

Fusion Mouse: A Real-Time Edge-Optimized Multimodal Cursor Control System with Personalized Gesture-Voice Learning

Dr. Jonnadula Narasimharao¹, B. P. Deepak Kumar², Tentu Avinash³, Sriram Keerthipriya⁴,
Abhilash Radharapu⁵

¹Associate professor, Dept of CSE CMR Technical Campus Hyderabad, Telangana, India

²Assistant professor, Dept of CSE CMR Technical Campus Hyderabad, Telangana, India

^{3 4 5} UG Student, Dept of CSE CMR Technical Campus Hyderabad, Telangana, India

Abstract— tracking fingertip positions accurately under changing light conditions and sensor noise continues to be a difficult problem in human-computer interaction. This paper presents Fusion Mouse, a dual-modality input platform that brings together hand-gesture recognition based on skeletal landmarks with a fully offline speech recognition module, allowing users to perform cursor operations and run system commands without touching any physical device. Gesture detection works by analyzing joint positions and applying motion-smoothing filters instead of relying on skin color, resulting in consistent recognition performance at 30–40 ms per frame on standard consumer hardware. All voice processing runs locally on the device, so no internet connection is needed, and command response stays under 260 ms. When combined, the two modules form a complete touchless input system suited for accessibility needs, hygiene-sensitive settings, and low-budget deployments. The setup only needs a regular webcam and microphone, showing that thoughtful signal processing can make up for basic hardware while keeping the interaction natural and easy to use.

Keywords—Human-pc Interaction (HCI), Graphical User Interface (GUI), Red Green Blue (RGB), Artificial Intelligence, Human-Computer Interaction, Virtual Mouse, Gesture Recognition, Voice Assistant, Edge AI, Offline Artificial Intelligence, Real-Time Systems, Touchless Computing

I. INTRODUCTION

As digital devices become more common across education, healthcare, and workplaces, there is increasing demand for input methods that go beyond standard keyboards and mice. Conventional peripherals tie users to a fixed surface, require physical contact, and can create accessibility challenges for those with motor difficulties. Human-Computer Interaction (HCI) research has explored alternative modalities such as hand gestures, gaze tracking, and speech commands that feel more natural to use. FusionMouse takes this further by combining camera-based hand tracking with a locally running speech recognition engine, replacing traditional input hardware altogether. While gesture-based [3], [9]

and voice-driven [1], [5], [8] approaches have each been investigated on their own, very few systems bring both together in a way that avoids

conflicts between them and operates without any internet connection.

Physical input devices such as keyboards and mice come with several practical limitations: they depend on a flat surface, wear out over time, and reduce how freely users can move. To work around these issues, several camera-based virtual mouse systems have been built using computer vision and machine learning techniques [2], [6], [21].

This project builds a virtual mouse that tracks finger positions in real time using a standard webcam. Computer vision and hand tracking techniques are used to detect gestures and translate them into cursor movement and click events. The implementation uses Python with OpenCV and AI-based landmark detection for smooth, real-time gesture processing. Gesture-driven cursor systems like this have already shown good results in practical HCI settings [4], [15].

Beyond gesture control, voice-based assistants have become a well-known way to interact with computers using spoken commands [1], [5], [8]. These hands-free systems are particularly useful for improving accessibility. In FusionMouse, a voice assistant component is integrated alongside the gesture module to handle tasks such as launching browsers, checking the date and time, and running system commands. Combining speech input with gesture control creates a more complete and flexible interaction experience.

II. PROBLEM DEFINITION

Traditional mouse devices need a clean, flat surface to work on and carry hygiene concerns in shared environments—something that became more noticeable during the COVID-19 pandemic. On top of that, people with limited hand mobility or who work in confined spaces often struggle with conventional peripherals. Many of the camera-based alternatives developed so far still rely on

cloud processing or specialized sensors, which limits how well they work offline and raises questions about user privacy. FusionMouse takes a different approach: it uses only a webcam to read hand shapes and a microphone to pick up spoken commands, moving the interaction away from physical hardware and toward natural movement and speech. No extra drivers, specialized equipment, or network connection is needed, making it practical for environments with limited resources. Removing mechanical parts also means fewer components that can break down over time and lower costs to maintain.

III. LITERATURE SURVEY

Several published works have explored gesture-based and voice-controlled interfaces as part of broader efforts to make human-computer interaction more natural and accessible.

