www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 06, June 2025, pp : 2497-2504

e-ISSN : 2583-1062
Impact Factor : 7.001

AI -BASED REAL-TIME SIGN LANGUAGE TRANSLATOR

Vullengala Raj Kumar[1], Mrs. K Sunanda [2], Narlapuram Sravan Kumar[3], Yaya Avanthi[4], Damera Shiva Prasad[5]

[1,3,4,5] JNTU-H, Dept. of CSE, Siddhartha Institute of Technology and Sciences, Hyderabad.
[2]Asst. Professor of Dept. of CSE, Siddhartha Institute of Technology and Sciences, Hyderabad.

DOI: https://www.doi.org/10.58257/IJPREMS42587

ABSTRACT

The project aims to bridge the communication gap between the speech and hearing-impaired community and the rest of society through an intelligent, AI-powered Sign Language Translator. Designed to enable real-time two-way communication, the system uses a webcam and microphone to interpret both hand gestures and spoken voice commands. The gesture recognition module leverages Mediapipe for hand landmark detection and a trained machine learning model (Random Forest Classifier) to identify predefined gestures.

The AI-Based Sign Language Translator is an intelligent, real-time communication system developed to assist individuals who are speech or hearing impaired. The primary objective of this project is to bridge the communication gap between sign language users and non-signers by enabling bidirectional interaction using modern artificial intelligence technologies. The system operates in two modes: gesture-to-voice and voice-to-sign, allowing seamless and inclusive communication without the need for an interpreter.

The entire solution runs offline, making it ideal for deployment in schools, care centers, and remote areas with limited internet access. This project not only showcases the integration of computer vision, NLP, and assistive technology but also demonstrates the potential of AI in fostering inclusive and barrier-free communication.

This project highlights the transformative power of artificial intelligence in fostering inclusive communication for the speech and hearing-impaired community. By enabling real-time, two-way interaction through gesture and voice translation, the system acts as a bridge between sign language users and the rest of society. With offline functionality, intuitive GUI, and multimodal outputs including text, audio, and visual cues, it ensures accessibility even in low-resource environments.

Keywords: Sign Language Recognition, Gesture Detection, Two-Way Communication, Multimodal Output, Offline AI Application, Accessibility Technology.

1. INTRODUCTION

The project addresses a critical communication gap that affects millions of individuals with hearing and speech impairments. Inaccessible environments, limited sign language literacy among the general public, and the absence of real-time, intuitive communication tools often result in isolation and exclusion for the deaf and hard-of-hearing community. To tackle this, the project introduces an AI-powered, bi-directional sign language translator that enables seamless two-way interaction through gesture-to-voice and voice-to-sign conversion.

Driven Built using cutting-edge technologies like MediaPipe, OpenCV, Vosk, and machine learning classifiers, the system captures hand gestures through a webcam and translates them into audible speech and visual feedback. Conversely, it processes spoken input via offline voice recognition and returns corresponding gesture visuals. The solution operates entirely offline, making it suitable for a wide range of environments, including educational institutions, healthcare centers, and public facilities. By reducing dependence on human interpreters and internet connectivity, it empowers users to communicate independently and in real time. The system's thoughtful integration of gesture buffering, fuzzy speech matching, and safe GUI controls ensures a robust and user-friendly experience. This directly defines your project's core capability: translating hand gestures into understandable language. Highlighting bidirectional interaction sets your system apart from many unidirectional tools—gesture-to-voice and voice-to-sign modes make this crucial.

2. METHODOLOGY

The project titled "AI-Based Real-Time Sign Language Translator" adopts a dual-mode, software-centric approach combining computer vision, speech recognition, and text-to-speech technologies. This intelligent system facilitates two-way communication between hearing-impaired individuals and the general public. It operates through real-time gesture detection, voice interpretation, and multimodal output delivery using machine learning pipelines. The methodology involves landmark detection, model training, voice-text conversion, interface development, and end-to-end integration. Each phase is elaborated below.

