Movement Epenthesis Detection in Compressed and Uncompressed Continuous Sign Language Videos

A
Thesis submitted
for the award of the degree of
DOCTOR OF PHILOSOPHY
By

ANJAN KUMAR TALUKDAR
Under the Supervision of
Prof. M.K. BHUYAN

Department of Electronics and Electrical Engineering
Indian Institute of Technology Guwahati
Guwahati-781039, India
June, 2022

Declaration

This is to certify that the thesis entitled "**Movement Epenthesis Detection in Compressed and Uncompressed Continuous Sign Language Videos**", submitted by me to the *Indian Institute of Technology Guwahati*, for the award of the degree of Doctor of Philosophy, is a bonafide work carried out by me under the supervision of Prof. M.K. Bhuyan. The contents of this thesis, in full or in parts, have not been submitted to any other University or Institute for the award of any degree or diploma.

Signed: _____

Anjan Kumar Talukdar
Department of Electronics and Electrical Engineering,
Indian Institute of Technology Guwahati,
Guwahati-781039, Assam, India.

Date: _____

Certificate

This is to certify that the thesis entitled "**Movement Epenthesis Detection in Compressed and Uncompressed Continuous Sign Language Videos**", submitted by Anjan Kumar Talukdar (10610202), a research scholar in the *Department of Electronics and Electrical Engineering*, *Indian Institute of Technology Guwahati*, for the award of the degree of Doctor of Philosophy, is a record of an original research work carried out by him under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Signed: _____

Supervisor: Prof. M.K. Bhuyan
Department of Electronics and Electrical Engineering,
Indian Institute of Technology Guwahati,
Guwahati-781039, Assam, India.

Date: _____

Acknowledgements

I have the pleasure to acknowledge with the deepest sense of gratitude, the help, and advice given to me, by many people, without which this work would not have been possible.

To begin with, I express my deepest and most sincere gratitude to my Ph.D supervisor Prof. M.K. Bhuyan sir who kindly accepted me to be in his research group and guided me throughout my research work. He has always been supportive to me, motivated me in my difficult times, and always encouraged me for new reserach findings. I have learned many new things about the research process from him.

I am very much thankful to the members of my doctoral committee, Prof. Prabin Kumar Bora sir, Prof. Kannan Karthik sir, and Prof. Arijit Sur sir for their support, encouragement and valuable suggestions on my research work. I would like to thank the faculty members and office-staff of Electronics and Electrical Engineering Department, IIT Guwahati for their help in carrying out the research work. I sincerely thank Prof. Kandarpa Kumar Sarma sir of Gauhati University who encouraged me to do research work at IIT Guwahati. I am also thankful to Prof. Manish Okade, NIT Rourkela for providing his helping hand in doing this research work. I am also grateful to Mr. Heramba Bhuyan who helped me a lot during the collection of the video datasets from Govt. B.D.S. Deaf and Dumb School, Kahilipara, Guwahati, Assam.

I am thankful to my friends and juniors at the Computer Vision and Image Processing Laboratory of IIT Guwahati, who were always ready for any kind of help required. I am also thankful to my wife Mrs. Ananya Choudhury for her assistance and encouragement for completing the Ph.D.

Above all, I am deeply grateful to my parents and my other family members. It would not have been possible to complete my work without their love, support, and sacrifice.

Sincerely
Anjan Kumar Talukdar

Abstract

Continuous sign language recognition (SLR) system suffers from a problem called coarticulation. The articulation of the present sign is modified by the articulation of the previous and the next signs, and this is termed as coarticulation. Movement epenthesis (me) is a special type of coarticulation which is a transition from one sign to other. It is a transitional movement of the hands which occurs between any two consecutive signs in continuous sign language. So, sign spotting is a process of segmenting out signs and fingerspellings from continuous utterance sentences by removing movement epenthesis components, which is a challenging task.

Most of the state-of-the-art approaches on vision-based SLR systems were developed for uncompressed raw videos. However, additional information like motion vectors (MVs), macroblocks (MBs), Discrete Cosine Transform (DCT) coefficients, etc. are available in compressed domain frameworks. To address the drawbacks of the existing uncompressed domain algorithms, a framework for continuous fingerspelling spotting in the compressed domain is proposed in ***Chapter 3****. The framework is based on motion vectors extracted from H.264/AVC compressed videos. A spatio-temporal Markov random field (ST-MRF) based model is employed to model the non-rigid motions of fingers as "sign" or "me". For considering camera motions, a novel framework for global motion estimation (GME) and global motion compensation (GMC) in the compressed domain is proposed. An average spotting rate of 75% is achieved.*

In the second approach, a continuous fingerspelling recognition system in the uncompressed domain is proposed in ***Chapter 4****. Adaptive thresholding and Finite State Machine (FSM) models are employed for efficient classification of "sign" and "me" frames. Segmented fingerspelled letters are further recognized by a conditional random field (CRF) based classifier and an average recognition rate of 91.29% is achieved.*

In the third work, a framework for continuous sign language spotting using a 2-state Hidden Markov Model (HMM) with Gaussian emission probability is proposed in ***Chapter 5****, in which advantages of using both uncompressed as well as compressed domain features are exploited. Features extracted from the entire sign-sequence video are inputted to the HMM and finally, the hidden state-sequence is decoded using the Viterbi algorithm.*

From the decoded state sequence, the sign spotting is done. Experimental results show that the proposed method can separate "sign" and "me" frames with an average spotting rate of 80%.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

me	Movement Epenthesis
ASL	American Sign Language
BME	BoostMap Embedding
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DL	Deep Learning
DTW	Dynamic Time Warping
FD	Fourier Descriptor
FN	False Negative
FP	False Positive
FSM	Finite State Machine
GM	Global Motion
GMC	Global Motion Compensation
GME	Global Motion Estimation
GOP	Group of Pictures
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
LK	Lucas-Kanade
MB	Macroblock
MC	Motion Compensation
MCER	Motion Compensated Error Residual
MCP	Maximum Curvature Point

MMSE Minimum Mean Square Error

MSE Mean Square Error

MV Motion Vector

MVF Motion Vector Field

NGT Netherland Sign Language

SLR Sign Language Recognition

ST-MRF Spatio-Temporal Markov Random Field

SVM Support Vector Machine

Chapter 1

Introduction

Sign language is a visual language used by the deaf community. Sign language comprises two components: signs and fingerspellings. Signs are dynamic hand gestures where the global motion of the hands takes place with some hand configurations. In fingerspelling, the motion of the hand is local where only the fingers move (except 'J' and 'Z' where the entire hand moves) [3]. Fig. 1.1 shows various fingerspelled alphabets used in American sign language (ASL). Signs are used for the pre-existing vocabulary of words. When no vocabulary exists for a word (e.g., name of a person, place, etc.), then fingerspelling is used to spell out the letters of the word one by one.

The aim of the sign language recognition (SLR) system is to transcribe sign language into text or speech so that it can be easily understood by the hearing community [14]. It plays an important role as a sign language interpreter to bridge the communication gap between the deaf and the hearing communities. There are three levels in an SLR system: fingerspelling (alphabets), isolated words, and continuous sign language (sentences) [15]. In isolated SLR, only one sign is articulated by the signer in a video, but in continuous SLR, a sentence may consist of a sequence of several signs in a video. There can be many non-sign movements (called *movement epenthesis*) between true signs which connect the

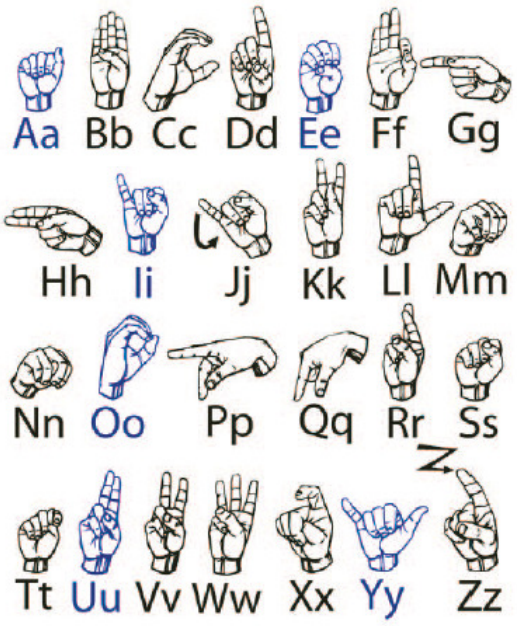


FIGURE 1.1: The ASL fingerspelled alphabets [1].

end location of the previous sign to the start location of the next sign. Fig. 1.2 shows the movement epenthesis (*me*) frames between signs "GATE" and "WHERE" in a continuous sign video [2]. The leftmost frame corresponds to the end location of sign "GATE" and the last frame corresponds to the start location of sign "WHERE". In between, there are intermediate movements (*me* frames) that connect these two signs.

Sign spotting is a process of finding out start and end frames of signs and fingerspellings by removing *me* frames from a continuous utterance sentence video which is a challenging task [3]. Fig. 1.3 shows an example of sign spotting for a sentence "Arrive-M-A-R-Y-Yesterday".

FIGURE 1.2: Movement epenthesis *(me)* between "GATE" and "WHERE" signs [2].

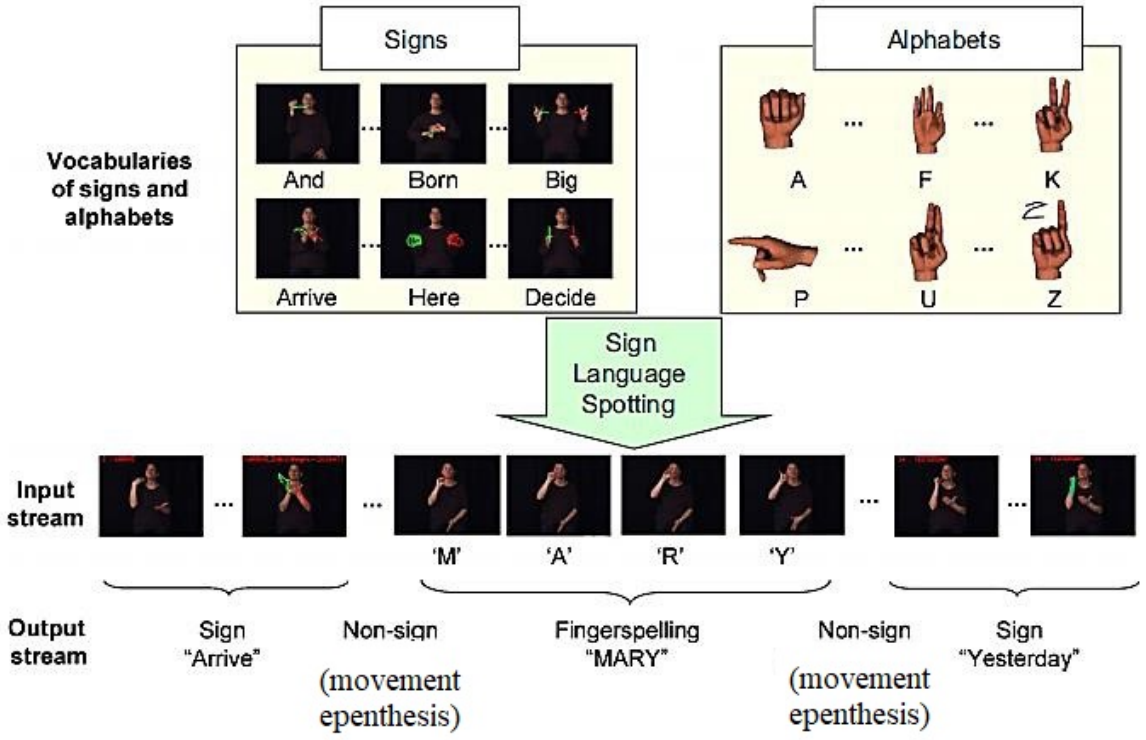


FIGURE 1.3: Sign spotting for the sentence "Arrive-M-A-R-Y-Yesterday" [3].

1.1 Coarticulation in continuous sign language

Continuous sign language recognition system suffers from a problem called coarticulation [16]. The articulation of the present sign is modified by the articulation of the previous and next signs which is termed as coarticulation. Modification can be addition, alteration, or deletion which can take place at the beginning, end, or inside the sign [17]. Some of the coarticulation effects are discussed in the following sections.

1.1.1 Coarticulation effects for handshapes due to contact type in Netherland sign language

The sign GOAT of Netherland sign language (NGT) starts at the chin (Fig. 1.4) and then moves away from it, while the sign OLD starts close to the chin and moves towards it [4]. In terms of the sign OLD, the transition from the final position of the preceding sign to the chin in the final contact sign OLD is thus predicted to consist of merely

a transitional movement, identical to the transitional movement to the first skeletal position of the initial contact sign GOAT.

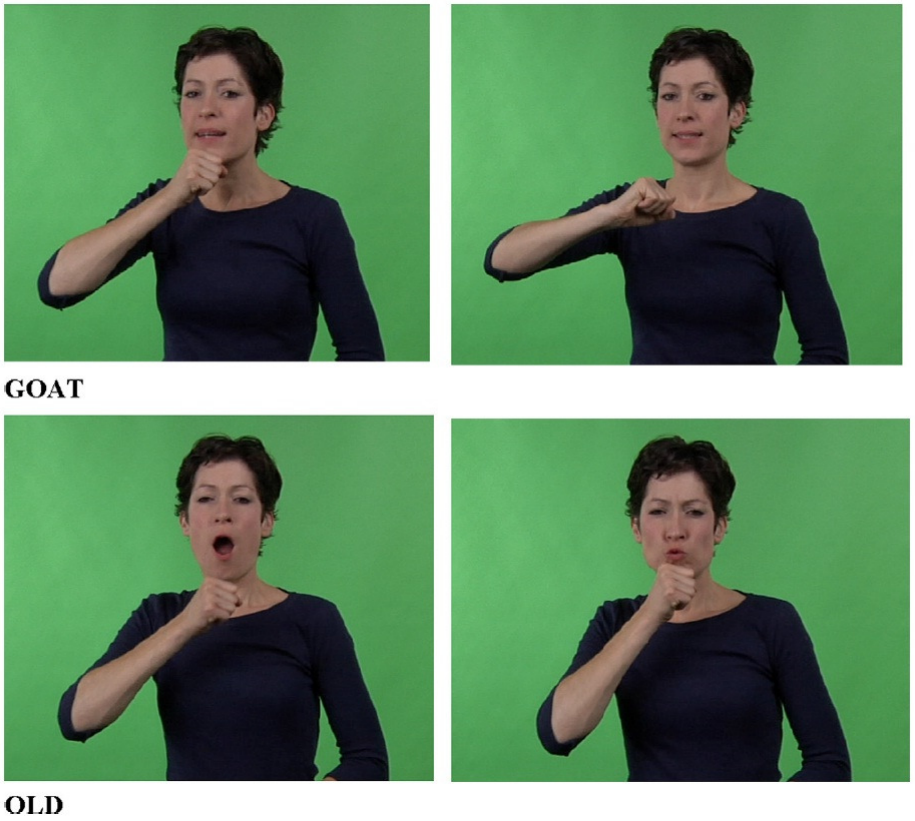


FIGURE 1.4: NGT signs GOAT (initial contact at the chin in the top left picture) and OLD (final contact at the chin in the bottom right picture) [4].

1.1.2 Coarticulation with respect to pinky extension

The pinky extension coarticulation is conditioned by both preceding and following handshapes where pinky extension is there. For example, Fig. 1.5(a) and Fig. 1.5(b) show the configurations of the handshapes for the fingerspelled letter –R-, for two different lexicons for ASL [5]. Fig. 1.6 shows the effects of the neighboring hand shapes on the letter -L-.

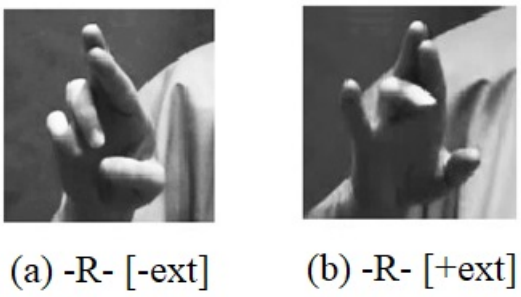


FIGURE 1.5: The configurations of the handshapes for the fingerspelled letter –R-, for (a) D-N-O-S-A-U-R (b) C-H-R-I-S [5].

FIGURE 1.6: The configurations of the handshapes from the fingerspelling of B-U-I-L-D-I-N-G in ASL.

1.1.3 Timing in ASL fingerspelling

It has been found that the initial and final letters are spelled slower than the medial letters. The hold time is longer for signs having movement and orientation. The hold time and transition time are also varied from signer to signer.

1.1.4 Coarticulation of thumb position in Netherland sign language

The preceding and the following signs predict the thumb state in target signs [6]. This turned out to be true for the degree of flexion of the thumb depending on the degree of flexion of the preceding and the following signs, as well as the degree of abduction of the thumb that similarly depending on the degree of abduction of the following or preceding signs (Fig. 1.7). Moreover, the degree of spreading of the fingers in the target signs is correlated with the position of the thumb of those signs.

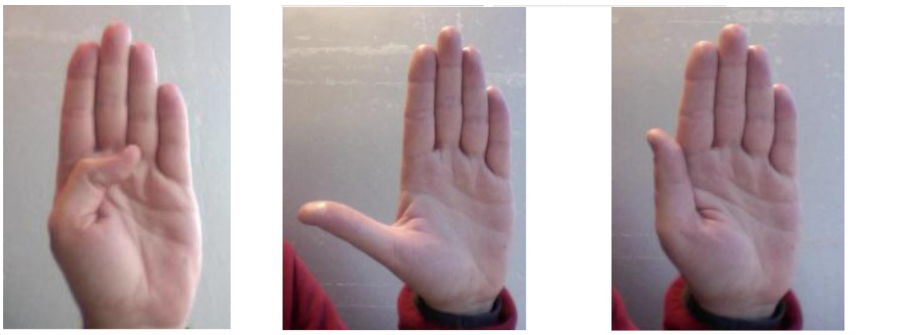


FIGURE 1.7: Three different flat handshapes [6].

1.1.5 Undershoot of ASL locations in fast signing

Few researchers studied phonetic effects such as undershoot in sign languages. Cheek [18] examined variation in the ASL index hand caused by the proximity of a 5-hand. She analyzed the index handshapes in both normal and fast signing and found that signing rate was a factor in the degree of variation in the handshape measure.

Mauk et al. [7] found similar results looking at the 5 and index handshapes over a range of signing rates. It may be expected that extreme contrasts of location will be difficult to execute as the signing rate increases. As a result, signers may undershoot the location of one or more signs.

Four native signers of ASL were asked to sign sets of sequences composed of three signs each. These sequences constituted four combinations of neutral space signs and forehead signs. For example, the sign sequence shown in Fig. 1.8 contains a neutral space sign CHILDREN preceded and followed by the forehead located sign SMART. As the signing rate increases, the location value for one or more of the three signs may be reduced. That is, in order to complete the sign sequence in a shorter amount of time, the signer may raise the neutral space sign or lower the two forehead signs as shown in Fig. 1.9.

FIGURE 1.8: ASL sign sequence SMART CHILDREN SMART [7].

FIGURE 1.9: Possible variants in ASL sign CHILDREN in normal (left) and fast (right) signings [7].

1.1.6 Coarticulation in American sign language (ASL) fingerspelling

Coarticulation in fingerspelling is typically characterized as assimilation (where sequential hand shapes become more similar to one another) or dissimilation (where sequential hand shapes become more different) [8]. Fig. 1.10 shows some cartoon images of letters of interest in ASL. Here we can see that the letters C is almost similar to A, but has a significant difference with U. If we consider the spelling of the letters one by one for the words DISCUSS and DISCARD (having same preceding letters before I and different following letters after C, we observe that there is a difference in movements from S to C for the two words. Similarly, if we consider the words DISCARD and CONFISCATE (having different preceding letters before I and same following letters after S), then also the same phenomenon happens. Again we observe that movement from S to C is slower when S is followed by A (assimilation) and faster when followed by U (dissimilation). The movement graphs are shown in Fig. 1.11.

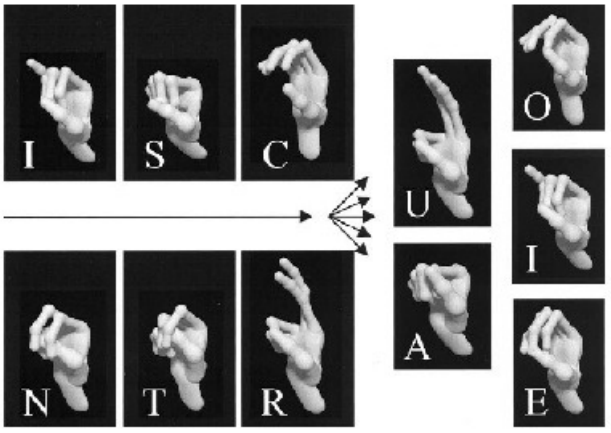


FIGURE 1.10: Cartoon images of hand shapes representing letters of interest [8].

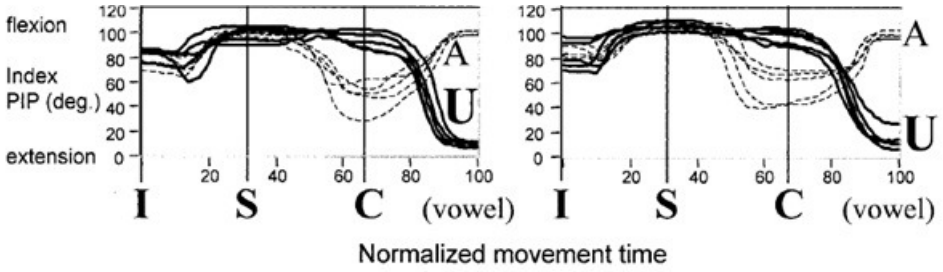


FIGURE 1.11: Joint angle data of the index finger proximal interphalangeal (PIP) joint for spelling I-S-C-A (dashed line) and I-S-C-U (solid line) [8].

1.1.7 Movement epenthesis in continuous sign language

Movement epenthesis (*me*) is a special type of coarticulation which is a transition from one sign to another [1], [2], [19]. This is an additional movement that occurs during the transition of a sign to another sign or during the beginning or the ending of a sign. Fig. 1.12 shows the movement epentheses between signs GATE and WHERE and during the beginning of GATE and after the ending of WHERE.

FIGURE 1.12: Movement epenthesis in continuous sign language [1].

1.2 Video processing in the compressed domain

Most of the works on vision-based sign language recognition systems have been reported using uncompressed raw videos. Presently, most of the videos are found in compressed form due to their reduced file size and requirement of less transmission time and bandwidth. For processing compressed images and videos, we first need to decompress them and supply suitable processing algorithms, which are usually developed considering their representations in spatial or spatiotemporal domains. Often it is also necessary to recompress the processed output, to make them interoperable with other applications. For example, a JPEG compressed image, after processing, may still be required to be represented by the JPEG compression standard for its display by a browser, storage in an archive, or transmission through a network. Under this scenario, every spatial domain processing of images is encapsulated by additional overheads of decompression and compression as depicted in Fig. 1.13.

To reduce the overhead involved in the decompression and recompression steps (Fig. 1.13), we may consider performing processing with the inter-mediate symbols of alternative representations of images/videos in the compressed form instead of working with

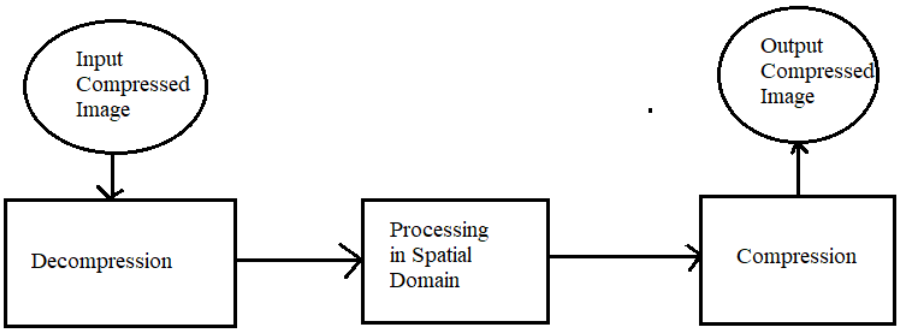


FIGURE 1.13: Basic steps in processing with compressed domain [9].

their original representations in the spatial domain. The processing in the domain of alternative representation is referred here as processing in the compressed domain [9].

There is also another advantage of working with the representation in the compressed domain. Usually, the size of data in this case is smaller than that of its spatial representation (or spatiotemporal representation of videos). This inherently makes the processing less memory intensive [20], [9].

Third, the property of factorization of an image in its alternative representation sometimes becomes useful in developing algorithms. We may consider emphasizing or deemphasizing some factors after analyzing their role in the reconstruction of the image.

So, sign spotting frameworks for continuous sign language videos in the compressed domain are more useful. Advantages of processing the videos in the compressed domain are [20]:

- As the processing is done on compressed data, the processing is faster.
- The inherent information in the video stream can be extracted using partial decoding.

  The video codec is discussed in more detail in the subsequent sections.

1.2.1 Video codec outline

In video sequence coding, such as video telephony, a combination of temporal and spatial coding techniques is used in order to remove the "predictable or redundant" image

contents and encode only the unpredictable information. Fig. 1.14 shows the simple encoder/decoder (codec) structure [10].

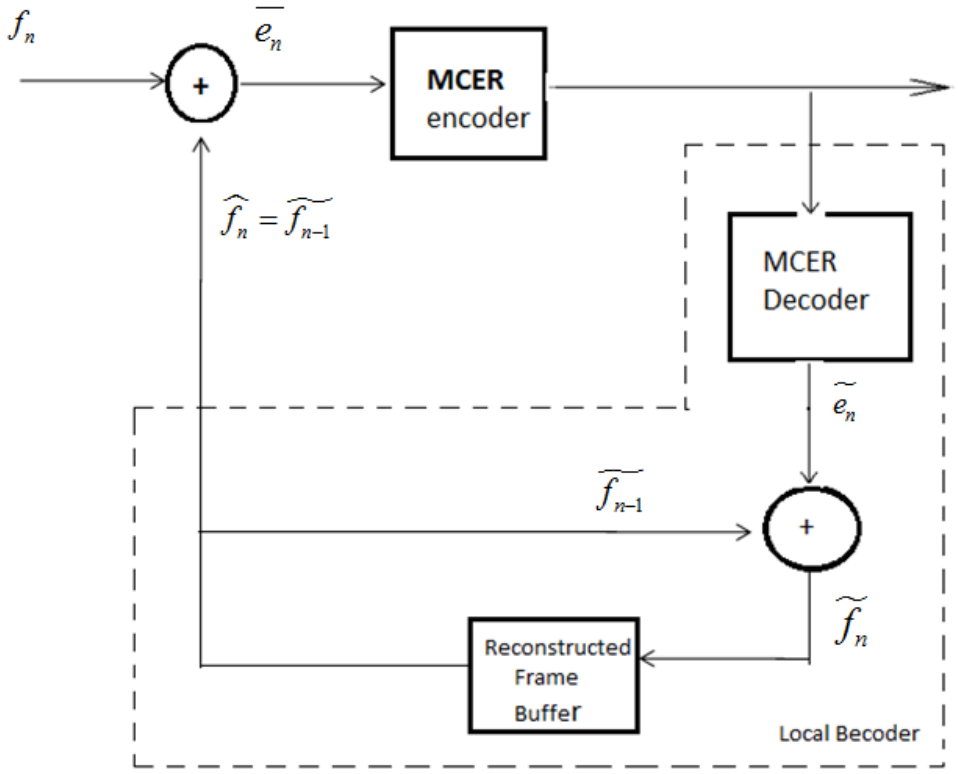


FIGURE 1.14: Basic video codec schematic using frame differencing [10].

Assuming that the 176 × 144 pixel Quarter Common Intermediate Format (QCIF) ITU standard videophone sequence to be encoded contains a head-and-shoulder video clip, the consecutive image frames f_n and f_{n-1} typically do not exhibit dramatic scene changes. Hence, the consecutive frames are similar, a property that we refer to as being correlated. This implies that the current frame can be approximated or predicted by the previous frame, which can be expressed as $f_n \approx \widehat{f_n}$ where $\widehat{f_n}$ denotes the n^{th} predicted frame. When the previous frame f_{n-1} is subtracted from the current one, namely f_n due to the speaker's movement, most of the areas of this difference frame are 'flat', having values close to zero, and the variance or second-order moment of it is significantly lower than that of the original frame.

The reduced variance difference signal, namely $e_n = f_n - f_{n-1}$, is referred to as the Motion Compensated Error Residual (MCER) since the associated frame-differencing effectively attempts to compensate for the motion of the objects between consecutive video frames, yielding a reduced-variance MCER. Thus, e_n requires a reduced coding rate, that is, a lower number of bits in order to represent it with a certain distortion than f_n. The e_n can be encoded as $\overline{e_n}$, with the required distortion using a variety of techniques. The quantized or encoded MCER signal $\overline{e_n}$ is then transmitted over the communication channel. Since the previous undistorted image frame f_{n-1} is not available at the decoder, image reconstruction has to take place using their available approximate values, namely, $\overline{e_n}$ and $\widehat{f}_{n-1}$, giving the reconstructed image frame as

$$\widetilde{f_n} = \widetilde{e_n} + \widetilde{f}_{n-1} \tag{1.1}$$

The above equation is portrayed in Fig. 1.14, where the current video frame f_n is predicted by $\widehat{f_n} = \widetilde{f}_{n-1}$, an estimate based on the previous reconstructed frame $\widetilde{f}_{n-1}$.

Instead of using the previous original frame, the so-called locally decoded frame is used in the motion compensation. The local decoding yields an exact replica of the video frame at the distant decoder's output. The local decoding is necessary because the previous original frame is not available at the distant decoder.

1.2.2 The principle of motion compensation

The simple codec of Fig. 1.14 used a low-complexity temporal redundancy removal or motion compensation (MC) technique, often referred to as frame differencing. The disadvantage of frame differencing is that it is incapable of tracking more complex motion trajectories.

Image sequences typically exhibit spatial redundancy within frames and temporal redundancy in consecutive frames. Motion compensation attempts to remove the latter. The motivation behind block-based motion compensation is to analyze the motion trajectory.

In an often-used MC implementation, the image frame is divided into a number of perfectly tiling 2×2 to 16×16 pixel blocks, which is then slid over a certain search area (as shown in Fig. 1.15) surrounding the corresponding location in the previous reconstructed frame $\widetilde{f}_{n-1}$ in order to identify the specific position from which each block has originated. The motion vectors (MVs) describing the motion translation are two-dimensional and are typically restricted to integer multiples of pixel separation. The motion vectors identifying this way are then applied to the entire group of pixels within this block. The predicted current block constituted by approximate motion translated, the previously reconstructed block is subtracted from the current block about to be encoded as portrayed in Fig 1.15. The operations are reflected in the modified codec schematic of Fig 1.16.

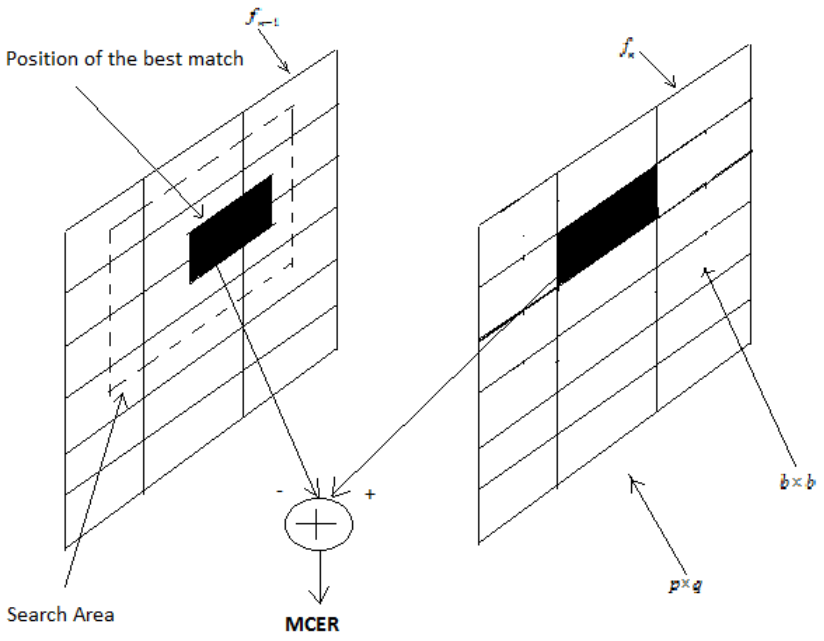


FIGURE 1.15: The process of motion compensation [10].

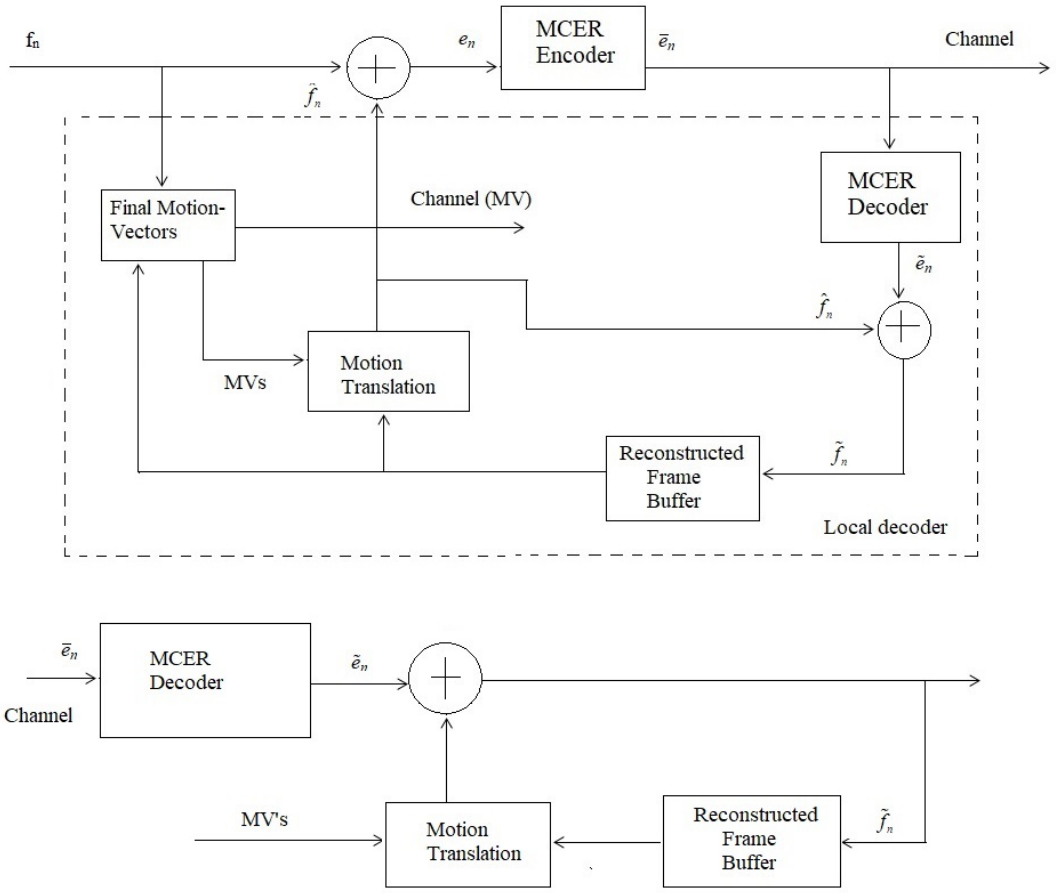


FIGURE 1.16: Motion-compensated video codec [10].

1.2.3 Distance measures

Three commonly used distance measures of image processing are discussed below. The widely accepted Mean Square Error (MSE) criterion is defined as [10]:

$$M_{mse} = MIN \sqrt{\sum_{i=1}^{b}\sum_{j=1}^{b}(f_n(x+i, y+j) - f_{n-1}(x+i-m_x, y+j-m_y))^2} \qquad (1.2)$$

where b denotes the block size, x and y represent the top left corner of the block under consideration, m_x, m_y constitute the coordinates of the motion vector $\vec{M}$. Eq. 1.2 evaluates the luminance difference of a given $b \times b$ blocks of f_n and f_{n-1} on a pixel by pixel basis. MC scheme finds the position, where this expression has a minimum over the search area of Fig. 1.15.

The mean absolute difference (MAD) criterion is defined as the sum of absolute differences rather than its second moment.

$$M_{mad} = MIN \sum_{i=1}^{n} \sum_{j=1}^{n} \left| f_n(x+i, y+j) - f_{n-1}(x+i-m_x, y+j-m_y) \right| \tag{1.3}$$

The pel difference classification (PDC) criterion of Eq. 1.4 compares the matching error for every pixel location of the block with respect to a preset threshold and classifies the pel as a "matching" or "mismatching" pel.

$$M_{pdc} = MIN \sum_{i=1}^{n} \sum_{j=1}^{n} T \left| f_n(x+i, y+j) - f_{n-1}(x+i-m_x, y+j-m_y) \right| \tag{1.4}$$

where, T denotes the threshold function defined as

$$T(y) = \begin{cases} 1, \text{ if } y > 0 \\ 0, \text{ otherwise.} \end{cases} \tag{1.5}$$

1.2.4 Picture reordering

Because of the conflicting requirements of random access and highly efficient coding, the MPEG suggested that not all pictures of a video sequence should be coded in the same way. They identified four types of the picture in a video sequence:

1. The first type is called I-picture, which is coded without reference to the previous picture. They provide the access points to the coded sequence for decoding. These pictures are intraframe coded as for JPEG.

2. The second type is the P-pictures, which are predictively coded with reference to the previous I or P-coded pictures. They themselves are used as a reference (anchor) for the coding of future pictures.

3. The third type is B-pictures, or bi-directionally coded pictures which may use past, future, or combinations of both for their predictions. This increases the motion compensation efficiency, since occluded parts of moving objects may be better compensated for from the future frame. B-pictures are never used for predictions.

Fig. 1.17 illustrates the relationship between these three types of pictures. Since B-pictures use I and P-pictures as predictors, they have to be coded later. This requires a reordering of incoming picture orders.

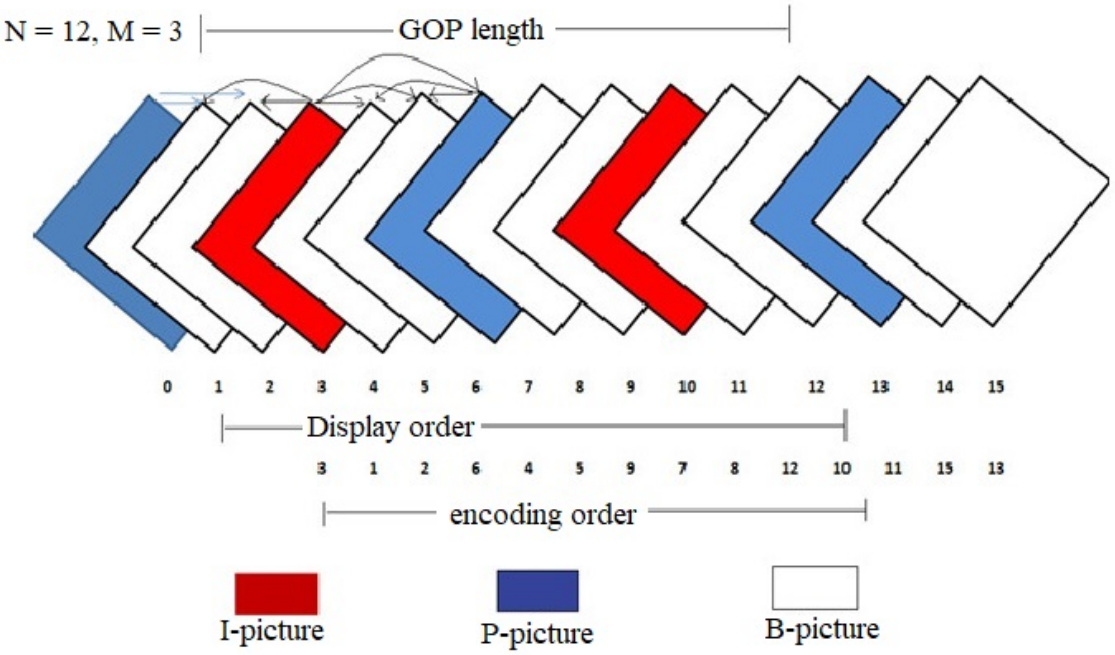


FIGURE 1.17: An example of MPEG -1 group of pictures [10].

4. The fourth type is the D-pictures. These are intraframe coded, where only the DC coefficients are retained. Hence, the picture quality is poor and normally used for fast forward. D-pictures are not part of the group of pictures (GOP).

1.2.5 Video structure

Group of pictures (GOP): A GOP is a series of one or more pictures to assist random access into the picture sequence [21]. The first coded picture is I- pictures. It is followed by an arrangement for P and B-pictures as shown in Fig. 1.17.

The GOP length is normally defined as the distance between I-pictures, which is represented by parameter N in the standard codecs. The distance between anchor I/P to P-pictures is represented by M. In Fig. 1.17, N = 12, M = 3.

1.3 Objectives

The main objective of our thesis work is to detect movement epentheses in continuous sign and fingerspelling sequences for efficient spotting of signs and fingerspellings. As no significant attempt has been made to deal with sign language spotting in the compressed domain, so another objective of our work is to develop sign language spotting frameworks fully or partially in the compressed domain. One more objective of our work is to remove the moving components arising due to camera motions in the compressed domain for sign language videos.

1.4 Outline of the thesis

The thesis mainly addresses the problem of continuous sign language spotting for the development of an efficient SLR system. In **Chapter 1**, we introduce coarticulation and *me* problems that arise in continuous sign language. Different works that have been reported on sign language spotting are reviewed in **Chapter 2**. Then the shortcomings of the reported works and advantages of using compressed domain analysis are discussed. In **Chapter 3**, a novel spatio-temporal Markov random field (ST-MRF) based framework for spotting of continuous fingerspelling sequence is reported. The framework is based on MVs extracted from H.264/AVC compressed videos. In **Chapter 4**, the design of a continuous fingerspelling recognition system in an uncompressed domain that segments a fingerspelling sequence into meaningful extracts and non-sign patterns and thereby recognizes the meaningful signs is reported. In **Chapter 5**, a work on continuous sign language spotting based on hidden Markov model (HMM) with Gaussian emission probability is reported. The feature set is comprised of features extracted from compressed as well as uncompressed domain analysis of H.264/AVC compressed sign videos. **Chapter 6**, concludes the thesis along with the future scopes of the work.

Chapter 2

Literature Survey

The continuous Sign Language recognition (SLR) system suffers from a problem called movement epenthesis (me) which involves certain intermediate connecting movement between two consecutive signs. In this chapter, a novel framework for spotting of continuous fingerspelling sequence is proposed, which can directly extract motion information of signs from a compressed video. The framework is based on motion vectors extracted from H.264/AVC compressed videos. A spatio-temporal Markov random field (ST-MRF) based model is employed to model non-rigid motions of fingers as "sign" or "me". The proposed framework is tested on several sign language videos encoded with an H.264/AVC JM encoder and the accuracy of spotting was found to be around 75%.

2.1 Literature survey

Sign spotting is a process of extraction of sign components from continuous sign language as well as fingerspelling sequence videos by removing *me* components as shown in Fig

2.1. Most of the approaches for continuous sign language spotting are mainly based on three approaches:

- Velocity-based approaches
- Template-matching based and
- Modeling of sign and *me*-based approaches.

