3-7-2008

Dynamic Programming with Multiple Candidates and its Applications to Sign Language and Hand Gesture Recognition

Ruiduo Yang
University of South Florida

Follow this and additional works at: https://scholarcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation
Yang, Ruiduo, "Dynamic Programming with Multiple Candidates and its Applications to Sign Language and Hand Gesture Recognition" (2008). Graduate Theses and Dissertations.
https://scholarcommons.usf.edu/etd/574

This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Dynamic Programming with Multiple Candidates and its Applications to Sign Language and Hand Gesture Recognition

by

Ruiduo Yang

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Sudeep Sarkar, Ph.D.
Dmitry B. Goldgof, Ph.D.
Arthur I. Karshmer, Ph.D.
Barbara L. Loeding, Ph.D.

Date of Approval:
March 7, 2008

Keywords: Sign Language Recognition, Movement Epenthesis, Hand Segmentation, Hidden Markov Models, Dynamic Time Warping, Level Building

© Copyright 2008, Ruiduo Yang
DEDICATION

To my family
# TABLE OF CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ABSTRACT vii

CHAPTER 1 INTRODUCTION 1
  1.1 Overview and Introduction 1
  1.2 Contributions 13

CHAPTER 2 OBJECTIVES AND RELATED WORKS 15
  2.1 Overview of Objectives 15
  2.2 Background on Sequence Labeling/Classification Algorithm 17
  2.3 Background on Hands Localization 20

CHAPTER 3 HIGH LEVEL MATCHING 22
  3.1 Problem Formulation 23
  3.2 The Enhanced Level Building Algorithm 24
    3.2.1 Dynamic Programming 24
    3.2.2 Grammar Constraint 29

CHAPTER 4 SINGLE SIGN MATCHING 32
  4.1 Coupling Groups with Deterministic Matching Algorithm 33
    4.1.1 Formulation of the Matching Process 33
    4.1.2 Dynamic Programming 36
  4.2 Coupling with Hidden Markov Models 38
    4.2.1 Maximal Observation, Summed State 39
    4.2.2 Summed Observation, Summed State 41
    4.2.3 Maximal Observation, Maximal State 43

CHAPTER 5 FEATURE REPRESENTATION 45
  5.1 Low Level Representation 45
    5.1.1 Detection of Hands 46
    5.1.2 Global Features 48
    5.1.3 Multiple Candidates Representation 51
LIST OF TABLES

Table 2.1 Summary of possible problems in ASL recognition. 17
Table 2.2 Image constraints we used in our imaging process. 18
Table 7.1 Summary of the data sets used in this work. 71
Table 7.2 Outline of study 1 – study 4. 75
Table 7.3 Outline of study 5 and study 6. 76
Table 7.4 Comparison of automatically chosen $\alpha$ and manually chosen $\alpha$. 81
Table 7.5 Framewise labeling results. 84
Table 7.6 List of matched scores for 3 test sentences. 97
Table 7.7 Matched positions and manually recognized positions. 98
Table 7.8 Compare results with grouping and without grouping. 100
Table C.1 Sequences used as the text corpus in $D_1$. 130
Table C.2 More sequences used as the text corpus in $D_1$. 131
Table C.3 Sequences used as the text corpus in $D_2$. 132
Table C.4 Sequences used as the text corpus in $D_3$. 132
LIST OF FIGURES

Figure 1.1 Example of me frames. 3
Figure 1.2 Different approaches to handling movement epenthesis (me). 4
Figure 1.3 Appearance change over time. 7
Figure 1.4 Overview of the multiple candidates recognition algorithm. 8
Figure 1.5 All the components in our experiments. 9
Figure 1.6 Example frame of the data set 1 we use. 10
Figure 1.7 Example frame of the data set 2 we use. 11
Figure 1.8 Example frame of the data set 3 we use. 11
Figure 1.9 Example frame of the data set 4 we use. 12
Figure 3.1 The recursive nature of the Level Building algorithm. 26
Figure 3.2 The result of the enhanced Level Building matching process. 28
Figure 4.1 Illustration of the minimization problem. 35
Figure 4.2 Nodes with the predecessors (shown partly for a better view). 36
Figure 4.3 Illustration of the local constraints. 37
Figure 4.4 Illustration of the indexed forward process. 42
Figure 5.1 Intermediate results for the process of hand segmentation. 47
Figure 5.2 Hand segmentation results using all frames. 49
Figure 5.3 Hand segmentation results using key background frames. 50
Figure 5.4 The confidence map and the generated candidate hands. 52
Figure 5.5 The proposed HMM model. 54
Figure 7.20  Recognition of hand gestures for five different approaches. 102
Figure 7.21  Recognition performance of each hand gesture. 103
Figure 7.22  Recognition performance of each subject separately. 104
Figure 7.23  The optimal groups corresponding to one of the hands. 105
Figure 7.24  The tracking results for the first test instance. 106
Figure A.1  Setup for capturing data set $D_2$ and $D_3$. 120
Figure B.1  Overview of the two-pass approach. 124
Figure B.2  The graphic user interface. 126
Figure B.3  Performance of the annotation tool. 127
Figure B.4  Illustration of the multiple candidates. 128
Dynamic programming has been widely used to solve various kinds of optimization problems. In this work, we show that two crucial problems in video-based sign language and gesture recognition systems can be attacked by dynamic programming with additional multiple observations. The first problem occurs at the higher (sentence) level. Movement epenthesis [1] (me), i.e., the necessary but meaningless movement between signs, can result in difficulties in modeling and scalability as the number of signs increases. The second problem occurs at the lower (feature) level. Ambiguity of hand detection and occlusion will propagate errors to the higher level. We construct a novel framework that can handle both of these problems based on a dynamic programming approach.

The me has only be modeled explicitly in the past. Our proposed method tries to handle me in a dynamic programming framework where we model the me implicitly. We call this enhanced Level Building (eLB) algorithm. This formulation also allows the incorporation of statistical grammar models such as bigrams and trigrams. Another dynamic programming process that handles the problem of selecting among multiple hand candidates is also included in the feature level. This is different from
most of the previous approaches, where a single observation is used. We also propose a grouping process that can generate multiple, overlapping hand candidates.

We demonstrate our ideas on three continuous American Sign Language data sets and one hand gesture data set. The ASL data sets include one with a simple background, one with a simple background but with the signer wearing short sleeved clothes, and the last with a complex and changing background. The gesture data set contains color gloved gestures with a complex background. We achieve within 5% performance loss from the automatically chosen score compared with the manually chosen score. At the low level, we first over segment each frame to get a list of segments. Then we use a greedy method to group the segments based on different grouping cues. We also show that the performance loss is within 5% when we compare this method with manually selected feature vectors.
1.1 Overview and Introduction

In this work, we propose algorithms, based on dynamic programming, to attack two fundamental problems in video-based sign language/hand gesture recognition. The first problem is the movement epenthesis (me) issue. This problem is brought by the transition between two signs. We propose an enhanced Level Building algorithm (eLB) to attack this problem without any explicit modeling of me. The second problem is the low level hand segmentation problem. We propose a grouping algorithm and match the groups with a new decoding process. This algorithm allows us to avoid the need for perfect segmentation at the low level feature extraction step.

Movement epenthesis (me) effect is one problem that occurs in the sign language/gesture sequence. In the phonological processes in sign language, sometimes a movement segment needs to be added between two consecutive signs [2]. This is called movement epenthesis (me). Fig. 1.1 shows an example of me frames. These frames do not correspond to any sign and can involve changes in hand shape and movement. They can be over many frames sometimes equal in length to actual signs.

The me effect has been considered in prior efforts. The earliest work that explicitly modeled movement epenthesis in a continuous sign language recognition system with dedicated HMMs can be found in [3] by Vogler et al.. They also used context dependent signs [4] to model movement epenthesis and signs together. Similar to their
approach, Yuan et al. [5] and Gao et al. [6] also model the movement epenthesis explicitly and do matching with both the sign model and movement epenthesis model. The difference is that they adopt an automatic approach to cluster the movement epenthesis in the training data first. Other than this, we [7] also use conditional random fields (CRF) to segment the sentence by removing me segments. In this work, we also compare the CRF approach with the new proposed framework.

The experimental results have shown that the approaches to explicitly model movement epenthesis yield results superior to both ignoring movement epenthesis effects and context dependent modeling. Our experimental results also show that using discriminative models such as CRF can achieve better results compared to a generative model like a Hidden Markov Model. However, the major question of scalability still remains, because these approaches need to explicitly model the me. To obtain enough training data to train the models of movement epenthesis with N signs, one may expect the number of movement epenthesis models to be $O(N^2)$. Also, to build the model of movement epenthesis, one has to extract the associated frames from a set of specific sentences in the training data, either manually or automatically. Hence, the model can be easily biased to this set of sentences.

Unlike previous approaches, we take a dynamic programming approach to address the problem of movement epenthesis without explicit modeling of me, building upon the idea in [8,9]. This matching does not place demands on the training data as much as probabilistic models such as HMMs do. We illustrate the difference between our approach with the one that ignores movement epenthesis or the one that explicitly models movement epenthesis in Fig. 1.2. Fig. 1.2 (a) represents a matching procedure that ignores me and matches all model signs in a model base to a test sentence. Note that the movement epenthesis between two signs can be falsely recognized as one of the signs. Fig. 1.2 (b), on the other hand, illustrates the process of explicitly modeling
all the possible movement epenthesises, where the $\text{me}$ frames in the test sequence are expected to be matched to the modeled $\text{me}$ frames, and not a sign. Fig. 1.2 (c) sketches our approach. We have constructed a model base that consists of all actual model signs, but not movement epenthesis. During the search for the optimal sign sequence in a sentence, we dynamically decide whether a match is a reliable match or not. If not, we label the test frame as an $\text{me}$. Determining the cost of this labeling is crucial and we have an effective, automated method for it. Specifically, we use the Bayesian boundary of the good matches and the bad matches as the cost of the labeling. The entire process is embedded in a dynamic-programming-based Level Building (eLB) algorithm coupled with a grammar model. The search process is conducted in a deterministic manner, where we use Dynamic Time Warping (DTW), constrained by a statistical grammar model. The advantage of the proposed matching process is that implicit segmentation of the sentence into signs happens without the need for modeling movement epenthesis. To create the model base, i.e., for training, we only need the sign frames in a continuous sentence without the associated movement epenthesis frames. This process is done manually.

The second problem we are attacking by using a dynamic programming process is the low level segmentation problem. For a pure video sign language sequence, a frequency domain representation of the frame generally cannot provide enough information for describing hand shape, hand position, orientation, motion, etc. Instead,
Figure 1.2 Different approaches to handling movement epenthesis (me). (a) If the effect of me is ignored while modeling, it will result in some me frames falsely classified as signs. (b) If me is explicitly modeled, building such models will be difficult when the vocabulary grows large. (c) The adopted approach in this work does not explicitly model mes. We allow for the possibility for me to exist when no good matching can be found.
preprocessing is usually required to track or segment the hands. Even for a simple background, this can be hard. The reason is that for a sign language gesture, two hands may come across each other and the two hands may come across the face. Due to these complex issues, previous continuous ASL recognition has mostly relied on external devices to obtain feature vectors. For example, Volger et al. [4, 10, 11] used a 3D tracking system and Cyber gloves, Wang and Gao et al. [12] used cyber gloves and a 3D tracker, Starner et al. [13, 14] used color gloves, accelerometers and head/shoulder mounted cameras, Kadous [15] used power gloves. Although using external devices can yield better results, it also makes the signers feel uncomfortable, and then changes the appearance of a normal sign. Some other approaches use only a single camera without external devices but with constraints. For example, Bauer and Kraiss used [16, 17] a single color camera without external devices. However, they did need a uniform background to perform the recognition. Cui et al. [18] used a segmentation scheme under a complex background, but their approach was working on an image sequence for isolated signs.

For pure video sequences, low level processes are never perfect. Skin color is the most commonly used cue for segmenting image parts from the hand or face in gesture analysis. However, this does not always produce perfect segmentation, with over segmentation being a particularly hard problem to handle. Fig. 1.3 shows an example of the illumination and shading change that can be represented in a gesture sequence. If parts of the image from the finger and the palm are not grouped together, high level matching will be starved of crucial information related to recognizing finger spelled words in sign language recognition.

To help overcome the problem of over segmentation, one approach of ours is to use an intermediate grouping process. The goal of this process is to form groups of low level image primitives which most likely form one part of interest, participating in the
action being observed, one obvious example being the hands. So as not to shortchange
the subsequent recognition process by insisting on disjointed groups, as is usually the
practice in grouping, we allow for overlapping groups, resulting in redundant sets of
groups.

At the beginning of such a grouping process, some region patches are selected
as seeds based on their size. We then grow these seeds with adjacent regions to
generate groups. As the seeds are grown, groups are checked for the possibility of
being possible hands based on size and shape. Grouping can be conducted based on
color, position, boundary smoothness or boundary gradient. These basic similarity
cues resemble those adopted by Hoogs and Mundy to group region patches [19] for
object recognition, where they used spatial intensity, parallelism and perimeter to
form an object hypothesis. However, unlike them we perform the grouping based on
each criterion independently of each other. Each criterion results in a set of groups,
which we refer to as a grouping layer. Thus we have grouping layers: the color
grouping layer, the proximity grouping layer and the boundary smoothness grouping
layer. The grouping strategy used is the same for each layer and is discussed later.

From our literature survey, we found that there are previous approaches that have
also used multiple hand candidate representations. For example, the combination of
top-down and bottom-up approaches in gesture sequence recognition can be found
in [20] and [21]. They both used skin and motion cues to generate multiple candi-
dates. However, these previous works are all based on isolated gesture recognition,
and they do not have an intermediate grouping scheme to produce enough shape
information. Our multiple candidate approach for continuous ASL recognition is
put into a Level Building framework where a continuous sign language sentence can
be analyzed. The elegant aspect of the approach is that we can analyze movement
epenthesis me, two-handed and single-handed signs in a unified framework. For single
sign/gesture recognition, our grouping scheme can also effectively generate the true hand candidates to prevent any errors at the feature detection level.

The overall structure, test and contributions are illustrated in Fig. 1.5. At the feature extraction level, we have tested both the traditional feature vectors and multiple candidate features. For a single feature vector, we tested on CRF, HMMs, LB and our proposed eLB algorithms. For multiple feature vectors, our tests were based on HMMs, DTW at the sign matching level and the eLB, LB at the sentence level. We also tested our grouping schemes based on HMMs and DTW models. Fig. 1.5 shows us the place where we proposed two core matching algorithms. The two modules with the box in shadow indicate the two main proposed matching algorithms in this work.

We experimented with different kinds of single view video data sets. Some sample frames are shown in Fig. 1.6, 1.7, 1.8 and 1.9, from which we can see that we use three data sets for American Sign Language and one data set for simple hand gestures. Fig. 1.6 shows the first data set, where a clean background is used. Fig. 1.7 shows the data set 2, where we have a complex and moving background. Fig. 1.8 shows the data
Figure 1.4 Overview of the multiple candidates recognition algorithm. First, the original frames are segmented, and the groups in each frame as well as the links between frames are produced. Then the sequences of linked candidate groups are matched to the model groups in the database.
Figure 1.5 All the components in our experiments. From bottom, the training data is processed and groundtruthed. From the top, we generate two types of feature vectors in our experiments. For a single feature vector, we tested using CRF, HMMs, LB and our proposed eLB algorithms. For multiple feature vectors, we tested based on HMMs, DTW at the sign matching level and the eLB, LB at the sentence level.
Figure 1.6 Example frame of the data set 1 we use.

set 3 where we have short sleeved clothes. Fig. 1.9 shows the data set 4, where we have a complex background for a single gesture sequence. Besides the different setup of data sets, we also tested different matching algorithms such as traditional Level Building (LB) approach, conditional random fields, Hidden Markov Models, etc. We will show the effectiveness of our proposed algorithm compared to these standard ones.