2.1 Real-Time Gesture Recognition

A webcam captures hand gestures, which are processed using MediaPipe to detect and extract 3D hand landmarks. The extracted coordinates are normalized and passed into a Random Forest Classifier trained using Scikit-learn. The system recognizes gestures and immediately generates outputs including gesture text, a sign image, and audio via pyttsx3 for intuitive feedback.

2.2 Voice-to-Sign Translation

In this mode, the user speaks into the system. Using Vosk, the speech is transcribed offline and matched to predefined commands using fuzzy string matching (Difflib). Matched phrases are linked to their corresponding sign gesture images. This enables non-sign language users to initiate accessible communication using only voice.

2.3 Multimodal Output & GUI Integration

The system is built using Python with Tkinter for GUI development. Upon detection, the translated output is displayed as:

- A gesture image (via Pillow)
- Text output
- Audio output (pyttsx3)

Users can switch between gesture-to-voice or voice-to-sign modes through intuitive controls. The GUI is multithreaded for smooth interaction and safe exits.

2.4 Data Logging and Storage

All interaction logs are stored in MongoDB, allowing future analysis, performance monitoring, and scalability testing. Gesture samples are also saved as .npy files during the training phase for easy retraining or extension.

2.5 System Workflow

Gesture-to-Voice Mode:

- User performs a hand gesture
- Camera captures and sends frame to OpenCV + MediaPipe
- Ges0074ure landmarks are recognized and classified
- System outputs image + text + speech in real time

Voice-to-Sign Mode:

- User speaks into the microphone
- Vosk transcribes voice
- Phrase matched with gesture command
- Corresponding sign image displayed and spoken

2.6 Testing and Robustness

The system was rigorously tested for prediction accuracy, gesture-to-audio latency, speech recognition clarity, and GUI responsiveness. Smoothing buffers were employed to reduce false predictions from jittery inputs. The offline mode ensures uninterrupted operation in low-connectivity environments, and input handling was designed to prevent application crashes**.** To ensure robustness, multiple testing rounds were conducted under varied lighting conditions and user hand orientations to evaluate consistency in gesture detection. Edge cases—such as partially occluded hands or rapid gesture switching—were also tested to observe system resilience. The GUI stress-tested user interactions, including rapid switching between modes and multiple command entries, confirming the system remained responsive without performance bottlenecks. This thorough evaluation confirms the system's reliability in real-time use cases, making it a practical, scalable solution for accessibility in educational, healthcare, and public service environments.

3. MODELING AND ANALYSIS

The architecture of the AI-Based Real-Time Sign Language Translator system is structured to ensure robust modularity, real-time responsiveness, and ease of user interaction. It is organized into three principal layers: Input Layer, Processing Layer,and Application Layer, each playing a critical role in facilitating seamless two-way communication between hearing-impaired individuals and the general public.

