Sign Language To Text And Speech Conversion Using Hand Landmarks

Alampally Sravani – 22911A1269   
*Department of Information Technology,*

*Vidya Jyothi Institute of Technology*Perka Vaishali – 22911A12A3  
*Department of Information Technology,*

*Vidya Jyothi Institute of Technology*

Tirumalasetti Jyothirmai– 22911A12C0  
*Department of Information Technology,*

*Vidya Jyothi Institute of Technology*

Uppuleethi Varshitha– 22911A12C1  
*Department of Information Technology,*

*Vidya Jyothi Institute of Technology*

Mrs. G. Indira Priyadarshini  
*Associate Professor, Department of Information Technology, Vidya Jyothi Institute of Technology priyadarshni@vjit.ac.in*Dr. A. Obulesu  
*HOD, Department of Information Technology, Vidya Jyothi Institute of Technology*

*Abstract*— This project presents a real-time application that converts American Sign Language (ASL) hand gestures into both text and speech. It uses a standard webcam to capture gestures and applies MediaPipe for detecting hand landmarks. A set of predefined rules is then used to recognize static ASL alphabet signs without the need for machine learning models or training data. Recognized characters are displayed on-screen and optionally converted to speech using a text-to-speech engine. The system is designed to work offline, making it a low-cost, accessible, and user-friendly solution for facilitating communication between hearing-impaired individuals and the general public.

# **Introduction**

Communication is fundamental to human interaction. For individuals with hearing or speech impairments, sign language plays a crucial role in expressing themselves. However, due to the limited understanding of sign language among the general public, these individuals often face challenges in daily life, such as during medical consultations, education, or customer service interactions. This communication gap leads to social isolation, dependence on interpreters, and restricted access to opportunities. With the advancement of computer vision and artificial intelligence, several attempts have been made to develop systems that can recognize sign language and translate it into spoken or written form. This project focuses on building a real-time, accessible, and cost-effective solution that converts static American Sign Language (ASL) alphabet gestures into text and speech using only a webcam and open-source libraries.

## **Existing Systems**

Several systems have been developed to recognize and translate sign language gestures, such as glove-based systems, depth camera solutions, image processing methods, deep learning models, and mobile applications.

Glove-based systems use sensors to detect finger movements with high accuracy, but they are often costly and inconvenient for everyday use. Depth camera systems like Microsoft Kinect offer reliable 3D gesture tracking but require expensive hardware and controlled environments. Image processing approaches work with standard webcams and use techniques like contour detection, but they are sensitive to lighting and background changes.

Deep learning models, including CNNs and RNNs, can provide high accuracy in gesture recognition but require large datasets and powerful hardware, making them less suitable

for offline use. Mobile applications offer portability but are limited in performance due to device constraints and camera positioning.

### **Drawbacks:**

* Require expensive and specialized hardware such as gloves or 3D cameras.
* Sensitive to environmental factors like lighting and background noise.
* High computational demand and need for training data in deep learning approaches
* Lack of offline functionality in many cloud-dependent models.
* Often not scalable or practical for real-world use, especially in low-resource settings.

## **Proposed Systems:**

To overcome these limitations, the proposed system introduces a rule-based approach that recognizes static ASL alphabet gestures using only a standard webcam and open-source libraries like MediaPipe, OpenCV, and pyttsx3. The system captures live video, detects hand landmarks using MediaPipe's real-time hand tracking module, and classifies gestures using predefined geometric rules without the need for training data or machine learning models. Recognized letters are appended into words or sentences, which are then displayed on-screen and can be converted into speech output using the text-to-speech engine.

1. ***Advantages:***

* Low cost implementation with no need for sensors or specialized hardware—only a basic webcam is required.
* Highly inclusive and accessible, ideal for use in educational institutions,rural areas, and assistive technology solutions.
* Simple and intuitive user interface featuring controls like Predict, Delete, Clear, Space, and Speak
* Real-time gesture recognition with immediate text and speech output.