Before we proceed, we have organized our work to answer the following research questions related to the two fundamental problems we are attacking. We will answer these questions in the following chapters.

1. Can we handle the movement epenthesis problem without the need for explicitly modeling me segments? If we do not explicitly model me segments, how can we associate a matching score to each me segment?

2. Not only do we need to detect the existence of each me in an ASL sentence, but also we need to explicitly locate the position and the length of each me, even though we do not know beforehand how many mes we will have in a sentence. In addition, an me can happen at any position in a sentence (in terms of frame number), and it is not always the same length. So in order to conduct the search,
Figure 1.7 Example frame of the data set 2 we use.

Figure 1.8 Example frame of the data set 3 we use.
we must search all the possible start positions, along with all the possible lengths and all the possible occurrences of me. This search space can be huge. How can we limit it?

3. How can a statistical grammar model, such as bigrams and trigrams, be incorporated into the solution approach?

4. How can we handle imperfect segmentation at the low level? How can one use feature grouping processes to overcome segmentation errors?

5. Can our proposed set of algorithms handle a complex background? Can we identify signs made by signers wearing both short and long sleeves, i.e., relax the typical clothing constraints?

6. How well does the recognition rate with the proposed approach match with that achieved through manually grouped segmentation?

Among these 6 research questions above, questions 1 – 3 are related to the movement epenthesis problem, which exists in sign language/continuous gesture recognition domains. Questions 4 – 6, however, are related to fundamental computer vision issues that cut across many different application domains, beyond sign lan-
guage/continuous gestures. We will try to answer these questions in their corresponding chapters in the remaining part of the dissertation.

1.2 Contributions

In this work, we strive to solve two fundamental problems in automatic video-based sign language and gesture recognition systems. The first problem is the movement epenthesis (me) problem. This problem results from the transition movements a signer makes between two signs. We propose an enhanced Level Building algorithm (eLB) [8, 9] to attack this problem without any explicit modeling of me. The second problem is the low level hand segmentation problem. We propose a grouping algorithm and match it with a new decoding process [22, 23]. This algorithm allows us to work without the need of perfect segmentation at the low level feature extraction step.

Previous approaches tried to explicitly model movement epenthesis, but scalability became a big problem. We are the first to use a boundary to model all of the dynamic effects of me instead of the me itself. And we embed it into an optimal framework to produce the labeling and segmentation simultaneously for continuous sign sentences. This approach greatly reduces the efforts to model the me directly and the problem of insufficient training data of me. Its effectiveness is tested in our experiments. We have compared our eLB algorithm with other conditional models that are state of the art, which are also good for limited training data sets. We show that the method of conditional random fields (CRF) can work better under a 2-class case [7]. In sign language recognition where we have large numbers of classes, these methods can not work effectively. However, the eLB methods can still produce good results.

In our experiments, we feed different types of feature vectors to the eLB framework. One of the important contributions in this step is that we use multiple grouped
observations as our input. We are the first to use multiple linked group sequences to match a sequence model. We develop algorithms to couple the group sequence in both a deterministic domain (Dynamic Time Warping) and a probability domain (Hidden Markov Models). We provide a novel grouping strategy that can generate multiple overlapping groups to reduce the chance of missing the true hands in the low level. We group the frame based on the over segmentation result and we use different grouping cues as the basis. We show that the grouping process can effectively reduce the chance of losing the real observations, but this cannot be done without a grouping process.

As a byproduct of this research, we have also produced various research tools that can be used by anyone in gesture recognition [24]. These are included in Appendix B. We share our source code for our algorithms and the tools on the website: "http://figment.csee.usf.edu/ASL/", so that others can reproduce our results or use our algorithms.

In the following parts of the dissertation, objectives and more related works are described in Chapter 2. We discuss the problem of me and the high level DP process in Chapter 3. Chapter 4 describes the low level DP process to handle the ambiguity problem. Chapter 5 states the feature level processing and the generation of hand candidates. We then describe the framework to label sign language sequences using conditional random fields in Chapter 6, which is a very popular conditional model for sequence labeling. We then present the experimental results in Chapter 7, and conclude at Chapter 8.
CHAPTER 2
OBJECTIVES AND RELATED WORKS

2.1 Overview of Objectives

In this work, our main objective is to build an automatic system for recognition of sign languages/gestures from a video sequence. Although most of the methods proposed in this work can work with different feature vectors, such as those features from magnetic gloves or accelerometers, this work is specifically targeting recognition of sign language/gesture sequence from a single image sequence, without any external devices. The motivation of doing this is that using external devices while signing may make the sign unnatural. Even in an application of continuous gestures, using any glove-like external devices will influence the way that gestures are performed. Hence, the gesture performed will be different from the one that is performed without such a device.

Of course, we are aware that this particular problem is not trivial. In fact, in such an automatic system, many more problems need to be considered other than a hand tracking problem. In Table 2.1, we list some common problems which are associated with such a system. We also indicate if we have considered these particular problems in our system. In Table 2.1, the first column states the specific problems. The movement epenthesis problem (me) is the problem of the inserted me segments to transport the hands between two successive signs. Note this problem is different from the problem of coarticulation [1]. Coarticulation in sign language refers to the changing
aspects of signs when they overlap in time [1]. The grammar problem is the problem of determining whether a sequence is meaningless or not. That is, the meaningless sentences need to be pruned from the final recognition. The coarticulation problem is the problem that a sign/gesture may be performed differently (changing hand shape, orientation, etc.) while in a sentence, then the overlapping part of two consecutive signs/gestures will have changed representations [1]. The non-manual problem is that the facial expressions will influence the meaning of a sign sentence even if the manual part of a sentence is the same. The signer independent problem is the fact that the variation of a performance of a sign/gesture among different signers can result in modeling failure and incorrect results. The short sleeve problem is that the short sleeved clothes may make it difficult to segment the hands, even if over segmentation is used. The hand segmentation problem is the general problem that a hand segmentation/tracking may fail for a video sequence. The view independent problem is the problem that the testing sequence can be taken from a different viewpoint from the training sequence and the feature vector used must be able to accommodate this.

In this work, we have developed methods for the problem, grammar problem, short sleeve problem, and the hand segmentation problem. We also provide experimental results for these problems. For signer independent problem, we do not propose any specific methods, but one of our test data does have signer independent cases. In this work we have not proposed any methods or experimental results to handle the coarticulation problem, the non-manual problem or the view independent problem.

On the other hand, these problems must be attacked under a specific imaging restriction. In [25], several imaging restrictions and constrains have been discussed. Our work is targeting an application of a sign language translator within the airport domain, which means we may have a complex moving background, but our camera can be still. According to this, Table 2.2 lists the image restrictions we may have.
Table 2.1 Summary of possible problems in ASL recognition.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Considered</th>
<th>Novel methods</th>
<th>Experimented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movement epenthesis problem</td>
<td>Yes</td>
<td>Yes, eLB</td>
<td>Yes</td>
</tr>
<tr>
<td>Grammar problem</td>
<td>Yes</td>
<td>Yes, bigram, trigram and sentence with eLB</td>
<td>Yes</td>
</tr>
<tr>
<td>Coarticulation problem</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Non-manual problem</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Signer independent problem</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Short sleeved clothing problem</td>
<td>Yes</td>
<td>Yes, skeleton with multiple candidates</td>
<td>Yes</td>
</tr>
<tr>
<td>Hand segmentation problem</td>
<td>Yes</td>
<td>Yes, using fragments and multiple candidates</td>
<td>Yes</td>
</tr>
<tr>
<td>View independent problem</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

corresponding to the discussions in [25]. From Table 2.2, we can see that we try to use as few restrictions as possible. We only use the first three types of restrictions for some of our data sets. We have considered both the moving background and short sleeved cases in our experiment, based on our proposed framework.

2.2 Background on Sequence Labeling/Classification Algorithm

The final objective of this system is to generate a label sequence related to the image sequence. Many algorithms have been proposed to label a sequence and they have been used in labeling a sign language/gesture sequence. For example, Dynamic Time Warping, Hidden Markov Models, and conditional random fields are the major labeling methods. Extensions based on these methods include statistical DTW [26], parallel HMM [27], Maximum Entropy Markov Models (MEMM) [28], Hidden CRF [29] and LDCRF [30], etc. Among them, CRF is a statistical discriminative model. It has several advantages such as its ability to label the sequence regarding to a global optimal manner, and its ability to directly model the posterior probabilities.
Table 2.2 Image constraints we used in our imaging process. These constraints are used in [25].

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Used</th>
<th>Discussed</th>
<th>Experimented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long sleeved clothing</td>
<td>Yes some data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Colored gloves</td>
<td>Yes some data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Uniform background</td>
<td>Yes some data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Complex but stationary background</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Head/face required to be stationary or have less movement than hands</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Constant movement of hands</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Fixed body location and pose or specific initial hand location</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Left hand or face excluded from the view</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Vocabulary restricted or unnatural signs to avoid overlapping hands or hand over face</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Field of view restricted to the hand which is kept at fixed orientation and distance to the camera</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
The works of video sign/gesture sequence recognition have been borrowing the successful methods used in the general sequence recognition domain. This is natural since a gesture sequence, can be regarded as a time series with observations at each time frame, just like a speech sequence or text sequence. Some works used a normal classifier instead of a sequence classifier for time series. These works normally focus on a classification of hand properties and motion types such as [31, 32] in sign language, and, [33–35] in general gesture classification. However, these methods lack the ability to model the contextual information in both 2D images and 1D sequences. Hence, their popularity has been overcome by specific sequence classifiers that can accommodate these such as HMMs and CRF. And since sign language/gesture sequences are sequences with highly contextual information, in the case of continuous sign language/gesture recognition, a time series classifier such as HMMs, CRF, and DTW is more often used. Among all of the works, the HMM is the most popular one along with its variations. The HMM produces a model which can statistically capture the different states of a sequence and also the changing properties among these states. It offers a concise representation of complex sequence models with different variations with the starting probabilities, transition probabilities and state distributions. It also offers the attributes that a sequence can be implicitly segmented while the labeling is done. Works in sign language recognition [12, 14, 27, 31, 36, 37] and general gesture recognition [20, 38–40] have used the HMM and its variations. Among them, subunits are used in [37], parallel modeling is used in [27], automatic clustering is used in [12, 31], grammar model is used in [31, 36], coupled modeling is used in [38], layered model is used in [39], multiple candidates are used in [20], and multi-linked modeling is used in [40].

The major problem of the HMM is that it is a generative model, which may need many data to train the state distribution. However, the state distribution is not
the final distribution that we are interested in, in a general sign/gesture classification problem. Therefore, modeling such a distribution is not absolutely necessary, although it is natural. The same problem occurs in the text sequence labeling community, where conditional model, recently, is proposed to overcome this problem. There have not been many works in sign language/gesture that use conditional models to classify gestures. Sminchisescu et al. are the first to use CRF for gesture recognition in [41] for human action classification. Variations of CRF can be seen in [30] using multiple states for eye gaze gesture and in [7] using key frames for sign languages. In this work, we also include the work using conditional random fields to simultaneously segment and label each sign language video sequence. We will show in the experiment section that CRF can outperform generative method in a 2-class situation. However, for a multi-class situation where the number of classes are too large, the model can hardly find a optimal boundary at training due to the fact that the number of parameters is too large.

2.3 Background on Hands Localization

Gesture recognition, and the related area of automated sign language recognition, is a rich area of research (see [25,42,43] for reviews) with many different applications and approaches, but sharing some common problems and solutions. Vision-based approaches all share the problem related to the vagaries of low level segmentation. The states in a state-space-based gesture representation, such as the Hidden Markov Models [3,44–48] or Dynamic Time Warping [35,49,50] or, Finite State Machine (FSM) [51] approaches are based on the low level features detected in the image. Motion tracks in trajectory-based gesture recognition approaches [32,52] are dependent on the robustness of the tracking process, which in turn, is dependent on the stability of the low level segmentation. This problem of low level segmentation is sometimes
addressed by engineering the imaging setup so as to ease the segmentation of hands by using controlled lighting, colored gloves or even non-vision-based aids such as magnetic or optical markers. Pure vision-based solutions usually rely on skin color and/or motion information to detect hands. The skin color related systems include [53–57]. However, approaches based on predefined skin color models suffer from sensitivity with respect to changing illumination conditions.

Motion-based hand segmentation approaches [54,58] rely on the assumption that the features important for the gesture will be associated with motion. This is not always true for sign recognition, which includes movement and hold phases. Fusion [59, 60], multi-modal [61], Haar-like Features [62], and accelerometers [63], 3D [64, 65] approaches can be used to arrive at better segmentation and detection. However, segmentation will never be perfect. Not only will there be missed detections, but there will also be false alarms. There is danger that these errors will be propagated to the recognition stage. In this work, we advocate using an intermediate grouping module, coupled with the recognition module, to handle low level segmentation errors. Such grouping processes have been found to be useful for object recognition tasks [66–68], but have not been used for gesture and sign recognition. The combination of top-down and bottom-up approaches in gesture sequence recognition can be found in [20] and [21]. Although these approaches can handle multiple candidate observations, there is no grouping process incorporated. For example, a sliding window is used along with a skin color model in both [21] and [20] to obtain the position of the moving hands. However, in real world application, bad lighting conditions may cause problems for skin color approaches, and a sliding window cannot be sufficient in some applications where exact hand shape is needed. Apart from hand gestures, Srinivasan et al. also proposed grouping method [69] to classify human bodies. However, their approach works only for single images.
CHAPTER 3
HIGH LEVEL MATCHING

In this chapter, we will answer the research questions 1 – 3. These questions are related to the movement epenthesis problem in sign languages. We will show how we can implicitly model me, how we solve the problem of combinatorics in the search for the optimal sign/me sequence, and how we can incorporate a grammar model in our framework. These three research questions are outlined again as below:

1. Can we handle the movement epenthesis problem without the need for explicitly modeling me segments? If we do not explicitly model me segments, how can we associate a matching score to each me segment?

2. Not only do we need to detect the existence of each me in an ASL sentence, but also we need to explicitly locate the position and the length of each me, even though we do not know beforehand how many mes we will have in a sentence. In addition, an me can happen at any position in a sentence (in terms of frame number), and it is not always the same length. So in order to conduct the search, we must search all the possible start positions, along with all the possible lengths and all the possible occurrences of me. This search space can be huge. How can we limit it?

3. How can a statistical grammar model, such as bigrams and trigrams, be incorporated into the solution approach?
3.1 Problem Formulation

Our model base of signs consists of instances of individual signs, \( \{S_1, S_2, \cdots, S_V\} \). Each instance of a sign, \( S_i \), is a sequenced set of individual frames. In addition to these signs, we use \( \text{me} \) symbols or labels of various lengths to represent movement epenthesis. They vary in length from 1 to \( N_{\text{max}} \), the maximum period over which the movement epenthesis effect can persist. We chose not to have explicit models corresponding to these symbols.

Let a query or test sequence of length \( M \) be denoted by \( T \). A solution to the matching problem would consist of a segmentation of \( T \) into signs and movement epenthesis, along with labels for each segment. We denote the segmentation of a sentence using the sequence of indices: \( \{j_0, j_1, \cdots, j_{L_k}\} \), where \( L_k \) denotes the number of segments in the \( k \) possible segmentations of the sentence. Thus, the first segment is from index \( j_0 = 1 \) to \( j_1 \), the second segment is from \( j_1 + 1 \) to \( j_2 \), and so forth. Let \( S_k \) denote the sign label for the \( i \)th segment over frames \( t^{j_{i-1}} \) to \( t^{j_i} \). Then, the \( k \)th possible solution sequence is denoted by

\[
S_k = \{S_{k_1}, S_{k_2}, \cdots, S_{k_{L_k}}\} \quad (3.1)
\]

where \( S_{k_1} \) is the first sign label in the sequence, \( S_{k_2} \) is the second sign label in the sequence, and so forth. Note that some of these sign labels could be the \( \text{me} \) labels of different length. \( L_k \) denotes the number of signs in sequence \( S_k \). The total number of such possible label sequences is, of course, exponential.

