

Touchless Web Navigation System

Arul E¹, Vishnu Prasanth B S², Suganth S³, Mithunraj P⁴, Kathirava N M⁵ and Punidha A⁶

¹⁻⁵Department of Information Technology, Coimbatore Institute of Technology, Coimbatore, Tamil Nadu, India
Email: arul.e@cit.edu.in, 71762107058@cit.edu.in, 371762107051@cit.edu.in, 471762207204@cit.edu.in, 571762107021@cit.edu.in

⁶Department of Artificial Intelligence and Data Science, KPR Institute of Engineering and Technology, Coimbatore, Tamilnadu, India
Email: punidha.a@kpriet.ac.in

Abstract—With the help of this technology, users may handle files and surf websites by speaking orders that are translated by a tracking system that recognizes their hand gestures. The operational system uses MediaPipe Hands motion tracking features in conjunction with the Google Speech Recognition API and TF-IDF-based machine learning categorization. Using vocal commands that also call for gesture motions, users may control their online browser and work with files. Through this approach, users benefit from improved usability and accessibility features as well as instant feedback about all orders that are completed. During experimental assessments, alternative speech and gesture input methods have produced more efficient user interactions.

Index Terms— Speech Recognition, Hand Gesture Control, Web Navigation, File Management.

I. INTRODUCTION

As computer usage increases, there is a corresponding need for easily available control techniques that enhance user productivity. People with physical limitations find it difficult to communicate using conventional input techniques like keyboards and mouse. To create smooth user interactions, the project creates a unified voice recognition and gesture control interface. Under this method, individuals of various abilities may operate web browsers and perform file management tasks using voice commands and hand gestures jointly. Machine learning algorithms are essential to the project's ability to effectively use speech and hand movements.

II. CHALLENGES

A. Accuracy of Speech Recognition

The likelihood of misinterpreted communication is increased by variations in speech accents, pronunciations, and background noise. Handling various speech patterns and reducing identification errors effectively are essential for smooth functioning.

B. Hand Gesture Accuracy

Making the distinction between intentional and unintentional hand movements proves to be difficult. The accuracy of identification system performance is influenced by a number of parameters, including camera angle, illumination setup, and hand positioning.

C. System Response Time

Efficient processing systems are necessary for current directives to be executed quickly. The following statement needs to be reworded to make it easier to grasp and retain its direct flow. When at all feasible, normalize verbalization.

D. Integration of Multi-Modal Inputs

The challenge exists in harmonizing speech and gesture controls so they do not interfere with each other. The system needs to accurately interpret user intent to perform the right command as users submit their inputs at the same time.

E. Context-Aware Execution

For sequential order executions that must function in a variety of operating situations, the system handles complex commands. Adaptive learning implementations will enhance context processing skills and overall operational functionality performance.

III. RELATED WORKS

A. Speech Recognition Systems

Although they lack gesture control capabilities, the current speech-activated systems, such as Google Assistant and Alexa, provide sufficient voice interactions. The system offers limited options for multi-mode interaction features since it relies on Natural Language Processing (NLP) for instruction recognition. The system is able to create a response that is more effective for hand-free use by combining the Google Speech Recognition API with machine learning command categorization.

B. Gesture-Based Navigation

Using hand-tracking technologies from MediaPipe Hands and OpenCV, the cursor and click functionalities function well. During deployment, the systems struggle to distinguish between intentional and inadvertent hand movements. To ensure precise and reliable hand gesture-controlled interactions, the system employs gesture filtering using landmark coordinate tracking of index finger bending.

C. Hybrid Approaches

Numerous research attempt to integrate gestures and voice instructions; nevertheless, synchronization issues lead to miscommunicated orders and delayed performance execution. In order to simultaneously operate gesture detection and voice recognition, the system uses multithreading technology to achieve parallel processing. The system offers automated command coordination in addition to real-time performance.

