© June 2025 | IJIRT | Volume 12 Issue 1 | ISSN: 2349-6002

Zephyr Cursor Movement

Anzalna Aslamkutty[1], Aswanth S Krishna[2], Czyril Imam[3], Jesna Jeemon[4], Ms. Aparna Chandran[5]
Rajadhani Institute of Engineering & Technology

***Abstract*—In the evolving landscape of Human-Computer Interaction (HCI), traditional input devices such as mice and touchpads present inherent limitations in accessibility and efficiency. This project introduces Zephyr Cursor Movement, an innovative approach to cursor control using real time hand gesture recognition. The system leverages Python, OpenCV, and MediaPipe to detect predefined hand gestures and translate them into cursor movements and various mouse actions, including left click, right click, double click, and drag and drop. Additionally, Zephyr incorporates an intelligent voice assistant, activated by the command "Hey Zephyr," to enhance user interaction. Special gesture-based controls are implemented for zooming, scrolling, and volume adjustments ensuring seamless and intuitive experience. By eliminating physical input constraints, Zephyr enhances accessibility not only for individuals with motor impairments but also for users in multitasking environments and hygiene-sensitive settings. The proposed system demonstrates high accuracy in gesture detection and responsiveness, making it an alternative for next generation human-machine interfaces.**

***Index Terms*—Human-Computer Interaction, Hand Gesture Recognition, Cursor Control, OpenCV, MediaPipe.**

I. INTRODUCTION

with advancements in human-computer interaction (HCI), touch-free input mechanisms are gaining prominence. Conventional input devices, such as the mouse and keyboard, have been the primary means of interacting with computers for decades. While these methods are effective, they presentaccessibility challenges for individuals with disabilities and and limitations in certain environments where hands-free operation is preferable.The emergence of gesture-based interaction and voice-controlled systems has paved the way for more intuitive, natural, and user-friendly computing experiences. By enabling users to control a cursor through hand gestures, Zephyr Cursor Movement eliminates the need for physical contact, making computing more inclusive and efficient. This system is especially beneficial for individuals with motor impairments, medical professionals in sterile environments, and users seeking an alternative input method for convenience.Zephyr Cursor Movement integrates computer vision and machine learning algorithms to recognize hand gestures in real-time and translate them into precise cursor actions. Additionally, voice recognition capabilities are incorporated to allow users to perform essential system functions, such as adjusting brightness and volume, further enhancing the hands-free experience.

II. SYSTEM ARCHITECTURE

The project utilizes a webcam to track hand gestures and control the computer cursor without physical contact. Using OpenCV and MediaPipe, the system detects hand landmarks in real time and identifies specific gestures. The cursor moves according to the index finger's position, while combinations of fingers trigger actions like left click, right click, and double click. Additional gestures enable drag and drop, zooming, scrolling, and volume control. PyAutoGUI is used to simulate mouse movements and clicks based on detected gestures. This approach provides an intuitive, touchless way to interact with a computer, enhancing accessibility and user experience.

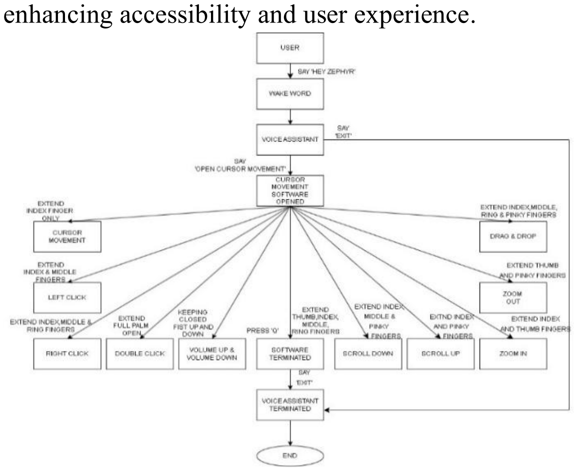

Fig. 1. System Architecture Diagram and Flowchart

This figure illustrates the sequential process of capturing hand gestures, processing them, mapping them to cursor actions, and executing the corresponding mouse movements and clicks.