# **Literature Survey**

**Survey 1**: ASL Finger-Spelling Recognition using Basic Image Processing by Krishna Modi and Amrita More (Mukesh Patel School of Technology and Management Engineering, Mumbai) presents a low-cost system that converts ASL finger-spelling into English text using a standard webcam. Using basic image processing techniques, the system achieves 96% accuracy in real time and supports both left- and right-handed users, making it suitable for educational and low-resource settings

**Survey 2**: ASL Gesture Translation using CNN and Text-to-Speech by Ankit Ojha, Ayush Pandey, Shubham Maurya, and Abhishek Thakur (JSS Academy of Technical Education, Bangalore) presents a CNN-based system that translates static ASL gestures into both text and speech. Trained on a gesture image dataset with 95% accuracy, the system uses OpenCV for image processing and pyttsx3 for speech output. The study highlights the model’s strength in handling complex gestures through deep learning, while noting its reliance on computational resources for optimal performance.

# **Methodology**

The methodology describes the systematic design and functioning of the proposed system. The application architecture, internal components, workflow, and module responsibilities are defined to ensure real-time performance, simplicity, and ease of deployment.

## **Architecture**

The methodology describes the systematic design and functioning of the proposed system. The application architecture, internal components, workflow, and module responsibilities are defined to ensure real-time performance, simplicity, and ease of deployment.

### **Components:**

**•** Webcam Input**:** Captures a live video stream of hand gestures in real time and continuously feeds each frame into the processing pipeline for analysis.

**•** Hand Tracking (MediaPipe): Utilizes MediaPipe's pre-trained model to detect 21 hand landmarks, including fingertips, joints, and the palm center, enabling precise gesture tracking.

**•** Gesture Recognition Logic: Implements rule-based logic to interpret landmark positions and classify static ASL alphabets by determining which fingers are raised or folded.

**•** Text Output Module**:** Displays the recognized characters on the graphical interface and appends them to form complete words and sentences for user readability.

**•** Text-to-Speech (pyttsx3): Converts the final sentence into audible speech using the pyttsx3 engine. This module operates fully offline and supports voice customization features.

**•** Graphical User Interface (GUI): Developed using Tkinter, the interface provides buttons for user interaction such as Predict, Clear, Delete, Speak, and Space, ensuring a simple and accessible user experience

### **Work Flow:**

1.Start the application and activate the webcam.

2. Live hand gestures are captured and passed to MediaPipe.

3. MediaPipe extracts hand landmarks (21 key points).

4.Gesture logic is applied to check whichfingers are open/closed.

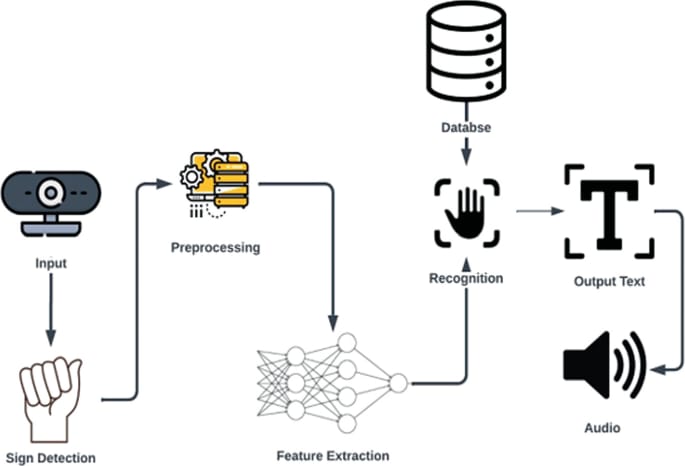
5. The system identifies the corresponding ASL alphabet.

6. The recognized letter is displayed and appended to the text buffer.

7. User can clear, delete, add space, or convert the sentence to speech

8. This cycle repeats for each hand gesture until stopped.

##### 



##### 

##### 

Fig. 1. System Architecture

## **UML Diagram**

### **Class Diagram:**

A class diagram represents the object-oriented structure of a system by showing the system's classes, their attributes, methods, and relationships. It helps in visualizing the static structure of the system, making it easier to understand how objects interact within the system.

