International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Integrating Indian Sign Language Recognition with Real-Time Speech Synthesis for video conferences

Biswajit Dey [1] **, Rajdeep paul** [2]

[1]*Department of Computer Application, Techno India University, Kolkata, India*
[2]*Department of Computer Application, Techno India University, Kolkata, India*

---***---

Abstract - Both hearing and deaf people commonly face major communication hurdles in their daily lives. To solve, this study presents a real-time video calling system that uses ai model to recognize Indian Sign Language (ISL). Peers are connected via WebSockets, and video data is shared with the AI model for identification. Our approach captures 30 frames a second and buffers them as groups of 3 seconds that a backend AI model interprets. The application using grid fragmentation-based splitting and k-NN prediction detects the hand movements very accurately and translates these movements to textual equivalents that other peers uses. In the meantime, regular video and audio communication proceeds uninterrupted, providing a natural conversation experience. The system was tested with 15 normal ISL sentences with an average accuracy of 97.2%. Performance measurement is reflected in a frame rate of 5.3 FPS, maximizing real-time interaction and processing efficiency. Future development will be in improving gesture prediction, increasing the dataset, and implementing speech-to-text capabilities to increase accessibility further. By integrating AI-powered sign language recognition with video calling, the app acts as a bridge between the deaf and hearing communities and fosters inclusivity and accessibility in digital communication.

1. INTRODUCTION

Indian Sign Language (ISL) is a visual language used by people with hearing and speech impairments to communicate with others through hand signs and gestures. Sign language is the mixture of facial expressions, hand gestures and body posture. It portrays the inner feelings of the person just like any other spoken language. There is a lack of research on ISL that addresses the challenges of these individuals.

The challenge is to close this communication gap. Despite specialised training in ISL (Indian Sign Language) can enable individuals to interact with the hearing- and speech-impaired, it is unrealistic to anticipate that the general populace will learn sign language. Thus, developing systems that can convert sign language into spoken language has emerged as a key area of study.

Existing research is mostly focused on popular sign languages like ASL (American Sign Language), Chinese Sign Language, etc. However, because of its complexity, there is less research on Indian Sign Language. This paper proposes to fill this gap by presenting a method for real-time ISL identification and its integration into video interactions to improve remote communication.

Given the complexity of ISL, our initial focus is on recognising numbers and alphabets and using those alphabets to form complete sentences with meaning. There exist 33 unique hand gestures (10 for digits and 23 for letters), as displayed in Fig. 1. This paper examines the use of machine learning and natural language processing (NLP) to convert gestures into structured text and subsequently into spoken language. It also addresses potential obstacles in the process, including latency and preserving of grammatical sequence for precise translation.

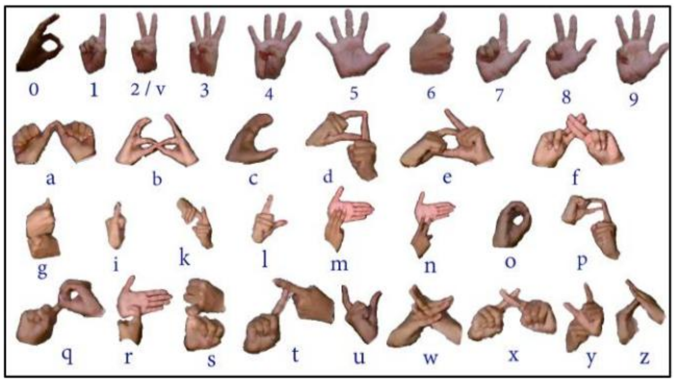

Fig. 1: Hand Gestures in ISL (Indian Sign Language)

In the following sections, we dive into the technical components of this system, including gesture detection algorithms, text-to-speech integration, and video call infrastructure. We also discuss the challenges encountered in the development process and propose solutions for overcoming them. This research focuses on developing a system capable of recognising Indian Sign Language in a video call and transcribing it into voice in real-time.

2. RESEARCH MOTIVE

The goal of this research is to reduce the communication gap between the deaf and mute communities and the rest of society by integrating sign language recognition with video calls. While many existing studies focus only on recognizing signs and converting them into text or voice, they do not address virtual communication systems. In the contemporary world, where the majority of communication occurs online, a viable solution for deaf and mute individuals to engage in such contexts is absent. Our objective is to develop a hybrid model that recognises signs and gestures in a video call and then processes them through an algorithm that maintains the sequence using NLP (Natural Language Processing). Moreover, the proposed approach enables individuals with hearing and speech problems to fully participate in society and communicate effectively, ultimately promoting equal opportunities and accessibility for all.

3. LITERATURE REVIEW

Some of the recent literature that can help us to build the proposed system is discussed below.

[1]C. K. M. Lee and K. K. H. Ng proposed an RNN identification and training methodology for ASL research. The study shows how successfully a Long-Short Term Memory (LSTM) Recurrent Neural Network and the k-Nearest-Neighbor (KNN) approach classify ASL alphabets. With an accuracy of 99.44% in identifying ASL alphabets utilising characteristics like sphere radius, finger placements, and angles between fingers, the model showed the potential of real-time sign language applications. However, the primary focus of this research is on instructional applications rather than practical video call communication technologies.

[2]Kothadiya et al. introduced a deep learning-based model for Indian Sign Language (ISL) recognition using LSTM (Long-Short Term Memory) and GRU (Gated Recurrent Unit) networks. By employing four different sequential combinations of LSTM and GRU layers, the proposed model attained an accuracy of 97% on 11 different ISL (Indian Sign Language) signs using the IISL2020 dataset. This work emphasizes the significance of deep learning models in improving sign language detection but does not address real-time communication challenges in virtual environments.

[3]After analyzing the challenges of real-time sign language identification, Arun Prasath and Annapurani proposed an end-to-end multi-layer convolutional neural network (ML-CNN) solution. The purpose of this research is to utilize the Indian Sign Language database and perform the operation that converts signs to spoken output. The proposed ML-CNN model outperformed more traditional techniques like BLSTM and HMM with a prediction accuracy of 87.5%.

