Unsupervised Sign Gloss Persuing from Continuous Sign Language Videos

Partha Mete
metepartha2001@gmail.com

June 26, 2025

Abstract

This project focuses on unsupervised sign gloss segmentation from continuous Indian Sign Language (ISL) videos. The primary objective is to segment a stream of sign language video into discrete gloss units without the use of frame-level manual annotations or labeled gloss boundaries. To achieve this, we leverage a multi-stage pipeline combining pose estimation, dimensionality reduction, clustering, and probabilistic sequence modeling.The dataset used comprises 352 continuous ISL videos, collected and provided by the Department of Computer Science, RKMVERI, Belur. Initially, skeletal features were extracted frame-by-frame using the MediaPipe Holistic model, yielding 3D keypoints for the full-body pose and both hands. To make these keypoints invariant to signer differences, spatial normalization techniques involving centering, scaling, and rotation alignment were applied to each frame.Subsequently, to learn compact and expressive frame-level representations, autoencoders were trained separately for pose and hand data. The encoded latent representations were concatenated and clustered using the K-Means algorithm to obtain discrete observation tokens representing joint posture configurations. These sequences of cluster IDs per video were then modeled using a Hidden Markov Model (HMM) to learn latent sign states and generate gloss-like segmentation boundaries.The overall approach is fully unsupervised and serves as a foundational step toward automatic gloss segmentation in sign language processing. It eliminates the need for extensive manual labeling and paves the way for large-scale annotation-free training of sign language understanding systems.

1 Introduction

Sign languages are natural visual languages used by the Deaf and hard-of-hearing communities around the world. Unlike spoken languages, sign languages utilize a combination of manual (hand gestures, finger positions) and non-manual (body posture, facial expression) modalities to convey meaning. Recognizing and interpreting sign languages using computational methods is a fundamental problem in the field of sign language processing. One particularly important subtask is *gloss segmentation*, which involves identifying temporal boundaries between discrete linguistic units (glosses) in a continuous sign language video stream.

Traditionally, gloss segmentation has relied on manually annotated datasets where each frame is labeled with its corresponding gloss. However, such annotation is highly labor-intensive, time-consuming, and requires linguistic expertise in sign language. This has significantly limited the availability of large-scale training data, especially for low-resource languages such as Indian Sign Language (ISL). Therefore, there is a growing need for unsupervised approaches that can perform this task without frame-level supervision.

In this project, we propose a fully unsupervised pipeline for segmenting continuous ISL videos into gloss-level units. Our method leverages pose and hand keypoint information extracted from RGB videos using the `MediaPipe Holistic` framework. This framework provides dense 3D keypoints for the entire upper body and both hands, which are critical for capturing the articulatory features of sign language. We process each frame of every video to extract 33 pose landmarks and 21 keypoints for each hand, resulting in a rich, multimodal skeletal representation.

To ensure signer-invariance and improve model robustness, a set of spatial normalization techniques are applied. These include centering the skeleton based on body landmarks, scaling based on inter-joint distances, and rotation alignment using the principal gesture direction. The normalized keypoints are then passed through lightweight autoencoder neural networks to learn compressed latent representations that capture meaningful postural configurations while reducing noise and dimensionality.

The resulting low-dimensional feature vectors for each frame are then clustered using the K-Means algorithm to discretize the continuous motion into a sequence of observation symbols. This process effectively converts the sequence of keypoint vectors into a symbolic representation that approximates distinct sign patterns. These discrete sequences are finally modeled using a *Hidden Markov Model* (HMM), which learns latent state transitions and outputs gloss-like segment boundaries via Viterbi decoding.

Key contributions of this work include:

- A complete unsupervised pipeline for gloss segmentation combining pose estimation, normalization, representation learning, clustering, and probabilistic sequence modeling.
- A robust keypoint normalization strategy for making skeleton-based inputs invariant to scale, orientation, and signer-specific variations.
- Use of autoencoders to learn compact embeddings from 3D pose and hand keypoints, enabling efficient clustering and noise reduction.
- Application of HMM-based sequence modeling on cluster-token sequences to extract latent gloss transitions without labeled data.
- A practical demonstration of gloss segmentation on a dataset of 352 ISL videos curated and provided by RKMVERI, Belur.

This work serves as a foundational step toward unsupervised sign language understanding and opens pathways to large-scale gloss discovery and alignment in low-resource sign language contexts.

2 Literature Review

The segmentation of continuous sign language into gloss units without explicit supervision remains a key challenge in sign language understanding. This project builds upon several prior studies across speech processing, video parsing, and unsupervised segmentation. Below, we summarize influential work from foundational models to modern deep-learning-driven approaches.

2.1 Hidden Markov Models and Sequential Modeling

Hidden Markov Models (HMMs) have been foundational in modeling temporal sequences, particularly in speech and gesture recognition. The classical formulation is covered in depth by Jurafsky and Martin [2], which explains the role of state transitions, emission probabilities, and their applicability to noisy, unaligned sequential data.

Emily Fox et al. [1] introduced an advanced Bayesian nonparametric model using the Beta Process for segmenting motion capture data. Their work demonstrates how complex temporal patterns can be modeled without pre-specified state counts—an insight that inspired the unsupervised HMM formulation in this project.

2.2 Unsupervised Sign Language Segmentation

Korhan Polat and Murat Saraclar [5] proposed a technique for unsupervised term discovery in continuous sign language by clustering repetitive units in a multimodal embedding space. While their approach was audio-visual, the clustering-based idea shares commonality with our re-clustered latent triplet pipeline.

Katrin Renz et al. [6] introduced Temporal Convolutional Networks (TCNs) for supervised gloss segmentation using deep video embeddings. Though supervised, their study highlights the importance of temporal filters in capturing gesture transitions—a notion approximated in our HMM-based segmentation.

2.3 Video and Motion Segmentation in Other Domains

Ozan Sener et al. [7] proposed a technique for unsupervised parsing of video collections using co-occurrence and clustering of visual phrases. Their model segments video data into semantically consistent actions, aligning with the high-level goals of unsupervised gloss segmentation in sign language.

The graph partitioning work by Olson et al. [4] applied spectral clustering to identify coherent robotic motion segments. Although their application was in robotics, the technique aligns conceptually with triplet-space clustering for identifying semantic motion boundaries in our pipeline.

2.4 Software Tools and Perception Pipelines

MediaPipe [3] served as the core perception framework for pose and hand landmark detection. Its real-time capability and skeletal fidelity enabled the extraction of rich, structured input features essential for representation learning and segmentation.

2.5 Summary

These prior works informed our model architecture and inspired key decisions:

- Use of HMMs for unsupervised segmentation [2, 1]
- Embedding-based clustering of gesture units [5, 6]
- Graph-structured or motion-parsed sequence interpretation [7, 4]
- Use of fast, modular perception tools for real-time keypoint extraction [3]

Together, these sources provide both the theoretical and applied foundation for the unsupervised sign gloss segmentation system developed in this project.

Distinction from Existing Methods

While several existing methods apply supervised learning or require annotated gloss boundaries [6, 7], the primary distinction of our proposed approach lies in its **fully unsupervised** and **modular architecture**. Unlike works that rely on deep temporal models with gloss-aligned supervision, our system:

- Uses pose and hand keypoints rather than raw video or RGB features.
- Trains modality-specific autoencoders to learn structured, low-dimensional embeddings.
- Employs symbolic clustering followed by Hidden Markov Modeling, without any ground truth gloss labels.

This unsupervised pipeline is particularly suited for low-resource sign languages like Indian Sign Language (ISL), where annotated corpora are scarce or unavailable.

3 Proposed Methodology

3.1 Dataset Overview and Preprocessing

3.1.1 Dataset Source and Description

The dataset used in this project comprises **352 continuous Indian Sign Language (ISL) videos**, collected and provided by the Department of Computer Science, RKMVERI, Belur Campus. Each

video contains a continuous sign sentence performed by a native ISL signer in a constrained indoor environment with a clean background and fixed frontal camera setup. The dataset captures both manual (hand gestures) and non-manual (pose, posture) modalities essential for accurate sign language understanding. The signers are recorded using standard RGB cameras, and no depth sensors or specialized gloves are used, thereby ensuring low-cost and scalable data acquisition.

