International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(22s) 2026 pp. 644-650
https://doi.org/10.70917/ijcisim-2026-5467
Article

Vision-Based Hand Gesture Keyboard for Game Control Using OpenCV

Pradnya Jadhav[1], Poorvi K. Joshi2*, Aarti R. Salunke[3], Rishi Pande[4], Harish Khandelwal[5], Aditya Arvikar[6], Aayush Pande[7]

2*,3,4,6,7Ramdeobaba University, Nagpur (MS), India.
[1]Yeshwantrao Chavan College of Engineering, Nagpur (MS), India
[5]MGMU Jawaharlal Nehru Engineering College of, Sambhaji Nagar, India

Abstract: A vision-based system is designed to enable real-time hand gesture control in gaming, without any need for physical accessories. The system uses a standard webcam to record and track hand gestures that mimic steering-wheel movements. A virtual steering indicator is also displayed on the screen, which rotates based on hand tilt to reflect direction and intensity of steering. These gestures are mapped to keyboard keys such as W (acceleration), A (left), D (right), and in certain configurations, S (reverse). The system is developed entirely in Python, using OpenCV for real-time image processing and MediaPipe for precise 21-point hand landmark detection. By programmatically simulating keyboard events, PyAutoGUI enables interaction with almost any PC game that supports conventional keyboard input. The system is able to recognize gestures with low latency and does not require any external equipment or integration with the game. To establish an initial steering reference, users align their hands in a straight position during a short calibration step, simulating a virtual steering wheel. After calibration, the system calculates tilt angles in real time and converts them into keyboard inputs. This touchless interface provides a new way of interaction, especially for users interested in immersive technologies, and also offers an alternative to traditional controllers. The system was tested on the kart racing game (Smash Karts) and demonstrated smooth performance. It also shows potential as a Human-Computer Interaction (HCI) technique in gaming, with scope for further improvements and application in more complex game environments.
Keywords: Recognition of hand gesture, OpenCV, Medi-aPipe, PyAutoGUI, Human Computer Interaction, Game control

1. Introduction

Especially in application areas such as gaming where im-mersive and touchless interaction is increasingly in demand, the possibility to interact with digital systems through hand recognition gestures is increasingly becoming appealing Nat-ural human interfaces based on computer vision are trans-forming the traditional input devices such as video game controllers, conventional keyboards, and mouse. Using Python and several powerful open-source libraries, like MediaPipe OpenCV, and PyAutoGUI, introduces a vision-based in real-time turning hands gestures system made specifically to control the game. Through this system, one can achieve game inputs, such as movement, shooting or jumping, only using hand motions to control the game, which makes the system very natural to use and immersive in nature. It also creates new avenues in being more accessible and futuristic gaming sanctuaries since bodily contact is removed. This system works on MediaPipe and monitors 21 key landmarks of the hands for every frame captured through the live webcam video. The subtle changes in this anglular position of the hands can be detected with the help of the stored data. It then requires a period of system calibration where the user places his hands in the center of the system like what we do while holding a steering wheel. The subsequent gestures all have this calibrated position as their neutral position. The system compares the initial position of the hands of the user with the inital reference frame and constantly observes the live video till the calibration is completed. This helps to effectively recognize directional movements, e.g., right turn or left turn, and other particular gestures, e.g. open fist and closed fist. These movements

are then converted into the input of a keyboard for real-time control of the game without having to touch it literally. Directional control gesture inputs are computed by the sys-tem based on change in angularity as the user shifts his hands to the left or right. MediaPipe gives us the exact landmark values to compute the correct angle and the OpenCV handles the computations of these directions in real time to extract and analyze the necessary visual data. Further it is simulated with the help of application PyAutoGui which simulates the keyboard keys. As an example, in a racing game, the right hand will accelerate when the hands are tilted to the left, and vice versa, so the A key will be pressed, the D key will be pressed, and the W key will be pressed on the same hand and steady forward facing hand position respectively. Releasing the following key is done to get the reverse angle,or as the user returns his hands to a neutral position. The system can also identify gestures such as a closed fist or a single-hand presence to initiate other in-game behaviors which increase the flexibility of this control. The system uses three tools to play video games using natural hand gesture instead of traditional physical controller. MediaPipe detects the exact position of hands, OpenCV processes the video input from webcam and PyAutoGUI translates the motion into actual game commands. This creates a seamless and responsive gameplay experience. The system was successfully tested on the browser based on game Smash Karts, to check its effectiveness. The game Smash Karts, allows the player to steer and move by the movements on hands in air just like out system.