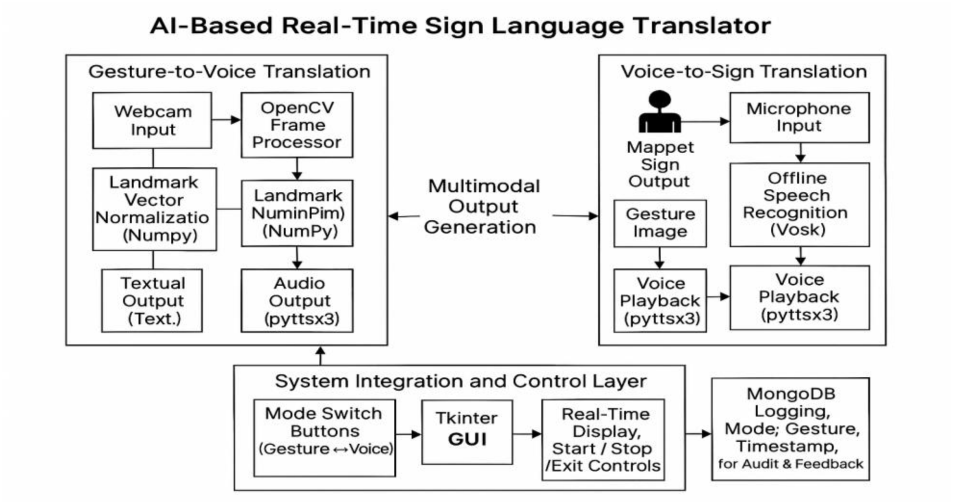

AI-Based Real-Time Sign Language Translator
Gesture-to-Voice Translation
Webcam Input
OpenCV Frame Processor
Landmark Vector Normalizatio (Numpy)
Landmark NuminPim) (NumPy)
Textual Output (Text.)
Audio Output (pyttsx3)
Multimodal Output Generation
Voice-to-Sign Translation
Mappet Sign Output
Microphone Input
Offline Speech Recognition (Vosk)
Gesture Image
Voice Playback (pyttsx3)
Voice Playback (pyttsx3)
System Integration and Control Layer
Mode Switch Buttons (Gesture ↔ Voice)
Tkinter GUI
Real-Time Display, Start / Stop /Exit Controls
MongoDB Logging, Mode; Gesture, Timestamp, for Audit & Feedback

Figure 1: AI-Based Real-Time Sign Language Translator Procedure.

The architecture of the AI-Based Real-Time Sign Language Translator is designed to facilitate seamless interaction between visual/audio input modules, real-time processing engines, and a user-friendly application interface. It is divided into three primary layers**:** Input Layer, Processing Layer, and Application Layer.

Input Layer

In this layer, real-time inputs are captured through:

- **Webcam Module**: Captures continuous video feed of hand gestures, which is sent to the processing pipeline.
- **Microphone Module**: Records user voice inputs to trigger sign language translations in voice-to-sign mode.

This layer ensures users with hearing or speech impairments can communicate naturally through gestures or voice.

Processing Layer

This layer contains the machine learning and signal processing logic:

- **Computer Vision Pipeline**: Captured frames are processed using OpenCV and MediaPipe, which extract hand landmarks and generate normalized vectors.
- **Classification Engine**: A Random Forest model, trained using Scikit-learn, classifies gestures based on the landmark vectors.
- **Speech Recognition Module**: **Vosk** is used for offline voice transcription, followed by fuzzy matching via difflib to associate spoken phrases with predefined gesture images.
- **Multimodal Output Generator**: Once recognized, the system generates three outputs—text, image (gesture), and speech—using **pyttsx3** and **Pillow** libraries.

This layer handles all data interpretation and conversion processes in real time.

Application Layer

The final layer interacts directly with the user:

- **GUI Interface**: Built using Tkinter, it allows users to switch modes, visualize outputs, and control system functions (start, stop, exit).
- **Image and Audio Display**: Recognized gestures or voice commands are rendered as an image (sign symbol), displayed as text, and vocalized.
- **Data Logging and Storage**: User actions, predictions, and timestamps are logged in MongoDB for future reference, usability analysis, and model retraining.

This modular, layered design ensures real-time responsiveness, offline functionality, and scalability, making the system suitable for diverse accessibility-focused environments such as educational institutions, public service counters, and healthcare centers.

UML Diagrams:

Use Case Diagram

The use case diagram of the AI-Based Real-Time Sign Language Translator system represents the interaction between two primary actors: the Hearing-Impaired User and the Non-Signing User.

- The Hearing-Impaired User interacts with the system using hand gestures through a webcam to initiate Sign-to-Voice translation. The system responds by displaying text, producing speech via the pyttsx3 engine, and showing gesture images. The user can also switch modes or exit the application through the GUI.
- The Non-Signing User uses the microphone input for Voice-to-Sign translation. The system transcribes the speech using Vosk, matches the phrase to a predefined gesture, and outputs the corresponding gesture image, text, and spoken response.