A. Input Layer

The input layer forms the foundation of the system by collecting all raw data required for gesture and voice-based interaction. It interacts directly with the user's physical environment through the following components:

1. Webcam: A standard USB or built-in webcam captures continuous video frames, serving as the input stream for hand gesture detection. The video is fed into the gesture recognition pipeline in real time.
2. Microphone: The system continuously listens through the system's microphone for the predefined wake word "Hey Zephyr". Once detected, it triggers the voice processing pipeline to accept and execute verbal commands.

This layer ensures high responsiveness and uninterrupted data acquisition, which are critical for real-time human-computer interaction.

B. Processing Layer

This layer is responsible for interpreting the raw inputs acquired from the input layer and converting them into meaningful data representations for decision-making.

1. Gesture Recognition Module

- OpenCV is used to manage the video feed, extract individual frames, and preprocess images (e.g., color space conversion, flipping, cropping).
- MediaPipe is used to identify and track hand landmarks within each video frame. These landmarks are then used to classify specific hand poses corresponding to user gestures such as index finger pointing (cursor movement), two fingers extended (left click), or palm open (double click).

This module ensures low-latency gesture detection with reasonable accuracy and robustness, even under varying lighting conditions.

2. Voice Command Module

Wake Word Detection: A lightweight engine like Porcupine or Speech Recognition library is employed to detect the phrase "Hey Zephyr". This event activates the assistant for a short listening window.

Speech Recognition: Once triggered, the system captures the user's spoken input and converts it to text using either offline libraries or online services such as Google Speech Recognition.

C. Logic Layer (Decision and Mapping)

Once the inputs are processed, the logic layer performs the role of interpreting and mapping them to predefined software actions.

1. Gesture Mapping Engine: Matches specific hand gestures to cursor control functions. For example:
- Single index finger: Move cursor
- Index + middle finger: Left click
- Three fingers: Right click
- Palm open: Double click
- Thumb + pinky: Scroll
2. Voice Command Interpreter: Maps recognized voice commands to corresponding system operations.

D. Action Execution Layer

This layer is responsible for performing system-level actions:

1. Uses PyAutoGUI to execute mouse events: movement, clicks, dragging

III. IMPLEMENTATION

A. Wake Word

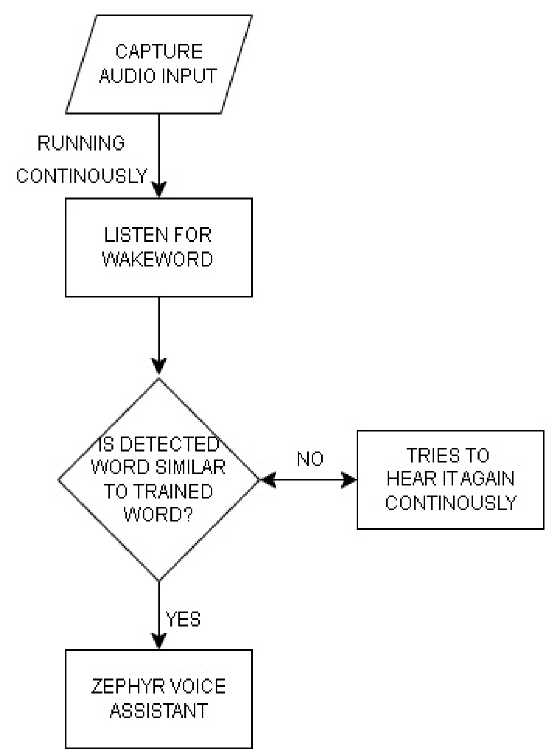

Fig. 2. Zephyr Wake Word

The wake word detection system continuously runs in the background, listening for the predefined phrase "Hey Zephyr" using a speech recognition model. It captures audio input through the microphone and processes it in real time using libraries like speech recognition or Porcupine. When the wake word is detected, the system triggers the execution of the Zephyr voice assistant, launching it automatically without requiring manual activation. This enables hands-free interaction, allowing users to invoke Zephyr at any time by simply speaking the wake word.