[4]Asari et al.. developed a hybrid-based LSTM-CNN model for the detection of critical signs. Based on their research, a VGG-19 + LSTM architecture can perform better than BLSTM-based systems in terms of accuracy, achieving 96.39%. Their research shows how successfully LSTM-based models preserve sign sequences to maintain fluency during live interactions.

[5]Sharma and Tulsian proposed an audio-to-sign language translation system for Indian Sign Language. The primary objective of this system is to generate appropriate sign language gestures by comparing the input text or audio with a database of ISL (Indian Sign Language) video gestures. They used multiple NLP techniques to increase the accuracy of the system. This research mostly focused on converting speech to sign motion.

[6]Hanmo Wang's research explores the advancements of sign language. He discussed the integration of computer vision and natural language processing to improve the result of the model when it converts videos into spoken language texts. This research underscores the necessity of combining CV and NLP techniques to improve sign language processing, aligning with the proposed approach of integrating neural networks and NLP for real-time communication in video calls.

[7]A recent study by Abdullah Baihan and Sunil Kumar Sharma focused on addressing the challenges of static and dynamic sign language recognition (SLR) using a modified deep learning and hybrid optimization approach. The study highlights the difficulty in creating a high-accuracy model that can recognize continuous signs independent of the signer due to variations in speed and duration. To overcome these issues, the proposed CNNSa-LSTM model integrates Convolutional Neural Networks (CNN) for spatial analysis, Self-Attention (SA) mechanisms for focusing on relevant features, and Long-Short-Term Memory (LSTM) to model temporal dependencies effectively. Motion features are extracted using the optical flow approach, while spatial and geometric information is gathered using the Visual Geometry Group 16 (VGG16). To improve performance, a Hybrid Optimizer (HO) combines the Pathfinder Algorithm (PFA) with the Hippopotamus Optimization Algorithm (HOA). The proposed model's implementation outperforms existing methods with high precision (98.5%), sensitivity (98.2%), and accuracy (98.7%).

[8]A real-time Indian Sign Language (ISL) recognition system using grid-based features has been developed to enhance communication for hearing and speech-impaired individuals. Unlike existing methods that lack real-time performance or accuracy, this system effectively identifies 33 hand poses and 12 gestures using a smartphone camera, eliminating the need for external hardware. It uses Face Detection, Object Stabilization, and Skin Color Segmentation to track hands, while Hidden Markov Models (HMM) identify gesture sequences and k-Nearest Neighbors (k-NN) classify hand positions. The system demonstrates its effectiveness and dependability in real-time applications with an accuracy of 99.7% for static postures and 97.23% for motions.

4. IMPLEMENTATION

The term **sign language recognition in video calls** may seem complex, but breaking it down into distinct components makes it more feasible for implementation. There are three main components required to develop a system that enables sign recognition over video calls and generates the desired output. These components include implementing video calls, processing video frames through a pre-trained AI model, and passing the output through an NLP-based model to reconstruct the data into a communicable format. Therefore, the first step is to gather input data from video calls. We begin by focusing on the video call system's implementation, latency optimisation, and guaranteeing smooth network connectivity between various devices.

4.1 Implementation of Video call

A low-latency video calling system is as crucial as the AI model for real-time sign language recognition. The system needs to capture video frames efficiently, send them with low latency, and provide stable connectivity under various network conditions. This is facilitated by WebRTC, which provides peer-to-peer (P2P) video communication with adaptive networking protocols and real-time optimization. Here is a brief explanation of how video calls operate, which can also help in more successfully implementing the AI model into video chats.

4.1.1 Network Protocols for Real-Time Video Calls

Reliable and efficient video transmission in real-time communication systems necessitates the use of specialized network protocols. In order to ensure synchronized, low-latency, and secure video chat transmission, the following protocols are essential:

1. **Real-Time Transport Protocol (RTP):** Enables the delivery of audio and video information over IP networks, including timestamping and sequence numbering to ensure synchronization and reduce packet loss.

2. **Datagram Transport Layer Security (DTLS):** Provides secure key exchange for WebRTC-based communications, providing media integrity and eavesdropping protection.

3. **Interactive Connectivity Establishment (ICE):** Assists in NAT traversal by selecting the most efficient network path between peers, improving connectivity in varying network conditions.

These protocols combined optimize the performance of WebRTC-based video calls to deliver adaptive bitrate streaming, secure transport, and low latency, which are essential for real-time sign language recognition via video calls.

4.1.2 System Architecture for Video Transmission

The video-calling system adopts a three-layer architecture to capture, send, and render video streams.

1. The **Capture & Encoding Layer** takes the task of capturing and optimizing the video frames to be sent out. The video is captured on the user's device at 30 FPS through the use of React Native Vision Camera to provide stable input to the subsequent processing. In case to reduce the system's bandwidth usage, we can consider compression techniques like VP9 or H.265 (HEVC), which are used to compress video frames while maintaining quality.

2. The **Transmission Layer** handles real-time data transfer and network optimization. WebRTC, in combination with ICE (Interactive Connectivity Establishment), establishes the most efficient transmission route between devices. It uses STUN/TURN servers to support NAT traversal and relay traffic where peer-to-peer communication is not possible directly. Media streams are secured using SRTP (Secure Real-Time Transport Protocol) for secure transmission support, and DTLS (Datagram Transport Layer Security) supports secure key exchange to avoid eavesdropping of data and unauthorized access.

3. The **Receiving and Processing Layer** decodes video streams and prepares them for AI analysis. It reduces latency and guarantees correct frame sequencing for seamless processing. The AI model analyzes extracted frames in real time, identifying motions and translating them into speech or text. Organizing the process in this manner makes video calls smooth without compromising the quality of visuals required for precise sign language interpretation.

4.1.3 WebRTC: Internal Core Components

WebRTC is a real-time communication framework that enables P2P video calling with minimal latency. The process involves:

Devices communicate through the Session Description Protocol (SDP) to agree on connection parameters after they capture and compress video frames. This is succeeded by the Interactive Connectivity Establishment (ICE) protocol through STUN/TURN servers to find the optimal transmission path. Datagram Transport Layer Security (DTLS) provides safe key exchange between two devices, while safe Real-Time Transport Protocol (SRTP) encrypts media streams once the connection is made.