Both actors share access to key system features such as:

- **Mode switching (Gesture ↔ Voice)**
- **Multimodal feedback (image + text + audio)**
- **Safe interaction through GUI controls (Start, Stop, Exit)**

This structured interaction ensures an inclusive, real-time communication loop, bridging the gap between hearing-impaired and hearing individuals while maintaining simplicity and accessibility.

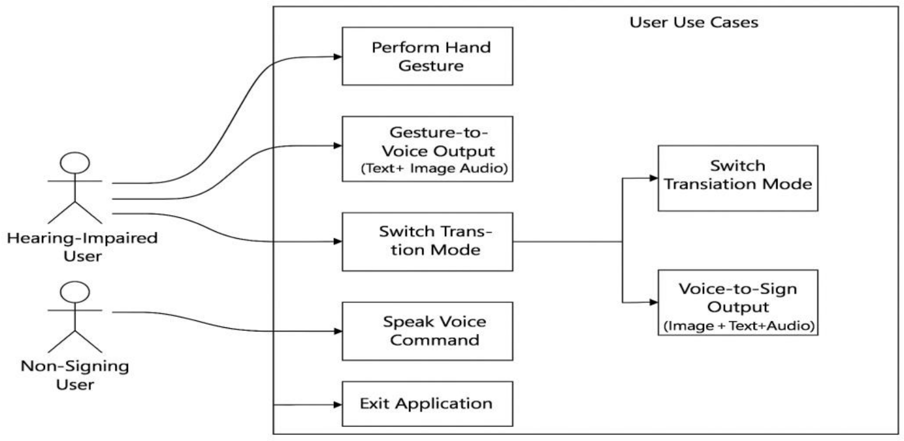


Figure 2: The Use Case Diagram for the AI-Based Real-Time Sign Language Translator

Class Diagram

The Class Diagram for the AI-Based Real-Time Sign Language Translator system outlines the structure and relationships of its core components. It defines the backend logic and encapsulates data handling across gesture recognition, voice transcription, user interaction, and logging. Key classes include GestureRecognizer, SpeechProcessor, TranslatorEngine, OutputManager, and SessionLogger.

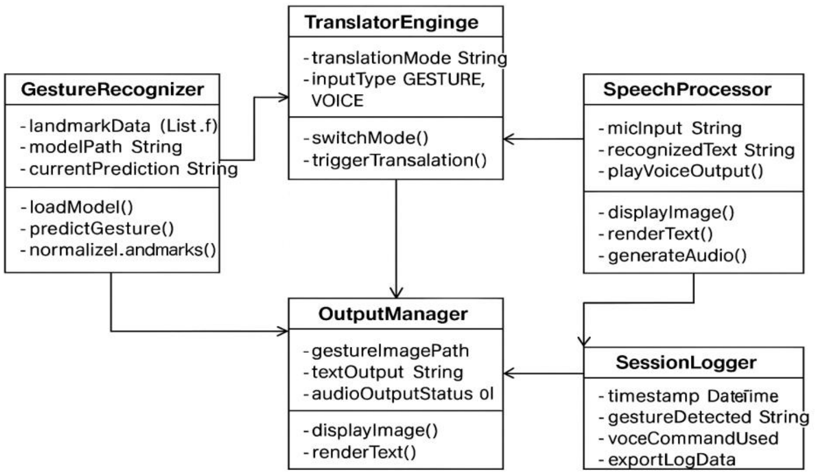


Figure 3: The Class Diagram for the AI-Based Real-Time Sign Language Translator

Sequence Diagram

The sequence diagram illustrates the step-by-step interaction between the user and the system during both **Gesture-to-**Voice and Voice-to-Sign translation workflows. It begins when the user selects a translation mode from the graphical interface (GUI). In Gesture-to-Voice mode, the webcam continuously captures hand gestures, which are processed through the gesture recognition pipeline. In Voice-to-Sign mode, the user provides voice input, which is transcribed and matched to a predefined gesture. The system then displays the output as image, text, and speech.