D. Context-Aware Interfaces

The prewritten command codes in the existing system restrict its capacity to carry out a variety of file management and browsing duties. Because Scikit-learn's classification models can recognize instructions dynamically thanks to TF-IDF vectorization, our approach makes it simple for users to change commands.

E. AI-Powered Assistants

Although the loop-based command structure serves as the virtual assistant's design framework, it is unable to allow customization of file and browser operations. By integrating Selenium programs for web surfing automation with operating system instructions for file management, the system enhances its current capabilities and gives users access to more advanced environmental control features than just voice commands.

IV. PROPOSED SYSTEM

By combining speech recognition, gesture control, and automation features, the Touchless Web Navigation System offers a completely hands-free file management and browsing experience [3]. In order to increase productivity and performance speed, the method functions differently from standard processes as users manage file handling and navigation using voice instructions in addition to hand gestures [6]. The Google Speech Recognition API is used by the system to precisely translate speech into text, allowing users to run commands that manage tabs, browser functionalities, and search activity [1]. The program uses MediaPipe Hands and OpenCV tracking to detect hand movements, enabling users to move their cursor and make clicks using real-time

gesture recognition. In order to browse without the usage of conventional computer hardware tools, users can move the mouse pointer with their index finger while controlling the click function with their finger bending [7]. Although web surfing remains the primary purpose, users may do automatic file management tasks, such as creating, opening, renaming, copying, pasting, and deleting folders, using voice controls alone [8]. Python's OS module allows users to manage directories without the need for a physical device. Together with TF-IDF vectorization, machine learning classification improves command recognition to get high accuracy when users speak instructions [1]. Because the system executes parallel operations and threads the voice and gesture inputs for simultaneous processing without any latency, the real-time responsiveness is improved. Confirmation of performed commands is provided via a system feedback feature in the interface, which streamlines processes while preserving efficient performance. This system is a cutting-edge human-computer interface choice that offers sophisticated services for digital navigation users that need accessibility or convenience thanks to real-time optimization of multi-modal processing with hands-free automation [2].

A. System Implementation

Using the help of voice requests and hand gestures made possible by automated file systems, users may navigate webpages using the Touchless Web Navigation System. The system can carry out connected file operations and browser instructions in real time by processing voice commands and hand gestures.

The voice recognition module uses an API from Google voice Recognition to convert spoken commands into written text. The TF-IDF vectorized processed text output is used by the machine learning model (SVM/Random Forest) to determine the proper browser or file activity.

In order to comprehend cursor commands and click intentions, the gesture recognition module tracks finger locations and uses MediaPipe Hands in conjunction with OpenCV to identify hand landmarks. The bending of the index finger generates click functions, and its motions serve as a mouse pointer mechanism [8]. The system performs common surfing tasks, such as document page scrolling, bookmarking, searching, and tab navigation training, using its browser control module [10]. Once the application recognizes speech or gesture instructions, it uses PyAutoGUI to simulate keystrokes and mouse movements.

The module offers hands-free file actions, such as creating and opening folders, renaming them, and performing copy and paste before deleting them. To carry out these instructions within the operating system environment, the system makes use of the OS module.

Within the feedback module, real-time audio feedback using pyttsx3 improves the user experience by giving prompt vocal answers for every action that is completed. The system functions best when it is multithreaded, allowing speech and gesture instructions to be sent simultaneously and instantly. Touchless web surfing and file management are made possible by the integration, which creates an extremely effective and user-friendly interface.

B. Algorithm

Step 1: Initialize System

The program needs to import four essential libraries: SpeechRecognition, MediaPipe Hands, PyAutoGUI, Selenium, keyboard, OS. The system starts by initializing both speech recognition and gesture recognition components. The system requires setup of execution configurations for real-time operations.

Step 2: Capture User Input

Speech Input. The system uses a microphone to record the user voice input. The system transforms verbal input into written text through operation of Google Speech Recognition API. **Gesture Input** The system obtains hand gesture data with the assistance of the MediaPipe Hands framework. The system detects finger positions which enable cursor movement along with performing both click and right-click actions.