Our objective is to find a sequence of sign and movement epenthesis (\( \text{me} \)) labels, \( S_e \), among all possible sign sequences such that the distance between \( S_k \) and \( T \) is
minimized. That is, we need to find $S_e$ such that

$$S_e = \arg \min \text{Cost}(S_k, T(1 : j_{L_k})) = \arg \min_{j_1, \ldots, j_{L_k}} \min\{S_{k1}, \ldots, S_{L_k}\} \sum_{i=1}^{L_k} D(S_{ki}, T(j_{i-1} : j_i))$$

where $D(.)$ is the function to compute the single sign matching cost and $T(j_{i-1} : j_i)$ is a segment of the query sentence between indices $j_{i-1}$ and $j_i$. The nature of this cost function can differ based on the situation at hand. For instance, if we have very good segmentation of hands and faces, then one could construct reliable feature vectors for each frame. In such situations, the distance would be constructed by Dynamic Time Warping of the segments. If on the other hand, we do not have reliable extraction of hands, then we suggest a more complex solution that involves optimizing over possible hand candidates. We will look into these distance computations methods. But, before that, let us consider how we perform the optimization in Eq. 3.2, assuming that we have been given the existence of distance measure.

3.2 The Enhanced Level Building Algorithm

The solution of Eq. 3.2 is over all the possible sign sequence candidates, with all possible lengths of each sign, $\{S_{k1}, S_{k2}, \ldots, S_{kL_k}\}$. This search space is very large. We structure the search for the optimal solution using dynamic programming, specifically, the Level Building approach [70] and enhance it to allow for movement epenthesis labels.

3.2.1 Dynamic Programming

The overall minimization can be expressed recursively as optimization of one label and the minimum cost for the remaining sentence. If we structure this optimization
separating the last label, we have

\[
\min \text{Cost}(S_k, T(1 : j_{L_k})) = \\
\min_{S_{k_{L_k}}, j_{L_k}} \left( D(S_{k_{L_k}}, T(j_{L_k} - 1 : j_{L_k})) + \text{Cost}(S_{L_k - 1}, T(j_1 : j_{L_k - 1})) \right)
\]  

(3.3)

Based on this decomposition of the problem, each level of the Level Building approach corresponds to the labels, in order, in the test sentence. Thus, the first level is concerned with the first possible label in the sentence. The first label could cover different possible lengths. The second level is concerned with the second possible label for the portion of the sentence that begins after the first label ends, and so forth. Each level is associated with a set of possible start and end locations within the sequence. And at each level, we store the best possible match for each combination of end point from the previous level. The optimal sequence of signs and labels is constructed by backtracking.

For each level \(l\), we store the optimal cost for matching between sign \(S_i\) and with the ending frame as \(m\) using a 3 dimensional array \(A(l, i, m), 1 \leq l \leq L_{\text{max}}, 1 \leq i \leq N, 1 \leq m \leq M\), where

\[
A(l, i, m) = \begin{cases} 
D(S_i, T(1 : m)) & \text{if } l = 1 \\
\min_{k,j} A(l - 1, k, j) + D(S_i, T(j + 1 : m)) & \text{otherwise}
\end{cases}
\]  

(3.4)

\(T(j : m)\) denotes a subsequence of the test sequence that starts at the \(j\)th frame and ends at the \(m\)th frame. This recursion is pictured in Fig. 3.1.

The quantity \(A(l, i, m)\) gives us the minimum cumulative score for matching \(l\) labels, with the \(i\)th model sign, \(S_i\), as the last label to the test sequence up to the
Figure 3.1 The recursive nature of the Level Building algorithm. To compute the $A(l, i, m)$, we search among all the results in the previous level $A(l-1, k, j)$ plus the current level’s matching $D(S_i, T(j+1 : m))$ and find the minimum value.

$m$th frame. The optimal matching score $D^*$ is:

$$D^* = \min_{l,i} A(l, i, M) \tag{3.5}$$

To enable us to reconstruct the optimal sign sequence by backtracking, we use a predecessor array $\psi$, whose indices correspond to $A$: $\psi(l, i, m), 1 \leq l \leq L_{\text{max}}, 1 \leq i \leq N, 1 \leq m \leq M$, where

$$\psi(l, i, m) = \begin{cases} -1, & \text{if } l = 1 \\ \arg \min_k A(l-1, k, j) + D(S_i, T(j+1 : m)) & \text{otherwise} \end{cases} \tag{3.6}$$

Fig. 3.2 illustrates the possible matching sequences searched during the recursive search process. At the end of each level, we obtain the best matched sequences. For example, at level 1, all the matching must start at frame 1. There are a range of possible ending frames for level 1. For each possible ending frame, we obtained
a best matching sign, for instance $S_1, S_5, S_2, S_{y+4}, S_2, S_9$ respectively, shown in the figure. Then at level 2, we again have a range of possible ending frames. The starting frame will be after the ending of the first level. For each ending frame, we find the best cumulative matching score we can have among all the signs and possible starting frames. We continue this process for all the levels. Matchings that end at the last frame result in one possible matching sequence, which can be constructed by backtracking from the last frame. Some example sequences shown in the figure are \{S_9, S_1\}, \{S_2, S_8, S_9\}, \{S_1, \text{me}, S_2, \text{me}\}. Note all the signs $S_{V+k}$ are actually \text{me} labels. This process also shows us our answer to the research question 2. To limit the search space, we use the dynamic programming approach, where the intermediate search for a partial sequence result can be used to build up towards the final search result for the whole sequence.

The use of the \text{me} label is the essential difference between the classical Level Building formulation for recognizing connected words in speech and our formulation for recognition of connected signs in sign languages. We enhance the classical formulation by allowing for such labels, hence the name enhanced Level Building (eLB). However, allowing for such label is not equivalent to the addition of an additional sign label since it is not obvious how to choose the cost of \text{me} label because there are no real sample of it. We choose the cost of associating an \text{me} label to an observation sequence to be proportional to its length.

$$D(S_{V+k}, T(j + 1, m)) = (m - j)\alpha$$  \hspace{1cm} (3.7)$$

This raises the question of how does one choose the proportionality constant, $\alpha$. One viewpoint is that this is really a penalty cost of assigning an \text{me} label to a frame. This penalty should be larger than a good match score we can find, since each time
Figure 3.2 The result of the enhanced Level Building matching process. There are 3 complete matched sequences ending at levels 2 through 4. The best one among these three will be returned as the matching result for these levels. Note, all the signs $S_{V+k}$ are actually $me$ labels.
we find a good match to a portion of the unknown sequence from our database, we want to keep it. At the same time, the penalty should be smaller than a non-match score, because each time we cannot find any good match, we need to make sure the match is selected. A non-match score is obtained for matching two different signs and a match score is obtained when matching different instances of the same sign. To obtain these scores we consider the distribution of match and non-match scores between signs in the training set, computed using Dynamic Time Warping (discussed later). The overall distances are normalized by the length of the warping path. The distribution of these scores typically has overlap. We search a threshold value that one can use to classify these scores into match and non-match ones. We choose the optimal $\alpha$ to be the optimal Bayesian decision boundary to accomplish this. However, instead of parametrically modeling each distribution (match and non-match) and then choosing the threshold, we use a histogram-based representation to search for it. With this, we answered our research question 1. Basically, the way we implicitly model $\text{me}$ is that we do not associate any actual frame to the $\text{me}$, but we use a boundary score to describe the matching cost of $\text{me}$ against the test frame directly. However, traditional methods which explicitly model $\text{me}$ will need the information from the actual frames (in training data), and the matching score is actually computed against the test data at the testing time.

3.2.2 Grammar Constraint

The explorations at each level can be constrained by statistical grammar information such as those captured by $n$-gram statistics. We illustrate this using a bigram model. We use a sample-based model of the bigram, instead of an histogram one. We
represent it using a relationship matrix \( R(i, j) \), \( 1 \leq i \leq N, 1 \leq j \leq N \), where we have

\[
R(i, j) = \begin{cases} 
1, & \text{if } S_i \text{ can be the predecessor of } S_j \\
0, & \text{if } S_i \text{ cannot be the predecessor of } S_j 
\end{cases}
\]  

(3.8)

We set \( R \) based on observed instances in the training text corpus. Entries are set to 1 or 0 if an example is either found or not found in the corpus. Note that this is different from the histogram of counts used in traditional \( n \)-grams. Due to the limited nature of the samples, we do not use counts. Essentially, if we have some evidence, we set the probability of that occurrence as being one. This is a very liberal choice of grammar constraint. To allow for \textit{me} labels before and after each sign, we use \( R(i, j) = 1 \), if \( i > V \) or \( j > V \).

After obtaining \( R \), the eLB algorithm can be constrained with the predecessor relationship based on the relationship matrix. Note that since we allow an \textit{me} label to exist between any two signs, local backtracking may need to be performed while enforcing grammar checking. For example, assume at the current level, we are examining the sign \( S_i \). If the predecessor we found along the optimal path is an \textit{me} label, we need to backtrack until we find a real sign \( S_j \) along the optimal path. Grammar checking is performed ultimately between \( S_i \) and \( S_j \).

We denote the result of the local backtracking for the minimum cumulative distance matrix \( A \) as:

\[
B(i, l, m, k, j) = \beta
\]

(3.9)

where \( S_{\beta} \) is the actual predecessor we found using the local backtracking scheme, when computing \( A(l, i, m) \), along the path where the predecessor is \((l - 1, k, j)\).
Hence, to incorporate a grammar constraint into our system, we can update Eq. 3.4 and Eq. 3.6 as:

\[
A(l, i, m) = \begin{cases} 
D(S_i, T(1 : m)) & \text{if } l = 1 \\
\min_{k,j} A(l - 1, k, j) + D(S_i, T(j + 1 : m)), \text{ such that } R(\beta, i) = 1, \beta = B_i^l(m, k, j) & \text{otherwise}
\end{cases}
\]  

(3.10)

and

\[
\psi(l, i, m) = \begin{cases} 
-1, & \text{if } l = 1 \\
\arg \min_k A(l - 1, k, j) + D(S_i, T(j + 1 : m)), \text{ such that } R(\beta, i) = 1, \beta = B_i^l(m, k, j) & \text{otherwise}
\end{cases}
\]  

(3.11)

In this section, we answered the research question 3. We use the text corpus to build the grammar constraint and use this to prune sentences that are not meaningful. This is the same as normal LB. The difference is that we need to do local backtracking to skip me to build the real sentence for grammar test in our framework.
CHAPTER 4
SINGLE SIGN MATCHING

To compute the final optimal sequence using the eLB framework, we need to be able to compute the cost between a model sign with a subsequence of the test data, which is mathematically expressed as $D(S_{k_1}, T(j_{i-1} : j_i))$ in Eq. 3.2. There are two scenarios that we consider for this matching cost. The first is when we have a single feature vector describing each image frame, and any sign is a sequence of these feature vectors. This would be possible when one has a fairly good hand detection capability by controlling the background and clothing. To compute the single sign matching cost under such situations, we simply compute the Dynamic Time Warping (DTW) cost between the two sequences. As the cost for matching one frame from a model to one observation frame, we consider one possible cost function based on spatial distribution of the image features. We will discuss this cost in a later section.

The second scenario, which is the most common one, arises when we do not have a single detected hand region for each hand in each frame, but have many possible hand regions. For each frame, we can detect many possible hand candidate regions. We can pair these candidates to generate many possible hand candidates. This arises in uncontrolled imaging situations with complex background and lack of control over clothing. Here, the use of global features is obviously not reasonable. One has to opt for more part-based representations. In this chapter, we will describe our modified algorithm to solve Eq. 3.2. This will involve a solution in both the deterministic case and the probabilistic case. We will show the two algorithms separately in Section 4.1
and Section 4.2. We strive to answer the first part of research question 4 in this chapter:

4. How can we handle imperfect segmentation at the low level? How can one use feature grouping processes to overcome segmentation errors?

4.1 Coupling Groups with Deterministic Matching Algorithm

We perform the recognition based on multiple observations using both the deterministic approach and statistical approach. For the deterministic approach, recognition is conducted by matching groups found in any given sequence to each model sequence. In the model sequence, the real hand group is extracted manually frame by frame. The goal of the matching is to find one candidate group sequence (out of the many available, directed by the linked structure), which can be best mapped to the model sequence. This process also allows for time warping, and is shown to be solvable by dynamic programming. After matching to each model sequence, the ones with lower distance scores are considered as the recognition results.

4.1.1 Formulation of the Matching Process

Let the $i$th candidate group in the $k$th frame be represented as $G_k^i$. Also, let $K$ be the number of frames in the test sequence. Similarly, the motion model will consist of a sequence of feature vectors, $M = \{m^1, \cdots, m^T\}$ that will have to be matched to the sequence of candidate groups, with each model feature vector mapped to one group. The matching score is represented with a 3D Matrix $S$, where the $(i, j, g)$ element of $S$ denotes the Mahalanobis distance between the $i$th model feature vector and the $g$th candidate group’s feature vector in the $j$th frame.
The warping path is a sequence of elements of S denoting the matching. Since the model group can be mapped to one of the candidate groups in one of the test sequence frames, the warping is conducted in both the time domain and the candidate group domain. If the cardinality of the candidate group’s feature vector set is one, then of course, this correspondence establishment is trivial (only time warping is needed). Otherwise, we have to select between the possible candidate groups. We cast this problem as a minimization problem that we solve using dynamic programming.

Formally, we have to find a sequence of elements, one from each candidate set, which best matches the model sequence of feature vectors. Let,

1. \( k(t) =< i, j, g > \) be a multi-valued function that maps the indices of the warping path, denoted by \( t \), to the 3D coordinates in S where the model’s \( i \)th frame is matched with the \( g \)th candidate group of the \( j \)th frame in the test sequence.

2. let \( d(m^i, G^j_g) \) represent the cost of matching the model group feature vector from the \( i \)th image frame, \( m^i \), with the \( g \)th group feature vector from the \( j \)th image frame, \( G^j_g \).

Then the total matching cost can be cast as a minimization problem. More formally,

\[
\min_{k(t)} \left( \sum_t d(m^i, G^j_g) \right)
\]

(4.1)

Fig. 4.1 illustrates the minimization space. It is a 3D space spanned by the model sequence time index, \( i \), the given image sequence time index, \( j \), and the feature vector index into the candidate group sets, \( g \). Each point in that space is associated with a cost defined between the corresponding image and model groups. We seek a curve, defined by \( k(t) \), that minimizes the total cost function over this curve, with the following constraint in both the candidate group domain and the time domain.
Figure 4.1 Illustration of the minimization problem. We have to find the warping path that minimizes the difference between the model sequence and the test sequence. Possible solutions are curves in a 3D space, spanned by model sequence index \( i \), image sequence index \( j \), and candidate group set index \( g \).

When we match a gesture to a gesture, where a model sign with length \( T_m \) is matched with a test sign with length \( K \), this curve starts at \( < i = 1, j = 1 > \) and ends at \( < i = T_m, j = K > \). When we match a sign/gesture to a sign sentence, where a model sign with length \( T_m \) is matched with a sentence with length \( K \), this curve can start at any place with \( i = 1 \) and end at any place with \( i = T_m \). We also enforce a constraint when associating adjacent frames. This constraint defines all the possible predecessors of a node on the warping path. The constraint we use when seeking the curve is illustrated in Fig. 4.2. In the time warping domain, we use the general local constraints \[71\]. In the candidate group domain, we use the predecessor relationship in Eq. 5.3 as the constraints.