Players could play by driving, accelerating and steering with their hands in front of a web

camera. According to the users, there was an easy and interactive experience with minimal latencies and high precision even when the users worked in normal lighting. This becomes an affirmation of the system implementation in a broader market beyond Smash Karts and maybe extend to other fast-paced games or even in virtual reality. In addition to better immersion, a more universal gaming experience to everyone, both users with limited mobility or a desire to interact in a more natural way, which is also achieved through its zero direct contact implementation. In comparison to traditional control systems relying on gesture recognition that have specialized hardware, the offered solution is airy, and it can be used with the laptops or desktops at hand having a webcam. Gen Z gamers who demand new forms of gaming and contactless experiences will be particularly interested because of its convenience in using, minimalism in the hardware, and the convenience of converting untrained gestures. This system offers another and more convincing command of the game without involving keyboards so long as the steering and movement commands can be carried out without involving the hands as long as it is possible to use the hands in their form of hand gestures. It demonstrates the feasibility of computer vision to interactive human-computer systems, as well. Furthermore, the system is modular and can be tailored and expanded, such as by introducing new gestures, playing with other games or even using voice commands in combination. The current project is a first step into the future of governing AI-powered gaming experiences that convert traditional methods as immersive technology continues to expand.

2. Literature Survey

Researchers have explored the vision-based systems for hu-man computer interaction instead of physical tools for virtual tasks and gaming [1], [2], [3], [4]. The study by Sriboonruang et al. focused on a board game interface where hand gestures replaced mouse and keyboard. This proved that gestures can be a natural way of giving commands to the computer and are reliable [5].Tools like MediaPipe and OpenCV are used to make mouse and keyboard systems easier. Players can interact with screen without using the system hardware by mapping specific hand movements to the commands [6], [7], [8], [9]. Patel et al. system used MediaPipe to instantly convert gestures into game inputs by tracing hands in real-time to provide a smooth experience to player [10].Researchers use Deep Learning Advanced AI models like convolution neural networks to improve accuracy. The com-puter can “see” the gestures even under tricky conditions [11], [12]. Other study by Kumar et al. have used these models to improve the poor lighting or solve the issues of “false” movements [13]. The old and new math models. Like decision forest continues to ensure the gesture detection is sharper and with more precision [14], [15].A gesture control game proposed by Adilzhanov and Omarov was built using Augmented Reality (AR). This system lets used to interact with virtual objects by tracking the features in real world environments. This highlights the significance of vision-based gaming in making the digital experience more accessible and connected to the physical world [16], [17], [18], [19], [20].

3. Methodology

The primary use case of this system is to detect and inter-pret real-time hand gestures using a ordinary web camera. The procedure starts with capturing video in real time with the help of OpenCV. The frames obtained from this video are fed as input to the hand tracking model of MediaPipe, which detects 21 hand landmarks including fingertips, joints, and wrist. These landmarks are considered as reference points for tracking hand movement, and this helps in more precise gesture recognition. The next step is the calibration stage, which is very crucial for gameplay. The player is asked to place both hands in a natural steering position in front of the webcam. A countdown of five seconds begins when both hands are properly detected.

During this process, the system calculates the average angle between the joints of the index finger of both hands. This angle is the neutral position for steering and considered as baseline for all further movements comparisons. This allows the system to adapt to different users and hand positions for better accuracy. Once the system detects the hands positions, it interprets the gestures and converts them into keyboard inputs using PyAutoGUI. A hand tilt gesture is used to control the steering, where left corresponds to A key and right corresponds to D key.The system adjusts the duration of key inputs based on the amount of tilt. Shorter key taps generate smaller tilts whereas longer key taps result in larger tilts, allowing smoother and more controlled steering. In addition to steering, thumb-based gestures are used for motion control. Other movements are triggered using separate keys such as W is mapped to specific thumb gesture, while space key is mapped braking. In certain configurations, reverse movement can also be triggered using gesture-based input mapped to the S key. If the system detects that your hands are not in the correct hand position, missing, or closed fists, it restricts input generation to prevent unintended actions. The primary advantage of this system is its speed, as it processes the movement instantly to make sure there is no lag during the game play. The system uses a standard webcam over expensive gears, to make the setup lightweight and easy to use like any other keyboard-based games. The player gets a natural and smooth experience giving much more immersive experience than using the physical controllers as the software handles all the complex tracking and button simulation in the background. PyAutoGUI , OpenCV and MediaPipe works together to make the system efficient and easy to use by automatically capturing and translating the movements into game actions.

