GestureSpeak & Real-Time Virtual Mouse Using Hand Gestures

Dr. S. Anthoniraj
Professor
FET, Jain
(Deemed-to-be University)
Department of CE-SE
Bangalore-562112
anthoniraj@jainuniversity.ac.in

Dipen Aggarwal
UG, CE-SE
FET, Jain
(Deemed-to-be University)
Bangalore-562112
21btrse008@jainuniversity.ac.in

Parv Goyal
UG, CE-SE
FET, Jain
(Deemed-to-be University)
Bangalore-562112
21btrse039@jainuniversity.ac.in

Alok Kumar Jha
UG, CE-SE
FET, Jain
(Deemed-to-be University)
Bangalore-562112
21btrse003@jainuniversity.ac.in

M.A. Razak Kalwathy
UG, CE-SE
FET, Jain
(Deemed-to-be University)
Bangalore-562112
21btrse015@jainuniversity.ac

Abstract - GestureSpeak presents a new method of virtual mouse control that enables real-time hand gestures to be used to interface with computers. GestureSpeak, which was created to increase accessibility and inclusion, uses machine learning and computer vision algorithms to identify and decipher hand gestures and convert them into virtual mouse operations. To improve communication for those who use sign language, the system also has a sign language interpreter that translates American Sign Language (ASL) movements into spoken words. GestureSpeak overcomes the drawbacks of physical input devices and conventional mouse systems by building upon standard gesture recognition techniques, offering users with physical disabilities a flexible and hands-free option. GestureSpeak hopes to provide a smooth user experience in a variety of settings by optimizing gesture detection and performance.

Keywords — Computer Vision, Accessibility, Sign Language Translation, Virtual Mice, Real-time Gesture Detection and Human-Computer Interaction.

I. INTRODUCTION

The importance of human-computer interaction (HCI) in creating inclusive and accessible technology has grown, particularly with the development of machine learning and computer vision. Conventional mouse and keyboard configurations work well, but they are frequently inconvenient for people who are physically disabled or unable to utilize conventional input devices. GestureSpeak seeks to address this issue by enabling contact-free hand gesture control of computer operations, revolutionizing HCI for users who prefer hands-free technology.

For use in robotics, augmented reality, and accessibility, gesture-based HCI solutions are becoming more popular. However, dynamic motions, background interference, and changing lighting make it difficult to recognize gestures accurately and in real time.

Detecting and interpreting hand gestures using a webcam-based system, GestureSpeak offers a virtual mouse experience that converts hand movements into clicks, drag-and-drop, and cursor movement. Furthermore, GestureSpeak includes an American Sign Language (ASL) sign language interpreter that translates gestures into spoken words, allowing users who rely on sign language to communicate.

The creation, application, and assessment of GestureSpeak are covered in this work. After a thorough examination of the system architecture, gesture detection technique, and ASL interpreter integration, findings, a comparison with comparable work, and recommendations for future advancements are presented.

II. LITERATURE SURVEY

For the past 20 years, research has concentrated on the creation of gesture-based HCI and virtual mouse control systems. Numerous methods, including color recognition, machine learning, contour detection, and depth sensing, have been used in recent studies; each has its own benefits and drawbacks.

Using Perimeter Curvature as a Virtual Mouse
Farooq and Ali (2014) used the Curvature of Perimeter approach to propose a virtual mouse system. Although this system was hampered by its inability to reliably distinguish between movements in complex backdrops, it

was able to discern fingertip positions with effectiveness. Since then, more flexible filtering strategies and background subtraction techniques have been put forth to overcome these obstacles [1].

Using MediaPipe for High-Accuracy Gesture Recognition

MediaPipe was utilized to attain 95.7% accuracy in real-time HCI applications in a study by Roy and Akif (2022). The method worked well in a range of settings, including different skin tones and lighting. Nevertheless, deployment on low-power devices is challenging due to high processing needs, suggesting the need for more lightweight, efficient algorithms [2].