Fig. 4.2 illustrates for us the nodes in the 3D space, with the predecessors shown as arrows. Only predecessors of a few of the nodes are illustrated in order to show the relationship for the reader. A local illustration is shown at Fig. 4.3, where \( (i, j, g_1) \) is a
node with 7 predecessors, \((i, j - 1, g_1^1)\) is the one which have the previous model frame, but the same test frame and candidate groups as \((i, j, g_1^1), (i - 1, j, g_2^1), (i - 1, j, g_2^1), (i - 1, j, g_3^1)\) are the ones which has the previous test frame, but the same model frame as \((i, j, g_1^1)\), and they have different candidate groups in the previous test frame. Similarly, \((i - 1, j - 1, g_1^2), (i - 1, j - 1, g_2^2), (i - 1, j - 1, g_3^2)\) are the ones which have the previous test frame and previous model frame compared to \((i, j, g_1^1)\), and they have different candidate groups in the previous test frame.

4.1.2 Dynamic Programming

The dynamic programming can be used to obtain the optimal warping path in our problem. In a 3D matrix \(D\), let \(D(i, j, g)\) represent the minimum cumulative cost of matching the model sequence, \(\{m_1^1, \cdots, m_t^t\}\), to the candidate group set sequence up to \((i, j, g)\). The optimal substructure of the problem allows the following recursive
Formula.

\[ D(i, j, g) = d(m^i, G^j_g) + \min \begin{cases} 
\min_{r \in \text{Pre}(G^j_g)} D(i, j - 1, r) \\
\min_{r \in \text{Pre}(G^j_g)} D(i - 1, j - 1, r) \\
D(i - 1, j, g) 
\end{cases} \quad (4.2) \]

Here we use a constraint that the coordinate \((i, j, g)\) in the dynamic programming space is dependent on the locations, \((i, j - 1, g), (i - 1, j, g),\) and \((i - 1, j - 1, g)\). This is based on the general local constraints [71]. The solution to \(D(i, j, g)\) is the solution to our problem. This also answers the first part of our research question 4. To obtain the solution without the need of perfect segmentation, we used a multiple observations approach. So at the low level, we do not have to make the hard decision about where the important parts (hands) are really at. We also have a new framework to recognize these multiple observations, based on a dynamic programming approach.
We will show in the next section that this kind of matching can also be conducted using a probabilistic approach.

### 4.2 Coupling with Hidden Markov Models

In this section, we show the multiple candidates sequence can also be matched to a statistical model like a HMM. While the structure and the training of the HMM is a fairly standard one, the decoding process, i.e., computing the likelihood of an image sequence to the HMM, is significantly different and new. Each gesture $g_i$ is modeled using a HMM $\lambda_i$ over $N$ states. The state at frame $k$ is denoted as $q_k$, where $q_k \in 1, \cdots, N$. $a_{ij} = P[q_{k+1} = j | q_k = i]$ is the state transition matrix. The initial state distribution is denoted as $\pi = \pi_i$, where $\pi_i = P[q_1 = i]$ is the probability that state is $i$ at $frame = 1$. The observation probability is modeled as a mixture of Gaussian distributions. The observation vector is denoted as $O = [O_1, \cdots, O_K]$ with $K$ to be the length of $O$. Its probability at state $j$ is computed as $b_j(O) = \sum_{t=1}^{M} c_{jt} \Omega(O, \mu_{jt}, \sigma_{jt})$, where $\Omega$ is a Gaussian with $\mu_{jt}$ as the mean vector and $\sigma_{jt}$ as the covariance matrix, $c_{jt}$ is the mixture factor and $M$ is the number of mixture components. At training, we have observation sequences $O = O_j, j = 1, \cdots, K$. The above parameters $[a_{ij}, \pi_i, c_{jt}, \mu_{jt}, \sigma_{jt}]$ are found to maximize the likelihood $P(O|\lambda)$. We use the Baum-Welch estimation process to train the HMM.

The decoding or matching process is radically different from conventional HMMs. In conventional HMMs, the actual state sequence is unknown, but the observation sequence is unique. However, in vision gesture application, we consider the observation sequence to be non-unique. In conventional HMMs, the input observation feature vector $O = [O_1, \cdots, O_K]$ is known for each frame and the likelihood $P(O|\lambda)$ can be computed using an iterative forward pass process. In our framework, however, we do not assume that we know the exact observation vector $O_k$ at each frame $k$. Instead,
we allow for multiple hypotheses about the observation. At frame $k$, we have the group sets $G^k = [G^k_1, \cdots , G^k_{c_k}]$, where each element in $G^k$ is one possible observation and $c_k$ denotes the total number of groups in frame $k$. We assume only one element in the observation set is the true observation. We do not decide upon the best group for each frame independently of the others. The entire sequence of group sets is used as the input. We will discuss the problem related to the optimal observation sequence and proposed 3 approaches to compute the matching score with such an input.

### 4.2.1 Maximal Observation, Summed State

We are given a sequence of group sets:

$$G = \langle G^1, \cdots , G^K \rangle,$$  \hspace{1em} (4.3)

where $G^k = [G^k_1, \cdots , G^k_{c_k}], 1 \leq k \leq K$ is the group set at frame $k$. The optimal observation sequence problem is to find one group sequence $\psi$ that maximizes the likelihood, summed over the possible HMM state transitions, $P_{\text{sum}}(\psi|\lambda)$, where $\lambda$ is the HMM and

$$\psi = \langle \psi_1, \cdots , \psi_K \rangle, \psi_i \in G^k, 1 \leq k \leq K, \psi_{k-1} \in \text{Pre}(\psi_k)$$  \hspace{1em} (4.4)

We denote the maximum value of likelihood probability by

$$P_{\text{max,sum}}(G|\lambda) = \max_{k=1,\cdots,\lambda_K} P_{\text{sum}}(\psi^k|\lambda)$$  \hspace{1em} (4.5)

where $\lambda_K$ is the number of all possible sequences of groups. The probability $P_{\text{sum}}(\psi^k|\lambda)$ represents the likelihood of the group sequence, summed over all the possible HMM
state sequences. For each sequence of groups, the computation of $P_{sum}(\psi^j|\lambda)$ can be done using the standard forward-backward algorithm used for HMMs.

A brute force solution for Eq. 4.5 will be to enumerate across the sets $G^1, \ldots, G^K$ to get all possible observation sequences $[\psi^1, \ldots, \psi^\lambda]$, compute the likelihood for each of the observation sequences, and select the maximum value. Obviously, exhaustive enumeration is computationally expensive. Hence we resort to approximation based on incremental construction of the optimal sequence.

To find the best group at frame $k$, suppose the observation sequence at frame $1, \ldots, k-1$ has been recovered as $\psi_1, \ldots, \psi_{k-1}$. We define the indexed forward variable $\alpha^j_k(i)$ as:

$$\alpha^j_k(i) = P(\psi_1, \ldots, \psi_k, q_k = i, \psi_k = G^k_j|\lambda)$$  (4.6)

That is, the probability of the partial observation sequence $<\psi_1, \ldots, \psi_k>$, at frame $k$ the state is $i$ and the observation vector is $G^k_j$, and $<\psi_1, \ldots, \psi_{k-1}>$ is the observation vectors we have found at time $1, \ldots, k-1$

The initialization of the variable is:

$$\alpha^j_1(i) = \pi_i b_i(G^1_j)$$  (4.7)

and we have

$$\psi_1 = G^1_p, p = \arg \max_j \sum_{i=1}^N \alpha^j_1(i)$$  (4.8)

The induction solution is

$$\alpha^j_{k+1}(i) = \left( \sum_{t=1}^N \alpha^p_t(a^j_t)b_t(G^{k+1}_j), \psi_k = G^k_p \right)$$  (4.9)
and then $\psi_{k+1}$ is selected as:

$$\psi_{k+1} = G_{p}^{k+1}, p = \arg \max_j \sum_{i=1}^{N} \alpha_{k+1}^j (i) \quad (4.10)$$

At frame $K$, the observation vector sequence is computed as $<\psi_1, \cdots, \psi_K>$. At the same time, the probability of this observation sequence given the HMM, can be computed as

$$P(<\psi_1, \cdots, \psi_K > | \lambda) = \max_j \sum_{i=1}^{N} \alpha_{K}^j (i) \quad (4.11)$$

Fig. 4.4 illustrates for us the indexed forward process. The summation of the product of the forward variables and the observation probabilities remain the same as in conventional HMMs. The difference is that we take the observation vector dynamically based on the previously decided observations, while a traditional HMM has a static fixed observation vector. Note the result of Eq. 4.11 is not an exact solution for Eq. 4.5. Instead, it is the solution to select the best current observation based on a certain selected partial observation sequence.

### 4.2.2 Summed Observation, Summed State

Instead of considering the maximum probability over all possible group sequences, we could consider the summation over all possible group sequences. Thus, the probability of interest is.

$$P_{\text{sum,sum}}(G|\lambda) = \sum_{k=1, \cdots, \lambda_K} P_{\text{sum}}(\psi^k|\lambda) \quad (4.12)$$

where the possible sequence of groups are $\psi^1, \cdots, \psi^{\lambda_K}$. The probability $P_{\text{sum}}(\psi|\lambda)$ represents the likelihood of the group sequence, summed over all the possible HMM state sequences. As before, for each sequence of groups, the computation of $P_{\text{sum}}(\psi^i|\lambda)$
Figure 4.4 Illustration of the indexed forward process. The horizontal line represent the time, the vertical line correspond to the candidate observations and the sub-vertical line denotes the $N$ states. Note at each time step, only one best observation is selected based on the previous selected observations and the forwarding results. In this example, the optimally selected observations (circled ones) are $<1, 2, 2, 3>$. 
can be done using the standard forward-backward algorithm used for HMMs. How-
however, we found the process of summing over all group sequences and over all state
sequences can be effectively merged in the dynamic programming process. To do this,
we define the grouping forward variable $\kappa^j_k(i)$ as:

$$
\kappa^j_k(i) = \sum_{\psi_1, \cdots, \psi_{k-1}} P(\psi_1, \cdots, \psi_k, q_k = i, \psi_k = G^k_j|\lambda) \tag{4.13}
$$

That is, the summation of the partial probability of all the group sequences that have
$\psi_k = G^k_j$ and $q_k = i$. The initialization is

$$
\kappa^j_1(i) = \pi_i b_i(G^1_j) \tag{4.14}
$$

The induction is

$$
\kappa^j_{k+1}(i) = \left( \sum_{p \in \text{Pre}(G^k_j)} \sum_{t=1}^{N} \kappa^p_k(t)a^t_i b_i(G^{k+1}_j) \right) \tag{4.15}
$$

And the result of Eq. 4.12 is obtained at the end of the process:

$$
P_{\text{sum, sum}}(G|\lambda) = \sum_{p \in \text{Pre}(O^k_j)} \sum_{t=1}^{N} \kappa^p_K(t) \tag{4.16}
$$

### 4.2.3 Maximal Observation, Maximal State

The third quantity of interest is maximum probability over all the possible group
sequences and HMM state sequences. Thus, the probability of interest is.

$$
P_{\text{max, max}}(G|\lambda) = 
\max_{\psi_1, \cdots, \psi_K} \max_{q_1, \cdots, q_K} P(\psi_1, \cdots, \psi_K, q_1, \cdots, q_K|\lambda) \tag{4.17}
$$
where the possible sequence of groups are $\psi^1, \cdots, \psi^K$ and $q_1, \cdots, q_K$ is a HMM state sequence. This quantity can again be computed using dynamic programming. We define the max-forward variable $\zeta^j_k(i)$ as:

$$
\zeta^j_k(i) = \max_{\psi_1, \cdots, \psi_{k-1}} P(\psi_1, \cdots, \psi_k, q_k = i, \psi_k = G^k_j | \lambda) \quad (4.18)
$$

This is the maximum partial probability among all the group sequences that have $\psi_k = G^k_j$ and $q_k = i$. The variable $\xi_k$ represents the backtrack index of the observations for the corresponding max-backward process. The initialization is:

$$
\zeta^j_1(i) = \pi_i b_i(G^1_j) \quad (4.19a)
$$

$$
\xi_1 = 0 \quad (4.19b)
$$

The induction is given by

$$
\zeta^j_{k+1}(i) = \left[ \max_{p \in \text{Pre}(G^k_j)} \max_{t=1}^N \zeta^p_k(t) a^i_t \right] b_i(G^j_{k+1}) \quad (4.20a)
$$

$$
\xi_k = \arg \max_{p \in \text{Pre}(G^k_j)} \max_{t=1}^N \zeta^p_k(t) a^i_t \quad (4.20b)
$$

$\xi_1, \xi_2, \ldots, \xi_K$ is obtained as the best group sequence (over the best state sequence) and this group sequence can be used to get the matching score.
CHAPTER 5
FEATURE REPRESENTATION

We have discussed both the high level matchings and low level matchings with multiple candidates in the previous two chapters, where we strive to solve the Eq. 3.2. In this chapter, we will discuss the different approaches we take to generate the multiple observation feature vectors to be fed into the low level matching framework. In this chapter, we will also answer the second part of our research question 4,

4. How can we handle imperfect segmentation at the low level? How can one use feature grouping processes to overcome segmentation errors?

5.1 Low Level Representation

In this section, we describe our low level processes. Many of the modules used are fairly standard ones, except for the background modeling scheme. Hence, we have placed this section after describing our core contributions, which is in the matching process. To segment the hands automatically, we use skin color and motion. After segmenting the hands, we will construct two kinds of features vectors: a global feature vector and a part-based feature vector. We will experiment with both these feature types in our experiments in head-to-head comparisons and also demonstrate that the matching method outlined in this work can be used in conjunction with different feature types.
5.1.1 Detection of Hands

The assumption that we make is that the hands move faster than other objects in the scene (including the face), and that the hand area can be somewhat localized by skin color detection. We use the mixed Gaussian model of Jones et al. [72], with a safe threshold allowing for some amount of non-skin pixels to be falsely classified as skin pixels.

To segment based on motion, we employ a key-frame-based background model. We represent the possibly changing (but slowly) background, using a set of key frames. These key frames are identified as frames that are sufficiently different from each other. We choose the first frame as one key frame and then sequentially search for the rest of the key background frames. We compute the difference of any frame with the previous key frame. If the non-component size in the thresholded difference image is large, then the frame is labeled as the next key frame. This process continues until the end of the sequence. Then we compute the difference image of each frame to all the key frames. This distance is thresholded and post processed using morphological operations.

The specifics of the approach are outlined below and some illustrative results are shown in Fig. 5.1, where Step 2 (e) generates the motion-skin confidence map. Step 2 (f) generates its boundary.

For each sentence $T$ with $N$ frames

1. Assign first key frame $k_1 = 1$, and initialize key frame counter $m = 1$. For frame $i = 2, \cdots, N$

   (a) Compute difference image between $T(i)$ and $T(k_m)$. Find the largest connected component in the difference image in terms of its number of valid pixels $N_p$. 

46
Figure 5.1 Intermediate results for the process of hand segmentation. (a) One frame in a sequence. (b) Consecutive frame difference image. (c) Skin pixels found. (d) Frame difference image with key frames. (e) Edges found in (d). (f) After dilating (e). (g) After an AND operation with the mask in (f) with (d). (h) After removing small components in (g). (i) Boundary of the component in (h), which is the final hand.
(b) If \( N_p > \text{threshold}(T_0) \), set \( m = m + 1 \), set \( k_m = i \).

(c) Set \( i = i + 1 \). If \( i > N \) go to next step, else repeat above steps.

2. For frames \( i = 1, \cdots, N \), repeat

   (a) Compute a difference image \( SD \), where \( SD = (\sum_{j=1}^{m} |S(i) - S(k_j)|)/(m-1) \).

   (b) Mask \( SD \) with the skin likelihood image. Do edge detection on \( SD \) and obtain the edge image \( E \).

   (c) Apply a dilation filter to \( E \).

   (d) For each valid pixel in \( E \), set the corresponding pixel of \( SD \) to be 0.

   (e) Remove the small connected components in \( SD \). This step generates the motion-skin confidence map.

   (f) Extract the boundary image \( B \).

