Hand-Gesture Recognition System for Sustainable Technology in Environmental Monitoring

Pooja Nayak[1], Kanchana M Dixit[2], Keerti Patil[3], Sowbhagya M P[4], Nandini G[5], Babu Kumar S[6], Anusha Preetham[7], Nagarathna C R[8]

[1]Associate Professor, Department of Information Science and Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, Karnataka, India, Pooja-ise@dsatm.edu.in

[2]Assistant Professor, BMS college of engineering , Karnataka, India, kanchanadixit23@gmail.com

[3]Assistant Professor, Department of CSE (IoT,Cybersecurity including Blockchain Technology), Dayananda Sagar College of Engineering, Bengaluru, Karnataka, India, keertipatil47@gmail.com

[4]Associate professor, Department of CSE, KSIT,Bangalore, Karnataka, India

[5]Associate Professor, Department of ISE, B.N.M. Institute of Technology, Bangalore, Karnataka ,India, nandinig@bnmit.in

[6]Assistant Professor, Christ University, Bangalore, Karnataka, India, babukumar.sbk@gmail.com

[7]Associate Professor, Department of Computer Science and Engineering, Dayananda Sagar College of Engineering, Bengaluru, Karnataka, India, raoanushapreetham@gmail.com

[8]Associate Professor, Department of AI and ML, BNM Institute of Technology, Bangalore, Karnataka, India, nagarathna.binu@gmail.com

Abstract— *In human-computer interactions, hand gesture recognition is crucial. As we can see, many new technological developments are taking place. One of these is biometric authentication, which we frequently see in smartphones. In a similar vein, hand gesture recognition is a contemporary method of human- computer interaction. Using this method, we can control our system by waving our hands in front of a webcam. Hand gesture recognition can be helpful for a variety of people. Utilizing machine learning is how the system's algorithm works. Without the use of a real mouse, the computer can be operated digitally and can execute left-click, right-click, scrolling, and computer cursor tasks based on hand gestures. Deep learning is the basis for the algorithm.*

KeyWords—Computer Vision, Open CV, Deep Learning, Image Processing.

I. INTRODUCTION

A computer mouse is an input tool that facilitates pointing and interaction with the object being pointed at [1]. Several other mice styles are currently popular, including the mechanical mouse, which is made of a single rubber ball that can rotate in any direction and controls the pointer's movement. Later, the optical mouse is introduced to replace the mechanical mouse[2]. A led sensor is used in optical mice to track the movement of the pointer. Years later, the laser mouse was developed to enhance precision and eliminate the shortcomings of the optical mouse. Later, when technology advanced significantly, wireless mice were developed to enable hassle-free movement of the mouse and to improve accuracy.

No matter how much the mouse's accuracy improves, there will always be limitations because the mouse is a hardware input device [3] and there can be some problems like a mouse click not working properly ad, etc. because the mouse a hardware device like any other physical object, the mouse will have a durability time within which it is functional and after its durability time we must change the mouse [4], [5].

Everything becomes virtualized as technology advances, including voice recognition [6][7]. The spoken language is recognized and translated into text using speech recognition technology. Thus, speech recognition and eye tracking, which utilize our eyes to operate the mouse pointer, could eventually replace keyboards[8],[9],[10]. The mouse may one day be replaced with eye tracking [11],[12],[13].

Gestures can take on any shape, such as a hand picture, pixel image, or any position that a human gives, as long as it doesn't demand a lot of computational resources or complexity to make the devices needed for recognition function [14]. Companies are putting forth various methods for gathering the data and

information required to build models for handwritten gesture recognition[15]. Certain models collaborate with specialized tools like data glove devices and color caps to establish a complex understanding of the gesture information provided by the user or human [16]. In a similar vein, OpenCV and PyAutogui make up the virtual mouse that we shall cover in this paper[17],. Webcam images are captured using the computer vision Python module known as OpenCV[24]. A Python package called Pyautogui is used to define keyboard and mouse operations. It used to be very different from how we operate and interact with systems . As a result, after the technology is produced, tools like standalone programmers and software are made and built for system ease of use. Artificial intelligence (AI) and machine learning (ML) are two blooming technologies that have been brought by technology and have let us effortlessly control and send commands to computers without any human involvement[18].