Step 3: Process the Command

A text preprocessing process converts every text to lowercase and removes unwanted noise. The system contains predefined actions (browser control and file management) which correspond to receive commands. For Gesture Commands System detects hand activities which produce cursor movement and click functions among others. Fingers should be tracked to recognize the bending motions needed to execute trigger actions.

Step 4: Execute the Action

To execute browser-related commands through Selenium and keyboard automation perform the following actions: The system executes commands for browser operations by opening or closing windows and tab navigation and creates or finds files in interfaces. When executing file management commands you should use

operating system commands to accomplish the task. The system allows users to make, access, rename, duplicate and delete entire file directories. A hand detection will trigger PyAutoGUI to perform appropriate mouse commands.

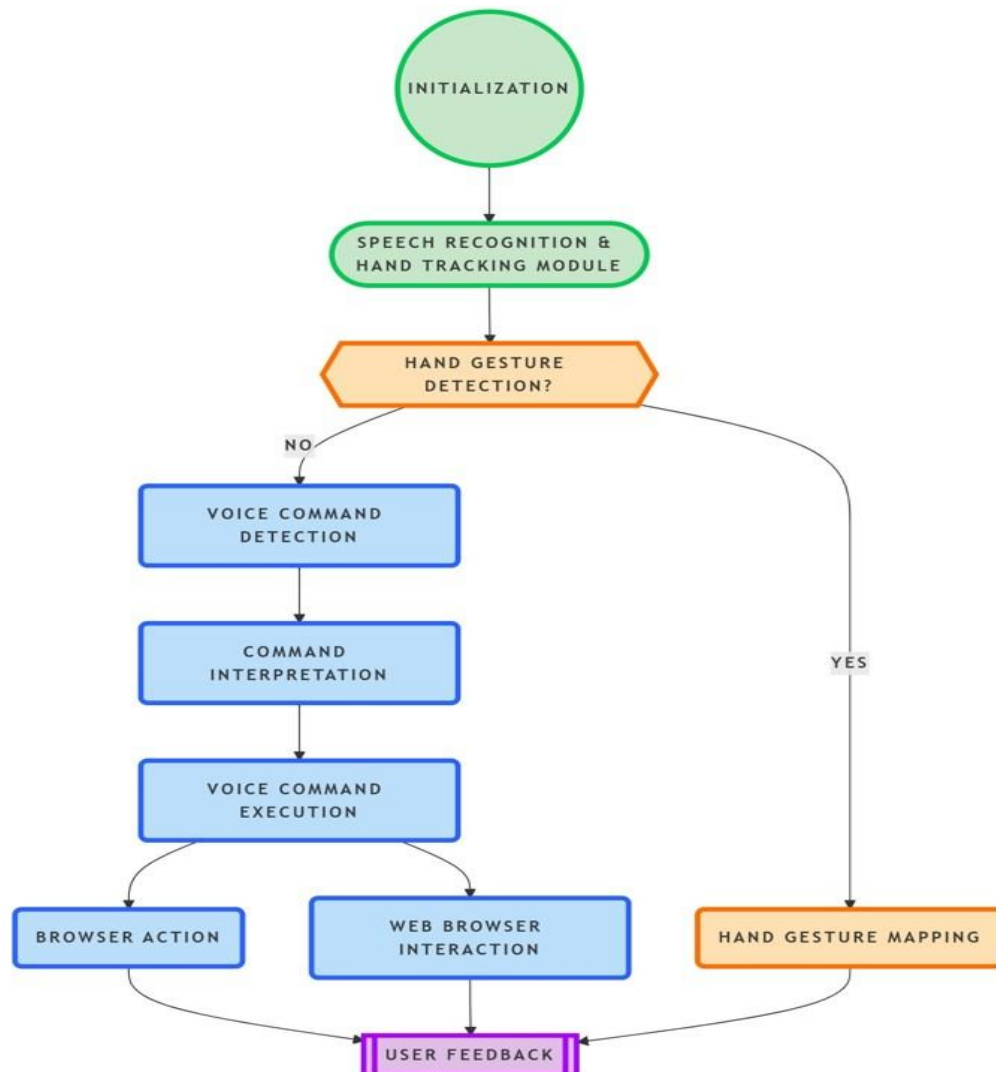


Figure.1 Proposed Touchless Web Navigation System Flow Diagram

Step 5: Provide Feedback

A confirmation of command completion becomes available through audio output. Maintain immediate system response along with restoration of input mechanisms.