We have found the use of key-frame-based background subtraction to be more effective than using all the frames to estimate the background, at least for our kinds of sequences. Fig. 5.2 shows one illustration of result we get when the key frames are not used. Instead, all the frames are used when \( SD \) is computed. Some features are blurred when there is slow motion or repeated motion in a sign. On the other hand, with the key frame approach in Fig. 5.3, we locate the hands with stronger confidence.

5.1.2 Global Features

We first generate the feature vectors using the boundary motion-skin confidence map obtained above (in Step 2 (f)). Given the hand boundaries, we capture the global spatial structure by considering the distribution (histogram) of the horizontal
Figure 5.2 Hand segmentation results using all frames. Hand segmentation results obtained with the full sequence, where each frame is treated as a key frame. We lost some features when the hand moves slowly.
Figure 5.3 Hand segmentation results using key background frames. We can see at Step 8 we have stronger confidence about where the hand is (brighter value over hand pixels). (SD is the cumulative difference with the background.)
and vertical distances between pairs of pixels in it. We compute the joint relational histogram of the displacement between all pairs of coordinates on boundary images. We then represent these relational histograms, normalized to sum to one, as points in a space of probability functions (SoPF), like that used in [73]. The SoPF is constructed by performing a principal component analysis of these relational histograms from the model images. The coordinates in the SoPF is the feature vector used in the matching process. We use the Mahalanobis distance as the distance measure.

5.1.3 Multiple Candidates Representation

For cases with controlled background and clothing, as is the case with most sign language databases, the hand detection method outlined performs reasonably well. However, under uncontrolled cases where we can have nuisance motion-skin blobs in the background, or if the signer is wearing a short sleeved shirt, or even in the case where the signer’s head (face) moves a lot, our approach (and most hand detection algorithms) will generate lots of false alarms. To handle such cases, we generate multiple hand candidates and then select among them during the matching process as outlined earlier.

To construct the multiple candidates, we first represent the motion-skin confidence map as a collection of connected components. All connected components that are compact and small are selected to be hand candidates. The compactness is measured by dividing the number of pixels by the number of boundary pixels with a threshold $T_1$. The size is measured by the number of pixels with threshold $T_2$. The remaining components that are too large to be the hand can still arise from the merging of the arm with the hands. The hands in these cases would most likely be at the boundary of these shapes. To find these, we compute their medial-axis by iteratively removing each boundary pixel that will not disconnect the connected component.
Figure 5.4 The confidence map and the generated candidate hands. We can see that the candidate hands can be generated correctly when the background is moving and the signer wears short sleeved clothes.

Then, we concentrate on all the leaf pixels on the medial-axis. These leaf pixels are then clustered using a nearest-location-neighbor clustering method with respect to a threshold $T_3$ until we get regions that are small enough to be hands. Fig. 5.4 shows us some results for some sample frames in our 3 different continuous ASL data sets.

5.2 Grouping of Low Level Primitives

The above algorithm can generate multiple hand candidate pairs. However, the shape information of the hands is still not clear. We provide another alternative method to generate multiple hand candidates, based on a grouping process. This method can provide detailed shape information, but it currently can only work for
a single hand, not a hand pair. We will experiment with this method using single sign/gesture recognition instead of continuous recognition.

Low level processes are never perfect. Skin color is the most commonly used cue for segmenting image parts from the hand or face in gesture analysis. However, this does not always produce perfect segmentation, with over segmentation being a particularly hard problem to handle. In our work, we allow for overlapping groups, resulting in redundant sets of groups. This answers the second part of our research question 4, where we can use over segmentation with grouping approach to generate redundant observations, this can reduce the risk that we will lose the true hand observation at the feature extraction step.

Our approach is depicted in Fig. 1.4. We use a top-down recognition process and bottom-up grouping process, integrated in a dynamic programming framework. First, we segment the image into a collection of non-overlapped regions. These non-overlapped regions are our grouping primitives. Some of these primitives are selected as our seed patches. Then, we use a greedy-search-based grouping approach to generate groups representing possible hands. We start from the seed patches, progressively adding new adjacent primitives, followed by checking to prune out the bad groups. We generate layers of groups, with each layer based on one attribute such as color, or proximity, or boundary gradient. The generated groups are not disjointed. Notice the color attributes are used as a similarity measure instead of a predefined model. The generated groups are then linked across adjacent frames to generate a set of candidate group sequences. Finally, we match each model sequence to the linked group structure. We find the best match and simultaneously a matching score between the model sequence and the input sequence. We have shown this matching can be conducted for both deterministic and statistic models. Based on the matching score, we use a
Figure 5.5 The proposed HMM model. For HMM, we do not have a unique observation sequence to match. Rather, we have a collection of possible observation sequences, implied by the sequence of multiple observations at each frame.

simple nearest neighbor rule to get the recognition result. By using this approach, we significantly reduce the need for perfect segmentation at the first step.

The models we use in our system include both statistical models (HMMs) and deterministic models. For the deterministic model, we simply store sequences of training signs in the database and match them with time warping techniques, which is essentially a dynamic programming process. A similar matching process can be seen at [21], with multiple observations, but no grouping. For HMMs, we show the structure in Fig. 5.5. We match each gesture HMM to the linked group structure to simultaneously compute the matching score and the best possible grouping for each frame. We later will show three different ways to actually conduct the matching, like those shown in [22].

5.2.1 Grouping Process

The low level primitives of the grouping process are constant color (or intensity for gray level images) region patches. We use the mean shift segmentation algorithm [74],
which is fast and effective, to generate these patches based on color or intensity. Let the set of low level primitives detected in the $k$th image frame be denoted by $S_k = \{p_{k1}^1, \ldots, p_{kN_k}^1\}$. A grouping, $G_k^i$ of these region primitives, will represent a subset of these primitives, $\{p_{k1}^i, \ldots, p_{kn}^i\}$.

We adopt a greedy approach to form the groups, outlined in the flow chart in Fig. 5.6. From the initial set of primitives $S_k$, we select a subset of primitives that are likely to come from a hand, based on the size of the patch. These are our seed patches. Given some knowledge of the approximate size of hands in the sequence, we can eliminate large, non-homogeneous region patches from further consideration. We use a list $L$ to store the possible groups. This list is initialized by choosing each selected primitive to be a singleton group. These groups would be merged to form a
Figure 5.7 Illustration of local adjacency graph. (a) An image frame. (b) Homogeneous region patches based on just intensity. (c) Local adjacency graph over the small region patches, which correspond to the hand. Each primitive patch is represented by a node. Links denote pairs of primitives that are adjacent to each other.

\[ L = \{ \{ p^k_x \} | a_s(p^k_x) \leq t_{\text{size}}, x = 1, \ldots, N_k \} \]  \hspace{1cm} (5.1)

Here \( a_s \) is the operator that returns the size of \( p^k_x \). For the entries in \( L \), we maintain an adjacency graph, whose nodes are the groups in \( L \), and links exist between groups that share a boundary. This graph is incrementally updated at each iteration. Fig. 5.7 shows us an example local adjacency graph (Look ahead to Fig. 5.8 for a sequence of iterations of this graph.).

The grouping process starts by picking the first group in \( L \), denoted here by \( p \), and searches its neighbors \( \{ N^i_p \} \). Each neighbor \( N^i_p \) is considered for grouping with \( p \) to generate a tentative larger grouping. We select the best local grouping, and denote it as \( g \). In color layer, the best neighbor is the one that has the smallest Euclidean distance with the base group in the RGB space. In the proximity layer, we choose the neighbor that is nearest to the base group according to the image coordinates of...
their centers. In the boundary layer, the neighbor that yields the smallest curvature score when grouping with the base group is selected as the best.

The group \( g \) is further tested to see if it can possibly represent a hand. This test is based on three attributes: \([a_n, a_s, a_{\text{cur}}]\), where \( a_n \) is the number of primitives in the group, \( a_{\text{cur}} \) is the boundary curvature of the group, \( a_s \) is the size of the bounding box.

\[
(a_s \leq t_{\text{size}}) \land (a_{\text{cur}} \leq t_{\text{curvature}}) \land (a_n \leq t_{\text{num}})
\] (5.2)

The test is conducted based on the result of Eq. 5.2, where \( t_{\text{size}}, t_{\text{curvature}}, t_{\text{num}} \) are the corresponding thresholds. Here, the boundary curvature is approximated as the integral of the squared root of second order derivative along the curve. If the group \( g \) passes this test, it is inserted into the final candidate group list, \( C \), else if \( a_s \leq t_{\text{size}} \) it is inserted at the end of the list \( L \), to be considered for further grouping.

Fig. 5.8 shows us the grouping process based on the adjacency graph, where the mechanism is essentially a greedy search process to the adjacency graph, starting from a chosen seed. In Fig. 5.8, a solid link represents a grouping between two nodes. Starting from the seed patch \( S \), a decision is made to group \( S \) with the best neighbor, denoted by \( N_c \) according to a layer \( c \), and generate the new group \( G_c \). After grouping, the adjacent graph is updated, where the new neighborhood will be the neighborhood of \( G_c \), and the process starts again to group one of \( G_c \)’s neighborhoods with \( G_c \), based on the same layer criteria. After detecting the seed primitives, the above process is used to generate 3 grouping layers based on color, position, and boundary gradient. This process reduces the possibility that the group corresponding to the hand will not be generated. On the downside, this step will triple the time and space complexity.

Note that the low level primitives and the groups are formed on a frame by frame basis. There is no tracking or frame to frame correspondence. Fig. 5.9 shows us the
Figure 5.8 Example of the generation process of groups. The process is repeated for each primitive as a seed.
Figure 5.9 Example of generated multiple candidates. (a) Original frame 1. (b) Segmented frame 1. (c) List of candidate groups in the color grouping layer in frame 1. (d) List of candidate groups in the proximity grouping layer in frame 1. (e) Original frame 2. (f) Segmented frame 2. (g) List of candidate groups in the color grouping layer in frame 2. (h) List of candidate groups in the proximity grouping layer in frame 2. (i) Original frame 3. (j) Segmented frame 3. (k) List of candidate groups in the color grouping layer in frame 3. (l) List of candidate groups in the proximity grouping layer in frame 3. (True hand is shown with white circle.)
grouping results for 3 different frames at the color grouping layer and the proximity grouping layer. For frame 1 in Fig. 5.9 (a), the color grouping layer at Fig. 5.9 (c) includes the real hand group (shown with a white circle), while the proximity grouping layer at Fig. 5.9 (d) failed to include it. For frame 2, however, proximity grouping layer gives us the true hand, while the color grouping layer does not. For frame 3, both of the color grouping layer and the proximity grouping layer have the true hand group in their list.

Also, we can see in Fig. 5.9 how the groups differ from each other in terms of missing fingers or added extraneous regions. This can confound the sign recognition process. Also note that we do not restrict ourselves to disjoint groups. Thus, we might have $G^k_i \cap G^k_j \neq \text{NULL}$. This is different from the usually employed disjoint groups constraint employed in segmentation and grouping. Allowing for overlapping groups allows us to avoid making hard decisions about group boundaries.

5.2.2 Associating Groups Across Frames

We denote the $j$th group detected in $k$th frame as $G^k_j$. The groups detected in each frame are associated with those detected in previous frames to result in a linked sequence of groups spanning all the frames. This structure will help us propagate constraints during the matching process and reduce the number of possible observation sequences to be searched. We define the predecessor’s set of each element in each group’s set as

$$Pre(G^k_j) = [G^{k-1}_{j_1}, \cdots, G^{k-1}_{j_n}], \quad (5.3)$$

where $G^{k-1}_{j_k}$ is one possible predecessor of $G^k_j$. The predecessor relationship between the groups from different time is based on feature similarity. It captures how likely the groups are from the same underlying cause in the image. Specifically, we test the
difference in location between the two groups, with a liberally chosen threshold value:

\[ \text{Distance}(G^j_g, G^j_{g-1}) \leq T_4, \; r \in \text{Pre}(G^j_g) \]  

(5.4)
CHAPTER 6
CONDITIONAL MODELS

Conditional random fields (CRF) has been considered as a popular method for modeling and labeling various kinds of sequences, including gesture sequences. CRF strives to model the posterior probability directly with one global representative function. In this chapter, we will show our proposed modifications on CRF. We will also show CRF results compared to our eLB methods in Chapter 7.

6.1 Conditional Random Fields for a Sign/Gesture Sequence

Unlike a Hidden Markov Model (HMM) that is a generative model based on likelihoods of observations, conditioned on states, and prior probabilities of states, CRF is a discriminative model that directly computes the posterior state probabilities. The HMM requires strict independence assumptions across multivariate features and conditional independence between observations, given the states. However, these independence assumptions are generally violated in sign languages, i.e., observations are not only dependent on the state but also on the past observations. The other disadvantage of using HMMs is that the estimation of the observation parameters requires a large amount of training data. This is a problem because it makes the training more difficult. If any condition of the system is changed, retraining the model will be harder.

Fig. 6.1 depicts the essential differences between HMMs and CRF. Fig. 6.1 (a) shows the structure of HMMs, where the directed links indicate the conditional likeli-
Figure 6.1 Difference of CRF, HMM and key frame CRF. (a) HMM defined with state and observation pairs using directed links. Multiple consecutive observations in any given sequence can be mapped onto the same state. (b) The CRF model uses pairwise probabilities over states and observations for each time instant. Each observation is associated with a state label. (c) Key frame CRF.

...
CRF has been used successfully by [75] to label and segment text sequential data. Recently, Sminchisescu et al. [41] used CRF to recognize whole body human movement, not sign language. They reported CRF outperformed the HMM, especially under large context dependent situations. However, the movements considered by them are basically consecutive performances of single gestures with no me effects. Also, unlike their approach, we do not use CRF for recognition, but rather for segmentation.

6.2 Key Frames Representation and Extraction

In our test for key-frame-based CRF, before the training and testing are conducted, the video frames are preprocessed by a local corner point tracker and then a key frames detector. We use motion snapshots to represent the frame based on the tracker result, which is simple and robust compared to the use of external devices or skin color blobs. Then the key frames subsets within each sentence are detected, by using a matrix formulation of the frame distances and the eigen vector, to indicate the best key frame subsets.

6.2.1 Motion Snapshot Representation

The low level image processing in hand gesture recognition, such as feature tracking and region segmentation, can be facilitated by using external devices. Nevertheless, the real world application, requiring signers to wear gloves or tracking markers on their hands while signing could be annoying and inconvenient in the real application. Skin color blobs are widely used to extract the moving hand in simple gesture recognition tasks. However, in ASL, the hand movements are more complex, which can result in situations where there are shadows on the hand, 2 hands are crossing each other, and a hand crosses the face, etc., where skin detection may generate am-
biguous blobs. Unlike these approaches, we take the plain 2D color video sequence as input, which consists of the sign sentences of American Sign Language. The low level processing is simply conducted by corner detection, feature correspondence, and construction of a motion snapshot within a small number of frames.

Specifically, for each frame in the input 2D video sequence, we considered those good feature points and their mappings to both the previous and next frame. The classic KLT (Kanade-Lucas-Tomasi) method is used to detect corner points and then we use the pyramidal implementation of the Lucas Kanade Tracker to estimate the motion of the corner points between adjacent images. The corresponding feature points are concatenated using the Bresenham Line-Drawing algorithm to form a motion trajectory map. And we only concatenate those features in the current frame that can find a good correspondence in both the previous and next frame. Hence, the tracking essentially exists between the neighbor frames only. We refer to this representation as motion snapshot.

With the obtained motion snapshot, we examine two pairs of relational features inside it, which are the horizontal distance and vertical distance among all the valid pixels. A joint 2D histogram is formed with regard to the two features. Then, the Principal Component Analysis (PCA) is applied to form the dominant vector of the sign frame space. Each frame is then projected to the obtained eigen space. We refer to [73] for the details of the method. After this step, the ASL sentence is represented as a sequence of feature vectors $S = < S_1, S_2, ..., S_N >$.