B. Voice Assistant

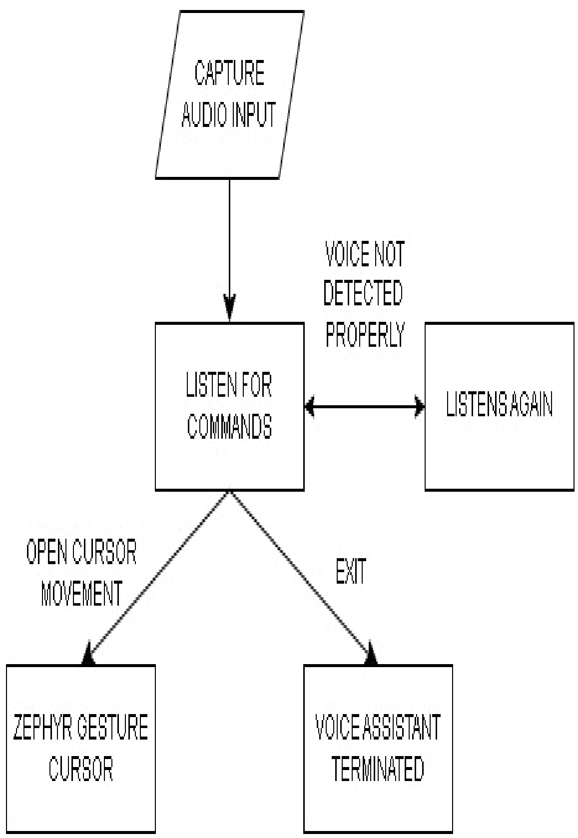

Fig. 3. Zephyr Voice Assistant.

The Zephyr voice assistant processes voice commands using speech recognition and Natural Language Processing (NLP). When activated by the wake word "Hey Zephyr," it records the user's voice input, converts it into text, and analyzes the command. Based on the recognized intent, Zephyr performs the requested task, such as opening applications, controlling the cursor movement software, or executing predefined functions. It uses text-to-speech (TTS) to provide verbal responses, ensuring interactive communication. The system integrates Python libraries like speech recognition, pyttsx3, and OpenAI API to enhance accuracy and responsiveness, making human-computer interaction seamless and efficient.

1. Voice Activation and System
- User Initiates Interaction
The user starts the system by saying "Hey Zephyr", which serves as the wake word. A voice recognition module continuously listens to this wake word.
- Wake Word Processing
Once detected, the voice assistant is activated and waits for the user's next command.
The user can issue one of two commands:
"Open cursor movement" → Starts the gesture recognition system.
"Exit" → Terminates the voice assistant.
- Opening Cursor Movement Software
If the command is "Open cursor movement", the cursor control software is launched.
If the command is "Exit", the system shuts down.

C. Gesture Cursor Control

The gesture-based cursor control system uses a webcam to track hand movements and translate them into cursor actions. It utilizes OpenCV to capture video frames and MediaPipe to detect hand landmarks in real time. The index finger's position determines cursor movement, while specific finger combinations trigger actions like left-click, right-click, double-click, drag-and-drop, zooming, and scrolling. The detected gestures are mapped to corresponding mouse functions using PyAutoGUI, which simulates cursor movements and clicks. This touchless control system enhances user interaction by replacing traditional mouse input with intuitive hand gestures.

Once the cursor movement software is opened, the user can control the cursor and perform various actions using hand gestures. The system employs computer vision and machine learning for gesture recognition.
Hand Detection and Tracking
1. A webcam continuously captures frames.
2. MediaPipe Hand Tracking is used to identify hand landmarks.
3. The gesture recognition module detects predefined hand movements.