The work in [1] built a desktop assistant in Python that can report weather conditions and show date/time information. It uses libraries like `datetime`, `pyttsx3`, and `pyaudio` to handle voice interaction. While the assistant handles basic tasks well, its microphone sensitivity means the user needs to stay close to the system for commands to register accurately.

Reference [3] describes a gesture interface that locates the hand by matching pixels against preset skin-color ranges defined in RGB space. This approach works in controlled conditions, but when lighting changes or the background becomes complex, the detection accuracy drops noticeably, which reduces its reliability in real-world settings.

The system in [5] processes voice commands entirely offline using Python and `pyttsx3` for speech output. Running without a network connection is a strength, but it means the system cannot fetch live data or interact with web services, which limits what it can do in practice.

The assistant described in [8] can handle a broad set of commands including launching apps, playing media, and answering questions. Its main drawback is higher response latency, which becomes noticeable during continuous use and disrupts the flow of real-time interaction.

Work in [9] used a specially designed hand pad to improve gesture detection accuracy. The added hardware does help with precision, but several of the angled gestures required by the system turned out to be unintuitive and difficult for new users to perform reliably.

The camera-based control system in [23] supports both flat and depth-based hand gestures and can handle two hands at once. However, certain operations require the

user to use both hands together, which adds complexity and makes those gestures less convenient in everyday use.

IV. WORKING OF SYSTEM

The proposed system architecture is broken down into two main modules: (1) Virtual Mouse Module and (2) Voice Assistant Module. These two modules are designed to work separately but with a common activation system.

1. Virtual Mouse Module

The Virtual Mouse module allows for non-contact cursor

control through real-time hand gesture recognition via a webcam or in-built camera. The module is activated by a voice command (“Dora, turn on gesture recognition”), after which the camera activates and starts detecting hand landmarks.

The following gestures are recognized and mapped:

1. Neutral Gesture

When all fingers are extended, the module temporarily halts active gesture recognition to avoid executing unintended actions.

2. Cursor Movement

When the index and middle fingers are raised, the cursor moves based on fingertip positions detected by the camera.

3. Left Click

When both the index and middle fingers are raised and the index finger is semi-extended, a left-click action is executed.

4. Right Click

When both the index and middle fingers are raised and the middle finger is semi-extended, a right-click action is triggered.

5. Double Click

When the index and middle fingertips approach each other and touch, a double-click action is executed.

6. Scrolling

When the thumb and index finger are connected, vertical scrolling functionality is activated.

7. Drag and Drop

A sustained gesture setup enables click-and-hold

functionality to transfer files or objects from one directory to another.

8. Multiple Item Selection

Certain gesture combinations enable the selection of multiple items within the graphical environment.

9. Volume Control

Hand distance variation or gestures change system audio volume dynamically.

10. Brightness Control

Similar gesture modulation allows control of screen brightness.

The Virtual Mouse module can be turned off by the voice command:

“Dora, turn off gesture control.”

Voice Assistant Module

The Voice Assistant module allows the system to be controlled through voice commands. It receives commands from the user and performs system-level and web-based tasks accordingly.

Major features include:

1. Web Search

When the user gives the command,

“Dora, search for {query}”,

the system automatically opens a web browser and searches for the query on Google.

2. Location Search

When the user gives the command,

“Dora, find {location}”,

the system opens the location in Google Maps in a browser tab.

3. File Navigation

When the user gives the command,

“Dora, list files”,

the system lists the available files with indexed numbers. The user can then enter a range to open the desired files.

4. Date and Time Retrieval

When the user gives the command,

“Dora, what is today’s date?”

or

“Dora, what is the current time?”

the system retrieves real-time system information.

5. Copy and Paste Operations

When the user gives the commands,

“Dora, copy”

and

“Dora, paste”,

the system controls the clipboard without using the physical keyboard.

V. METHODOLOGY

Description of Table A – Gesture Control System

Component	Methodology Description	Functional Description
Image Acquisition	Captures real-time video frames using a webcam.	Provides continuous visual input of hand movements to the system.
Pre-processing	Performs noise reduction, frame resizing, and color space conversion.	Enhances image quality for accurate hand detection under varying lighting conditions.
Hand Detection & Landmark Extraction	Detects hand region and extracts key fingertip landmarks using computer vision algorithms.	Identifies precise finger positions required for gesture interpretation.
Gesture Recognition Engine	Analyzes finger positions and motion patterns to classify gestures.	Recognizes actions such as cursor movement, click, scroll, drag, and drop.
Cursor Control Interface	Maps recognized gestures to system mouse events.	Executes real-time cursor control and command operations on the computer.