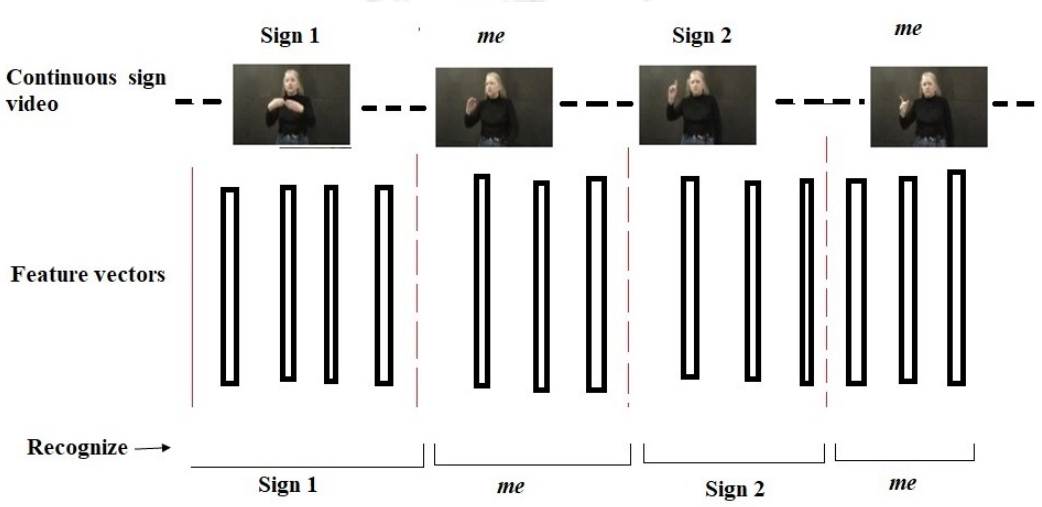


FIGURE 2.1: Continuous sign language spotting.

In the first category of approaches, Oz. et al. [22] proposed an isolated sign language recognition framework that detected "signing" and "non-signing" segments using a velocity network. The velocity network classified a "signing" segment when the hand first showed a change in velocity until it became low velocity. Bhuyan et al. [23] proposed a gesture spotting framework using a novel set of features. They proposed (a) orientation and (b) length of an ellipse least-squares fitted to motion-trajectory points and (c) the position of the hand. Yang and Sarkar [24] proposed an approach for segmentation of signs from a continuous stream by identifying the keyframes and then finding out the short-term motion of salient corner points over these frames. In [25], Choudhury et al. used height of the hand trajectory as a salient feature for separating the meaningful signs from the movement epenthesis patterns. Furthermore, they incorporated a unique set of spatial and temporal features for efficient recognition of the signs encapsulated within the continuous sequence.

Dynamic Time Warping (DTW) is a widely used algorithm for template-matching based approaches. It is a well-known technique to find out an optimal alignment between two given (time-dependent) sequences under certain restrictions (Fig. 2.2). Yang et al. [26] addressed a level building approach to simultaneously segment and match signs to continuous sign language sentences in the presence of *me* using DTW based approach. They further extended their work in [26] where they had a model base that consisted of all actual model signs, but not *me*. During the search for an optimal sign sequence in a sentence, if a frame matched with any of these models, then it was considered as a sign frame, otherwise as a *me* frame. This work was modified by Yang et al. [15] by replacing the DTW algorithm with Hidden Markov Model (HMM) to implement the level building algorithm. Ong et al. [27] proposed to model the spatio-temporal signatures of a sign using an extension of sequential patterns that contained temporal intervals called Sequential Interval Patterns (SIP). They then proposed a novel multi-class classifier that organized different sequential interval patterns in a hierarchical tree structure called a Hierarchical SIP Tree (HSP-Tree). Multiple trees were then combined together into a forest of HSP-Trees resulting in a strong classifier that could be used to spot signs. They obtained an accuracy of 71.1%. Elakkiya and Selvamani [28] proposed a novel subunit sign modeling framework using a novel methodology of the Enhanced Dynamic Programming (EDP) approach in subunit sign modeling. This EDP framework worked with a combination of DTW and spatiotemporal clustering techniques. The DTW was used as a distance measure to compute the distance between two adjacent signs in sign trajectories. This distance was used as a temporal feature vector during the clustering of spatial feature vectors using Minimum Entropy Clustering (MEC). This process was done recursively to cluster all the epentheses movements dynamically without using any explicit or implicit modelling and an average accuracy of 98.9% was achieved. Tang et al. [29] proposed a Structured Dynamic Time Warping (SDTW) approach for continuous hand trajectory recognition. They first proposed an automatic continuous trajectory segmentation approach that combined templates and velocity information to spot the beginning and ending points in hand gesture trajectories. Then they assigned different weights to feature sequences based on the structured information, from the positions of corner points in the arbitrary trajectories. Finally, they evaluated the SDTW on the

Continuous Letter Trajectory (CLT) database. Zadghorban and Nahvi [30] provided a system based on machine vision to recognize the signs in continuous Persian sign language video. The system generally consisted of two main phases: sign word extraction and their classification. They decomposed sign language video into the sign words using motion and hand shape features. In the classification phase, separated words were classified and recognized using a hidden Markov model and a hybrid KNN-DTW algorithm, respectively. They achieved an average rate of 93.73% for accurate boundary detection.

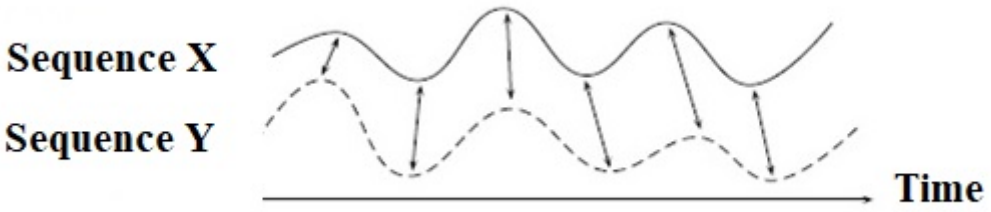


FIGURE 2.2: Dynamic time warping [11].

In model-based approaches, most of the methods employ models for describing each sign and a separate model for *me* to segment out the required sign frames. But to obtain enough training data to model *me* is a real issue [26]. So some of the model-based approaches do not have an explicit model for *me*. Lee and Kim [12] proposed an HMM-based threshold approach for gesture recognition where they used a threshold model by combining all the gesture models as shown in Fig. 2.3. Yang et al. [31] proposed a

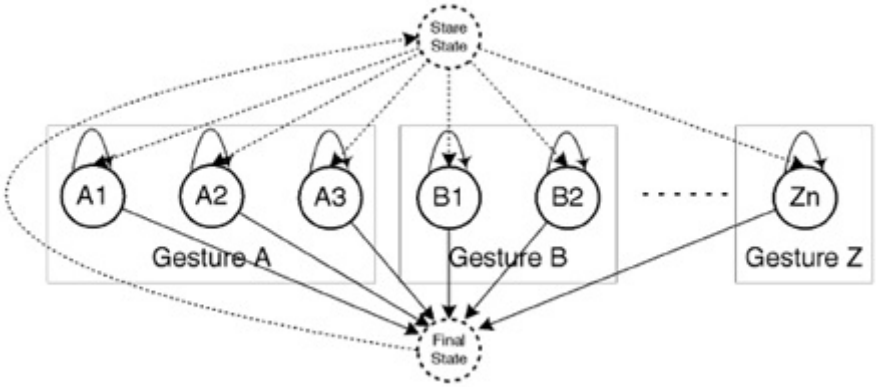


FIGURE 2.3: A simplified version of the threshold model [12].

conditional random field (CRF) based adaptive threshold model for segmenting out signs in vocabulary and non-sign patterns from continuous ASL sign sentences. They adopted a two-layer CRF architecture where in the first layer, a threshold model with CRF was

used to discriminate in-vocabulary signs and non-sign patterns; and in the second layer, a CRF is applied to find subsign patterns. They obtained a success rate of around 87% for spotting signs from continuous data. In [32], Kelly et al. proposed an HMM-based gesture recognition system that accurately classified a given gesture sequence as one of the pre-trained gestures as well as calculated the probability that the given gesture sequence was a *me* or not. In [33], Yang and Lee incorporated a two-layer CRF consists of a conventional CRF and a threshold CRF (TCRF) to distinguish signs, fingerspellings and non-sign patterns. Further, they used the motion of fingerspellings as a cue for discriminating among fingerspellings having similar handshapes and dissimilar hand motions. Kim et al. [1] developed a semi-Markov CRF approach for automatic recognition of fingerspelled words of ASL by defining feature functions namely language model feature, baseline consistency feature, hand shape classifier-based feature and, peak detection feature to catch the quick, minute motions of fingers. Ricco and Tomasi [34] put forward a principle for recognizing a dynamic range of fingerspelling sequences of ASL by modeling the transitions between letters instead of recognizing the isolated static handshapes for each letter. They demonstrated that the elimination of an explicit segmentation step for letters, which is generally error-prone due to high speeds used by native signers, can improve the recognition accuracy. Additionally, the inclusion of multiple HMM states for each letter based on surrounding context makes the method able to differentiate between hand-shapes caused by coarticulation. Kumar et al. [35] also proposed a two-stage framework for recognizing signs and fingerspellings using a Leap motion sensor. In the first stage, signs and fingerspellings were discriminated using a support vector machine (SVM) classifier. In the second stage, the discriminated gestures were recognized using two separate Bidirectional Long Short-term Memory Neural Networks (BLSTM-NN). They obtained an overall accuracy of 63.57% for both types of gestures. Wang et al. [36] proposed a Hierarchical Grassmann Covariance Matrix (HGCM) model for sign description. Furthermore, a Multi-Temporal Belief Propagation (MTBP) based segmentation approach was presented for continuous sequence spotting. A sign was represented by multiple covariance matrices, followed by evaluating and selecting their most significant singular vectors. These covariance matrices were transformed into a more compact and discriminative HGCM, which was formulated on the Grassmann Manifold.

Continuous sign sequences could be recognized frame-by-frame using the HGCM model, before being optimized by MTBP, which was a carefully designed graphic model. Jebali et al. [37] presented a computer vision-based system to recognize the signs in continuous sign language video. This system was based on two main phases : sign words extraction and their classification. They presented a new algorithm able to detect accurate word boundaries in a continuous sign language video. Using hand shape and motion features, this algorithm extracted isolated signs from video and it showed better efficiency. In the recognition phase, the extracted signs were classified and recognized using HMM and achieved recognition accuracy of 95.18% for one-handed gestures and 93.87% for two-handed gestures.

Apart from these, some studies utilized explicit segmentation methods such as manual gesture-based commands, physical buttons, voice signals, etc. for indicating the start and end of signs in the fingerspelling sequence. In [1], Kim et al. used an explicit partition approach for indicating the start and end (or boundary) of each sign by pressing a button. Although methods using explicit preambles provide an accurate indication of the start and end of sign frames, however, this process is cumbersome and hinders the naturalistic approach of signing. Hassan et al. [38] presented a comprehensive comparison between two different recognition techniques for continuous Arabic SLR, namely a Modified k-Nearest Neighbor which is suitable for sequential data and Hidden Markov Models (HMMs) techniques based on two different toolkits. They used database collected using both motion tracker and sensor gloves. From the experimental works, they found that the modified KNN solution was inferior to HMMs in terms of the computational time required for classification.

Recently, deep learning (DL) based frameworks have been mostly implemented for recognition of isolated signs, although a few have considered it for continuous sign language recognition. DL uses multi-layered neural networks (as shown in Fig. 2.4) which go deep into the features. Cui et al. [39] developed a framework for continuous SLR using deep neural networks. Shi et al. [40] proposed an ASL fingerspelling sequence recognition system "in the wild" using collected videos from websites. The system consisted of a signing hand detector followed by a sequence recognizer. The hand detection network

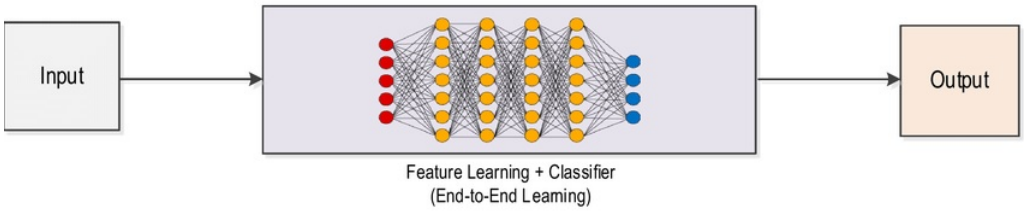


FIGURE 2.4: Deep neural networks for continuous SLR [13].

was implemented using Faster Region-based Convolutional Neural Network (R-CNN) object detector and sequence recognizer was based on connectionist temporal classification (CTC)-based models. The accuracy obtained was around 42%. Jalal et al. [41] presented a capsule-based deep neural network (DNN) sign posture translator for isolated ASL fingerspelling with an accuracy of around 99%. Papadimitriou and Potamianos [42] proposed a system that recognized 24 static fingerspelled alphabet signs of ASL video data. Their method combined DL with traditional image processing methods where face detection was done to drive skin tone-based hand segmentation, followed by motion tracking of the segmented skin regions. These regions were then classified by a variant of convolutional neural network (CNN) where linear convolution was substituted by a quadratic function which improved the performance significantly. Nguyen and Do [43] developed an isolated ASL fingerspelling alphabet (excluding 'J' and 'Z') identification system using image processing technique, supervised machine learning, and DL. The system used CNN as a feature extractor for a multiclass linear SVM classifier and obtained a recognition rate of 98.30%. Suri and Gupta [44] proposed a novel one-dimensional deep capsule network (CapsNet) architecture for continuous Indian Sign Language recognition by means of signals obtained from a custom designed wearable Inertial Measurement Unit (IMU) system. The performance of the proposed CapsNet architecture was assessed by altering the dynamic routing between capsule layers. The proposed CapsNet yielded improved accuracy values of 94% for 3 routings and 92.50% for 5 routings. Elakkiya et al. [45] addressed the characterization of manual and non-manual gestures in recognizing the sign language gestures from continuous video sequences. They introduced a novel hyper-parameter based optimized Generative Adversarial Networks (H-GANs) to classify the sign gestures, and it worked in three phases. In phase-I, it adapted the stacked variational auto-encoders (SVAE) and Principal Component Analysis (PCA) to get the pre-tuned data with reduced feature dimensions. In Phase-II, the H-GANs employed Deep Long

Short Term Memory (LSTM) as a generator and LSTM with 3D Convolutional Neural Network (3D-CNN) as a discriminator. The generator generated random sequences with noise from the real sequence of frames, and the discriminator detected and classified the real frames of sign gestures. In Phase-III, the proposed approach employed Deep Reinforcement Learning (DRL) for hyperparameter optimization and regularization. By getting the reward points, Proximal Policy Optimization (PPO) optimized the hyperparameters, and Bayesian Optimization (BO) regularized the hyperparameters. Cheng et al. [46] proposed a fully convolutional network (FCN) for online SLR to concurrently learn spatial and temporal features from weakly annotated video sequences with only sentence-level annotations given. A gloss feature enhancement (GFE) module was introduced in the proposed network to enforce better sequence alignment learning. The proposed network was end-to-end trainable without any pre-training. Koller et al. [47] introduced the end-to-end embedding of CNN into a HMM to improve the continuous SL recognition task. The hybrid CNN-HMM combined the strong discriminative abilities of CNNs with the sequence modelling capabilities of HMMs. Jiang et al. [48] proposed a method of one-shot sign spotting, i.e. given an example of an isolated sign (query), they wanted to identify whether/where this sign appeared in a continuous, co-articulated sign language video (target). To achieve this goal, they proposed a transformer-based network, called Sign-Lookup. They employed 3D CNNs to extract spatio-temporal representations from video clips. To solve the temporal scale discrepancies between the query and the target videos, they constructed multiple queries from a single video clip using different frame-level strides. Self-attention was applied across these query clips to simulate a continuous scale space. They also utilized another self-attention module on the target video to learn the context within the sequence. Finally mutual-attention was used to match the temporal scales to localize the query within the target sequence. Extensive experiments demonstrated that the proposed approach could not only reliably identify isolated signs in continuous videos, regardless of the signers' appearance, but could also generalize to different sign languages. By taking advantage of the attention mechanism and the adaptive features, their model achieved state-of-the-art performance on the sign spotting task with an accuracy as high as 96% on challenging benchmark datasets. Sharma et al. [49] proposed deep transfer learning for recognition of sign sequences in

sentences continuously signed in the Indian sign language using sufficient labelled data of isolated signs and a limited amount of labelled sentence data. The data was collected using multiple six degree-of-freedom inertial measurement units (IMUs) on both hands of the signer. The proposed deep learning model consisted of CNN, two bidirectional LSTM layers and connectionist temporal classification (CTC) to enable end-to-end sentence recognition without requiring the knowledge of sign boundaries. Koulierakis et al. [50] presented a recommendation system for (semi-) automatic annotation of sign language videos exploiting deep learning techniques, which handled handshape recognition in continuous signing data. Major tools in their approach had been the keypoint output of OpenPose and the use of HamNoSys in sign annotation of the training data. The system had been trained on the Danish Sign Language lexicon and has also been applied to POLYTROPON, a lexicon of the Greek Sign Language (GSL), for which they received satisfactory recognition results. Niu and Mak [51] proposed a novel stochastic modeling of various components of a continuous SLR system that was based on the transformer encoder and connectionist temporal classification (CTC). They modeled each sign gloss with multiple states, and the number of states was a categorical random variable that followed a learned probability distribution, providing stochastic fine-grained labels for training the CTC decoder. They further proposed a stochastic frame dropping mechanism and a gradient stopping method to deal with the severe overfitting problem in training the transformer model with CTC loss. These two methods also helped to reduce the training computation, both in terms of time and space, significantly.

Some works have been reported using depth images instead of color-based images due to their robustness against illumination and background variations [52]. Kang et al. [53] proposed a real-time isolated ASL fingerspelling recognition system for 31 alphabets using CNNs from the Depth map. Depth images were collected using Creative Senz3D camera and classified using Caffe implementation (CaffeNet) of the CNNs. The system achieved an accuracy of 99.99%. Tazhigaliyeva et al. [54] developed a Cyrillic manual alphabet recognition system using RGB-D data for SL interpreting robotic system. The RGB-D data were collected using a Leap Motion sensor. The learning architecture used two hierarchically arranged self-organizing neural networks and the accuracy obtained

was 93%. Aly et al. [52] proposed a user-independent ASL isolated fingerspelling recognition system using depth images captured from the low-cost Microsoft Kinect depth sensor. An unsupervised CNN using Principal Component Analysis Network (PCANet) was employed to describe segmented hand gestures. The extracted features were then recognized by using linear SVM classifier. The accuracy obtained was around 88.70%. Kumar et al. [55] proposed the use of graph matching (GM) to enable 3D motion capture for Indian sign language recognition. The continuous 3D dataset was recorded using a Vicon motion capture setup with eight cameras and a video camera involving the RGB color model. The 3D dataset contained 200 signs that formed meaningful sentences known as continuous sign frames (CSFs). Each frame in the sentence dataset was represented by a graph of 57 spatial points known as vertices and 56 joint pairs known as edges. The intergraph matching algorithm (IGM) was applied to two consecutive frames, and a threshold was set to extract high-motion sequences from sign videos. The same process was applied to query sign frames (QSFs) for enabling motion frame separation. QSFs were the testing input 3D sign video sequence. An adaptive graph matching (AGM) algorithm was proposed for obtaining the QSFs in the CSFs. IGM calculated the similarity between corresponding vertices and edges for consecutive QSFs and CSFs. However, for sign language 3D data, the IGM model produced negative matching because of the small variations between signs in most of the cases. This problem was solved by using an AGM model. In that model, each vertex and edge in a QSF and CSF were matched both spatially and temporally. The accuracies obtained were 99.30% and 98.21% for the same subject and cross subject respectively, for 3D sign.

2.2 Continuous sign language spotting in compressed domain

State-of-the-art sign language recognition approaches are mainly spatial domain-based. Spatial domain-based approaches exploit visual features such as shape, color, texture, Shift-invariant feature transform (SIFT), Speeded up robust feature (SURF), Principal component analysis, Linear discriminant analysis (LDA), frequency domain features such as Fourier descriptor etc. [56], [57]. Pixel domain methods are usually more precise

and reliable, but are computationally expensive [58]. On the other hand, compressed-domain approaches exploit inherent compressed domain data such as motion vectors (MVs), macroblocks (MBs), discrete cosine transform (DCT) coefficients, etc. These methods have low complexities, but suffer from poor localization of object boundaries and inconsistency in the number of object boundaries [20]. This is due to the fact that the MVs are calculated using block-based approaches which do not give accurate MVs [59] because they try to minimize the difference error between two blocks.

The motion information, including global camera motion and moving objects in video scenes, is very valuable for detecting semantic events such as walking, Violence or people-marching etc. Most of the existing works on motion analysis are based on visual features of frame images, that means a time-consuming uncompressing process is needed [60]. So for the applications on large-scale video corpus, a faster compressed domain method is necessary. A few compressed domain methods have been proposed, which use the motion vectors and/or DCT coefficients directly to avoid the IDCT process and motion compensation. Although these methods perform well in some special cases, they all have to face some difficulties in general applications. First, the DCT coefficients of interceded MBs are not calculated from the real pixel data, but from the residual errors of pixel data between the current MB and its reference region. Second, there are many intra-coded MBs in video streams which cannot provide motion information, especially, I-frames are totally intra-coded. The last and most important problem is that MVs are highly noisy. Motion vectors of MPEG videos are calculated by a fast MB matching algorithm in the encoder, which may not accord with the real motion of objects, especially in uniform (non-textured) regions. The solutions to these problems can strongly influence the performance of a compressed domain method [60]. Compressed domain approaches are used for many applications such as scene change detection [61], moving region segmentation [57], [58], video stabilization [62], action recognition [63], etc.

Fig. 2.5 shows the overall process of sign spotting in the compressed domain. At first, motion vectors (MVs) of each frame are extracted from the compressed video. MV-based processing block will process these MVs to extract the sign frames only by removing the *me* frames. After that, particular frames which are corresponding to signs only will

be fully decoded. After that, a low-level processing block will process these frames for finding some suitable features, and using these features, the high-level processing block will recognize the signs. This process will enhance the computational complexity.

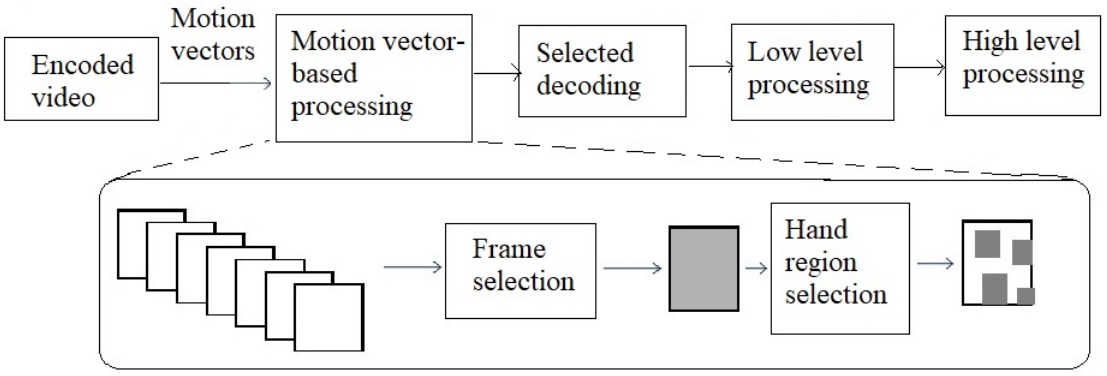


FIGURE 2.5: The overall process of sign language recognition in the compressed domain.

2.3 Motivation and problem formulation

The need for sign language recognition (SLR) systems is increasing in recent times, as they have become a key ingredient in the process of intercommunication between the hearing impaired and the hearing people. From the review of related works on continuous SLR and in the field of compressed domain video analysis, it is found that no significant attempt has been made to deal with SLR in the compressed domain. Moreover, compressed videos are now abundant everywhere and there are many advantages of using compressed domain analysis of videos. Sometimes, a fusion of both compressed and uncompressed domain features gives better results compared to individual domains. So, these factors have motivated us to deal with the problem of continuous sign language spotting in individual domains as well as the fusion of both.

The overall problem of sign language spotting for efficient continuous sign language recognition can be subdivided into two problems:

1. Detection of fingerspelling and *me* frames from continuous fingerspelling videos.
2. Detection of sign and *me* frames in case of continuous sign (the global movement) videos.

Moreover, most of the SLR systems have been reported for the recording of sign videos using static cameras only. As per our study, no work has been reported so far on SLR using a moving camera. In SLR systems, motion information of hands and fingers are considered important cues. The presence of camera motion often makes it difficult to find out motion information of the hands and the fingers which ultimately worsens the performance of the system [57]. So, to overcome this issue, we have also considered the problem of removing the moving components arising due to camera motion for sign language videos in the compressed domain.

TABLE 2.1: Summary of Sign Language recognition approaches

Lee and Kim [12]	Continuous hand gestures	Model-based	Continuous	An HMM-based threshold approach for gesture recognition	93.14%
Yang and Sarkar [24]	ASL	Velocity-based	Continuous	Segmentation of signs from a continuous stream by identifying the keyframes and then finding out the short-term motion of salient corner points over these frames	85%
Oz. et al. [22]	ASL	Velocity-based	Isolated	Detected "signing" and "non-signing" segments using a velocity network	95%
Yang et al. [26]	ASL	DTW	Continuous	A level building approach to simultaneously segment and match signs to continuous sign language sentences in the presence of *me*	80%
Kelly et al. [32]	ASL	Model-based	Continuous	An HMM-based gesture recognition system that accurately classified a given gesture sequence as sign or *me*.	0.949 (AUC)

TABLE 2.1: Summary of Sign Language recognition approaches (continued)

Ricco and Tomasi [34]	ASL	Model-based	Continuous	Modeled the transitions between letters instead of recognizing the isolated static handshapes.	94.75%
Yang et al. [26]	ASL	DTW	Continuous	A level building approach to simultaneously segment and match signs to continuous sign language sentences in the presence of *me*	80%
Yang and Lee [33]	ASL	Model-based	Continuous	A two-layer CRF to distinguish signs, fingerspellings and non-sign patterns.	83% for signs, 78% for fingerspellings
Tazhigaliyeva et al. [54]	Kazakh SL	Deep learning	Isolated	Used two hierarchically arranged self-organizing neural networks	93%
Kim et al. [1]	ASL	Model-based	Continuous	A semi-Markov CRF approach for automatic recognition of fingerspelled words.	11.6% (LER)

TABLE 2.1: Summary of Sign Language recognition approaches (continued)

Ong et al. [27]	German Sign Language	DTW	Continuous	model the spatio-temporal signatures of a sign using an extension of sequential patterns that contain temporal intervals called Sequential Interval Patterns (SIP).	71.4%
Bhuyan et al. [23]	Hand Gesture	Velocity-based	Continuous	(a) Orientation and (b) length of an ellipse least-squares fitted to motion-trajectory points	88.9%
Kang et al. [53]	ASL	Deep learning	Isolated	Used CNNs from the Depth map	99.99%
Yang et al. [15]	Chinese SL	DTW	Continuous	Replaced the DTW algorithm of [26] with Hidden Markov Model (HMM) to implement the level building algorithm.	12.20% (error rate)
Choudhury et al. [25]	ASL	Velocity-based	Continuous	Height of the hand trajectory as a salient feature for separating the meaningful signs from the movement epenthesis patterns.	92.8%

TABLE 2.1: Summary of Sign Language recognition approaches (continued)

Kumar et al. [35]	ASL	Model-based	Continuous	Two-stage framework for recognizing signs and fingerspellings using a Leap motion sensor.	63.57%
Tang et al. [29]	continuous hand trajectory recognition	DTW	Continuous	combined templates and velocity information to spot the beginning and ending points in hand gesture	88.80% (F score)
Elakkiya and Selvamani [28]	German Sign Language	DTW	Continuous	Model the spatio-temporal signatures of a sign using an extension of sequential patterns that contain temporal intervals called Sequential Interval Patterns (SIP).	71.4%
Kumar et al. [55]	Indian SL	Motion-based	Continuous	The intergraph matching algorithm (IGM) was applied to two consecutive frames	99.3% for same subject , 98.21% for cross subject.

TABLE 2.1: Summary of Sign Language recognition approaches (continued)

Koller et al. [47]	German SL	Deep learning	Continuous	Proposed end-to-end embedding of CNN into a HMM.	30.0% (WER) PHOENIX 12 dataset, 32.5 % (WER) PHOENIX 14 dataset, 7.4 % (WER) for SIGNUM dataset.
Zadghorban and Nahvi [30]	Persian SL	DTW	Continuous	Decomposed sign language video into the sign words using motion and hand shape features.	93.73%
Shi et al. [40]	ASL	Deep learning	Continuous	The hand detection network was implemented using Faster Region-based CNN (R-CNN) object detector and sequence recognizer was based on connectionist temporal classification (CTC)-based models.	42%
Jalal et al. [41]	ASL	Deep learning	Isolated	Capsule-based DNN sign posture translator.	99%

TABLE 2.1: Summary of Sign Language recognition approaches (continued)

Wang et al. [36]	Chinese SL	Model-based	Continuous	Multi-Temporal Belief Propagation (MTBP) based segmentation approach for continuous sequence spotting.	0.708 (Recall)%
Hassan et al. [38]	Arabian SL	Model-based	Continuous	A comprehensive comparison between Modified k-Nearest Neighbor and Hidden Markov Models (HMMs).	The modified KNN solution was inferior to HMMs
Cui et al. [39]	German SL	Deep learning	Continuous	Deep CNNs with stacked temporal fusion layers as the feature extraction module, and bi-directional RNNs as the sequence learning module.	2.80% (WER)
Papadimitriou and Potamianos [42]	ASL	Deep learning	Isolated	Combined DL with traditional image processing.	74.24%
Nguyen and Do [43]	ASL	Deep learning	Isolated	Used CNN as a feature extractor for a multiclass linear SVM classifier.	98.30%

TABLE 2.1: Summary of Sign Language recognition approaches (continued)

Suri and Gupta [44]	Indian SL	Deep learning	Continuous	Used deep capsule network (CapsNeT) architecture.	94% (for 3 routings), 92.50% (for 5 routings)
Aly et al. [52]	ASL	Deep learning	Isolated	CNN using Principal Component Analysis Network (PCANet)	88.70%
Cheng et al. [46]	Chinese SL, German SL	Deep learning	Continuous	Used fully convolutional network (FCN).	23.3% (WER) for development set, 25.1% (WER) for testing set.
Niu and Mak [51]	German SL	Deep learning	Continuous	Proposed stochastic modeling of various components of a continuous SLR system.	24.9% (WER) for dev set, 25.3% (WER) for test set.
Elakkiya and Selvamani [28]	German SL	DTW	Continuous	Model the spatio-temporal signatures of a sign using an extension of sequential patterns that contain temporal intervals called Sequential Interval Patterns (SIP).	71.4%

TABLE 2.1: Summary of Sign Language recognition approaches (continued)

Jiang et al. [48]	German SL	Deep learning	Continuous	Proposed a transformer-based network, called Sign-Lookup.	96.0% spotting accuracy
Sharma et al. [49]	Indian SL	Deep learning	Continuous	Proposed deep transfer learning.	89.9%
Koulierakis et al. [50]	Danish SL and Greek SL annotations	Deep learning	Continuous	Used OpenPose and HamNoSys.	95.7% (hand shape) 96.8% (hand location).
Jebali et al. [37]	ASL	Model-based	Continuous	Hand shape and motion features were used to extract isolated signs from video.	95.18% (for one-handed) 93.87 % (for two-handed)

Chapter 3

A Framework for Continuous Fingerspelling Spotting for H.264/AVC Compressed Videos using Spatio-Temporal Markov Random Field

The continuous Sign Language recognition (SLR) system suffers from a problem called movement epenthesis (me) which involves certain intermediate connecting movement between two consecutive signs. In this chapter, a novel framework for spotting of continuous fingerspelling sequence is proposed, which can directly extract motion information of signs from a compressed video. The framework is based on motion vectors extracted from H.264/AVC compressed videos. A spatio-temporal Markov random field (ST-MRF) based model is employed to model non-rigid motions of fingers as "sign" or "me". The proposed framework is tested on several sign language videos encoded with an H.264/AVC JM encoder and the accuracy of spotting was found to be around 75%.

3.1 Introduction

Sign language fingerspelling in particular comprises local hand motions, where only the fingers move during transition signing (except 'J' and 'Z') [3]. The automatic translation of sign language into English has been a topic of research in computer vision for over a decade. To find the required single frame corresponding to the letter, a motion-based technique was used. Researchers applied the threshold to the motion of the fingers to find the low motion frame which corresponds to the required frame. Several techniques have been used ranging from the total energy in the difference of the two consecutive frames [64] to the velocity by the Lucas-Kanade optical flow method [65]. Ricco and Tomasi [34] proposed a technique where they recognized transition between letters rather than the letters themselves. This technique can find the most important video frame for recognition. Yang and Lee [3] proposed a two-layer conditional random field (CRF) consisting of a threshold model with CRFs (T-CRFs) and a conventional CRF. The T-CRF can select sign, fingerspelling and non-sign patterns using the information of hand motions and hand locations. Yang et al. [2] developed a framework for implicit segmentation of signs without modeling *me* for continuous SLR. The entire process was embedded in a dynamic programming-based level building algorithm coupled with a grammar model. Kim et al. [1] suggested a method for unconstrained fingerspelling recognition based on a semi-Markov random field. They used feature functions based on the scores of letter classifiers, as well as the classifiers of handshape features suggested by linguistic research on American sign language (ASL). Chuan et al. [66] proposed an ASL recognition system using k-nearest neighbor and support vector machine (SVM) for classification of 26 letters of English alphabets. Kane and Khanna [67] proposed a framework for live isolated fingerspelling recognition with a modified shape matrix, which could capture the salience of the fingerspelling postures by a precise sampling of contours and regions. Avola et al. [68] used Recurrent Neural Network (RNN) for ASL recognition by employing spacing between finger bones as features acquired by a leap motion controller sensor.

Most of the state-of-the-art approaches on vision-based SLR systems were developed for uncompressed raw videos. The importance of ASL videos in compressed form

was explained in [69]. Real-time, two-way transmission of ASL videos over the cellular networks is important for natural SL-based interactions. There are many advantages of using compressed videos for storage and transmission [70], [71]. Some works have been reported on the compression of SL videos for making real-time SL video-based communication through mobile phones. Chon et al. [70] in their video compression project MobileASL, proposed their prototype system with the goal to enable people to access through SL communication over cell phones in real-time. Chon et al. [71] again investigated some video-compression techniques for the prevention of data loss in the case of SL conversation through cellular networks.

To address the drawbacks of the existing algorithms and utilize the advantages of working in compressed videos, a compressed domain framework for continuous fingerspelling spotting using a spatio-temporal Markov random field (ST-MRF) based model is proposed [72]. The idea behind the work is to separate the sign and *me* frames in continuous fingerspelling sequences in the compressed domain by partially decoding the compressed videos. Partial decoding is done to extract MVs and MB sizes that are inherent in the video itself. The framework uses the MVs extracted from H.264/AVC compressed videos as features. Once the sign frames are separated out from the video, these frames can be fully decoded to recognize the complete sign in the subsequent stages. Thus, it will make the system faster as compared to uncompressed domain approaches because compressed domain approaches deal with a few sets of data. So, the designing of MV-based ST-MRF model for removal of outliers followed by sign spotting in the compressed domain is an important aspect of this work. For considering camera motions, a novel framework for global motion estimation (GME) and global motion compensation (GMC) in the compressed domain is also proposed.

The chapter is organized as follows: Section 3.2 describes the outliers present in compressed domain videos, Section 3.3 describes the proposed framework for removal of outliers, Section 3.4 describes the proposed ST-MRF based framework for spotting of continuous fingerspelling sequences, Section 3.5 describes the results obtained and finally in Section 3.6, conclusions drawn from the work are discussed.

3.2 Outliers

It is observed that there may be some stray MVs present in the parts where significant moving objects actually do not exist in a video as shown in Fig. 3.1. These are termed as outliers which actually represent the unwanted MVs. The outliers can be of two types:

- Type 1: "Noisy" MVs, which are introduced during the encoding process, as the encoder's motion estimation algorithm falsely identifies local motions especially in the homogeneous regions [58]. These outliers are often found in areas where motion estimation fails to capture real motion, such as regions with little or no texture, boundary regions of moving objects, and regions with repetitive texture patterns [73]. These outliers are present in both cases i.e. static camera and moving camera. Fig. 3.1 shows this type of outliers for a static camera for a sign video.
- Type 2: This type of outlier is present in the case of moving cameras only. These outliers are the MVs from any object that do not fit the eight-parameters motion model. Hence there are two sources of Type –2 outliers [73] :
  1. Moving objects (i.e. local motion) in the video relative to the background.
  2. Static objects that are not far from the camera.

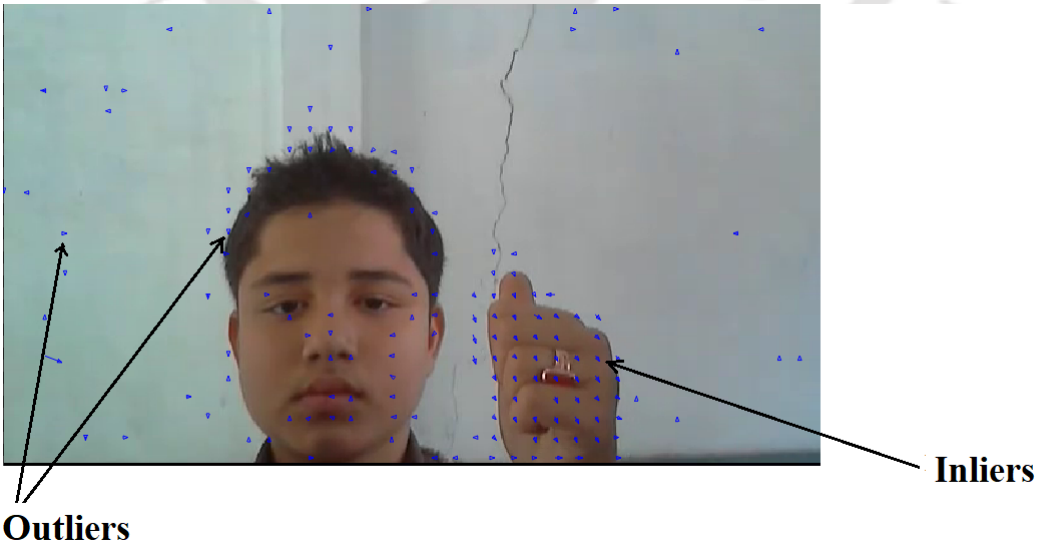


FIGURE 3.1: Outliers and inliers in a frame.

3.3 Proposed outlier removal framework for fingerspelling spotting

Although the MV-based approaches are computationally efficient, the presence of outliers may affect the overall performance of the system. So, the removal of outliers is essential to identify the actual MVs associated with a particular fingerspelling. Fig. 3.2 shows the proposed outlier removal framework for spotting of fingerspellings in continuous fingerspelling sequences. The framework is based on the MVs extracted from the video itself. Firstly, the H.264/AVC compressed bitstreams are partially decoded to extract the MVs of the frames. While capturing a video, the camera may be static, moving, or a combination of both. Okade et. al [62] used selective video stabilization which saved a considerable amount of computational time. So, we have taken the idea from their work to determine whether camera motion is present or not in the video which is described below.

The norm and of a vector $\overrightarrow{\mathrm{MV}}$ is defined as

$$\left|\overrightarrow{\mathrm{MV}}\right| = \sqrt{{\mathrm{MV_x}}^2 + {\mathrm{MV_y}}^2} \tag{3.1}$$

where, $\overrightarrow{\mathrm{MV_x}}$ and $\overrightarrow{\mathrm{MV_y}}$ are the X and Y components of $\overrightarrow{\mathrm{MV}}$.

Here, the histogram of the norm of MVs for a frame is used as a feature to determine whether camera motion is present or not for that particular frame. In the case of a static camera (i.e. without global motion), the peak of the histogram will be in the first bin of the histogram (i.e. '0') as shown in Fig. 3.3(a). This is because as most of the pixels in the frame (background) are static and hence their MVs are zero. On the other hand, in the case of a moving camera, the majority of the pixels will have some MV except those having local motion. So, the peak of the histogram will shift from the first bin as shown in Fig. 3.3(b). So, by using this feature, we can easily detect whether global motion is present or not for that particular frame. If the global motion is present, then the frame has to undergo the global motion estimation (GME) and global motion compensation (GMC) otherwise, it will directly proceed to the "Type 1" outlier removal block.

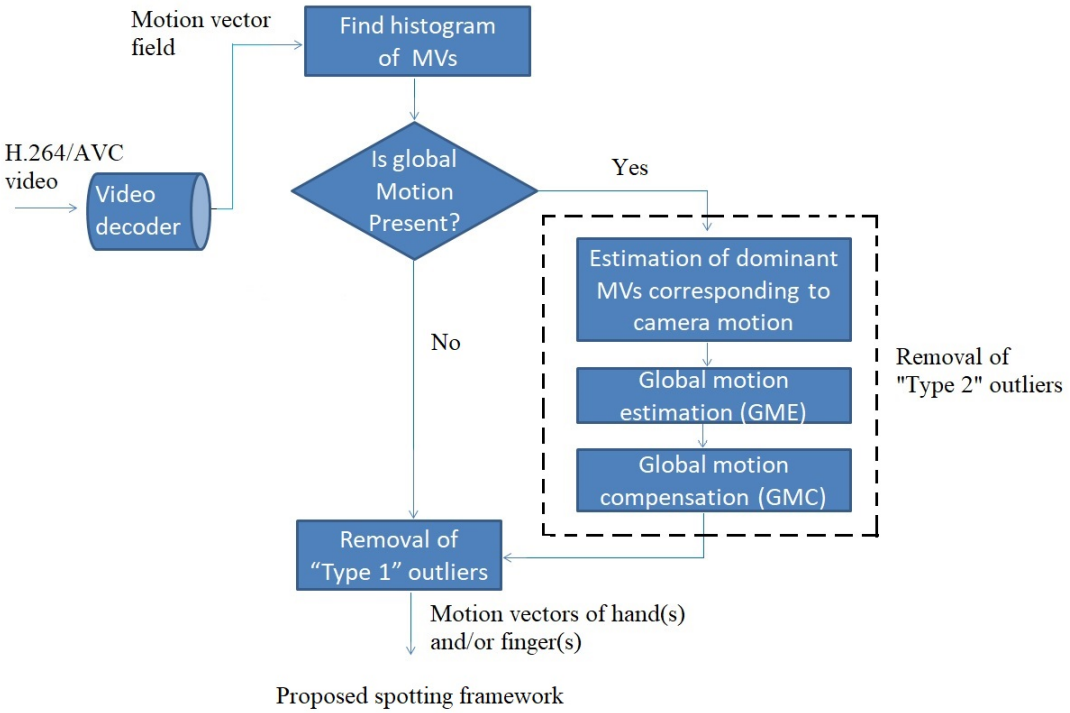


FIGURE 3.2: Proposed outlier removal framework for spotting of fingerspellings.