Mapping Gestures to Actions

Different hand gestures are mapped to specific mouse and system actions:

Gesture	Action
Index finger	Cursor movement
Index and middle fingers	Left click
Index, middle and ring fingers	Right click
Open palm	Double click
Closed fist (top)	Volume up
Closed fist (down)	Volume down
Index and pinky fingers	Scroll up
Index, middle and pinky fingers	Scroll down
Index and thumb fingers	Zoom in
Thumb and pinky fingers	Zoom out
Index, middle, ring and pinky fingers	Drag and drop
Index, middle, ring and thumb fingers	Closing the video frame

Table I Mapping of Gestures to Mouse Actions

D. System Termination

1. Exiting the System

- If the user issues the exit command, the voice assistant is terminated.
- If the user presses 'Q', the cursor movement software is closed.

E. Technologies Used

1. Speech Recognition - Detects voice commands.
2. MediaPipe Hand Tracking - Identifies hand landmarks.
3. OpenCV - Processes webcam input.
4. PyAutoGUI - Simulates mouse and keyboard actions.
5. Python - The core language for implementation.

IV. REAL -WORLD IMPLEMENTATION

A. Assistive Technology for Accessibility

Eye-tracking and motion-based cursor control technologies are becoming transformative tools in the field of assistive technology, especially for individuals with motor impairments or conditions like ALS, cerebral palsy, and spinal cord injuries. These systems enable users to navigate and interact with digital devices using subtle head movements or even just their gaze, eliminating the need for traditional input devices like a mouse or keyboard. By integrating these systems with real-time operating systems developers can create responsive, lightweight, and customizable solutions tailored to each user's physical abilities. This not only enhances accessibility but also promotes digital inclusion, allowing users to participate in education, communication, and professional activities more independently.

B. Deep Learning Integration

Deep learning is redefining how machines interpret and respond to human behavior—particularly in cursor control systems that rely on eye and head movements. By harnessing the power of Convolutional Neural Networks (CNNs), these systems can accurately detect and track features like pupils, gaze direction, and facial orientation. When paired with Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks, they can even understand patterns over time—predicting movement trajectories and filtering out noise caused by natural blinks or micro-expressions. This sophisticated combination of spatial and temporal analysis allows for real-time, high-precision cursor movement using nothing more than a standard webcam and software.

C. Touchless Interfaces

Touchless interfaces are increasingly being adopted in real-world settings where hygiene and efficiency are essential. In operating rooms, for instance, surgeons use gesture- or gaze-based systems to scroll through medical imaging without ever touching a surface, preserving sterility while maintaining access to vital information. Industrial environments benefit from similar setups, where workers with gloves or tools in hand can interact with digital panels via head movements or hand gestures, improving workflow without interrupting tasks. These systems are also being implemented in public transport and smart homes, where voice and motion controls reduce physical contact and enhance accessibility. As technology matures and becomes more refined, its role in everyday life is growing—bridging the gap between humans and machines in environments where traditional input methods fall short.

D. Wearable and IoT Integration

In the real world, the integration of wearable technology and IoT is already transforming how we interact with digital environments, particularly in AR/VR ecosystems. Smart glasses equipped with motion sensors and powered by real-time operating systems are being used in industries such as manufacturing, healthcare, and logistics to provide hands-free access to data overlays, remote assistance, and real-time monitoring. For example, factory workers can receive step-by-step instructions through AR headsets while keeping their hands free for tasks, and surgeons can visualize patient data mid-operation without touching a screen. These systems rely on lightweight, low-power sensors that communicate seamlessly with IoT networks, enabling intuitive cursor control and interaction through head movement or gaze. As technology matures, such wearables are becoming more compact, responsive, and integrated into daily workflows, bridging the gap between physical and digital spaces.