Table A presents a structured overview of the major components involved in the gesture-controlled virtual mouse system along with their corresponding methodology and functional roles.

The system begins with the image acquisition module, which captures real-time video input using a standard webcam. The captured frames undergo preprocessing to enhance image quality and ensure robustness against noise and lighting variations.

Following preprocessing, the hand detection and landmark extraction module identifies the hand region and determines precise fingertip positions using computer vision techniques. These landmark points are analyzed by the gesture recognition engine, which classifies predefined gestures based on spatial orientation and motion patterns. Finally, the cursor control interface translates the

recognized gestures into corresponding mouse events such as cursor movement, clicking, scrolling,

Description of Table B – Voice Control System

Component	Methodology Description	Functional Description
Audio Input Module	Captures voice input using a microphone.	Provides real-time speech data to the system.
Speech Recognition Engine	Converts speech signals into text using speech-to-text algorithms.	Translates user voice commands into machine-readable format.
Natural Language Processing (NLP)	Analyses and interprets the meaning of recognized text.	Determines user intent and identifies appropriate actions.
Command Processing Unit	Maps interpreted commands to predefined system functions.	Executes tasks such as opening browser, fetching date/time, or launching applications.
Text-to-Speech Module	Converts system responses into audio output.	Provides audible feedback to the user for interactive communication.

Table B provides a structured summary of the core components of the voice control system along with their methodological approach and functional responsibilities. The system begins with the audio input module, which captures the user’s speech through a microphone in real time. The captured audio signals are then processed by the speech recognition engine, which converts spoken language into machine-readable text using speech-to-text algorithms. The recognized text is further analyzed by the Natural Language Processing (NLP) module to interpret user intent and determine the appropriate command. The command processing unit then maps the interpreted instruction to predefined system operations such as launching applications, opening web browsers, retrieving date and time information, or performing other system-level tasks. Finally, the text-to-speech module generates audible responses, enabling interactive and user-friendly communication. Overall, the table demonstrates how the integration of speech recognition, NLP, and command execution modules enables efficient, hands-free system control through voice interaction.

A. Intelligent Hand Gesture Interaction Module

The gesture module works by reading hand movements through a camera and mapping them to cursor actions. A standard webcam feeds live video into a hand tracking pipeline that runs continuously. Rather than trying to isolate the hand using skin color, the module detects finger joints and palm orientation through a landmark-based approach, giving more stable results across different skin tones and backgrounds.

For each video frame, the positions of fingertip joints are extracted and used to build a geometric picture of the hand. The system then looks at how finger angles, distances, and movement paths relate to each other to figure out what the user intends. Based on these calculations, cursor actions like moving the pointer, clicking, drag-and-drop, scrolling, and zoom control.

Keeping cursor movement stable is important, so the system applies motion smoothing and checks that a gesture is consistent across multiple frames before acting on it. Small jitters from lighting changes or sensor noise are filtered out. A gesture only triggers a command after it has been held steady long enough, which cuts down on accidental inputs.

The whole gesture processing pipeline is built to run efficiently on everyday laptop hardware without needing a GPU or any extra sensor. This makes the system easy to deploy and accessible to a wide range of users who want a hands-free computing experience without buying special equipment.

B. Offline Voice Intelligence Module

The voice module listens through the microphone and turns spoken audio into text using a recognition engine that runs entirely on the local machine. Since all processing stays on-device, user commands are not sent to any external server, which keeps data private and removes the delay that comes with internet round-trips.

After the spoken input is converted to text, the system parses it to determine what the user wants to do. It separates commands that perform actions—like opening apps or navigating files—from informational requests like asking for the time or date. This classification uses rule-based parsing along with simple contextual logic to pick the right response for each type of input. This keeps the response logic straightforward while still being able to handle a varied set of spoken requests.