### 6.2.2 Detecting Key Frames

A number of key frame and video boundary detection technologies have been proposed earlier. For example, Zhong et al. [76] used an unsupervised approach to detect unusual events in a long video, where a graph is constructed, each small chunk
Figure 6.2 Illustration of relational distribution representation. The ASL sign “CAN” consist of 3 frames which are in the first column. The second column is the tracked result for the local motion trajectory. The third column is the 3D mesh visualizing the relational distribution for each frame.
of video is represented as a node and the edge weights are associated with frame difference. We also use similar graph representation and eigen vector computations, but the detection is for per frame and it is conducted under a semi supervised way.

In our approach, we define key frames of either a sign or the me to be those frames that are the most different from the frames of other signs or the me. The training set for key frame selection is a set of individual signs that are manually segmented with the me portion removed. We denote the training set as \( T = \{ t_1, t_2, ..., t_l \} \) where \( t_i = < t^1_i, t^2_i, ..., t^{l_i}_i > \), \( l \) is the size of the training set and \( l_i \) is the length of the \( i \)th training signs. Formally, given a sentence \( S = < S_1, S_2, ..., S_N > \) with \( N \) frames, we denote the key frames sequence \( K = < S_{k_1}, S_{k_2}, ..., S_{k_m} > \) as a subsequence of \( S \) where \( k_i \in \{ 1, 2, ..., N \}, i = 1, 2, ..., m \) and \( k_1 < k_2 < ... < k_m \). We define within-sign distance of each frame as its average distance from other frames within the sentence. We also define between-sign distance of each frame as its average distance from all the frames in the training set. We select \( K \) as the most coherent set, which maximizes the sum of the within-sign distance and between-sign distance. We define the adjacency matrix \( A \) for one sentence as:

\[
A_{i,j} = \frac{diff(S_i, S_j)}{k_m}, i \neq j
\]

\[
A_{i,i} = \sum_{p=1}^{l} \sum_{q=1}^{l_i} \frac{diff(S_i, t^p_{q_i})}{N_t}
\]

(6.1)

where \( diff \) is the operator to compute Euclidean distance and \( N_t \) is the number of total frames in the training sign set. The diagonal member of \( A \) represents the between-sign distance while the ones which are not the diagonal members are the within-sign distance. If we consider \( \xi = \Sigma A x_i x_j \) as our object function where \( x_i \) and \( x_j \) represent the participant scores for the \( i_{th} \) and \( j_{th} \) frame to the chosen subsets of the graph. The participant set that maximize \( \xi \) will also maximize the sum of within-sign distance and between-sign distance, which gives us the desired set for key
frames. Specifically, the first eigen vector (the eigen vector with the largest eigen value) of $A$ denotes the participation of each frame to the most coherent cluster in $S$. We refer to [77] for the details of this method. Suppose the first eigen vector is obtained as $E$. We find all the local minimals of $E$ w.r.t a small window and the corresponding frame is selected as one key frame.

6.3 Conditional Random Fields over Key Frame Sequences

We select key frames for each sentence in the training data set. The key frames are manually labeled as a sign or me. We use a linear chain model of CRF, where the observations are denoted as $K = < K_1, K_2, ..., K_t >$ and the corresponding labels are $L = < L_1, L_2, ..., L_t >$ and $L_i \in \{SIGN, me\}$. $< L, K >$ is a conditional random field if when globally conditioned on $K$, $L$ obeys the Markov rule in the linear graph. That is:

$$P(L_i|K, L - \{L_i\}) = P(L_i|K, N(L_i)) \quad (6.2)$$

where $N(L_i)$ is the neighbors of $L_i$. Let us consider the linear chain graph $G$ constructed by $< K, L >$, let $C(K,L)$ denote the set of cliques in $G$. By the fundamental theorem of random fields, the probability of a label sequence $L$, given the observation sequence $K$, can be represented as:

$$P(L|K) \propto \exp \sum_{c \in C(K,L)} f_\theta F_\theta(c,K) \quad (6.3)$$

where $\{F_\theta\}$ are the feature functions defined over all the cliques and $f = \{f_\theta\}$ are the parameters set weighted the corresponding feature functions. In a linear chain graph, the cliques can be the pair of adjacent labels $< L_{t-1}, L_t >$ and the pair of label-observation pair $< L_t, K_t >$. For example, at the startup of an ASL sentence, usually
the signer lifts the hands up. Let us denote the key frame of this action as $K_0$ and the corresponding label as me. Then a penalty of assigning me to $K_0$ is selected and then weighted by the corresponding $f_\theta$. For a transition feature, similarly, suppose we have 2 adjacent key frames $K_0$ and $K_1$ which are labeled both as me, then a penalty of assigning me – me to an edge is selected and weighted by corresponding $f_\theta$.

Note that unlike HMMs, where strict independence does not allow us to represent the relationship between the labels and observations in different time, in CRF this can be represented with an arbitrary window $w$ as $< L_t, K_{t\pm w} >$, which can be more context dependent and is much more flexible. For training, we considered the labeled ASL key frame sequences $< L_d, K_d >, d \in 1, 2, ..., N_s$ where $N_s$ is the size of the training database. The parameter set $f$ can be found by maximizing the log likelihood:

$$L_f = \sum_{d=1}^{N_s} \log(P(L_d|K_d))$$

$$= \sum_{d=1}^{N_s} \sum_{c \in C(K,L)} (f_\theta F_\theta(c,K)) - \log(Z_\theta(K))$$  \hspace{1cm} (6.4)

where $Z_\theta$ is the normalization factor depending on the observation sequences. We use a gradient-based approach with a random start point to seek the maximal point of 6.4. A belief propagation (BP) method is used to do inference over the chain structure. The inference result is our decoded sequence for the sign language sequence.
We have conducted extensive experimentation of the approaches proposed in this work in the context of the task of recognizing continuous American Sign Language (ASL) sentences and single gestures/signs from image sequences. We present not only visual results of labeling continuous ASL sentences, but also quantify their performance.

For continuous ASL sentence experiments, we compare the performance with that obtained by classical Level Building, which does not account for movement epenthesis, and the frame labeling results obtained from two state of the art methods: conditional random fields (CRF) and Latent Dynamic-CRF (LDCRF). We were not able to compare with other explicit model-based approaches to handling movement epenthesis and some generative methods such as the HMM, since they require large amount of training data, which is either not available or difficult to acquire. For the vocabulary size used in this work, we would need about 1000 labeled ASL sentences. We also present empirical evidence of the optimity of the choice of the $\alpha$ parameter that is used to decide on the ending mapping cost and present the impact of the grammar model on recognition.

For single sign/gesture recognition, we experiment with both deterministic sample-based models (DTW) and statistical models (HMMs) with our proposed decoding processes. We tested the grouping algorithm coupling with both of the two models. We show the results compared to the methods without a grouping approach and the
methods that use manually selected groups. We also show the tracking results of our true groups as byproducts of our decoding algorithm.

In this chapter, we will answer the following research questions:

5. Can our proposed set of algorithms handle complex background? Can we identify signs made by signers wearing both short and long sleeves, i.e., relax the typical clothing constraints?

6. How well does the recognition rate with the proposed approach match with that achieved through manually grouped segmentation?

7.1 Data Sets and Experiment Setup

We have used 4 data sets, summarized in Table 7.1. Example frames from these three data sets are shown in Fig. 1.6, 1.7, 1.8 and 1.9. As we can see, the data sets vary in terms of the background. The background in data set $D_1$ is uniform, static, and with no texture. This is typical of sign language data sets. The background in
$D_3$ is static, but it is textured. The lighting in this data set is not directly on the subject. This data set is harder in terms of illumination and background conditions than $D_1$. This data set is not typical of sign language data sets, especially in the use of short sleeves. The data set $D_2$ is the toughest one, with complex background and with moving people in the background. There are several patches in the background with skin color. $D_4$ is the data set for isolated gesture sequences [78], which has complex but static background, and it has two views. For each frame in data sets $D_2$, $D_3$ and $D_4$, we have multiple hand candidates. Only for $D_1$ can we use global features.

The train and test for these data sets are structured as follows. In $D_1$, we have 5 samples per sentence. We perform 5-fold cross validation experiments, with 4 samples of each sentence for training and one for test. For $D_2$ and $D_3$, we have different sentences in the training and testing set. Methods that explicitly or implicitly rely on me models will have a hard time.

The gesture data set $D_4$ in our experiments is a 7 hand gesture data set. The data set consists of 280 training sequences, 40 for each gesture and 210 test sequences from 3 subjects, and 10 for each subject and each gesture. This data set has two views. Since we have enough training data on $D_4$, we will show results based on the Hidden Markov Model. $D_4$ has 24 fps, complex background, and colored gloves with long sleeves.

To enable us to quantify the performance, we manually labeled the frames corresponding to the signs in the sentences. We also used the tool in [24] to manually generate the true hands groups for the model signs. We also refer the reader to Appendix A for the process of data collections and Appendix B for the annotation process. To quantitatively evaluate the results, we use error measures as advocated in [79]. If the recognized sentence inserts a sign that does not actually exist, one insertion error
is counted. If, however, the recognized sentence omits a sign where it actually exists, one *deletion error* is counted. If the recognized sentence reports a wrong sign, we consider it as a *substitution error*. We computed these errors automatically by computing the Levenshtein distance using a dynamic programming approach [80] between the actual results and manually labeled groundtruth. We name this measurement to be "word level rate". We also evaluate the frame wise labeling result, which means the total number of correctly labeled frames divided by the total number of frames. We call this measurement the "frame level rate".

We use the same set of thresholds for all the experiments. We set these thresholds to be a liberal value based on heuristics. For example, we set $T_0 = 100$ (pixels), $T_1 = 2$, $T_2 = 4000$ (pixels), $T_3 = \text{imageheight}/8$, $T_4 = 300$ (pixels).

While high recognition rates (in the order of $>90\%$) of isolated ASL signs and isolated finger spelled signs have been reported, reported performances for recognition in continuous sentences vary quite a bit ($58\% - 99\%$ [81]), depending on vocabulary size, and length of sentences, and possibly other factors yet to be explored, such as the degree to which humans can recognize each sign under various conditions such as complex background, etc.

We conducted six studies. In the first study, we focused on the analysis of the eLB algorithm and the estimation of parameter $\alpha$. We used global features in this study. We tested using both bigram and trigram grammar built using a text corpus of 150 sentences. The entire text corpus is shown in Appendix C. The performance was measured using the word level rate. In the second study, we compared our labeling approach with CRF/LDCRF approaches. Since CRF/LDCRF only produce a frame level rate result, we used this as performance measure for this study. In the third study, we compared the results between global features with the part-based candidate hands approach, we experimented on both $D_1$ and $D_2$. In this study, we
used a sentence-based grammar, which is stronger than just bigrams and trigrams. In the fourth study, we used $D_1$, $D_2$, and $D_3$. The eLB and part-based candidate hands are used. We show the results with variation in the algorithm used to detect hands. In the fifth study, we conducted experiments with $D_1$, where we show the grouping algorithm coupling with a deterministic sample-based model. In the sixth study, we conducted experiments with $D_4$ using the grouping algorithm coupling with three different HMM decoding processes. The details of the setup of the experiments are listed in Table 7.2 for study 1 − study 4. We also show the experiment setup of study 5 and study 6 in Table 7.3. These experiments consist of single sign analysis and the grouping analysis.

For eLB setup, we assigned the parameters values as $L_{\text{max}} = 20$ and $N_{\text{max}} = 145$, which means we allowed one sentence to have a maximum of 20 signs, and the maximum duration of movement epenthesis $\text{me}$ to be 145 frames. We used the first 7 coefficients of the Space of Probability Functions (SoPF) space representation as the global feature vector [73]. In our experiments, we have found these choices to be stable. Varying them did not change the performance significantly (within 1%).

7.2 Study 1: eLB vs. LB with Grammar and Parameter Analysis

The primary focus of this set of experiments is to test the effectiveness of the eLB algorithm to overcome the $\text{me}$ problem. We also studied the choice of the $\text{me}$ labeling cost $\alpha$. We conducted studies using the data set $D_1$, where background related issues are least likely to confound the movement epenthesis recognition problem.

The labeling results for three sentences are presented in Fig. 7.1. Each horizontal bar represents a sentence, and is partitioned into signs or $\text{me}$ blocks. The size of each block is proportional to the number of frames corresponding to that label. For each sentence, we present the groundtruth as determined by an ASL expert and the
Table 7.2 Outline of study 1 – study 4. The table shows the different matching, feature, grammar, and error measurements used in our four tests for continuous ASL sentence test.

<table>
<thead>
<tr>
<th>Name</th>
<th>Study 1</th>
<th>Study 2</th>
<th>Study 3</th>
<th>Study 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Analysis of eLB and α</td>
<td>Comparison between eLB and CRF</td>
<td>Comparison between eLB using global features and eLB using part-based candidates</td>
<td>Comparison between using skeleton and without using skeleton</td>
</tr>
<tr>
<td>Data sets used</td>
<td>$D_1$</td>
<td>$D_1$</td>
<td>$D_1$ and $D_2$</td>
<td>$D_1$,$D_2$ and $D_3$</td>
</tr>
<tr>
<td>Matching algorithms</td>
<td>eLB and LB</td>
<td>eLB, CRF and LD-CRF</td>
<td>eLB</td>
<td>eLB</td>
</tr>
<tr>
<td>Features</td>
<td>Global</td>
<td>Global</td>
<td>Part-based candidates and global</td>
<td>Part-based candidates</td>
</tr>
<tr>
<td>Grammar</td>
<td>Bigram and trigram</td>
<td>Trigram</td>
<td>Sentence</td>
<td>Sentence</td>
</tr>
<tr>
<td>Text corpus</td>
<td>Extended (150 sentences)</td>
<td>Extended (150 sentences)</td>
<td>Non-extended (same number as the test sentences)</td>
<td>Non-extended (same number as the test sentences)</td>
</tr>
<tr>
<td>Error measurements</td>
<td>Word level rate</td>
<td>Frame level rate</td>
<td>Word level rate</td>
<td>Word level rate</td>
</tr>
</tbody>
</table>
Table 7.3 Outline of study 5 and study 6. The table shows the different matching, feature, grammar, and error measurements used in our two tests for the single sign/gesture test data set.

<table>
<thead>
<tr>
<th>Name</th>
<th>Study 5</th>
<th>Study 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Analysis of grouping using deterministic sample-based model</td>
<td>Analysis of grouping using Hidden Markov Models</td>
</tr>
<tr>
<td>Data sets used</td>
<td>$D_1$</td>
<td>$D_1$</td>
</tr>
<tr>
<td>Matching algorithms</td>
<td>3D DTW</td>
<td>HMMs with max-max, max-sum, sum-sum approaches</td>
</tr>
<tr>
<td>Features</td>
<td>Grouping results, automatic and manual</td>
<td>Grouping results, automatic and manual</td>
</tr>
<tr>
<td>Grammar</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Text corpus</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Error measurements</td>
<td>Word rate</td>
<td>Word level rate</td>
</tr>
</tbody>
</table>
Figure 7.1 Labeling results for three sentences. Each horizontal bar represents a sentence that is partitioned into signs and \textit{me} labels. The length of the horizontal bar is proportional to the number of frames in the sentence. For each sentence we present groundtruth partitioning and the algorithm output.

results from the algorithm. It is obvious that the signer is signing at different speed for each sign. For instance, the sign I is spread over a large number of frames. The framework can easily handle such cases. Apart from a 1 to 2 frame mismatch at the beginning and at the end, the labeling matches fairly well. Fig. 7.2 shows the sign level error rates we obtained with the optimal $\alpha$ (more on this later) for each test set in the 5-fold validation experimentation, using a trigram model on data set $D_1$. The sign level error rate for each test set ranges between 9% and 28%. On average, the error rate is 17%, with a corresponding correct recognition rate of 83%. In Fig. 7.3, we present results of a head-to-head comparison of the error rates obtained using the enhanced Level Building algorithm presented here and classical Level Building that does not account for movement epenthesis. We found the insertion error has been
Figure 7.2 Sign level error rates using eLB on data set $D_1$. It is broken into insertion, deletion, and substitution. The results are for each test set in the 5-fold cross validation.

decreased by using the proposed method from 10% to 4%. At the same time, the substitution error is reduced from 63% to 5%. Next, we studied the need for the grammar model. Fig. 7.4 shows us side by side the error rates we obtained by using a trigram model and a bigram model. We constructed the grammar models based on a text corpus of 150 sentences. These sentences did not all have corresponding video data. By using trigram model, the average error rate dropped from 32% to 17%. The constraint imposed by a bigram model is more relaxed than that imposed by a trigram model. It may be reiterated that we are using a 0-1 representation of the $n$-grams, i.e., for any instance of a relationship in the corpus, the corresponding count is set to 1, otherwise it is zero. By far, the most important parameter is the **me** labeling cost, $\alpha$. As described earlier, we select the value of $\alpha$ to be the optimal Bayesian decision boundary between match and non-match scores. Fig. 7.5 (a) shows us the match and non-match scores on the training set in data set $D_1$ for one of the 5-fold experiments. As we can see, a matched score usually averages approximately
Figure 7.3 Error rates for eLB and LB. The result is based on data set $D_1$.