Each video is named uniquely (e.g., `s0001_f_w000067.mp4`) and varies in duration depending on the sentence complexity. Videos range from approximately 2 seconds to over 5 seconds in length.

3.1.2 Motivation for Using Unlabeled Data

Unlike conventional sign language datasets with gloss-level annotations (e.g., RWTH-PHOENIX-Weather), this dataset is **completely unlabeled at the frame level**. No information regarding gloss boundaries or temporal alignment is available.Only the total no of unique words expressed with sign languages from the whole dataset is known. As a result, the problem setup is inherently **unsupervised**, making the task both challenging and practically relevant for low-resource languages such as ISL, which lack large-scale annotated corpora.

Initial Experiment with CPMC and SCGP Framework

Before finalizing the proposed pipeline using pose and hand-based segmentation with autoencoders and HMM, we initially explored an alternative framework based on Constrained Parametric Min Cuts (CPMC) and Single Cluster Graph Partitioning (SCGP) algorithms.

Method Overview

- **CPMC**: This algorithm was used to generate object proposals by combining superpixel generation (via SLIC) with GrabCut-based foreground refinement. The method outputs several foreground segmentations per frame.
- **Feature Extraction**: Each proposal was cropped using its bounding box, and features were extracted using a pre-trained AlexNet model by accessing its `fc7` layer. These vectors aimed to represent visual semantics of each segmented region.
- **SCGP**: The extracted features were clustered based on cosine similarity. Dominant eigenvectors of the similarity matrix were used to discover cohesive clusters (termed as visual atoms) across frames. An iterative thresholding scheme was used for selection.

Limitations Observed

Despite the sophisticated proposal mechanism and feature clustering strategy, this framework failed to meet the requirements of our task for the following reasons:

- **Over-segmentation**: CPMC focused on large, distinct objects. While it performed reasonably well at detecting gross hand or body movement, it was unable to reliably isolate subtle differences in local motion patterns — especially when only left or right hand changed, while the rest of the pose remained static.

- **Lack of modality-specific control**: Since the proposals were purely visual and holistic, the approach was not designed to disentangle modality-specific signals such as fine-grained hand articulations or shoulder movements separately.
- **Generalization to pose dynamics**: SCGP clustered visual atoms across entire videos but lacked temporal continuity or alignment mechanisms (unlike HMM), which hindered its utility for sequence modeling and segmentation tasks.
- **Computational Overhead**: The proposal generation, especially when scaled to multiple frames per video, introduced a significant overhead. Even with optimizations (e.g., downsampling, parallelism), the approach remained inefficient compared to our lightweight and structured keypoint-based autoencoding approach.

Summary

Although the CPMC + SCGP pipeline produced object-level proposals and clustered high-level patterns effectively, it was not tailored for the precision required in gloss segmentation. The failure to differentiate small-scale hand/pose variations led us to redesign the pipeline with modality-specific encoders and clustering, ultimately yielding better results.

3.1.3 Landmark Extraction using MediaPipe Holistic

We employed the **MediaPipe Holistic** model developed by Google to extract skeletal and hand landmark keypoints from RGB video frames. The model returns frame-wise keypoints for the following modalities:

- **Pose landmarks:** 33 3D landmarks of the upper and lower body, each with x, y, z, visibility components.
- **Left hand landmarks:** 21 keypoints, each with x, y, z coordinates.
- **Right hand landmarks:** 21 keypoints, each with x, y, z coordinates.

The holistic model processes each frame individually and outputs these landmarks in normalized coordinates (i.e., values are in $0, 1$ relative to the frame size). For each video, the entire frame sequence is passed through MediaPipe and the extracted keypoints are serialized and saved in a `.pkl` format.

3.1.4 Coordinate Transformation and Scaling

To convert the normalized coordinates into pixel-level spatial coordinates, each keypoint is scaled by the original frame's resolution:

$$x_{\text{scaled}} = x_{\text{norm}} \cdot \text{frame width}, \quad y_{\text{scaled}} = y_{\text{norm}} \cdot \text{frame height}, \quad z_{\text{scaled}} = z_{\text{norm}} \cdot \text{frame width}$$

This transformation enables meaningful visualization and preparation for 3D modeling.

3.1.5 Skeleton Normalization and Invariance

To ensure that keypoint features are invariant to the signer's scale, body proportions, and position in the frame, we apply a robust 3D skeleton normalization pipeline:

- **Centering:** Pose keypoints are centered using the midpoint between shoulder joints (landmarks 11 and 12). Hand keypoints are centered around the wrist (landmark 0).
- **Scaling:** The skeleton is scaled based on bone-length metrics. For pose, the inter-shoulder distance is used; for hands, the wrist-to-fingertip chain length (e.g., $0 \rightarrow 9 \rightarrow 12$) is computed.
- **Rotation Alignment:** A rotation matrix is computed using the direction of the spine (neck–hip vector) or middle finger direction to align the skeleton vertically. The up-vector is standardized as $0, 1, 0$ using Rodrigues' formula to compute the alignment rotation.

These transformations are applied per-frame and make the keypoint data invariant to viewpoint and signer orientation, which is crucial for consistent feature learning across the dataset.

3.1.1 Storage and Data Representation

The processed keypoints for each video are saved into a Python dictionary with the following structure:

```
{
    "Video height": H,
    "Video width": W,
    "pose": ndarray [F x 33 x 4],
    "left hand": ndarray [F x 21 x 3],
    "right hand": ndarray [F x 21 x 3]
}
```

Where F is the number of frames. These dictionaries are stored in `.pkl` files named according to the input video, enabling efficient loading during training and inference.

3.1.7 Visualization and Verification

To validate the quality and consistency of the extracted and normalized keypoints, we implemented multiple visualization modules:

- **2D visualization:** Overlaid pose and hand keypoints on the original frame.
- **3D visualization:** Rendering pose and hand skeletons in 3D using Matplotlib and Plotly.
- **Skeleton offset correction:** Properly anchoring hand landmarks to wrist positions based on pose joints.

These visualizations help confirm anatomical accuracy and verify that the MediaPipe model generalizes well across different ISL signers and videos.

3.1.8 Summary

This preprocessing pipeline extracts and prepares high-quality skeletal and hand keypoint data from raw RGB videos in an unsupervised, annotation-free manner. The output is a consistent set of normalized, scalable, and signer-invariant representations that form the basis for downstream feature learning and sequence modeling in later stages of the system.

3.2 Feature Extraction and Representation Learning

To obtain meaningful, low-dimensional representations from high-dimensional 3D skeletal data, a dedicated autoencoder-based representation learning pipeline was developed. The approach decomposed the full-body keypoint data into anatomically relevant components—pose, left hand, and right hand—and learned compact embeddings for each.

3.2.1 Dataset Consolidation and Serialization

After per-video normalization and preprocessing, all valid frames across the 352 videos were aggregated into a centralized dictionary structure. This facilitated efficient training of downstream models.

Batch Processing of Videos Each video was read using OpenCV and processed frame-by-frame. Using the MediaPipe Holistic model:

1. Pose, left hand, and right hand keypoints were extracted per frame.
2. Frames missing any of these components were excluded.
3. Remaining frames underwent normalization (Section 3.1.4).

Keypoint Dictionary Structure Each frame was indexed using the convention: {`video_id`}_{`frame_index`}. Each dictionary entry contained:

- `"Video height"` and `"Video width"`.
- `"pose"`: $33, 4$ array of normalized (x, y, z, visibility).
- `"left hand"` and `"right hand"`: Each a $21, 3$ array.

Serialization The full dataset was stored using `pickle` as:

`all_transformed_keypoints.pkl`

offering efficient data access and reproducibility.