4.1.4 ICE Servers & Connection Establishment

To determine the most efficient way of connecting users, WebRTC makes use of ICE (Interactive Connectivity Establishment). It utilizes:

1. **STUN (Session Traversal Utilities for NAT)**
   - Helps devices discover their public IP for a direct connection.
   - Used when both users are on open networks.

2. **TURN (Traversal Using Relays around NAT)**
   - Serves as a relay server when a direct connection is impossible.
   - Used when one or both users are behind firewalls or restrictive NATs.

3. **Direct P2P Connection**
   - Wherever available, WebRTC allows a direct link between users, avoiding relay servers for reduced latency.

4.1.5 Low-Latency Video Call Optimizations

For enhancing performance and stability, the following optimizations are utilized:

1. **Adaptive Bitrate Streaming (ABR)**
   - Dynamically adjusts video quality based on network conditions.
   - Prevents buffering or lag by scaling between 720p, 480p, and 360p.

2. **Congestion Control (Google BBR Algorithm)**
   - Dynamically manages bandwidth to avoid packet loss.

The formula for congestion control:

$$Cwnd = min(Cwnd_{max}, Bandwith \times RTT_{min})$$

In Fig. 2, the WebRTC architecture illustrates the process of video call operations, detailing the key components and their interactions.

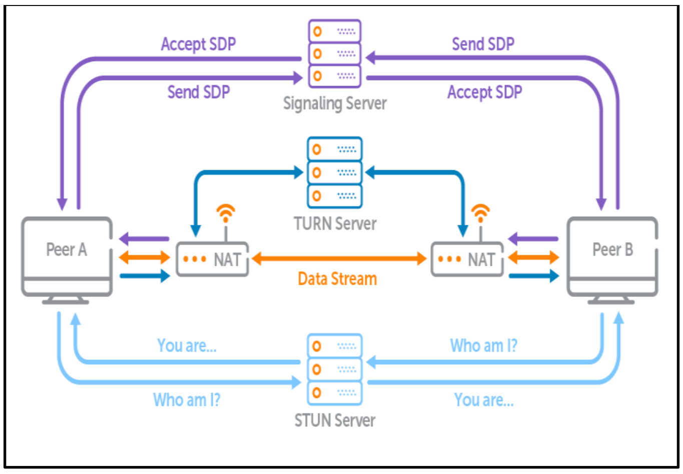

Fig. 2: WebRTC Architecture and Workflow

Now we are moving forward to our second core component which is our AI model that recognises the frames of gestures and converts them to text messages.

4.2 Implementation of the AI Model

From the Video call media stream, the gestures and signs performed by an individual using Indian Sign Language (ISL) are extracted, and their frames are transmitted to a server for processing. Before these frames can be used for recognizing gestures and hand poses, they must undergo pre-processing. Face removal, stabilization, and skin color segmentation—which removes background details—begin this procedure. Then, morphological operations are applied to minimize noise. The system extracts and tracks the person's hand in each frame.

For hand pose recognition, relevant features are extracted from the hand and input into a classifier. The identified hand pose class is then transmitted back to the Android device. When classifying hand gestures, intermediate hand poses are recognized, and a pattern is established using these recognized poses and their sequential motion. This pattern is represented in tuples and subsequently encoded for Hidden Markov Model (HMM) processing. The HMM chain that achieves the highest score using the forward-backward algorithm determines the recognized gesture for the given pattern. An overview of this process is illustrated in Fig. 3.

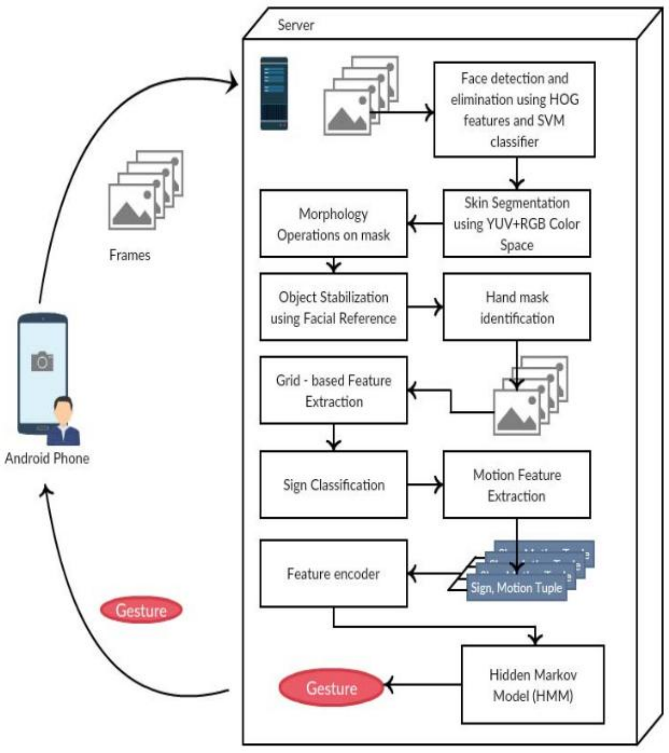
Server
Face detection and elimination using HOG features and SVM classifier
Frames
Skin Segmentation using YUV+RGB Color Space
Morphology Operations on mask
Object Stabilization using Facial Reference
Hand mask identification
Grid - based Feature Extraction
Android Phone
Sign Classification
Motion Feature Extraction
Feature encoder
Sign, Motion Tuple
Gesture
Gesture
Hidden Markov Model (HMM)

Fig. 3: Flow diagram for Gesture Recognition.

4.2.1 Dataset used

For the digits 0 to 9 in ISL, an average of 1000 images per digit was captured. For ISL letters, around 200 images were taken for each letter, making up a total of 15,200 images. Most of these were captured using smartphone cameras, while a smaller number came from webcams. The images vary in resolution.