The strategies were created in a way that even machines and systems have the intelligence to recognize and distinguish between many things that are simple for humans but are now made simple for systems to comprehend using a new method of learning [19].With the use of hand gestures, users can manipulate a virtual mouse utilizing hand gesture recognition, and the system's webcam is used to track hand motions. Gesture recognition is accomplished using computer vision techniques. To collect data from a live video, OpenCV has a package called video capture.

II. METHOD A.

Open CV

Real-time computer vision is the primary focus of the open Python library known as Open CV (Open Source Computer Vision Library). It is also offered in Java and C++. It is a collection of free machine-learning software[20]. It utilizes Numpy, a Python module for creating multi-dimensional arrays and matrices as well as performing complex mathematical operations on these arrays. OpenCV primarily focuses on image processing and video capture because it is used to collect data from live video. OpenCV is mostly utilized for video capture in this study. Applications like face identification, optical character recognition (OCR), vision-guided robotic surgery, 3D human organ reconstruction, QR code scanner, etc. also use OpenCV. When analyzing films, we may estimate the motion in the video, remove the background, and track objects in it using OpenCV. We can also do specialized object detection, such as the detection of eyes, faces, and other specific items. The fundamental data structures used to create OpenCV applications, such as scalars, points, and others, are covered by OpenCV. 'import cv2' in Python is used to import the OpenCV library.

B. Anaconda

Data science, machine learning, data processing, predictive analysis, etc. are all specially done using the open source platform known as Anaconda for the computer languages of Python and R. Conda prompt, an Anaconda command prompt, and Anaconda Navigator, a desktop GUI application, make up Anaconda. Without utilizing command-line tools, Anaconda Navigator is used to run programs and manage conda packages[20] Using command-line commands, the Anaconda Prompt is used to run applications and manage conda packages. Jupyter Lab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, and Visual Studio are the pre-installed programs in Anaconda by default.

Anaconda Cloud is a cloud that belongs to Anaconda. We may access, save, and share both public and private notebooks, environments, and different conda and PyPI packages using this package management feature offered by Anaconda. Numerous Python packages, environments, and notebooks are hosted via Anaconda Cloud for a range of applications. We can use pip or conda to install Python packages with the aid of this cloud[13]. Since the data that the models are working with demands significant processing power, we are employing a system that is compatible with GPUs and powerful systems to run the models. This enables quick and simple user interaction with the models[21].

C. **Mediapipe**

The Google-developed open-source MediaPipe framework offers a cross-platform method for creating realtime multimedia processing pipelines[13]. Developers may build a variety of applications involving computer vision, machine learning, and audio processing because of the large selection of pre-built

components and algorithms it provides. Landmark identification is one of the primary features offered by MediaPipe[22]. Landmarks are identifiable and trackable precise spots or significant characteristics on things or human bodies. Applications such as augmented reality, virtual reality, facial recognition, gesture identification, and body tracking all benefit greatly from MediaPipe's landmark detection capabilities[23].

The landmark identification feature on MediaPipe makes use of deep learning models to recognize and pinpoint important points or landmarks on objects or human bodies within still or moving picture frames. These landmarks can stand in for body components like hands, joints, or skeletal structures or facial characteristics like the eyes, nose, and mouth. For landmark detection, the framework offers pre-trained models, but it also gives developers the option to train their models using unique datasets[24].
MediaPipe uses machine learning methods like convolutional neural networks (CNNs) and regression models to recognize landmarks[7]. To precisely identify and localize the desired landmarks, the models are trained using sizable annotated datasets[25]. Developers can track movement, estimate postures, measure distances, or add virtual upgrades to the detected items or human bodies after the landmarks have been identified.

The landmark identification module from MediaPipe is made with efficiency and real-time performance in mind[26]. It is suitable for applications that require real-time processing and responsiveness since it has a low latency for processing video streams or image sequences. Overall, MediaPipe's landmark detection capabilities provide developers with a powerful tool for incorporating real-time object and body tracking into their applications, enabling a wide range of interactive and immersive experiences[6][27].