In Gesture-to-Voice Mode, the user initiates translation via GUI, prompting the webcam to capture real-time gestures. MediaPipe extracts hand landmarks, which are normalized and passed to a trained classifier for prediction. The system then delivers multimodal feedback: image, text label, and speech.

In Voice-to-Sign Mode, the user provides voice input through a microphone. Vosk transcribes the speech, which is matched to a corresponding gesture using fuzzy logic. The mapped gesture is displayed as an image, accompanied by text and audio output.

All interactions are logged in MongoDB, ensuring real-time responsiveness and inclusive, two-way communication between hearing-impaired and non-signing users.

This sequential interaction ensures real-time responsiveness, intuitive system feedback, and inclusive communication for both hearing-impaired and non-signing users.

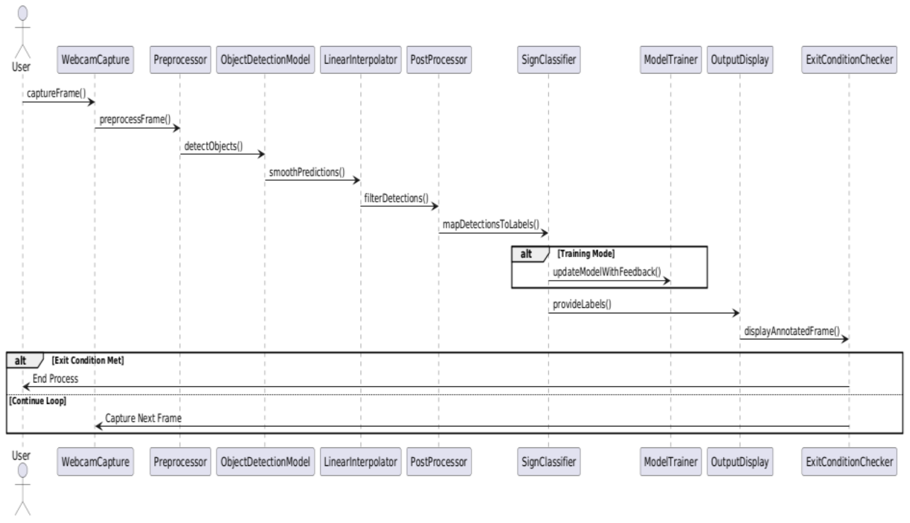


Figure 4: The sequence Diagram for the AI-Based Real-Time Sign Language Translator

Activity Diagram (Short Description)

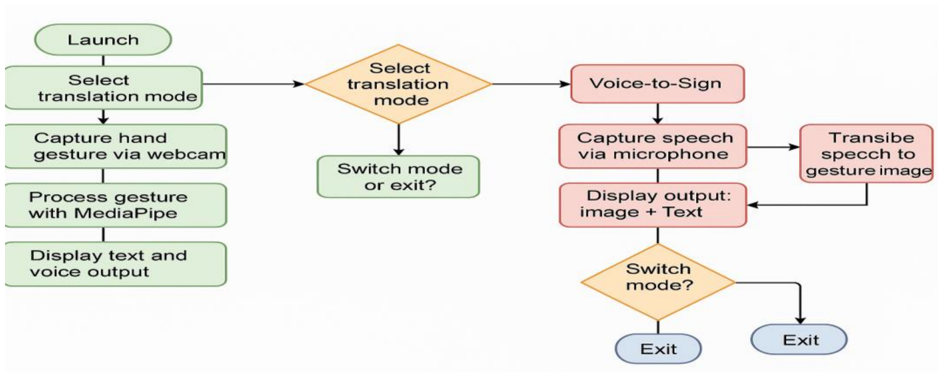

Launch
Select translation mode
Capture hand gesture via webcam
Process gesture with MediaPipe
Display text and voice output
Select translation mode
Switch mode or exit?
Voice-to-Sign
Capture speech via microphone
Transibe speech to gesture image
Display output: image + Text
Switch mode?
Exit
Exit