For training HMMs, 50 gesture videos were captured for each of the 15 one-handed pre-selected gestures (Are you Free Today, Can you repeat that please, Congratulations, Help Me Please, I am fine, I love you, Please come, Welcome, Talk slower please, Thank You, What Happened, What are you doing, What do you do, how are you, no, yes). Each video consists of 50 sets, where some sets use the left hand and others use the right hand.

4.2.2 Pre-processing

1. **Face detection and elimination**

In Indian Sign Language (ISL), hand poses and gestures are primarily represented by specific hand movements, making facial features unnecessary. Additionally, the presence of the face can make hand extraction more challenging. To solve this, face detection was carried out using Histogram of Oriented Gradients (HOG) descriptors [8], followed by a linear Support Vector Machine (SVM) classifier. This approach utilizes an image pyramid and a sliding window technique to detect faces in an image accurately, as described in [9].

By combining HOG feature extraction with a linear classifier, the false positive rate is reduced by more than an order of magnitude compared to the most efficient Haar wavelet-based detector [9]. Once the face is detected, the contour of the face region is identified, and the entire face-neck area is blacked out, as illustrated in Fig. 4.

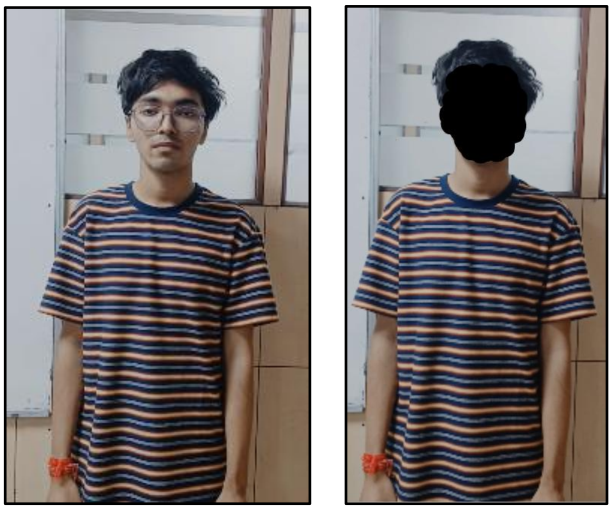
Fig. 4: Face detection and elimination operation

2. Skin colour segmentation

To detect skin-like regions within an image, skin colour segmentation is performed using the YUV and RGB colour models, which yield highly accurate results [8]. This model was selected as it outperformed other colour spaces, including HSV, YCbCr, RGB, YIQ, YUV, and various combinations of these [10]. The frame is first converted from RGB to YUV colour space using the transformation equation provided in [11], as specified in equation (1).

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} +0.299 & +0.587 & +0.114 \\ -0.147 & -0.289 & +0.436 \\ +0.615 & -0.515 & -0.100 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} \quad (1)$$

The resulting segmentation mask effectively minimizes noise and reduces false positive detections. A visual representation of the segmentation mask is provided in Fig. 5.

3. Morphology operations

Morphological operations were applied to eliminate any noise produced during the skin colour segmentation process. There are two types of errors in skin colour segmentation:

1. False positives – Non-skin pixels mistakenly classified as skin.
2. False negatives – Skin pixels incorrectly classified as non-skin.

Morphology involves 2 basic sub-operations:

1. Erosion – Reduces the size of active regions (white areas) in the mask
2. Dilation – Expands the size of active regions (white areas) in the mask

When non-skin regions are mistakenly picked up as skin, we use a technique called Morphological Opening. It starts by "eroding" away the small, unwanted areas and then "dilates" to restore the important details. To correct false negatives (when actual skin areas are missed), Morphological Closing is applied, doing dilation first, then erosion. The results of these operations can be seen in Fig. 5.

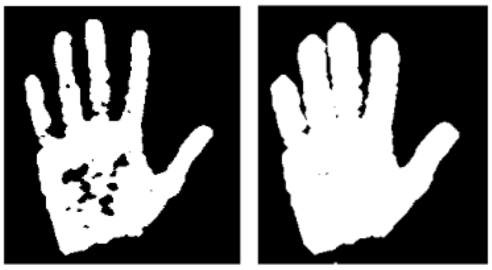
Fig. 5: Skin segmentation mask and effect of morphology operations. (Left) Segmentation mask; (Right) Mask after application of Morphology operations.

4. Object Alignment Using Facial Landmarks

To accurately track hand motion, maintaining a stable camera position is essential. However, hand tremors or unintentional movements by the person recording the video can introduce false motion detections. This issue is addressed through object stabilization.

Assuming that the sign demonstrator's face is consistently present in the gesture video, facial tracking is used to stabilize hand movements. The tracker is initialized using coordinates obtained from the face detection process before the face is removed. It then identifies the facial region and compensates for any detected motion by shifting the entire frame in the opposite direction of the face's movement.

For tracking, the system employs the Kernelized Correlation Filter (KCF) tracker, implemented in the OpenCV library, to follow the face in each frame. This tracking operation is applied before the face is blacked out.

4.2.3 Hand extraction and tracking

Since ISL hand poses and gestures rely entirely on hand movements, detecting and tracking the hand is a vital part of the system. After pre-processing each frame, a black-and-white image is generated, where white areas represent skin. Facial features are excluded, leaving only the hand and other skin-like regions from the original image.

In each frame, either one hand is visible or both hands are touching, ensuring that the most prominent contour belongs to the hand. To identify it, the system calculates the area of all contours in the frame and selects the largest one. Since the most dominant contour represents the hand, this extracted contour defines the hand region.

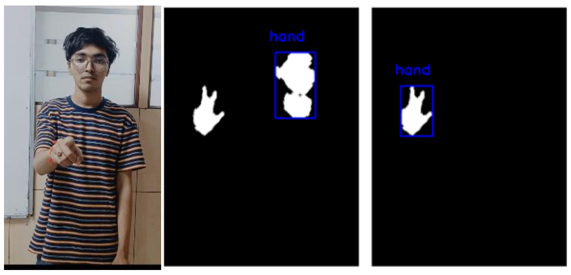

Fig. 6: Importance of eliminating face before, hand extraction.