A. System Initialization

The system lowers the resolution and disables the heavy model layers to keep the gameplay fluid; this helps in sig-nificantly cutting down processing time for each frame. As a result, gesture recognition takes place almost instantly, which is important because even a tiny bit of lag can ruin the fast-paced experience of the game. By tracking only 2 hands, the system tries to avoid any unnecessary processing load, and using a confidence threshold ensures that accidental movements in background are not mistaken as commands. These optimizations keep the track stable and accurate without impacting the speed. This gives a player a reliable, natural-feeling control system where speed and time plays a vital role.

B. Calibration Phase

The system begins with an essential step, calibration phase, which acts as the foundation for gesture recognition. This step allows the software to understand the starting position or the “neutral” position of hands before the game begins. MediaPipe identifies points on both the hands of the players like wrist and index finger joints using a standard web camera. To start this phase the player simply holds hands in air as if gripping an invisible steering wheel. Once the system detects the correct position, it triggers a five-second countdown. This five-second time is given to avoid incorrect baseline reading, as it is important to keep the hands perfectly steady. This brief pause gives time to the player to get into a comfortable and natural posture with a stale reference point without any shaking of sudden movements.

During the countdown, the system measures the angle between the index finger joints and considers it as baseline for reference. As the players start the game, the system continuously compares the current hand position with the baseline to determine if the players want to move left or right. This technique ensures accessibility as each player

has a unique way of holding their hands. Factors like posture, length of your arm or even the height of the web camera can change the way computer looks at your gesture. The system sets up a new baseline every time a player plays the game to adapt to the player individually. This reduces the wrong input to ensure that accidental twitches or small shifts in chairs won't be mistaken as deliberate movements. This ultimately provides a more personalized and controlled gaming experience.

C. Real-Time Gesture Tracking

Once everything is set, the system starts watching the video and understands each frame as it comes. MediaPipe finds 21 important points on your hand and keeps tracking them as your hand moves. The system looks at the base joints of both index fingers and checks if they are in the same position as before (original angle saved during setup). The game decides to turn left or right based on the angle it calculates. It shows how much the player moved (tilted) their hands from the normal position. If the final angle calculated by the system is negative, the game turns left. If the angle is positive, the game turns right. Instead of using fixed key presses, the system presses the key longer or shorter based on how much you tilt your hand (like more tilt → longer key press, less tilt → shorter key press). Small tilts give short presses, and big tilts give longer presses, making control smoother and better.

- Left steering (mapped to 'A' key) → Negative deviation
- → Positive deviation

To move forward, the system depends on a gesture instead of just seeing the presence of both hands. When you use your thumb gesture, it presses the W key to move forward. Similarly, braking (stopping or slowing down) is done using another gesture that is connected to the the space key or presses the space key. You can also set a gesture to press the S key to move backward. If one or both hands are no longer detected or move significantly outside the expected area, the system stops all keys. This avoids wrong moves and keeps the game control steady and reliable. This avoids wrong moves and keeps the game control steady and reliable. The game follows your hand movements in real time, so the gameplay directly based on your hand movements. Each frame updates the game, so even small hand movements affect it, making the game feel interactive and real.

D. Key Simulation and Control Logic

The system reacts instantly to every move you make; this makes the game incredibly interactive. Even a small shift if hands are detected which makes the control smooth and natural. PyAutoGUI library is used to bridge the gap between players hands and the game as it translates the physical gesture into keyboard commands automatically. A custom control logic is used constantly to monitor your hands to check if the system should tap, hold or release a specific key. Instead of simply checking the truing of hands, the system measures the angle of hand tilt, to figure out how much you want to steer. This is where the system gets smart, just for little tilt of hands, the system performs quick, short key taps. If the player tilts hands significantly, it holds the key down longer for a sharp aggressive turn. This variable input mimics a real steering wheel, making the experience more realistic than simply tapping buttons on keyboard. The system relies on specific gesture-based inputs for basic movement. A thumb-based gesture is mapped to "W" key to accelerate forward, while a completely different "space" key is mapped for braking. There is a built-in safety feature to keep games from glitching out, if players' hands move out of the camera's view the system immediately releases all active key inputs. This prevents your vehicle from driving off on its own, ensuring stable and reliable gamming session.

