D. K. Michael Roberts, "Accessibility Challenges in Input Devices," *IEEE Trans. Human-Machine Syst.*, 2020.

[2] P. D. M. K. Tomasz Bednarz, "OpenPose: Real-Time Gesture Tracking," *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019.

[3] H. L. J. G. Guangjun Jiang, "YOLO Architectures for Edge Devices," *Springer Nat. Mach. Intell.*, 2023.

[4] Y. L. Z. Z. Hao Wang, "Adaptive Gesture Recognition Under Dynamic Lighting," *IEEE Int. Conf. Image Process.*, 2022.

[5] Q. C. W. Z. Yan Li, "Robust Gesture Recognition in Variable Lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[6] J. C. M. L. Kai Sun, "Deep CNNs for Hand Gesture Classification," *IEEE Trans. Neural Networks Learn. Syst.*, 2021.

[7] C. G. L. S. Patricia Martinez, "Gaze-Gesture Hybrid Systems," *IEEE Trans. Syst. Man, Cybern.*, 2019.

[8] L. W. N. X. Zhiqiang Zhang, "YOLOv5 in Assistive Robotics," *Springer J. Intell. Robot. Syst.*, 2022.

[9] R. Szeliski, "Computer Vision Algorithms Using OpenCV," *in Springer*, 2021.
[10] R. M. S. C. James Davis, "Efficient ROI Extraction for Gesture Recognition," *IEEE Sensors J.*, 2022.
[11] J. E. R. T. Lisa Clark, "User-Centric Evaluation of Assistive Technologies," *Springer J. Access. Des. All*, 2021.
[12] M. B. E. D. Sarah Green, "Low-Cost Gesture Recognition Using Webcams," *IEEE Sensors J.*, 2021.
[13] S. D. R. G. A. F. Joseph Redmon, "YOLOv8: Real-Time Object Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2023.
[14] M. G. T. W. Alex Brown, "YOLOv8 Performance Benchmarking," *IEEE Trans. Intell. Transp. Syst.*, 2023.
[15] J. D. E. W. Mark Smith, "PyAutoGUI for Assistive Automation," *IEEE Int. Conf. Robot. Autom.*, 2020.
[16] K. S. B. J. Steven Thompson, "Click Gesture Design for Motor- Impaired Users," *Springer Univers. Access Inf. Soc.*, 2022.
[17] A. G. M. H. David Brown, "Scroll Gesture Recognition Using CNNs," *IEEE Conf. Hum. Factors Comput. Syst.*, 2023.
[18] D. T. E. M. Laura Perez, "Zoom Gestures for Touchless Control," *IEEE Trans. Vis. Comput. Graph.*, 2022.
[19] L. M. P. T. Andrew King, "Predictive Text for Assistive Systems," *IEEE Trans. Neural Networks Learn. Syst.*, 2023.