Fig. 6 demonstrates why removing the face is essential in this process. If the face were not eliminated, it would likely be the largest contour detected, leading to misclassification as a hand, as shown in the figure.

For tracking hand motion, the centroid of the hand is calculated in each frame. If there is movement of the hand, the coordinates of the centroid of the hand will change. The Slope of the line formed by the centroid of the hand in the current frame and the centroid of the hand in the previous frame is then determined. Based on the slope value, movement is classified as follows:

- If -1 < slope < 1 and the x-coordinate increases, the hand moves left.
- If -1 < slope < 1 and the x-coordinate decreases, the hand moves right.
- If |slope| > 1 and the y-coordinate increases, the hand moves up.
- If |slope| > 1 and the y-coordinate decreases, the hand moves down.

Fig. 7 illustrates this slope-based motion tracking. It is important to note that the motion seen by the camera appears opposite to the actual movement performed by the sign demonstrator. The system leverages the OpenCV library to compute the contour area and track the centroid of the hand.

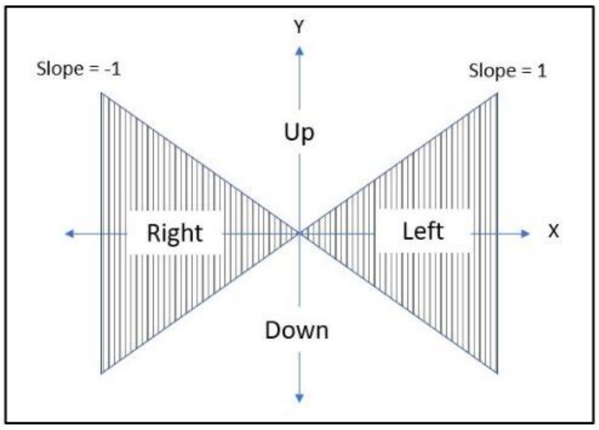

Fig. 7: Determining hand motion using slope.

To minimize tracking noise, the system places an imaginary circle with a 20-pixel radius around the previous hand centroid. If the new centroid is within this radius, the shift is treated as noise, and the movement is disregarded. In this case, the previous centroid stays the same for comparison.

However, if the new centroid moves beyond the 20-pixel threshold, the shift is recognized as actual hand movement. When movement is detected, the radius is set to 7 pixels instead of 20 pixels until there is no movement, after which the radius is restored to 20 pixels. This use of an imaginary circle reduces noise to a greater extent and gives highly accurate tracking of hand movements. As discussed in [8].

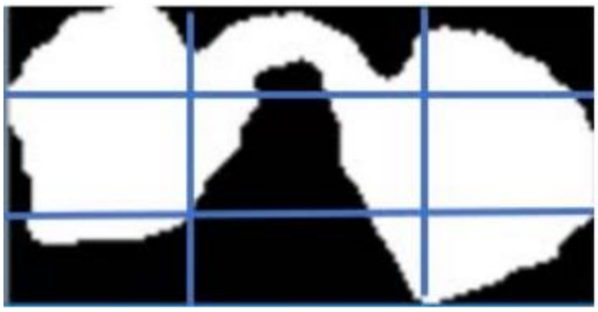

Fig. 8: The hand pose ‘a’ in ISL fragmented by a 3x3 grid.

4.2.4 Feature Extraction Through Grid-Based Segmentation Method

In this method, based on the approach described in [8], the extracted hand sample is segmented into an M × N grid, resulting in M × N smaller regions. A feature value is computed for each region, collectively forming a feature vector consisting of M × N elements. The feature value for each section is calculated based on the proportion of the hand contour it contains, as described in Equation (2).

$$\textit{Feature value} = \frac{\textit{Area of hand contour}}{\textit{Area of fragment}} \tag{2}$$

If no hand contour is detected in a block, the feature value is assigned as 0. For illustration, Figure 8 shows a 3 × 3 grid applied to a sample. Because each position covers a different number of grid sections and fragmented areas, this technique is highly adaptable to various hand orientations. This allows the feature vector to accurately capture both the shape and position of the hand.

To get a clearer picture of the extracted features, we applied Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE), following a similar approach to [8]. We began with a 10 × 10 grid, creating 100 features per sample. First, PCA narrowed the features down to 40, and then t-SNE shrank the data into two dimensions, making it easier to see and interpret.

4.2.5 Classification of Gestures

4.2.5.1 Identifying ISL hand poses using the k-NN algorithm

When we were processing the data, we observed that the data had a tendency to group naturally, and some of the hand poses had occurred more than once in the groups. To classify such data efficiently, we required an algorithm that was capable of dealing with these kinds of patterns. K-Nearest Neighbors (k-NN) suited the best as it performs efficiently with grouped data.

For each frame in the live feed, the extracted hand undergoes feature extraction using the previously mentioned grid-based fragmentation technique. This process creates an M × N dimensional feature map. To classify a sample, we employ Euclidean distance to match the sample against stored data and select the k closest matches from the training model.

A simple brute-force method for calculating the distance is to measure the Euclidean distance between the sample and each stored sample and then select the k closest. However, for larger datasets, more efficient methods like KD-tree and Ball Tree can help speed up the process. Brute force works well for small datasets, KD-tree is effective for low-dimensional data, and Ball Tree performs best for high-dimensional data [12].

Finally, the classifier determines the sample's class based on the most frequently occurring class among its k nearest neighbors.

4.2.5.2 Gesture Classification using HMM

Gestures often exhibit slight variations, even when performed by the same individual. To accommodate these inconsistencies, a statistical model is required. Hidden Markov Models (HMMs) serve as an effective statistical approach for managing such variations [15]. HMMs are classified into two types: continuous and discrete. In a continuous HMM, the number of potential observation symbols in each component of the observation sequence is unlimited, while in a discrete HMM, it is finite.