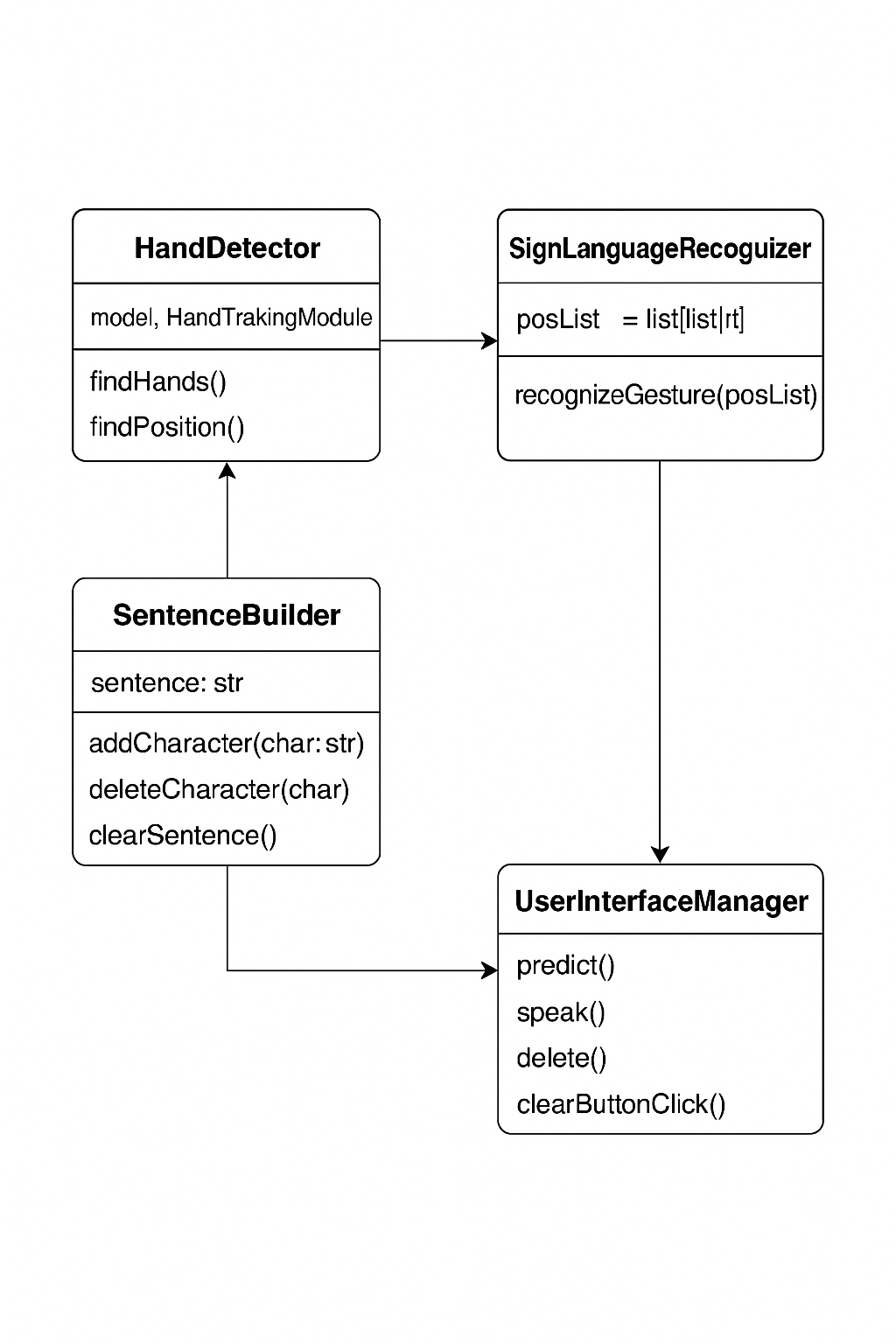


Fig. 2. Class Diagram

### **Sequence Diagram:**

A sequence diagram is used to illustrate how objects or components interact with each other over time. It shows the dynamic flow of messages between participants (such as classes, objects, or system components) in the system. The diagram provides insight into the order of interactions and how the system components collaborate to complete a specific process