3.2.2 Autoencoder Dataset Preparation and Design

Upper Body Keypoint Selection To reduce pose dimensionality while preserving semantics, only 9 upper-body landmarks were retained:

- **Nose** (0), **Eyes** (2, 5), **Shoulders** (11, 12), **Elbows** (13, 14), **Wrists** (15, 16)

Each point had 3 coordinates: x, y, z, giving a total of 27 features for pose.

Hand Vectorization Each hand used all 21 landmarks (3D), forming 63-dimensional vectors:

$$\mathbf{x}_{\text{left}}, \mathbf{x}_{\text{right}} \in \mathbb{R}^{63}$$

Final Dataset Format Each sample had the form:

$$\mathbf{x}_{\text{pose}} \in \mathbb{R}^{27}, \quad \mathbf{x}_{\text{left}} \in \mathbb{R}^{63}, \quad \mathbf{x}_{\text{right}} \in \mathbb{R}^{63}$$

PyTorch Dataset Loader A custom `torch.utils.data.Dataset` class accessed the serialized pickle and yielded data triplets, supporting batch loading via PyTorch's `DataLoader`.

3.2.3 Autoencoder Architectures

Pose Autoencoder

- Encoder: $27 \to 64 \to 32 \to 16$
- Decoder: $16 \to 32 \to 64 \to 27$
- Activation: Sigmoid

Hand Autoencoders (Left and Right)

- Encoder: $63 \to 128 \to 64 \to 16$
- Decoder: $16 \to 64 \to 128 \to 63$
- Activation: ReLU

All models used `nn.Sequential()` modules in PyTorch.

3.2.4 Training Strategy and Optimization

Loss Function Minimization of Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}^i - \hat{\mathbf{x}}^i\|^2$$

Optimizer and Training Control

- Optimizer: Adam ($\eta = 0.001$)
- Scheduler: `ReduceLROnPlateau`
- Early stopping after 50 stagnant epochs

Checkpointing Each model's best weights (lowest validation loss) were stored using:

- `pose_aesefull.pth`
- `left_hand_ae4.pth`
- `right_hand_ae4.pth`

3.2.5 Reproducibility Measures

To ensure training consistency, the following seeds were fixed:

- `random.seed(42)`
- `numpy.random.seed(42)`
- `torch.manual_seed(42)`

3.2.6 Summary

Through separate autoencoder pipelines for upper-body pose and each hand, the system achieved dimensionality reduction and modular representation learning. These encoders were critical for producing latent embeddings used in downstream clustering and sequence modeling via HMMs.

3.3 Clustering and Observation Sequence Generation

Following latent code extraction via autoencoders, the next step involved discretizing the continuous gesture embeddings into symbolic representations. This symbolic form is required for modeling temporal dynamics using probabilistic methods like Hidden Markov Models (HMMs). The conversion was achieved using a two-step clustering strategy: initial component-wise clustering followed by joint triplet re-clustering.

3.3.1 Latent Code Extraction

Each frame from the dataset was passed through the encoder modules of the trained autoencoders for:

- **Pose** $\rightarrow$ 16-dimensional vector
- **Left hand** $\rightarrow$ 16-dimensional vector
- **Right hand** $\rightarrow$ 16-dimensional vector

These yielded three modality-specific latent vectors per frame, denoted as:

$$\mathbf{z}_{\text{pose}} \in \mathbb{R}^{16}, \quad \mathbf{z}_{\text{left}} \in \mathbb{R}^{16}, \quad \mathbf{z}_{\text{right}} \in \mathbb{R}^{16}$$

3.3.2 Modality-Wise K-Means Clustering

Rather than clustering the full 48-dimensional concatenated latent vector, clustering was performed independently for each modality. This approach reduces inter-modal interference and leverages the anatomical independence of pose and hand movements.

Clustering Settings Using elbow plot analysis to determine optimal cluster sizes:

- **Pose latent vectors:** K-Means with $K = 50$
- **Left hand latent vectors:** K-Means with $K = 100$
- **Right hand latent vectors:** K-Means with $K = 100$

Clustering was performed using `sklearn.cluster.KMeans` with random seed 42 to ensure reproducibility. Each latent vector was mapped to a discrete cluster ID, producing:

$$\texttt{pose\_cluster\_id} \in 0, 49, \quad \texttt{left\_cluster\_id} \in 0, 99, \quad \texttt{right\_cluster\_id} \in 0, 99$$

3.3.3 Triplet Construction

For every frame, the three modality-specific cluster IDs were merged into a unique triplet:

$$\texttt{triplet\_id} = \texttt{pose\_id}, \texttt{left\_id}, \texttt{right\_id}$$

This generated a total vocabulary of **9870 unique triplets**, each representing a distinct gesture configuration across pose and both hands.

3.3.4 Re-Clustering for HMM Compatibility

Since HMMs operate more efficiently with a manageable number of discrete observations, these 9870 triplet IDs were further clustered into **512 global observation symbols**. This was achieved via a second-level K-Means clustering on the 3D integer triplet space:

$$\texttt{pose\_id},\ \texttt{left\_id},\ \texttt{right\_id} \rightarrow \texttt{cluster\_id} \in 0, 511$$

This reduced vocabulary of 512 symbols was used to represent each frame as a single discrete observation.

3.3.5 Observation Sequence Formation

Each video was now represented as a time-ordered sequence of integers:

$$\texttt{obs\_sequence}_i = c_1,\ c_2,\ \ldots,\ c_{T_i}$$

where $c_j \in \{0, 1, \ldots, 511\}$ corresponds to the re-clustered observation symbol for the j-th frame. All such sequences were stored as a list of lists:

$$\texttt{obs\_sequences} = \texttt{obs\_sequence}_1, \texttt{obs\_sequence}_2, \ldots, \texttt{obs\_sequence}_{352}$$

3.3.6 Output Format and Serialization

The final observation sequences and mappings were serialized using Python's `pickle` module and saved as:

- `bphmm_obs_sequences_int.pkl`: Contains `obs_sequences` and `id_to_triplet`.
- `bphmm_obs_sequences_int_reclustered.pkl`: Final re-clustered symbolic sequences for HMM input.

3.3.7 Summary

This section established a pipeline for converting continuous latent gesture representations into discrete symbolic sequences by:

- Performing modality-wise K-Means clustering on pose and hand latent codes.
- Generating joint triplet IDs for each frame.
- Re-clustering triplets into 512 HMM-compatible observation tokens.

These sequences were subsequently used as input to train the Hidden Markov Model for unsupervised gloss segmentation.

3.4 Hidden Markov Model-Based Gloss Segmentation

To discover temporal sign boundaries and model gloss transitions from frame-wise observation sequences, we employed a Hidden Markov Model (HMM). The HMM is a widely used probabilistic model suitable for analyzing time series where the observed sequence is generated by a sequence of unobserved (hidden) states.

3.4.1 What is a Hidden Markov Model?

A Hidden Markov Model (HMM) is a statistical model that represents a system with:

- **Hidden states:** Unobservable variables that evolve over time following a Markov process.
- **Observations:** Emitted symbols (discrete or continuous) that depend probabilistically on the current hidden state.

An HMM is formally defined by the following components:

- $\mathbf{S} = \{s_1, s_2, \ldots, s_N\}$: Set of N hidden states.
- $\mathbf{O} = \{o_1, o_2, \ldots, o_T\}$: Sequence of observed symbols.
- $\mathbf{A} = a_{ij}$: State transition probability matrix, where $a_{ij} = P s_{t1} = j \mid s_t = i$.
- $\mathbf{B} = b_j k$: Emission probability matrix, where $b_j k = P o_t = k \mid s_t = j$.
- $\pi = \pi_i$: Initial state distribution, where $\pi_i = P s_1 = i$.

Given these parameters $\lambda = \mathbf{A}, \mathbf{B}, \pi$, an HMM is used to:

1. Compute the likelihood of an observed sequence (*Forward algorithm*).
2. Decode the most likely sequence of hidden states (*Viterbi algorithm*).
3. Learn model parameters from data (*Baum–Welch algorithm*, a form of Expectation-Maximization).