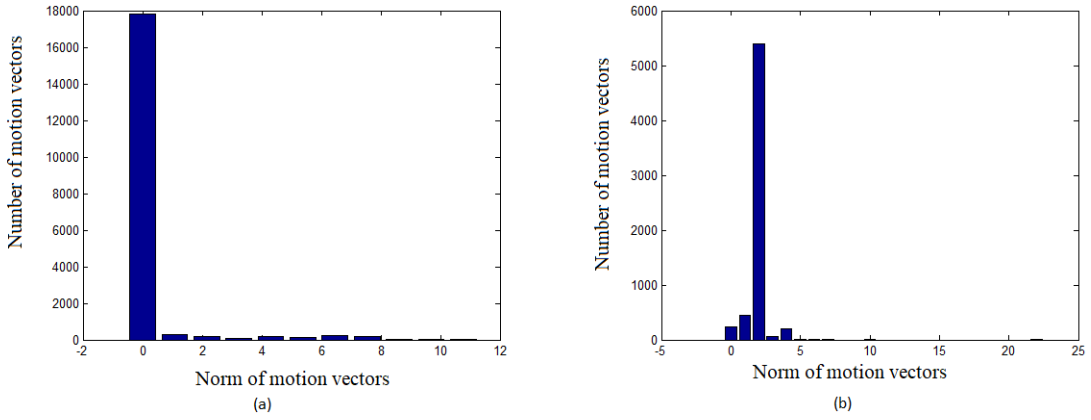


FIGURE 3.3: Histogram of the norm of motion vectors for (a) Static camera (b) Moving camera.

3.3.1 Removal of "Type 1" outliers using MVs from a frame

In [73], Cheng and Bajic used a cascade-of-rejectors for removing outlier MVs in order to achieve efficient and accurate global motion estimation. Our approach is based on their method, but with different filtering techniques[1]. The proposed cascade consists of three

[1]The related work is published in IEEE International Conference on Recent Advances in Intelligent Computational Systems (RAICS), 2018 (Refer to *List of Publications* page for details).

filters as shown in Fig. 3.4.

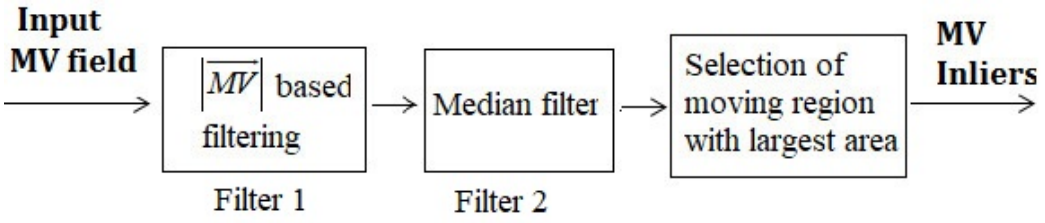


FIGURE 3.4: Proposed MV outliers removal cascade.

Input MV field is subjected to test in the first filter i.e. Filter 1 and then the MVs declared as inliers are further tested in Filter 2. To test each input MV, the filters in the cascade employs the following strategy.

Let $\overrightarrow{\mathrm{MV}_\mathrm{i}}$ be the input MV to be tested in Filter j, where $\mathrm{j} \in \{1, 2\}$. Associated with MV_i is the set $\mathrm{S}_\mathrm{i}^\mathrm{j}$ of MVs filtered out from Filter j. In Filter 1, we use a histogram of the norm of MVs as shown in Fig. 3.5. It is observed that the norm of MVs of the dominant object (palm and/or fingers in the case of sign language videos) is generally greater than those of the outliers. So, in order to remove the outliers, we select a threshold T, based on the processing of the histograms. This T is adaptive to each frame under consideration. The procedure for the selection of T is discussed below:

To find T, we use some statistical parameters of histograms [74]. Let r denotes a discrete random variable representing discrete MV magnitude in the range $[0, \mathrm{L}-1]$ and let $\mathrm{p}(\mathrm{r_i})$ denotes the normalized histogram component corresponding to the i^{th} value of r. We may view $\mathrm{p}(\mathrm{r_i})$ as an estimate of the probability of occurrence of MV magnitude level $\mathrm{r_i}$. The n^{th} moment of r about its mean is given as

$$\mu_\mathrm{n}(\mathrm{r}) = \sum_{\mathrm{i}=0}^{\mathrm{L}-1} (\mathrm{r_i} - \mathrm{m})^\mathrm{n} \mathrm{p}(\mathrm{r_i}) \tag{3.2}$$

where m is the mean value of r (its average magnitude of MVs), which is given by

$$\mathrm{m} = \sum_{\mathrm{i}=0}^{\mathrm{L}-1} \mathrm{r_i} \mathrm{p}(\mathrm{r_i}) \tag{3.3}$$

We use the mean, m as the global threshold T. In the proposed method, we firstly remove the first bin i.e., "0" bin from the initial histogram as shown in Fig. 3.5(a). This

is because it represents the background or static regions and then the new histogram is found as shown in Fig. 3.5(b). From this histogram, we find the normalized histogram $p(r_i)$ and calculate the mean as given by Eq. 3.3. The position of T is shown in Fig. 3.5 (b). Then we consider the conditions given in Eq. 3.4 for removal of the outliers based on this T.

$$\overrightarrow{MV_i} = \begin{cases} \overrightarrow{MV_i}, \text{ if } \left|\overrightarrow{MV_i}\right| \geq T \\ 0, \text{ if } \left|\overrightarrow{MV_i}\right| < T \end{cases} \tag{3.4}$$

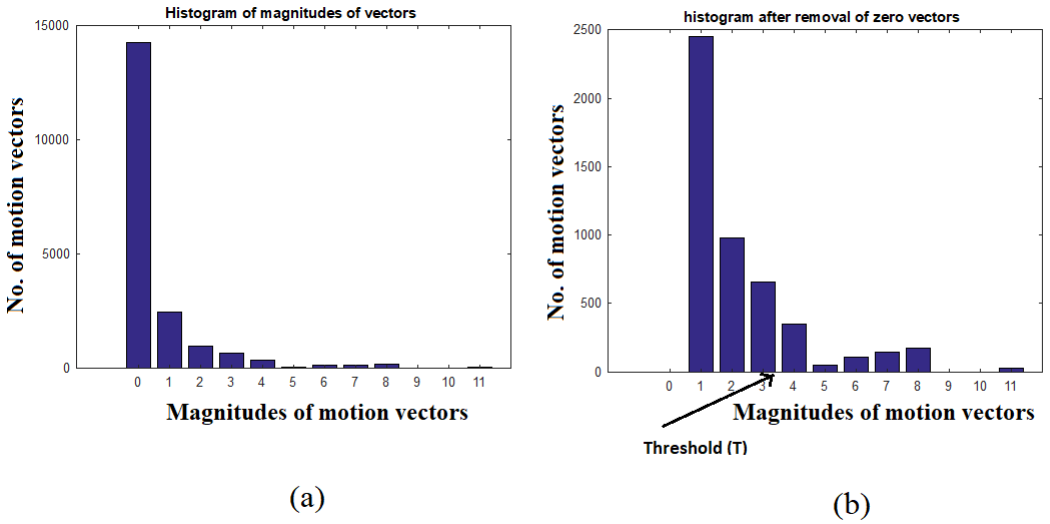


FIGURE 3.5: Histogram of motion vectors (norm).

The filtered MVs are then represented as set S_i^1 . The S_i^1 is then passed to Filter 2. This thresholding operation also creates a binary mask $M_{i,1}$ based on the above operation on $\overrightarrow{MV_i}$ as shown in Eq. 3.5. This mask $M_{i,1}$ will be used in the subsequent steps for further processing.

$$M_{i,1} = \begin{cases} 1, \text{if } \left|\overrightarrow{MV_i}\right| \geq T \\ 0, \text{if } \left|\overrightarrow{MV_i}\right| < T \end{cases} \tag{3.5}$$

After applying the first filter, there still remain some outlier residues whose characteristics are like salt and pepper noise as shown in Fig. 3.6. So, to remove these outliers, a median filter of window size 8×8 is used. Here, the Region of Interest (ROI)-based median filter is considered as the median filtering is performed on the binary mask $M_{i,1}$. This filtering saves computational time.

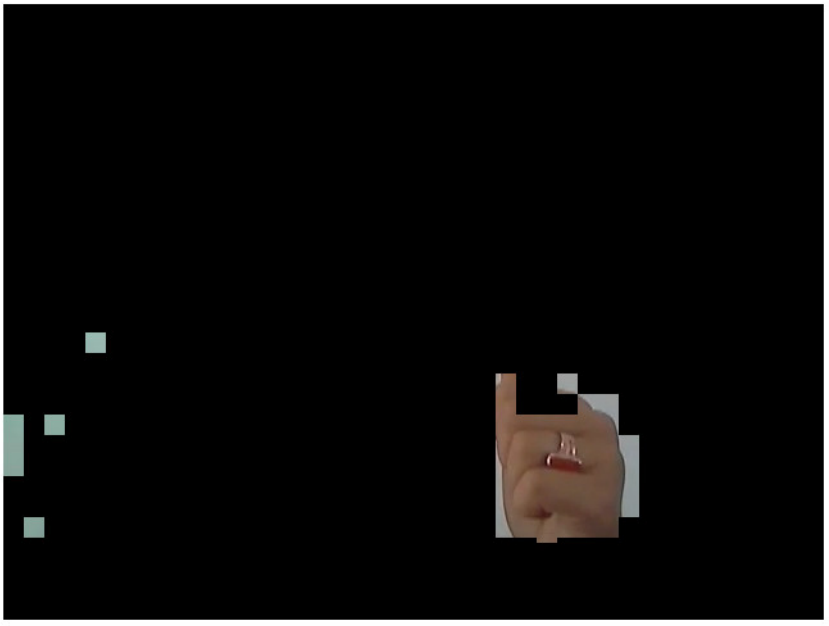


FIGURE 3.6: Output of Filter 1.

Selection of the dominant moving region

The above mentioned median filter significantly removes the remaining outliers from S_i^1 and form S_i^2 . But S_i^2 contains certain outliers in some worst cases which may affect the overall result. Fig. 3.7 shows such outliers. To remove these outliers, we calculate the number of objects in the frame and finally select the object with the maximum area. We consider an 8-connected parameter for finding the number of objects. In an 8-connected neighborhood, all of the pixels that touch the pixel of interest are considered, including those on the diagonals as shown in Fig. 3.8. This means that if two adjoining pixels are connected, they are part of the same object, regardless of whether they are connected along the horizontal, vertical, or diagonal direction. All these processings are done on the binary mask $M_{i,1}$ obtained in Section 3.2. Here, we consider the fact that our palm (and/or fingers) will be the dominant moving part, so it will have the maximum area.

The proposed filtering technique for removal of "Type 1" outliers was tested on several standard H.264 4:2:0 sequences at VGA (640 × 480) resolution all with a frame rate of 30 frames per second. All sequences were encoded using H.264/AVC JM V.18.0 encoder [75] with baseline profile and at different bitrates, with GOP structure IPPP..,

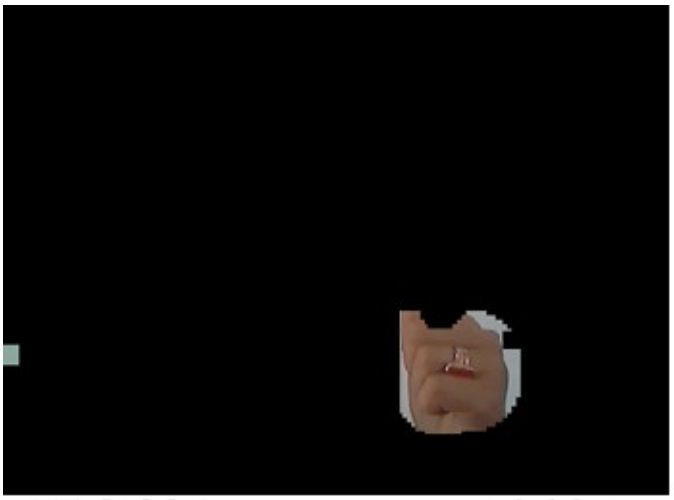


FIGURE 3.7: Output of Filter 2.

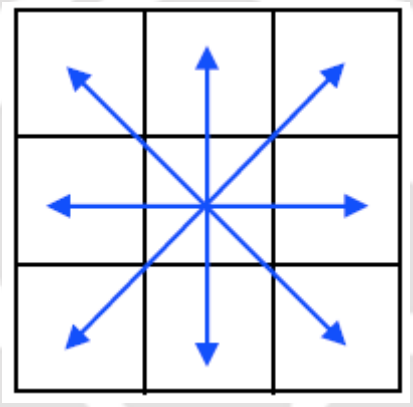


FIGURE 3.8: 8-connected neighborhood.

i.e., the first frame was coded as Intra (I), and the subsequent frames were coded predictively (P) with GOP size 30. Block size of 4×4 was used, as this was the lowest unit possible and was more likely to represent the moving objects. In order to have a uniformly sampled motion vector field, all the MVs of variant block sizes were re-mapped to 4×4 block sizes to maintain consistency [58]. The whole system was implemented using MATLAB 7.14 (R2012a) on an Intel Core i3-330 M CPU running at 2.13 GHz with 2 GB RAM.

For the first case, we considered a video of fingerspelling sequence "A-E-O" with a static camera articulated by a native signer. We considered an intermediate frame with "frame # 25" to show some intermediate results. After extracting the MVs for the frame, we calculated the histograms of the magnitude of the MVs. For the magnitude histogram, we considered the range of MVs from "zero" to maximum magnitude of MVs present in

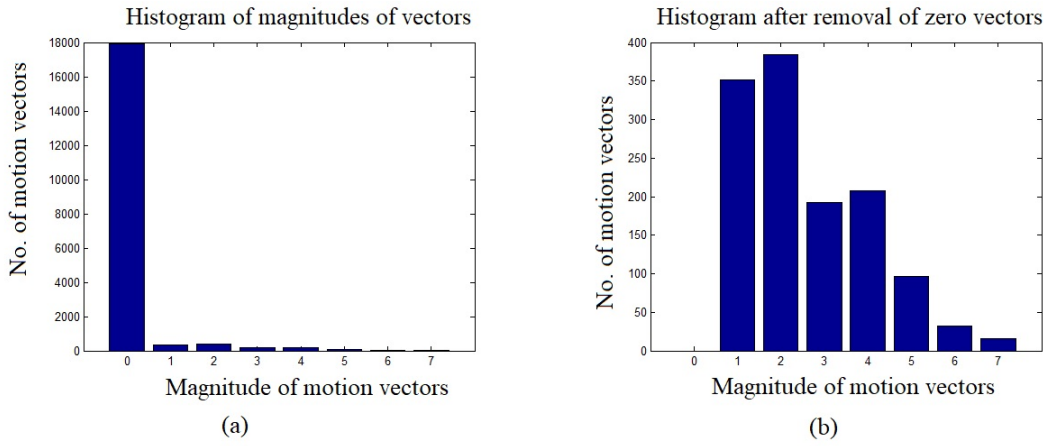


FIGURE 3.9: Characteristics of MVs for the fingerspelling sequence "A-E-O" for "frame # 25" (a) Histogram of magnitudes of MVs (b) Histogram of the magnitude of MVs after removal of "0" bin.

the frame with a bin size of "1". The histogram is shown in Fig. 3.9(a). After that, we removed the dominant bin i.e. "0" bin; the reason for doing this is already explained in Section 3.2. The new histogram after removing the "0" bin is shown in Fig. 3.9(b). This step has to be followed after each filtering as the values of outlier MVs become zero after each filtering.

The resultant frame containing the sub-MBs having non-zero MVs as shown in Fig. 3.10(a) was passed through the subsequent stages as discussed in previous sections and their corresponding results are shown in Fig. 3.10. The outputs of the proposed filtering technique for some intermediate frames of the fingerspelling sequence "A-E-O" are shown in Fig. 3.11.

Another set of results for another fingerspelling sequence "A-S-E" for "frame # 140" articulated by another native signer is shown in Fig. 3.12. Also, outputs for some intermediate frames of the fingerspelling sequence "A-S-E" are shown in Fig. 3.13.

Again, the segmentation of palm and/or finger(s) for the fingerspelling sequence "D-F-O" articulated by another third native signer can be viewed in Fig. 3.14.

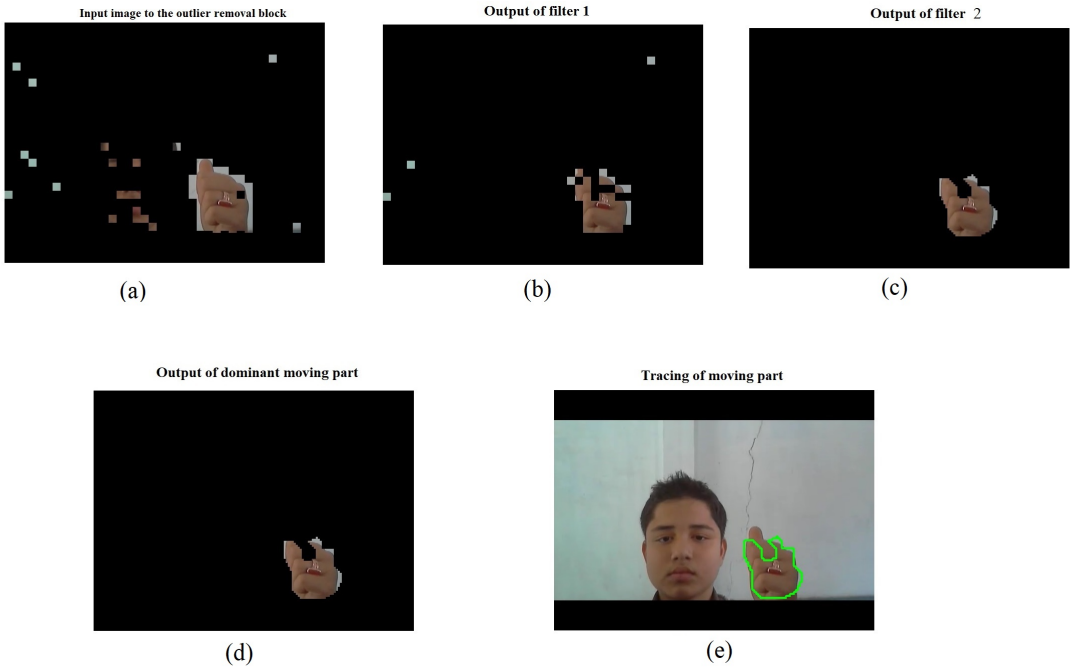


FIGURE 3.10: Outputs of outlier removal filters for "frame # 25" for fingerspelling sequence "A-E-O" (a) Input image to Filter 1 (b) Output image of Filter 1 (c) Output image of Filter 2 (d) Detection of dominant moving part (e) Tracing of dominant moving part.

3.3.2 Global motion estimation and global motion compensation framework in compressed domain for sign language videos

Global motions (GMs) that are caused by camera motion are often modeled by parametric transformations of two-dimensional (2-D) images [76]. The process of estimating the transformed parameters is termed as GM estimation (GME). Few works have been reported on Sign Language Recognition (SLR) using moving cameras. In SLR, motion information of hand(s) and finger(s) are considered as important cues [1]. So, for removing the moving components arising due to camera motion, the estimation of motion caused by moving camera followed by motion compensation is required. Here, we report a novel framework for GME and global motion compensation (GMC) for H.264/AVC compressed sign language videos.

There are many algorithms that have been proposed for GME and GMC. They can be broadly classified into two categories: pixel domain and compressed domain. Pixel domain approaches are more accurate but computationally inefficient [73], [58]. In [76], Su and Sun proposed a non-iterative MV-based GME algorithm for estimating

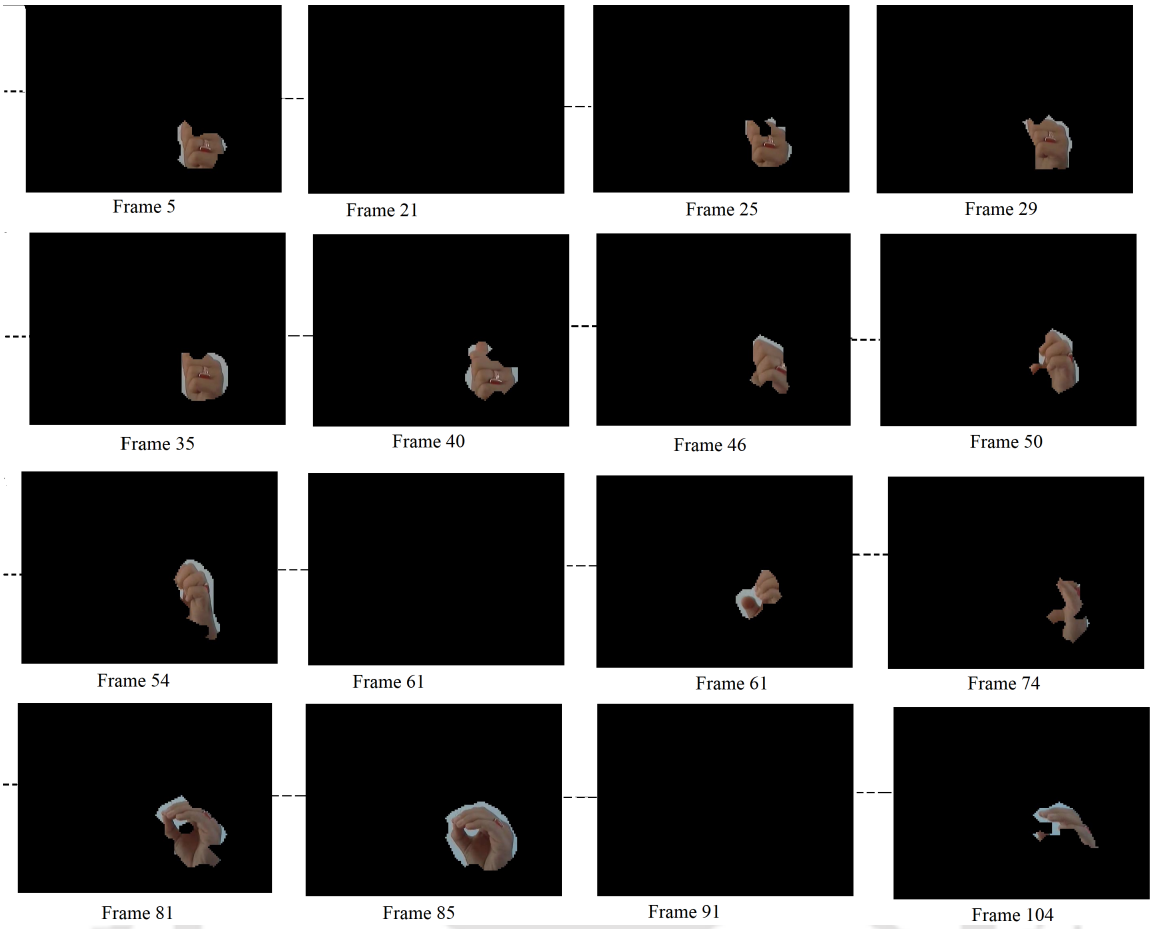


FIGURE 3.11: Segmentation of moving regions of palm and/or fingers for the fingerspelling sequence "A-E-O".

the perspective transform model GM parameters from the MVs obtained from the block matching process. In [77], they extended the previous work by a method that minimized the fitting error between the input MVs and MVs generated from the estimated motion model using the Newton-Raphson method with outlier rejections. In this technique, the compressed video had to be fully decoded prior to GME. On the other hand, in compressed videos, there are some inherent information available in the form of bitstreams along with the video such as DCT coefficients, macroblock (MB) types, motion vectors (MVs) etc. which can be partially decoded for further use [20]. Although GME using MVs of MBs are computationally efficient, their performance suffers due to the presence of outliers [73]. So, GME and GMC in presence of outliers is an important aspect of this work.

In the compressed domain, Smolic et al. [78] presented an algorithm for low complexity GME that worked with block-coded video (e.g. MPEG 2). A superimposed

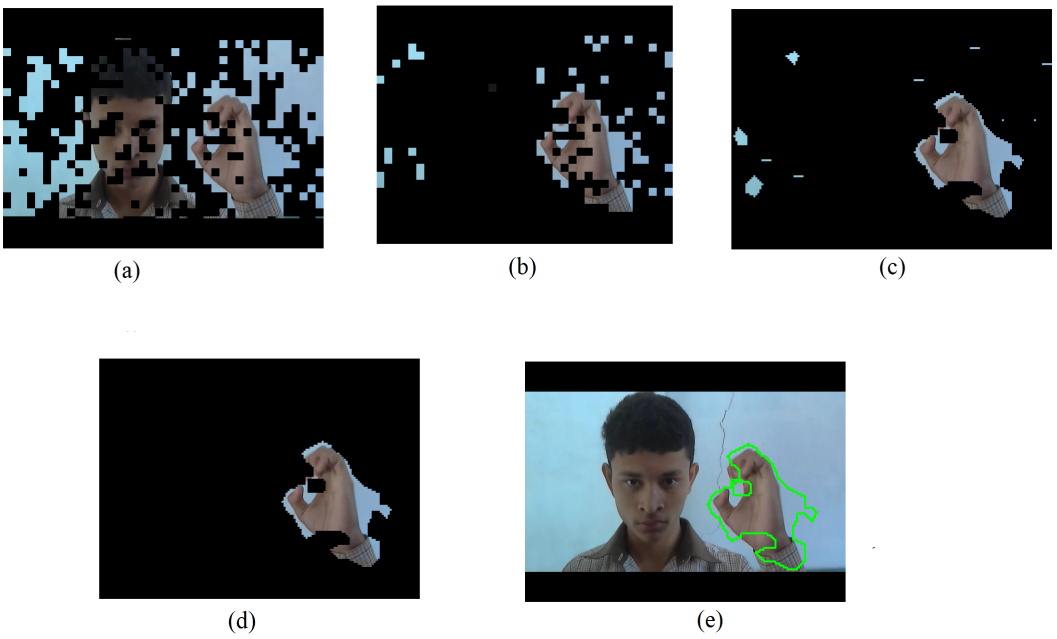


FIGURE 3.12: Outputs of outlier removal filters for "frame # 140" for fingerspelling sequence "A-S-E" (a) Input image to Filter 1 (b) Output image of Filter 1 (c) Output image of Filter 2 (d) Detection of dominant moving part (e) Tracing of dominant moving part.

GM model is fitted to decoded P-frame MV field, providing real-time performance for MPEG-7 application. In [73], Chen and Bajic used a method for simultaneous object segmentation and GME from a coarsely sampled MV field. Khatoonabadi and Bajic [79] presented a method for tracking moving objects in H.264/AVC compressed videos using a spatio-temporal Markov random field model. They preprocessed the MVs first, through intracoded block motion approximation and GMC. Okade et al. [62] proposed a novel learning-based camera motion characterization scheme along with its application to the video stabilization problem. The proposed scheme represented the compressed domain block MVs using polar angle and magnitude histograms and discriminative features from these two histograms were extracted and fed to a supervised learning-based hierarchical classifier for recognizing the six-camera motion patterns i.e. pan, tilt, zoom, and/or a combination of these basic components.

Here, a histogram-based technique to select MBs for estimation of perspective GM parameters for efficient GME and GMC in presence of outliers for sign language videos

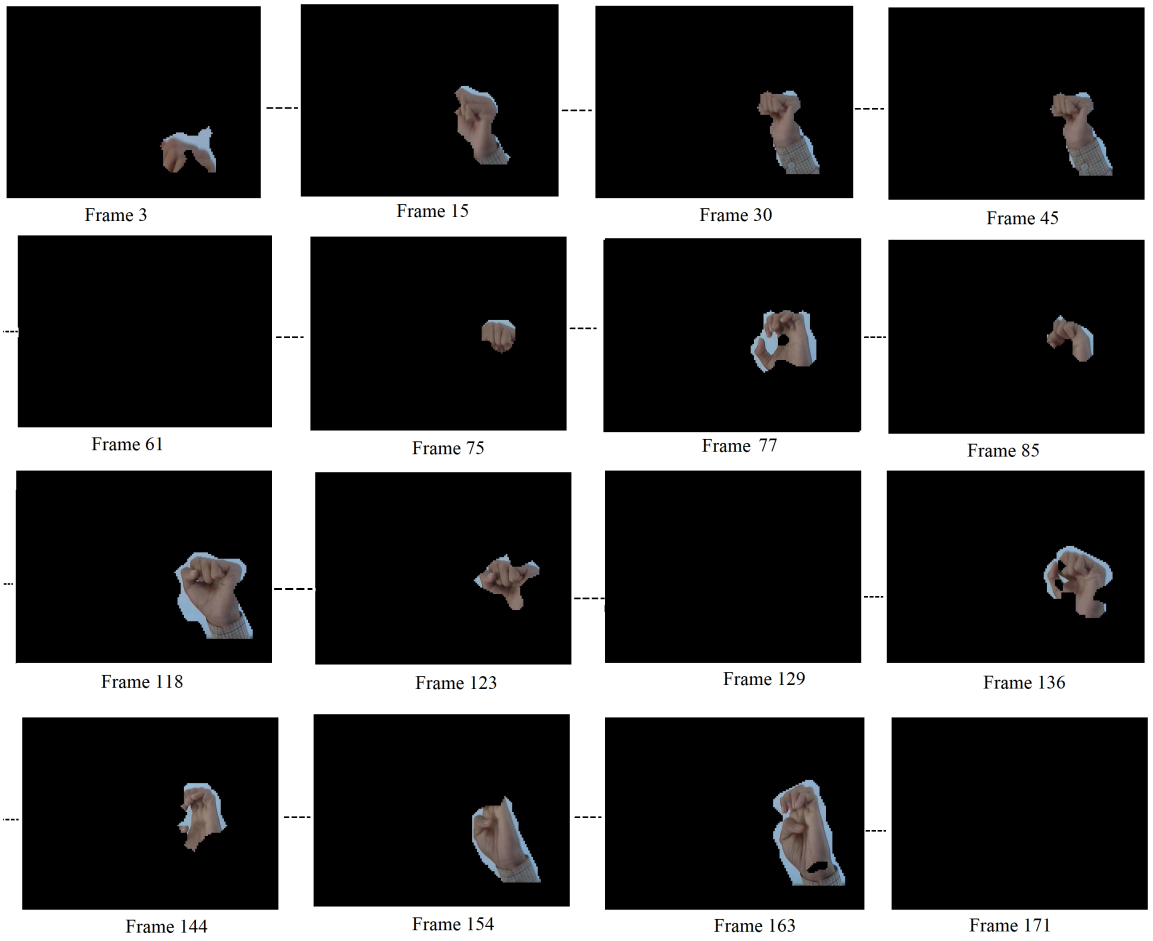


FIGURE 3.13: Segmentation of moving regions of palm and/or fingers for the fingerspelling sequence "A-S-E".

is proposed[2]. The framework for the proposed algorithm is shown in Fig. 3.15.

GME and GMC

Various 2-D parametric models such as translational, geometric, affine, and perspective have been defined in [77] out of which the eight-parameter perspective model is the most general one. The perspective model [73] is parametrized by an 8-D vector $m = [m_0, m_1, ...m_7]$. Given (x, y) and (x', y') as the coordinates in the current and reference frame respectively (as shown in Fig. 3.16), the perspective transformation is defined as follows:

$$x' = \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1}, \; y' = \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1}. \tag{3.6}$$

[2]The related work is published in IEEE International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2020 (Refer *List of Publications* page for details).

FIGURE 3.14: Segmentation of moving regions of palm and/or fingers for the video sequence "D-F-O".

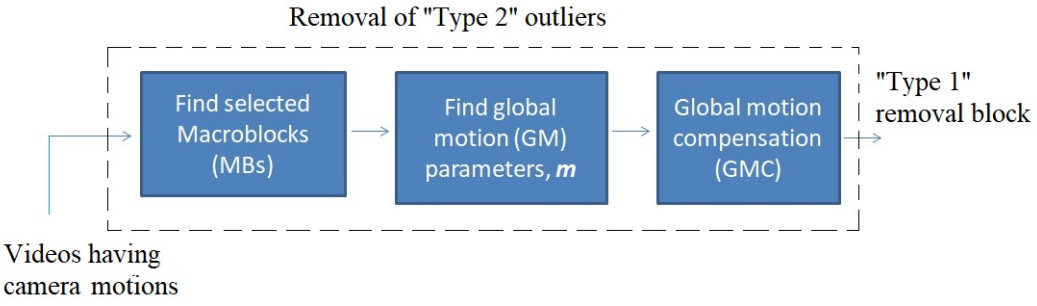


FIGURE 3.15: Proposed framework for GME and GMC using selected MBs.

The $\overrightarrow{\mathrm{MV_x}}$ and $\overrightarrow{\mathrm{MV_y}}$ for the block centered at pixel (x, y) in the current frame, corresponding to the motion model m, are given by

$$\mathrm{MV^X}(\mathrm{x}, \mathrm{y}; \mathrm{m}) = \mathrm{x}^{'} - \mathrm{x}, \ \mathrm{MV^Y}(\mathrm{x}, \mathrm{y}; \mathrm{m}) = \mathrm{y}^{'} - \mathrm{y}. \tag{3.7}$$

The goal of MV-based GME is to infer parameter vector m from the MVs in the case of compressed video. Let, $\mathrm{m_t}$ be the vector of estimated GM parameters in frame t and $\mathrm{MV}(\mathrm{x}, \mathrm{y}, \mathrm{t})$ be the MV of the block centered at pixel (x, y) in the frame t. Then the GM

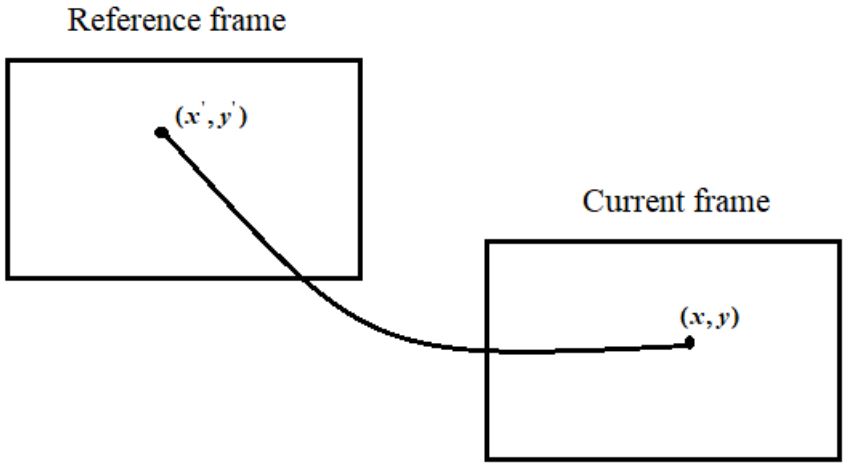


FIGURE 3.16: GMC with perspective motion model.

can be compensated from $\mathrm{MV}(x, y, t)$ by

$$\mathrm{MV}^{\mathrm{res}}(x, y, t) = \mathrm{MV}(x, y, t) - \mathrm{MV}(x, y; m_t) \tag{3.8}$$

where, $\mathrm{MV}^{\mathrm{res}}(x, y, t)$ is the compensated MV whose block is centered at pixel (x, y) in the frame t, and $\mathrm{MV}(x, y; m_t)$ is obtained as in Eq. 3.6 and Eq. 3.7. The goal of GME is to find m_t that minimizes the difference between $\mathrm{MV}(x, y; m_t)$ and $\mathrm{MV}(x, y, t)$. Non-linear least square error criterion can be used for it which is formulated as follows:

$$m_t = \arg\min_{m} \sum \|\mathrm{MV}(x, y, t) - \mathrm{MV}(x, y; m_t)\|^2 \tag{3.9}$$

Su and Sung [76] proposed a non-iterative method to solve Eq. 3.9. They used algebraic distance in the target function and finally derived the following matrix equation:

$$A^T A m = A^T b \tag{3.10}$$

where, **A** is a matrix containing motion information, **b** is a 2N-D vector containing transformed coordinates from Eq. 3.6, and **N** is the total number of MVs. The matrix equation can be solved by using pseudo-inverse via singular value decomposition [80] which is given by the following equation

$$m = \left(A^T A\right)^{-1} A^T b \tag{3.11}$$

The GM model discussed above fits for the MVs of background pixels which are far from the camera and therefore can be approximated by a planar surface [73], [57]. But the MVs due to "Type 2" outliers (moving hand(s) and finger(s) in case of sign videos) do not fit the model very well. So they need to be removed to improve GM estimation.

Many approaches have been proposed in the literature to remove these outliers [73], [57]. In [73], they performed iterations between GME and moving region segmentation. Firstly, they found GM parameters from the available MVs and then segmented moving regions from the compensated MVs. MVs of the segmented moving regions are treated as outliers in the context of GME for the next level of iteration.

We propose a method for finding the MVs of the background MBs (i.e. inliers) by avoiding the outlier MVs for each frame. The technique is based on the histogram of magnitudes of MVs as shown in Fig. 3.3. Fig. 3.17 shows the MV field for a frame and its corresponding histogram. As for a frame under moving camera, MBs of the background will have almost the same MV, and if we consider that the total size of the "Type 2" outliers is small as compared to frame size which is generally observed in sign language videos (less than around 35% of frame size), then the bin corresponding to the peak of the histogram will include most of the background MVs. So, by selecting the MBs correspond to that bin, we can separate the background MBs from the outlier MBs. Thus, we can use the MVs of the background MBs for our motion model for the calculation of m.

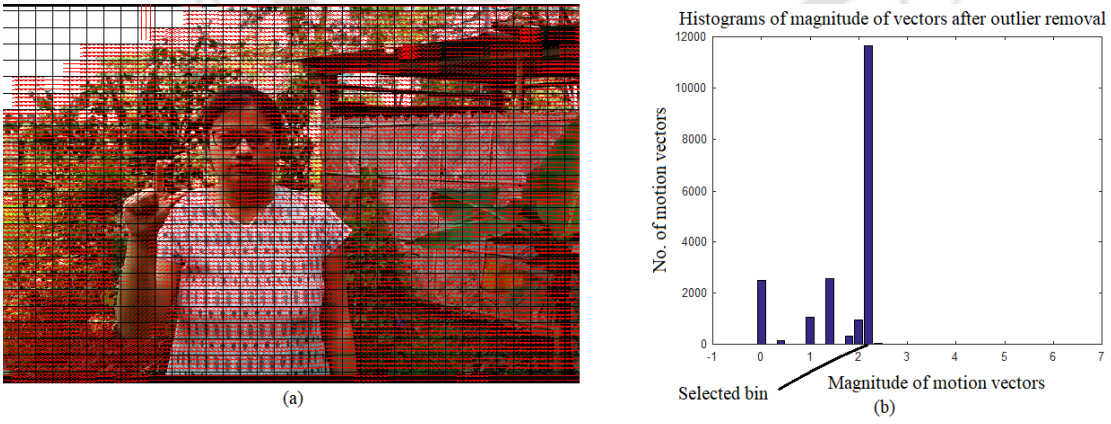


FIGURE 3.17: Frame with global motion showing (a) MV field (b) Magnitude histogram.

Initialization

An initial estimate of m for the first frame is done by calculating the average of sampled MVs [77] as

$$\begin{aligned} m_0 = m_3 = 1,\ m_1 = m_4 = m_6 = m_7 = 0 \\ m_2 = \frac{1}{N}\sum_{i=1}^{N} MVx_i,\ m_5 = \frac{1}{N}\sum_{i=1}^{N} MVy_i \end{aligned} \tag{3.12}$$

where, N is the total number of MVs involved.

We consider 15 different videos comprising of signs from American Sign Language recorded under a moving camera. Fig. 3.18 shows the output of various stages of the algorithm for the "Frame # 23" of the sign video "Always". Fig. 3.18(a) shows the original MVF of the frame which is extracted from the compressed video using the decoder. Fig. 3.18(b) shows the histogram of the magnitudes of the MVs. From the figure, it is seen that the peak of the histogram is not present in the "0" bin. So, according to our algorithm, global motion is present in the video. So the frame has to be processed through the GME and GMC block. Fig. 3.18(c) shows the GMC MVs if the global motion parameter vector m is estimated using all the MBs in the frame. The estimated GM vector field is shown in Fig. 3.19(a). When these GMC MVs are passed through the "Type 1" removal filter, the resulting GMC MVs are shown in Fig. 3.18(d). For removal of "Type 1" outliers, we use a median filter of window size 16×16 as this size gives optimum result. Fig. 3.18(e) shows the resulted GMC MVs when m is estimated using our proposed algorithm which is discussed in Section 3.3.2. According to the algorithm, as the peak value of the magnitude histogram is found in bin "3", the MBs corresponding to MVs falling in that bin are selected for fitting into the GM model of Eq. 3.10. The estimated GM vector field is shown in Fig. 3.19(b). When these GMC MVs are passed through the "Type 1" outlier removal filter, the resulting GMC MVs are shown in Fig. 3.18(f). It shows that our proposed method gives better results than without selected MBs.

A similar set of intermediate results for "Frame # 9" of the sign video "Hide" is shown in Fig. 3.20. Here also, it is found that our proposed method gives better results.

Fig. 3.21 shows the output of the GM detection block when there is no GM. As

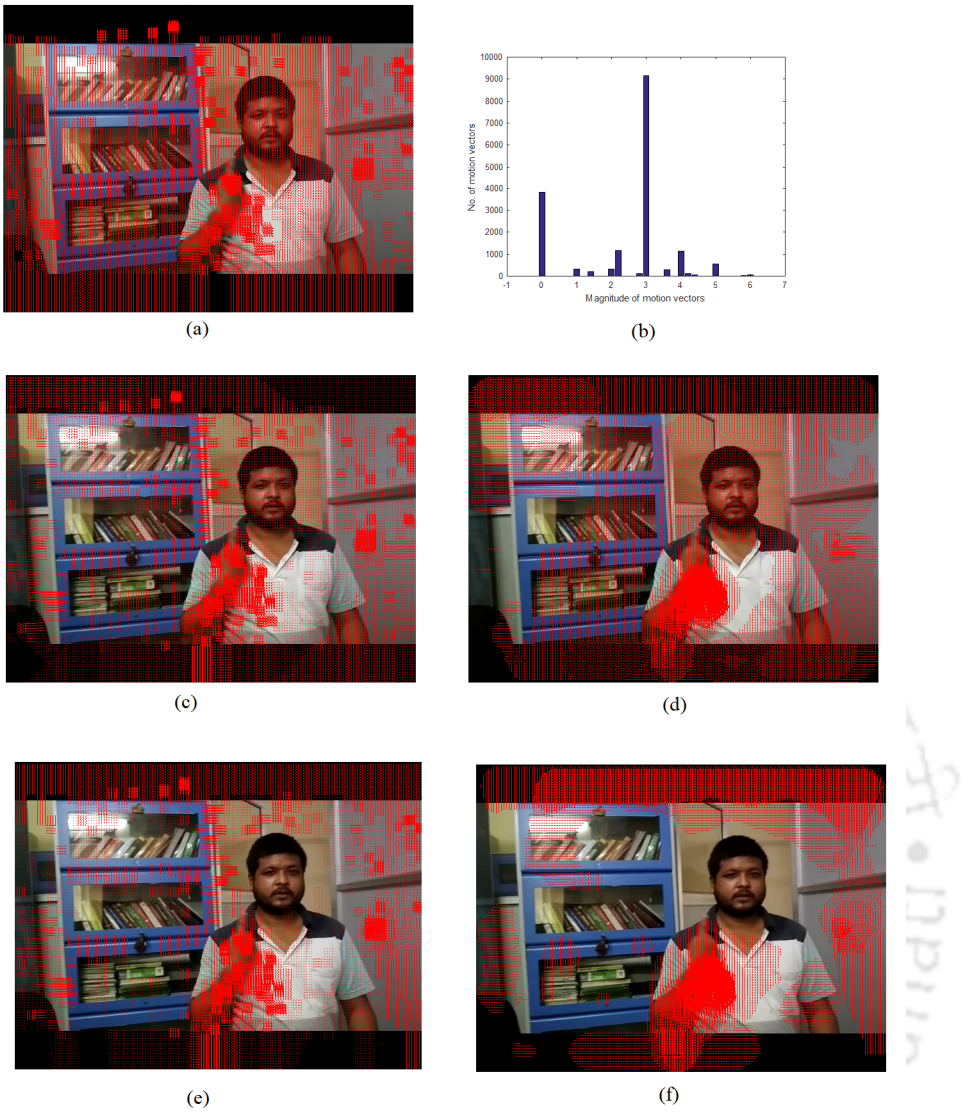


FIGURE 3.18: The output of the proposed GME and GMC framework for "Frame # 23" of the sign video "Always" (a) Original MVF (b) Histogram of magnitudes of MVs (c) GMC MVs without selected MBs (d) GMC MVs after removal of "Type 1" outliers without selected MBs (e) GMC MVs with selected MBs (f) GMC MVs after removal of "Type 1" outliers with selected MBs.

there is no GM, the peak of the histogram is found in the "0" bin.