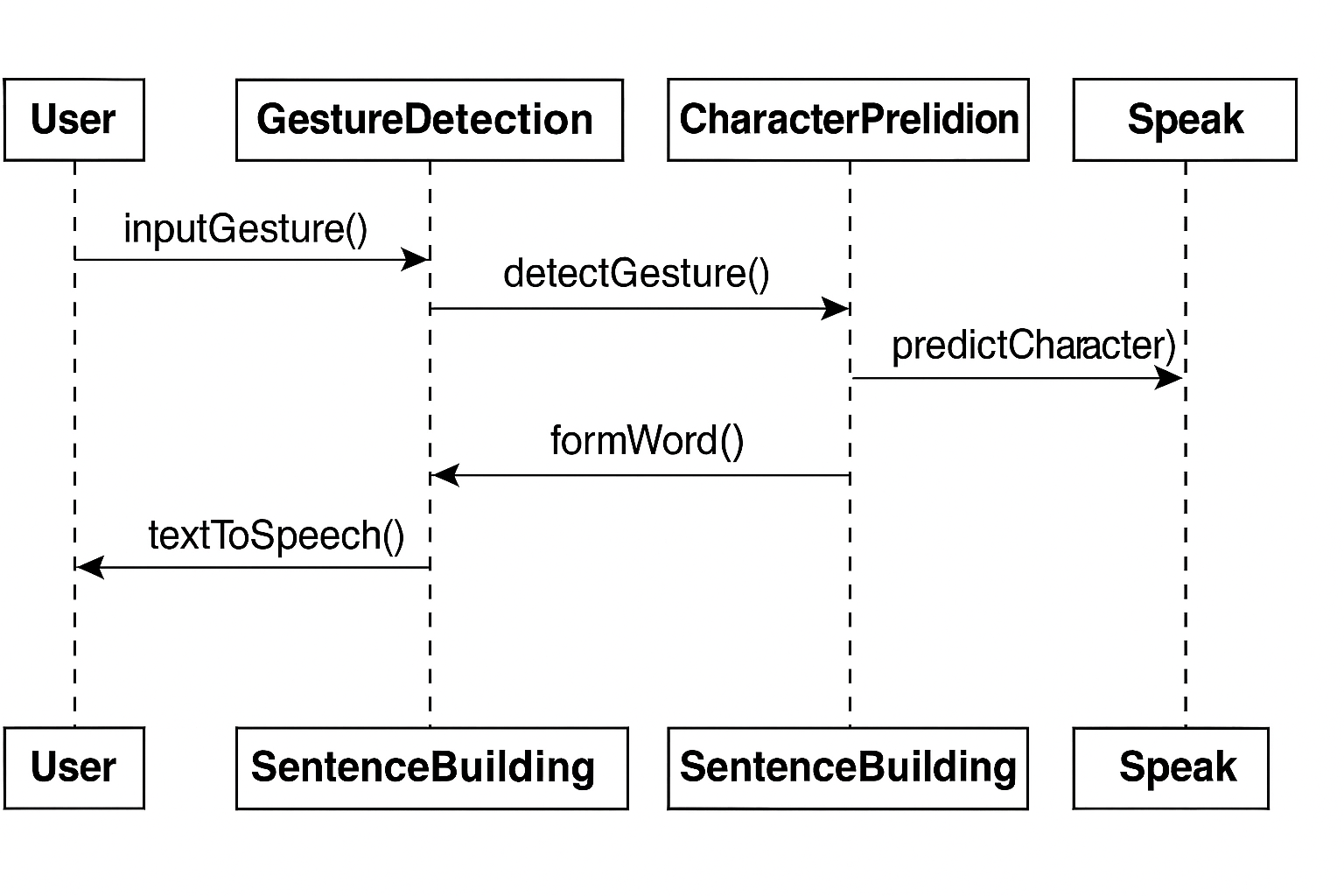


Fig. 3. Sequence Diagram

### **Use case Diagram:**

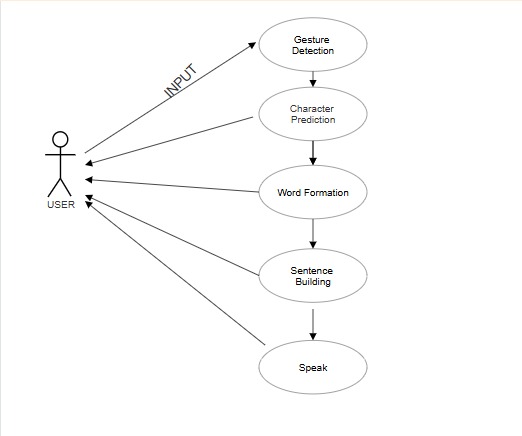


Fig. 4. Activity Diagram

## **Module Design**

Each module is designed to perform a dedicated task, contributing to a seamless flow from gesture input to text/speech output.

### **Hand Tracking Module:**

* Uses MediaPipe’s hand model for detecting 21 landmarks per hand.
* Handles multiple frames per second for real-time responsiveness.
* Filters out unstable detections with consistency checks.

### **Gesture Recognition Module:**

* Contains a set of if-else logic statements to determine which fingers are raised or bent.
* Matches finger positions to a set of static ASL alphabet rules (A–Z).
* Outputs one character per stable gesture.

### **Sentence Builder Module:**

* Stores and appends recognized letters to form sentences.
* Allows operations such as adding space, clearing the buffer, or deleting the last character.

### **Text-to-Speech (TTS) Module:**

* Uses pyttsx3 to convert text into spoken output.
* Supports multiple voices and adjustable speech rate.
* Functions offline and responds immediately.

### **User Interface Module:**

* Built with Tkinter to offer a minimal and interactive GUI.
* Includes buttons:
* Predict: Displays recognized letter
* Clear: Erases all text
* Delete: Removes last character
* Speak: Converts text to speech
* Space: Adds a space character

# **Implementation of Modules**

The implementation phase brings together all the modules designed in the methodology stage and integrates them into a functional, real-time application. Each component was developed and tested individually before being combined into a single, efficient pipeline that processes hand gestures, recognizes letters, forms sentences, and provides text-to-speech output. The implementation focuses on accuracy, responsiveness, and user accessibility.