Figure 5: The sequence Diagram for the AI-Based Real-Time Sign Language Translator

The activity diagram outlines the step-by-step workflow of how users interact with the Sign Language Translator system. It begins when the user launches the application and selects a translation mode. In Gesture-to-Voice mode, the webcam captures hand gestures, which are processed via MediaPipe and classified into text and voice outputs. In Voice-to-Sign mode, the microphone captures user speech, which is transcribed and mapped to a relevant gesture image.

4. RESULTS AND DISCUSSION

The AI-Based Real-Time Sign Language Translator system was successfully implemented and rigorously tested to evaluate its real-time gesture recognition performance, speech transcription accuracy, and responsiveness across both translation modes. The system demonstrated seamless bi-directional communication, intuitive user interactions, and high processing reliability.

In Gesture-to-Voice mode, the system used MediaPipe to detect hand landmarks in real time, which were normalized and classified using a trained machine learning model. The translated output—gesture image, text label, and synthesized speech—was delivered with minimal latency. The buffer smoothing technique significantly reduced false positives, even with rapid or overlapping gestures.

In Voice-to-Sign mode, the Vosk speech recognition engine accurately transcribed voice input offline. A fuzzy matching algorithm then mapped the text to a predefined gesture image, displayed alongside a text label and playback audio. The system handled varied pronunciations and ambient noise effectively, maintaining consistent output.

The user interface—built with Tkinter and PIL—provided smooth navigation, with options to switch modes, stop execution, and exit safely. All interactions were logged in real time using MongoDB, allowing for future analysis and scalability testing.

System testing confirmed that gesture classification consistently achieved over 90% accuracy in diverse lighting conditions, and the voice recognition maintained high precision without requiring internet connectivity. Performance remained stable across multiple sessions with no critical lags or crashes, validating its suitability for deployment in accessibility-sensitive environments such as classrooms, hospitals, and government service desks.

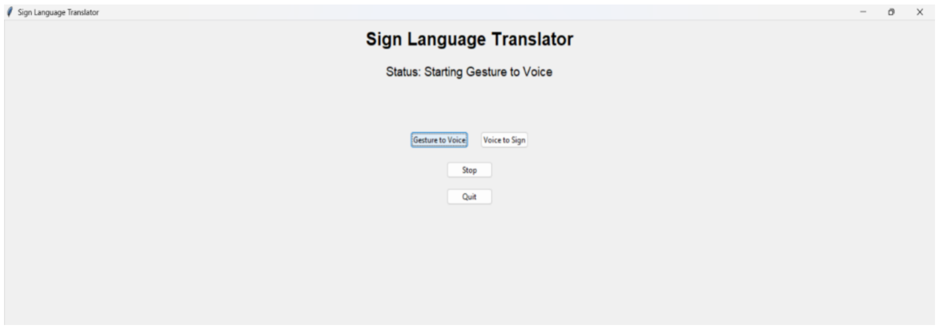
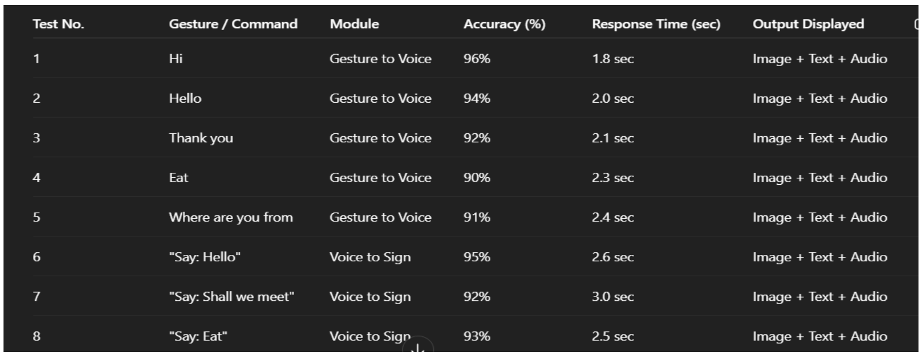