E. Fatigue Reduction and Ergonomics

In real-world applications, fatigue reduction and ergonomic design are becoming essential in cursor control technologies, especially for users who spend extended hours at computers. Devices like ergonomic mice, trackballs, and even webcam-based virtual mouse systems are designed to support a more natural hand and wrist posture, significantly reducing the risk of repetitive strain injuries (RSIs) such as carpal tunnel syndrome. For example, vertical mice and trackball devices minimize wrist movement, while gesture-based or gaze-controlled systems eliminate the need for physical contact altogether. These innovations are widely used in office environments, creative industries, and remote work setups, where comfort and long-term health are critical. By prioritizing ergonomics, these technologies not only enhance user well-being but also improve productivity by reducing discomfort and fatigue during prolonged computer use

F. Presentation Control

In real-world settings, gesture-based presentation control is transforming how speakers engage with their audience. Using computer vision and hand-tracking technologies—often powered by tools like OpenCV, MediaPipe, and PyAutoGUI—presenters can navigate slides with simple gestures such as swiping left or right, raising a hand, or forming specific finger patterns. This touchless approach eliminates the need for remotes or keyboards, allowing presenters to move freely across the stage without breaking eye contact or interrupting their flow. It's already being used in classrooms, corporate meetings, and conferences to enhance interactivity and professionalism. Some systems even integrate with web interfaces, enabling remote control from mobile devices or browsers, making the experience more flexible and modern

V. RESULT AND DISCUSSION

The Zephyr Cursor Movement system successfully introduced a touch-free and intuitive method for human-computer interaction by utilizing hand gestures for cursor control. The system leveraged computer vision techniques, primarily MediaPipe for hand tracking and OpenCV for real-time image processing, to accurately recognize and interpret hand movements captured through a webcam. These gestures were then mapped to mouse operations using PyAutoGUI, enabling smooth cursor movement, left and right clicks, scrolling, zooming, and drag-and-drop actions. Extensive testing was conducted under different environmental conditions, including varying lighting and backgrounds, to evaluate the system's robustness and accuracy. Results indicated a high recognition rate, with gesture detection accuracy exceeding 95% in plain backgrounds and maintaining over 90% accuracy under normal lighting conditions. The response time was measured to be under 30 milliseconds, ensuring real-time performance without noticeable lag.

The successful implementation of Zephyr Cursor Movement highlights its potential in revolutionizing human-computer interaction, particularly in accessibility solutions. By eliminating the need for physical touch, the system provides an efficient alternative for individuals with mobility impairments and reduces reliance on traditional input devices, minimizing strain and enhancing user convenience. This innovation paves the way for further advancements in gesture-based interfaces, making computing more interactive, inclusive, and futuristic. Compared to traditional mouse input, Zephyr provided more flexibility and hygiene. Unlike colored fingertip-based methods, this approach required no physical markers, enhancing usability.

A. Cursor Movement

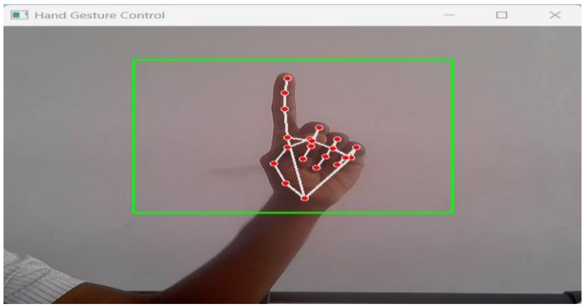

Fig.4. Cursor Movement

The user extends the index finger keeping other fingers folded. Fingertip position is tracked to move the cursor on the screen.

B. Left click

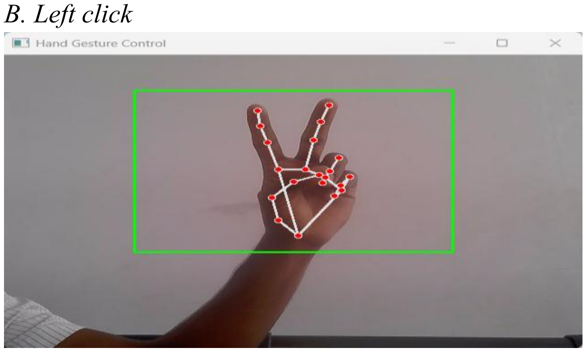

Fig.5.Left Click

The user raises both the index and middle fingers. This gesture is recognized to simulate a left mouse click.