E. Visualization and Feedback

The system shows live feedback on the webcam screen so it is easy and natural to use. The user can see their hand movements on the screen like a mirror. A steering icon is shown in the center of the screen, like a virtual steering

wheel. It turns left or right based on your hand tilt and shows the direction (how strong the turn is) and strength. This helps the user quickly understand and adjust their hand movements for better control.

In addition, a line is drawn between your hands to show the angle the system is using. Important hand points such as fingertips, joints, and wrist points are also displayed on the screen. These points help the user check if the system is detecting the hand correctly or if the hand is out of view. The system also shows FPS (frames per second) to show how well it works and how smooth it runs. This helps the user see if the system is running smoothly without lag. Since everything is shown live, the user gets less confused and feels more confident while playing. The system runs most keyboard games by pressing keys instantly, making the game smooth and fast. Overall , this system makes it easy to understand and helps users control the game naturally using their hands.

6. Results

Once the system is set up it successfully turns your hand movements into game controls and real time. After quick calibration, you simply tilt your hands to left and right. “W” key is used to

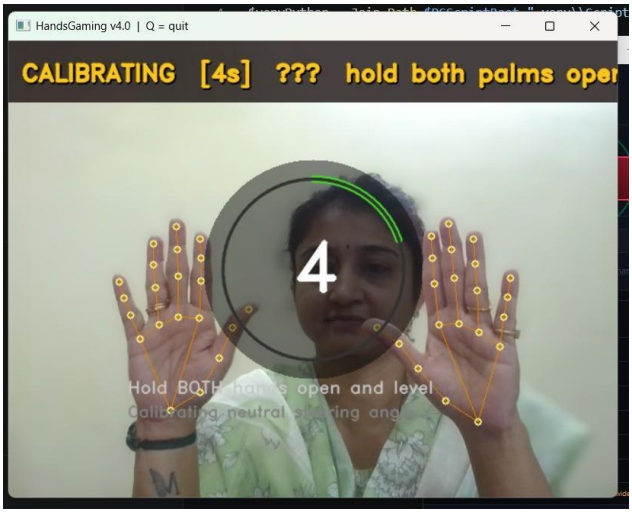

Fig. 1. Calibration phase: User holding both hands open to set neutral steering position

Turns are detected based on the relative tilt angle between both hands, allowing smooth and responsive steering. The system continuously monitors hand positions and com-putes the steering angle in real time. Based on the orientation, it generates corresponding key inputs such as A (left) and D (right).

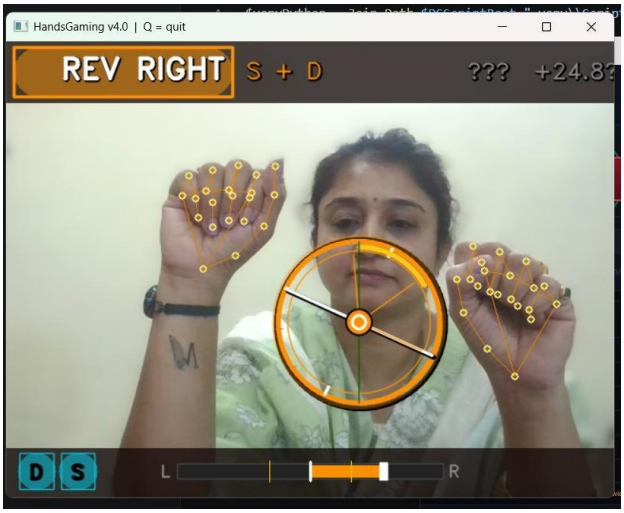

Fig. 2. System recognizing right turn gesture and generating ‘D’ key input

Instead of fixed key presses, the system adjusts the duration of key inputs depending on the amount of tilt, resulting in smoother and more controlled steering. In addition to steering, gesture-based control is used for motion inputs.