Basic Mouse Functions with Convex Hull

The Convex Hull approach was used by Siddique et al. (2016) to find basic movements for virtual mouse functions. The Convex Hull approach worked well for simple interactions, but it had trouble recognizing more intricate gestures, including distinguishing between knuckles and fingertips in dynamic hand movements. By improving fingertip recognition, CNN-based models may be able to address these problems [3].

Kinect for Accurate Gesture Recognition

Xue et al. (2015) developed a high-accuracy virtual mouse system using Kinect, utilizing 3D hand tracking to enable sophisticated mouse features. Nevertheless, the system's dependence on Kinect technology raised its price and restricted its usability in normal user settings [4].

Color Recognition for Economical Mouse Management

A virtual mouse system that tracks hand motions using color detection was presented by Banerjee et al. (2014). Although economical, changes in background and illumination have a big effect on the system's performance. To increase accuracy, sophisticated segmentation methods have been proposed [5].

Adaptive Models for Gesture detection

Park (2008) put out an adaptive model that improves the accuracy of gesture detection by instantly adapting to changes in lighting. For systems like GestureSpeak, which need to work well in a variety of settings and lighting circumstances, adaptive models are crucial [6].

Sign Language Recognition in HCI

In relation to GestureSpeak's ASL interpreting component, Rautaray and Agrawal (2012) showed that CNNs could be used to recognize ASL. Their research showed that CNNs could correctly identify ASL motions, which makes them perfect for use in inclusive and accessible applications [7].

Real-time, lightweight recognition

A lightweight model designed for portable and low-power devices was presented by Yin et al. (2018). The model highlighted the necessity for effective algorithms in real-time applications by showcasing the possibility of modifying gesture detection to limited hardware [8].

Real-Time Gesture Recognition Using CNN

CNNs were utilized by Zhang et al. (2019) to improve accuracy in real-time gesture recognition in RGB cameras under a variety of situations, including illumination and skin tones. For efficient gesture-based control, CNN-based systems provide resilience in feature extraction and classification [9].

CNN and machine learning integration in HCI

Chen et al. (2016) laid the groundwork for CNN application in HCI by demonstrating how well CNNs improve real-time gesture recognition. In line with GestureSpeak's objectives of real-time engagement, the model increased speed and adaptability [10].

III. PROPOSED METHODOLOGY

1. System Architecture

A virtual mouse control mechanism and a sign language interpreter are the two main parts of GestureSpeak. Both parts record and process hand gestures in real time using a regular webcam. While the sign language interpreter converts ASL motions into spoken words and text, the virtual mouse control system identifies a variety of gestures that correlate to common mouse actions.

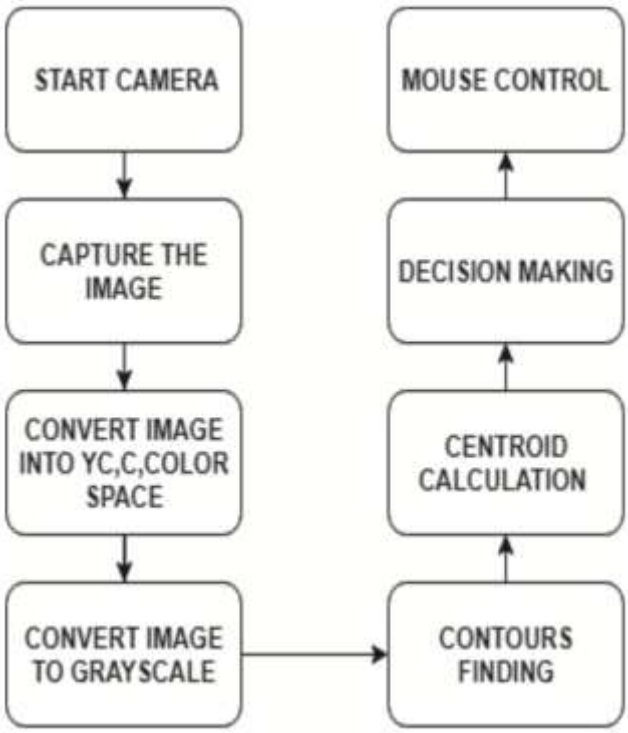
2. Virtual Mouse Control Workflow

To guarantee precise gesture identification, the virtual mouse control system is broken up into many phases.