III. System Implementation

To ensure correct functionality, the code implementation has several system prerequisites. It needs a Python environment with the following packages and their specific versions: screen-brightnesscontrol==0.9.0,pyautogui==0.9.53, opencv-python==4.5.3.56,mediapipe==0.8.6.2, comtypes==1.1.10, and pycaw==20181226. These containers may use pip, the Python package manager, to install.

1. 'pyautogui==0.9.53': This library offers cross-platform mouse and keyboard control. It enables the code to mimic keyboard input, mouse motions, and clicks. Make sure you have 'pyautogui' installed with version 0.9.53 or a similar version.
2. 'opencv-python==4.5.3.56': OpenCV is an effective computer vision library that enables image and video processing. The code uses OpenCV to recognize hands and capture video frames from the built-in camera. Ensure that you have 'opencv- python' 4.5.3.56 installed, or a similar version.
3. 'mediapipe==0.8.6.2': Mediapipe is a framework that offers ready-to-use, adaptable solutions for a range of perceptual problems, including hand tracking. The code makes use of Mediapipe to find and follow hand landmarks in video frames. Make sure you have 'mediapipe' installed with version 0.8.6.2 or a comparable version.
4. 'comtypes==1.1.10': Comtypes is a Python[28][29] module that enables interaction between the code and the Windows Component Object Model (COM) interfaces. The 'pycaw' library is used to regulate the system volume. Ensure that 'comtypes' is installed with version 1.1.10 or a similar version.
5. 'screen-brightness-control==0.9.0': This library enables system-wide control of the screen brightness. The system brightness is changed by the code using'screen- brightness-control' in response to hand movements Make sure'screen-brightness-control' is installed with version 0.9.0.

A. Model Understanding

1. 'get_hand_landmarks(image, results)': This function accepts as inputs an image and the hand detection results. It takes the hand landmarks' coordinates out of the results and returns a list of them. The landmarks stand in for particular parts of the hand, like the joints and fingertips.

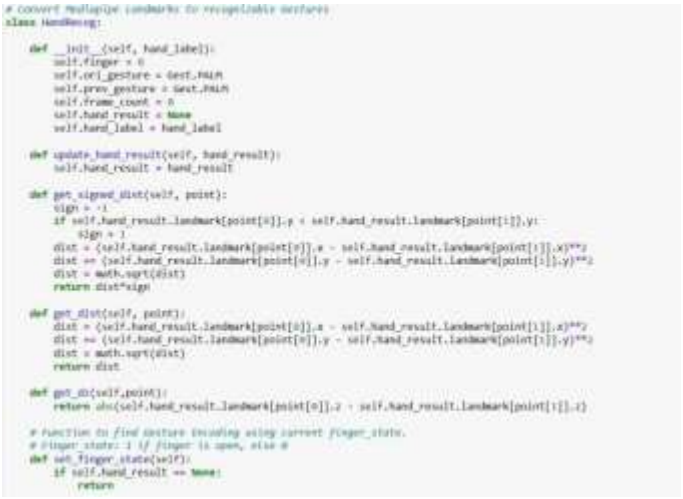


Fig 1. This part of the code is to get the landmarks on the hand.

2. 'pinch_control_init(hand_result)': This function is used to set up the pinch control functionality. The starting location and depth of the hand pinch motion are defined using the hand landmarks as input. To calculate the successive changes in depth during the pinch gesture, this initialization is required.

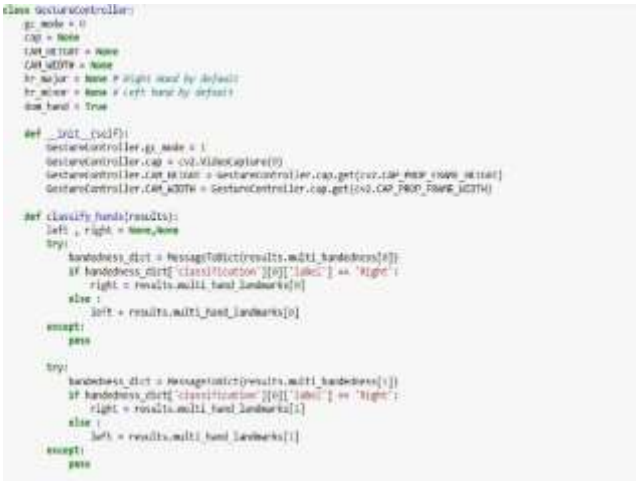


Fig 2 This part of the code is used for gesture control.