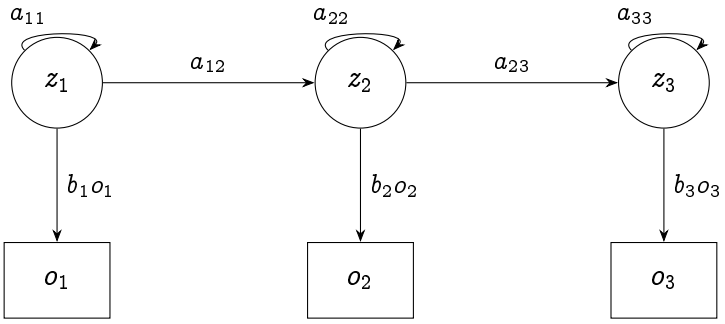


Figure 1: Illustration of a Hidden Markov Model (HMM). Each hidden state z_t emits an observation o_t with probability $b_t o_t$ and transitions to the next state with probability a_{ij}. In our context, z_t represents a latent gloss-like state and o_t is a re-clustered gesture token.

3.4.2 HMM in the Context of Sign Gloss Segmentation

In our task, the HMM is used to segment continuous sequences of sign gestures (represented by discrete tokens) into temporally coherent gloss units. Specifically:

- **Observations:** Frame-wise cluster IDs (integers in $0, 511$) generated by re-clustering the pose-hand triplet vectors.
- **Hidden States:** Latent gloss-level labels that the model learns to assign to each frame, representing unknown gloss transitions.
- **Sequence:** Each video is modeled as a sequence of observed tokens emitted by an underlying sequence of gloss-like hidden states.

3.4.3 Model Configuration

The model was implemented using `hmmlearn`'s `CategoricalHMM`, configured as follows:

- **Number of hidden states:** 270 (based on the known 303 unique signs, adjusted for model smoothness).
- **Number of observation symbols:** 512 (as obtained from re-clustering of 9870 triplet IDs).
- **Training:** 60 epochs, one EM iteration per epoch.
- **Initialization:** Random initialization of emission probabilities; transition structure retained across iterations.

The complete observation data across all videos was concatenated and used to train the model globally.

3.4.4 Training Workflow

1. All re-clustered observation sequences were flattened into one long sequence.
2. The model was initialized and trained using the Baum–Welch algorithm for 60 EM iterations.
3. After each epoch, the model's log-likelihood was computed and saved for convergence tracking.
4. Intermediate models were saved at every epoch using `joblib`.

3.4.5 Decoding with Viterbi Algorithm

Once training was complete, the **Viterbi algorithm** was used to infer the most probable sequence of hidden states for each video:

$$\texttt{latent\_state\_sequence}_i = z_1, z_2, \ldots, z_T$$

These latent state IDs provide an unsupervised approximation of the underlying gloss structure.

3.4.6 Segmentation via State Grouping

To extract segment boundaries:

- Consecutive frames with the same predicted latent state were grouped.
- Each segment was stored as a tuple: (state ID, segment length).

This produced a sequence of contiguous state segments per video, interpreted as predicted gloss boundaries.

3.4.7 Qualitative Evaluation

Model predictions were qualitatively validated using:

- **Latent state timeline plots:** Per-video visualization of hidden state transitions.
- **State frequency distribution:** Most common latent states across the dataset.
- **Histogram of segment lengths:** Duration analysis for most frequent states.

These analyses confirmed that the model learned meaningful patterns and discovered recurring sign structures across multiple videos.

3.4.8 Outputs and Persistence

The following were saved:

- `hmm_viterbi_latent_states.pkl`: Latent state sequences for all 352 videos.
- `hmm_video_segments.pkl`: Segment boundaries as (state, duration) tuples.
- `hmm_models_categorical/`: Trained HMM models across epochs.

3.4.9 Summary

In summary, the Hidden Markov Model provided a powerful statistical framework to convert re-clustered token sequences into gloss-like segmentations. Despite operating without any labeled data, the HMM learned latent states that aligned with semantic structure in sign gesture streams, demonstrating the viability of unsupervised gloss segmentation in continuous sign language.

4 Experimental Results

4.1 Datasets Used

The dataset used for this project comprises **352 continuous sign language videos** recorded and curated by the Department of Computer Science, RKMVERI, Belur Campus. These videos capture complete ISL sentences performed by fluent native signers under consistent indoor conditions using a fixed frontal camera. All recordings are RGB videos in `.mp4` format with variable lengths depending on sentence complexity.

4.1.1 Dataset Characteristics

- **Total videos:** 352
- **Modality:** RGB (no depth or audio)
- **Format:** `.mp4`, standard resolution
- **Framerate:** 60 FPS
- **Setting:** Controlled indoor background, uniform lighting
- **Gloss vocabulary:** 303 known unique glosses (used only for estimating latent state count)

4.1.2 Annotation Status

No gloss-level annotation or frame-level labeling was available in the dataset. Thus, the project operates in a **fully unsupervised setting**. The only known ground truth was the number of distinct glosses used in the recordings.

4.1.3 Justification for Dataset Choice

- **Practicality:** Reflects a realistic case of low-resource sign language where annotation is scarce.
- **Complexity:** Contains coarticulated sign sequences requiring temporal modeling.
- **Consistency:** Uniform background and signer alignment facilitated consistent keypoint extraction.

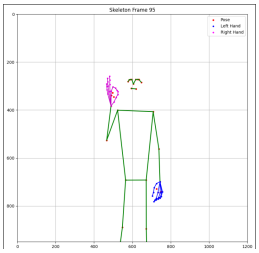
4.1.4 Preprocessing Summary

- Extracted 3D pose and hand landmarks using MediaPipe Holistic.
- Applied normalization (centering, scaling, rotation alignment).
- Serialized frame-wise data into a structured pickle dictionary: `all_transformed_keypoints.pkl`
- Generated latent vectors for each frame through autoencoders.
- Produced symbolic sequences via KMeans and re-clustering (for HMM modeling).

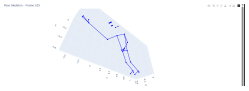
4.1.5 Outcome

This dataset formed the foundation for the complete unsupervised pipeline — from keypoint representation learning to gloss segmentation using a Hidden Markov Model, without requiring any manually annotated data.

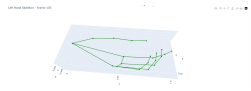
Keypoints Plots



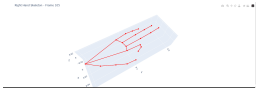
(a) Raw Full Skeleton



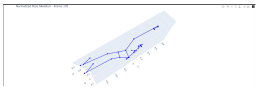
(b) Raw Pose (3D)



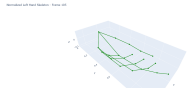
(c) Raw Left Hand (3D)



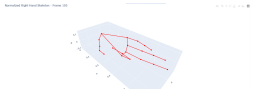
(d) Raw Right Hand (3D)



(e) Normalized Pose



(f) Normalized Left Hand



(g) Normalized Right Hand

Figure 2: Visualization of extracted and normalized skeletal components: full body, pose, and hands. Raw 3D coordinates are shown above, and normalized vectors used for training are shown below.

4.1.6 Dataset Overview

The evaluation was performed on a dataset comprising **352 video samples** of Indian Sign Language gestures provided by RKMVERI, Belur. Each video was preprocessed and normalized to extract skeletal keypoints, resulting in a total of several thousand usable frames after filtering incomplete samples.