Figure 7.4 Error rates with trigram and bigram constraints.
Figure 7.5 Choosing the movement epenthesis (me) labeling cost $\alpha$. The result is for one of the 5-fold experiments. (a) shows us the match and non-match distance scores in the training set used to choose the optimal $\alpha$. The optimal value is 0.89. (b) shows the variation of the errors with different choices of $\alpha$. 

---

Matched score
Non-matched score

---

Insertion error
Deletion error
Substitution error
Total error

---

Error rate
Table 7.4 Comparison of automatically chosen $\alpha$ and manually chosen $\alpha$. Error rates with eLB on data set $D_1$, with automatically (Auto) chosen $\alpha$ and the one (Opt.) that minimizes the error on the test set.

<table>
<thead>
<tr>
<th>Test</th>
<th>Insertion</th>
<th>Deletion</th>
<th>Substitution</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4%</td>
<td>8%</td>
<td>7%</td>
<td>4%</td>
</tr>
<tr>
<td>2</td>
<td>4%</td>
<td>0%</td>
<td>0%</td>
<td>3%</td>
</tr>
<tr>
<td>3</td>
<td>3%</td>
<td>1%</td>
<td>8%</td>
<td>5%</td>
</tr>
<tr>
<td>4</td>
<td>3%</td>
<td>3%</td>
<td>4%</td>
<td>4%</td>
</tr>
<tr>
<td>5</td>
<td>7%</td>
<td>3%</td>
<td>8%</td>
<td>1%</td>
</tr>
<tr>
<td>Avg.</td>
<td>7%</td>
<td>3%</td>
<td>2%</td>
<td>3%</td>
</tr>
</tbody>
</table>

0.4, while a non-matching score is centered around 1.4. The optimal value for the $\alpha$ for this training data set is 0.89.

How good are the trained me labeling costs, $\alpha$? To study this, we computed the best $\alpha$ that minimized the overall error rate on the test set. Fig. 7.5 (b) shows us the variation of the errors with different $\alpha$ for one of the test sets. We see that the automatically chosen $\alpha$ value of 0.89 is near the minimum of the actual error plots. In Table 7.4, we list the errors with the automatically chosen $\alpha$s for each of the 5-fold experiments and compare them with the actual possible minimums. The errors are within 4%. This shows that our method for choosing the optimal $\alpha$ is fairly robust.

### 7.3 Study 2: Comparison with Other Approaches

In this study, we first use the CRF model to effectively detect me segments using the algorithms described in Chapter 6. We take one of the 5 shots of sentences as the test data. The individual signs are taken out from the other 4 sentences to form the training space. The corner detection method usually generates 50-100 feature points. The relational features are counted by $32 \times 32$ bins. With the feature sequences, we use the window size of $w = 7$ to find the key frames. Fig. 7.6 shows us the result of
key frame detection. Note, we do not restrict 1 key frame for each sign or me part. Rather, multiple distinctive frames may be chosen. For example, in Fig. 7.6, we have 2 key frames detected to indicate the starting portion and the end portion of the sign "GATE". Fig. 7.7 shows us the ROC curve for detecting the me point at the key frame sequences, where 4 shots are used as training data, with each of them having 25 sentences. We run a HMM detector also as the baseline algorithm. Additionally, we use the window size of 1, 3, 5 to incorporate adjacent observations. Note, it is difficult for the HMM to use these observations because of the independence assumption. We then compare the performance of our approach with two state of the art methods: conditional random fields [82] and Latent Dynamic-CRF [30]. We use the code from [30] to generate our results. These particular models have been developed in a gesture recognition context, where we have labels corresponding to the gestures (signs). The posterior probability is maximized or estimated directly in training and testing. While the number of labels increases, the model could have a large number of parameters to estimate depending on the selection of feature functions.

For both methods, we use a chaining structure where we have 3 hidden states for each label for LDCRF. Although CRF [82] and LDCRF [30] have shown improved results for limited number of labels, in our experiments we had to use them for 40+ labels. We quantify performance based on using the frame level error rate, i.e., what percentage of the frames are wrongly classified in the test set.

Table 7.5 lists the results. As we can see, CRF and LDCRF perform quite poorly. This is because the number of parameters that needs to be estimated for these models is huge compared to our method, which makes the training unstable. Also, both CRF and LDCRF implicitly model me as 1 single class, which is not realistic. From the results in this section, we can see that CRF works well under a 2-class cases, actually outperforming HMMs. However, in a high number of classes case, which is
Figure 7.6 Key frame detection. (a) shows us the plot of the element of first eigenvector for the sentence, where key frame is detected by selecting the local minimals. Detected key frames are marked as either a sign or \textit{me}, which is shown in (b).
Figure 7.7 The ROC curve for detecting me using HMMs and CRF.

Table 7.5 Framewise labeling results. The comparison of eLB, LB, CRF, LDCRF is included.

<table>
<thead>
<tr>
<th>Methods</th>
<th>eLB</th>
<th>LB</th>
<th>CRF</th>
<th>LDCRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>1</td>
<td>0</td>
<td>1968</td>
<td>15990</td>
</tr>
<tr>
<td>Classes</td>
<td>41</td>
<td>40</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>Data set used</td>
<td>$D_1$</td>
<td>$D_1$</td>
<td>$D_1$</td>
<td>$D_1$</td>
</tr>
<tr>
<td>Grammar model</td>
<td>Trigram</td>
<td>Trigram</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Total test frames</td>
<td>2234</td>
<td>2234</td>
<td>2234</td>
<td>2234</td>
</tr>
<tr>
<td>Correct labeled frames</td>
<td>1530</td>
<td>406</td>
<td>642</td>
<td>460</td>
</tr>
<tr>
<td>Error rate</td>
<td>31%</td>
<td>82%</td>
<td>71%</td>
<td>89%</td>
</tr>
</tbody>
</table>
generally true in sign language recognition, a traditional CRF model has too many parameters to estimate. The accuracy of the model has been decreased a lot and may not perform as well. However, in this case, our eLB method can still get correct sentence recognition results.

7.4 Study 3: Global Features vs. Multiple Local Candidates

The eLB framework can handle both global features that are computed based on the whole image frame and part-based features. Fig. 7.8 shows the result we obtained for both $D_1$ and $D_2$ using these two feature types. For $D_1$ the global feature vector method works well since there is not too much background noise (10% error). However, when we use global feature vectors on $D_2$, where a complex and changing background exists, the error increases significantly (73% error). With a part-based candidate approach, the corresponding error rate is 36%. Note, although the part-based approach has a 47% error on $D_1$, this is because the vocabulary size of $D_1$ is almost twice as big as $D_2$. We can see the part-based candidate approach provides a more robust solution for the low level uncertainty problem.

Fig. 7.9 shows us a visual example of how the candidate hands are selected along with the eLB algorithm. It shows a side by side sub-sampled image sequence from one continuous sentence. The side by side image has the detected hands shown with the original image at the left side, and has all the candidate hands shown on the right side. Note that if the frame is labeled as an me, no hand candidate will be selected. From Fig. 7.9, we can see, during the process that the frame is labeled, the candidate hands are simultaneously selected. It can even work when a second person is working behind the signer which generates more noisy hand candidates. For this, a global feature will definitely fail. It is also interesting to see that for the sentence in 7.9, although the sentence recognition is correct (which is what we want), the framewise
Figure 7.8 Compare global features and part-based candidate hands. The results are based on data sets $D_1$ and $D_2$.

labeling is not completely right. This is due to the fact that we only use very coarse features such as position and moving directions to conduct the match. The signs in between can be easily mixed up with each other. However, the eLB framework can still make the final recognition for the sentence correct based on text corpus and the best matched sign sequence.

The result in this section also answers for us the first part of our research question 5. Our approach does improve the result when there is complex and moving background. We accomplish this by using multiple observations instead of a single observation, where we can reduce the chance of losing an important observation at the low level. And this gives us a more stable result when we apply our algorithms to both single/complex background.
Figure 7.9 Labeling for "FINISH BUY TICKET NOW FINISH". The side by side image has the detected hands shown with the original image at the left side, and has all the candidate hands shown on the right side. The actual label is below the image. Note for me frame no hand candidate will be selected.
7.5 Study 4: Short Sleeves vs. Long Sleeves

In most sign language data sets, clothing is usually controlled. The signer usually wears long sleeved shirts so that just the hands can be segmented using skin color. However, with short sleeved shirts, the hand region can get merged with arm. Merging can also happen with long sleeved shirts when hands cross each other or when the hand crosses the face. Sometimes we can lose the real hands due to over segmentation. We use the medial-axis guided detection approach described in Section 5.1.3 to address this problem. We tested on all of the 3 data sets, using eLB algorithm, with and without the medial-axis-based detection approach. The results are shown in Fig. 7.10. A significant improvement (30%) can be observed over not using the medial-axis-based approach. Fig. 7.11 shows us a visual example of how the candidate hands are selected along with the eLB algorithm. It shows a side by side sub-sampled image sequence from one continuous sentence. The side by side image has the detected hands shown with the original image at the left side, and has all the candidate hands shown on the right side. Note that if the frame is labeled as an me, no hand candidate will be selected. From Fig. 7.11, we can see, during the process that the frame is labeled, the candidate hands are simultaneously selected. The medial-axis representation of the candidate has the advantages of separating the merged arms/hands. It can work under cases where the signer is wearing short sleeved clothes. These candidates cannot be effectively generated without this approach, and errors will propagate to the core matching algorithm level.

The result in this section also answers for us the second part of our research question 5. Our approach does improve the result when the signer wears short sleeved clothes. This is accomplished by using our skeleton representation, by which we can segment the hand from the arms. Hence, we can avoid losing the hand observation
when short sleeved clothes are used, which will lead to a more stable result when we apply our algorithms to both short sleeved/long sleeved cases.

7.6 Study 5: Grouping Results with the Deterministic Model

In this experiment, we use $D_1$ with the deterministic approach. The objective of this experiment is to test the grouping method coupling with a deterministic approach, shown in Chapter 4. The model sign data set is formed from four of the five instances of each sentence. Specifically, for each sign we have 4 examples. We manually select the groups of region patches that are from each hand frame by frame. The sequences of these manually selected groups form the model sequences. Since the number of training samples is limited and it will be hard to estimate the HMM accurately, we use the deterministic matching in this experiment.

For feature vectors, we fit the hand groups with an ellipse in a least square error manner, suppose the ellipse has a major axis $a$, minor axis $b$, and the angle between major axis and x axis is $\theta$. We then have a 10 dimensional feature vector to represent
Figure 7.11 The labeling results for the sequence "TABLE THAT". The side by side image has the detected hands shown with the original image at the left side, and has all the candidate hands shown on the right side. The actual label is below the image. Note that if the frame is labeled as an me, no hand candidate will be selected.
the hand group: x-axis, y-axis, motion displacement at x direction, motion displacement at y direction, length of major axis $a$, length of minor axis $b$, sine of $2\theta$, cosine of $2\theta$, eccentricity of the ellipse, and area of the ellipse.

Fig. 7.12 and Fig. 7.13 show us examples of the generated groups for one frame. As we can see, even for the simple background and simple clothes, the hand can be very fragmented. Fig. 7.12 has more than 100 candidate groups, where the real hands can be generated during the grouping process. Without grouping, we can not guarantee to have the real hand in the candidate list, as shown in Fig. 7.13.

The matched signs are ranked according to their matching scores. Table 7.6 shows us the actual list after ranking the matching score for a few sentences. The signs with a checkmark (√) are actually in the sentence, with the scores listed beside them. The correct signs are towards lower rank, which is what we want. Note that this result was obtained without using higher level grammar. Table 7.7 shows the result for the same sentences, but with the matched starting and ending points listed. Groundtruth starting and ending points are in the brackets. Fig. 7.14 shows us one match result for the test sequence: PEOPLE LONGLINE WAIT ANGRY. Fig. 7.14 (a) is the warped path in the 3D space where the warping is from both the candidate hands selection and the time warping. Fig. 7.14 (b) shows the projection of the same data onto the $X-Y$ plane, which is essentially only the time warping process. Fig. 7.14 (c), is the projection of the same data onto the $X-Z$ plane, which reveals the detected hand’s $X$ coordinates. Fig. 7.15 shows us the recovered position of the hand $X$ coordinates and their hand movements in the test sequence. The result is shown by four parts, each of which corresponds to a sign in the sentence. These results show the real hand position is finally recovered from the recognition result even when the hand is crossing the face. This is a particular hard problem to overcome in gesture recognition. The overall recognition result for this database is shown at Fig. 7.16. In
Figure 7.12 Candidate groups for ASL data set with grouping. While grouping we set $t_{num} = 10$. In (d), there are 125 groups generated, the groups with a circle are the real left and right hands.
while grouping we set $t_{num} = 1$, basically no grouping. In (d), there is no grouping process, just a segmentation, we can see the hand are highly fragmented. Without grouping we can not get the real hand in the list.

this result, 125 sentences are counted with a total of 348 signs. Each model word is matched to the test sequence and the results are ranked according to their matching score. Those words which are in the original sentence but have a larger ranking than 6, 7, 8, 9, 10 will be counted as one error for the ranking 6, 7, 8, 9, 10, respectively. Fig. 7.16 shows us the performance under different number of primitives allowed in one group, from 1, 5 to 10, 20. The upper curve is the recognition rates achieved using manually selected hands. We can see the recognition results increase when the number of primitives increase from 1, 5 to 10. There is a slight drop on 20, which is led by the introduced noisy groups. The overall performance drop compared with the manually selected hands is within 1%-5%.

Notice that the experiment is done without any pre-defined hand model, and similarly the modeling of sign dynamics is very weak with the simple nearest neighbor rules. At rank 6, we achieved a recognition rate around 90% to 94.
Figure 7.14 Matching path of an ASL sentence. (a) The matched sequences in the candidate-time space of the test sentence: PEOPLE LONGLINE WAIT ANGRY. (b) The matched sequences (time warped) of the test sentence. (c) The recovered hand position ($x$ coordinates).
Figure 7.15 The recovered right hand in the test sequence.
Figure 7.16 The test results for ASL data set. Each curve represents the result when one shot is used as test sequence. The horizontal axis denotes the different ranks, the vertical axis denotes the recognition rates.
Table 7.6 List of matched scores for 3 test sentences. The signs with √ are actually in the sentence, with the scores listing beside them.