3. 'pinch_control(hand_result, scrollHorizontal, scrollVertical)': Based on the hand landmarks and scroll direction, this function manages the pinch control. It computes the depth difference between the pinch gesture's original depth and the hand's current depth. Using the 'pyautogui' and'screen_brightness_control' libraries, it executes actions like horizontal scrolling, vertical scrolling, altering system brightness, or changing system volume, depending on the scroll direction[30] (horizontal or vertical).

4. 'handle_controls(gesture, hand_result)': Based on the recognized gesture[30], this function takes the appropriate action. It executes the appropriate action after receiving information from the gesture and hand landmarks. For instance, based on the recognized movements, it moves the mouse pointer, makes left- or right-clicks, double- clicks, or engages pinch control.

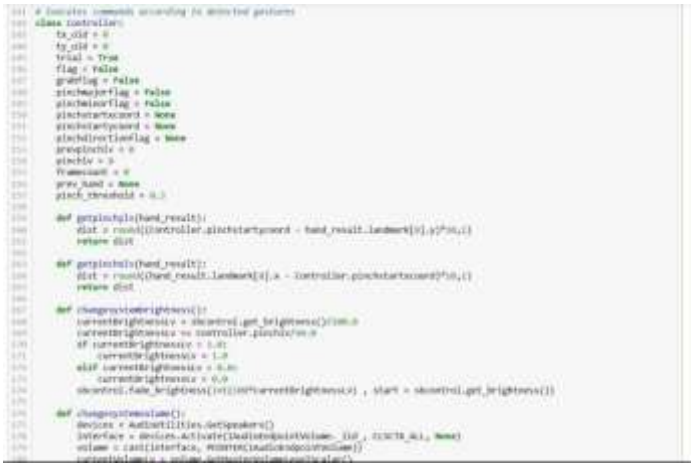


Fig 3. This part of the code takes appropriate actions based on the recognized gesture.

5. The main function, "main(),"[31] carries out the logic for recognizing and controlling hand gestures.It launches a video capture from the built-in camera and starts the'mp_hands' module for hand detection.The hand detection module continuously reads frames from the camera, analyses them, and extracts the hand landmarks. The 'handle_controls' function is then used to carry out operations based on the gestures that were discovered. The'mp_drawing' module is used to display the processed frames with marked landmarks.

IV. RESULTS

1. Cursor Stop

Let H be the hand gesture state described by the hand landmarks and finger states. The hand gesture state H satisfies the "Cursor Stop" condition if and only if H matches the hand form depicted in Fig 4, and regardless of subsequent hand movements, the cursor remains stationary

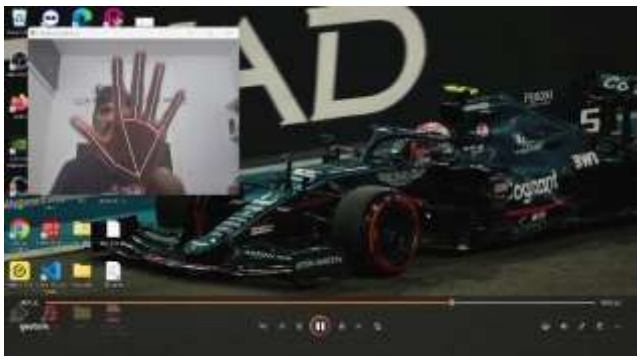


Fig 4. Hand indication for stopping the movement of the cursor.

1. Cursor Move

Let H be the hand gesture state described by the hand landmarks and finger states. The hand gesture state H satisfies the "Cursor Move" condition if and only if H matches the hand form depicted in Fig 5, and the cursor moves in the direction of the hand movement.