In addition, HMMs can be designed as ergodic or left-to-right. In a left-to-right HMM, the transitions are in one direction only, i.e., once the model moves to the next state, it cannot revert to a previous state, as shown in Fig. 10. Conversely, an ergodic HMM allows transitions between any state. The initial state probabilities (π) and transition probabilities associated with the left-to-right HMM are illustrated in Fig. 9.

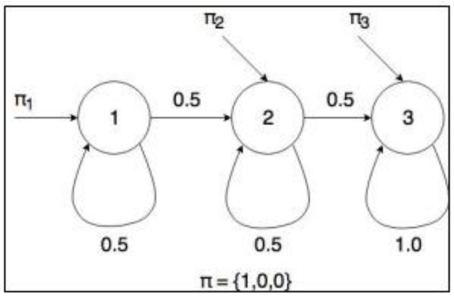

Fig. 9: HMM chain for gesture with 3 hidden states (E.g. Good Afternoon).

The human brain interprets gestures as a sequence of intermediate hand poses combined with specific hand movements arranged in a defined order. Following this concept, ISL gestures are composed of stationary intermediate hand poses combined with the transitional movements linking them. As a result, this system implements a discrete left-to-right HMM, utilizing segmented hand centroid motion and pose classification results to identify the provided observation sequence as one of the 15 predefined gestures.

The input for this HMM comes from an observation sequence extracted from the video feed. The total number of possible observation symbols is determined by the sum of the tracked movement directions and the intermediate stationary hand poses used during training. In this system, four-movement directions are tracked, and the model is trained with nine intermediate stationary hand poses, including 'Thumbs Up,' 'Sign Me,' and 'Fist,' as illustrated in Fig. 10.

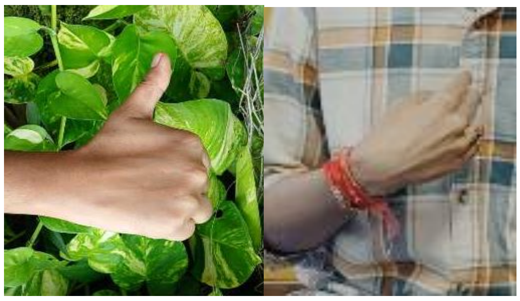

Fig. 10: Thumbs_Up and Sign_Me stationery hand poses.

The recognition of intermediate hand poses also employs a grid-based feature extraction method. This process is similar to identifying hand poses for letters and digits but is only performed when no movement is detected. As a result, there are a total of 13 possible observation symbols, meaning the observation sequence can contain values ranging from 0 to 12.

At each frame, a tuple is generated in the format <S, M>, where M represents the hand's motion relative to the previous frame, and S denotes the classified hand pose if no movement is detected. When motion is detected, it is assigned a corresponding observation symbol—'Upwards' (0), 'Rightwards' (1), 'Leftwards' (2), and 'Downwards' (3). If no motion is present, the detected hand pose is mapped accordingly based on a predefined mapping system.

Thus, each frame contributes an observation symbol, and the entire video sequence forms an observation sequence that encodes both motion and hand pose data. For example, the time series [<Sign_Me, None>, <Sign_Me, None>, <Sign_Me, None>, <None, Up>, <None, Up>, <Sign_Fine, Up>, <None, Up>, <Sign_Thumbs_Up, None>, <Sign_Thumbs_Up, None>, <Sign_Thumbs_Up, None>] corresponds to the gesture "I Am Fine" When converted into an observation sequence, it results in [4, 4, 4, 0, 0, 0, 5, 5, 5]. A visual representation of this gesture is provided in Fig. 11.

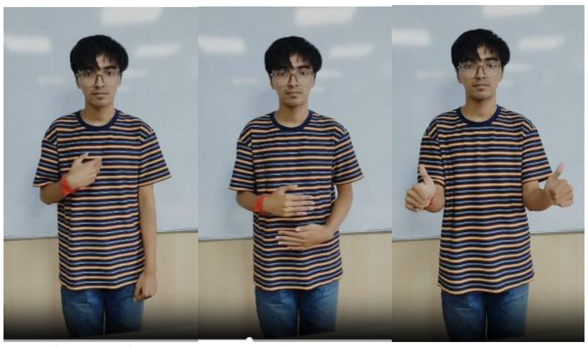

Fig. 11: 3 frames of Gesture "I Am Fine".

The gesture recognition in this model makes use of 12 HMM chains, where each chain is used for a particular gesture. The number of hidden states per chain is derived from breaking down the gesture into a sequence of hand poses and the transition between them. For instance, as illustrated in Fig. 11, the "I Am Fine" gesture consists of three states: the 'sign_me' hand pose, the 'sign_fine' with 'Upwards' motion, and the 'sign_thumbs_up' hand pose.

Since all these HMM chains follow a left-to-right structure, their initial state probabilities and transition probabilities are similar to those depicted in Fig. 9. For an HMM with n hidden states, the state transition probability matrix is of size n × n, and the emission probability matrix is of order n × 13. The emission probability matrices were initially fixed according to empirical probabilities, which were calculated by examining the similarity between various hand poses and their nearest corresponding motions. This is done to enhance the chances of the HMM model converging to the global maximum during training. This method enhances the likelihood of the HMM model successfully converging to the global maximum during training.

Once all parameters are initialized as outlined in [13], the estimation and transition probabilities for the HMM chains are trained using the Baum-Welch algorithm [13, 14]. Training is conducted using a gesture database, the details of which are provided in Section III. After training, a new observation sequence is fed into the HMM chains, and the chain that produces the highest score using the forward-backward algorithm [13] is identified as the recognized gesture.

4.2.5.3 Temporal Segmentation

The gesture recognition module needs video segments with only the target gesture. Without temporal segmentation, it is impossible to recognize continuous gestures. To solve this, a simple rule is used: if the hand goes out of the frame, it indicates the end of the ongoing gesture, and recognition is done based on the frame sequence captured. When the hand comes back into the frame, it indicates the start of a new gesture. This rule successfully performs temporal segmentation.