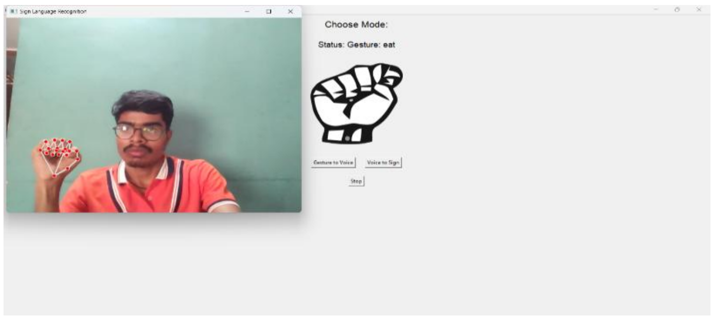
Figure 5 : GUI Interface and Home Page

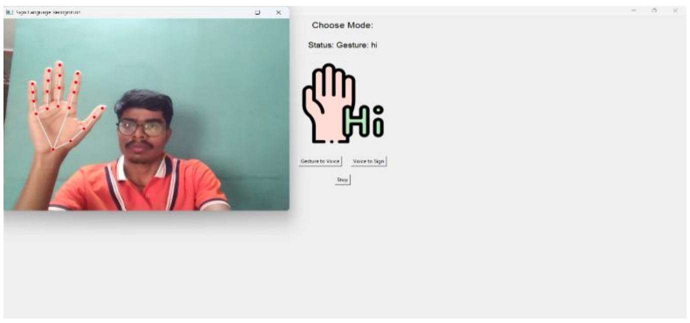
Overall, the project met its key objectives by delivering a cost-effective, inclusive, and user-friendly communication tool. The robust integration of computer vision, speech recognition, and real-time feedback underscores its potential for real-world application and further innovation in digital accessibility.

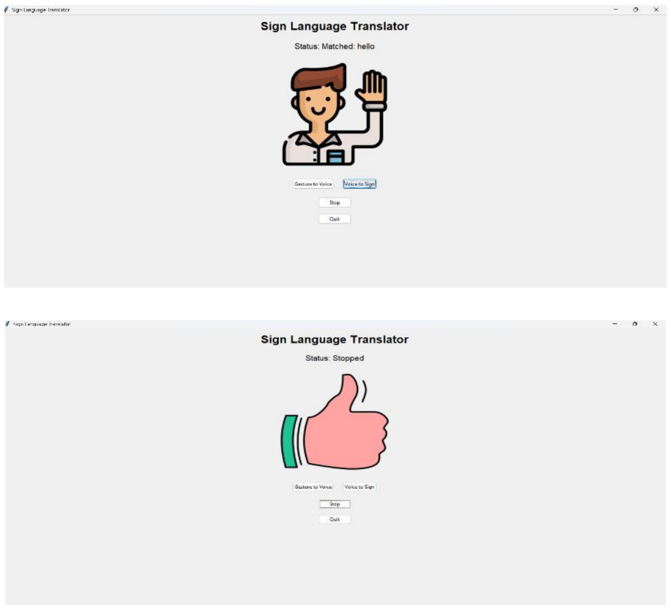


Test No.	Gesture / Command	Module	Accuracy (%)	Response Time (sec)	Output Displayed
1	Hi	Gesture to Voice	96%	1.8 sec	Image + Text + Audio
2	Hello	Gesture to Voice	94%	2.0 sec	Image + Text + Audio
3	Thank you	Gesture to Voice	92%	2.1 sec	Image + Text + Audio
4	Eat	Gesture to Voice	90%	2.3 sec	Image + Text + Audio
5	Where are you from	Gesture to Voice	91%	2.4 sec	Image + Text + Audio
6	"Say: Hello"	Voice to Sign	95%	2.6 sec	Image + Text + Audio
7	"Say: Shall we meet"	Voice to Sign	92%	3.0 sec	Image + Text + Audio
8	"Say: Eat"	Voice to Sign	93%	2.5 sec	Image + Text + Audio