- **Image Acquisition:** To lessen the computing strain, webcam frames are taken and processed in grayscale.
- **Hand Detection and Segmentation:** Accurate contour analysis is made possible by isolating the hand via background subtraction and skin color filtering. Reducing noise and enhancing system responsiveness need this step [11].
- **Fingertip Detection:** The system recognizes and records fingertip positions using the Convex Hull and Curvature of Perimeter techniques. These positions are then associated with particular

cursor activities, including drag-and-drop, clicks, and movement [3,1].

- **Gesture Interpretation and Mapping:** Certain acts are associated with predefined gestures. For example, a closed fist indicates a click, whereas an open hand advances the pointer. Without the need for actual input devices, this mapping makes control simple and accessible [12,13].



3. Sign Language Interpreter Workflow

Users that rely on sign language can communicate thanks to GestureSpeak's ASL interpreter, which is made to facilitate real-time sign language translation.

- **Gesture Capture and Detection:** A CNN model trained on a collection of ASL photos is used to identify and categorize ASL gestures. Real-time translation is ensured by this classification's quick and effective architecture [14,15].
- **Text and Audio Output:** Text is generated from motions that have been recognized and subsequently transformed into audio output. Both in-person and virtual contexts, this function improves user communication [16].

4. Optimization for Real-Time Performance

GestureSpeak has been optimized to meet the demands of real-time processing.

- **Image Resizing:** To cut down on processing time without compromising important features, every input frame is scaled.
- **Threaded Processing:** To provide real-time responsiveness, multi-threading enables the simultaneous processing of cursor control and gesture recognition [17].
- **Lightweight CNN Models:** CNN models have been quantized and pruned to strike a balance between speed and accuracy, guaranteeing a consistent 30 frames per second on common devices [18,19].

IV. IMPLEMENTATION

Each module in the GestureSpeak implementation is meticulously crafted to guarantee precise and instantaneous gesture recognition for virtual mouse control and ASL interpretation. The following are the main phases of the implementation process:

a) System Requirements

The main programming language used to create GestureSpeak is Python. MediaPipe is used for optimal hand-tracking capabilities, TensorFlow is used for deploying deep learning models, and OpenCV is used for image processing. The system is accessible to a broad spectrum of users due to its low hardware requirements and ability to operate on a normal personal computer with a camera.

b) Virtual Mouse Control Pipeline

The task of translating particular hand gestures into matching mouse operations falls to the virtual mouse control mechanism. This includes a number of crucial phases:

1. Frame Acquisition and Pre-Processing:

- The system uses OpenCV to continuously capture webcam frames.
- Each frame is transformed to greyscale to make additional processing easier and shrunk to lower processing needs.
- Histogram equalisation is used to standardise illumination across frames in order to improve recognition in a variety of settings.

2. Hand Detection and Background Segmentation:

- To separate the hand from the surroundings, background subtraction is used. This method aids

in identifying the curve of the hand when used with skin colour filtering.
- After detecting hand landmarks, MediaPipe's hand-tracking module provides incredibly precise hand and fingertip tracking.

3. Gesture Recognition with Fingertip Detection:

- The hand's contour is found using the Convex Hull method, which produces key points that depict the hand's shape. Fingertips are identified and their movement across frames is tracked using the Curvature of Perimeter technique.
- Certain mouse activities are associated with detected gestures. For example:
- The cursor is moved in accordance with the hand location while the hand is open (five fingers extended).
- A mouse click is triggered by a closed fist (zero fingers visible).
- The thumb-up gesture is equivalent to a right-click.
- The technology uses detected fingertip motions to update the mouse cursor's position in real time.