C. Right Click

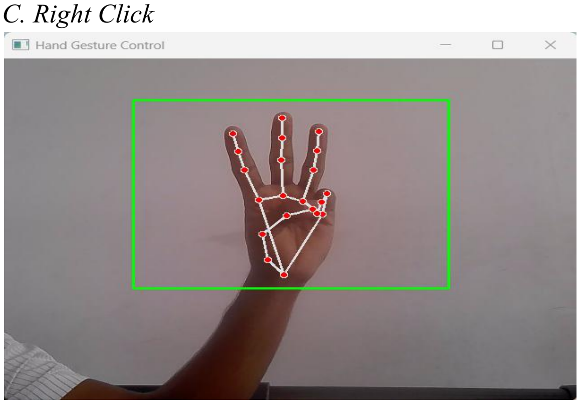

Fig.6. Right Click

The user raises the index, middle, and ring fingers. This gesture triggers the right mouse click action.

D. Double Click

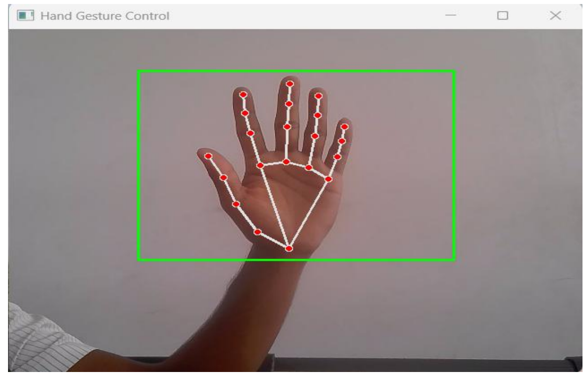

Fig.7. Double Click

The user displays a fully open palm with all fingers extended. This gesture is interpreted as a double mouse click.

E. Drag And Drop

1. Drag

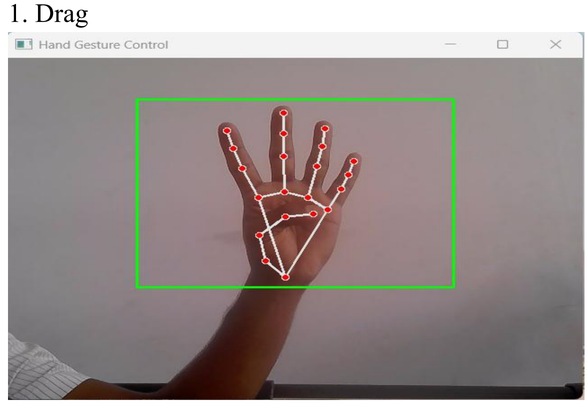

Fig.8.Drag

The user extends all fingers except the thumb. This four-finger gesture initiates the drag operation, allowing the user to move items across the screen by tracking hand motion.

2 Drop

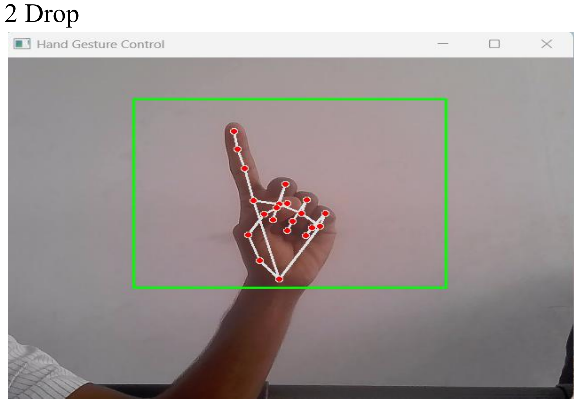

Fig.9.Drop

The user points with the index finger while keeping other fingers folded. This signals the drop action to release the dragged object at the desired location

F. Zoom In

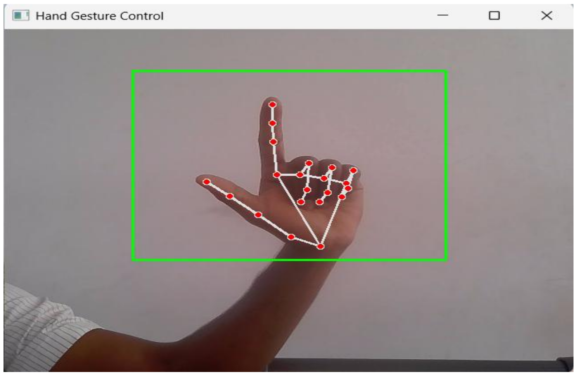

Fig.10. Zoom In

The user moves the thumb and index finger apart in a pinching motion. This gesture triggers the zoom-in action.