## **Technologies**

* **Python 3.7+:**Chosen for its simplicity, robust library support, and strong developer community. Python enables seamless integration of computer vision, GUI, and text-to-speech features within a single application.
* **OpenCV (cv2):**Handles real-time video streaming from the webcam, displays live gesture feeds, and overlays visual elements such as bounding boxes, landmarks, and text for enhanced user interaction.
* **MediaPipe:**A ready-to-use framework by Google for detecting 21 hand landmarks, including fingertips, joints, and the palm center. It enables real-time hand tracking without requiring custom training or datasets.
* **pyttsx3 (Text-to-Speech Engine):**Provides offline
* speech synthesis with support for multiple voice options. Users can adjust speech rate and volume, enabling smooth and responsive spoken output for enhanced communication.
* **Tkinter (GUI Toolkit):**Used to build a lightweight and responsive graphical interface. The GUI includes buttons for Predict, Delete, Clear, Space, and Speak, along with real-time display of recognized characters and complete sentences.
* math & time Libraries:Used to calculate distances and angles between hand landmarks, aiding in the rule-based classification of ASL letters. The time.sleep() function helps prevent repeated detection of the same gesture by introducing a short delay between frame evaluations.

# **Results and Discussions**

The system was tested in various lighting conditions and user scenarios to evaluate its real-time performance, accuracy, and usability. The goal was to ensure that the system provides an effective and user-friendly experience for recognizing ASL gestures and delivering both text and speech output.