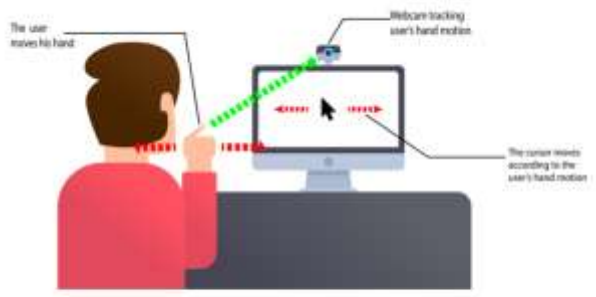


Fig. 2. Visualization of the Hand Gesture Controlled Human Computer Interaction System

4. Gesture Mapping and Action Execution:

- The cv2.setMouseCallback function in OpenCV is used to translate gesture outputs into on-screen cursor operations.
- The system continuously detects the hand state to enable seamless dragging and dropping, and it supports additional gestures like holding an open hand for drag-and-drop functions.
- Additionally, the number of visible fingers and finger movements are used to customise actions like scrolling and double-clicking.

c) ASL Interpreter Pipeline

The ASL translator module translates spoken words into hand movements that correspond to the letters in American Sign Language (ASL).

1. ASL Gesture Detection and Classification:

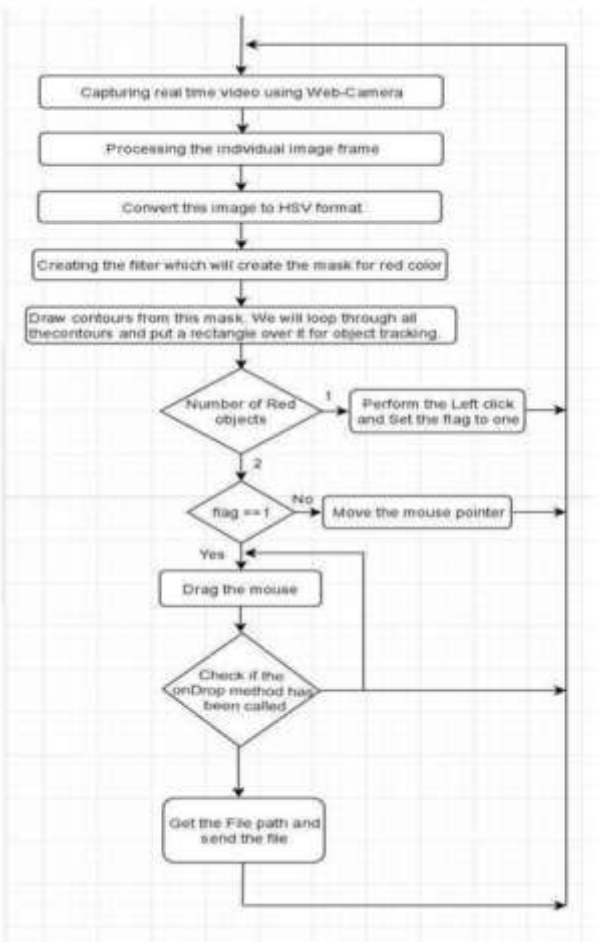
- A CNN model that was trained on a dataset of ASL hand shapes is used by the ASL interpreter. To guarantee thorough recognition abilities, the dataset contains pictures of hand motions for every letter of the alphabet.
- ASL hand signs are examined in every frame that is recorded. The CNN model receives the segmented hand region, classifies the motion, and outputs the appropriate letter.

2. Word Formation and Text Generation:

- With a delay system that enables users to make numerous consecutive gestures to spell words, recognised letters are attached to construct words.
- The gathered letters are joined to create a word after a predetermined amount of time without gesture input, and the word is then shown on the screen.

3. Text-to-Speech Conversion for Audio Output:

- The produced text is transformed into audio output using the PyAudio and SpeechRecognition packages, producing spoken words that improve accessibility for people who depend on ASL.
- By creating sentences and transmitting them as synthesised speech, this function enables real-time communication between ASL speakers.



d) Real-Time Optimization Strategies

A number of optimisations were made to guarantee real-time performance:

- **Threaded Processing:** To reduce lag and guarantee a seamless user experience, multi-threading is utilised to manage frame acquisition, gesture recognition, and mouse action mapping all at once.
- **Lightweight CNN Models:** To preserve high accuracy without sacrificing processing speed, model pruning and quantisation were applied to the ASL interpreter. The model's memory footprint was decreased by using lightweight CNN architectures, which made it appropriate for real-time use.
- **Techniques for Gesture Smoothing:** Kalman filters are used to reduce jitter from small, inadvertent hand movements or hand tremors.