Step 6: Loop Continuously

Stay idle for the next command until the command “close browser” is said.

C. Machine Learning Model for Speech Recognition

In order to perform file system and browser tasks, the module may take voice commands. The Google Speech Recognition API is used for live text transcription. Before a Scikit-learn classification model can identify the intended command, the recognized text must first undergo TF-IDF vectorization. This is where model training begins. The model learns over a series of training sessions using the training data from the preset command datasets.

- Commands Used in the Browser: The Chrome browser may start an application and then shut it with a command.
- Tab Navigation: Previous Tab, Next Tab

- Internet Explorer instructions include "CIT" search followed by "down" and "up" scrolling, which moves the cursor to the bottom and top of the screen.
- Page Administration: Page Refresh, Back, Forward
- The Bookmark and Downloads commands, which include the buttons to Open Bookmarks, Bookmark This Page, Close Bookmark, and View Downloads, allow users to access browser functionality.

D. Gesture Recognition Module

Users move the cursor and execute clicks, file commands and path transitions through their gestures. The system applies MediaPipe Hands for hand landmark detection alongside OpenCV for image processing functions. Through PyAutoGUI functions the application moves the mouse cursor at various positions that the system detects. The tracking system runs automatically to track ongoing finger motion enabling fluid system interaction. The index finger bending allows users to execute a click through Click Operation.

E. File Management Module

This module allows end users to create folders, rename or delete them, and copy and paste using gestures and voice commands. The OS module handles file system interactions, whereas the util module handles file operations involving folder moves. To increase user efficiency, the file management system relies on vocal commands that initiate particular system functions. File Management Commands Folder Creation & Deletion: Create [folder name], Delete [folder name] Folder Navigation: Open [folder name] Folder Modification: Rename [folder name] The commands for File Operations combine Copy followed by the [folder name] option alongside Paste using the specified [folder name] when needed.

F. Browser Control Module

The browser control module performs page flipping, scrolling, and new tab actions using performed instructions. The program improves online navigation by combining PyAutoGUI to emulate mouse and keyboard motions with Selenium to automate browser functionality. The synchronization of hand gestures and spoken instructions enables web navigation control with this technique.

G. Integrated Feedback Module

After each command execution, a usability enhancement feature plays auditory confirmation over speakers. All command execution statuses are converted into voice-based output that notifies users of system answers via the pyttsx3 text-to-speech engine. Because the feedback system is most effective for voice-based users, it improves accessibility and usability. These modules overcome the drawbacks of single-modal systems while achieving an effective, hands-free, and accessible online navigation and file management.

H. The Process

Using Google's Speech API, this system recognizes speech input, turns it into text, and then parses the text to match pre-written Python methods [4]. Among other tasks, functions can be used to open programs, create files, search the web, or enter text. Systemd integration and Bash scripting are deployed to guarantee that the operating system boots up automatically. To guarantee an easy run-through of all commands and features, the whole Python code runs in a particular environment known as myenv. Thus, a combination of all these technologies is used to provide a smooth and effective user experience [7].