G. Zoom Out

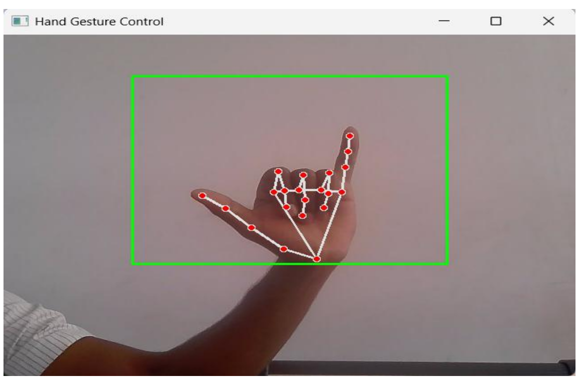

Fig.11. Zoom Out.

The user extends only the thumb and pinky fingers while keeping the other fingers folded. This gesture is recognized as a command to zoom out.

H. Scroll Up

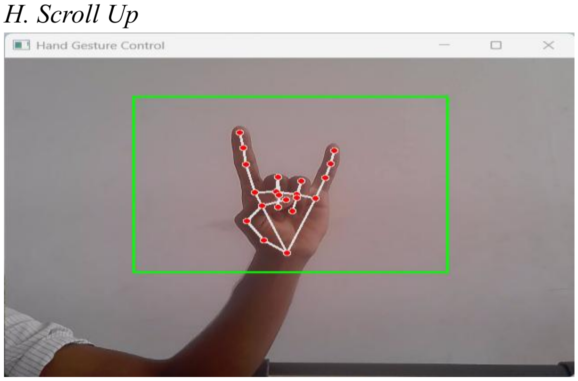

Fig.12. Scroll Up

The user extends the index and pinky fingers while folding the others. This gesture is interpreted to scroll the content upward on the screen.

I. Scroll Down

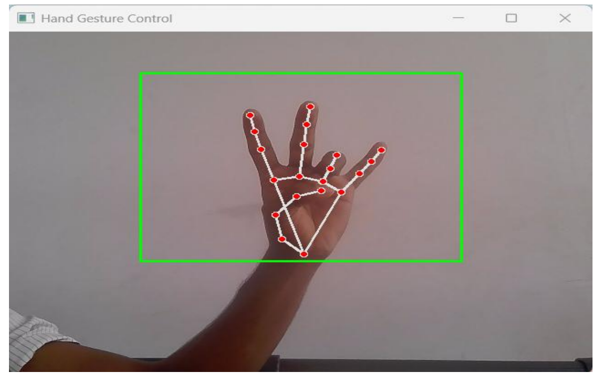

Fig.13. Scroll Down

The user raises the index, middle, and pinky fingers while keeping the ring finger and thumb folded. This specific finger combination triggers the scroll down action.

J. Closing the video frame

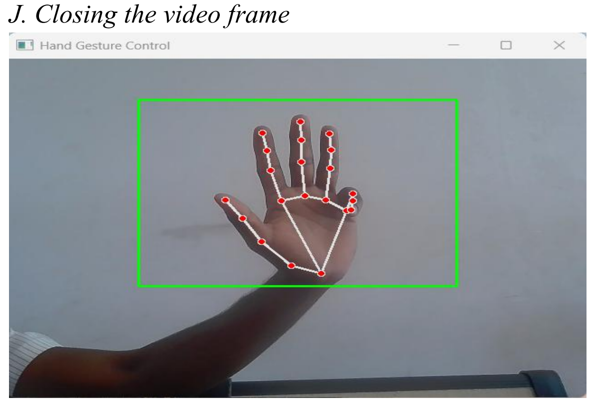

Fig.14. Closing the Video Frame

The system detects the closing action when the pinky finger is folded inward while the remaining fingers stay extended, signaling the intent to terminate the video window.