4.1.7 Training Parameters

Each autoencoder was trained independently using the following configuration:

- **Batch Size:** 512 for pose, 1024 for hands
- **Epochs:** Up to 4000 (with early stopping)
- **Learning Rate:** 1×10^{-3}, reduced on plateau
- **Optimizer:** Adam
- **Loss Function:** Mean Squared Error (MSE)
- **Learning Rate Scheduler:** ReduceLROnPlateau with factor 0.5 and patience of 50 epochs
- **Early Stopping:** Triggered if no improvement for 50 epochs

4.1.8 Checkpointing and Logging

Training progress, including loss and learning rate changes, was logged per epoch. The best model weights were saved in:

- `pose_aesefull.pth`
- `left_hand_ae4.pth`
- `right_hand_ae4.pth`

These models were later used for reconstruction, analysis, and latent space visualization.

4.1.9 Reproducibility Measures

To ensure that results were deterministic and could be reproduced, random seeds were fixed for all relevant libraries including NumPy, PyTorch, and Python's built-in `random` module. Additionally, GPU non-determinism was minimized using PyTorch backend flags.

4.1.10 Evaluation Criteria

Model performance was evaluated primarily using:

- **Mean Squared Error (MSE)** between input and reconstruction
- **Visual fidelity** of 3D skeleton reconstructions
- **Qualitative comparison** across multiple frames and gesture types

These evaluations formed the basis for assessing the success of the learned representations.

4.2 System Configuration and Frameworks

This section describes the hardware, software, and deep learning frameworks used to implement and train each stage of the proposed unsupervised gloss segmentation pipeline.

4.2.1 Hardware Setup

All experiments were conducted using Google Colab's hosted GPU environment with the following specifications:

- **GPU:** NVIDIA Tesla T4 (16 GB VRAM)
- **CPU:** 2-core Intel Xeon (virtualized)
- **RAM:** 12–16 GB available

- **OS:** Ubuntu-based virtual Linux environment

This setup enabled reasonable training and processing times for large video datasets, keypoint extraction, clustering, and model training.

4.2.2 Software Stack

The implementation used a combination of open-source Python libraries. Key packages include:

- **Pose Extraction:** `mediapipe==0.10.2` (Holistic model for pose and hand keypoints)
- **Data Processing:** `numpy`, `pickle`, `OpenCV`, `matplotlib`, `seaborn`
- **Deep Learning:** `PyTorch==2.0.1` (for autoencoders and reproducibility)
- **Clustering:** `scikit-learn==1.2.2` (KMeans for pose/hand/triplet clustering)
- **Sequence Modeling:** `hmmlearn==0.3.0` (Categorical HMM for gloss segmentation)
- **Model Saving & Debugging:** `joblib`, `torch.save`, and custom checkpoint utilities

4.2.3 Reproducibility Setup

To ensure deterministic behavior, all random seeds were explicitly fixed:

- `random.seed(42)`
- `numpy.random.seed(42)`
- `torch.manual_seed(42)`

In addition, model checkpoints and dataset pickles were consistently saved after each major stage, ensuring that the full pipeline could be rerun from any point.

4.2.4 Intermediate Artifact Organization

Each stage of the pipeline generated persistent intermediate outputs for later stages. Major files included:

- `all_transformed_keypoints.pkl` – normalized keypoint data
- `pose_aesefull.pth`, `left_hand_ae4.pth`, `right_hand_ae4.pth` – trained autoencoder models
- `bphmm_obs_sequences_int.pkl` – observation triplets

- `bphmm_obs_sequences_int_reclustered.pkl` – 512-symbol sequence used in HMM
- `hmm_models_categorical/` – trained HMM checkpoints

These modular artifacts enabled stage-wise evaluation, debugging, and time complexity analysis.

4.3 Model-wise Experimental Settings

This subsection describes the key hyperparameters, training behavior, and architectural validation for each of the major components in the pipeline: autoencoders, KMeans clustering, and the Hidden Markov Model (HMM). While the methodology outlined their structure, this section focuses on empirical tuning and training performance.

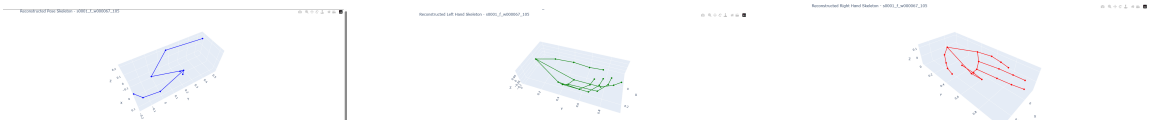
4.3.1 Autoencoder Training Parameters

Each of the three autoencoders (pose, left hand, right hand) was trained independently on the preprocessed dataset.

- **Optimizer:** Adam
- **Learning Rate:** 1×10^{-3}
- **Loss Function:** Mean Squared Error (MSE)
- **Batch Size:** 512
- **Epochs:** Up to 4000 (with early stopping)
- **Scheduler:** ReduceLROnPlateau (factor 0.1, patience 20)
- **Stopping Condition:** No improvement in validation loss for 50 epochs

4.3.2 Autoencoder Training Behavior

- All models showed stable convergence within 1500–2000 epochs.
- Pose autoencoder converged faster due to lower input dimensionality (27D).
- Hand autoencoders took longer due to higher complexity and dimensionality (63D each).
- No overfitting was observed, and reconstruction quality visually preserved spatial patterns.



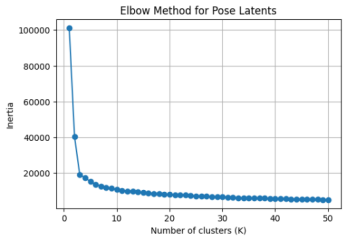
(a) Reconstructed Pose (b) Reconstructed Left Hand (c) Reconstructed Right Hand

Figure 3: Output from trained autoencoders showing reconstruction quality for upper-body pose and hand modalities. These visuals illustrate the ability of the encoders to retain key spatial structures despite dimensionality reduction.

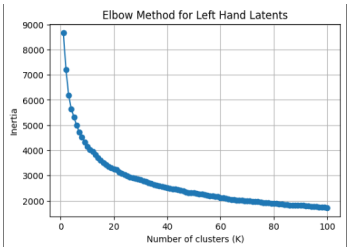
4.3.3 KMeans Clustering Parameters

- **Clustering Algorithm:** `sklearn.cluster.KMeans`
- **Initialization:** k-means++
- **Max Iterations:** 300
- **Random Seed:** 42
- **Clusters:**
  - Pose latent vectors: 50 clusters
  - Left hand latent vectors: 100 clusters
  - Right hand latent vectors: 100 clusters

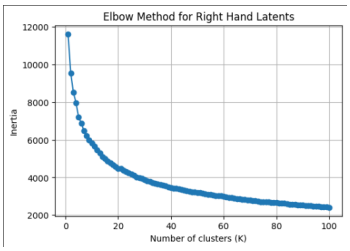
Elbow plots were used to empirically determine the number of clusters for each modality by plotting the distortion score against k.