Fig. 3.22 shows the output of the proposed framework when "Type 2" outlier is absent i.e. when only camera motion is present without any moving region in the video itself. In this case, it is seen that both the methods i.e. selected MBs and without selected MBs to give almost the same results.

Since standard datasets for sign videos under a moving camera is not available, we also tested our proposed framework for some standard video databases. Fig. 3.23

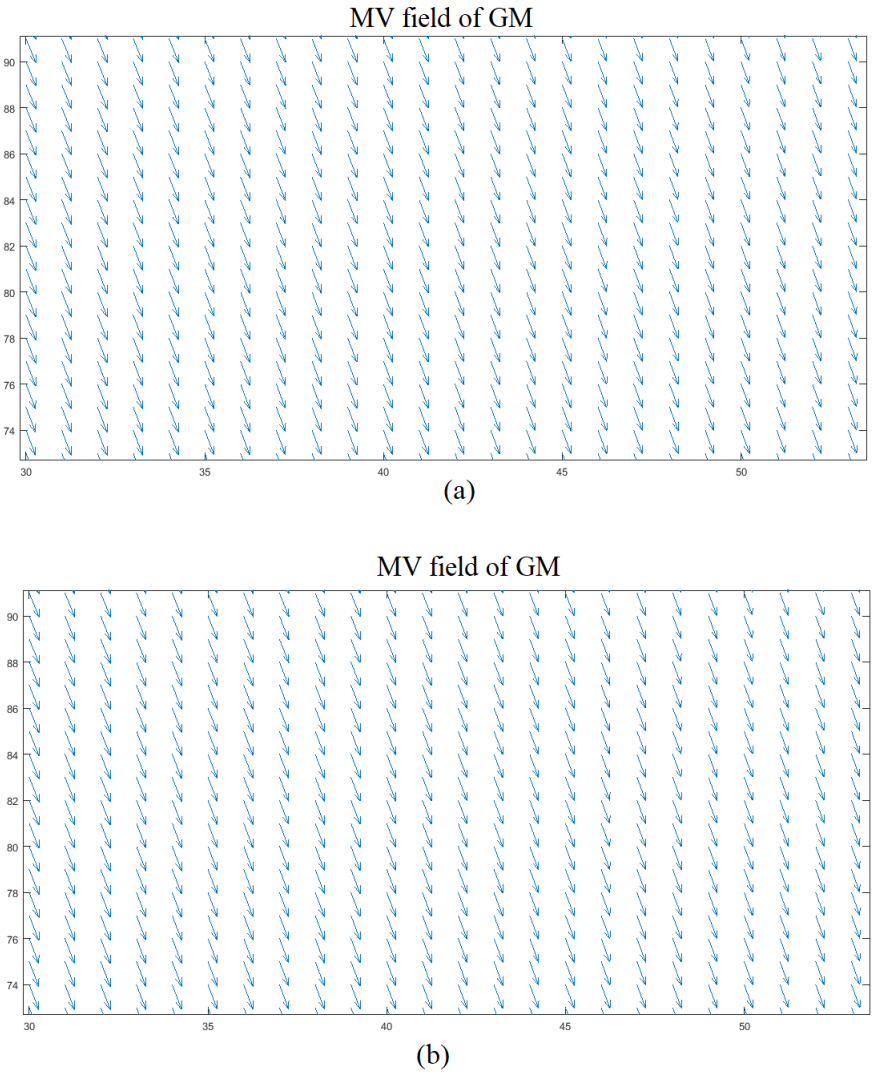


FIGURE 3.19: GM vector field (a) Without selected MBs (b) With selected MBs.

shows the output of various stages for "frame # 56" for the video "Stefan" (available at: http://media.xiph.org/video/derf). Here also we saw that our proposed method performed well.

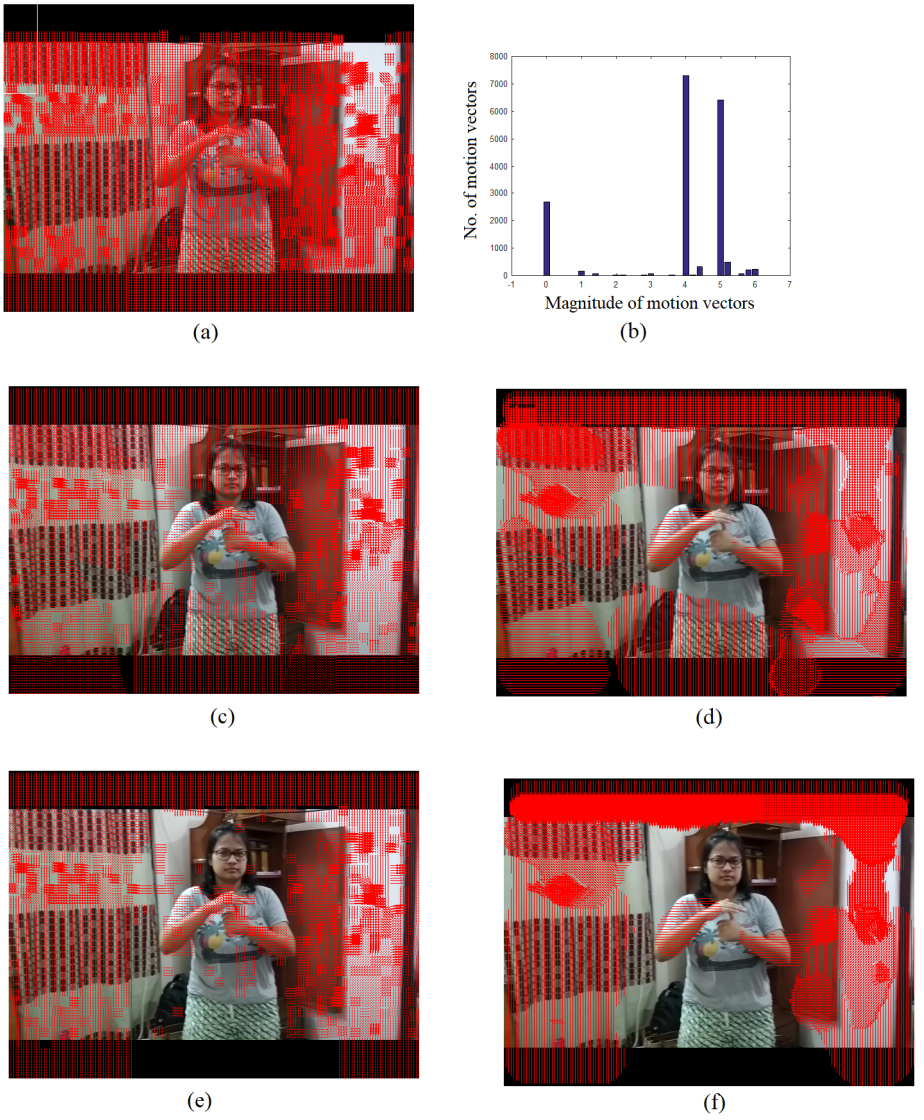


FIGURE 3.20: The output of the proposed GME and GMC framework for "Frame # 9" of the sign video "Hide" (a) Original MVF (b) Histogram of magnitudes of MVs (c) GMC MVs without selected MBs (d) GMC MVs after removal of "Type 1" outliers without selected MBs (e) GMC MVs with selected MBs (f) GMC MVs after removal of "Type 1" outliers with selected MBs.

3.4 Proposed spatio-temporal Markov random field (ST-MRF) based framework for spotting continuous fingerspelling

The proposed filtering technique for removal of "Type 1" outliers can remove outliers when the movements of the fingers are significant. However, the proposed filtering technique fails when the fingers move very slowly (approach towards a standstill position)

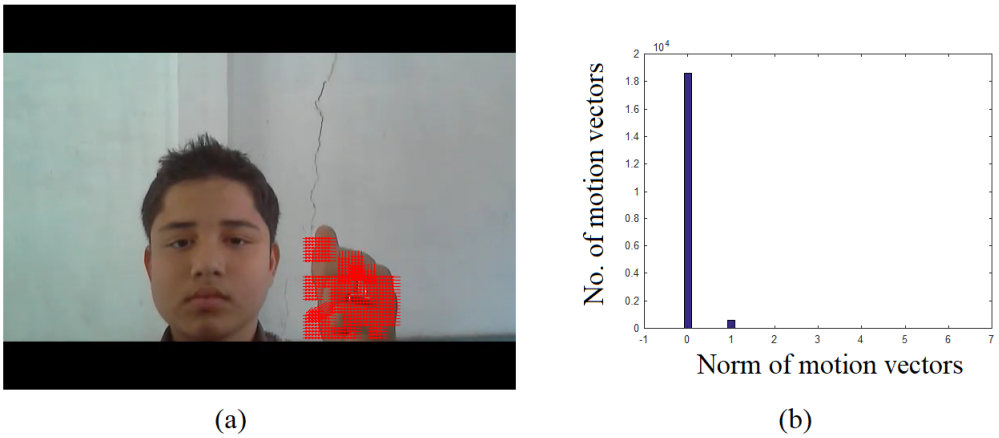


FIGURE 3.21: The output of GM detection block when there is no camera motion for "Frame # 19" for the fingerpelling video "A". (a) Original MVF (b) Histogram of magnitudes of MVs.

and/or any other body parts move significantly as shown in Fig. 3.24. Our proposed filtering scheme only uses MVs from the current frame without considering the temporal information. So, we propose a Markov random field (MRF) based model for removing the outliers[3].

The proposed framework for continuous fingerspelling spotting for H.264/AVC compressed videos is shown in Fig. 3.25. At first, the H.264/AVC video is partially decoded to extract the bitstreams containing the MVs for each frame. Then, this MV field (MVF) is inputted to a spatio-temporal Markov random field (ST-MRF) based model for removal of "Type 1" outliers. After the removal of these outliers, spotting of fingerspellings is done. The proposed sign and *me* separation algorithm is based on observation of various data sets where hand (and/or fingers) changes its representation (shape) faster than other gesturing body parts (including the background and the face) during the *me* phase [19]. The ST-MRF is employed to exploit the contextual dependencies of the MVs of fingers [81] in a frame and the temporal dependencies of MVs between two consecutive frames. For this ST-MRF model, the position of the centroid of the palm is required. The centroid of palm for each of the first 10 frames is found out (as shown in Fig. 3.26) using the technique discussed in section 3.3 and the average value is taken. The proposed ST-MRF-based model for removal of all the outliers present in a frame and labeling of each of the frames is described below:

[3]The related work is published in Multimedia Tools and Applications, Springer, 2021 (Refer *List of Publications* page for details)

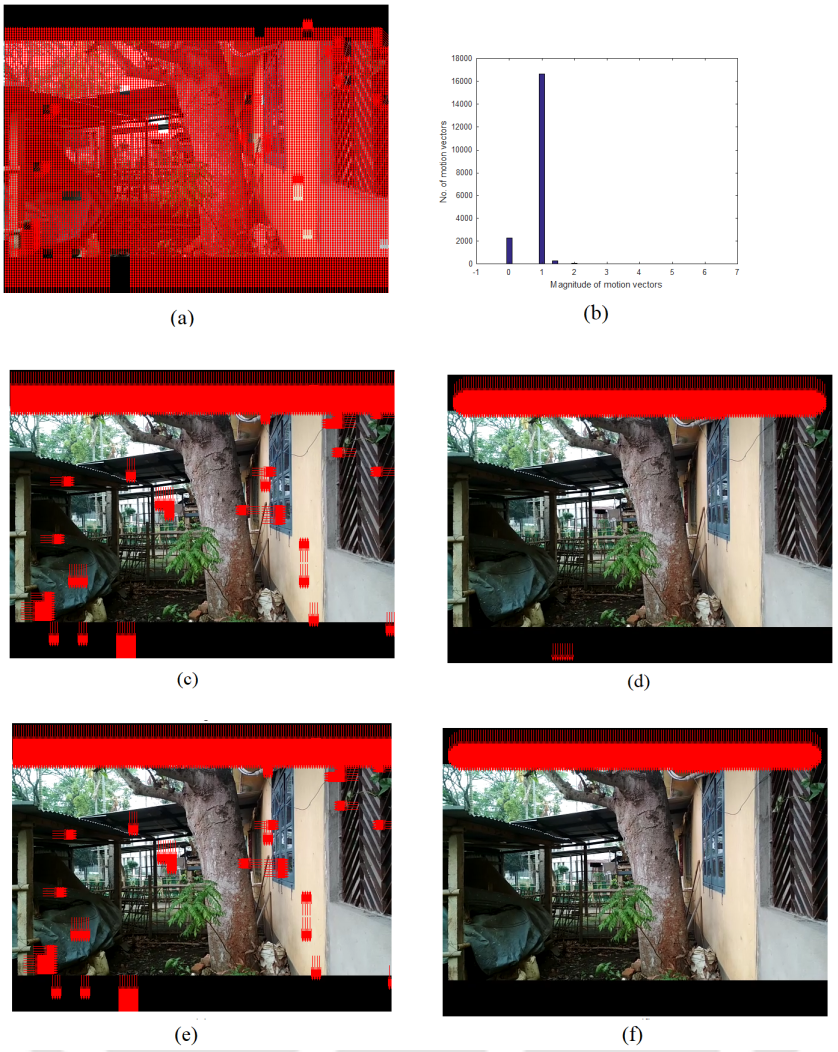


FIGURE 3.22: The output of the proposed GME and GMC framework for "Frame # 9" of the video without "Type 2" outlier (a) Original MVF (b) Histogram of magnitudes of MVs (c) GMC MVs without selected MBs (d) GMC MVs after removal of "Type 1" outliers without selected MBs (e) GMC MVs with selected MBs (f) GMC MVs after removal of "Type 1" outliers with selected MBs.

Let, the set of sites be $S = \{1, 2, ..., m\}$ where, $m = \{x, y\}$ is the position of the sub-MB with nonzero MV. Consider the set $L = \{object, outlier\}$ where, object is the MV corresponding to the moving finger(s) and palm. In this case, object and outlier MVs are represented as label '1' and label '0', respectively so that $L = \{1, 0\}$.

The removal of outliers can be done by assigning a label $f_i \in L$ to the site $i \in S$. When each site is assigned with a unique label $f_i = f(i)$, it can be expressed as a mapping from S to L, i.e., $f : S \rightarrow L$. We want to infer f_i^t for the frame t given the label f_i^{t-1} for the previous frame $(t-1)$ and the observed motion information v_i^t extracted from the compressed video. The criterion for choosing "the best" labeling f_i^t is that it should

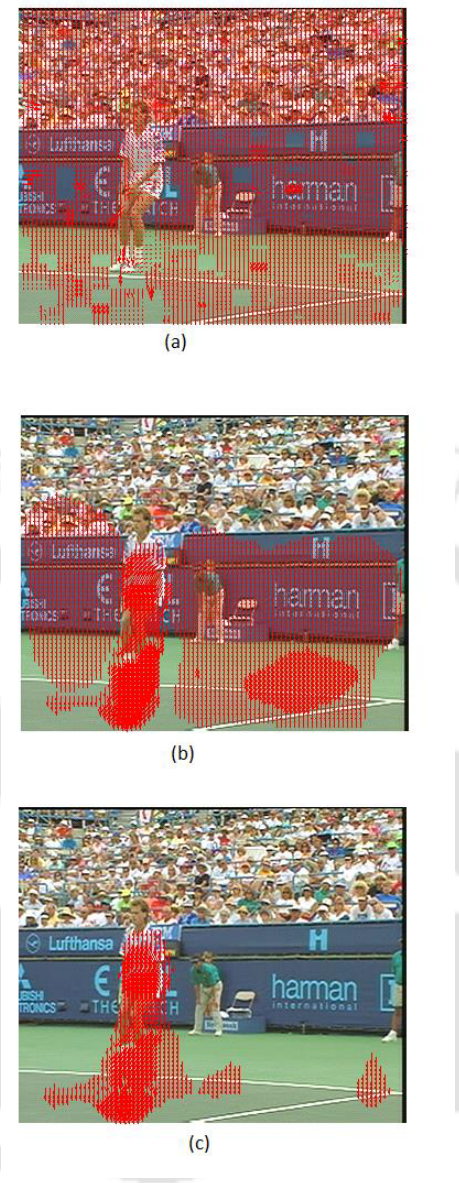


FIGURE 3.23: The output of the proposed method for "frame # 56" of the video "Stefan" (a) Original MVF (b) Output of GME-GMC considering all MBs (c) Output of the proposed method.

maximize the posterior probability $P(f_i^t|f_i^{t-1}, v_i^t)$. Considering four spatial neighbors in the image plane and two temporal neighbors in time, this problem can be expressed as a maximum a posteriori (MAP) MRF labeling maximizing $P(f_i^t|f_i^{t-1}, v_i^t)$. The posterior probability in terms of interframe likelihood $P({f_i}^{t-1}|{f_i}^t, v_i^t)$, the intra-frame likelihood $P(v_i^t|f_i^t)$ and the *a priori* probability $P(f_i^t)$ is given by [79]:

$$P(f_i^t|f_i^{t-1}, v_i^t) = \frac{P(f_i^{t-1}|f_i^t, v_i^t).P(v_i^t|f_i^t).P(f_i^t)}{P(f_i^{t-1}, v_i^t)} \tag{3.13}$$

FIGURE 3.24: Wrong detection of MVs.

As the denominator part i.e., $P(f_i^{t-1}, v_i^t)$ is independent of f_i^t, the MAP solution for f_i^t is found by maximizing the numerator of Eq. 3.13.

$$f_i^{t*} = \arg\max_{f_i^t \in L} \left\{ P(f_i^{t-1}|f_i^t, v_i^t).P(v_i^t|f_i^t).P(f_i^t) \right\} \tag{3.14}$$

According to the Hammersley-Clifford theorem [82], the probability distribution $P(f)$ can be expressed as Gibbs distribution of the form

$$P(f) = \frac{1}{Z} e^{-\frac{1}{T}E(f)} \tag{3.15}$$

where, $Z = \sum_{f \in F} e^{-\frac{1}{T}E(f)}$ is a normalizing constant called the partition function. T is a constant called *the temperature* and $E(f)$ is called the *energy function*. The energy,

$$E(f) = \sum_{c \in C} V_c(f) \tag{3.16}$$

is a sum of clique potentials $V_c(f)$ over all possible cliques.

Using Eq. 3.15, the probabilities in Eq. 3.14 can be expressed as

$$P(f_i^{t-1}|f_i^t, v_i^t) = \frac{1}{Z_1} e^{-\frac{1}{T_1}E_1\left(f_i^{t-1}|f_i^t, v_i^t\right)} \tag{3.17}$$

$$P(v_i^t|f_i^t) = \frac{1}{Z_2} e^{-\frac{1}{T_2}E_2(v_i^t|f_i^t)} \tag{3.18}$$

$$P(f_i^t) = \frac{1}{Z_3} e^{-\frac{1}{T_3}E_3(f_i^t)} \tag{3.19}$$

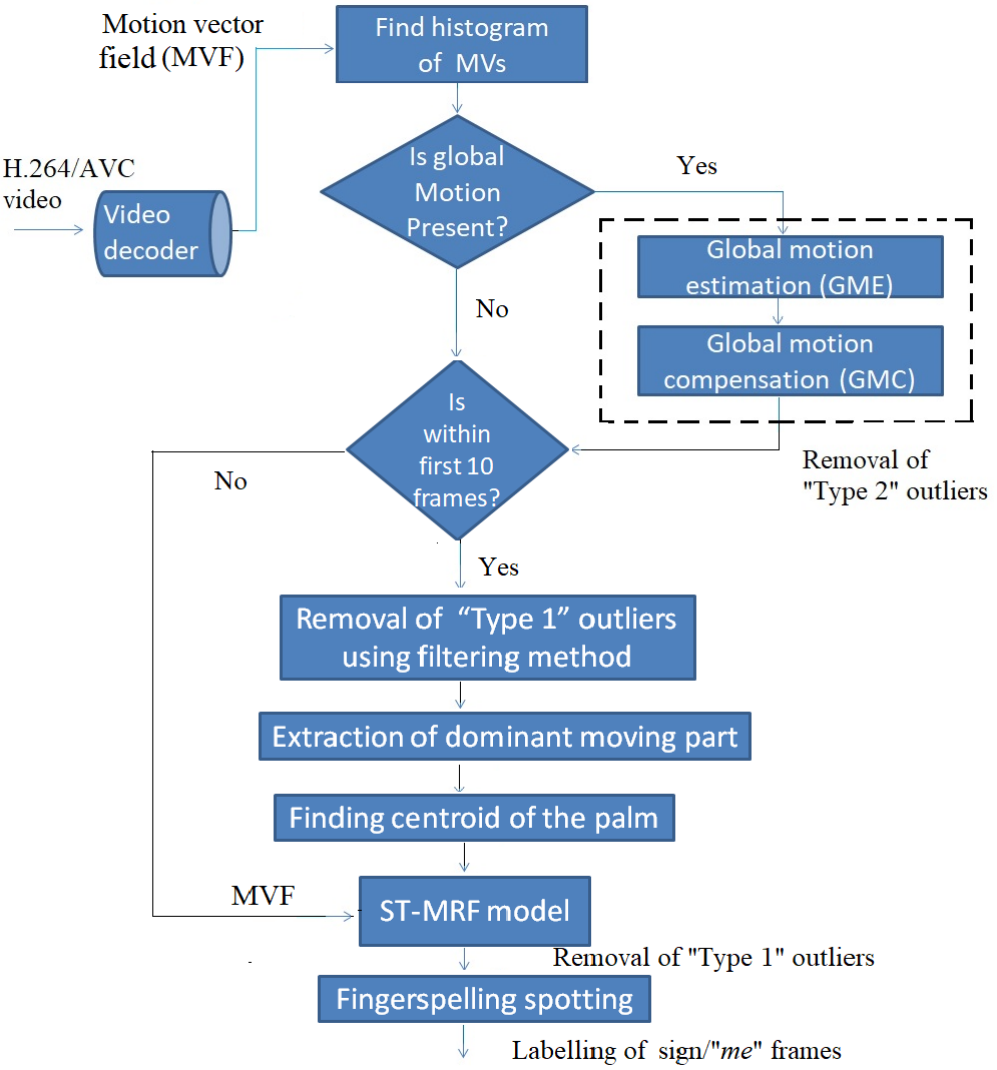


FIGURE 3.25: Proposed ST-MRF based framework for continuous fingerspelling spotting.

So, Eq. 3.14 can be written as

$$f_i^{t*} = \arg\max_{f_i^t \in L} \frac{1}{Z_1 Z_2 Z_3} e^{-\left\{\frac{1}{T_1}E_1\left(f_i^{t-1}|f_i^t, v_i^t\right)+\frac{1}{T_2}E_2\left(v_i^t|f_i^t\right)+\frac{1}{T_3}E_3\left(f_i^t\right)\right\}} \tag{3.20}$$

So, the MAP estimator is equivalent to

$$f_i^{t*} = \arg\min_{f_i^t \in L} \left\{\frac{1}{T_1}E_1\left(f_i^{t-1}|f_i^t, v_i^t\right) + \frac{1}{T_2}E_2\left(v_i^t|f_i^t\right) + \frac{1}{T_3}E_3\left(f_i^t\right)\right\} \tag{3.21}$$

The first term measures the temporal discontinuity of labeling between consecutive frames, the second term represents the spatial incoherences among object MVs, and

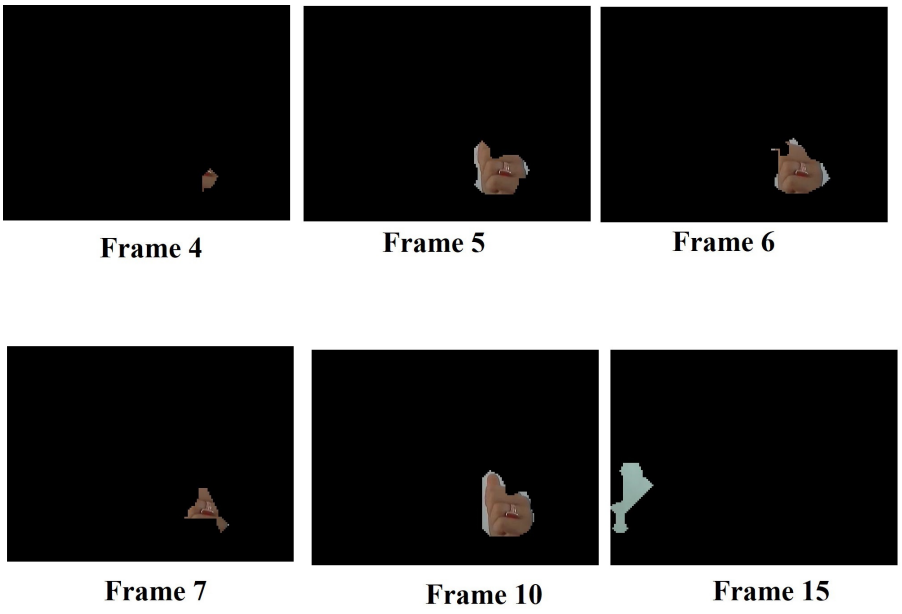


FIGURE 3.26: Segmented palms of the signer with centroids.

the third term measures the compactness of the object's shape [79]. These three terms are discussed in detail below:

A. Temporal continuity:

In the case of continuous fingerspelling, as the movements of the fingers are not abrupt, there is a temporal dependency of the movement of the finger(s)/palm between two consecutive frames. It acts as an important cue for tracking the fingers/palm. Temporal continuity is measured by the overlap between the labeling of the previous frame and labeling of the backward projected current frame to the previous frame.

Let us consider a block i in the current frame that is assigned to the inlier, i.e., $f_i^t = 1$. The block is projected back to the previous frame along with its MV, v_i^t as shown in Fig. 3.27. Here, black and white blocks represent inlier and outlier blocks respectively. The degree of overlap, $D(i + v_i^t; f_i^{t-1})$ is computed as the difference between the labels of the previous frame and current projected frame at position i. Lesser the value of $D(i + v_i^t; f_i^{t-1})$ for a macroblock, more is the continuity.

B. Context coherence:

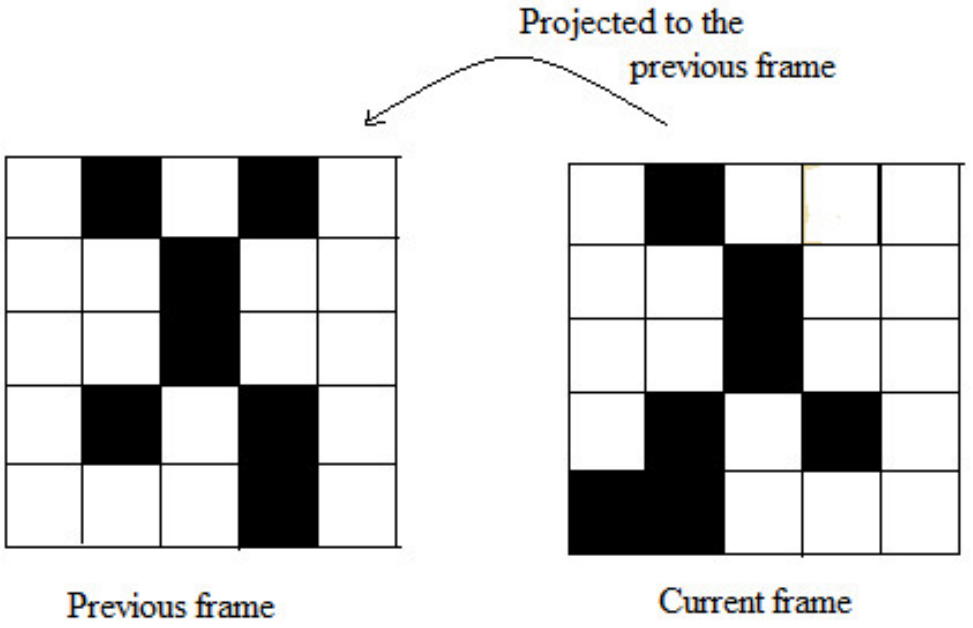


FIGURE 3.27: Finding temporal continuity between two consecutive frames.

Although the movement of the fingers are non-rigid in case of ASL fingerspelling, the MV of an inlier's MB is consistent with its neighboring MB's MV. This is termed as the coherence of the MVs belonging to fingers.

For finding coherence of an MB with $f_i^t = 1$ and MV v(i), a pair-site clique in second-order neighborhood system [81] for the interior site v(i) is considered as shown in Fig. 3.28. The coherence energy is measured as the average of the differences of v(i) to its 8-neighbor MVs v_k where, $k = 1, 2, ..., 8$, which is given by:

$$E_2\left(v_i^t, f_i^t\right) = \frac{1}{8}\sum_{k=1}^{8} |v(i) - v_k| \tag{3.22}$$

Lesser the value of $E_2\left(v_i^t, f_i^t\right)$, more is the coherence.

C. Compactness:

In the case of ASL fingerspelling, the sub-MBs belonging to finger(s)/palm are closely packed. For finding compactness, the Euclidian distance d(i) between the centroid of the palm in the current frame (x_c, y_c) and the position $i = (x, y)$ of the macroblock having $f_i^t = 1$ (as shown in Fig. 3.29) is calculated as:

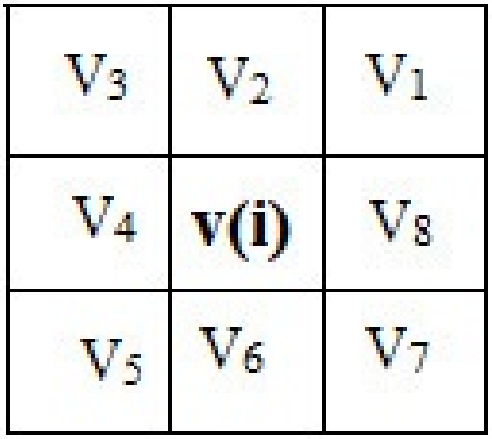


FIGURE 3.28: Finding the coherence from an 8-neighborhood system.

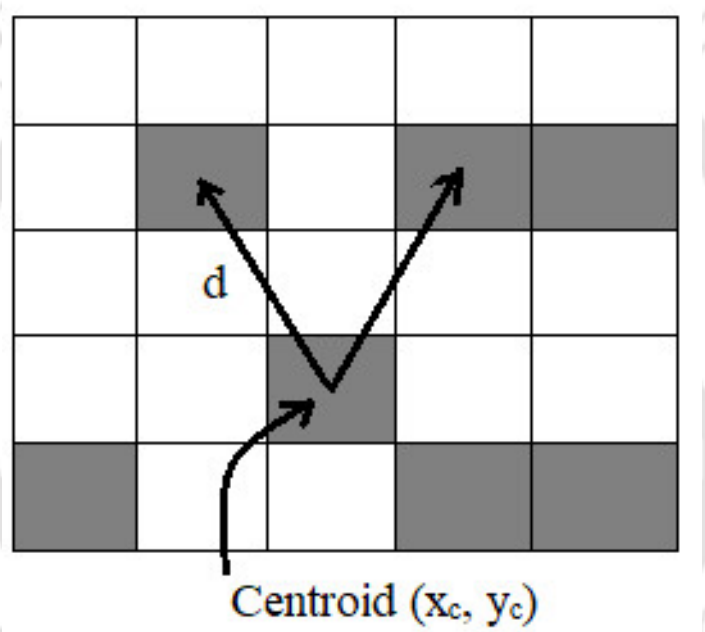


FIGURE 3.29: Finding the compactness of an object's MBs.

$$d(i) = \sqrt{(x - x_c)^2 + (y - y_c)^2} \tag{3.23}$$

This distance is normalized between 0 and 1 by using the relation

$$E_3\left(f_i^t\right) = d'(i) = \frac{d(i) - d_{min}(i)}{d_{max}(i) - d_{min}(i)} \tag{3.24}$$

where,

$$d_{max}(i) = \mu + \sigma$$
$$d_{min}(i) = \mu - \sigma$$

μ and σ are the mean and standard deviation of the distribution of d(i) for all MBs. More the value of $d'(i)$, less is the context coherence. The centroid for the current frame (x_c, y_c) is updated as

$$\begin{aligned} x_c &= x_{c-1} + mv_{x_c} \\ y_c &= y_{c-1} + mv_{y_c} \end{aligned} \tag{3.25}$$

where, (x_{c-1}, y_{c-1}) is the position of the centroid in the previous frame and (mv_{x_c}, mv_{y_c}) is the MV of the centroid in the current frame. The values of different constants used in the model are chosen based on the observations and experiments done.

3.4.1 ST-MRF optimization

All the terms of Eq. 3.21 are defined in sections A, B, and C. In this work, Iterated Conditional Modes [83] is used to solve Eq. 3.21 for its simplicity. For each block i with label 1, the total energy is calculated. The block is converted to '0' block if the total energy is greater than a predefined threshold value. This iterative process continues for 5 iterations which give the optimum result. Here, the threshold is selected as $\max(5, \mu + \sigma/2)$. These μ and σ are calculated from the histogram of the total energy. When no outlier is present, the energy of each block will be less than $\max(5, \mu + \sigma/2)$.

3.4.2 Spotting of frames

After removing the outliers and detecting the dominant moving part, we are now in a state to separate the sign and *me* frames from the continuous fingersplleing sequence. Here, the frame having the motion of palm and/or fingers can be considered as *me* frame, while the frame where the palm and fingers become almost static can be considered as a sign frame as shown in Fig. 3.30. So, from the output of the earlier step, we can easily determine whether a frame is a sign or *me* by checking the mask obtained after processing as discussed in the previous section. If the frame contains any MB of the

palm and/or fingers, then it is considered as a *me* frame, otherwise considered as a sign frame.

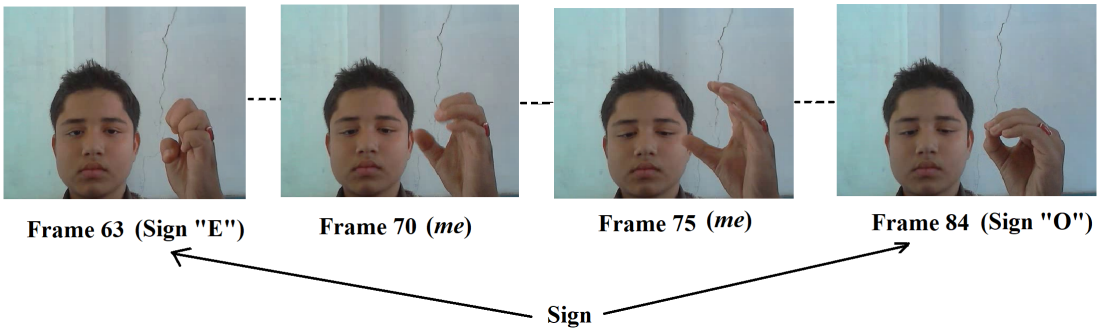


FIGURE 3.30: Fingerspelling spotting.

3.5 Experimental results

For experimental results, we collected video data from a local deaf and dumb school. The videos were captured by a camera with a VGA (640×480) resolution with a frame rate of 30 frames per second under normal illumination conditions with a simple background. The fingerspelled videos were articulated by three native and two non-native signers. Each signer articulated three ASL fingerspelling sequences, each containing three different alphabets from the vocabulary of 24 unique ASL alphabets (excluding 'J' and 'Z'). Different fingerspelling sequences are : A-S-E, A-E-O, G-O-A, C-O-E, M-A-Y, B-F-O, K-L-D, M-N-T, N-O-S, P-Q-G, R-U-V, U-V-W, K-L-D, M-N-T, N-O-S. So our dataset comprises 45 fingerspelling alphabets performed by both native and non-native signers. The videos were then encoded to H.264/AVC 4:2:0 sequences using the H.264/AVC JM V.18.0 encoder [75] with a baseline profile and with different bitrates, with GOP structure IPPP.., i.e., the first frame is coded as Intra (I), and the subsequent frames are coded predictively (P) with GOP size 30. A block size of 4×4 is used, as this is the lowest unit possible and is more likely to represent the moving objects. In order to have a uniformly sampled motion vector field, all the MVs of variant block sizes are re-mapped into 4×4 blocks to maintain consistency. The proposed algorithm is implemented in

MATLAB 9.1 (R2016b) on an Intel Core i3-330 M CPU running at 2.13 GHz with 4 GB RAM.

3.5.1 ST-MRF based moving fingers segmentation

The results obtained after applying the MAP ST-MRF model for segmentation of the moving fingers are shown in Fig. 3.31. It shows the outputs for some intermediate frames for the fingerspelling sequence "A-E-O" performed by Signer 1. The first column shows the input MV field (MVF), the second column shows the MVF after removing the outliers and the third column shows the resultant segmented moving regions. Similar

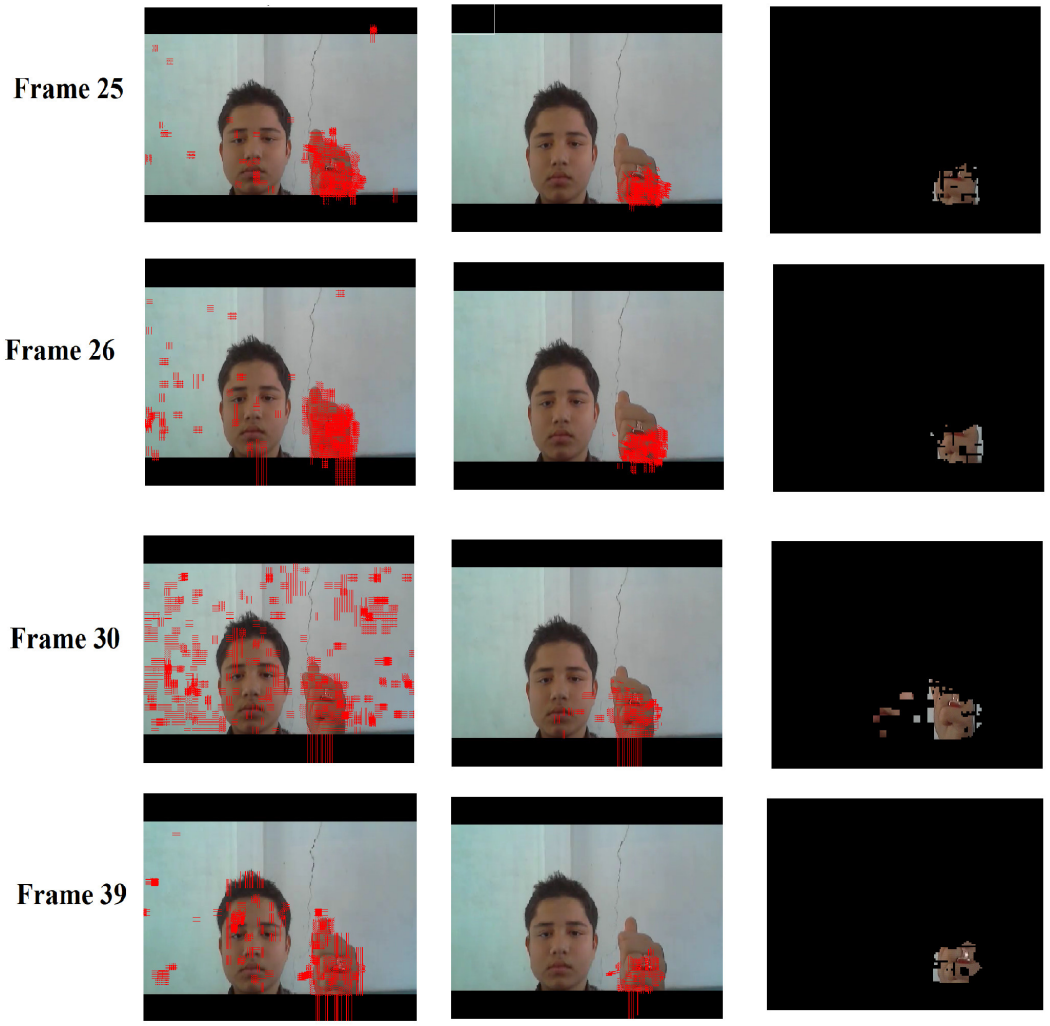


FIGURE 3.31: Moving region segmentation using ST-MRF for the fingerspelling sequence "A-E-O".

results are shown for another sequence "A-S-E" performed by Signer 2 in Fig. 3.32. From

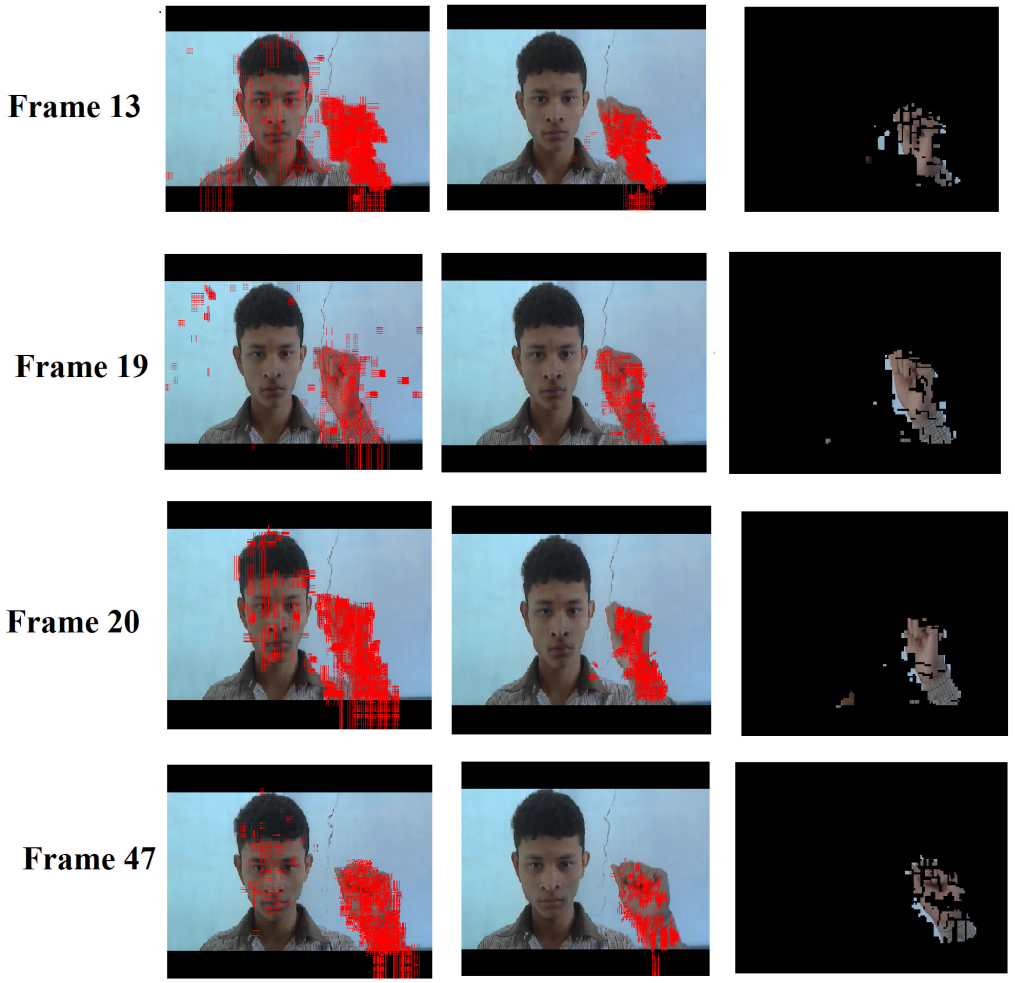


FIGURE 3.32: Moving region segmentation using ST-MRF for the fingerspelling sequence "A-S-E".

the results, it is seen that the proposed method is able to remove almost all the outliers present in the frame effectively.