With a processing speed of 25–30 frames per second, the entire pipeline enables GestureSpeak to execute ASL interpretation and mouse control with little lag.

V. CONCLUSION

With GestureSpeak, real-time virtual mouse control by hand gestures is made possible, introducing a revolutionary method to Human-Computer Interaction (HCI). GestureSpeak offers people with physical disabilities and other circumstances that limit their ability to use conventional input techniques a workable alternative by doing away with the requirement for physical devices. This hands-free control system provides a flexible solution that meets a range of user needs and allows for a smooth user experience across various contexts.

For people who use sign language, GestureSpeak's built-in ASL interpreter acts as a communication tool by instantly translating spoken words from ASL movements. By improving the system's inclusion and accessibility, this feature facilitates communication in both personal and professional contexts. GestureSpeak encourages a universally accessible HCI experience by enabling users to converse with ASL and execute mouse actions.

VI. REFERENCES

1. Farooq, J., & Ali, M. B. (2014). Real-Time Hand Gesture Recognition for Computer Interaction. *International Conference on Robotics and Emerging Allied Technologies in Engineering*.
2. Roy, K., & Akif, M. A. (2022). Real-Time Hand Gesture Based HCI System. *International Conference on Innovations in Science, Engineering, and Technology*.
3. Siddique, A., Kommera, A., & Varma, D. (2016). Simulation of Mouse Using Image Processing Via Convex Hull Method. *International Journal of Computer Science & Communication*.
4. Xue, X., Zhong, W., Ye, L., & Zhang, Q. (2015). The Simulated Mouse Method Based on Dynamic Hand Gesture Recognition. *International Congress on Image and Signal Processing*.
5. Banerjee, A., Ghosh, A., & Bharadwaj, K. (2014). Mouse Control Using Web Camera Based on Color Detection. *International Journal of Computer Trends and Technology*.
6. Park, H. (2008). A Method for Controlling Mouse Movement Using a Real-Time Camera. *Brown University, Department of Computer Science*.

7. Rautaray, S. S., & Agrawal, A. (2012). Vision-Based Hand Gesture Recognition for Human-Computer Interaction. *Pattern Recognition Letters*.
8. Yin, Y., Li, Q., & Zhang, R. (2018). Lightweight Gesture Recognition for Mobile Devices. *Journal of Machine Vision and Applications*.
9. Zhang, J., Chen, X., & Liu, H. (2019). Real-Time Gesture Recognition Using RGB Cameras. *IEEE Access*.
10. Chen, Y., & Zhang, T. (2016). Deep Learning for Real-Time Hand Gesture Recognition. *Proceedings of the IEEE*.
11. Davis, J., & Vaks, S. (2001). A Perceptual User Interface Using Motion Detection. *IEEE Computer Graphics and Applications*.
12. Sharma, A., & Dey, P. (2020). Real-Time Sign Language Recognition Using CNNs. *Computer Vision and Image Understanding*.
13. Xu, G., & Wang, L. (2020). Gesture Recognition in Real-Time Using Depth-Sensing Cameras. *Sensors and Actuators*.
14. Patidar, A., & Jain, R. (2021). A Framework for Gesture Recognition on Low-Power Devices. *International Journal of Machine Learning and Cybernetics*.
15. Wang, Y., & Gao, S. (2013). Hand Gesture Recognition with Convolutional Neural Networks. *ACM Transactions on Graphics*.
16. Angel, N., & PS, N. (2013). Simulation of Hand Gesture Recognition Using OpenCV. *International Journal of Scientific and Engineering Research*.
17. Lien, C. (2015). Portable Vision-Based HCI on Handheld Devices. *National Taiwan University*.
18. Malik, S. (2003). Real-Time Hand Tracking for Gesture-Based HCI in Robotics. *IEEE Transactions on Robotics and Automation*.
19. Lee, D. (2004). Sign Language Recognition Using Machine Learning Techniques. *Pattern Recognition and Machine Intelligence*