Forward movement is triggered using a thumb-based ges-ture, while braking is performed using a separate gesture mapped to the space key.

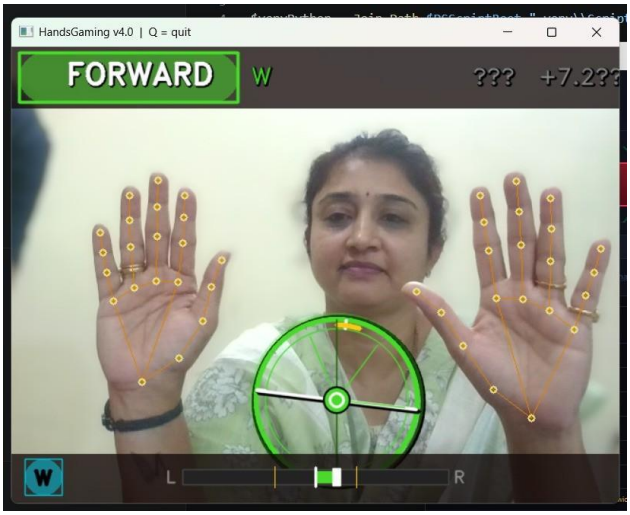

Fig. 3. System detecting forward gesture and issuing ‘W’ key for acceleration

In some configurations, reverse movement can also be implemented using gesture mapping.

Testing was done in varying light and background condi-tions. The average frame rate obtained by the system was between 25–30 FPS, thus ensuring low latency during the tracking. The application of proportional steering ensured that the control was smoother, hence the vehicle could make gradual turns. The performance of the system was reliable in the Smash Karts arcade racing game.

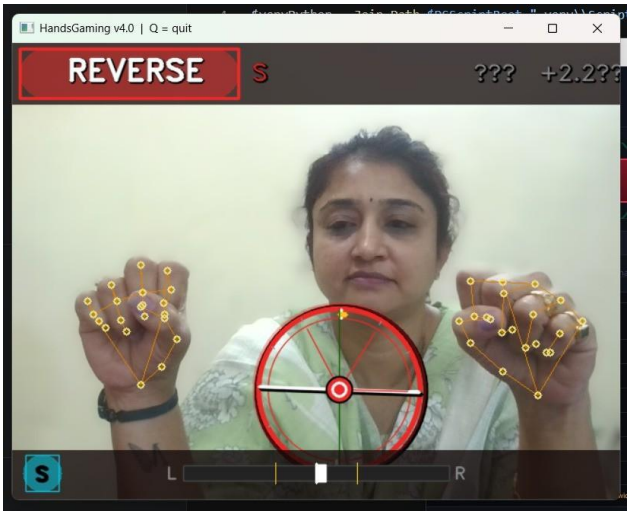

Fig. 4. System detecting reverse gestures and issuing ‘S’ key input

7. Conclusion

Racing games are discussed in this paper in relation to an approach to playing them using only hand gestures, replacing traditional controls on the keyboard. To implement the pro-posed method, Python modules are used, namely, MediaPipe for hand landmark recognition, OpenCV for image processing and calculation of angles, as well as PyAutoGUI to simulate keystrokes. Since the implementation does not need any extra hardware, the system is affordable and convenient for its users. To guarantee gesture detection accuracy, there is a step called ”calibration.” During calibration, the user has to hold his or her hands in a manner resembling the driver’s grip on the steering wheel, and the program calculates the reference angle. After calibration, hand positions are continuously tracked, and steering direction is calculated based on the tilt between them and mapped to the keystroke direction, for example, A or D. The angle value can further be used to determine how long the key press lasts. For moving forward, there is a specific gesture performed with the thumb, and it is mapped to W. As for braking, the gesture for it differs from the first one, and its mapping includes the spacebar. Reverse gear can also be activated through a different gesture mapping if it is needed. The system was tested on the race car game named Smash Karts. The test showed that, despite being executed on a low-power laptop, the program had enough performance, delivering smooth gameplay at about 25–30 FPS. Detection problems occurred under poor lighting conditions but gen-erally the system worked without difficulties. As for future work on this project, it should be considered implementing adaptive thresholds, lighting compensation, and a possibility to play without one’s other hand. Also, other gestures may be introduced for controlling the game.