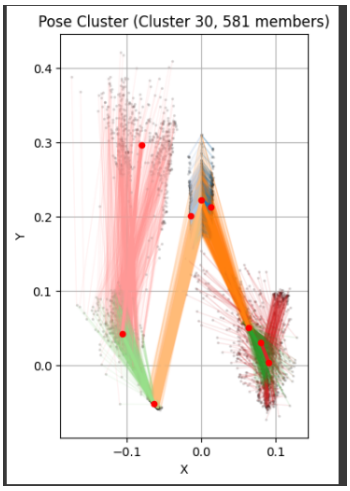
(a) Pose Elbow Curve



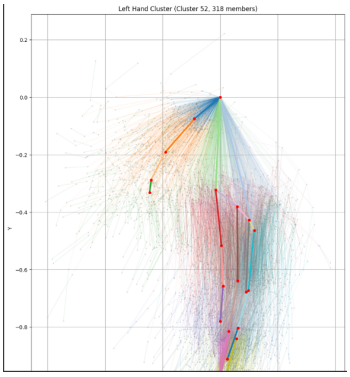
(b) Left Hand Elbow Curve



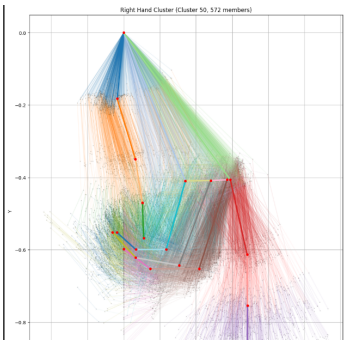
(c) Right Hand Elbow Curve



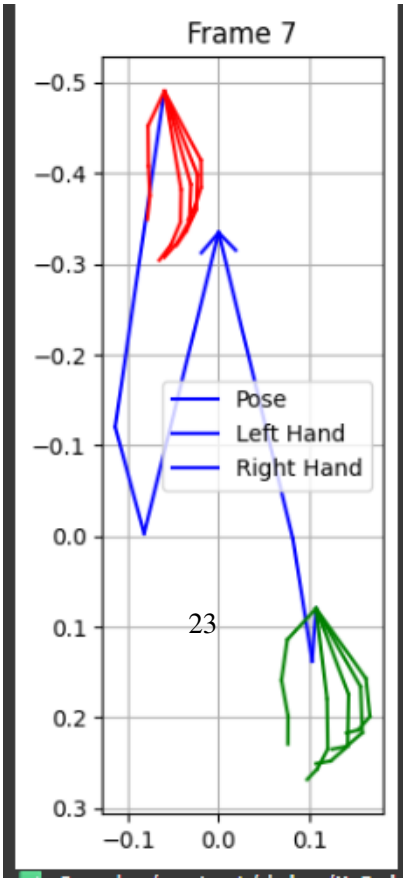
(d) Pose Cluster Centers + Members



(e) Left Hand Cluster Centers + Members



(f) Right Hand Cluster Centers + Members



(g) Cluster Centers in a Sample Frame

4.3.4 Triplet Re-clustering

- 9870 unique triplet combinations of (pose ID, left ID, right ID) were generated.
- These were re-clustered into 512 discrete symbols using KMeans on the triplet space.
- The result was a vocabulary suitable for HMM-based modeling of temporal dynamics.

4.3.5 Hidden Markov Model Parameters

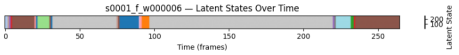
- **Model:** `hmmlearn.hmm.CategoricalHMM`
- **Number of Hidden States:** 270 (inferred from known gloss count of 303)
- **Number of Observation Symbols:** 512
- **Training Iterations:** 60 (one EM iteration per epoch)
- **Verbose Logging:** Enabled for log-likelihood tracking
- **Evaluation:** Viterbi decoding and segment visualization

4.3.6 HMM Training Behavior

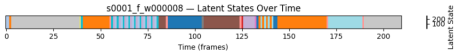
- Training log-likelihood increased steadily.
- Model checkpoints were saved at each epoch for reproducibility and recovery.
- Latent states began forming stable patterns and repeated transitions across videos.

4.4 Experimental Results and Qualitative Evaluation

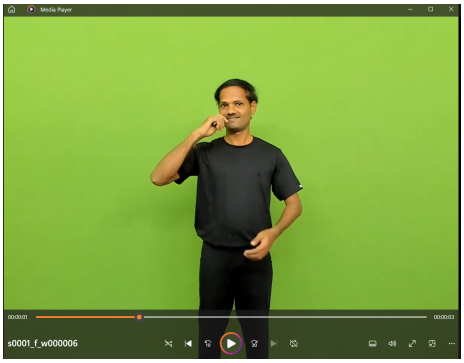
Since the dataset contains no ground truth gloss annotations at the frame level, evaluation was primarily qualitative. However, rich insights were obtained from segment statistics, latent state behavior, and visual inspection of predicted gloss-like sequences.



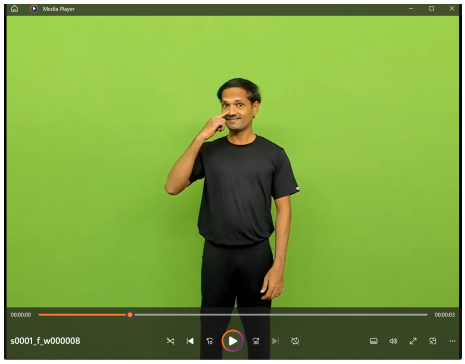
(a) Video A: HMM State Timeline



(b) Video B: HMM State Timeline

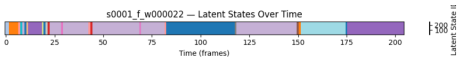


(c) Video A: Matching Gesture Snapshot

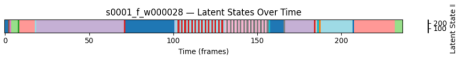


(d) Video B: Matching Gesture Snapshot

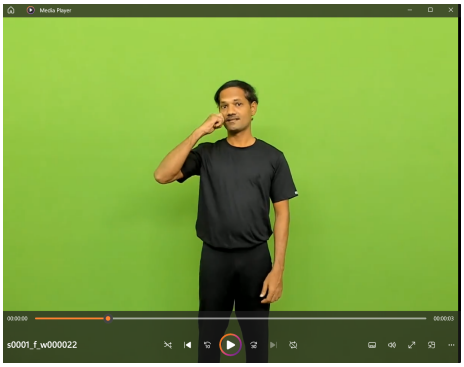
Figure 5: Pairwise gesture consistency. Videos A and B exhibit a shared HMM latent state corresponding to visually similar gestures. Top: identical color-coded latent state segment in both timelines. Bottom: corresponding gesture frames from each video.



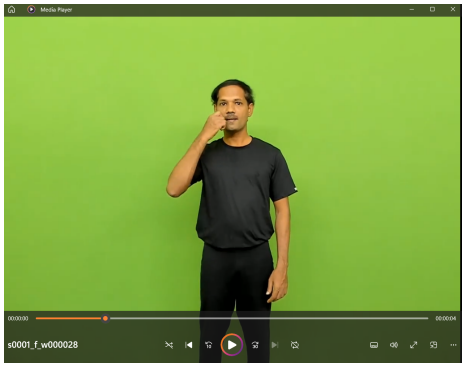
(a) Video A: HMM State Timeline



(b) Video B: HMM State Timeline

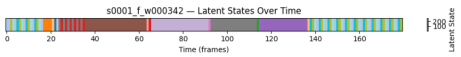


(c) Video A: Matching Gesture Snapshot

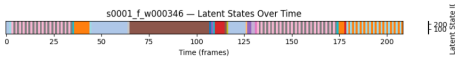


(d) Video B: Matching Gesture Snapshot

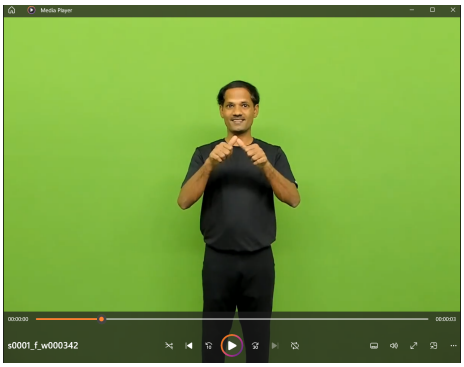
Figure 6: Pairwise gesture consistency. Videos A and B exhibit a shared HMM latent state corresponding to visually similar gestures. Top: identical color-coded latent state segment in both timelines. Bottom: corresponding gesture frames from each video.