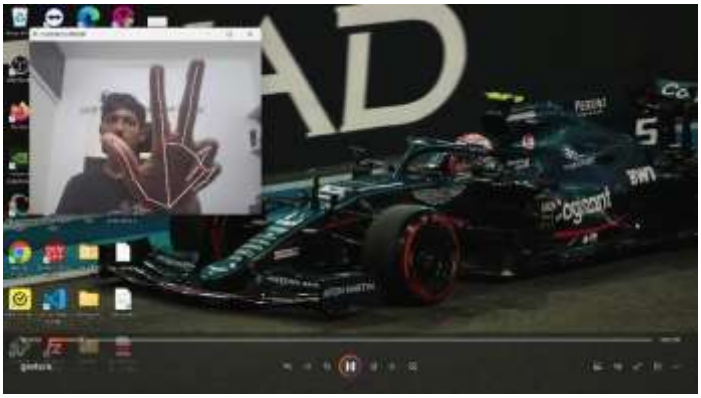


Fig 5. Hand indication to move the cursor around.

2. Left Click

Let H be the hand gesture state described by the hand landmarks and finger states. The hand gesture state H satisfies the "Left Click" action if and only if H matches the hand form depicted in Fig 6, and the left click action is performed on the computer

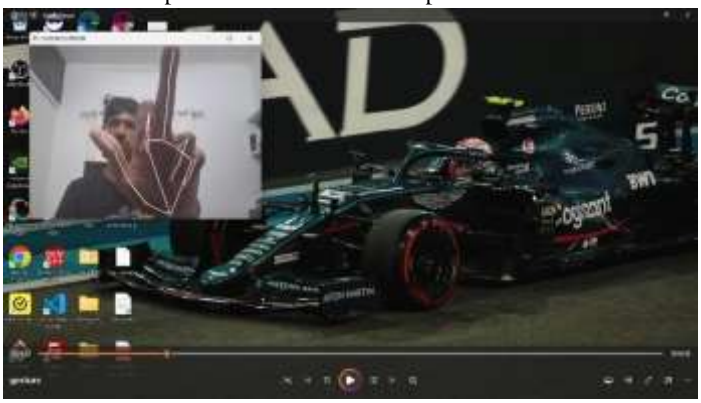


Fig 6. Hand indication for left click.

3. Right Click

Let H be the hand gesture state described by the hand landmarks and finger states. The hand gesture state H satisfies the "Right Click" action if and only if H matches the hand form depicted in Fig 7, and the Right-click action is performed on the computer.

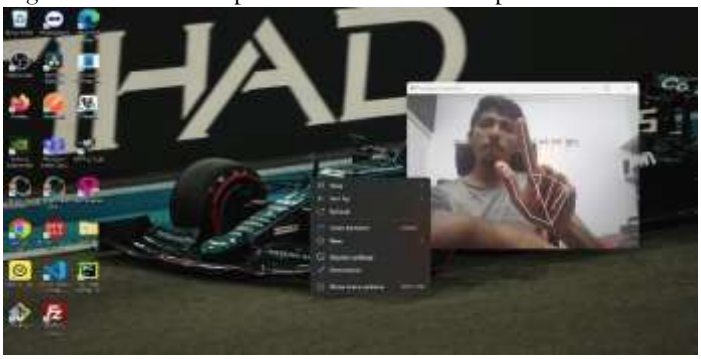
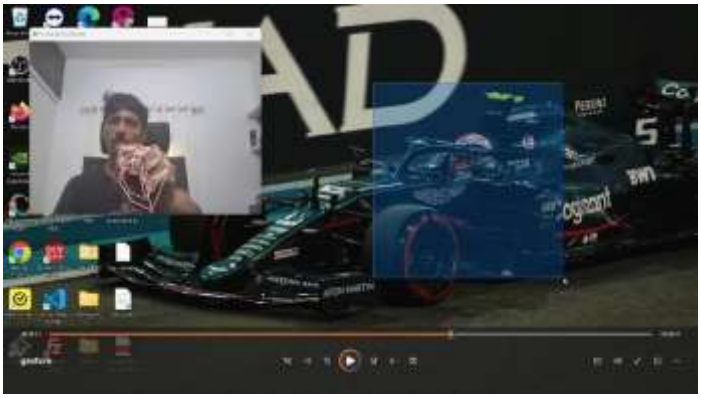


Fig 7. Hand Indication for right click.

4. Multiple Options Select

Let H be the hand gesture state described by the hand landmarks and finger states. The hand gesture state H satisfies the "Multiple Options Selection" action if and only if H matches the hand form depicted in Fig 8, and the action of clicking on a mouse and dragging is performed.