## **Simple Results**

* Gesture Recognition accuracy: The system achieved an average recognition accuracy of around 92% for static ASL alphabet gestures. Accuracy was highest in good lighting conditions with consistent hand placement.
* Response Time: The system responds in real time, typically identifying and displaying the gesture within 500–700 milliseconds of stabilization. This ensures a smooth user experience without noticeable lag.
* Offline Performance: Since the system does not require internet access, all functionalities (gesture recognition, sentence formation, and text-to-speech) work flawlessly in offline mode, even on systems with basic specifications.
* Compatibility: The application runs on standard desktops or laptops with a webcam and supports cross-platform functionality (Windows/Linux).

## **What the System Shows to Users**

* Live Camera Feed: Displays the user’s hand in real time, overlaid with hand landmarks for visual confirmation.
* Current Gesture Recognition: Shows the most recently recognized ASL alphabet on screen (e.g., A, B, C...).
* Sentence Buffer Display: As users show each gesture, the corresponding letters are appended to form words and full sentences. This buffer is shown clearly on the interface.
* Control Buttons:

1. Predict – Displays the detected alphabet.
2. Space – Inserts a space between words.
3. Delete – Removes the last character from the buffer.
4. Clear – Clears the entire sentence.
5. Speak– Converts the text buffer into speech using the TTS engine.

* Audio Feedback: When the "Speak" button is clicked, the final sentence is spoken aloud using the pyttsx3 engine.

## **Limitations**

* The system supports only static ASL alphabet gestures, not dynamic signs.
* Performance may reduce under poor lighting or background clutter.
* Designed for single-hand gestures only.
* Facial expressions and emotions are not recognized.
* Gestures must be held briefly to ensure accurate detection.