4.2.6 EXPERIMENTAL RESULTS

The results shown in this section were obtained using a personal computer with 16GB RAM, an NVIDIA GTX 1650 GPU and 4GB of VRAM, and an AMD Ryzen 5 processor. The operating system was Fedora Linux, and all implementations were carried out using Python. Image processing and classification were done using OpenCV, NumPy, and scikit-learn.

The dataset employed in this study comprised 15 commonly used ISL sentences, including:
"Are you free today?", "Can you repeat that, please?", "Congratulations", "Help me, please", "I am fine", "I love you", "Please come, Welcome", "Talk slower, please", "Thank you", "What happened?", "What are you doing?", "What do you do?", "How are you?", "No" and "Yes".

For effective hand pose classification, an appropriate grid size had to be determined for feature extraction. We experimented with six different grid sizes—5×5, 10×10, 10×15, 15×15, 15×20, and 20×20—to extract features from the training data. These attributes, which had been extracted, were then applied to train the k-NN classifier. Identical grid dimensions were applied to the test data, and classification correctness was quantified by comparing extracted features with the trained k-NN model.

$$Accuracy = \frac{Number\ of\ samples\ classified\ correctly}{Total\ number\ of\ samples} \quad (3)$$

An optimal grid dimension should form distinctive clusters of hand poses so that the k-NN classifier would be able to identify them accurately. The accuracy of the classifier was obtained by using Equation (3), and the results of the various grid sizes were compared in Figure 12. From the findings, the best accuracy of 99.71% was recorded by the 10×10 grid and hence was chosen to perform further feature extraction. The average time required to extract features from an image of size 300×300 using a 10×10 grid was approximately 1 millisecond.

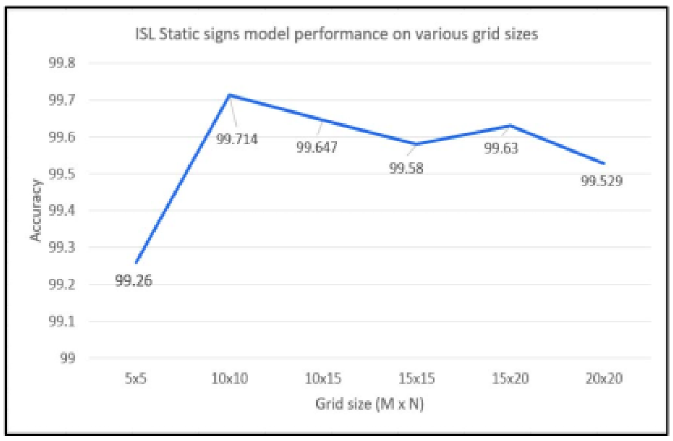

Fig. 12: Comparison of accuracy of k-NN classification on features extracted using various grid sizes on hand poses' data.

The testing dataset was 30% of the total dataset, and classification performance was analyzed using a confusion matrix, shown in Figure 13. The confusion matrix indicates that the model accurately distinguished between different hand poses. Table I presents the time taken per frame for each phase of the system. Based on these timings, our application achieves a frame rate of approximately 5.3 FPS.

Sr. No	Phase	Average Time per Frame (ms)
1	Data Transfer over WLAN	46.2
2	Skin Color Segmentation and Morphological Operations	10.2
3	Face Detection and Elimination	90.8
4	Object Stabilization	12.8
5	Feature Extraction	7.2
6	Hand Pose Classification	1.9
Average Time per Frame		169.1

Once the gesture frames were processed, the time series data was extracted for gesture recognition. The Hidden Markov Model (HMM), consisting of 15 HMM chains (one for each sentence in the dataset), was used for classification. The average processing time for HMM-based gesture classification was 3.7 ms per sequence.

A 10×10 grid and k-NN classifier were used to recognize intermediate hand poses, similar to hand pose recognition. The system was evaluated with 50 real-time trials per sentence under good lighting conditions, with the sign demonstrator wearing a full-sleeve shirt to avoid arm skin interference. With an overall average accuracy of 97.2%, the confusion matrix obtained from these tests (Fig. 13) indicates that the correct classification rate exceeded 94%.

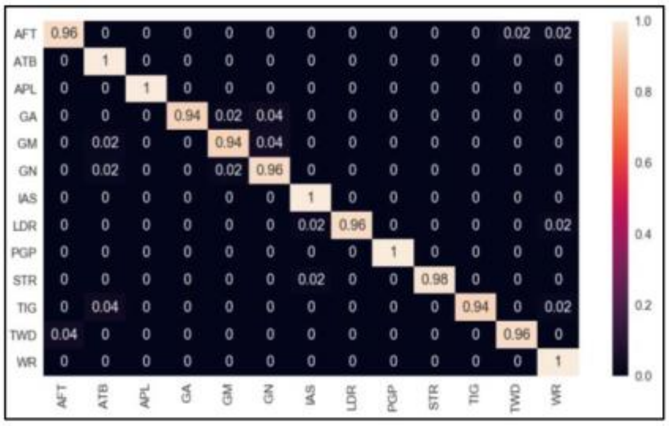

Fig. 13: Heatmap of Confusion matrix of ISL gestures.

These findings validate that the hand tracking and classification achieved a level of precision sufficient to produce reliable time-series data for sentence recognition using HMM.

5. INTEGRATING AI MODEL AND VIDEO CALL FUNCTIONALITY INTO THE APPLICATION

We have created a React Native app that harmoniously combines real-time AI-powered ISL recognition with video calling functionality. Using WebSockets, the application creates a peer-to-peer communication channel between two peers, providing a seamless and continuous video call experience.

For sign language recognition, the app takes 30 frames per second, clusters them into 3-second video chunks. The chunks are then sent to a middle-layer processor, where our AI model inspects the frames and detects the gestures. The detected gestures are then translated into text and immediately forwarded to the peer on the other end, improving mutual comprehension and communication.

While the AI recognizes the video frames in the backend, normal communication happens seamlessly between the peers. This means the audio and video continue as usual for both users, and when a gesture is recognized, it is displayed to the opposite peer who does not understand sign language.