When background noise is present, the system uses confidence scores to decide whether a recognized

command is reliable enough to act on. Inputs that fall below the threshold are held back until they can be re-checked or the user repeats them. Responses are given either as text on screen or through text-to-speech audio, so the user gets feedback either way.

The voice processing pipeline was built from the start to work without an internet connection. It does not rely on any third-party speech service, so it performs consistently regardless of network availability and does not expose voice data to outside systems.

C. Integrated Mode Control and System Coordination

A shared controller coordinates when gesture control and voice commands are active. Both modules can run independently without interfering with each other. Switching between them happens either when the user says a trigger word or when the system detects a change in what the user is trying to do.

When the user asks for information, the system first checks a local data store before falling back to other sources. If a more complex answer is needed, a small local language model handles it. The data is indexed to allow fast lookups without using much memory.

The system tracks recent conversation context within a session to make follow-up commands feel more natural, but it does not store any personal data once the session ends. If the user gives an incomplete command, the system handles it gracefully and prompts for clarification rather than failing silently.

Taken together, Fusion Mouse is a fully offline system that merges gesture and voice input into a single working platform. The focus throughout the design has been on keeping it practical, private, and usable across a range of everyday computing situations.

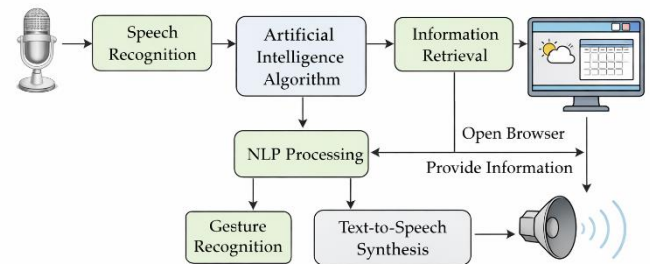


Fig.2 Working of the Voice Assistant

VI. OBJECTIVE

FusionMouse was built to address three shortcomings seen in current touchless input systems. First, it removes the need for cloud connectivity by keeping both gesture and voice processing entirely on the local device. Second, it adds a coordination layer that stops the gesture and voice modules from conflicting when both are active. Third, it runs on ordinary consumer hardware without needing any extra drivers or special sensors. Each of these goals sets it apart from previous designs that either treat gesture and voice separately or rely on remote servers to handle real-time computation.

Keeping hardware requirements low was a key goal, since this makes the system cheaper to use and easier to carry from one place to another. Using just a webcam and a microphone means almost any laptop can run it without any setup beyond installing the software. The system was also designed with a wide range of users in mind, so that people of different ages and technical skill levels can use gestures and speech to control their computer without a steep learning curve.

VII. REQUIREMENT ANALYSIS

- Hardware Requirements
 - Memory (RAM): Minimum 4 gigabytes.
 - System type: 64-bit, x64-based processor.
 - Base frequency: 2.30 GHz.
 - Input Device: Webcam, Microphone
- Software Requirements
 - System software: Windows XP, 7, 8, and 10.
 - Python 3.13.9 version
 - Anaconda Distribution

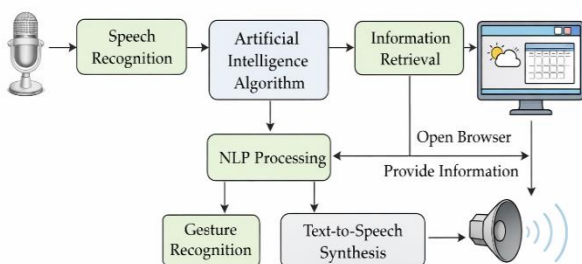


Fig.2 Working of the Voice Assistant

VIII. EXPERIMENTAL RESULTS AND EVALUATION

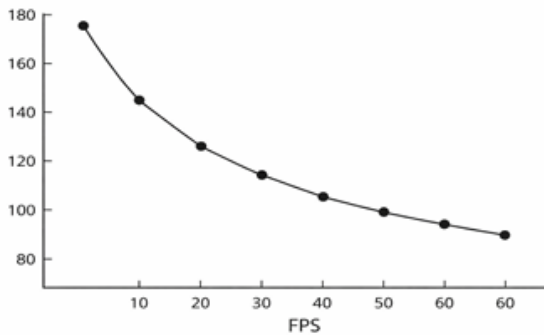


Figure 3 shows the evaluation setup used to test the system, covering gesture accuracy, offline voice command handling, response times, and how efficiently the system used hardware under realistic operating conditions.