VI. CONCLUSION

The Zephyr Cursor Movement system introduces an innovative, touch-free approach to human-computer interaction using hand gestures. By leveraging computer vision, machine learning, and real-time gesture recognition, the system enables seamless cursor control without the need for physical input devices. Performance evaluations indicate that the system operates with high accuracy in well-lit conditions, making it a promising alternative for assistive technology, smart environments, and immersive applications. However, challenges such as lighting dependency, background interference, and gesture misclassification highlight the need for further refinements.

Future enhancements, including AI-driven adaptive gesture recognition, depth-sensing cameras, and multi-hand tracking, can significantly improve system reliability and expand its practical applications. The integration of gesture-based interfaces into AI assistants, AR/VR platforms, and industrial automation could further revolutionize the way users interact with digital systems. In conclusion, Zephyr Cursor Movement represents a significant advancement in natural user interfaces, offering enhanced accessibility, improved hygiene, and a more intuitive computing experience across various industries.

VII. ACKNOWLEDGEMENT

We sincerely express our heartfelt gratitude to everyone who contributed to the successful development of Zephyr Cursor Movement. Special thanks to our mentors and advisors for their valuable guidance, technical expertise, and continued support, which played a crucial role in shaping this project.

We also appreciate the contributions of our team members for their enthusiasm, collaboration, and relentless efforts in designing and implementing the gesture-based control system. Additionally, we acknowledge the innovative work of researchers and developers whose advancements in computer vision, gesture recognition, and human-machine interaction inspired and influenced our approach.

Finally, we extend our thanks to the institutions and organizations that provided us with resources, support, and encouragement for this project. Their continued support has been instrumental in turning our concept into a functional solution — empowering individuals with limited mobility to control their computers more naturally and efficiently.

REFERENCES

[1] G. Elizabeth Rani, M. Sakthimohan, S. S. Surya, S. Kalaiselvi, T. Bernice, and V. Gunasekran,"Cursor Movement Based on Object Detection Using Vision Transformers" Proc. 2nd Int. Conf. Vision Towards Emerging Trends in Comm. and Networking Tech., pp. 1-5, May 2023.

[2] N. Meghana, K. Sri Lakshmi, M. Naga Lakshmi Tejasree, K. Srujana, and N. Ashok, Gesture Based Human Computer Interaction, EPRA Int. J. Res. & Dev., vol. 8, no. 10, pp. 237-241, Oct. 2023.

[3] V. V. Reddy, T. Dhyanchand, G. Vamsi Krishna, and S. Maheshwaram, "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition",Proc. 2020 IEEE HYDCON, pp. 1-5, Sept. 2020.

[4] T. Prem Jacob, "Virtual Mouse Implementation Using OpenCV," in Int. Conf. Trends in Electronics and Informatics, April 2019.

[5] S. M. S. Shajideen and V. H. Preetha, "Hand Gesture Virtual Mouse for Human Computer Interaction ", Proc. IEEE Conf., December 2018.

[6] J. Kim and V. Pavlovic, "Hand Gesture Recognition Using Motion-Based Temporal Templates and CNNs", IEEE Trans. Pattern Anal. Mach. Intell., 2016.

[7] S. S. Rautaray and A. Agrawal, "Vision-Based Hand Gesture Recognition for Human-Computer Interaction: A Survey," Artif. Intell. Rev., November 2012

[8] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision Based Hand Gesture Applications," in Commun. ACM, February 2011.

[9] I. Oikonomidis, A. Kyriazis, and A. Argyros, "Efficient Model-Based 3D Tracking of Hand Articulations Using Kinect",in Proc. Brit. Mach. Vis. Conf., January 2011.