## **Output**

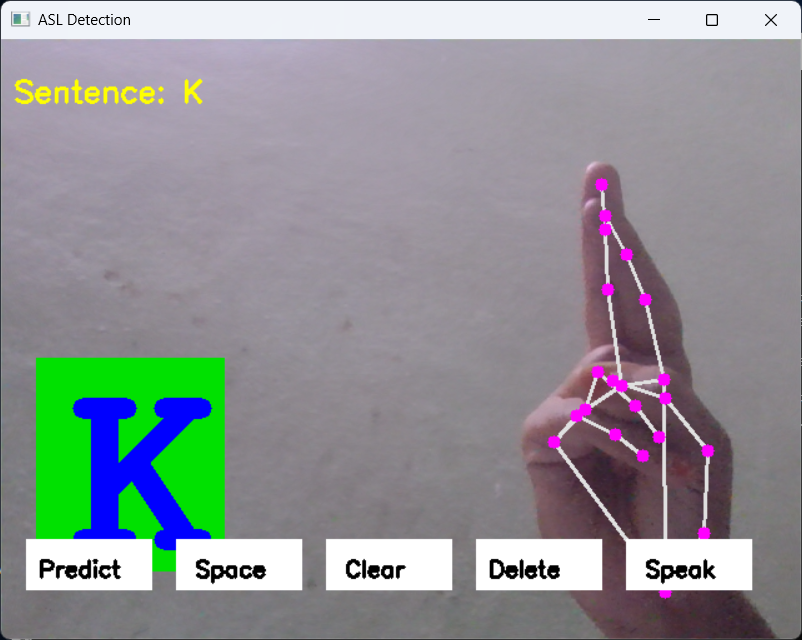


Fig. 5. Character Prediction

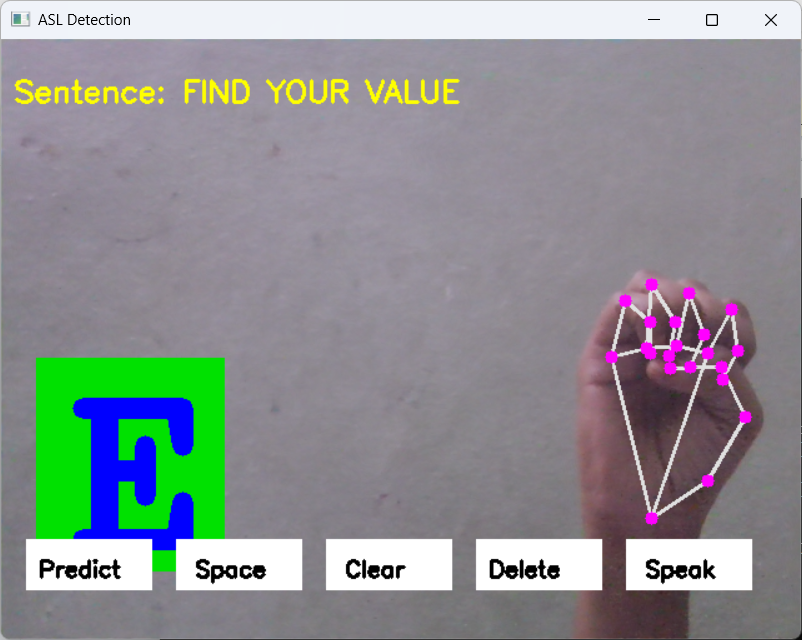


Fig. 6. Sentence Building

# **Conclusion**

## **Conclusion and Future Work**

This project presents a real-time, offline system for converting American Sign Language (ASL) gestures into text and speech using only a standard webcam and open-source tools. By combining MediaPipe for hand landmark detection and a rule-based logic system for gesture classification, the solution effectively bridges the communication gap for individuals with hearing or speech impairments. The application is low-cost, easy to use, and platform-independent, making it accessible for educational institutions, public service environments, and rural areas. With an average recognition accuracy of around 92%, the system demonstrates promising results in practical, real-world scenarios without relying on complex machine learning models or internet access.

## **Future Enhancements**

* Support for Dynamic Gestures: Implementing models like LSTM or CNN to recognize motion-based signs and full-word gestures.
* Multilingual Sign Language: Extending support beyond ASL to include regional languages such as Indian Sign Language (ISL).
* Mobile/Web Integration: Developing Android or web-based versions for portability and wider deployment.
* Facial Expression Analysis: Integrating computer vision techniques to interpret emotional context through facial expressions.
* Gesture Learning Mode: Adding a guided learning module to help users practice and learn ASL signs interactively.

# **References**

1. D. G. Kadam, P. V. Kale, “A Survey on Hand Gesture Recognition Techniques and Applications,” *International Journal of Advanced Research in Computer Science*, vol. 10, no. 6, pp. 45–50, 2019.
2. S. Starner, J. Weaver, A. Pentland, “Real-time American Sign Language Recognition using Convolutional Neural Networks,” *IEEE International Symposium on Computer Vision*, 1998.
3. A. Kumar, M. Sharma, “Sign Language Recognition using Deep Learning on Custom Dataset,” *International Journal of Computer Applications*, vol. 176, no. 17, pp. 10–15, 2020.
4. A. Rautaray, A. Agrawal, “Vision-Based Real-Time Sign Language Recognition Using Support Vector Machine,” *International Journal of Computer Vision and Image Processing*, vol. 5, no. 1, pp. 33–43, 2015.
5. H. Fang, S. Wang, X. Zhao, “Real-Time Hand Gesture Recognition for Human-Computer Interaction,” *IEEE Transactions on Multimedia*, vol. 21, no. 4, pp. 1011–1022, 2019.
6. L. Molchanov, S. Gupta, K. Kim, J. Kautz, “Dynamic Hand Gesture Recognition using LSTM Networks,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
7. M. T. Hasan, A. Mollah, “Real-Time Static Hand Gesture Recognition Using Convolutional Neural Networks,” *Proceedings of IEEE ICCCNT*, 2021.
8. R. Kaur, D. Sharma, “Sign Language Recognition Using Deep Learning: A Review,” *International Journal of Engineering and Technology*, vol. 8, no. 2, pp. 65–72, 2022.