.Fig 8. Hand Indication for Multiple Options Select.

V. Further Development

There are numerous prospects for advancement in the realm of color detection models. We can distinguish particular colors in colored photos thanks to these models. Furthermore, improvements in mouse movement technology enable the development of virtual mice with functionality that is identical to that of a real mouse. These developments make it possible to utilize the keyboard and mouse virtually, rather than physically, to traverse computer systems. Convolutional neural networks (CNNs)[32] are used to increase the performance of these models during training, leading to better results. These models can be developed using a variety of methods, including the use of innovative software packages like "pyauto GUI." Using this technology, commands can be sent to recognize particular inputs and run associated system operations. For example, if a specific color is found, the model

VI. Conclusion

This model can be brought to a conclusion by utilizing computer vision concepts like open CV, color variation techniques, and the development of mouse movement through the use of specific packages like "mouse," which will be used to move the mouse using coordinates related to the detected color. This can make systems and numerous other applications easier to use. As a result, the open CV offers its customers a variety of models that are easily accessible and will simplify their lives.

VII. References

1. zhang, z., & li, j. (2018). robust real-time hand gesture recognition using depth sensors. multimedia tools and applications, 77(22), 29247-29266.
2. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. In CVPR (pp. 7291-7299).
3. Rangegowda, N. C., Mohanchandra, K., Preetham, A., Almas, M., & Huliyappa, H. (2023). A Multi-Layer Perceptron Network-Based Model for Classifying Stages of Alzheimer's Disease Using Clinical Data. Revue d'Intelligence Artificielle, 37(3), 601.
4. Nagarathna, C. R., Jayasri, A., Chandana, S., & Amrutha, A. (2023). Identification of Image Forgeries using Machine Learning-A Review. Journal of Innovative Image Processing, 5(3), 323-336.
5. M. U, T. R M, N. C R and Y. T. Ravikumar, "Face Recognition using MTCNN, Inception - Resnet with Ensemble Approach," 2025 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE), Bangalore, India, 2025, pp. 1-6, doi: 10.1109/IITCEE64140.2025.10915271.
6. Chen, Y., Wang, Z., Peng, S., Zhang, G., & Yu, J. (2018). CV-GAN: a semantic-preserving content-based image retrieval framework using conditional GANs. Multimedia Tools and Applications, 77(19), 25153-25173.
7. Preetham, A., Vyas, S., Kumar, M., & Kumar, S. N. P. (2024). Optimized convolutional neural network for land cover classification via improved lion algorithm. Transactions in GIS, 28(4), 769-789.
8. Preetham, A., & Battu, V. V. (2023). Soil Moisture Retrieval Using Sail Squirrel Search Optimization-based Deep Convolutional Neural Network with Sentinel-1 Images. International Journal of Image and Graphics, 23(05), 2350048.
9. Zhang, D., Jiao, L., Ren, S., & Ren, Z. (2021). A deep learning-based virtual mouse system for controlling computer cursor. Journal of Ambient Intelligence and Humanized Computing, 12(2), 2511-2523.
10. Wu, C., Yang, Y., Liu, M. Y., Luo, P., & Wang, X. (2020). CPM-RF: Real-time hand pose estimation with cascaded pose regression forest. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp.7982-7991).
11. Nagarathna, C. R., & Kusuma, M. M. (2023). Early detection of Alzheimer's Disease using MRI images and deep learning techniques. Alzheimer's & Dementia, 19, e062076.