3.5.2 Spotting of fingerspelling sequence

Spotting of continuous fingerspelling sequence i.e., separation of sign and *me* frames is based on the observation as discussed in section 3.4.2. Fig. 3.33 shows the results of fingerspelling spotting in the case of sequence "A-E-O" articulated by Signer 1. Here, the sign frames (i.e., frame # 31, # 61, and # 91 for the signs "A", "E", and "O" respectively) are seen to be completely dark as there is no movement of any one of the fingers and palm, while the other frames are *me* frames. Similarly, Fig. 3.34 shows the results of

fingerspelling spotting for another sequence "B-F-O" articulated by Signer 3. Here, the sign frames are frame # 20, # 61, and # 80 for the signs "B", "F", and "O" respectively.

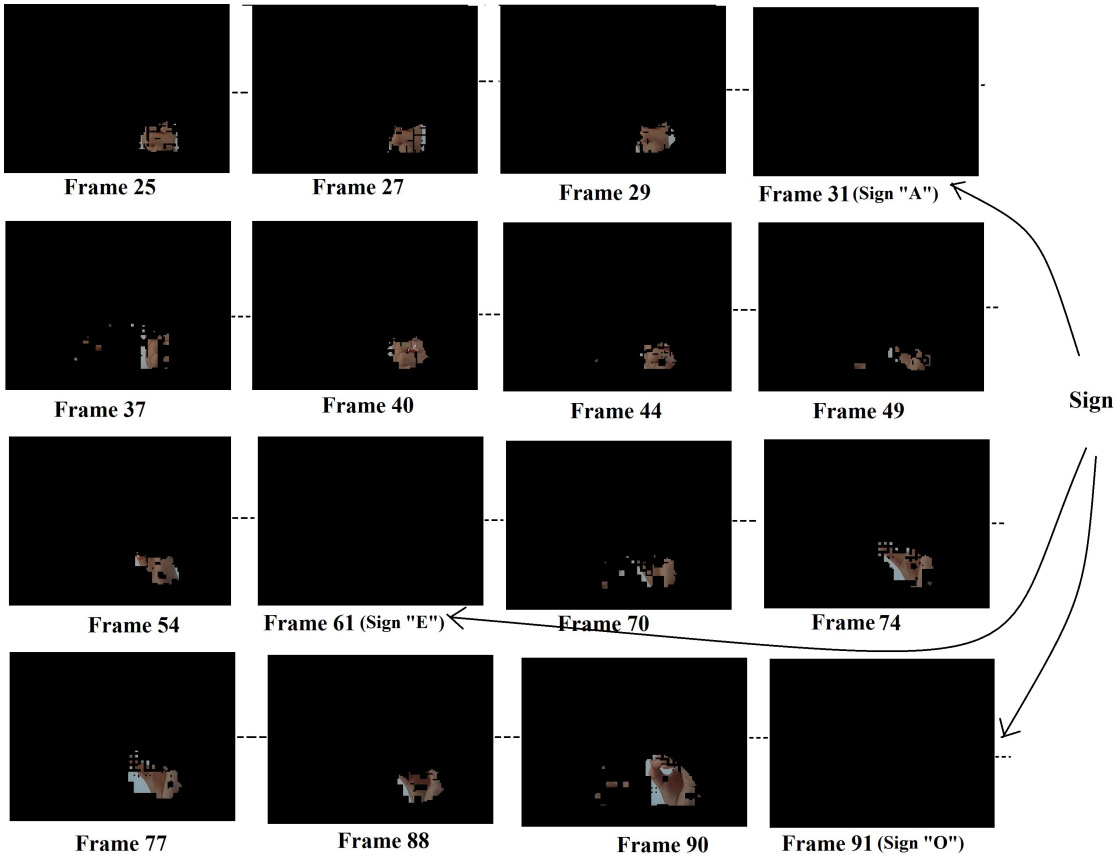


FIGURE 3.33: Continuous fingerspelling spotting for the sign sequence "A-E-O".

If we express the probability of a frame i to be a *me* frame as

$$p_{me}(i) = \frac{\text{Total number of MBs with nonzero MVs}}{\text{Total number of MBs in the frame}} \tag{3.26}$$

Then the probability of getting a sign frame is given by

$$p_{sign}(i) = 1 - p_{me}(i) \tag{3.27}$$

In our experiment, the size of each MB is 4×4 and the frame size is 640×480. So, the total number of MBs present in a frame is $= (640/4) \times (480/4) = 19,200$. Fig. 3.35 (a) shows the probability $p_{me}(i)$ for the fingerspelling sequence "A-E-O" for various

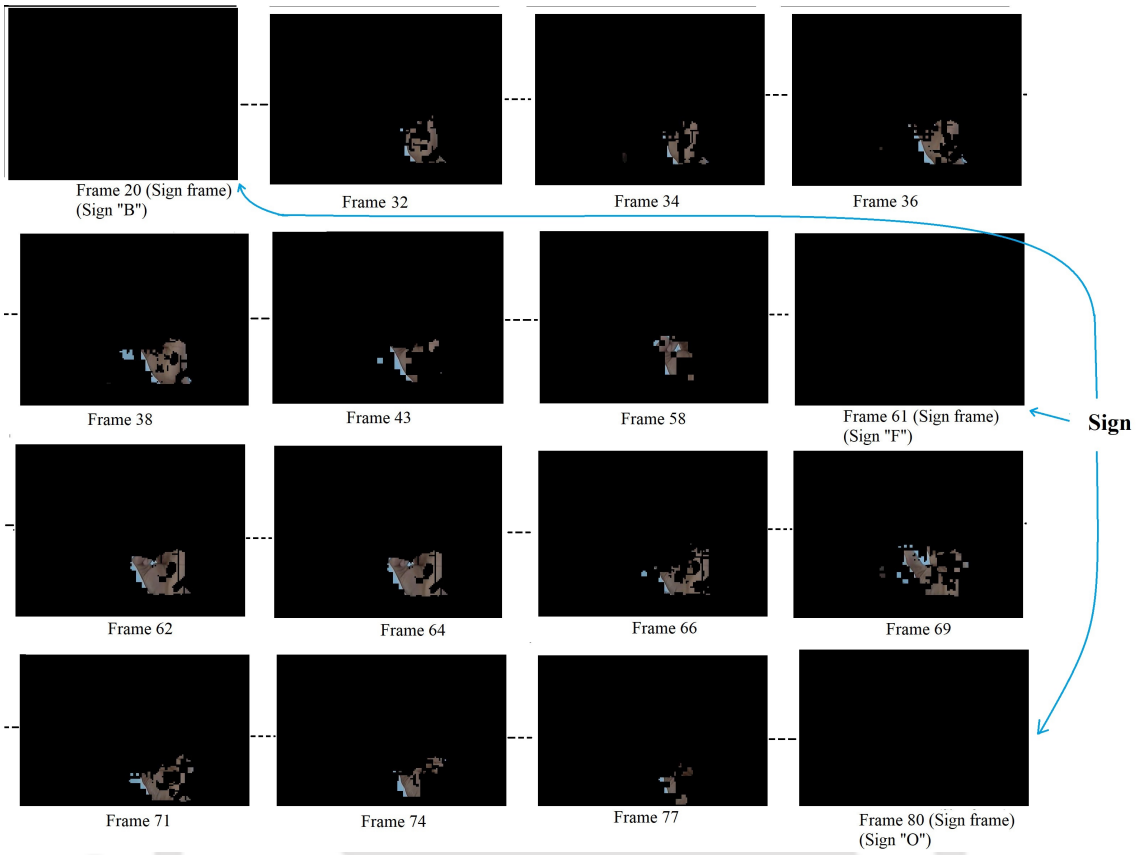


FIGURE 3.34: Continuous fingerspelling spotting for the sign sequence "B-F-O".

frames, *i*. The regions of sign frames are highlighted as circles. A similar result for fingerspelling sequence "B-F-O" is shown in Fig. 3.35 (b).

Quantitative evaluation of the proposed method is carried out by correct spotting rate (R) which is calculated as [84]:

$$\mathrm{R} = \frac{\mathrm{C}}{\mathrm{N}} \times 100 \quad (\%) \tag{3.28}$$

where, C is the total number of correct spottings and N is the total number of test signs. The correct spotting c_s, is defined as

$$c_s = \begin{cases} \text{True, if } (\Delta S + \Delta E) < 10 \\ \text{False, otherwise} \end{cases} \tag{3.29}$$

where, $\Delta S = |g_s - s_s|$ and $\Delta E = |g_e - s_e|$. s_s and s_e are the start and end frame numbers of the spotted sign respectively, and g_s and g_e are the start and end frame numbers of

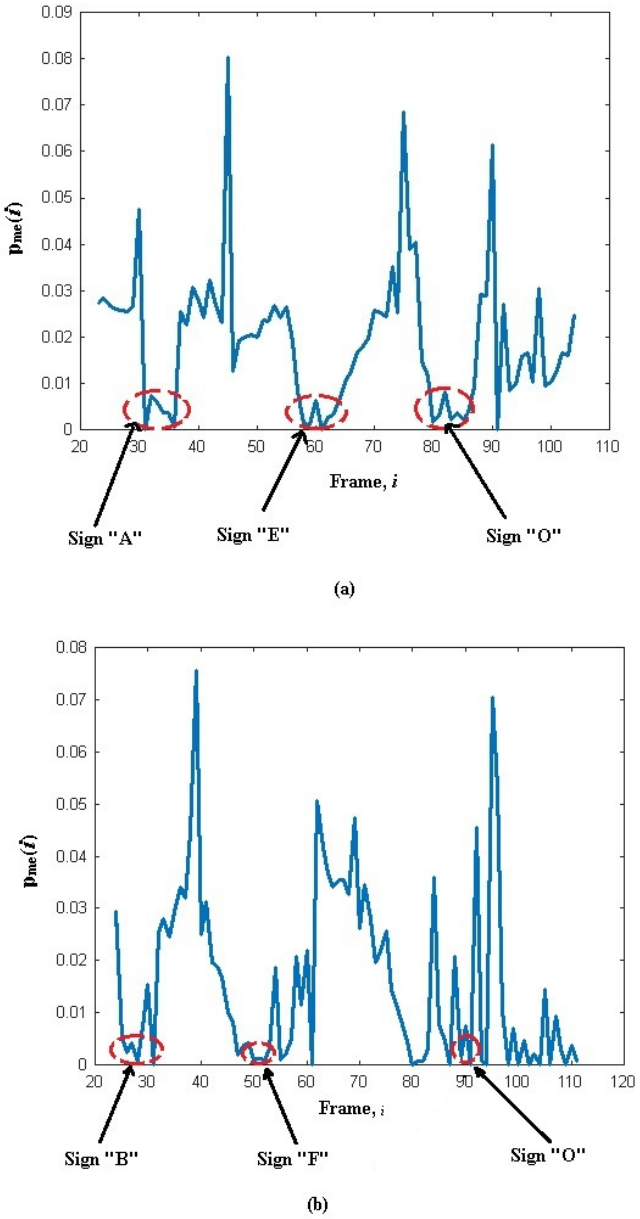


FIGURE 3.35: Probabilities of getting *me* frames for fingerspelling sequences (a) A-E-O (b) B-F-O.

the ground truth of the spotted sign, respectively. Table 3.1 shows the fingerspelling spotting results for three fingerspelling sequence videos. From the table, it is seen that for fingerspelling sequence "A-E-O", all the three letters are spotted correctly according to Eq. 3.29. In the case of fingerspelling sequence "B-F-O", letters "B" and "O" are correctly spotted, but letter "F" is not spotted correctly. Similarly, in the case of "C-O-E", all the letters are spotted correctly.

After testing the proposed algorithm for various fingerspelling videos, the average

TABLE 3.1: Fingerspelling spotting results

Videos	Fingerspelling sequence	Number of fingerspelling segments	Ground-truth for fingerspelling	Spotted fingerspelling segments
1	A-E-O	3	A (39-42) E (63-64) O (86-100)	A (31-32) E (58,61) O (91-92)
2	B-F-O	3	B (30-38) F (65-72) O (93-102)	B (31-32) F (50-52) O (91, 94, 98, 100)
3	C-O-E	3	C (15-30) O (49-60) E (84-87)	C (17, 28, 31) O (47, 59, 61) E (83, 87)

spotting rate is found to be 75%. As there are limited works on fingerspelling spotting in the compressed domain, to compare our approach with other works, we considered [3], which was done on uncompressed sign videos. In [3], Yang and Lee proposed a novel machine vision method for sign language spotting. In order to spot signs and fingerspellings in a signed utterance, they used a hierarchical framework as shown in Fig. 3.36. First, a two-layer CRF discriminate between candidate segments of signs and fingerspellings. After detecting the candidate segments, their hand shapes were verified using BoostMap embeddings. Finally, the motions of fingerspellings were verified with the CRF.

The two-layer CRF was useful for representing information about the hand trajectory and hand location. However, there are ambiguities between signs that have similar hand movements and different hand shapes. In addition, all fingerspellings are represented with a single label. Therefore, the hand shapes of the detected candidate segments of signs and fingerspellings were recognized using BoostMap embeddings (BMEs). The main goal of the hand shape-based sign and fingerspelling verification method was two-fold: (1) Deciding whether or not a sign segment spotted via a motion- and location-based spotting method was accepted as a sign. (2) Identifying the alphabets in a fingerspelling

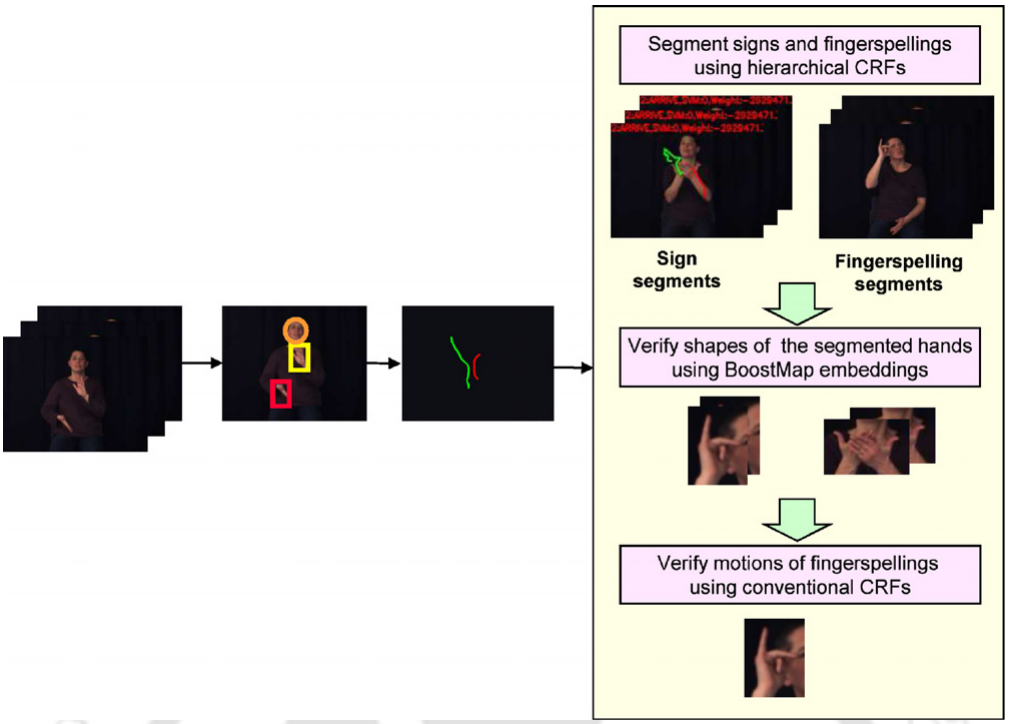


FIGURE 3.36: Overview of the method proposed in [3] for spotting sign and fingerspelling

segment. This method was robust to various scales, rotations and sizes of the signer's hands.

The letters 'I', 'J' and 'Z', '1' have similar hand shapes and this can lead to ambiguities in fingerspelling spotting. There are ambiguities between alphabets that exhibit similar hand shapes and different hand motions. The hand motions of alphabets were verified in order to distinguish between alphabets that had similar hand shapes and different hand trajectories.A conventional CRF was applied in order to verify the hand motions of alphabets.

The HMM and conventional CRF were implemented and compared in the ASL spotting application. A discrete HMM with five states was constructed for spotting signs and fingerspellings. A fixed threshold that maximized the correct detection rate was selected for the HMM and CRF. Also, the BoostMap embeddings and SVM were implemented and compared in order to verify the hand shapes. As these methods used a fixed threshold for detecting signs, fingerspellings and non-sign patterns, our proposed model gives better results than these models.

As shown in Table 3.2, our proposed model is superior to five other models except for the last model which is based on BMEs. In the case of the last model with BME, a hierarchical framework is used with three steps: (1) Candidate segments of signs and fingerspelling were done using a two-layer CRF, which consisted of a T-CRF to segment signs, fingerspelling and non-sign patterns followed by a conventional CRF to recognize sub-sign patterns. (2) BMEs were used to verify hand shapes of detected signs and fingerspelling. (3) The motions of fingerspelling were verified in order to distinguish fingerspelling with similar handshapes but different hand trajectories. Although their spotting rate is high, there are many other advantages of using the compressed domain method, such as extraction of inherent information like MVs, MBs sizes, etc. from compressed videos as discussed earlier.

TABLE 3.2: Comparison of various fingerspelling spotting methods

Models	**R(%)**
HMM^{BME}	26.0
CRF	30.0
CRF^{SVM}	40.0
Lee proposed $model^{SVM}$	70.0
Our proposed model	**75.0**
Lee proposed $model^{BME}$	78.0

3.6 Conclusion

In this chapter, we propose a real-time continuous fingerspelling spotting framework for H.264/AVC videos. The framework is based on the MVs extracted from the video itself. Compressed domain techniques suffer from the unwanted MVs present in the frames which are termed as outliers. There are two types of outliers present: "Type 1" in case of static camera and "Type 2" in case of moving camera. Firstly, we propose a filtering technique for the removal of "Type 1" outliers from a frame based on the MVs present in that frame. But this technique fails when the palm and/or fingers move very slowly (approach towards a standstill position) and/or any other body parts move significantly.

So to overcome this, we propose an ST-MRF-based model for removing "Type 1" outliers considering the temporal information also. The first 10 frames of the video are used for automatic detection of the centroid of the palm, which is subsequently utilized by the proposed ST-MRF model. We also consider the problem of any camera movement that may be present during the capturing of the sign videos. For that, we propose a framework for GM detection, GME, and GMC by removing "Type 2" outliers for H.264/AVC encoded sign language videos. After the removal of these outliers, fingerspelling spotting is done. We tested our framework for various combinations of ASL alphabets except for "J" and "Z" and achieved an accuracy of around 75%. The proposed framework is robust to any kind of outliers and movement of any body parts apart from palm/fingers. Careful natural articulation of the signs is important so that the signer does not make an unnecessary pause during the signing. The framework can further be combined with any existing pattern recognition algorithm for the recognition of the signs. The work, first of its kind, offers lots of scopes for further investigation.

Chapter 4

An Adaptive Thresholding Based Movement Epenthesis Detection Technique using Hybrid Feature Set for Continuous Fingerspelling Recognition

Sign language recognition (SLR) systems are gaining importance in recent times as these have established themselves as important elements of human-computer interaction (HCI). Also, these provide an opportunity for the deaf and hearing-impaired to communicate with the common people without the need for an interpreter. Yet there are plenty of challenges in this field which are worth exploring and solutions formulated. In this chapter, we address the design of a continuous fingerspelling recognition system in the uncompressed domain which segments a fingerspelling sequence into meaningful extracts and non-sign patterns and thereby recognizes the meaningful signs. Sign segmentation is carried out by means of a unique set of features which comprises of hand shape, the velocity of gesturing hand, and displacement of palm centroids between successive frames. Specialized

techniques like adaptive thresholding and Finite State Machine (FSM) models are also incorporated into our system for efficient classification of sign and movement epenthesis (me) frames. We validate the performance of our proposed system taking into account the continuous fingerspelling alphabets of American Sign Language (ASL) considering both native and non-native signers. Our proposed system also has the potential to tackle complex backgrounds involving multiple objects, backgrounds with multiple signers, and different brightness conditions.

4.1 Introduction

Sign languages are natural means of interaction that use different types of expressions for communication among persons with special abilities. Of late, these have become important elements of human-machine interaction (HMI) systems and are accepted as significant aids for persons with hearing impairments [85]. So, the requirement to develop sign language recognition (SLR) systems has received considerable attention as these facilitate the dissemination of information between healthy people and people with special abilities. Despite the popularity of such SLR systems, there are certain limitations that impose constraints on their unrestricted use. These are difficulties faced while performing automatic SLR in backgrounds with complex content, multiple signers, brightness variations, etc. In addition, coarticulation is accepted to be another crucial factor that puts constraints on the performance of continuous SLR systems [86]. So, the above-mentioned facts have motivated us to develop a robust SLR system that can tackle the above-mentioned difficulties and limitations.

From the literature survey discussed in Section 2.1, it is seen that although the accuracy of DTW-based methods is high, they require more computational time. Again, model-based approaches require a large set of training datasets which may not be available in some cases. Again simple motion-based features may fail to spot signs and *me* accurately. So, in this work, we propose a motion-based hybrid feature set accompanied with a statistical adaptive thresholding methodology for automatically spotting the signs accurately from continuous fingerspelling sequences. Our proposed system also provides

an efficient hand segmentation scheme which provides effective segmentation results irrespective of complex background, background with multiple signers, and also under different illumination conditions. The objective of our work is to develop a continuous fingerspelling recognition system for recognizing ASL alphabets embedded in a continuous fingerspelling stream by advocating a two-level approach. The key contributions of the work are:

1. We propose a composite set of features namely hand shape, the velocity of gesturing hand, and displacement of palm centroids between successive frames, for discriminating between sign and *me* frames.
2. Based on the above feature set, adaptive thresholds are formulated and a Finite State Machine (FSM) model is proposed for classification of sign and *me* frames.

Recognition of the extracted alphabets is done by means of a characteristic feature vector comprising of Fourier descriptors of the maximum curvature points of the valid sign segments. Our proposed system also demonstrates the ability to handle different background conditions like complex background, background with multiple signers, and also different lighting conditions like daylight and dim-light.

This chapter is organized as follows: the proposed framework is presented in detail in Section 4.2, Section 4.3 shows the experimental results and finally Section 4.4 concludes the work.

4.2 Proposed system

The overall block diagram of the continuous fingerspelling recognition system for recognizing the ASL alphabets interspersed within a fingerspelling sequence involving local motion is shown in Fig. 4.1. The system mainly consists of three modules:

1. **Hand segmentation module**: This module is incorporated to extract out the hand portion which is our region of interest. It comprises skin color detection followed by morphological operations and combined with contour processing, which enables the system to tackle complex background situations.

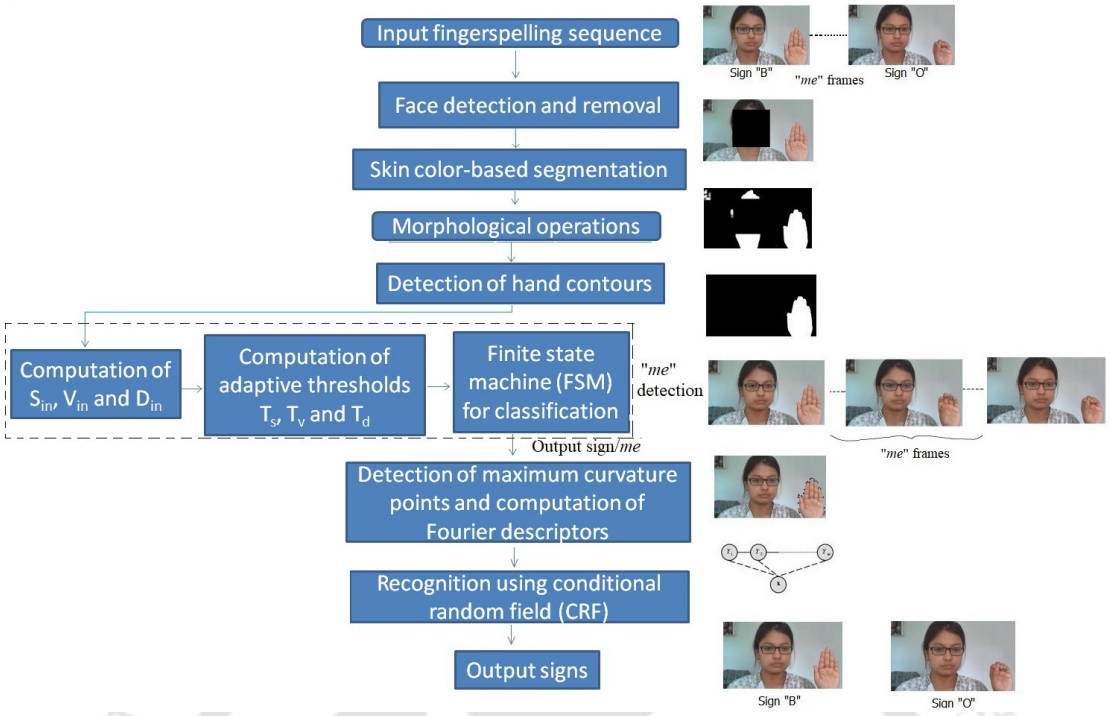


FIGURE 4.1: Block diagram of the proposed system.

2. **Movement epenthesis detection**: This block is responsible for the detection and removal of unwanted *me* frames from continuous fingerspelling sequences. It employs an adaptive thresholding technique and an FSM block for accurate classification of sign and *me* frames.

3. **Feature extraction and recognition module**: This module is responsible for extracting out some characteristic features from segmented signs and their subsequent recognition. Here we use Fourier descriptor as a feature which is given as an input to a CRF classifier for recognition.

It has several blocks starting with an input segment which provides the sign sequences to the face detection and removal block. The novelty of the work is the use of composite feature set, adaptive thresholding, and application of FSM for sign and *me* classification [1].

The detailed working of our proposed system is described on the next page:

[1]The related work is published in SN Computer Science, Springer, 2021 (Refer *List of Publications* page for details).

4.2.1 Hand segmentation

Hand segmentation is a very crucial task in automatic SLR as it is greatly affected by background conditions like complex background, dynamic background, background with multiple gestures, and varying illumination. The accuracy of segmentation greatly determines the overall recognition rate of the system. Two of the commonly used hand segmentation techniques are skin color segmentation and background subtraction [86]. Here we use a novel technique proposed by us in [87] which is robust to various background conditions. In this technique, a combination of skin color detection, frame differencing and contour matching is employed. The working principle is described below:

At first, the input frames are captured from the web-camera, and face detection and removal using Haar classifier [88] is done to mask out the face region. After this, skin color segmentation [89] is done to separate out the hand region. This is followed by a morphological opening and closing operation to filter out noise and to fill-up holes. However, skin color segmentation will give a very noisy output for a complex background as well as for a background with multiple signers. So, we introduce a contour processing operation to counteract such background situations. The detailed working of this operation for single as well as double-handed signs is described in [87].

The flowchart of the contour processing stage for continuous ASL fingerspelling recognition is shown in Fig. 4.2. In this stage, at first all the contours present in the input frame are found out. Followed by it, the largest contour is computed. There might be a situation where an unintended signer in the background may have a contour greater than that of the actual signer. In such situations, an incorrect signer hand might be reported as a correct signer hand output. So, to find out the correct signer hand, contour matching with the previous largest contour is done. Finding contour moments that take into account the entire structure of the contours being matched is the best way for finding contour matching. A moment is the gross characteristic of the contour computed by integrating over all the pixels of the contour. But we often use normalized moments so that they become scale invariant [90]. Hu invariant moments are a linear combination of normalized central moments so that they are invariant of translation, rotation, and scale change [91]. So, after computation of the largest shape, contour

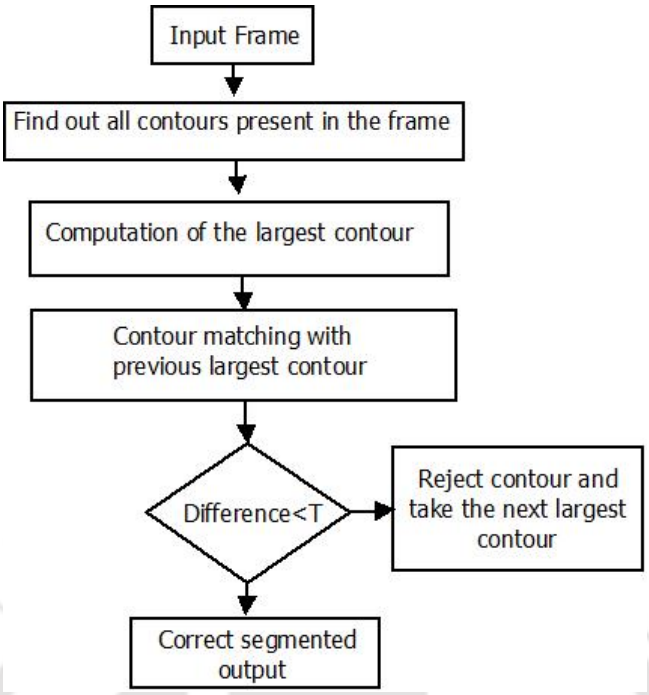


FIGURE 4.2: Flowchart of the contour processing stage.

matching using Hu invariant moments is performed between the largest contour in the previous frame and that in the current frame. The match value is given by

$$I(A,B) = \sum_{i=1}^{7} |1/(m_i^A) - 1/(m_i^B)| \tag{4.1}$$

m_i^A and m_i^B are defined as:

$$m_i^A = \text{sign}(h_i^A).\log|h_i^A| \tag{4.2}$$

$$m_i^B = \text{sign}(h_i^B).\log|h_i^B| \tag{4.3}$$

where h_i^A and h_i^B are the Hu moments of contours A and B respectively [90]. Lesser the value of $I(A,B)$, the more is the similarity between A and B.

For contour matching, a training and testing method is employed. A threshold value of difference (T) is selected empirically and if the obtained value of difference is less than T, it will be interpreted as correct segmented output, otherwise it will be rejected.

The qualitative and quantitative results justifying the efficiency of the hand segmentation module are illustrated in Section 4.3.1.

4.2.2 Movement epenthesis (*me*) detection

Based on the review of related work in the domain of continuous fingerspelling recognition, it is seen that most of the existing techniques use explicit modeling of *me* to separate out sign and *me* parts from continuous fingerspelling sequences. However various patterns of *me* may exist between meaningful sign patterns. So, to form the *me* model, a huge database of all possible *me* patterns will be required which will be a cumbersome process. In this context, our proposed technique does not require any such model for detecting *me*, instead an adaptive threshold is formulated to distinguish the minute characteristics between sign and *me* frames using a fusion of three unique features. This technique is adaptive for all possible combinations of fingerspelling sequences. The block diagram of the proposed *me* detection technique is shown in Fig. 4.3.

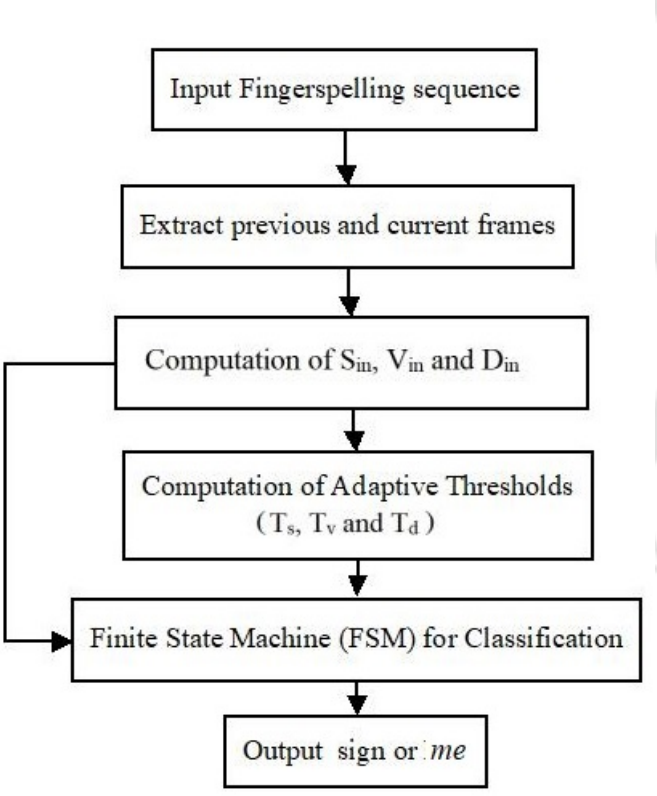


FIGURE 4.3: Block diagram of the *me* detection technique for fingerspelling sequence involving local motion.

Researchers have used many features for continuous fingerspellings recognition. Goh and Holden [92] extracted geometric and motion features using optical flow. Lee and Kim

[12] used motion and location of the hand segmentation to classify a signed utterance into sign, fingerspelling and non-sign segments (*me*). Again, they used the hand shape feature to identify the alphabets in the fingerspelling segment. Ricco and Tomasi [34] used changes of hand shapes over time to recognize fingerspelling sequences from ASL. Kim et al. [1] used linguistic handshape features using the Histogram of Oriented Gradients (HOG) descriptor, along with expected motion profiles for recognition of continuous fingerspellings. Aly et al. [52] used PCANet for feature extraction to recognize fingerspellings. Nguyen and Do [43] used HOG and Local Binary Pattern (LBP) features of each fingerspelled letter for fingerspelling recognition. Yang and sarkar [24] used a motion snapshot based representation, capturing short term movement over a few frames using the KLT (Kanade-Lucas-Tomasi) method.

We define three features-

1. Handshape matching using Zernike moments (S_{in}),
2. Velocity change of gesturing hand using Lucas-Kanade optical flow method (V_{in}) and
3. Displacement of palm centroid using geometric moments (D_{in}).

The three proposed parameters shall also model assimilation and dissimilation phenomena, which hold significant importance while modeling *me* in continuous fingerspelling sentences involving local motion.

Methodology:

At first, the current and previous frames are acquired from the input fingerspelling sequence. Following it, the values of the three proposed features S_{in}, V_{in}, and D_{in} for every successive pair of frames are computed. Now, it is observed that the hand remains nearly static (i.e. no abrupt change occurs) during the signing period compared to the transition phase. We consider this property for modeling *me* in local motion. So, during the transition from a sign frame to a *me* frame, the value of all the three parameters (S_{in}, V_{in}, and D_{in}) will increase and reach a peak value. On the other hand, during the

transition from a *me* frame to a sign frame, the values of S_{in}, V_{in}, and D_{in} will decrease and remain almost constant until the next transition occurs.

So, by selecting appropriate threshold values T_s, T_v, and T_d for S_{in}, V_{in}, and D_{in} respectively by means of an adaptive thresholding technique, we can easily classify whether a particular frame is a sign or *me* depending on whether S_{in}, V_{in}, and D_{in} are less than or greater than their respective thresholds T_s, T_v, and T_d. The classification of sign and *me* frames is performed by means of an FSM model. The FSM uses S_{in}, V_{in}, and D_{in} and their respective thresholds T_s, T_v, and T_d as inputs for performing this classification.

Another matter of concern is that a fixed threshold level will not be effective for all possible fingerspelling sequences. This problem is also observed while spotting continuous ASL signs as reported in [31]. So, to overcome this difficulty, we adopt an adaptive thresholding technique which is effective for any possible fingerspelling sequence which is described in Section 4.2.2.

Proposed features for modeling *me* in local motion

We propose three features for modeling *me* in local motion, since a single feature may not have the ability to discriminate among sign and *me* frames, of all possible fingerspelling sequences.

1. **Shape matching using Zernike Moments:** Zernike moments are a class of orthogonal moments that are effective for image representation. They are rotation, scale, and translation invariant and can be constructed to any arbitrary order. Teague [93] first introduced the use of Zernike moments to overcome the shortcomings of information redundancy present in the popular geometric moments like Hu invariant moments [94].

   Zernike moments: Zernike [95], [96] introduced a set of complex polynomials which form a completely orthogonal set over the interior of the unit circle, i.e. $x^2 + y^2 = 1$. Let the set of these polynomials be denoted by $\{V_{nm}(x, y)\}$. The

form of these polynomials is:

$$V_{nm}(x, y) = V_{nm}(\rho, \theta) = R_{nm}(\rho) \exp(jm\theta) \tag{4.4}$$

where,

n Positive integer or zero

m Positive and negative integers subject to constraints $n - |m|$ even, $|m| \leq n$.

ρ Length of the vector from the origin to (x, y) pixel

θ The angle between vector ρ and $x-$ axis in the counterclockwise direction

$R_{nm}(\rho)$ Radial polynomial defined as

$$R_{nm}(\rho) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!} . \rho^{n-2s} \tag{4.5}$$

These polynomials are orthogonal. Zernike moments are the projection of the image function onto these orthogonal basis functions. The Zernike moment of order n with repetition m for a digital image function $f(x, y)$ that vanishes outside the unit circle is

$$Z_{nm} = \frac{n+1}{\pi} \sum_{x} \sum_{y} V^*_{nm}(\rho, \theta) f(x, y); \qquad x^2 + y^2 \leq 1 \tag{4.6}$$

To compute the Zernike moments of a given image, the center of the image is taken as the origin and pixel coordinates are mapped to the range of unit circle, i.e. $x^2 + y^2 \leq 1$. Those pixels falling outside the unit circle are not used in the computation. Also, $Z^*_{nm} = Z_{n-m}$.

The Zernike moments on a rotated image differ from those of the original unrotated image in phase shifts, but not in magnitudes. Therefore, $|Z_{nm}|$ can be used as a rotation-invariant feature of the image [94].

In our work, we implement Zernike moments for describing the shape of the signs in the current and previous frames and thereby computing the similarity between them using the expression

$$S_{in}(prev, curr) = \left| \frac{1}{Z_{prev}} - \frac{1}{Z_{curr}} \right| \tag{4.7}$$

where, Z_{prev} and Z_{curr} are the Zernike moments for signs in previous and current frames respectively.

S_{in}(prev, curr) is minimum for almost similar handshapes and it increases as the variation between any two shapes increases. In the case of continuous fingerspelling sequence, during the articulation of signs, the handshapes between two successive frames are almost similar as the movement of hand and fingers becomes almost static and hence S_{in} is minimum. In contrast, the similarity between the handshapes decreases during the transition from one sign to the next (*me* phase), and hence S_{in} increases.

Using this experiment, we analyze the shape matching profile variations for different order Zernike moments and have found that S_{in} using Zernike moment Z_{42} provides a distinct variation in characteristics which shall subsequently enable us to distinguish between sign and *me* frames easily. The idea of the selection of Zernike moment order is similar to the paper in Ref. [94], where they tested recognition rate for various classifiers using different orders of Zernike moment to check the one giving optimum result.

Here, we compare the shape matching results by taking Zernike moments of different orders and experimentally, Zernike moment of order n = 4 and m = 2 (i.e. Z_{42}) provided us with optimum results. So, while calculating S_{in} between previous and current frames, we use Zernike moments of order n = 4 and m = 2.

2. *Velocity change using Lucas-Kanade optical flow method:* To measure the minute change in velocity for every successive frame, we consider the motion of salient corner points as detected by the Lucas-Kanade (LK) optical flow method. Optical flow is a method used for estimating motion between frames of video. The optical flow that is computed for each pixel in a frame could be used to determine which regions of the video frame contain large amounts of flow or motion [65]. The LK algorithm relies only on local information that is derived from some small window surrounding each of the points of interest. This method calculates the displacement vectors of individual features rather than tracking all of the pixels within a frame and rendering a full-motion vector field [97]. Then LK optical flow

equation describing the velocity of a particular pixel in 2D is given by:

$$I_x u + I_y v + I_t = 0 \tag{4.8}$$

$$\Rightarrow \nabla I^T V_{in} = -I_t \tag{4.9}$$

where, I_x and I_y are derivative of intensity over space and I_t is the derivative between images over time. u and v are the x-component and y-component of velocity [90], $V_{in} = \begin{bmatrix} u \\ v \end{bmatrix}$ and $\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}$.

3. *Displacement of centroid using geometric moments:* For detecting small changes in the position of the centroid between successive frames, we calculate the displacement of the centroid of the sign segment from the previous position using

$$D_{in} = \sqrt{(X - X_{prev})^2 + (Y - Y_{prev})^2} \tag{4.10}$$

where, (X, Y) and (X_{prev}, Y_{prev}) are the centroids of the sign segments in the current frame and previous frame respectively. The centroid (X, Y) is defined as [98]

$$(X, Y) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right) \tag{4.11}$$

where, M_{pq} is the moment of the image $I(x, y)$ and is defined as

$$M_{pq} = \sum_{i=1}^{n} I(x, y) x^p y^q \tag{4.12}$$

where, p is the x-order and q is the y-order. The summation is over all the pixels of the image (as denoted by n in Eq. (4.12)) [90].