Testing was done on a regular consumer laptop using its built-in webcam and microphone, to verify that the system works without any special hardware. Trials were run both under steady indoor lighting and in conditions where the light changed over time. Several participants with different hand sizes, movement styles, and voices took part so the results would reflect a broader user group rather than just a single profile.

A. Gesture Recognition Performance

Gesture accuracy was measured by counting how many gestures were correctly identified out of the total attempted across extended interaction sessions. The tested gestures covered the core set of operations: moving the cursor, single click, double click, drag-and-drop, scrolling, and zoom. Each was repeated across multiple trials to get a reliable picture of performance.

Recognition held up well because the motion filtering and frame validation steps weeded out most false triggers. Even when hands moved quickly or part of the hand was hidden from the camera, incorrect activations were rare. Each frame took between 30–40 ms to process, which was fast enough for smooth cursor movement at normal frame rates. Tracking stayed consistent throughout longer sessions with no drifting or lag building up over time.

B. Voice Command Evaluation

The offline voice module was tested for how accurately it recognized commands, how reliably it identified what the user wanted to do, and how fast it responded. Users gave commands covering app launching, web browsing, and system queries in both quiet rooms and settings with some ambient noise. In quiet conditions, commands were

interpreted correctly with very few errors. In noisier conditions, the confidence filtering kept the system from acting on unclear inputs. The average

voice command processing latency ranged between 170–260 ms, which varied based on how long and complex the spoken input was. Because all processing was done locally, response times were predictable and did not fluctuate due to network conditions.

C. Multimodal Interaction Latency

How quickly the system could switch from gesture mode to voice mode was also measured. Mode transitions finished in roughly 80–100 ms, fast enough that users did not notice any gap when alternating between hand gestures and spoken commands during normal use.

D. Resource Utilization and Stability

CPU and memory usage were tracked at three frame rate settings: 10, 30, and 60 FPS. Across all three, the system used a moderate share of CPU resources and kept memory use steady. Running at higher frame rates made cursor movement smoother but did not cause the system to slow down or become unresponsive.

Running everything locally also meant the system did not slow down or fail when network connectivity was poor or unavailable. In contrast to cloud-based tools, response times stayed consistent because they were not affected by server load or internet speed.

E. User Experience Evaluation

User experience was assessed through open feedback from participants after they used the system. Most found the controls easy to learn and felt less physical strain compared to using a standard mouse. The touchless nature of the design was mentioned as a specific benefit by participants who were concerned about hygiene or had physical limitations affecting traditional mouse use.