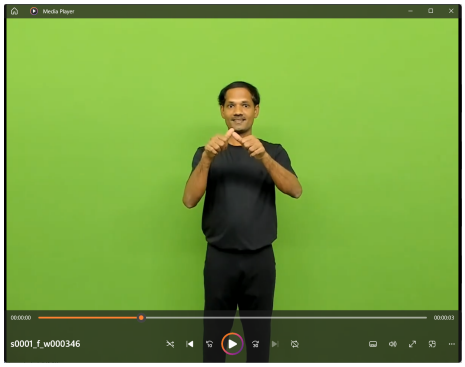
(a) Video A: HMM State Timeline



(b) Video B: HMM State Timeline

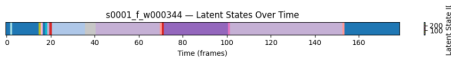


(c) Video A: Matching Gesture Snapshot

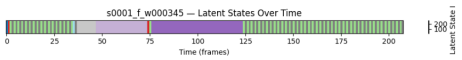


(d) Video B: Matching Gesture Snapshot

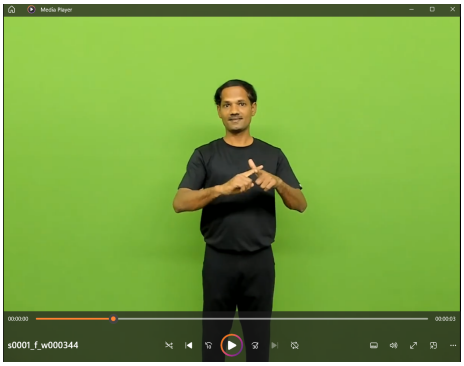
Figure 7: Pairwise gesture consistency. Videos A and B exhibit a shared HMM latent state corresponding to visually similar gestures. Top: identical color-coded latent state segment in both timelines. Bottom: corresponding gesture frames from each video.



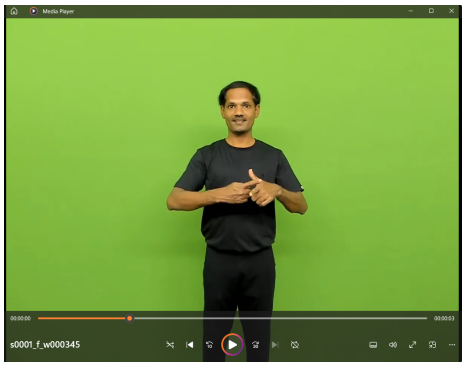
(a) Video A: HMM State Timeline



(b) Video B: HMM State Timeline

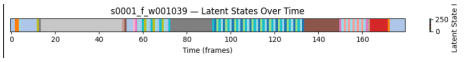


(c) Video A: Matching Gesture Snapshot

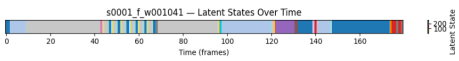


(d) Video B: Matching Gesture Snapshot

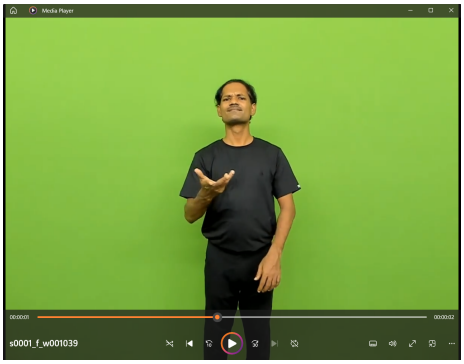
Figure 8: Pairwise gesture consistency. Videos A and B exhibit a shared HMM latent state corresponding to visually similar gestures. Top: identical color-coded latent state segment in both timelines. Bottom: corresponding gesture frames from each video.



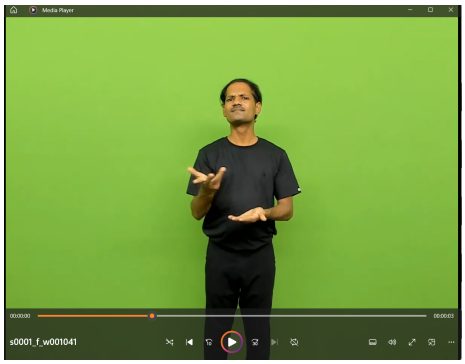
(a) Video A: HMM State Timeline



(b) Video B: HMM State Timeline

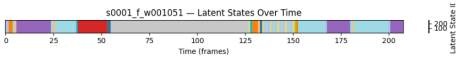


(c) Video A: Matching Gesture Snapshot

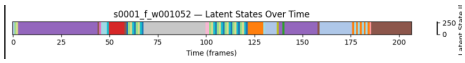


(d) Video B: Matching Gesture Snapshot

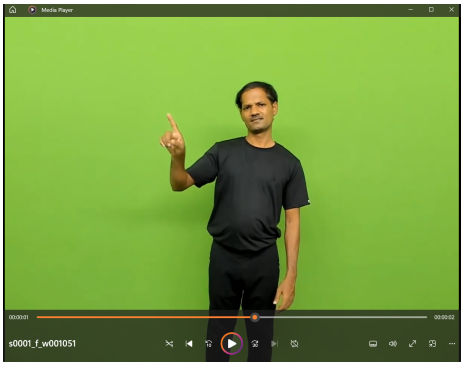
Figure 9: Pairwise gesture consistency. Videos A and B exhibit a shared HMM latent state corresponding to visually similar gestures. Top: identical color-coded latent state segment in both timelines. Bottom: corresponding gesture frames from each video.



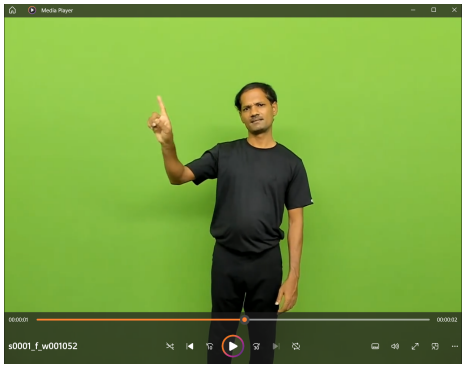
(a) Video A: HMM State Timeline



(b) Video B: HMM State Timeline



(c) Video A: Matching Gesture Snapshot



(d) Video B: Matching Gesture Snapshot

Figure 10: Pairwise gesture consistency. Videos A and B exhibit a shared HMM latent state corresponding to visually similar gestures. Top: identical color-coded latent state segment in both timelines. Bottom: corresponding gesture frames from each video.

4.4.1 Latent State Behavior

- Out of the 270 available hidden states, typically 258 were utilized across the dataset.
- Certain latent states were found to recur across multiple videos and signers, suggesting the HMM successfully discovered frequent sub-gestures or sub-gloss patterns.
- Some states were uniquely associated with specific videos, indicating that the model can also adapt to signer-specific or context-specific variations.

4.4.2 Gloss Segmentation Insights

- Viterbi-decoded state sequences showed clear transitions between discrete segments.
- Consecutive frames with identical state IDs were grouped to form gloss-like segments.
- The average number of predicted segments per video varied between 10 and 30, depending on sentence complexity and signer speed.
- Segment durations ranged from 2 to 80 frames, with most falling between 10 to 40

4.4.3 Visualization and Segment Statistics

Multiple visualizations were generated to interpret model predictions:

- **State Timeline Plots:** Frame-wise latent states were color-coded to illustrate transitions for each video.
- **State Frequency Histograms:** Top 20 most frequent states were identified and plotted.

These tools validated that the system was able to uncover meaningful temporal structure, even without labeled supervision.

4.4.4 Interpretation

The qualitative evidence strongly suggests that the proposed pipeline captures underlying gesture dynamics and approximates gloss segmentation boundaries:

- Repeated state transitions reflected consistent gesture sub-units.
- State clusters were temporally stable and not noisy.
- Segmentation patterns visually aligned with observable changes in sign pose/posture.

4.4.5 Comparison to Supervised Systems

Direct comparison to state-of-the-art CSLR systems is not possible due to lack of ground truth. However, key advantages of the proposed approach include:

- Fully unsupervised pipeline requiring zero annotations.
- Interpretable latent states and temporal structures.
- Potential utility for bootstrapping future labeled datasets via weak supervision.

Limitations Observed Despite the overall pipeline producing interpretable segmentations in several videos, we observed instability in the HMM outputs for a subset of samples. Specifically:

- Some sequences exhibited prolonged repetition of the same latent state, leading to under-segmentation.
- Others displayed overly frequent transitions between states, indicating possible over-segmentation or instability.
- These issues suggest that further tuning of the number of HMM states, re-cluster granularity, or latent smoothing strategies could improve segmentation reliability.