6. FUTURE WORK

Shortly, we plan to develop the precision and performance of AI-based ISL recognition through additional advanced deep models designed for real-time operation. To enhance the gesture recognition process, we shall increase the volume of our database by including varied types of ISL gestures as well as structures of sentences for greater compatibility in varying signing forms.

The second area of importance for future development is the implementation of AI-based gesture prediction algorithms, enabling the system to predict and interpret gestures even before the completion of a complete motion pattern. This will further improve real-time interaction and decrease processing lag. We will also explore the application of edge AI computing to reduce dependence on cloud processing, such that the system becomes more efficient and privacy-focused.

Furthermore, improved low-latency support across all network conditions will make it further accessible.

7. CONCLUSIONS

This work proposes an AI-driven ISL recognition system integrated with live video calling to facilitate seamless communication between signers and non-signers. With 30 frames per second recording and segmentation into 3-second blocks, our AI model processes gestures in the background in a seamless way while providing unbroken video and audio communication. The identified gestures are translated into text-based equivalents, making conversations inclusive and accessible.

By using grid-based fragmentation and k-NN classification, we obtained a mean accuracy of 97.2%, proving the efficiency of our solution. The use of WebSocket-based communication guarantees low-latency, real-time, peer-to-peer interaction. Our system maintains a frame rate of 5.3 FPS, with a good balance of accuracy and performance for fluency in user experience.

By filling the gap between signers and non-signers, our app is an effective instrument for inclusive communication. With ongoing improvements in gesture prediction, speech integration, and multi-platform optimization, this system can potentially transform AI-powered sign language translation and render digital conversation truly barrier-free.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the **research community and open-source developers** whose contributions have been invaluable to this work. Their tools and frameworks, including **React Native, OpenCV, and NumPy**, played a crucial role in the development of this project.

I also acknowledge the work of **Kartik Shenoy, Tejas Dastane, Varun Rao, and Devendra Vyavaharkar** from the **Department of Computer Engineering, K. J. Somaiya College of Engineering, University of Mumbai**. Their research paper, **"Real-time Indian Sign Language (ISL) Recognition"**, provided key insights that helped shape my approach and methodology.

A special thanks to my **friends, peers, and family** for their constant support, encouragement, and helpful discussions throughout this process.

REFERENCES

1. C. K. M. Lee, K. K. H. Ng, C.-H. Chen, H. C. W. Lau, S. Y. Chung, and T. Tsoi, “American sign language recognition and training method with recurrent neural network,” *Expert Systems with Applications*, vol. 167, p. 114403, Apr. 2021, doi: 10.1016/j.eswa.2020.114403.
2. D. Kothadiya, C. Bhatt, K. Sapariya, K. Patel, A.-B. Gil-González, and J. M. Corchado, “Deepsign: Sign Language Detection and Recognition Using Deep Learning,” *Electronics*, vol. 11, no. 11, p. 1780, Jun. 2022, doi: 10.3390/electronics11111780.
3. G. Arun Prasath and K. Annapurani, “Prediction of sign language recognition based on multi layered CNN,” *Multimedia Tools and Applications*, vol. 82, no. 19, pp. 29649–29669, Mar. 2023, doi: 10.1007/s11042-023-14548-1.
4. M. A. As’ari, N. A. J. Sufri, and G. S. Qi, “Emergency sign language recognition from variant of convolutional neural network (CNN) and long short term memory (LSTM) models,” *International Journal of Advances in Intelligent Informatics*, vol. 10, no. 1, p. 64, Feb. 2024, doi: 10.26555/ijain.v10i1.1170.
5. P. Sharma, D. Tulsian, P. Sharma, and N. Nancy, “Indian Sign Language Generation using Natural Language Processing and Audio Speech,” Research Square Platform LLC, Jun. 2022. Accessed: Feb. 11, 2025. [Online]. Available: https://doi.org/10.21203/rs.3.rs-1676438/v1
6. H. Wang, “Overview of Sign Language Translation Based on Natural Language Processing,” *ITM Web of Conferences*, vol. 70, p. 02010, 2025, doi: 10.1051/itmconf/20257002010.
7. A. Baihan, A. I. Alutaibi, M. Alshehri, and S. K. Sharma, “Sign language recognition using modified deep learning network and hybrid optimization: a hybrid optimizer (HO) based optimized CNNSa-LSTM approach,” *Scientific Reports*, vol. 14, no. 1, Oct. 2024, doi: 10.1038/s41598-024-76174-7.
8. K. Shenoy, T. Dastane, V. Rao, and D. Vyavaharkar, “Real-time Indian Sign Language (ISL) Recognition,” in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, Jul. 2018, pp. 1–9. Accessed: Feb. 11, 2025. [Online]. Available: https://doi.org/10.1109/icccnt.2018.8493808
9. Y. Xiao *et al.*., “Low-Latency Video Conferencing via Optimized Packet Routing and Reordering,” in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, IEEE, Jun. 2024, pp.. 1–10. Accessed: Feb. 19, 2025. [Online]. Available: https://doi.org/10.1109/iwqos61813.2024.10682858
10. Z. H. Al-Tairi, R. W. Rahmat, M.I. Saripan and P.S. Sulaiman, “Skin Segmentation Using YUV and RGB Color Spaces,” J Inf Process Syst, vol. 10, no. 2, pp. 283-299, June 2014.
11. B. C. Ennehar, O. Brahim, and T. Hicham, “An appropriate color space to improve human skin detection,” INFOCOMP Journal of Computer Science, vol. 9, no. 4, pp. 1-10, 2010.
12. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O.Grisel et al., “1.6. Nearest Neighbours – scikit-learn 0.19.1 documentation,”2011. [Online]. Available: http://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbor-algorithms. [Accessed: 12- Sep- 2017].
13. L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” Proceedings of the IEEE, vol. 77, no. 2, February 1989.
14. L. Baum, “An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Process,” Inequalities III: Proceedings of the Third Symposium on Inequalities, ssvol. 3, pp. 1-8, 1972.
15. C. Vogler, D. Metaxas, “Handshapes and movements: Multiple channel ASL Recognition,” Gesture-Based Communication in Human- Computer Interaction, pp. 247-258, 2004.