12. Chandana, S., Nagarathna, C. R., Amrutha, A., & Jayasri, A. (2024, January). Detection of image forgery using error level analysis. In 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE) (pp. 1-5). IEEE.
13. Baek, S. H., & Kim, K. I. (2018). Efficient video-based hand pose estimation for real-time applications. In European Conference on Computer Vision (pp. 102-117). Springer, Cham.
14. Bhavyashree, H. L., Nagarathna, C. R., Preetham, A., & Priyanka, R. (2019, March). Modified cluster based certificate blocking of misbehaving node in MANETS. In 2019 1st international conference on advanced technologies in intelligent control, environment, computing & communication engineering (ICATIECE) (pp. 155-161). IEEE.
15. Nagarathna, C. R., Kusuma, M., & Seemanthini, K. (2023). Classifying the stages of Alzheimer's disease by using multi layer feed forward neural network. Procedia computer science, 218, 1845-1856.
16. Nagarathna, C. R., & Mohanchandra, K. (2023). An overview of early detection of Alzheimer's disease. International Journal of Medical Engineering and Informatics, 15(5), 442-457.
17. Tokola, R. A. (2020). Blob detection using computer vision. International Journal of Applied Engineering Research, 15(16), 329-334.
18. Wang, Y., Liu, Z., Zhu, X., & Gong, X. (2019). Real-time hand gesture recognition and virtual mouse control based on a depth camera. Multimedia Tools and Applications, 78(1), 235-253.
19. Rangegowda, N. C., Mohanchandra, K., Preetham, A., Almas, M., & Huliyappa, H. (2023). A Multi-Layer Perceptron Network-Based Model for Classifying Stages of Alzheimer's Disease Using Clinical Data. Revue d'Intelligence Artificielle, 37(3), 601.
20. Rahim, M. A., Yao, H., Shah, S. A. A., Saroj, A. B. M., & Song, S. (2021).Real-time hand gesture recognition using deep learning for human-computer interaction. In 2021 2nd International Conference on Smart Grid and Renewable Energy (pp. 116-120). IEEE.
21. Himaja, G., Kundan, K. M., & Nagarathna, C. R. (2024). Car parking slot detection using Hough line transform and CNN model. In Computer Science Engineering (pp. 703-711). CRC Press.
22. Xiang, X., Xu, J., Zhou, D., & Wang, R. (2021). A lightweight virtual mouse control system based on motion recognition. Journal of Ambient Intelligence and Humanized Computing, 12(4), 5109-5121.
23. Nagamalla, V., Kumar, B. M., Janu, N., Preetham, A., Parambil Gangadharan, S. M., Alqahtani, M. A., & Ratna, R. (2022). Detection of adulteration in food using recurrent neural network with internet of things. Journal of Food Quality, 2022(1), 6163649.
24. Debnath, S., Preetham, A., Vuppu, S., & Kumar, S. N. P. (2023). Optimal weighted GAN and U-Net based segmentation for phenotypic trait estimation of crops using Taylor Coot algorithm. Applied Soft Computing, 144, 110396.
25. Nagarathna, C. R., Bhagya, K. G., Devaraju, H. B., Somashekara, A. S., & Kishore, H. N. (2025). Machine Learning Techniques for EEG-Based Alzheimer's Disease Classification. Mathematical Modelling of Engineering Problems, 12(3).
26. Al-Hammadi, M., Muhammad, G., Abdul, W., Alsulaiman, M., Bencherif, M. A., Alrayes, T. S., ... & Mekhtiche, M. A. (2020). Deep learning-based approach for sign language gesture recognition with efficient hand gesture representation. Ieee Access, 8, 192527-192542.
27. Tan, Y. S., Lim, K. M., & Lee, C. P. (2021). Hand gesture recognition via enhanced densely connected convolutional neural network. Expert Systems with Applications, 175, 114797.
28. Emporio, M., Ghasemaghaei, A., Laviola Jr, J. J., & Giachetti, A. (2025). Continuous hand gesture recognition: Benchmarks and methods. Computer Vision and Image Understanding, 104435.
29. Cui, C., Sunar, M. S., & Su, G. E. (2025). Deep vision-based real-time hand gesture recognition: a review. PeerJ Computer Science, 11, e2921.
30. Amprimo, G., Masi, G., Pettiti, G., Olmo, G., Priano, L., & Ferraris, C. (2024). Hand tracking for clinical applications: Validation of the Google MediaPipe Hand (GMH) and the depth-enhanced GMH-D frameworks. Biomedical Signal Processing and Control, 96, 106508.
31. Meng, Y., Jiang, H., Duan, N., & Wen, H. (2024). Real-Time Hand Gesture Monitoring Model Based on MediaPipe's Registerable System. Sensors, 24(19), 6262.
32. Grif, H. S., & Turc, T. (2018). Human hand gesture based system for mouse cursor control. Procedia Manufacturing, 22, 1038-1042.