Future iterations of this work will explore additional regularization of the HMM and refinement of the clustering pipeline to improve segment-level accuracy.

4.5 Time Complexity and Runtime Analysis

This subsection outlines the computational complexity and real-time performance of each major component in the proposed sign gloss segmentation pipeline. Actual runtime values were measured on Google Colab GPU/CPU hybrid systems.

4.5.1 Preprocessing and Keypoint Extraction

- **Theoretical Complexity:** $\mathcal{O}F$ per video, where F is the number of frames.
- **Tool Used:** MediaPipe Holistic (executed on CPU).
- **Average Time per Video:** 20–30 seconds (depending on length).
- **Output:** 50,000+ valid frames across all 352 videos, saved as normalized pose/hand landmarks.

4.5.2 Autoencoder Training Time

- **Theoretical Complexity:** $\mathcal{O}n \cdot d \cdot e$ per model (where n is number of frames, d is input dimension, e is number of epochs).
- **Training Duration (each model):**
  - Pose autoencoder: 3 hours
  - Left hand autoencoder: 4 hours
  - Right hand autoencoder: 4 hours
- **Device Used:** Google Colab GPU (Tesla T4)
- **Epochs:** 3000-4000 (early stopping after convergence)
- **Batch size:** 512

4.5.3 Clustering and Re-clustering

- **Modality-wise Clustering:**
  - Pose (50 clusters): 5 minutes
  - Left Hand (100 clusters): 8 minutes
  - Right Hand (100 clusters): 8 minutes
- **Re-clustering (9870 triplets $\rightarrow$ 512 tokens):** 10 minutes
- **Tool Used:** Scikit-learn's `KMeans`, executed on CPU

4.5.4 Hidden Markov Model Training and Inference

- **Training Library:** `hmmlearn` (no GPU support)
- **Number of Hidden States:** 270
- **Observation Vocabulary:** 512 symbols
- **EM Iterations:** 60
- **Training Time:** 7–8 hours on CPU
- **Inference (Viterbi decoding):** 1–2 seconds per video

4.5.5 Overall Runtime Summary

- **Total Autoencoder Training Time:** 11 hours (GPU)
- **Total Clustering Time:** 20–25 minutes (CPU)
- **HMM Training Time:** 7–8 hours (CPU only)
- **Preprocessing Time:** 30–40 minutes total for 352 videos
- **End-to-End Inference Time (per video):** 6–8 seconds

While certain components such as autoencoder and HMM training were time-intensive, these stages are performed only once. The inference pipeline remains fast, modular, and scalable for large-scale or real-time use.

5 Summary

This project addressed the challenging task of **unsupervised sign gloss segmentation** from continuous Indian Sign Language (ISL) videos. The primary objective was to segment raw RGB sign sequences into meaningful gloss-level units without any frame-wise or gloss annotations.

A full pipeline was developed using the following stages:

- **Pose and hand keypoint extraction:** MediaPipe Holistic was used to extract high-resolution skeletal data (pose, left hand, right hand) from each video frame.
- **Feature encoding:** Three autoencoders were trained separately on pose and hand vectors to learn compact 16D latent representations for each modality.
- **Symbol generation:** KMeans clustering was applied to modality-specific embeddings (50 for pose, 100 for each hand), forming 9870 latent triplet IDs, which were re-clustered into 512 discrete observation symbols.
- **Sequence modeling:** A Categorical Hidden Markov Model (HMM) with 270 hidden states was trained on the sequence of clustered symbols. Viterbi decoding was used to obtain latent state sequences per video.
- **Segmentation:** Consecutive frames with the same latent state were grouped to form unsupervised gloss segments.

The system was evaluated qualitatively through latent state timelines, segment statistics, and visual inspections of output sequences. Despite the absence of ground truth labels, the HMM discovered meaningful latent structures, with many state transitions corresponding to observable gesture boundaries.

The entire pipeline was implemented using reproducible Python code and trained using publicly available GPU resources. Intermediate outputs (latent vectors, cluster IDs, HMM checkpoints, segment maps) were saved at each stage to ensure modularity and interpretability.

This work demonstrates that unsupervised sign language segmentation is achievable using keypoint-based embeddings and sequence models, and sets the stage for future research in self-supervised sign language understanding.

6 Limitations and Future Work

While the proposed unsupervised gloss segmentation system shows promising results, several limitations were encountered during development and experimentation. These limitations highlight areas where future work can enhance the performance, robustness, and generalizability of the system.

6.1 Current Limitations

- **Data Scarcity:** Although 352 videos were processed, the overall number of frame-level training samples was limited to approximately 75,491. In contrast, the Hidden Markov Model (HMM) had over 210,000 free scalar parameters (transition and emission probabilities), leading to the warning:

  ```
  WARNING: Fitting a model with 210869 free scalar parameters with
  only 75491 data points will result in a degenerate solution.
  ```

  This imbalance led to potential overfitting and less stable convergence of the HMM.

- **Lack of Ground Truth Annotations:** The dataset did not include any gloss-level or frame-level labels, making it impossible to evaluate segmentation quality quantitatively. All evaluations were qualitative or statistical.

- **HMM Library Constraints:** The `hmmlearn` package used for training lacks GPU acceleration and supports only categorical distributions. This restricted model scalability and training speed, especially over 60 EM iterations.

- **Under/Over Segmentation Issues:** In several videos, the HMM produced excessively long or short latent states. These unstable transitions indicate a need for stronger temporal regularization or alternative models such as Segmental HMMs or duration-aware models.

- **Cluster Granularity Limitations:** The 512-token vocabulary, while a balance between expressivity and tractability, may have merged or split some semantically meaningful motion segments. More nuanced vector quantization methods (e.g., VQ-VAE) could improve symbolic alignment.

6.2 Future Work Directions

- **Larger and Labeled Dataset Integration:** Incorporating datasets like RWTH-PHOENIX-Weather or newly annotated ISL corpora can enable both supervised evaluation and semi-supervised training.
- **Model Architecture Enhancements:** Future pipelines could replace vanilla autoencoders with contrastive learning or variational methods to better preserve gesture-specific differences.
- **Improved Sequence Models:** Alternatives to standard HMMs such as Segmental HMMs, Transformer-based sequential decoders, or RNNs with attention mechanisms could provide more robust temporal segmentation.
- **End-to-End Training:** Instead of a modular approach, an end-to-end differentiable pipeline combining encoding, clustering, and segmentation may yield smoother gradient flows and improved performance.
- **Gloss-to-Text Pretraining:** If pseudo-gloss segments can be mapped to weak textual annotations (e.g., keywords or subtitles), the system could evolve into a full sign-to-text prototype using weak supervision.

Addressing these limitations can significantly enhance both the quality and applicability of unsupervised gloss segmentation systems in low-resource sign language settings.

7 References

References

[1] Emily B. Fox, Michael C. Hughes, Erik B. Sudderth, and Michael I. Jordan. Joint modeling of multiple time series via the beta process with application to motion capture segmentation. *The Annals of Applied Statistics*, 8(3):1281–1313, 2014.

[2] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson, 3rd edition, 2025. Draft dated January 12, 2025.

[3] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[4] Edwin Olson, Matthew Walter, Seth Teller, and John Leonard. Single-cluster spectral graph partitioning for robotics applications. In *Robotics: Science and Systems (RSS)*, Cambridge, MA, 2005. MIT Press.

[5] Korhan Polat and Murat Saraclar. Unsupervised term discovery for continuous sign language. In *Interspeech 2019*, Graz, Austria, 2019. ISCA.

[6] Katrin Renz, Nicolaj C. Stache, Samuel Albanie, and Gul Varol. Sign language segmentation with temporal convolutional networks. *arXiv preprint arXiv:2302.00590*, 2023.

[7] Ozan Sener, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4480–4488. IEEE, 2015.