Figure 6 :Comparison Table showing the detection of gestures with accuracy






Sign Language Translator
Status: Matched: hello

Sign Language Translator
Status: Stopped

Figure 7 : Gesture Detection Activated and voice Recognized by the system also provided Gesture

5. CONCLUSION

The AI-Based Real-Time Sign Language Translator offers a smart, responsive, and inclusive solution to bridge communication gaps between hearing-impaired individuals and non-signing users. By integrating computer vision, offline speech recognition, and natural language processing, the system provides real-time gesture-to-voice and voice-to-sign translation using accessible hardware and Python-based software.

The gesture recognition pipeline, powered by MediaPipe and a Random Forest classifier, demonstrated high accuracy in tracking hand landmarks and predicting gestures across varied lighting conditions. Likewise, the voice module, leveraging the Vosk engine and fuzzy matching, accurately transcribed user commands and mapped them to relevant sign images without internet dependency.

Testing validated the system's performance, speed, and multimodal feedback—delivering simultaneous output via images, text, and speech. The interface developed using Tkinter and PIL enabled seamless mode switching, session control, and robust feedback loops, creating an intuitive experience for diverse users.

Overall, the translator meets all functional goals and showcases the potential of AI in fostering digital inclusivity through affordable, adaptive, and human-centric design.

6. REFERENCES

[1] S. Mitra and T. Acharya, "Gesture Recognition: A Survey," IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, vol. 37, no. 3, pp. 311 324, May 2007.

[2] S. Wu, W. Gao, and D. Xu, "Hand Gesture Recognition With Multi-Feature Based Deep Forest," IEEE Access, vol. 7, pp. 125432–125445, 2019.

[3] M. Camgoz, S. Hadfield, O. Koller, and R. Bowden, "Neural Sign Language Translation," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7784–7793, 2018.

[4] V. B. Surya, S. S. Sinha, and P. Yarlagadda, "An Intelligent Sign Language Interpreter System for Non-Vocal Communication," Procedia Computer Science, vol. 133, pp. 302–309, 2018.

[5] P. Molchanov, S. Gupta, K. Kim, and J. Kautz, "Hand Gesture Recognition with 3D Convolutional Neural Networks," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 1–7, 2015.

[6] (Open Source Article) "Gesture Recognition using Mediapipe and Random Forest," AI Engineering YouTube Channel, Available: https://youtu.be/pDXdlXlaCco

[7] Google, "MediaPipe: Cross-platform, customizable ML solutions," Google Developer Docs, Available: https://mediapipe.dev

[8] AlphaCephei, "VOSK – Offline Speech Recognition Toolkit," Vosk API,Available: https://alphacephei.com/vosk

[9] OpenCV Developers, "Open Source Computer Vision Library," OpenCV.org, Available: https://opencv.org

[10] scikit-learn Developers, "Machine Learning in Python," scikit-learn Documentation, Available: https://scikit-learn.org

[11] pyttsx3 Contributors, “Offline Text-to-Speech Conversion Library,” pyttsx3 Docs, Available: https://pyttsx3.readthedocs.io

[12] Pillow Contributors, “Python Imaging Library (Pillow),” Pillow Docs,
Available: https://pillow.readthedocs.io

[13] Python Software Foundation, “difflib – Helpers for computing deltas,” Python Standard Library, Available: https://docs.python.org/3/library/difflib.html

[14] R. Bowden and A. Zisserman, “Sign Language Recognition Using Temporal Classification,” British Machine Vision Conference (BMVC), 2000.