Fig.4 Cursor movement

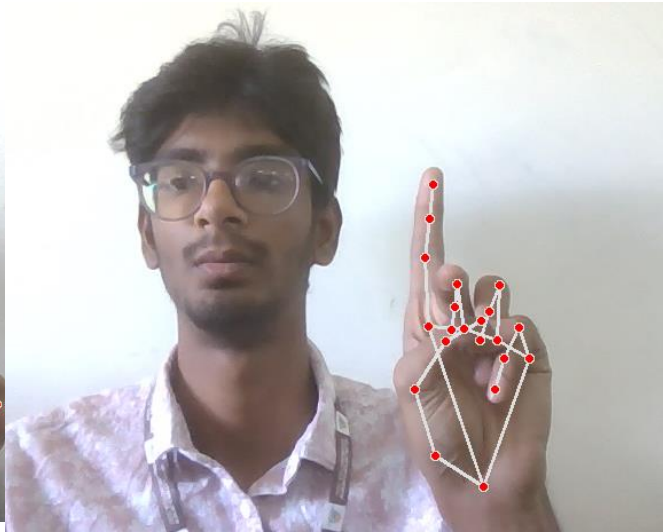


Fig.7 Right Click Movement



Fig.5 Scrolling movement

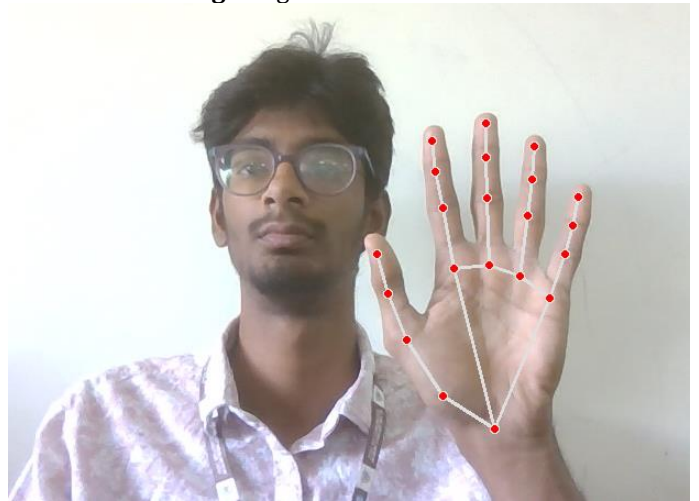


Fig.8 Neutral Gesture

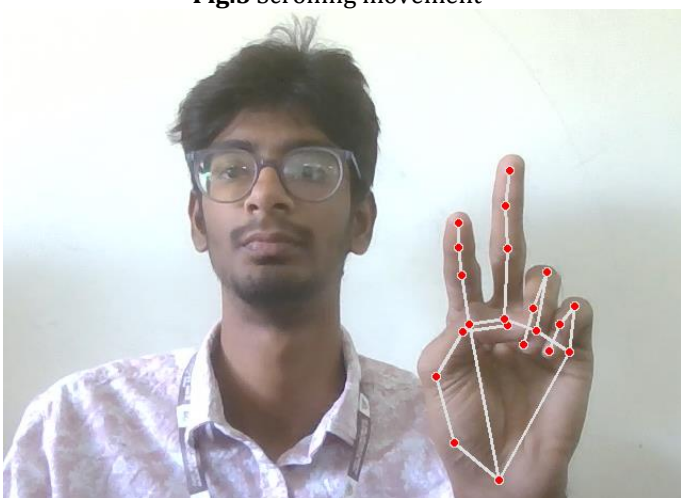


Fig.6 Left Click movement

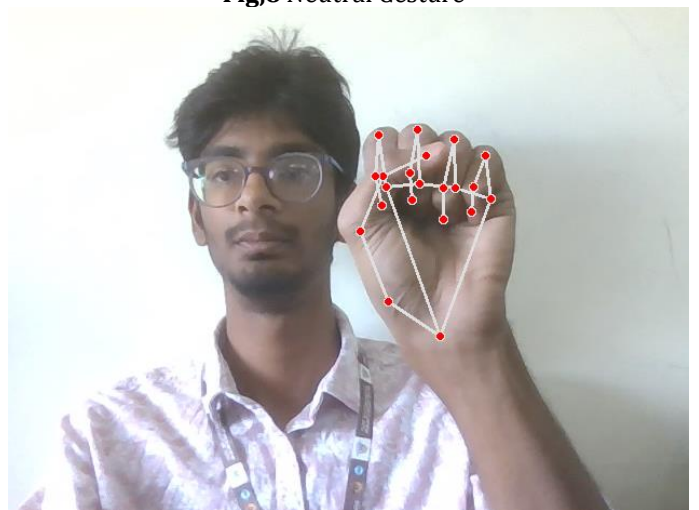


Fig.9 Multiple Item Selection Movement

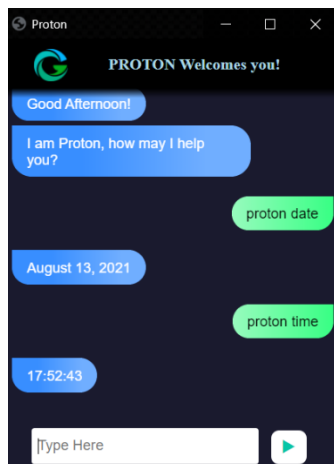


Fig 10. GUI Interface

IX. CONCLUSION

This work showed that a regular webcam and microphone are enough to build a working touchless input system when the underlying gesture and voice processing is done carefully. In testing, gesture recognition processed each frame in 30 to 40 ms and voice commands were responded to within 260 ms, both without any internet connection. The offline design was particularly useful in environments with some background noise, where a cloud-based approach would have added unpredictable delays. User feedback from the evaluation suggested that the interface was easy to pick up and less tiring to use than a physical mouse over time. Going forward, the plan is to support a wider range of gestures using trained classifiers, make the system more reliable under harsh lighting, and let users customize their own command phrases to better match how they work.