Adaptive thresholding methodology

While performing fingerspelling sequences, assimilation and dissimilation phenomena often occur because of which there is a variation of ups and downs in profiles for different

sequences. Due to these variations, a fix threshold cannot serve the purpose of discriminating between sign and *me* frames for all possible fingerspelling sequences. Thus an adaptive threshold is necessary. Here, we formulate a statistical approach for determining the adaptive thresholds T_s, T_v, and T_d for S_{in}, V_{in}, and D_{in} respectively. The proposed model for adaptive thresholding is shown in Fig. 4.4. The actual threshold value at a

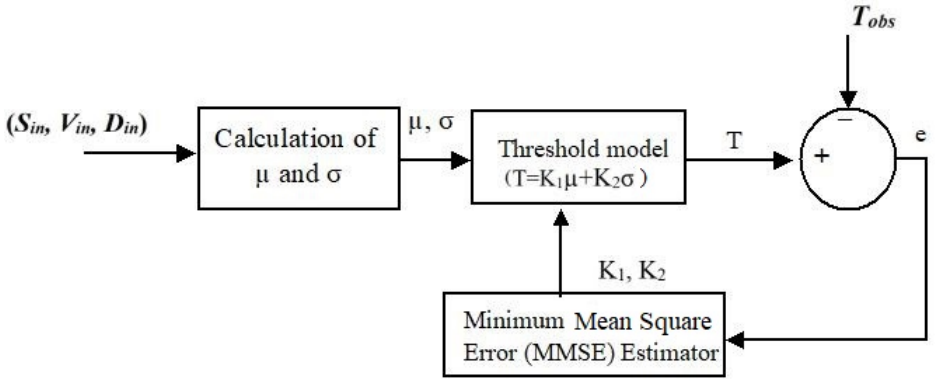


FIGURE 4.4: Proposed model for adaptive thresholding.

particular frame is calculated using [98]

$$T = K_1.\mu + K_2.\sigma \tag{4.13}$$

where, μ and σ are the mean and standard deviation of the input S_{in}, V_{in} and D_{in} values up to the current frame. K_1 and K_2 are constants that are profile dependent.

The mean (μ) at a particular frame is given by [99]

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x(i) \tag{4.14}$$

The standard deviation (σ) at a particular frame is given by [99]

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x(i)-\mu)^2} \tag{4.15}$$

where, x is the input i.e. S_{in}, V_{in}, and D_{in}, and n is the frame count.

Thus in Eq. (4.13), the parameters μ and σ give the statistics of the profile up to a particular frame and to make the threshold adaptive, the constants K_1 and K_2 have to be adaptive.

The constants K_1 and K_2 should be selected such that when the deviation increases (during the transition from sign to *me*), the threshold level should fall and when the deviation decreases (during the transition from *me* to sign), the threshold level should rise up. But, it is difficult to select fixed constants K_1 and K_2 that will be effective for all fingerspelling sequences empirically due to assimilation and dissimilation phenomena. So, to estimate the constants K_1 and K_2, we have incorporated a minimum mean square error (MMSE) estimator which is fed with an error signal (e).

The error signal (e) at a particular frame is defined as

$$\begin{aligned} e &= T - T_{obs} \\ \Rightarrow e &= K_1.\mu + K_2.\sigma - T_{obs} \end{aligned} \tag{4.16}$$

where, T_{obs} is the observed threshold value at the given frame. From the experimental observation of shape matching (S_{in}), velocity change (V_{in}), and displacement profiles (D_{in}), it is defined as:

$$T_{obs} = \begin{cases} max_{value}, & input < threshold; \\ max_{value}/100, & input \geq threshold. \end{cases} \tag{4.17}$$

Here, input indicates either S_{in} or V_{in} or D_{in}, the threshold is a fixed value. Experimentally, it is taken to be 6.5, 6.5, and 1.0 respectively for S_{in}, V_{in}, and D_{in}. (However, these fixed threshold values may not be practically applicable for different types of fingerspelling sequences. So, we have proposed the adaptive thresholding methodology). Further, max_{value} denotes the maximum value of the input (S_{in} or V_{in} or D_{in}) up to the current frame.

Now, the mean square error (MSE) is given as

$$MSE = E[e^2] \tag{4.18}$$

According to MMSE criteria, the MSE should be minimum [100]. Therefore, to obtain the MMSE estimate of K_1 and K_2, we need to solve

$$\frac{\partial}{\partial \mathrm{K}_1}(\mathrm{MSE}) = 0 \tag{4.19}$$

and,

$$\frac{\partial}{\partial \mathrm{K}_2}(\mathrm{MSE}) = 0 \tag{4.20}$$

Solving Eq. (4.19) and Eq. (4.20), we have

$$\Rightarrow \mathrm{K}_1 = \frac{\mathrm{T}_{\mathrm{obs}}.\mathrm{E}[\sigma] - \mathrm{K}_2.\mathrm{E}[\sigma^2]}{\mathrm{E}[\mu].\mathrm{E}[\sigma]} \tag{4.21}$$

$$\mathrm{K}_2 = \frac{\mathrm{T}_{\mathrm{obs}}\left(\mathrm{E}\left[\mu\right] - \frac{\mathrm{E}[\sigma]\mathrm{E}\left[\mu^2\right]}{\mathrm{E}[\mu]\mathrm{E}[\sigma]}\right)}{\left(\mathrm{E}\left[\mu\right]\mathrm{E}\left[\sigma\right] - \frac{\mathrm{E}[\mu^2]\mathrm{E}[\sigma^2]}{\mathrm{E}[\mu]\mathrm{E}[\sigma]}\right)} \tag{4.22}$$

Thus, using the values of K_1, K_2, μ and σ, from Eq. (4.21), Eq. (4.22), Eq. (4.14) and Eq. (4.15) respectively, we can calculate the actual threshold using Eq. (4.13). The experimental results derived from the proposed adaptive thresholding technique is discussed in Section 4.3.2. The algorithm for determining adaptive thresholds $\mathrm{T_s}$, $\mathrm{T_v}$, and $\mathrm{T_d}$ is shown in Algorithm 1.

Steps for determining adaptive thresholds $\mathrm{T_s}$, $\mathrm{T_v}$, and $\mathrm{T_d}$ are given below:

1. Select an appropriate window with window size $\mathrm{W_{in}}$. Place the window at the current frame (such that the last position of the window coincides with the current frame) and compute μ, σ and $\mathrm{max_{value}}$ (for $\mathrm{S_{in}}$, $\mathrm{V_{in}}$, and $\mathrm{D_{in}}$ respectively) from the input $\mathrm{S_{in}}$, $\mathrm{V_{in}}$, and $\mathrm{D_{in}}$ values which are within the window. If the frame count (n) is less than the window size ($\mathrm{W_{in}}$), then the sequence is padded with ($\mathrm{W_{in}}$-n) zeros and then the computation is done.

   Here, window size $\mathrm{W_{in}}$ has to be selected such that it encompasses an adequate number of frames showing the subtle variations of $\mathrm{S_{in}}$, $\mathrm{V_{in}}$, and $\mathrm{D_{in}}$. Experimentally, it is found that choosing window size $\mathrm{W_{in}} = 40$ frames is sufficient to track the immediate change at a particular frame with respect to the previous ones. This reduces the computational complexity and also provides satisfactory results.

Algorithm 1 Algorithm for determining adaptive thresholds T_s, T_v, and T_d for a particular frame

Input: Shape matching (S_{in}), velocity change (V_{in}), displacement (D_{in}) profiles

Output: Adaptive thresholds T_s, T_v and T_d (for S_{in}, V_{in} and D_{in} respectively)

$i \leftarrow$ current frame number

$n \leftarrow 40$

$Th_{S_{in}} \leftarrow 6.5$

$Th_{V_{in}} \leftarrow 6.5$

$Th_{D_{in}} \leftarrow 1$

for $j = S_{in}, V_{in}, D_{in}$ **do**

$$\mu_j = \sqrt{\frac{1}{n}\left(\sum_{k=i-n+1}^{i} j(k)\right)}$$

$$\sigma_j = \sqrt{\frac{1}{n}\left(\sum_{k=i-n+1}^{i} \left(j(k) - \mu_j\right)\right)}$$

if $j < Th_j$ **then**

$T_{obs} = \max[j(i-n+1), j(i-n+2), ..., j(i)]$

else

$T_{obs} = \max[j(i-n+1), j(i-n+2), ..., j(i)]/100$

end if

$$K_{2_j} = \frac{T_{obs}\left(E[\mu_j] - \frac{E[\sigma_j]E[\mu_j^2]}{E[\mu_j]E[\sigma_j]}\right)}{\left(E[\mu_j]E[\sigma_j] - \frac{E[\mu_j^2]E[\sigma_j^2]}{E[\mu_j]E[\sigma_j]}\right)}$$

$$K_{1_j} = \frac{T_{obs}.E[\sigma_j] - K_2.E[\sigma_j^2]}{E[\mu_j].E[\sigma_j]}$$

$$T_j = K_{1_j}.\mu_j + K_{2_j}.\sigma_j$$

end for

$T_s, T_v, T_d \leftarrow T_{S_{in}}, T_{V_{in}}, T_{D_{in}}$

2. Determine the observed threshold, T_{obs} (for S_{in}, V_{in}, and D_{in} respectively) using Eq. (4.17).

3. Determine K_1 and K_2 (for S_{in}, V_{in}, and D_{in} respectively) using Eq. (4.21) and Eq. (4.22).

4. Compute actual thresholds, T_s, T_v, and T_d (for S_{in}, V_{in}, and D_{in} respectively) using Eq. (4.13).

5. Repeat steps 1 to 5 until all the frames are processed.

TABLE 4.1: Parameters used in adaptive thresholding model

Parameters	**Description**
S_{in}	Shape matching between successive frames
V_{in}	Velocity change between successive frames
D_{in}	Displacement of centroid between successive frames
μ	Mean of input values up to current frame
σ	Standard deviation of input values up to current frame
K_1, K_2	Profile dependent constants
T	Actual threshold
T_{obs}	Observed threshold
e	Error signal
MSE	Mean square error

FSM model for classification of sign and *me* frames

An FSM is a sequential circuit whose past behavior can affect their future behavior in a finite number of ways, i.e. they are machines with a fixed number of states [101]. The FSM classifier makes the same successful classifications as artificial neural networks and binary classification trees. The main advantages of the proposed classifier are that it requires fewer iterations than neural networks and does not require rules such as splitting criterion as in binary decision trees [102]. Another advantage is that learning can be performed while data is flowing in one direction, without the need for a back-propagation

method such as that used in neural networks. The use of FSMs for classification has been done primarily on the classification of Polymerase Chain Reaction (PCR) primers for the maize [103]. Yadav and Corns [104] used Finite State Classifiers for the classification of PCR primers in mice with 60% accuracy. Elizabeth et al. [103] proposed a method for the detection of malignant melanoma using a Finite State Classifier and obtained a classification rate of 88%. Bhuyan et al. [105] proposed a FSM for gesture classification. Since a gesture can be defined as an ordered sequence of states in the spatial-temporal space, they represent a particular gesture as a sequence of key frames and the corresponding key frame duration, which constituted a FSM. They achieved an accuracy of 85%. In [106], Li and Principie proposed a novel state-based approach to insect flight dynamics modelling and automatic insect recognition using FSM and achieved an accuracy of 92.15%. Li et al. [107] proposed a framework based on FSM to model the robot action sequences in manipulation tasks. The framework showed the way in which a robot could slide its level of autonomy by considering the result of the transition action and the input of the human operator. Nirmalee and Ranatunga [108] developed an interactive read highlighter for text-heavy documents which employed the built-in webcam of a laptop, by limbus tracking combined with FSM for a smooth reading experience. The system also tracked the eye-blinking to alert the user of drowsiness while reading. Eye-tracking was performed using the Haar cascade classifier followed by iris detection using image processing techniques. The FSM was implemented to alleviate for the insufficiency of vertical accuracy with the underlying assumption that the user reads the text without skipping back and forth across lines. Kim and Kim [109] proposed a novel abandoned and removed objects (AROs) classification using hierarchical FSM. Sales et al. [110] proposed an approach applied to autonomous mobile robot's navigation integrating localization and navigation using a topological map based on the proposed Adaptive FSM technique. Isabel et al. [111] proposed a framework to analyse human behaviour using FSM. Zaragova and Oncina [112] presented a statistical model to recognize pen-based music compositions using stroke recognition algorithms and FSM and achieved an accuracy of 90%.

While performing adaptive thresholding, a situation might occur, where all the three parameters S_{in}, V_{in}, and D_{in} may not be greater than their respective thresholds for

classifying a particular frame as *me*. So, we impose a logic whereby if any two of the three parameters S_{in}, V_{in}, and D_{in} are greater than their respective thresholds for a particular frame, then that frame will be classified as *me* frame, or else it will be treated as sign frame. This logic is implemented mathematically, using an FSM model, which is capable of segregating the sign and *me* frames from a continuous fingerspelling sequence. The reason behind implementing an FSM model for this purpose is that such a logical operation based classifier will be suitable compared to supervised models such as HMM, CRF, Support Vector Machine (SVM), Decision Tree, etc. from computational time, training requirement and hardware implementation perspectives.

The overall process of classification of sign and *me* frames based on our proposed FSM model is shown in the block diagram of Fig. 4.5. For a particular frame, the FSM model will check the values of feature combinations (S_{in}, V_{in}, D_{in}) in all the possible states (S, V, D) by comparing them with adaptive thresholds T_s, T_v, T_d and decide whether it is a sign or *me*. As seen from Fig. 4.5, the adaptive threshold computation block will provide the thresholds T_s, T_v, and T_d respectively for the features S_{in}, V_{in} and D_{in} respectively. These input features are first binarized by comparing them with their respective thresholds. Based on the binarized feature values ($S_{in}^{'}$, $V_{in}^{'}$, $D_{in}^{'}$) and the current state (SVD), the input X to the FSM is determined. It can take either of the 2 values: 0 or 1. X = 1 if all the selected inputs are greater than their respective thresholds, otherwise X = 0. For e.g., if the current state (SVD) is 110, this implies that the selected inputs are shape matching (S_{in}) and velocity change (V_{in}). In this case, X will be 1 only if both the selected inputs S_{in} and V_{in} are greater than T_s and T_v respectively. So, the input X is implemented using an 8×1 multiplexer (MUX) as shown in Fig. 4.6. In our model, we consider '000' and '111' as end states, so input conditions are not considered for these cases in the MUX. Thus, using the 8×1 MUX of Fig. 4.6, the expression of input X of the FSM is defined as

$$\begin{aligned} X = S.\overline{V}.\overline{D}.S_{in}^{'} + \overline{S}.V.\overline{D}.V_{in}^{'} + \overline{S}.\overline{V}.D.D_{in}^{'} + \\ S.V.\overline{D}.S_{in}^{'}.V_{in}^{'} + \overline{S}.V.D.V_{in}^{'}.D_{in}^{'} + S.\overline{V}.D.S_{in}^{'}.D_{in}^{'} \end{aligned} \tag{4.23}$$

Finally, based on the computed value of X and the current state SVD, the FSM model will determine the output Z which signifies whether the current frame under consideration is

sign or *me*.

The block diagram of the FSM for modeling *me* in local motion is shown in Fig. 4.8. The different variables used for describing the behavior of the FSM are listed in Table 4.2. For an incoming frame, the next state $S_1V_1D_1$ based on current state SVD and current input X. This will be the current state in the next iteration. Simultaneously, the output logic will determine the output Z based on current state SVD and current input X and checks for all possible combinations for the present frame to be a *me* or sign frame.

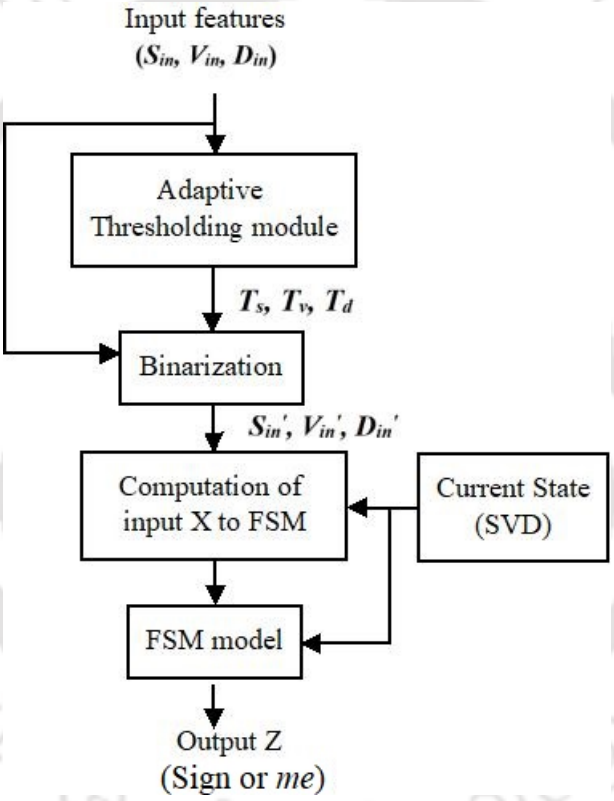


FIGURE 4.5: The overall process of classification of sign and *me* frames based on our proposed FSM model.

The working of the FSM model can also be summarized by the state diagram in Fig. 4.7, and corresponding state Table 4.3. In Fig. 4.7, '100' is the start state and there are two end states i.e, '000' (representing sign state) and '111' (representing *me* state). As the machine does not transit to any other state from the end states, so they are marked as don't care ('d') in the state table. The expressions for output (Z) and next state variables (S_1, V_1, and D_1) using their respective K-maps (derived based on the

TABLE 4.2: Parameters used in FSM model

Parameters	**Description**
SVD	Current state
$S_1V_1D_1$	Next state
X	Input (0 or 1)
Z	Output (0 or 1)
Z_{stop}	Stopping condition
X/Z	Input and corresponding output for a particular state

state table 4.3) are given as

$$Z = S.V.X + V.D.X + S.D.X \tag{4.24}$$

$$\begin{aligned} S_1 &= S.V + S.X + V.D.X \\ V_1 &= V.X + S.X + \overline{V}.\overline{D} \\ D_1 &= V.\overline{D} + V.X + S.D.X \end{aligned} \tag{4.25}$$

As shown in the state diagram, if for a particular frame the output Z = 1 and the next state $S_1V_1D_1$ is '111', then the machine will stop iterating and that particular frame will be classified as *me* frame. On the other hand, if the output Z = 0 and the next state $S_1V_1D_1$ is '000', then the machine will stop iterating and that particular frame will be classified as a sign frame. This condition is called the stopping condition (Z_{stop}) of the FSM and is implemented using Eq. (4.26).

$$Z_{stop} = (S_1 + V_1 + D_1) \oplus Z \tag{4.26}$$

Thus as indicated in the block diagram of Fig. 4.8, if Z_{stop} is 0, the current state register will not be triggered in the next iteration and as a result, the FSM will stop. The classification results obtained from the proposed FSM model is described in Section 4.3.3.

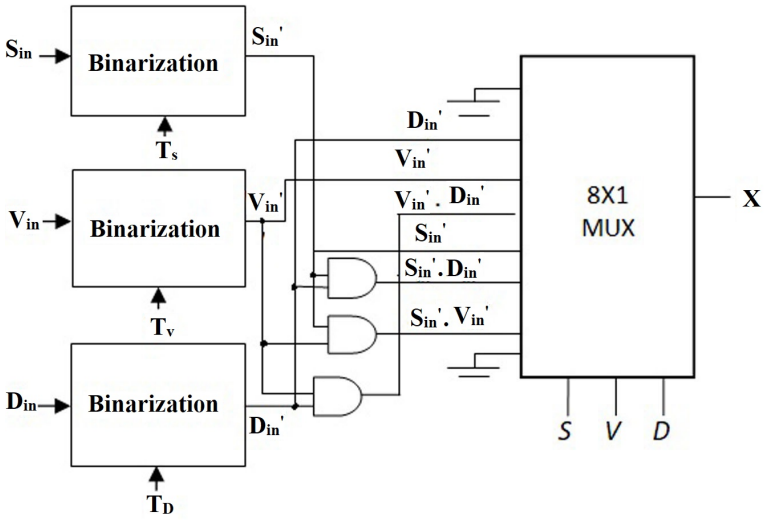


FIGURE 4.6: An 8×1 MUX for implementing the input (X) of the FSM.

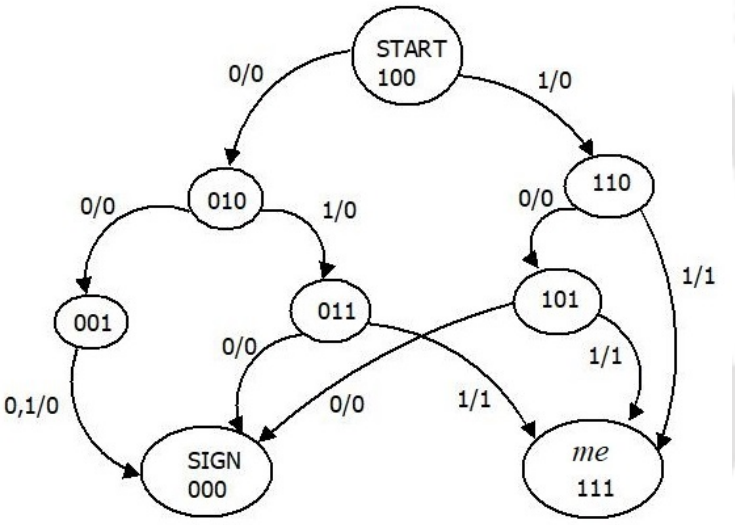


FIGURE 4.7: State diagram of the proposed FSM model.

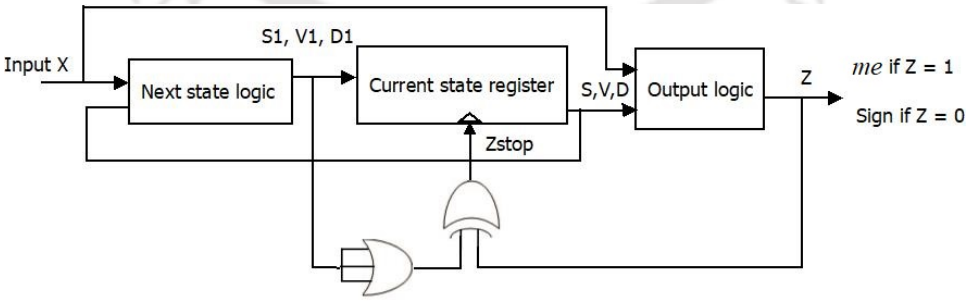


FIGURE 4.8: Block diagram of the proposed FSM model for classifying sign and *me* frames.

4.2.3 Feature extraction process

After obtaining the valid sign segments from the output of the FSM model, it is necessary to extract some characteristic features from the boundary (or contour) of the extracted

TABLE 4.3: State table of FSM for classifying sign and *me* frames

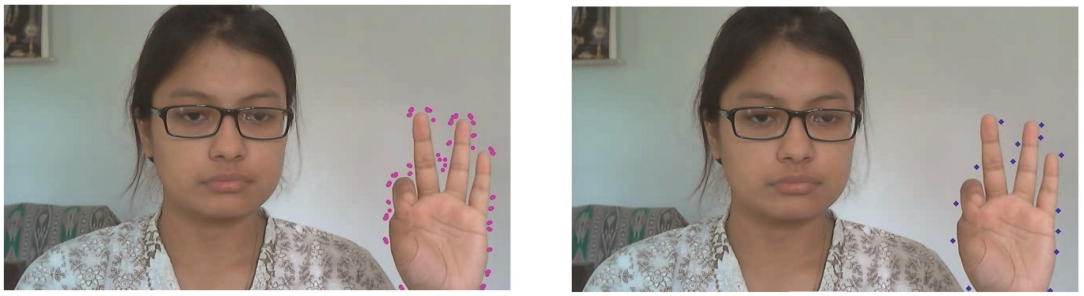
Present State			**Input**	**Next State**			**Output**
S	***V***	***D***	***X***	S_1	V_1	D_1	***Z***
0	0	0	0	'd'	'd'	'd'	'd'
0	0	0	1	'd'	'd'	'd'	'd'
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	1	0
0	1	0	1	0	1	1	0
0	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	0	1	0	0
1	0	0	1	1	1	0	0
1	0	1	0	0	0	0	0
1	0	1	1	1	1	1	1
1	1	0	0	1	0	1	0
1	1	0	1	1	1	1	1
1	1	1	0	'd'	'd'	'd'	'd'
1	1	1	1	'd'	'd'	'd'	'd'

sign, so as to describe the shape of the sign accurately. The feature extraction process is characterized into three phases:

Detection of corner points

The contour of the valid sign segment will consist of innumerable points. So, for dimensionality reduction, the corner points (a point that has more information about a curve than other points) of the contour are extracted out, which will reduce the number of points considerably. Corner point detection is carried out, by means of a corner detection algorithm proposed by Chetverikov [113]. It is a two-step algorithm. In the first step, the corner detector selects the candidate corner points, by scanning along the

contour and constructing triangles on the interior of the contour according to some assumptions. The second step is a post-processing step, to remove superfluous candidates. The corner points detected for alphabet 'F' are shown in Fig. 4.9(a).



(a) (b)

FIGURE 4.9: Detection of (a) Corner points (b) Maximum curvature points for fingerspelling alphabet "F".

Detection of maximum curvature points

To further reduce the dimensionality of the feature points, we need to find out only those significant points, which are sufficient to contribute to the shape of the sign. For this purpose, we compute the maximum curvature points (MCPs) from the available corner points of the contour. MCPs are calculated from the available corner points of the contour, by selecting only those points for which the degree of curvature is greater than a selected threshold.

The steps for detecting the MCPs are as follows:

1. Calculate the slope (θ_1, θ_2) for every consecutive corner point (C_1, C_2).
2. The points for which the slope difference $(\Delta\theta = |\theta_1 - \theta_2|)$ is greater than a predefined threshold, are considered as MCPs.

The maximum curvature points detected for alphabet 'F' are shown in Fig. 4.9(b). It is seen from Fig. 4.9 that the number of corner points is considerably reduced by extracting out the MCPs. These MCPs are determined by comparing the absolute slope difference of consecutive corner points with a selected threshold value of 1.20^o. The Fourier descriptors are computed for these detected MCPs which represent our feature vector.

Computation of Fourier descriptors

Fourier descriptor (FD) is a contour-based shape descriptor, which takes into account only the pixels present in the boundary or contour of an object [114].

The Fourier descriptors are computed for each of the detected MCPs for describing the shape of a sign. The FD (based on Discrete Fourier Transform (DFT)) of an N-point boundary is given by [91]

$$F(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}}; k = 0, 1, ..., N-1 \tag{4.27}$$

Thus, our feature vector (FV), depicting the shape of a fingerspelling alphabet is defined as

$$FV = \{F(1), F(2), F(3), ..., F(N-1)\} \tag{4.28}$$

where, $F(1), F(2), \cdots F(N-1)$ are the FD and N is the number of MCPs.

Advantages of the proposed feature vector: The main advantages of our proposed feature vector are:

- Invariant to translation, rotation, and scaling.
- The original shape of the sign can be reconstructed by finding out Inverse Discrete Fourier Transform (IDFT).
- Operation is on a limited number of pixels since the feature extraction process considers the aspect of dimensionality reduction.

These attributes contribute to the performance of the system.

4.2.4 Recognition using CRF classifier

A CRF classifier is statistical in nature based on conditional probability approaches for segmenting and labeling sequential data [31]. In our work, a CRF classifier is trained

with the feature vectors (comprising of FD sequence) of all fingerspelling alphabets of ASL (excluding alphabets 'J' and 'Z' which involve global motion). To measure the accuracy of the proposed system, the sign recognition rate is calculated using

$$R = \frac{C}{N} \times 100\% \tag{4.29}$$

where, C is the number of correctly recognized and N is the number of test fingerspelling alphabets [31]. The overall recognition results are presented in Section 4.3.4.

4.3 Experimental results and discussions

For experimental evaluation, we take 80 different fingerspelling sequences, of variable length from 3 to 5 letters from the vocabulary of 24 unique ASL alphabets (excepting 'J' and 'Z') as shown in Table 4.8. These fingerspelled sequences are performed by 4 different signers comprising of 2 native and 2 non-native signers (including both male and female signers). The frame rate is 15 frames/sec and the resolution is 640×360. In order to testify the robustness of our proposed system, we consider fingerspelling sequences performed by both native and non-native signers because in practical situations it is observed that, there is a wide variation between signs performed by these two categories of signers like

- The speed of articulation for native signers is more than that of non-native signers.
- The hold time for native signers is less than for non-native signers.

Further, *me* in fingerspelling, is typically characterized as assimilation (where sequential hand shapes become more similar to one another) or dissimilation (where sequential hand shapes become more different) [8]. So, our database of fingerspelling sequences is selected such that, all the alphabets of ASL (except 'J' and 'Z') are included and also to check whether our proposed system is capable of modeling *me* in all the four possible cases:

- Assimilation-Assimilation where all the three alphabets are almost similar - e.g., A-S-E.
- Dissimilation-Dissimilation where all the three alphabets are dissimilar - e.g., P-Q-G.
- Assimilation-Dissimilation where the 1st and 2nd alphabet are similar and, 2nd and 3rd alphabet are dissimilar - e.g., R-U-V.
- Dissimilation-Assimilation: where the 1st and 2nd alphabet are dissimilar and, 2nd and 3rd alphabet are similar - e.g., N-O-S.

Thus our experimental database comprises of:

- Number of fingerspelling sequences: 80 (20 for each signer).
- Number of alphabets involved in these sequences: 180.

The database can be extended to several other fingerspelling sequences comprising of different numbers of ASL alphabets with different combinations.
The experimental results obtained at different stages of our proposed system are presented in separate sections as these are related to different blocks, all of which make the complete system work. ‘

4.3.1 Hand segmentation results under various background conditions

To evaluate the performance of our proposed hand segmentation method, we captured some sign video data under various background conditions, such as complex backgrounds containing various objects and backgrounds with multiple signers. Also, some videos were captured under daylight and dim-light conditions. All these videos are standard videos of frame rate 15 frames/sec with resolution 640×360.

Fig. 4.10(a) and Fig. 4.10(c) show the inputs and outputs of our proposed model under complex background and background with multiple signers respectively. Fig. 4.10(b)

and Fig. 4.10(d) show the ground truth hand contours for corresponding frames respectively. Thus, it is seen that our system is robust to the complex background as well as

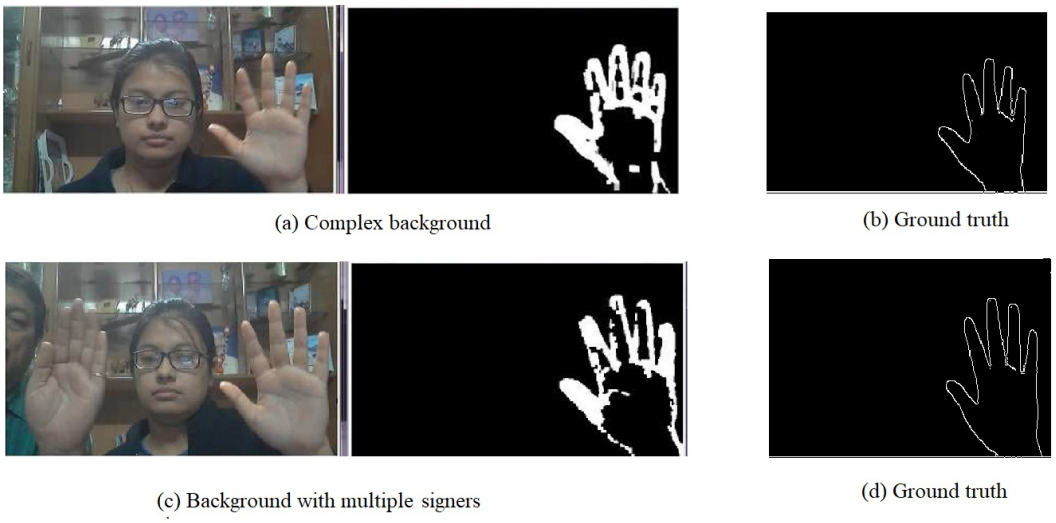


FIGURE 4.10: Segmented output using the proposed method.

for background with multiple signers, as it detects the hand of the intended signer only.

The proposed model was also tested under day-light and dim-light conditions, results of which are shown in Fig. 4.11(a) and Fig. 4.11(b) respectively. Here, the contour processing block plays a significant part in producing accurate segmentation results, where the shape matching threshold (T) was empirically selected as 0.605.

Our proposed method of hand segmentation proves to be more effective than traditional

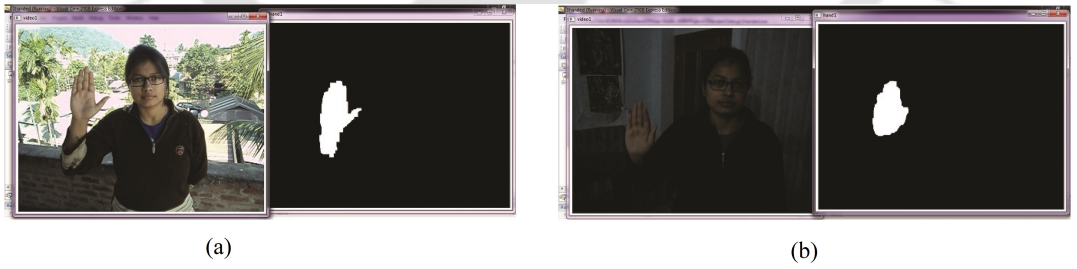


FIGURE 4.11: Segmented output using proposed model under (a) day-light condition (b) dim-light condition.

methods like skin color segmentation and frame differencing. This is justified by means of quantitative analysis of error rate based on a number of false positives (FPs) and a number of false negatives (FNs) as shown in Table 4.4. For a particular fingerspelling video sequence, the ground truth hand contour is manually derived for every frame (as shown in Fig. 4.10) and is compared with the obtained hand segmentation results. A

TABLE 4.4: Comparative segmentation results in terms of TN and FN

Approaches	Background	Total no. of frames	No. of FP	No. of FN	Error rate (%)
Skin Color Segmentation	Complex	165	32	0	19.39
	Multiple gesturers	480	32	0	6.67
	Daylight	435	28	0	6.64
	Dimlight	60	15	2	28.33
Frame differencing	Complex	165	32	0	19.39
	Multiple gesturers	480	32	2	7.08
	Daylight	435	32	0	7.35
	Dimlight	60	32	0	53.33
Skin color + frame differencing	Complex	165	1	0	0.6
	Multiple gesturers	480	30	0	6.25
	Daylight	435	25	0	5.75
	Dimlight	60	11	2	21.67
Proposed method	**Complex**	**165**	**0**	**0**	**0**
	Multiple gesturers	**480**	**0**	**0**	**0**
	Daylight	**435**	**1**	**0**	**0.23**
	Dimlight	**60**	**0**	**1**	**1.67**

false negative (FN) is spotted if a desired ground truth hand contour is not detected in the resulting frame of hand segmentation, while a false positive (FP) is spotted if an undesired contour is detected in the hand segmentation output which is different from the ground truth hand contour. Here, the number of frames with FPs and FNs obtained for different hand segmentation schemes for video sequences taken under four different background conditions are shown. To compute the error rate for all the cases, we consider the following formula:

$$\text{Error rate} = \frac{\text{FP} + \text{FN}}{\text{Total number of frames}} \times 100\% \tag{4.30}$$

As seen from Table 4.4, the proposed model of hand segmentation gives the least error rate in comparison to the other three methods. Hence, this method proves to be more robust and effective and is used for the subsequent steps.

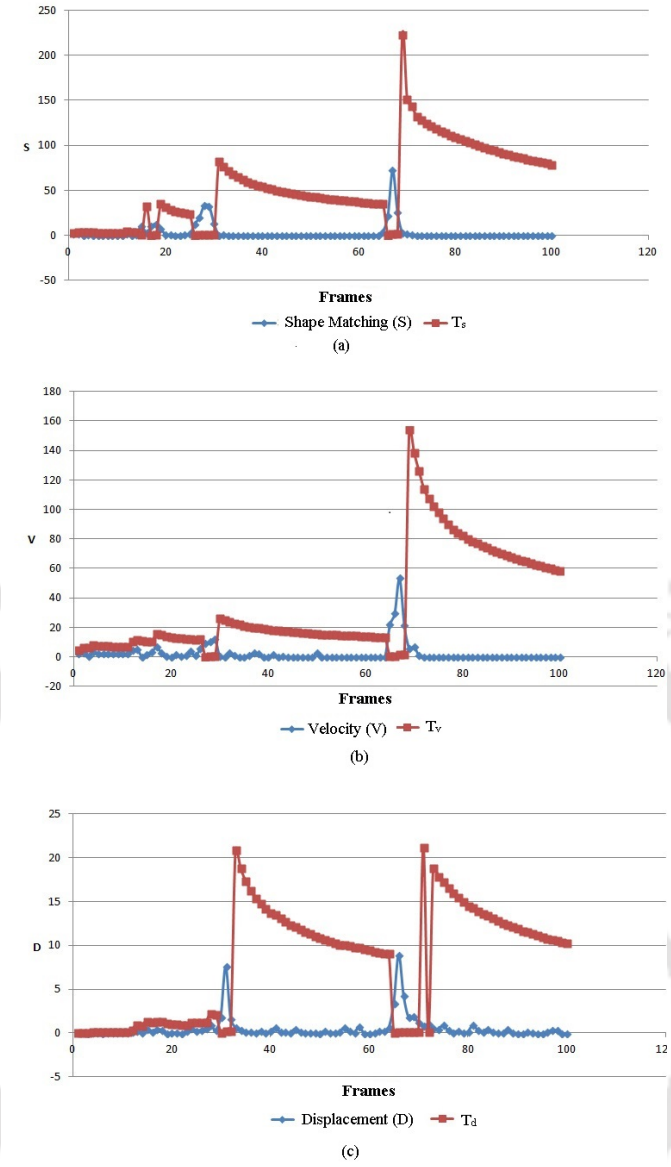


FIGURE 4.12: Variation of the proposed features and adaptive thresholds for fingerspelling sequence A-S-E (a) S and T_s (b) V and T_v (c) D and T_d.

4.3.2 Results of variation of proposed features for modeling *me* in local motion and adaptive thresholding

Here, we check the performance of the system by considering each of the individual features (hand shape, velocity of gesturing hand, and displacement of palm centroids) separately and also forming the composite set. The feature profiles of fingerspelling sequence A-S-E for the three features are shown individually in Fig. 4.12.

Here, we incorporate three features, because in certain cases a single feature may

not be able to catch the minute changes which occur during the transition from a sign to a *me* frame. So, in such cases, the remaining two features will play a complementary role to catch the characteristic change missed by that feature.

Also, a different profile derived from the composite set for the fingerspelling sequence A-S-E is shown in Fig. 4.13. The success rate for the case when individual features are

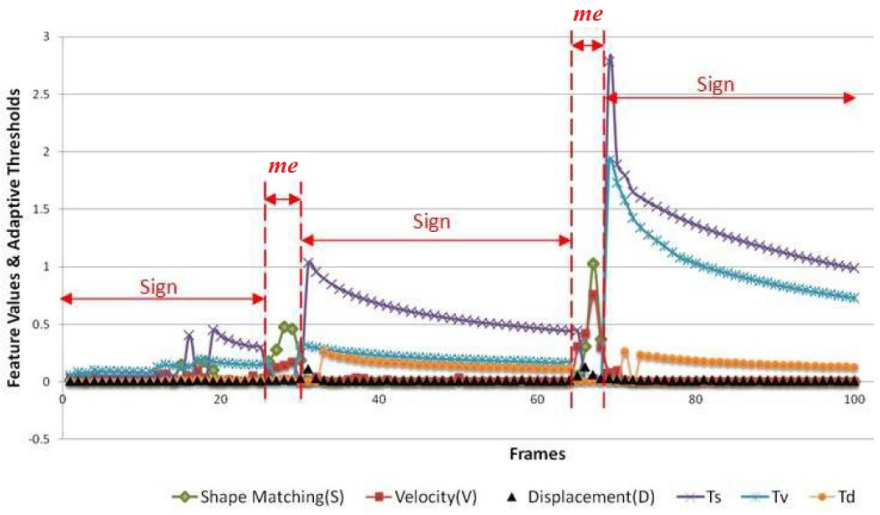


FIGURE 4.13: Combined profile showing the variation of the three proposed features and their corresponding adaptive thresholds for fingerspelling sequence A-S-E.

considered is around 81%, while it is about 94% while considering the combination of any two or all of the three features. This is an advantage of the proposed feature set which plays a supplementary complementary role in feature extraction. A frame that might be misclassified due to the incorrect response provided by one feature is overcome by considering the correct responses derived from the other two. Thus, the disadvantage of one feature is covered by the advantage of the other and vice versa. This helps in capturing precise frame details which aid subsequent processing.
Also, the error rate showing the percentage of misclassification taking different combinations of the proposed feature set is shown in Table 4.5.

The misclassification rates of combinations in Sl. nos. 4, 5 and 6 of Table 4.5 show that three specific feature forms namely $S_{in} + V_{in}$, $V_{in} + D_{in}$, and $S_{in} + D_{in}$ provide the minimum errors. The three feature combinations as shown in Sl. no. 7 shows a misclassification rate which is slightly more than the combinations in Sl. nos. 4, 5 and 6. This is because, for classifying a particular frame as a sign frame or *me* frame all the

TABLE 4.5: Misclassification results of the proposed feature combinations for 80 utterances performed by 4 signers

Sl. no.	**Feature combinations**	**Error rate (%)**			
		Signer 1	**Signer 2**	**Signer 3**	**Signer 4**
1	S_{in}	3.70	3.2	4	5
2	V_{in}	5.18	5	6	5.5
3	D_{in}	4.44	4.3	5.2	5.4
4	$S_{in} + V_{in}$	0.74	0.82	1.2	1.6
5	$V_{in} + D_{in}$	0.74	0.88	0.96	1.20
6	$S_{in} + D_{in}$	0.74	0.85	0.98	1.30
7	$S_{in} + V_{in} + D_{in}$	1.48	1.65	2.20	2.5

three features S_{in}, V_{in}, and D_{in} may not be less than or greater than their respective thresholds simultaneously. The decrease in average error is around 48% while considering two feature combinations instead of three. So, for our subsequent work based on FSM, the combinations in Sl. nos. 4, 5, and 6 are used.

4.3.3 FSM classification results

A sign frame is considered to be correctly spotted (c_s), if the following condition is fulfilled [84]:

$$c_s = \begin{cases} \text{True, if } (\Delta S + \Delta E) < 10 \\ \text{False, otherwise,} \end{cases} \tag{4.31}$$

where, $\Delta S = |g_s - s_s|$ and $\Delta E = |g_e - s_e|$. s_s and s_e are the start and end frame numbers of the spotted sign, respectively, and g_s and g_e are the start and end frame numbers of the ground truth of the spotted sign, respectively.

The classification results obtained at the output of our proposed FSM model for a few fingerspelling sequences performed by Signer 1 and Signer 2 are shown in Tables 4.6 and 4.7 respectively. Similar results for Signer 3 and Signer 4 are shown with the help of column charts in Fig. 4.14 and Fig. 4.15 respectively. As seen from the results, some letters are not spotted accurately according to Eq. (4.31). Although we consider diverse forms of fingerspelling sequences with variations in assimilation and dissimilation,

however, there are certain sequences with a high degree of assimilation such as A-S-E, N-O-S, P-Q-G etc. where the frame range of the spotted sign segments differ much from their ground truths. Fig. 4.16 depicts an example of the sequence A-S-E. In this sequence, letter "E" is not spotted accurately in the case of Signer 1 as observed from Table 4.6. This is because, for such sequence the variations in S_{in}, V_{in}, and D_{in} are very less which make the letters differentiable from one another very difficult. Due to this effect, the letters involved in these sequences are sometimes wrongly recognized which reduces the recognition accuracy.