REFERENCES

1. G. V. S. M. Kumar *et al.*, “Game Controlling using Hand Gestures,” in *Proc. IEEE Int. Conf. Advancements in Smart, Secure and Intelligent Computing (ASSIC)*, 2022, pp. 1–5.
2. A. Adilzhanov and N. Omarov, “Development of Augmented Reality Game Using Computer Vision Technology,” in *Proc. IEEE 4th Int. Conf. Smart Info Systems and Technologies (SmartInfoSysTech)*, 2024, pp. 386–390.
3. H. S. Harish, “Computer Vision Based Hand Gesture Recognition System,” *NeuroQuantology*, vol. 20, no. 7, pp. 2859–2866, Jul. 2022.
4. O. Ramirez-Valdez, C. Baluarte-Araya, R. Castillo-Lazo, *et al.*, “Control Interface for Multi-User Video Games with Hand or Head Gestures in Directional Key-Based Games,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 1, 2025.
5. R. Jany, M. N. Hasan, M. S. Islam, and S. M. Abir, “NFS: A Hand Gesture Recognition Based Game Using MediaPipe + OpenCV + PyGame,” *arXiv preprint arXiv:2204.12536*, Apr. 2022.
6. S. Dhamodaran, P. P. Phukan, M. Singh, and S. Nandakumar, “Imple-mentation of Hand Gesture Recognition Using OpenCV,” *Int. J. Res. Publ. Rev.*, vol. 5, no. 5, pp. 8039–8050, May 2024.
7. A. Sen, T. K. Mishra, and R. Dash, “Deep Learning Based Hand Gesture Recognition System and Design of a Human-Machine Interface,” *arXiv preprint arXiv:2207.04512*, Jul. 2022.
8. J. Qi,“Computer vision-based hand gesture recognition for human-robot interaction: monocular and RGB-D cameras,” Complex and Intelligent Systems, vol. 10, no. 1, pp. 1203–1219, Jan. 2024. doi: 10.1007/s40747-023-01173-6.
9. A. Mujahid, A. S. Malik, R. M. Daudpoto, and S. H. Khokhar, “Real-Time Hand Gesture Recognition Based on Deep Learning Using YOLOv3 and DarkNet-53,” Applied Sciences, vol. 11, no. 9, pp. 4164, May 2021. doi: 10.3390/app11094164.
10. K. Gao, “Challenges and solutions for vision-based hand gesture inter-pretation techniques in human–computer interaction systems,” Computer Vision and Image Understanding, vol. 238, pp. 103726, Apr. 2024. doi: 10.1016/j.cviu.2024.103726.
11. P. Hrishikesh, “Vision Based Gesture Recognition,” Procedia Computer Science, vol. 227, pp. 707–714, 2024. doi: 10.1016/j.procs.2024.07.087.
12. A. Sen, T. K. Mishra, and R. Dash, “Novel Human Machine Interface via Robust Hand Gesture Recognition using Channel Pruned YOLOv5s Model,” arXiv preprint arXiv:2407.02585, Jul. 2024.
13. S. Gupta, K. Kim, and J. Kautz, “Hand gesture recognition with 3D convolutional neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), 2015, pp. 1–7.
14. C. Wolf, G. Taylor, and F. Nebout, “Multi-scale deep learning for gesture detection and localization,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2014, pp. 474–490.
15. X. Yang, S. Gupta, K. Kim, and J. Kautz, “Online detection and classification of dynamic hand gestures with recurrent 3D CNNs,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 4207–4215.
16. H. Liang, J. Yuan, and D. Thalmann, “3D convolutional neural networks for efficient and robust hand pose recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 4, pp. 956–970, Apr. 2019.
17. F. Kırac¸, Y. E. Kara, and L. Akarun, “Hand pose estimation and hand shape classification using multi-layered randomized decision forests,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2012, pp. 852–863.
18. W. Lee, and S. Lee, “Real-time hand gesture recognition using finger segmentation,” IEEE Access, vol. 5, pp. 1–10, 2017.

19. T. Simon, S. Wei, and Y. Sheikh, “Realtime multi-person 2D pose estimation using part affinity fields,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 7291–7299.
20. M. M. Trivedi, “Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach,” IEEE Trans. Intell. Transp. Syst., vol. 15, no. 6, pp. 2368–2377, Dec. 2014.