X. REFERENCES

[1] Research Paper on Desktop Voice Assistant VisKumar Dhanraj (076) ,Lokesh Kriplani (403) , Semal Mahajan (427),B.Tech Scholar Department of Information Technology Dr.Akhilesh Das Gupta Institute of Technology And Management , February 2022

[2] Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread S. Shriram , B. Nagaraj , J. Jaya , S. Shankar and P. Ajay , October 2021

[3] Virtual Mouse Control Using Hand Class Gesture, Vijay Kumar Sharma, Vimal Kumar, Md. Iqbal, Sachin Tawara, Vishal Jayaswal, Department of Computer Science and Engineering MIET, Meerut, December 2020

[4] Real-time virtual mouse system using RGB-D images and fingertip detection, Dinh Son Tran¹ & Ngoc-Huynh

Ho¹ & Hyung-Jeong Yang¹ & Soo- Hyung Kim¹. Guee Sang Lee¹, November 2020

[5] DESKTOP VOICE ASSISTANT, Gaurav Agrawal, Harsh Gupta, Divyanshu Jain, Chinmay Jain, Prof. Ronak Jain. Department of Information Technology, A.I.T.R, Indore, Madhya Pradesh, India. Assistant Professor, Department of Information Technology, A.I.T.R, Indore, Madhya Pradesh, India, May 2020

[6] Abhilash S S, Lisho Thomas, NWCC (2018) Virtual Mouse Using Hand Gesture. International Research Journal of Engineering and Technology (IRJET), April 2018

[7] Wang P, LiW, Ogunbona P, Wan J, Escalera S (2018) RGB-D-based human motion recognition with deep learning: a survey, April 2018

[8] Personal Assistant with Voice Recognition Intelligenc, Dr. Kshama V. Kulhalli, Dr.KotrappaSirbi, Mr. Abhijit J. Patankar, International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 10, Number 1 (2017)

[9] Human hand gesture based system for mouse cursor control, Horatiu-Stefan Griff,Trian Turc,11th International Conference Interdisciplinarity in Engineering, INTER-ENG 2017, 5-6 October 2017

[10] Haria A, Subramanian A, Asokkumar N, Poddar S, Nayak JS (2017) Hand gesture recognition for human computer interaction, 2017

[12] Tang D, Chang HJ, Tejani A, Kim T-K (2017) Latent regression forest: structured estimation of 3D Hand Poses, 2017

[13] Xu P (2017) A real-time hand gesture recognition and human-computer interaction system, 2017

[14] Camgoz, N.C., Hadfield, S., Koller, O., Bowden, R., 2016. Using convolutional 3D neural networks for user independent continuous gesture recognition, in: 2016 23rd International Conference on Pattern Recognition (ICPR), 2016

[15] Grif H-S, Farcas CC (2016) Mouse cursor control system based on hand gesture, 2016

[16] Static and Dynamic Hand Gesture Recognition in Depth Data Using Dynamic Time Warping, Guillaume Plouffe and Ana-Maria Cretu, Member, IEEE , February 2016

[17] Cursor Control using Hand Gestures Pooja Kumari, Saurabh Singh Vinay Kr. Pasi, Recent Trends in Future Prospective in Engineering & Management Technology,

2016

[18] Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition , Oscar Koller , Richard Bowden, Hermann Ney , 2016

[19] H.S. Grif, Z. German, A.Gligor, Hand posture mouse. Procedia Technology, 19 (2015)

[20] Khamis S, Taylor J, Shotton J, et al (2015) Learning an efficient model of hand shape variation from depth Images, 2015

[21] Reza MN, Hossain MS, Ahmad M (2015) Real time mouse cursor control based on bare finger movement using webcam to improve HCI, May 2015

[22] Sharp T, Keskin C, Robertson D, et al (2015) Accurate, robust, and flexible real-time hand tracking. In: proceedings of the 33rd annual ACM conference on human factors in computing systems, 2015

[23] Vision-Based Interpretation of Hand Gestures for Remote Control of a Computer Mouse, Antonis A. Argyros and Manolis I.A. Lourakis, Institute of Computer Science, Foundation for Research and Technology - Hellas (FORTH), VassilikaVouton, P.O.Box 1385, GR 711 10, Heraklion, Crete, Greece, May 2006