TABLE 4.6: FSM classification results for Signer 1 (non-native signer)

Videos	Fingerspelling sequence	Number of fingerspelling segments	Ground-truth for fingerspellings	Spotted fingerspelling segments
1	A-S-E	3	A (1-23) S (33-59) E (76-81)	A (1-26) S (31-64) E (69-100)
2	R-U-V	3	R (1-30) U (42-68) V (73-104)	R (1-31) U (38-71) V (77-106)
3	P-Q-G	3	P (1-43) Q (55-126) G (144-188)	P (1-46) Q (56-132) G (141-208)
4	K-L-D	3	K (1-35) L (50-96) D (104-160)	K (1-41) L (53-100) D (108-161)
5	N-O-S	3	N (1-40) O (49-120) S (135-196)	N (1-47) O (57-126) S (130-205)
6	M-A-R-S	4	M (1-20) A (29-49) R (64-87) S (124-130)	M (1-20) A (26-51) R (63-100) S (112-132)

TABLE 4.7: FSM classification results for Signer 2 (non-native signer)

Videos	Fingerspelling sequence	Number of fingerspelling segments	Ground-truth for fingerspellings	Spotted fingerspelling segments
1	A-S-E	3	A (1-74) S (81-159) E (170-234)	A (1-74) S (81-166) E (172-231)
2	R-U-V	3	R (1-70) U (70-161) V (160-218)	R (1-72) U (77-157) V (164-219)
3	P-Q-G	3	P (1-80) Q (90-170) G (184-243)	P (1-74) Q (82-175) G (188-240)
4	K-L-D	3	K (5-80) L (94-160) D (180-227)	K (1-83) L (99-161) D (168-221)
5	N-O-S	3	N (1-105) O (123-193) S (209-287)	N (1-101) O (117-196) S (206-252)
6	M-A-R-S	4	M (1-30) A (38-60) R (84-117) S (134-150)	M (1-32) A (39-69) R (78-114) S (124-146)

Fig. 4.17 shows the sign and *me* frames obtained while signing the fingerspelling sequence P-Q-G by native Signer 3. Here, it is seen that a *me* phase occurs before signing the first alphabet i.e., P and also at the end of the last alphabet i.e., G. Apart from these, *me* phase also occurs during the transition from P to Q and Q to G. For signs showing assimilation, the transition time is usually less, and hence the number of *me* frames obtained is also less. On the contrary, for signs showing dissimilation, reverse is the case. This observation is justified from the FSM classification results shown in

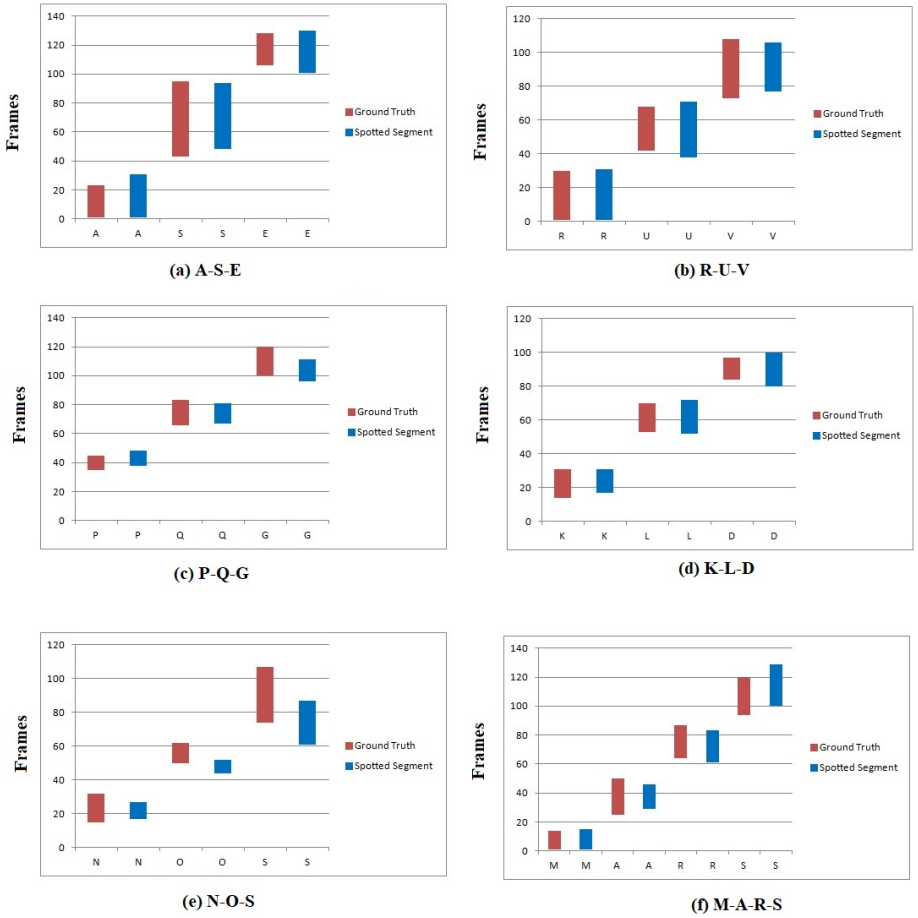


FIGURE 4.14: FSM classification results for Signer 3 (native signer).

tables.

4.3.4 Recognition results using CRF classifier

The recognition rates (RR) obtained for different signers using the trained CRF model are shown in Table 4.8. The CRF model was trained with 24 ASL fingerspelling alphabets except for J and Z (as mentioned in the dataset description in Section 4.3). Each alphabet was performed 10 times by each of the 4 signers resulting in a total of 960 training samples. Our CRF model was constructed using a DFT-based feature vector and trained using the Maximum Likelihood optimization technique [31]. On the other hand in the testing phase, we considered 80 fingerspelling sequences performed by the 4 signers, where the signs were first spotted using the combined adaptive thresholding and FSM model, and then recognized using the trained CRF model. Our proposed continuous SLR system offers an overall RR of around 91.29%. For non-native cases, the RR is better.

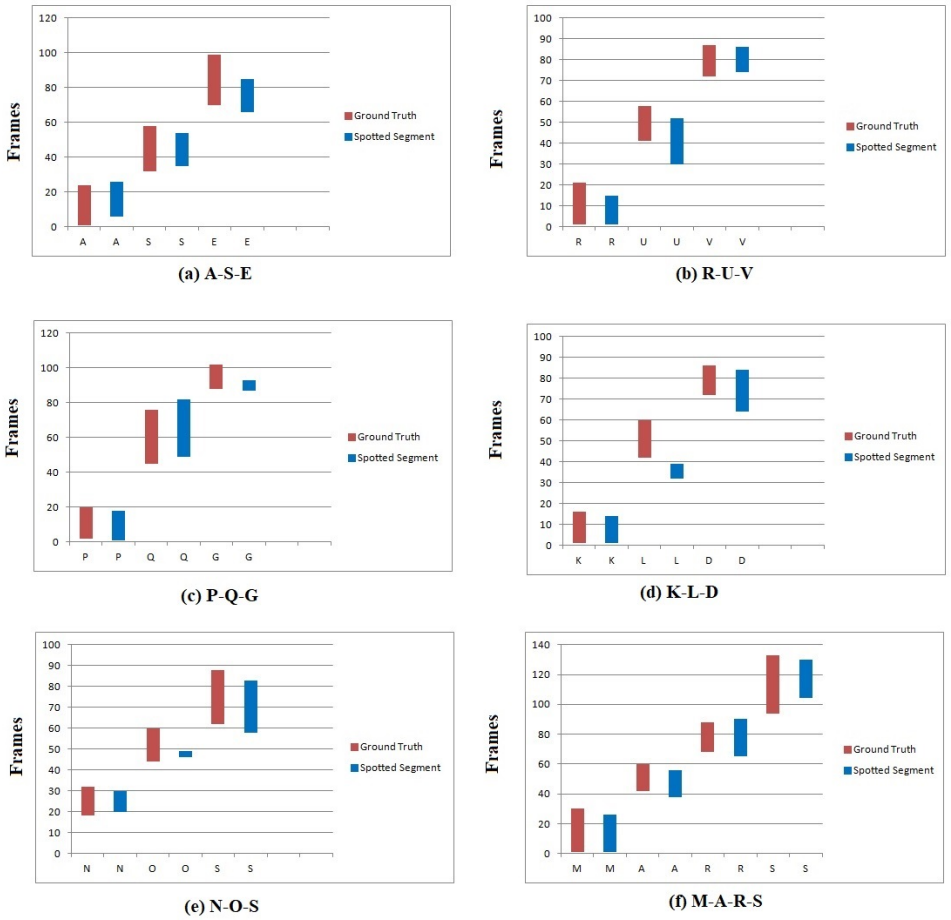


FIGURE 4.15: FSM classification results for Signer 4 (native signer).

FIGURE 4.16: Fingerspelling sequence A-S-E having high assimilation.

We compared our recognition results with that of a traditional SVM classifier. The results are depicted in the last column of Table 4.8. The multiclass SVM classifier is constructed using the method called the "one-against-one" as they are more suitable for practical use [115]. This method constructs $k(k-1)/2$ classifiers, where k is the total number of classes. We trained each classifier with data from two classes. The kernel used for training data is Radial Basis Function (RBF) kernel $K(x_i, x_j) \equiv e^{-\gamma||x_i - x_j||^2}$, where γ defines how much influence a single training example has. The larger γ is, the closer other examples must be to be affected.

We compared our proposed method with the work of Kim et al. [116] and Shi et al.

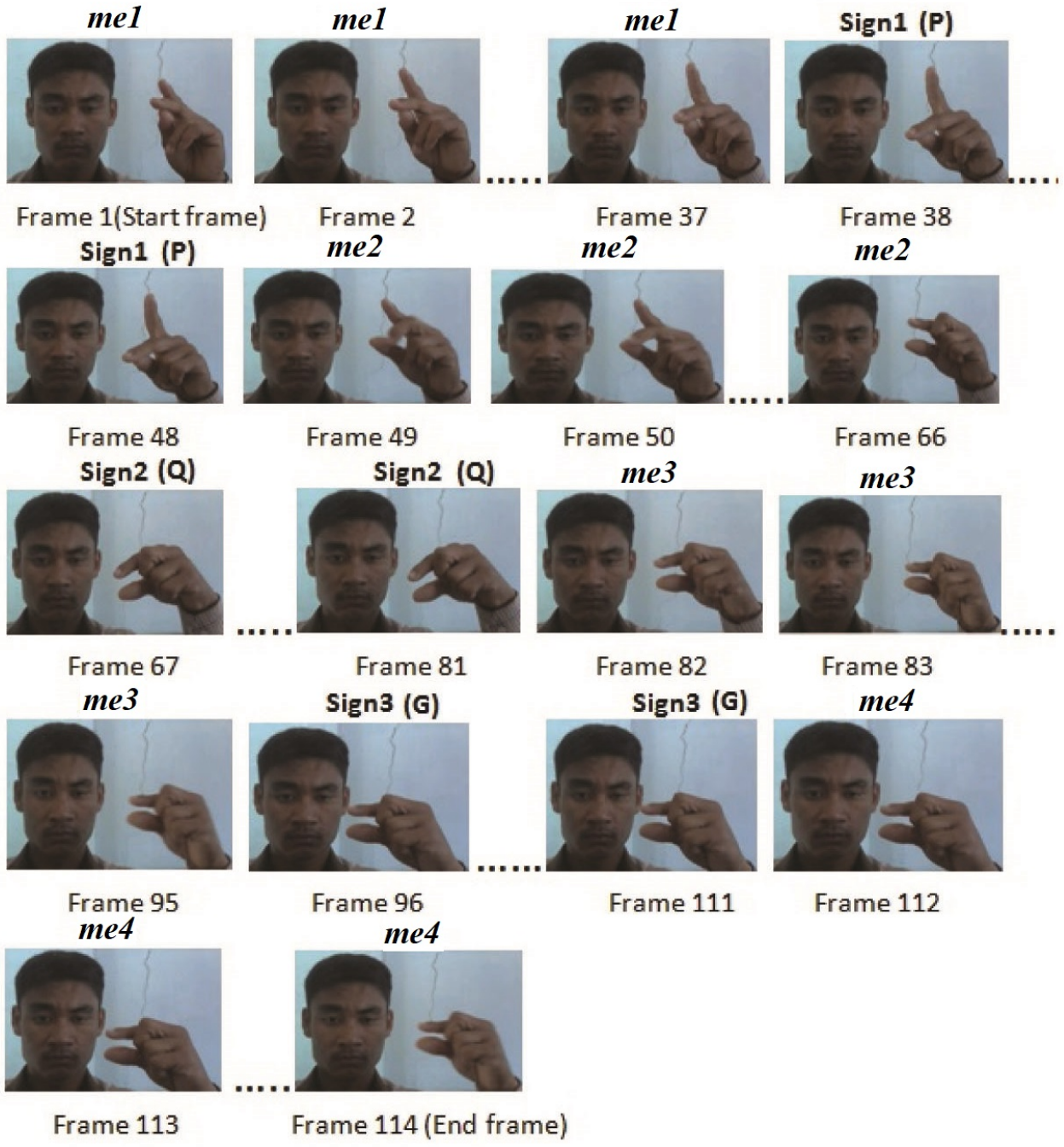


FIGURE 4.17: FSM classification results for fingerspelling sequence P-Q-G performed by Signer 3.

[117] as shown in Table. 4.9. Kim et al. [116] addressed the problem of recognition of fingerspelled letter sequences in ASL in a signer-independent setting. They first used a first-pass decoding segmental CRF which was independent of any frame-based recognizer. They used average DNN outputs over each segment, samples of DNN outputs within the

TABLE 4.8: Recognition results for continuous fingerspelling sequence involving local motion

Fingerspelling Sequences	RR (%) (Non-Native Signer)		RR (%) (Native Signer)		Overall RR (%)	Overall RR (%) (using SVM)
	Signer 1	Signer 2	Signer 3	Signer 4		
A-S-E, A-E-O, G-O-A C-O-E, M-A-Y, B-F-O, K-L-D, M-N-T, N-O-S, P-Q-G, R-U-V, U-V-W, X-D-I, M-A-R-S, A-U-D-I, P-A-R-I-S, C-O-L-A, M-A-R-Y, P-E-A-R, A-P-R-I-L	93.01	92.20	91.3	88.20	91.29	86.3

segment, duration and bias. The DNNs were first trained in a signer-independent way. The inputs were the image features concatenated over a multi-frame window, which were fed through several fully connected layers followed by a softmax output layer. They considered adaptation by fine-tuning, i.e. updating all of the DNN weights on adaptation data starting from the signer independent weights. They obtained a recognition rate of 82.7% with ground-truth frame labels adaptation data. Shi et al. [117] focused on recognition of fingerspelling sequences in ASL videos collected in the wild, mainly from YouTube and Deaf social media. They proposed an attention-based fingerspelling recognition model, trained end-to-end based on an iterative attention mechanism, without explicit hand detection or segmentation. The attention model was based on a convolutional recurrent architecture. The accuracy obtained was 86.1% for native signers and 83.1% for non-native signers.

TABLE 4.9: Comparison of various continuous fingerspelling recognition methods

Method	Recognition rate
Kim et al. [116]	82.7%
Shi et al. [117]	86.1%
Our proposed model	**91.29%**

The results indicate that our system is quite efficient in modeling both assimilation and dissimilation which occur during fluent fingerspelling. The classification results obtained from the FSM model and the recognition rates achieved using the CRF classifier

establishes the robustness of the feature set and the efficiency of the proposed system in dealing with the sign language sentences containing a range of fingerspelling alphabets and *me*. Also, the error rate obtained in our case is less than the error rate of 11.6% obtained in Ref. [1].

4.4 Conclusion

In this chapter, we report the design of a continuous ASL fingerspelling recognition system for spotting and recognizing fingerspellings encapsulated within a continuous fingerspelling stream involving *me* with an accuracy of over 91.29%. For detecting *me* in local motion, we implement a fusion of three features in our proposed system. This characteristic makes our system quite expedient in comparison to those systems which construct separate *me* models through data collection, training, and labeling of associated *me* segments. Moreover, we develop an adaptive thresholding methodology and an FSM model for effective segmentation of fingerspelling and *me* frames, which are novel contributions of our work. In addition to these, our system also has the ability to work in backgrounds with complex content, multiple signers, and different illuminations.

In terms of accuracy, the uncompressed domain techniques have higher recognition accuracy of signs and movement epentheses in comparison to compressed domain techniques. This is because, the uncompressed domain techniques exploit visual features such as shape, color, texture etc. On the other hand, in compressed domain techniques, very limited information such as motion vectors, DCT coefficients etc. are available. But, in terms of computational time, compressed domain techniques require less computational time than uncompressed domain techniques because the compressed domain techniques deal with a compressed amount of data.

The deficiency of this proposed system is that it has not taken alphabets J and Z into consideration since these involve global motion. In future work, the system can be augmented with some additional features to overcome this constraint. The proposed system can be extended for detecting *me* in case of double-handed signs. Facial features can be used as an adjunct to our system for making it more expressive.

Chapter 5

Continuous Sign Language Spotting using Gaussian Hidden Markov Model

Sign language is the language used by the hearing-impaired community for day-to-day conversations among themselves and with healthy people. An automatic sign language recognition system can play an important role in mitigating the communication gap between the two communities. Continuous sign language recognition system is a challenging task because of the presence of movement epenthesis (me) which is the extra movements of hand(s) during the transition from one sign to the next sign. In this chapter, we report work on continuous sign language spotting, i.e. separation of signs from the sign sequence by removing the me phase from H.264/AVC compressed videos. The work is based on a 2-state Hidden Markov Model (HMM) with Gaussian emission probability. The HMM is fitted with the features extracted from the entire sign-sequence video and finally the state sequence is decoded using the Viterbi algorithm. From the decoded state sequence, the sign spotting is done. The feature set comprises features extracted from compressed domain

as well as uncompressed domain analysis of the sign videos. The ASL video database is collected from Boston University database. From the experimental results, it is seen that the proposed method can separate the sign and me frames with an average spotting rate 0f 80%.

5.1 Introduction

The difficulty of sign language spotting is that instances of signs vary in both motion and appearance [31]. The hidden Markov model (HMM) is useful in dealing with time-varying nature of features [118], [119]. Mohandes et al. [119] proposed a signer-independent Arabic sign language recognition (SLR) system using an HMM. The HMM was used to present the time-varying feature patterns and the accuracy obtained was more than 95%. S. Azar and S. Hadi [118] presented work on dynamic Persian sign language recognition using HMM. They used a region growing technique to extract the motion trajectory of the hand along with three other shape information. Kumar et al. [120] proposed a multi-sensor fusion framework for SLR using Coupled HMM (CHMM) for recognition of dynamic isolated sign gestures and achieved recognition rate as high as 90.80%. In [14], Gao et al. proposed a signer-independent continuous SLR system using the self-organizing feature maps (SOFM) as different signer's feature extractor for continuous HMM and an accuracy of 86.3% was obtained. Yang et al. [15] proposed a continuous SLR system using level building based on HMM. HMM was used to calculate the similarity between the sign model and testing sequence, and a fast algorithm for computing the likelihood of the HMM.

Generally, in model-based approaches for continuous sign language spotting using HMM, the first and third problems of HMM, i.e. evaluation and fitting are used. But in our work, the second problem of the HMM, i.e. inference is extensively used. In this chapter, we propose a continuous sign language spotting framework using a 2-state HMM in a novel way. A fusion of both compressed as well as spatial domain features is also used. The chapter is organized as follows: Section 5.3 presents the proposed work for continuous sign language spotting using Gaussian hidden Markov model, Section 5.4 presents the experimental results obtained and finally Section 5.5 concludes the chapter.

5.2 Hidden-Markov model for isolated sign language recognition

Let each sign be represented by a sequence of vectors of observations O, defined as

$$O = o_1, o_2, ...o_T \tag{5.1}$$

where, o_t is the observed vector observed at time t.

The isolated sign language recognition problem can be regarded as that of computing

$$\arg\max_i \{P(w_i|O)\} \tag{5.2}$$

where, w_i is the i^{th} sign.

This probability is not computable directly but using Baye's Rule gives

$$P(w_i|O) = \frac{P(O|w_i)\,P(w_i)}{P(O)} \tag{5.3}$$

Thus for a given set of prior probabilities $P(w_i)$, the most probable articulated sign depends only on the likelihood $P(O|w_i)$.

The direct estimation of the joint conditional probability $P(o_1, o_2, ...o_T|w_i)$ from the examples of the articulated signs is not practicable. However, if a parametric model of the sign production such as a Markov model is assumed, then estimation from data is possible since the problem of estimating the class conditional observation densities $P(O|w_i)$ is replaced by the much simpler problem of estimating the Markov model parameters.

In HMM-based sign recognition, it is assumed that the sequence of observed vectors corresponding to each sign is generated by a Markov model.

A Markov model is a finite state machine that changes the state once every time unit [121]. Each time t that a state j is entered, an observed vector o_t is generated from

the probability density $b_j(o_t)$ which is given by

$$b_j(o_t) = P\left[o_t \text{ at } t | q_t = j\right], \quad \begin{matrix} 1 \leq j \leq N \\ 1 \leq t \leq T \end{matrix} \tag{5.4}$$

where, q_t denotes the state at time t and N is the total number of states.

Further, the transition from state i to state j is also probabilistic and is governed by the discrete probability $a_{i,j}$ where,

$$a_{ij} = P\left[q_{t+1} = j | q_t = i\right], \quad 1 \leq i, j \leq N \tag{5.5}$$

Given that the state sequence Q is unknown, the required likelihood is computed by summing over all possible state sequences $Q = q_1, q_2, q_3 \ldots\ldots q_T$, that is

$$P(O|\lambda) = \sum_{Q} a_{q_0 q_1} \prod_{t=1}^{T} b_{q_t}(o_t) a_{q_t q_{t+1}} \tag{5.6}$$

where, q_0 is constrained to be the model entry state and q_{T+1} is constrained to be the model exit state.

As an alternative to Eq. 5.6, the likelihood can be approximated by only considering the most likely state sequence, that is

$$\hat{P}(O|\lambda) = \max_{Q} \left\{ a_{q_0 q_1} \prod_{t=1}^{T} b_{q_t}(o_t) a_{q_t q_{t+1}} \right\} \tag{5.7}$$

Given a set of models λ_i corresponding to signs w_i, Eq. 5.2 is solved by using Eq. 5.3 and assuming that

$$P(O|w_i) = P(O|\lambda_i) \tag{5.8}$$

The model parameters of an HMM is given by

$$\lambda = (A, B, \pi) \tag{5.9}$$

where,

the state transition probability distribution, $A = \{a_{i,j}\}$, the observation symbol probability distribution, $B = \{b_j(o_t)\}$ and initial state distribution $\pi = \{\pi_i\}, \pi_i = P(i_1 = i), 1 \leq i \leq N$. The overall process is shown in Fig. 5.1.

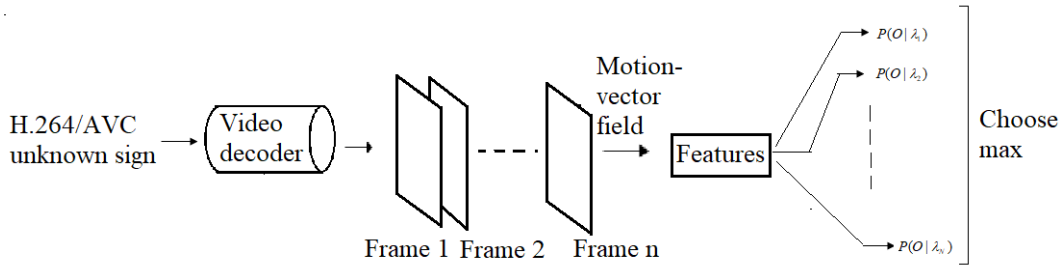


FIGURE 5.1: HMM for isolated sign language recognition.

Three basic problems for HMM

There are three basic problems of interest that must be solved for the model to be useful in sign language classification applications [122], [123], [119]. These problems are:

1. Evaluation: Given the observation sequence $O = o_1, o_2, ..., o_T$ and a model $\lambda = (A, B, \pi)$, how to compute $P(O|\lambda)$, the probability of occurrence of the observation sequence $O = o_1, o_2, ...o_T$.

2. Inference: Given the model $\lambda = (A, B, \pi)$, how to choose a state sequence $Q = q_1, q_2, q_3, ..., q_T$ so that $P(O, I|\lambda)$, the joint probability of the observation sequence $O = o_1, o_2, ...o_T$ and the state sequence given the model is maximized.

3. Fitting (or training): How to adjust the HMM model parameters $\lambda = (A, B, \pi)$ so that $P(O|\lambda)$ (or $P(O, I|\lambda)$) is maximized.

Although for SLR, problems 1 and 3 are given more emphasis, in our work for continuous sign language spotting problem, solution of problem 2 is extensively used.

5.3 Continuous sign language spotting using Gaussian hidden Markov model

Continuous sign language spotting can be considered as a two-class problem i.e., classification of a frame into "sign" or "*me*" frame. For that reason, a 2-state HMM is considered as shown in Fig. 5.2 where, one state is "sign" and the other one is "*me*" state. "Sign" state is represented by "1" and "*me*" state is represented by "0". Firstly, the HMM is trained by the feature set derived from the sequence of signs. Once training of the HMM is done, hidden states of the model are decoded by using the Viterbi algorithm [122] as shown in Fig. 5.3. From the decoded state sequence, spotting of signs, i.e. finding of start and end frames of the signs is done. For a particular sign, the transition from state "0" to state "1" can be considered as "start" frame of that sign and the transition from state "1" to state "0" can be considered as "end" frame of the sign as shown in Fig. 5.3. The intermediate frames corresponding to state sequence "0000....0" can be considered as "non-sign" or "*me*" and the frames corresponding to state sequence "11.....1" can be considered as "sign" as shown in Fig. 5.3.

The overall proposed framework for continuous sign language spotting is shown

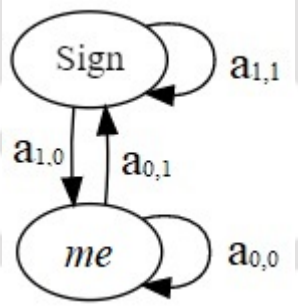


FIGURE 5.2: State transition model.

in Fig. 5.4. Firstly, the input H.264/AVC video is decoded to extract the frames along with the motion vectors (MVs) that are inherent in the video itself. After that, the face of the signer is detected and removed. After removing the face, both the palms of the signer's hands are detected using skin color-based segmentation. After that, a feature set comprised of one temporal feature and three spatial features is derived. This feature set

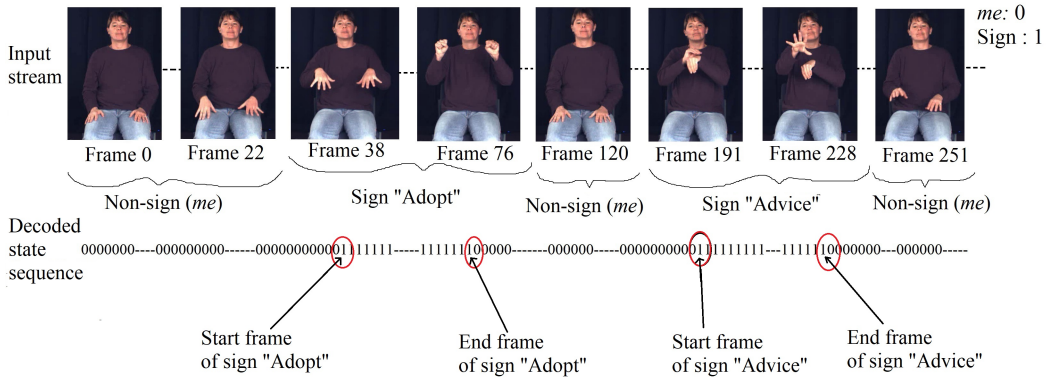


FIGURE 5.3: Decoding of hidden state sequence for continuous sign language spotting.

is used to train the HMM which is finally used for separation of "sign" and "*me*" frames.

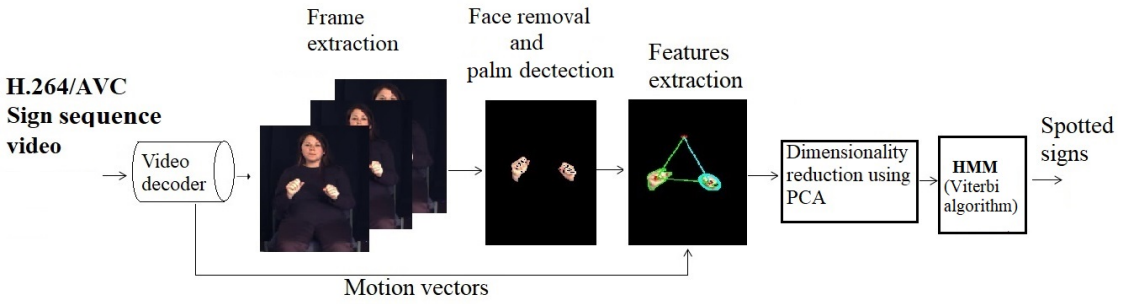


FIGURE 5.4: Proposed HMM-based framework for continuous sign language spotting.

The graphical model of the HMM considered for the sign language sequence for observations is shown in Fig. 5.5 where, $q_t \in \{sign, me\}$, $1 \leq t \leq T$.

The state-transition table for the 2-state HMM is shown in Table 5.1 The trellis constructed for the model is shown in Fig. 5.6.

TABLE 5.1: State transition table

A	**Sign (1)**	***me* (0)**
Sign (1)	$a_{1,1}$	$a_{1,0}$
***me* (0)**	$a_{0,1}$	$a_{0,0}$

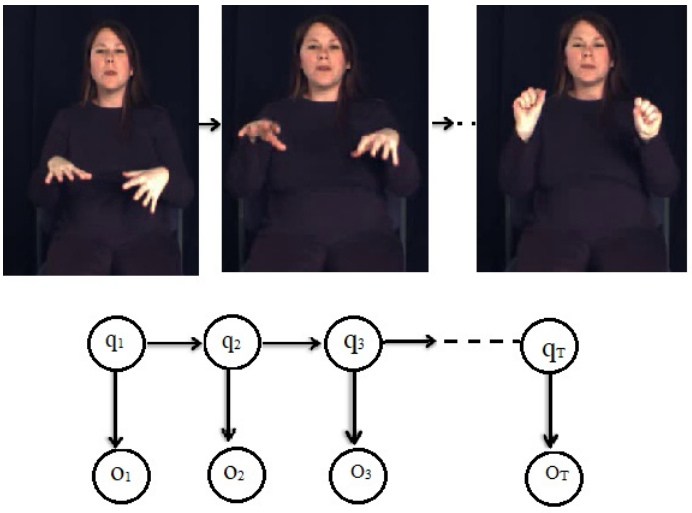


FIGURE 5.5: Graphical model of the HMM for the sign language sequence.

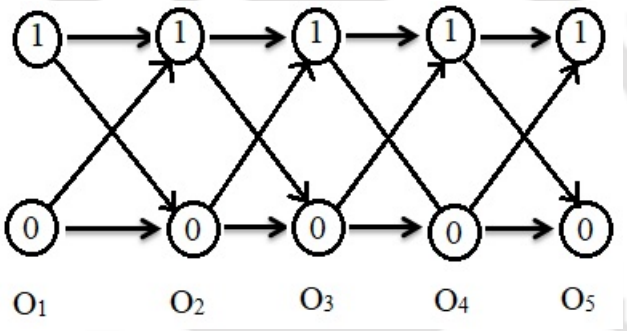


FIGURE 5.6: Trellis diagram for the HMM.

Solutions to the problems of HMM

1. Solutions to evaluation problem: The straightforward way to determine $P(O|\lambda)$ is to find $P(O, I|\lambda)$ for a fixed state sequence $Q = q_1, q_2, q_3 \ldots\ldots q_T$, then multiply it by $P(Q|\lambda)$ and sum-up over all possible Q's as shown in Eq. 5.5. But the problem with this method is that the direct computation of Eq. 5.5 involves the order of $2TN^T$ multiplications. Hence to simplify the computation, a procedure called forward-backward procedure [123] is used.

2. The solution to inference problem: For inference, it is needed to find a state sequence $Q = q_1, q_2, q_3 \ldots\ldots q_T$ that will maximize $P(O, I|\lambda)$. For this, there is a standard algorithm called Viterbi Algorithm [122]. It is a dynamic programming algorithm for finding the most likely sequence of hidden states-called the Viterbi path, for the desired observed sequence as shown in Fig. 5.7.

3. Fitting (or training) of HMM: The third problem, and by far the most difficult problem of HMMs is to determine a method to adjust the model parameters (A, B, π) to maximize the probability of the observation sequence given the model. There is no known way to analytically solve for the model which maximizes the probability of observation sequence. However, $\lambda = (A, B, \pi)$ can be chosen which maximizes $P(O|\lambda)$ locally by using an iterative procedure such as the Baum-Welch method (or equivalently the EM (expectation-maximization) method [123]).

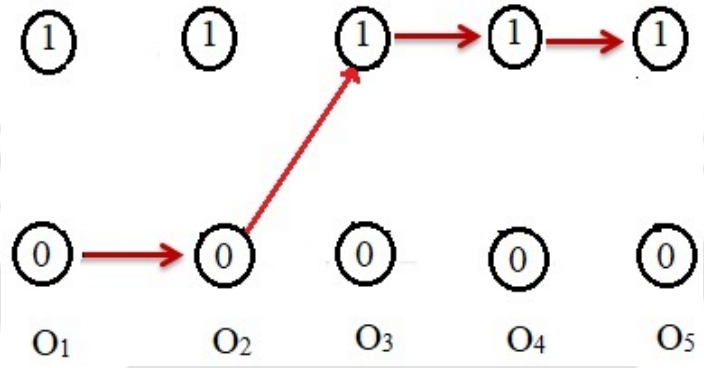


FIGURE 5.7: Viterbi path for finding the optimal state sequence.

The observation probability $b_j(o_t)$ is estimated as a Gaussian pdf with mean vector μ_j and covariance matrix $\sum_j$ in state j which is of the form [122]

$$b_j(O) = \mathbb{N}\left[O, \mu_j, \sum_j\right], \ 1 \leq j \leq N \tag{5.10}$$

where, O is the vector being modeled and $\mathbb{N}$ is Gaussian density and

$$\int_{-\infty}^{\infty} b_j(x)\,dx = 1, \ 1 \leq j \leq N \tag{5.11}$$

5.3.1 Features extraction

A mixture of uncompressed and compressed domain features is used for training the HMMs. The four features used here are:

1. Spatial domain features:

(a) Position of the hands

(b) Orientation of the hands

(c) Shape of the hands

2. Compressed-domain feature:

   (a) Motion vectors of the moving hands

The first three features are extracted from the uncompressed video and the last feature is extracted with the help of the inherent feature (i.e. motion vector) of the compressed video.

Pre-processing of the video

Before finding the above-mentioned features, the following pre-processing of the frames are done:

1. Face detection: The face of the signer is detected and removed using the Haar-cascade classifier [124]. It is a famous object detection technique based on the Adaboost learning algorithm. The result of the face detector is shown in Fig. 5.8.

2. Hand segmentation: After removing the face, hand segmentation is done using the skin color-based segmentation technique. Firstly, the RGB image is converted into the HSV color space. Then, the lower and upper ranges of H, S, and V for skin color are chosen for selecting the lower and upper thresholds. Then using these thresholds, hand segmentation is done. The output of the hand segmentation is shown in Fig. 5.9. The hand segmentation is followed by the dilation operation to fill-up the holes created inside the segmented hands.

After performing the above-mentioned preprocessing steps of the frames, feature extraction is done. The extracted features are discussed one by one.

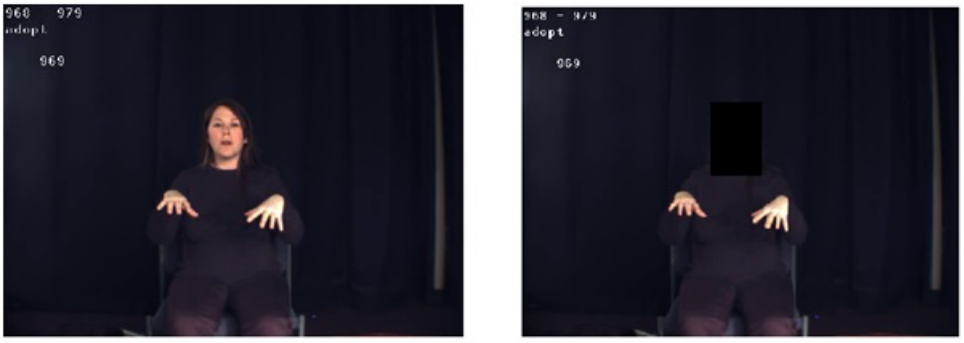


FIGURE 5.8: Face removal output (a) Input frame (b) Face removed.

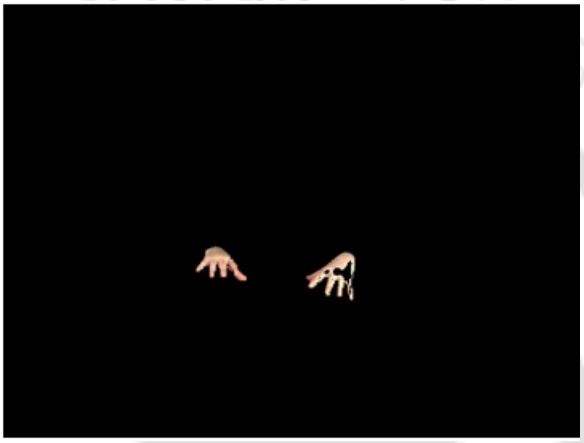


FIGURE 5.9: Hand segmentation output.

Position of the hands

The positions of the left and right hands are found by finding the positions of centroids of both palms w.r.t face as shown in Fig. 5.10. Let (x_r, y_r) and (x_f, y_f) be the coordinates of the right palm and the face respectively. Then the position of the right-hand w.r.t. the face is represented by $P = (d, \theta)$ where,

$$\begin{aligned} d &= \sqrt{(x_f - x_r)^2 + (y_f - y_r)^2} \\ \theta &= \tan^{-1}\left(\frac{y_f - y_r}{x_f - x_r}\right) \end{aligned} \tag{5.12}$$

Similarly, we can find the position of the left hand also.

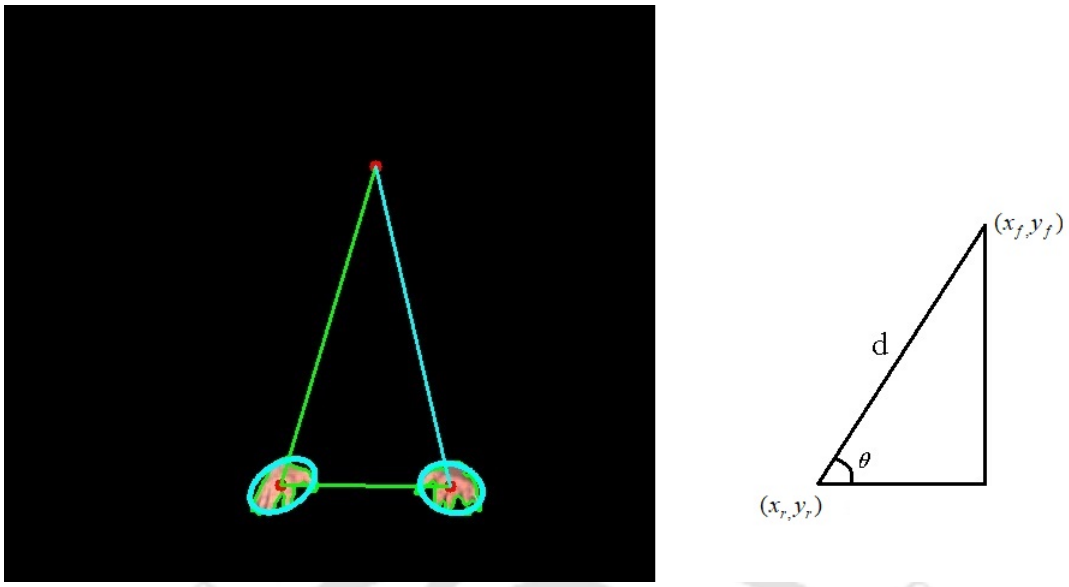


FIGURE 5.10: Finding positions of the hands.

Orientation of the hands

The orientation of a hand is obtained by fitting an ellipse into the hand and finding the orientation of the major axis of the ellipse, as shown in Fig. 5.11 (a). Fig.5.11 (b) shows the orientation of the left hand, θ_1 .

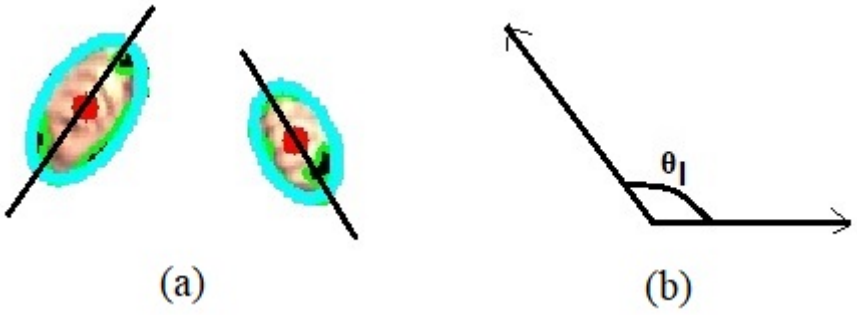


FIGURE 5.11: Finding orientation of hands (a) Fitting ellipse (b) Orientation of the major axis of the ellipse for the left hand.

Shape of the hands

For finding hand shape, the Histogram of Oriented Gradients (HOG) feature is used. This is a robust feature used by Dallal and Triggs [125] for the first time for pedestrian detection. The method is based on evaluating well-normalized local histograms of image

gradient orientation in a dense grid. For finding the HOG feature of a hand, first, the hand portion is cropped based on palm center information as shown in Fig. 5.12.

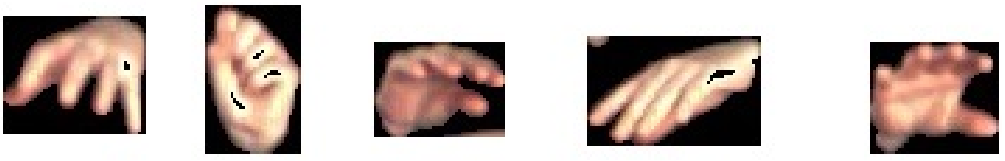


FIGURE 5.12: Some of the cropped hand images for finding HOG features.

Movement of the hands

The movement of a hand is found by using the motion vector (MV) of the palm center of that particular hand. For finding the MV, the compressed domain technique is used [74], [126]. Firstly, the MVs which are inherent in the H.264/AVC compressed video itself are partially extracted from the video as shown in Fig. 2.1. After that, MV components (i.e. $\overrightarrow{\mathrm{mv_x}}$ and $\overrightarrow{\mathrm{mv_y}}$) of each palm centroid are found out as shown in Fig. 5.13.

FIGURE 5.13: Finding movement of the hands (a) Centroids of the hands (b) Motion vector (mv) components for left and right hands.

Training of HMMs

After finding all the above-mentioned features, a feature set is formed for each frame by combining all the four features. The dimensionality of the feature set obtained for a particular frame is 132×1 which is very high. To reduce it, the principal component analysis (PCA) technique is used which is a dimensionality reduction technique of a large

data set [127]. It reduces the dimensionality of the feature set from 132×1 to 42×1 with inclusion of 95% variance which results in the reduction of the computational complexity of the model. Finally, feature sets for all the frames of the video are combined to form the complete feature set. This feature set is used to train the HMM as shown in Fig. 5.14. The overall process is shown in Fig. 5.15.