V. RESULT AND DISCUSSION

TABLE I. COMPARISON TABLE OF MODULES AND PERFORMANCE

Module	Technologies Used	Functionality	Performance Results	Remarks
File Management	os, util, Speech API	Voice-controlled folder creation, deletion, copy-paste	92–95% accuracy in command recognition; Real-time response in <1 sec	Highly intuitive; minimal training needed; worked well under varied conditions
Browser Control	PyAutoGUI, Selenium, MediaPipe Hands	Gesture and voice-driven browser navigation	90% gesture detection accuracy; Smooth tab switching and scrolling	Slight lag in complex gestures; Chrome-compatible; future work needed on multi-browser support
Integrated Feedback	pyttsx3 (TTS engine)	Spoken confirmation of each command execution	100% feedback success rate; Clear and consistent voice output	Strong accessibility support; Users relied heavily on this feature for confidence
Speech Recognition &	Google Speech API, Python functions	Speech-to-function mapping and execution	~95% speech accuracy in quiet spaces; Fast text	Background noise slightly reduced accuracy; Consideration for

Execution			parsing and function mapping	offline fallback recommended
System Integration	Systemd, Bash, virtual environment (myenv)	Ensures auto-launch and stable runtime	Fully automated startup; No crashes during 9+ hour test	Crucial for long-session usage; helps maintain environmental control and version consistency

Summary of Key Observations:

- The multi-modal interface (voice + gesture) significantly improved usability and accessibility.
- Auditory feedback closed the loop on user interaction, reducing the need for screen validation.
- Integration across modules was seamless, with each component enhancing the overall hands-free experience.
- Real-world limitations such as noise and hardware latency were noted but did not severely hinder performance.
- Future improvements could include AI-based contextual command recognition, offline speech support, and gesture customizations.

VI. CONCLUSION

A hands-free web and file management system that combines gesture tracking and speech recognition is presented in a research study. There is a system that uses machine learning techniques, natural language processing, and computer vision to improve accessibility and usability while increasing interaction efficiency. Accurate command understanding and practical gesture control are made possible by MediaPipe Hands technology and TF-IDF-based speech categorization. Future updates to the system will include improved command support, contextual awareness integration based on artificial intelligence, and performance optimization in a variety of contexts.

REFERENCES

- [1] Amberkar, Aditya & Awasarmol, Parikshit & Deshmukh, Gaurav & Dave, Piyush. (2018). Speech Recognition using Recurrent Neural Networks. 1-4. 10.1109/ICCTCT.2018.8551185.
- [2] Christopher Collins, Denis Dennehy, Kieran Conboy, Patrick Mikalef, Artificial intelligence in information systems research: A systematic literature review and research agenda, International Journal of Information Management, Volume 60, 2021, 102383, ISSN 0268-4012, <https://doi.org/10.1016/j.ijinfomgt.2021.102383>.
- [3] Suman, Shashikant & Lohani, Bhanu & Singh, Vijay & Gupta, Amar & Khan, Akhilesh & Kumar, Anil. (2023). Two-way Speech to Sign Language Converter application using Python, OpenCV and NLP. 255-260. 10.1109/IC3I59117.2023.10398166.
- [4] G. Preethi , Abishek. K , Thiruppugal S , Vishwaa D A, 2022, Voice Assistant using Artificial Intelligence, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 11, Issue 05 (May 2022), <https://doi.org/10.48550/arXiv.2312.05640>
- [5] <https://doi.org/10.48550/arXiv.2312.05640>
- [6] Kulkarni, Mayuresh & Kamble, Torana. (2020). Integration of Machine Learning into Operating Systems: A Survey. 1270.
- [7] Rithika V. Jadhav, Prajakta S. Bait, Sumedh Pundkar. Controlling web browser via different hand gestures and speech recognition, International Journal of Advance Research, Ideas and Innovations in Technology, www.IJARIIT.com.
- [8] Kumaran, Natarajan & Surya, Balina & Vamsi, Bobba. (2023). WEB BROWSER CONTROL USING HAND GESTURES. 7. 10.55041/IJSREM21244.
- [9] Senthilkumar, Anitha. (2024). GestureSpeak: Real-Time Hand Gesture Translation to Text and Audio For Speech-Impaired.
- [10] Radostev, V., Berger, S., Tabrizi, J., Kamyshev, P., Suzuki, H. (2019) Speech-Based Web Navigation for Limited Mobility Users. Proc. Interspeech 2019, 976-977.