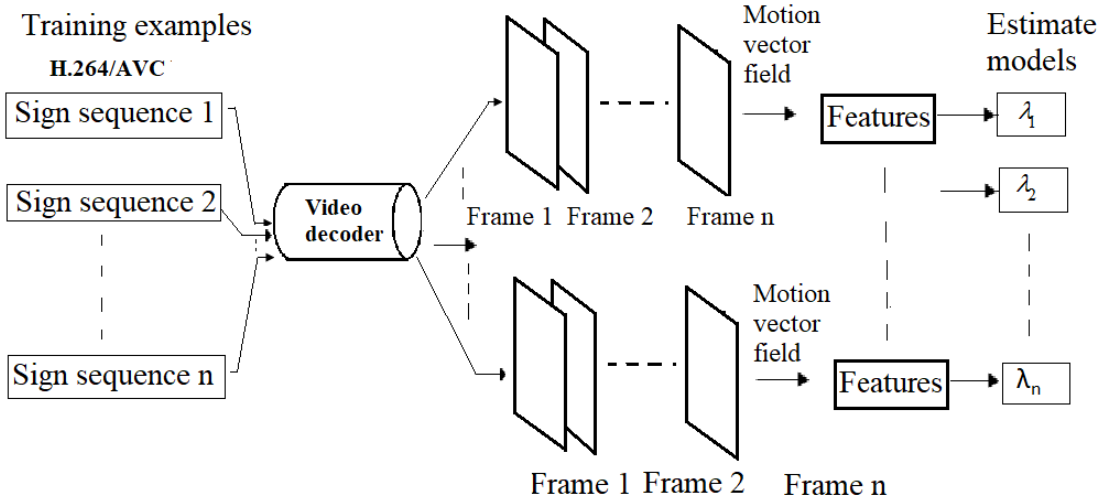


FIGURE 5.14: Training of HMMs.

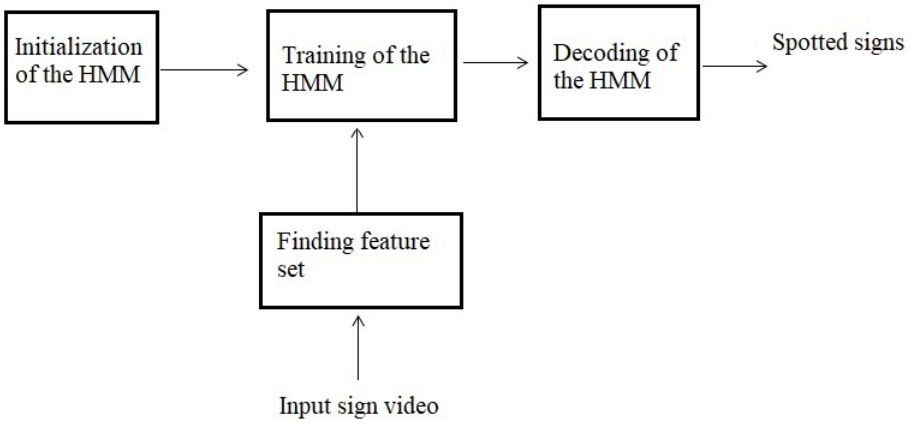


FIGURE 5.15: The overall process of using HMM for continuous sign language spotting.

Initialization of the HMM

For training the HMM, it is to be initialized first. It is observed from the sign sequences (as shown in Fig. 5.3) that throughout the sequence, there are state transitions from

state "0" to state "1" and vice versa only at start frames and at end frames of the signs respectively. So, the state transition probabilities are initialized as $a_{0,0} = a_{1,1} = 0.8$ and $a_{0,1} = a_{1,0} = 0.2$ which are shown in Table 5.2. The initial state probability distribution is considered as $\pi = [0.5, 0.5]$.

TABLE 5.2: Initialization of state transition probabilities

A	**Sign (1)**	***me* (0)**
Sign (1)	0.8	0.2
***me* (0)**	0.2	0.8

5.4 Experimental results

For experimental results, the American Sign Language lexicon video dataset (ASLLVD) [128] prepared at Boston University is used. The dataset consists of thousands of ASL signs, each produced by 1-8 native signers captured from different angles. Some of the signers are shown in Fig. 5.16. We use the dataset captured by camera1 (the front view) with VGA (640×480) resolution recorded at 60 frames per second (fps). All sequences are encoded using H.264/AVC JM V.18.0 encoder with baseline profile and at different bitrates, with GOP structure IPPP..., i.e., the first frame is coded as Intra (I), and the subsequent frames are coded predictively (P) with GOP size 30. Block size of 4×4 is used, as this is the lowest unit possible and is more likely to represent the moving objects.

Some of the intermediate results of the pre-processing steps for continuous ASL sign sequence "Adopt-Advice" performed by native Signer 1 and Signer 2 are shown in Fig. 5.17 and Fig. 5.18 respectively.

5.4.1 Continuous sign language spotting

The results of the sign spotting algorithm for various sign-sequences articulated by different signers are shown in Table 5.3 and Table 5.4.

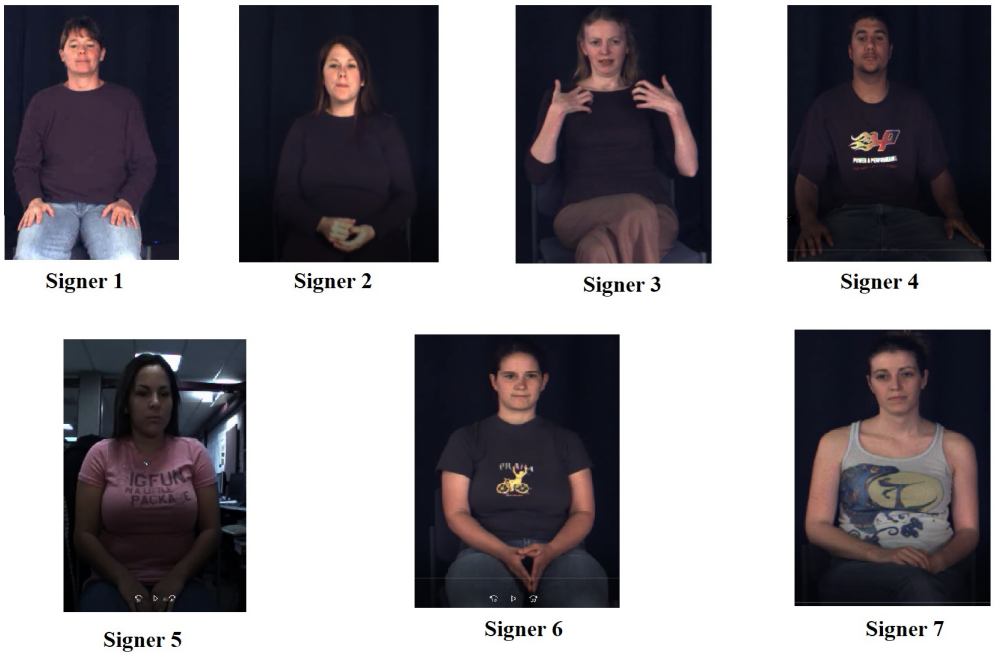


FIGURE 5.16: Some signers of ASLLVD dataset.

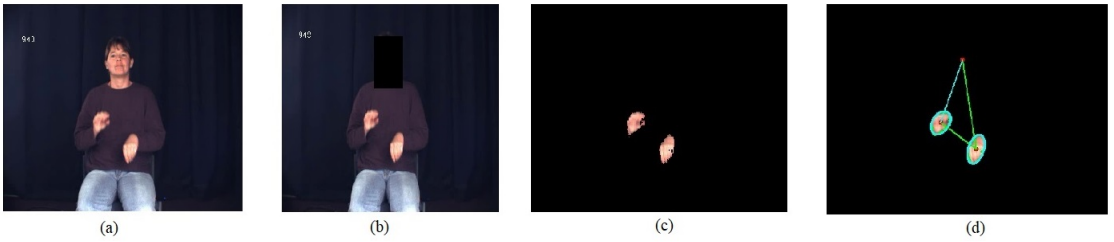


FIGURE 5.17: Intermediate results for "frame # 108" of the sign sequence "Adopt-Advice" performed by Signer 1 (a) Input frame (b) Face removed (c) Hand segmentation output (d) Features extraction.

Quantitative evaluation of the proposed method is carried out by the correct spotting rate, R which is already defined in section 3.5.2.

Table 5.5 shows the sign spotting results for four sign-sequence videos articulated by different signers. From the table, it is seen that for sign-sequence "Camping-Can", both the signs are spotted correctly according to Eq. 3.29. But, in the case of sign-sequence "Adopt-Advice", sign "Adopt" is correctly spotted, but sign "Advice" is not spotted correctly. The overall spotting rate is found to be about 80% which is superior to the proposed method based on Support Vector Machine by Yang and Lee [3].

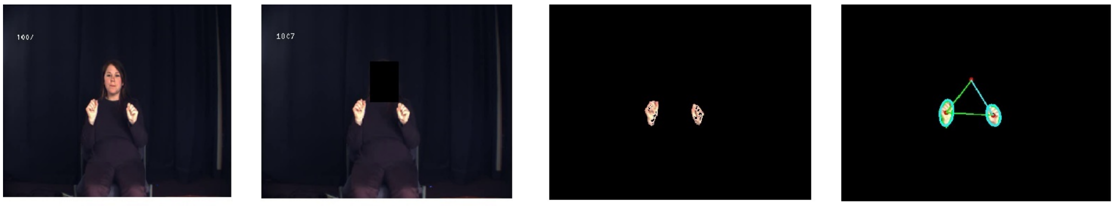


FIGURE 5.18: Intermediate results for "frame # 108" of the sign sequence "Adopt-Advice" performed by Signer 2 (a) Input frame (b) Face removed (c) Hand segmentation output (d) Features extraction.

TABLE 5.3: Continuous sign spotting results for Signer 1

Signer	**Sign sequence**	**State sequence**
Signer 1	Adopt-Advice	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
	Awful-Awkard	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

TABLE 5.4: Continuous sign spotting results for Signer 2

Signer	**Sign sequence**	**State sequence**
Signer 2	Adopt-Advice	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1
	Building-Busy	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1

5.5 Conclusion and future work

In this chapter, we present a continuous sign language spotting framework based on Gaussian-HMM (GHMM). The framework consists of a 2-state HMM with Gaussian emission probability. One state is considered for “sign” and another is for “*me*” state. The HMM is fitted with the features extracted from the whole video sequence and finally

TABLE 5.5: Continuous sign spotting results

Videos	Sign sequence	Number of sign segments	Ground-truth for signs	Spotted sign segments
1	Adopt-Advice	2	Adopt (37-100) Advice (188-248)	Adopt (40-107) Advice (175-252)
2	Camping-Can	2	Camping (10-88) Can (138-162)	Camping (4-91) Can (139-162)
3	Building-Busy	2	Building (28-85) Busy (170-238)	Building (33-90) Busy (168-258)
4	All-Accident	2	All (10-80) Accident (146-171)	All (11-97) Accident (140-170)

the hidden state sequence is decoded using the Viterbi algorithm. From the decoded state-sequence, the sign spotting i.e. separation of sign and *me* frames is done. The feature sets are composed of features extracted from the compressed domain as well as uncompressed domain analysis of the sign videos. The database is composed of an ASL video database collected from Boston University. From the experimental results, it is seen that the proposed method can separate the sign and *me* frames with a spotting rate of about 80%.

The work can be further extended for the spotting of sign sentences containing more than two signs. Once the sign spotting is done, each sign of the sign-sequence video can be recognized by a suitable recognizer.

Chapter 6

Conclusion and Future Work

Sign language spotting is the task of extracting the sign components from a continuous sign language video by removing the non-sign (movement epenthesis) components. There have been many research works reported on continuous sign language spotting in the uncompressed domain for recognition of the constituent signs embedded in the sentence. In this thesis, we report our approaches on continuous sign language spotting in the fully compressed domain, uncompressed domain, and mixture of both domains analyses of the sign videos. A spatio-temporal Markov random field based framework is proposed for spotting of continuous fingerspelling sequence in fully the compressed domain. A novel global motion estimation and compensation framework in the compressed domain is reported for sign language videos. An adaptive thresholding-based framework for continuous fingerspelling recognition in the uncompressed domain is reported. Gaussian hidden Markov model-based approach is reported for spotting continuous sign language videos by using both compressed and uncompressed domain analyses of the videos. We consider American Sign Language as a case study to verify the efficiency of our proposed techniques.

6.1 Conclusion

Sign language recognition (SLR) systems play an important role as an interpreter to bridge the gap between the hearing community and the hearing impaired community. Continuous SLR systems suffer a problem called coarticulation which is the effect of the neighboring signs on the articulation of the present sign. Movement epenthesis (*me*) is a special type of coarticulation which is an additional movement of the hands during the transition from one sign to the next sign and also at the beginning and end of the sentence. Sign language spotting is the task of extracting the meaningful sign components from sign language sentences by removing the movement epentheses present in the video. It is an important and challenging task for efficient recognition of the signs present in a sentence. Different approaches have been proposed so far for sign spottings in continuous sign language. But, most of the approaches have relied on uncompressed analysis of the videos. There are many advantages of analyzing the videos in the compressed domain and so far as per our knowledge, no work has been reported on sign language spotting in the compressed domain. Therefore in this thesis, we focus on sign language spotting mostly in the compressed domain. For that, we propose movement epenthesis detection frameworks for continuous fingerspelling as well as for continuous sign sentences.

In **Chapter 1**, we reviewed different types of coarticulations present in continuous sign languages along with movement epenthesis. We also reviewed various works reported for continuous sign language spotting, their advantages, and drawbacks in **Chapter 2**. To address some of the drawbacks, we proposed sign language spotting in the compressed domain. In **Chapter 3**, we proposed our framework for continuous fingerspelling spotting in the compressed domain. In continuous fingerspelling, the movements of the fingers are not rigid and there are temporal dependencies among their movements. So to exploit these properties, a spatio-temporal Markov random field-based framework was presented for efficient removal of outliers present and spotting of the fingerspelling sequences. We also considered the aspect of motion components that might arise due to any camera motion present during the capturing of the videos. For that, a novel global motion estimation and global motion compensation framework was considered. In **Chapter 4**, we addressed the design of a continuous fingerspelling spotting and recognition system

in the uncompressed domain. Adaptive thresholding and Finite State Machine (FSM) models were also incorporated into our system for efficient classification of fingerspelling and *me* frames. A conditional random field based classifier was used for recognition of the spotted fingerspelled letters. In **Chapter 5**, a Hidden Markov Model (HMM) based framework was presented for continuous sign language spotting. A 2-state HMM with Gaussian emission probability was considered where one state was for "sign" phase and another was for "*me*" phase. We trained our model by various sign sequences and finally, the hidden states were decoded using the Viterbi algorithm. The features considered were extracted using both compressed as well as uncompressed domain analyses of the videos.

The following Table 6.1 shows the comparisons of the results obtained in the compressed domain, the uncompressed domain and the mixture of the two.

TABLE 6.1: Comparison of the results in different domains

SI no.	Work done	Domain	Accuracy
1	Continuous fingerspelling spottings	Compressed domain	75%
2	Continuous fingerspelling spottings	Uncompressed domain	91.29%
3	Continuous sign language spottings	A mixture of compressed and uncompressed domains	80%

6.2 Future works

Although sign spotting in the compressed domain has many advantages over the uncompressed domain analysis, still it is very challenging as we have only limited information in hand. We used only the MVs and MBs inherent in the video, but other compressed domain information such as DCT coefficients, Group of pictures (GOPs), etc. maybe considered along with MVs and MBs to increase the spotting rate.

The proposed Gaussian HMM-based framework can be extended for spotting of sign sentences containing more than two signs.

Nowadays, deep learning-based frameworks are becoming very popular in the field of sign language recognition. Deep Learning techniques may be employed with the proposed frameworks for better spotting.

List of Publications

International Journals

- A. K. Talukdar and M.K. Bhuyan, "**Continuous Sign Language Spotting using Gaussian Hidden Markov Model**," IEEE Sensors Letters, 2022, doi: 10.1109/LSENS.2022.3185181, (IF = 3.6).
- A. K. Talukdar and M.K. Bhuyan, "**A framework for continuous finger-spelling spotting for H.264/AVC compressed videos using spatio-temporal Markov random field**," Multimedia Tools and Applications, Springer, pp. 1-19, 2021, (IF=2.757).
- A. Choudhury, A. K. Talukdar, K.K. Sarma and M.K. Bhuyan, "**An Adaptive Thresholding-Based Movement Epenthesis Detection Technique Using Hybrid Feature Set for Continuous Fingerspelling Recognition**," SN Computer Science, Springer, vol.2, no. 2, pp. 1-21, 2021.

International Conferences

- A. K. Talukdar and M.K. Bhuyan, "**Movement Epenthesis Detection in Continuous Fingerspelling from a Coarsely Sampled Motion Vector Field in H.264/AVC Video**," in Proceedings of IEEE Recent Advances in Intelligent Computational Systems (RAICS), 2018, pp. 26-30.
- A. K. Talukdar and M.K. Bhuyan, "**A Novel Global Motion Estimation and Compensation Framework in Compressed Domain for Sign Language Videos**," in Proceedings of IEEE International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), 2020, pp. 20-24.
- M.K. Bhuyan, A. K. Talukdar, P. Gupta and R. H. Laskar, "**Low Cost Data Glove for Hand Gesture Recognition by Finger Bend Measurement**," in

Proceedings of IEEE International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), 2020, pp. 25-31.

Bibliography

[1] T. Kim, G. Shakhnarovich, and K. Livescu, "Fingerspelling recognition with semi-Markov conditional random fields," in *Proc. IEEE Int. Conf. Comput. Vision*, 2013, pp. 1521–1528.

[2] R. Yang, S. Sarkar, and B. Loeding, "Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 462–477, 2010.

[3] H.-D. Yang and S.-W. Lee, "Simultaneous spotting of signs and fingerspellings based on hierarchical conditional random fields and boostmap embeddings," *Pattern Recognit.*, vol. 43, no. 8, pp. 2858–2870, 2010.

[4] E. Ormel, O. Crasborn, and E. van der Kooij, "Coarticulation of hand height in sign language of the Netherlands is affected by contact type," *J.Phonetics*, vol. 41, no. 3-4, pp. 156–171, 2013.

[5] J. Keane, D. Brentari, and J. Riggle, "Coarticulation in ASL fingerspelling," in *Proc. North East Linguistic Soc.*, vol. 42, 2012.

[6] E. Ormel, O. Crasborn, G. J. Kootstra, and A. d. Meijer, "Coarticulation of handshape in sign language of the Netherlands: A corpus study," *Lab. Phonology*, vol. 8, no. 1, pp. 1–21, 2017.

[7] C. E. Mauk, B. Lindblom, and R. P. Meier, "Undershoot of ASL locations in fast signing," *Signs Time. Selected Papers TISLR*, vol. 8, pp. 3–24, 2008.

[8] T. E. Jerde, J. F. Soechting, and M. Flanders, "Coarticulation in fluent fingerspelling," *J. Neuroscience*, vol. 23, no. 6, pp. 2383–2393, 2003.

[9] J. Mukhopadhyay, *Image and video processing in the compressed domain.* CRC Press, 2011.

[10] L. Hanzo, P. Cherriman, and J. Streit, *Video Compression and Communications: From Basics to H.261, H.263, H.264, MPEG4 for DVB and HSDPA-Style Adaptive Turbo-Transceivers.* John Wiley & Sons, 2007.

[11] M. Müller, *Information Retrieval for Music and Motion.* Springer, 2007.

[12] H.-K. Lee and J.-H. Kim, "An HMM-based threshold model approach for gesture recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 10, pp. 961–973, 1999.

[13] J. Wang, Y. Ma, Z. Huang, R. Xue, and R. Zhao, "Performance analysis and enhancement of deep convolutional neural network," *Business Inf. Syst. Eng.*, vol. 61, no. 3, pp. 311–326, 2019.

[14] W. Gao, G. Fang, D. Zhao, and Y. Chen, "A Chinese sign language recognition system based on SOFM/SRN/HMM," *Pattern Recognit.*, vol. 37, no. 12, pp. 2389–2402, 2004.

[15] W. Yang, J. Tao, and Z. Ye, "Continuous sign language recognition using level building based on fast hidden Markov model," *Pattern Recognit. Lett.*, vol. 78, pp. 28–35, 2016.

[16] G. Fang, W. Gao, X. Chen, C. Wang, and J. Ma, "Signer-independent continuous sign language recognition based on SRN/HMM," in *Int. Gesture Workshop*. Springer, 2001, pp. 76–85.

[17] A. Choudhury, A. K. Talukdar, K. K. Sarma, and M. K. Bhuyan, "An adaptive thresholding-based movement epenthesis detection technique using hybrid feature set for continuous fingerspelling recognition," *SN Comput. Sci.*, vol. 2, no. 2, pp. 1–21, 2021.

[18] D. A. Cheek, *The Phonetics and Phonology of Handshape in American Sign Language*. The University of Texas at Austin, 2001.

[19] R. Yang, S. Sarkar, and B. Loeding, "Enhanced level building algorithm for the movement epenthesis problem in sign language recognition," in *2007 IEEE Conf. Comput. Vision Pattern Recognit.*, pp. 1–8.

[20] I. B. Kayaalp, "Video segmentation using partially decoded MPEG bitstream," Ph.D. dissertation, Middle East Technical University, 2003.

[21] M. Ghanbari, *Standard Codecs: Image Compression to Advanced Video Coding*. IET, 2003, no. 49.

[22] C. Oz and M. C. Leu, "Linguistic properties based on American Sign Language isolated word recognition with artificial neural networks using a sensory glove and motion tracker," *Neurocomputing*, vol. 70, no. 16-18, pp. 2891–2901, 2007.

[23] M. K. Bhuyan, D. A. Kumar, K. F. MacDorman, and Y. Iwahori, "A novel set of features for continuous hand gesture recognition," *J. Multimodal User Interfaces*, vol. 8, no. 4, pp. 333–343, 2014.

[24] R. Yang and S. Sarkar, "Detecting coarticulation in sign language using conditional random fields," in *18th Int. Conf. Pattern Recognit. (ICPR'06)*, vol. 2. IEEE, 2006, pp. 108–112.

[25] A. Choudhury, A. K. Talukdar, M. K. Bhuyan, and K. K. Sarma, "Movement epenthesis detection for continuous sign language recognition," *J. Intell. Syst.*, vol. 26, no. 3, pp. 471–481, 2017.

[26] R. Yang, S. Sarkar, and B. Loeding, "Handling movement epenthesis and hand segmentation ambiguities in continuous sign language recognition using nested dynamic programming," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 462–477, 2010.

[27] E.-J. Ong, O. Koller, N. Pugeault, and R. Bowden, "Sign spotting using hierarchical sequential patterns with temporal intervals," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2014, pp. 1923–1930.

[28] R. Elakkiya and K. Selvamani, "Enhanced dynamic programming approach for subunit modelling to handle segmentation and recognition ambiguities in sign language," *J. Parallel Distributed Comput.*, vol. 117, pp. 246–255, 2018.

[29] J. Tang, H. Cheng, Y. Zhao, and H. Guo, "Structured dynamic time warping for continuous hand trajectory gesture recognition," *Pattern Recognit.*, vol. 80, pp. 21–31, 2018.

[30] M. Zadghorban and M. Nahvi, "An algorithm on sign words extraction and recognition of continuous persian sign language based on motion and shape features of hands," *Pattern Anal. Applicat.*, vol. 21, no. 2, pp. 323–335, 2018.

[31] H.-D. Yang, S. Sclaroff, and S.-W. Lee, "Sign language spotting with a threshold model based on conditional random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 7, pp. 1264–1277, 2009.

[32] D. Kelly, J. McDonald, and C. Markham, "Recognizing spatiotemporal gestures and movement epenthesis in sign language," in *2009 13th Int. Mach. Vision Image Process. Conf.* IEEE, pp. 145–150.

[33] H.-D. Yang and S.-W. Lee, "Robust sign language recognition with hierarchical conditional random fields," in *2010 20th Int. Conf. Pattern Recognit.* IEEE, pp. 2202–2205.

[34] S. Ricco and C. Tomasi, "Fingerspelling recognition through classification of letter-to-letter transitions," in *Asian Conf. Comput. Vision.* Springer, 2009, pp. 214–225.

[35] P. Kumar, R. Saini, S. K. Behera, D. P. Dogra, and P. P. Roy, "Real-time recognition of sign language gestures and air-writing using leap motion," in *5th IAPR Int. Conf. Mach. Vision Appl. (MVA).* IEEE, 2017, pp. 157–160.

[36] H. Wang, X. Chai, and X. Chen, "A novel sign language recognition framework using hierarchical grassmann covariance matrix," *IEEE Trans. Multimedia*, vol. 21, no. 11, pp. 2806–2814, 2019.

[37] M. Jebali, A. Dakhli, and M. Jemni, "Vision-based continuous sign language recognition using multimodal sensor fusion," *Evol. Syst.*, vol. 12, no. 4, pp. 1031–1044, 2021.

[38] M. Hassan, K. Assaleh, and T. Shanableh, "Multiple proposals for continuous arabic sign language recognition," *Sensing Imaging*, vol. 20, no. 1, pp. 1–23, 2019.

[39] R. Cui, H. Liu, and C. Zhang, "A deep neural framework for continuous sign language recognition by iterative training," *IEEE Trans. Multimedia*, vol. 21, no. 7, pp. 1880–1891, 2019.

[40] B. Shi, A. M. Del Rio, J. Keane, J. Michaux, D. Brentari, G. Shakhnarovich, and K. Livescu, "American sign language fingerspelling recognition in the wild," in *2018 IEEE Spoken Lang. Techn. Workshop (SLT)*, pp. 145–152.

[41] M. A. Jalal, R. Chen, R. K. Moore, and L. Mihaylova, "American sign language posture understanding with deep neural networks," in *Proc. 2018 21st Int. Conf. Inf. Fusion (FUSION)*. IEEE, pp. 573–579.

[42] K. Papadimitriou and G. Potamianos, "Fingerspelled alphabet sign recognition in upper-body videos," in *Proc. 2019 27th Eur. Signal Process. Conf. (EUSIPCO)*. IEEE, pp. 1–5.

[43] H. B. Nguyen and H. N. Do, "Deep learning for American sign language fingerspelling recognition system," in *Proc. 2019 26th Int. Conf. Telecommun. (ICT)*. IEEE, pp. 314–318.

[44] K. Suri and R. Gupta, "Continuous sign language recognition from wearable imus using deep capsule networks and game theory," *Comput. Elect. Eng.*, vol. 78, pp. 493–503, 2019.

[45] R. Elakkiya, P. Vijayakumar, and N. Kumar, "An optimized generative adversarial network based continuous sign language classification," *Expert Syst. Applicat.*, vol. 182, p. 115276, 2021.

[46] K. L. Cheng, Z. Yang, Q. Chen, and Y.-W. Tai, "Fully convolutional networks for continuous sign language recognition," in *European Conf. Comput. Vision*. Springer, 2020, pp. 697–714.

[47] O. Koller, S. Zargaran, H. Ney, and R. Bowden, "Deep sign: Enabling robust statistical continuous sign language recognition via hybrid cnn-hmms," *Int. J. Comput. Vision*, vol. 126, no. 12, pp. 1311–1325, 2018.

[48] T. Jiang, N. C. Camgöz, and R. Bowden, "Looking for the signs: Identifying isolated sign instances in continuous video footage," in *2021 16th IEEE Int. Conf. Automat. Face Gesture Recognit. (FG 2021)*. IEEE, pp. 1–8.

[49] S. Sharma, R. Gupta, and A. Kumar, "Continuous sign language recognition using isolated signs data and deep transfer learning," *J. Ambient Intell. Humanized Comput.*, pp. 1–12, 2021.

[50] I. Koulierakis, G. Siolas, E. Efthimiou, S.-E. Fotinea, and A.-G. Stafylopatis, "Sign boundary and hand articulation feature recognition in sign language videos," *Mach. Translation*, vol. 35, no. 3, pp. 323–343, 2021.

[51] Z. Niu and B. Mak, "Stochastic fine-grained labeling of multi-state sign glosses for continuous sign language recognition," in *European Conf. Comput. Vision*. Springer, 2020, pp. 172–186.

[52] W. Aly, S. Aly, and S. Almotairi, "User-independent American sign language alphabet recognition based on depth image and PCANet features," *IEEE Access*, vol. 7, pp. 123 138–123 150, 2019.

[53] B. Kang, S. Tripathi, and T. Q. Nguyen, "Real-time sign language fingerspelling recognition using convolutional neural networks from depth map," in *Proc. 2015 3rd IAPR Asian Conf. Pattern Recognit. (ACPR)*. IEEE, pp. 136–140.

[54] N. Tazhigaliyeva, N. Kalidolda, A. Imashev, S. Islam, K. Aitpayev, G. I. Parisi, and A. Sandygulova, "Cyrillic manual alphabet recognition in RGB and RGB-D data for sign language interpreting robotic system (SLIRS)," in *Proc. 2017 IEEE Int. Conf. Robotics Automat. (ICRA)*, pp. 4531–4536.

[55] D. A. Kumar, A. Sastry, P. Kishore, and E. K. Kumar, "Indian sign language recognition using graph matching on 3d motion captured signs," *Multimedia Tools Applicat.*, vol. 77, no. 24, pp. 32 063–32 091, 2018.

[56] M. J. Cheok, Z. Omar, and M. H. Jaward, "A review of hand gesture and sign language recognition techniques," *Int. J. Mach. Learning Cybern.*, vol. 10, no. 1, pp. 131–153, 2019.

[57] Y.-M. Chen, I. V. Bajic, and P. Saeedi, "Moving region segmentation from compressed video using global motion estimation and Markov random fields," *IEEE Trans. Multimedia*, vol. 13, no. 3, pp. 421–431, 2011.

[58] M. Okade and P. K. Biswas, "A novel moving object segmentation framework utilizing camera motion recognition for H.264 compressed videos," *J. Vis. Commun. Image Representation*, vol. 36, pp. 199–212, 2016.

[59] J.-G. Kim, H. S. Chang, J. Kim, and H.-M. Kim, "Efficient camera motion characterization for MPEG video indexing," in *Proc. 2000 IEEE Int. Conf. Multimedia Expo. ICME2000. Latest Advances Fast Changing World of Multimedia (Cat. No. 00TH8532)*, vol. 2, pp. 1171–1174.

[60] K. Tao, S. Lin, and Y. Zhang, "Compressed domain motion analysis for video semantic events detection," in *2009 WASE Int. Conf. Inform. Eng.*, vol. 1. IEEE, pp. 201–204.

[61] J. Meng, Y. Juan, and S.-F. Chang, "Scene change detection in an MPEG-compressed video sequence," in *Proc. Digit. Video Compression: Algorithms and Technologies*, vol. 2419. International Society for Optics and Photonics, 1995, pp. 14–25.

[62] M. Okade, G. Patel, and P. K. Biswas, "Robust learning-based camera motion characterization scheme with applications to video stabilization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 3, pp. 453–466, 2015.

[63] R. V. Babu, B. Rangarajan, S. Sundaram, and M. Tom, "Human action recognition in H.264/AVC compressed domain using meta-cognitive radial basis function network," *Appl. Soft Comput.*, vol. 36, pp. 218–227, 2015.

[64] M. V. Lamari, M. S. Bhuiyan, and A. Iwata, "Hand alphabet recognition using morphological PCA and neural networks," in *Proc. IJCNN'99. Int. Joint Conf. Neural Netw. (Cat. No. 99CH36339)*, vol. 4. IEEE, 1999, pp. 2839–2844.

[65] C. Neustaedter, "An evaluation of optical flow using Lucas and Kanade's algorithm," *University of Calgary Department of Computer Science, Calgary, AB, T2N 1N4, Canada*, 2002.

[66] C.-H. Chuan, E. Regina, and C. Guardino, "American sign language recognition using leap motion sensor," in *Proc. 2014 13th Int. Conf. Mach. Learn. Appl.* IEEE, pp. 541–544.

[67] L. Kane and P. Khanna, "A framework for live and cross platform fingerspelling recognition using modified shape matrix variants on depth silhouettes," *Comput. Vision Image Understanding*, vol. 141, pp. 138–151, 2015.

[68] D. Avola, M. Bernardi, L. Cinque, G. L. Foresti, and C. Massaroni, "Exploiting recurrent neural networks and leap motion controller for the recognition of sign language and semaphoric hand gestures," *IEEE Tran. Multimedia*, vol. 21, no. 1, pp. 234–245, 2019.

[69] F. M. Ciaramello and S. S. Hemami, "A computational intelligibility model for assessment and compression of American sign language video," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3014–3027, 2011.

[70] J. Chon, N. Cherniavsky, E. A. Riskin, and R. E. Ladner, "Enabling access through real-time sign language communication over cell phones," in *Proc. 2009 43rd Asilomar Conf. Signals, Syst. Comput.* IEEE, pp. 588–592.

[71] J. Chon, S. Whittle, E. A. Riskin, and R. E. Ladner, "Improving compressed video sign language conversations in the presence of data loss," in *Proc. 2011 Data Compression Conf.* IEEE, pp. 383–392.

[72] A. K. Talukdar and M. K. Bhuyan, "A framework for continuous fingerspelling spotting for H. 264/AVC compressed videos using spatio-temporal Markov random field," *Multimedia Tools Appl.*, pp. 1–19, 2021.

[73] Y.-M. Chen and I. V. Bajic, "A joint approach to global motion estimation and motion segmentation from a coarsely sampled motion vector field," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 9, pp. 1316–1328, 2011.

[74] A. K. Talukdar and M. K. Bhuyan, "Movement epenthesis detection in continuous fingerspelling from a coarsely sampled motion vector field in H. 264/AVC video," in *Proc. 2018 IEEE Recent Advances Intell. Comput. Syst. (RAICS)*, pp. 26–30.

[75] "The H.264/AVC JM Reference Software." accessed Aug., 2012. [Online]. Available: http://iphome.hhi.de/suehring/tml/

[76] Y. Su and M.-T. Sun, "A non-iterative motion vector based global motion estimation algorithm," in *Proc. 2004 IEEE Int. Conf. Multimedia Expo (ICME)(IEEE Cat. No. 04TH8763)*, vol. 1, pp. 703–706.

[77] Y. Su, M.-T. Sun, and V. Hsu, "Global motion estimation from coarsely sampled motion vector field and the applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 2, pp. 232–242, 2005.

[78] A. Smolic, M. Hoeynck, and J.-R. Ohm, "Low-complexity global motion estimation from P-frame motion vectors for MPEG-7 applications," in *Proc. 2000 Int. Conf. Image Process. (Cat. No. 00CH37101)*, vol. 2. IEEE, pp. 271–274.

[79] S. H. Khatoonabadi and I. V. Bajic, "Video object tracking in the compressed domain using spatio-temporal Markov random fields," *IEEE Trans. Image Process.*, vol. 22, no. 1, pp. 300–313, 2013.

[80] W. T. Vetterling, W. H. Press, S. A. Teukolsky, and B. P. Flannery, *Numerical Recipes Example Book (C++): The Art of Scientific Computing.* Cambridge University Press, 2002.

[81] S. Z. Li, *Markov Random Field Modeling in Image Analysis.* Springer Science & Business Media, 2009.

[82] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, pp. 192–225, 1974.

[83] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 48, no. 3, pp. 259–279, 1986.

[84] H.-D. Yang, S. Sclaroff, and S.-W. Lee, "Sign language spotting with a threshold model based on conditional random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 7, pp. 1264–1277, 2008.

[85] F. Karray, M. Alemzadeh, J. A. Saleh, and M. N. Arab, "Human-computer interaction: Overview on state of the art," *Int. J. Smart Sens. Intell. Syst.*, vol. 1, no. 1, pp. 137–159, 2008.

[86] A. Choudhury, A. K. Talukdar, and K. K. Sarma, "A review on vision-based hand gesture recognition and applications," in *Intelligent Applications for Heterogeneous System Modeling and Design*. IGI Global, 2015, pp. 256–281.

[87] A. Choudhury, A. K. Talukdar, and K. K. Sarma, "A conditional random field based indian sign language recognition system under complex background," in *2014 4th Int. Conf. Commun. Syst. Netw. Technol.* IEEE, pp. 900–904.

[88] Q. Chen, N. D. Georganas, and E. M. Petriu, "Hand gesture recognition using haar-like features and a stochastic context-free grammar," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 8, pp. 1562–1571, 2008.

[89] S. L. Phung, A. Bouzerdoum, and D. Chai, "Skin segmentation using color pixel classification: analysis and comparison," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 1, pp. 148–154, 2005.

[90] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 2008.

[91] R. Gonzalez and R. Woods, *Processing Digital Image*. Pearson Education India, 2009.

[92] P. Goh and E.-J. Holden, "Dynamic fingerspelling recognition using geometric and motion features," in *2006 Int. Conf. Image Process.* IEEE, pp. 2741–2744.

[93] M. R. Teague, "Image analysis via the general theory of moments," *J. Opt. Soc. Am.*, vol. 70, no. 8, pp. 920–930, 1980.

[94] H. Hse and A. R. Newton, "Sketched symbol recognition using Zernike moments," in *Proc. 17th Int. Conf. Pattern Recognit. 2004. (ICPR 2004).*, vol. 1. IEEE, pp. 367–370.

[95] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 5, pp. 489–497, 1990.

[96] D. Zhang and G. Lu, "Review of shape representation and description techniques," *Pattern Recognit.*, vol. 37, no. 1, pp. 1–19, 2004.

[97] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, vol. 2, 1981, pp. 674–679.

[98] J. N. Wilson and G. X. Ritter, *Handbook of Computer Vision Algorithms in Image Algebra*. CRC press, 2000.

[99] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*. Tata McGraw-Hill Education, 2002.

[100] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, 2009.

[101] A. A. Kumar, *Fundamentals of Digital Circuits*. PHI Learning Pvt. Ltd., 2016.

[102] Y. E. Tetık, "Finite state machine based binary classifier," in *2017 25th Signal Process. Commun. Applicat. Conf. (SIU)*. IEEE, pp. 1–4.

[103] E. A. Cudney and S. M. Corns, "A comparison of finite state classifier and mahalanobis-taguchi system for multivariate pattern recognition in skin cancer detection," in *2011 IEEE Symp. Comput. Intell. Bioinformatics Comput. Biology (CIBCB)*. IEEE, pp. 1–7.

[104] S. R. Yadav and S. M. Corns, "Improved pcr design for mouse dna by training finite state machines," in *2010 IEEE Symp. Comput. Intell. Bioinformatics Comput. Biology*. IEEE, pp. 1–5.

[105] M. K. Bhuyan, D. Ghosh, and P. K. Bora, "Key video object plane selection by mpeg-7 visual shape descriptor for summarization and recognition of hand gestures." in *ICVGIP*, vol. 2004. Citeseer, 2004, pp. 638–643.

[106] K. Li and J. C. Principe, "Flight dynamics modeling and recognition using finite state machine for automatic insect recognition," in *2017 Int. Joint Conf. Neural Networks (IJCNN)*. IEEE, pp. 3733–3740.

[107] J. Li, X. Xu, J. Tao, L. Ding, H. Gao, and Z. Deng, "Interact with robot: An efficient approach based on finite state machine and mouse gesture recognition," in *2016 9th Int. Conf. Human Syst. Interactions (HSI)*. IEEE, pp. 203–208.

[108] D. Nirmalee and L. Ranathunga, "Reader text highlighter based on gaze tracking and finite state machine," in *2018 18th Int. Conf. Advances ICT Emerging Regions (ICTer)*. IEEE, pp. 168–173.

[109] J. Kim and D. Kim, "Accurate abandoned and removed object classification using hierarchical finite state machine," *Image Vision Comput.*, vol. 44, pp. 1–14, 2015.

[110] D. O. Sales, D. O. Correa, L. C. Fernandes, D. F. Wolf, and F. S. Osório, "Adaptive finite state machine based visual autonomous navigation system," *Eng. Applicat. Artificial Intell.*, vol. 29, pp. 152–162, 2014.

[111] A. Fernández-Isabel, P. Peixoto, I. M. de Diego, C. Conde, and E. Cabello, "Combining dynamic finite state machines and text-based similarities to represent human behavior," *Eng. Applicat. Artificial Intell.*, vol. 85, pp. 504–516, 2019.

[112] J. Calvo-Zaragoza and J. Oncina, "Recognition of pen-based music notation with finite-state machines," *Expert Syst. Applicat.*, vol. 72, pp. 395–406, 2017.

[113] D. Chetverikov, "A simple and efficient algorithm for detection of high curvature points in planar curves," in *Proc. Int. Conf. Comput. Anal. Images Patterns*. Springer, 2003, pp. 746–753.

[114] C. Dalitz, C. Brandt, S. Goebbels, and D. Kolanus, "Fourier descriptors for broken shapes," *EURASIP J. Advances Signal Process.*, vol. 2013, no. 1, pp. 1–11, 2013.

[115] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, 2002.

[116] T. Kim, W. Wang, H. Tang, and K. Livescu, "Signer-independent fingerspelling recognition with deep neural network adaptation," in *2016 IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*. IEEE, pp. 6160–6164.

[117] B. Shi, A. M. D. Rio, J. Keane, D. Brentari, G. Shakhnarovich, and K. Livescu, "Fingerspelling recognition in the wild with iterative visual attention," in *Proc. IEEE/CVF Int. Conf. Comput. Vision*, 2019, pp. 5400–5409.

[118] S. G. Azar and H. Seyedarabi, "Trajectory-based recognition of dynamic Persian sign language using hidden Markov model," *Comput. Speech Lang.*, vol. 61, pp. 1–20, 2020.

[119] M. Mohandes, M. Deriche, U. Johar, and S. Ilyas, "A signer-independent Arabic sign language recognition system using face detection, geometric features, and a hidden Markov model," *Comput. Elect. Eng.*, vol. 38, no. 2, pp. 422–433, 2012.

[120] P. Kumar, H. Gauba, P. P. Roy, and D. P. Dogra, "Coupled HMM-based multi-sensor data fusion for sign language recognition," *Pattern Recognit. Lett.*, vol. 86, pp. 1–8, 2017.

[121] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey *et al.*, *The HTK Book*. Cambridge University Engineering Department, 2002.

[122] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[123] R. Dugad and U. B. Desai, "A tutorial on Hidden Markov Models," Signal Processing and Artificial Neural Networks Laboratory Department of Electrical Engineering Indian Institute of Technology — Bombay Powai, Bombay 400 076, India, Tech. Rep., 1996.

[124] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. 2001 IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR 2001)*, vol. 1, pp. 1–9.

[125] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR 2005)*, vol. 1, pp. 886–893.

[126] A. K. Talukdar and M. K. Bhuyan, "A novel global motion estimation and compensation framework in compressed domain for sign language videos," in *2020 Int. Conf. Wireless Commun. Signal Process. Netw. (WiSPNET)*. IEEE, pp. 20–24.

[127] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *arXiv preprint arXiv:1403.2877*, 2014.

[128] V. Athitsos, C. Neidle, S. Sclaroff, J. Nash, A. Stefan, Q. Yuan, and A. Thangali, "The American sign language lexicon video dataset," in *2008 IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. Workshops*, pp. 1–8.