<table>
<thead>
<tr>
<th>Test</th>
<th>TICKET</th>
<th>BUY</th>
<th>PEOPLE LONG-LINE</th>
<th>WAIT</th>
<th>ANGRY</th>
<th>GATE</th>
<th>WHERE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>In</td>
<td>Signs</td>
<td>Scores</td>
<td>In</td>
<td>Signs</td>
<td>Scores</td>
<td>In</td>
</tr>
<tr>
<td>1</td>
<td>√</td>
<td>Buy</td>
<td>1.51</td>
<td>√</td>
<td>Wait</td>
<td>0.52</td>
<td>√</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>I</td>
<td>2.24</td>
<td>√</td>
<td>LongLine</td>
<td>1.56</td>
<td>Suitcase</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Wait</td>
<td>2.35</td>
<td>Buy</td>
<td>3.07</td>
<td>Not</td>
<td>3.9</td>
</tr>
<tr>
<td>4</td>
<td>√</td>
<td>Ticket</td>
<td>2.44</td>
<td>√</td>
<td>People</td>
<td>3.07</td>
<td>Phone</td>
</tr>
<tr>
<td>5</td>
<td>√</td>
<td>Finish</td>
<td>3.23</td>
<td>I</td>
<td>3.66</td>
<td>√</td>
<td>Where</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>You</td>
<td>6.89</td>
<td>√</td>
<td>Angry</td>
<td>3.87</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Have</td>
<td>7.07</td>
<td>Not</td>
<td>4.16</td>
<td>Need</td>
<td>5.28</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Mad</td>
<td>7.4</td>
<td>Have</td>
<td>4.26</td>
<td>Have</td>
<td>5.81</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Need</td>
<td>7.5</td>
<td>Again</td>
<td>4.26</td>
<td>ThatOne</td>
<td>6.12</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Phone</td>
<td>8.38</td>
<td>Airplane</td>
<td>4.66</td>
<td>Postpone</td>
<td>6.35</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Mean</td>
<td>8.87</td>
<td>Ticket</td>
<td>4.79</td>
<td>Yes</td>
<td>6.37</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>Suitcase</td>
<td>9.11</td>
<td>Lipread</td>
<td>4.82</td>
<td>Can</td>
<td>6.81</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>Cannot</td>
<td>9.84</td>
<td>Gave</td>
<td>5.11</td>
<td>Again</td>
<td>6.85</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>It</td>
<td>10.1</td>
<td>Phone</td>
<td>6.54</td>
<td>Mad</td>
<td>7.63</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Can</td>
<td>10.52</td>
<td>Finish</td>
<td>6.64</td>
<td>Understand</td>
<td>7.71</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>People</td>
<td>11.17</td>
<td>Just</td>
<td>7.21</td>
<td>Key</td>
<td>8.34</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>Again</td>
<td>11.58</td>
<td>Key</td>
<td>7.24</td>
<td>Table</td>
<td>8.45</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>Gave</td>
<td>11.95</td>
<td>Gate</td>
<td>7.73</td>
<td>Lipread</td>
<td>8.8</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Not</td>
<td>12.62</td>
<td>Understand</td>
<td>8.52</td>
<td>Angry</td>
<td>9.59</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>Gate</td>
<td>13.68</td>
<td>Need</td>
<td>8.9</td>
<td>I</td>
<td>9.62</td>
</tr>
</tbody>
</table>

7.7 Study 6: Grouping Results with Hidden Markov Models

To study the effect of grouping with HMM-based matching, we need to use a data set that supports the HMM learning. The ASL data sets do not have sufficient number of repetitions per sign to allow this. Hence we use the Human Computer Interaction (HCI) data set that has been recently collected by another research group, i.e., Just and Marcel [78], which is also referred as $D_4$. The data set is for recognizing 7 hand actions: push, rotate front, rotate back, rotate left, rotate right, rotate up, and rotate down. The authors of the data have explicitly separated the training and test data, where the training data consist of 4 subjects, each of whom performed the 7 actions...
Table 7.7 Matched positions and manually recognized positions. The manually recognized positions are in the bracket.

<table>
<thead>
<tr>
<th>Test</th>
<th>TICKET</th>
<th>PEOPLE</th>
<th>LONGLINE</th>
<th>GATE WHERE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BUY FINISH</td>
<td>WAIT ANGRY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>Signs</td>
<td>Start</td>
<td>End</td>
<td>Signs</td>
</tr>
<tr>
<td>1</td>
<td>Ticket</td>
<td>18(18)</td>
<td>21(21)</td>
<td>People</td>
</tr>
<tr>
<td>2</td>
<td>Buy</td>
<td>28(28)</td>
<td>31(31)</td>
<td>Long</td>
</tr>
<tr>
<td>3</td>
<td>Finish</td>
<td>42(42)</td>
<td>47(47)</td>
<td>LineWait</td>
</tr>
<tr>
<td>4</td>
<td>Angry</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10 times, with 5 of them at one session time and 5 of them at the other. The test data has the same shots but with 3 different subjects. The total number of test sequences is 210. The data set has shots from 2 fixed cameras, one shot from the left side and the other shot from the right side. We used the joined results of the two views in this work. For this experiment, since we have sufficient number of training data, we use HMMs to conduct the matching instead. Since this data set was collected with yellow and blue colored gloves, it allows us to make comparisons with color-based hand segmentation schemes. As baseline performance comparison, we considered (i) manually segmented hands, and (ii) hands segmented using the information about the color of the gloves. For color-based hand segmentation, each glove color is modeled as a mixture of 3 Gaussians in the color space. For the proposed approach, we considered just region segmentation patches, detected as outlined earlier. Note that although we use color for segmentation and grouping, we do not use the knowledge that a specific color corresponds to the hand. Fig. 7.17 shows examples of region segmentation and groups. Note that some hypotheses correspond to non-hand parts of the image or for other hands that might be present. We used the same feature vector as we used in Experiment 7.6.

Fig. 7.18 and Fig. 7.19 show us all the candidate groups for one frame. We can see that the list in Fig. 7.18 consists of the real hand group and the group
Figure 7.17 HCI data set results. Candidate groups of regions generated for some frames. Notice there are 3 hands in the frame. (a) Original frame. (b) Segmented image (boundary). (c) Segmented image. (d) Primitives around the third hand. (e) Primitives around left hand. (f) Primitives around the right hand. (g) The candidate groups for the third hand. (h) The candidate groups for the left hand. (i) The candidate groups for the right hand.
Table 7.8 Compare results with grouping and without grouping.

<table>
<thead>
<tr>
<th>Low level Matching</th>
<th>Grouping</th>
<th>No grouping</th>
<th>Grouping</th>
<th>No grouping</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sum-sum</td>
<td>Sum-sum</td>
<td>Max-max</td>
<td>Max-max</td>
</tr>
<tr>
<td># of total test frames</td>
<td>7249</td>
<td>7249</td>
<td>7249</td>
<td>7249</td>
</tr>
<tr>
<td># of groups/frame, view</td>
<td>94</td>
<td>22</td>
<td>94</td>
<td>22</td>
</tr>
<tr>
<td># of total sample seq.</td>
<td>210</td>
<td>210</td>
<td>210</td>
<td>210</td>
</tr>
<tr>
<td># of correct samples seq.</td>
<td>193</td>
<td>58</td>
<td>190</td>
<td>60</td>
</tr>
</tbody>
</table>

that generates it. In Fig. 7.19, these groups do not exist because no grouping is done. The following Table 7.8 shows us the number of groups per frame and the number of correctly recognized gestures. Without grouping, all the candidate groups are singleton groups, the real hand normally is not included because the hand area is fragmented. Grouping is a necessary process, even for such a data set with color gloves. We consider recognition with each of the three probabilistic measures outlined earlier. The correct recognition rates are shown in Fig. 7.20. The 5 approaches (the two baselines and the three HMMs ones) give us the recognition rates: 79%, 94%, 91%, 92%, and 91%. This result actually gives us the answer of research question 6. From this result we can also observe:

1. For each frame, above 95% of the groups generated were noisy, with some being just random patches. However, their contribution to the final overall sequence is quite small, since they were not well linked across frames. Our approach allows us to recover from such errors. However, for the commonly used color-based hand segmentation approach, if any one frame has noisy hands, the recognition might fail. This is the reason why the recognition with hands segmented using just color information results in low performance.

2. Our approach that accommodates imperfect segmentation only has a 2% performance loss compared with the approach with manual segmentation.
Figure 7.18 Candidate groups for the gesture data set with grouping. While grouping we set $t_{num} = 10$. In (d), there are 118 groups generated, the group with a rectangle is the real hand, and the group with a circle is the one that the real hand is generated from.
Figure 7.19 Candidate groups for the gesture data set without grouping. While grouping we set $t_{num} = 1$. In (d), since there is no grouping process, just a segmentation, we can see the hand are highly fragmented, without grouping we cannot get the real hand in the list.

Figure 7.20 Recognition of hand gestures for five different approaches. The first two are based on manual and color-based segmentation of the hands. The next three does not use the knowledge of the hand color and take into account fragmented observations. The three corresponds to the three different kinds of probabilities that can be computed, $P_{\text{max, sum}}$, $P_{\text{sum, sum}}$, and $P_{\text{max, max}}$ using the HMM proposed in this work.
Fig. 7.21 and Fig. 7.22 show the recognition rate on a per-gesture and per-subject basis. We can see the majority of errors come from one subject and the three gestures that can be easily mixed up. Subject 1 performed each gesture with larger motion than the other subjects in the training data. Such a case is hard to improve by using only the position features, hence subject 1 produced the majority of the errors. Among the gestures, rotate front, push and rotate right all have motions moving forward and backward. There are only subtle orientation change in the palm. Hence these actions produced majorities of the errors. However, the performance measure of interest for this work is how well the recognition rate with multiple grouped observations match that with perfect segmentation. On this account, the performance is quite strong.

Fig. 7.23 shows a visual example of the optimal groups selected for the best match corresponding to the rotate back action. There are two parts to the movement, backwards and forwards. Fig. 7.23 (a) and (b) shows the selected groups for these two parts over laid on each other. Fig. 7.23 (c) shows the X (horizontal) coordinates
Figure 7.22 Recognition performance of each subject separately. They are conducted by using summed-summed approach and Manual Segmentation.

of the revealed hand by using the optimal state and sequence pair approach, we can see the nature of the change of X coordinates match the hand positions. The indexed forward approach produces similar results. After this, Fig. 7.24 shows the tracking results of the other 7 gestures.
Figure 7.23 The optimal groups corresponding to one of the hands. It is done by using the HMM. (a) is the first part of the "rotate back" gesture. (b) is the second part of the "rotate back" gesture. (c) is the computed horizontal position of the hand.
Figure 7.24 The tracking results for the first test instance. These test instances are from the other 6 gestures and the result is generated by max-max method. The first column is the first frame of the sequence. The second column is the tracking of the first part, followed by the tracking of the second part in the third column, and finally arrive at the last frame of the sequence in column 4.
CHAPTER 8
CONCLUSION AND FUTURE WORK

In this work, we strived to attack two fundamental problems in automatic video-based sign language and gesture recognition systems. The first problem is the movement epenthesis ($\text{me}$) problem. This problem is due to our need to exclude from analysis, the extra movements (i.e., movement epenthesis) that signers naturally have to make to transit or move their hands from one sign to the next. If our recognition system also analyzed these extra movements, it might be misled and generate extra signs where there were none present. We proposed an enhanced Level Building algorithm (eLB) to attack this problem without any explicit modeling of $\text{me}$. The second problem is the low level hand segmentation problem. Ambiguity of hand detection can always happen. If the hand is not detected correctly, the high level matching process will be misled and generate wrong matching results. We proposed a grouping algorithm and matched the groups with several new decoding processes. This algorithm allowed us to avoid the need for perfect segmentation at the feature extraction level, and the grouping algorithm effectively reduced the chance of losing the true hand we wanted.

Initially, we presented the enhanced Level Building algorithm, built around dynamic programming, to address the problem of movement epenthesis in continuous sign sentences. Our approach did not need to explicitly model movement epenthesis. Hence, the demand on annotated training video data was low. We compared the performance of enhanced Level Building with classical Level Building algorithm,
which has been used for connected word recognition in speech. We found significant improvements. To overcome the low level hand segmentation errors, we incorporated another dynamic programming process, nested within the first one, to optimize over possible choices from part-based multiple hand candidates. Our results have shown that the part-based candidate approach is more stable for a complex and changing background. Our extensive experiments demonstrated the effectiveness of the matching between the test data and the training data. We demonstrated this by extensively testing the important parameters, such as the automatically chosen $\alpha$, and the number of primitives in a group. In the context of ASL, we moved forward the area of recognition of signs in sentences, while accounting for movement epenthesis. We also contributed towards the ability to handle general backgrounds and relaxation of clothing restrictions. The developed enhanced Level Building algorithm solved the general problem of recognizing motion patterns from streams of compositions of motion patterns with portions, for which we did not have any model. Such situations could also arise in human computer interaction where one has to consider compositions of individual gestures or in long term monitoring of a person performing multiple activities.

We also compared the eLB algorithm with state of the art labeling algorithms such as conditional random fields (CRF), etc. We first used conditional random fields (CRF) formulation along with the concept of key frames, capturing frames with the distinctive short term motion, to detect and label me in a sign language sentence. The CRF had the advantage of directly modeling the posterior probability and could allow any dependence between the states and observations, which is desired for labeling a sequence with highly related context such as ASL sentences. Our experiments found that the CRF-based approach significantly outperformed an HMM-based one. However, this was a 2-class case. We then did an experiment based on 40-class
models to compare the performance of CRF and eLB, where we could see CRF did not work properly. This was because CRF had a large number of parameters and the training could not guarantee a good point of convergence when searching for the best parameter set. However, our eLB was more effective and straightforward. We only had 1 parameter to train, which was used to model the boundary between a match and a non-match. We discovered that the boundary found in our test was very effective at separating the signs and the segments, which led to the correctly labeled results in the end.

For the low level segmentation problem, we proposed a new grouping method for gesture and sign recognition from videos which do not rely on skin color models and can work with imperfect segmentation of scenes. We addressed the hard problem of hand segmentation by coupling it with recognition, via an intermediate grouping process. The grouping process generated layers of overlapping groups that were linked across time in a graph structure. We showed how the search for the optional sequence of groups could be arrived at with different matching models. For HMMs we showed how three different kinds of probabilities could be computed, based on maximization and averaging over the underlying states and groups. We demonstrated its efficiency for HCI hand action recognition tasks using a publicly available data set spanning multiple subjects and actions, against complex backgrounds. The recognition rates were very close (91% compared to 93%) of those achieved by manual segmentation and much better (91% compared to 79%) than that achieved by color-based hand segmentation. As a byproduct of the recognition problem, we also segmented the hand in each frame. We demonstrated its efficiency for sign recognition and HCI hand action recognition tasks. As our results show, using the coupled framework, we were also able to provide an overall solution based on the segmentation and matching results, and could improve the results when the hand segmentation was not successful.
In this work, we have focused on the two proposed matching algorithms. We have fully investigated the important parameters for the matching process, such as the $\alpha$ in eLB algorithm, and the number of primitives in the grouping algorithm. In the future, we will also need to investigate the parameters regarding the imaging process and the low level processing process. These parameters include the temporal/spatial resolutions, lighting conditions, edge detection parameters, and segmentation parameters, etc. On the other hand, we focused on solving the two problems in a continuous sign language recognition system in this work. In order to build a more comprehensive and robust system, we will need to address other important problems in the future. We have discussed some of these problems in Chapter 2. These problems include, but are not limited to the recognition of non-manual aspects of signed sentences, recognition of signs made by different signers or filmed from different view angles and dealing with the problem of how to recognize signs that are made slightly differently based on which signs precede or follow them (coarticulation). Other than these, our system could not work for real time video currently. Our future work may include an investigation to speed up the matching process so that we can work towards applications which work in real time. One way to speed up the process is by using a more representative model such as a probabilistic model. Note, although our current eLB matching process is conducted under an example-based deterministic approach, the eLB algorithm can be extended to a probabilistic model to further capture the variations among the data, like a normal LB does.

We have shared the source code of our algorithms, including the eLB algorithm, Hidden Markov Models with groups, and the annotation tools online. The reader can refer to "http://figment.csee.usf.edu/ASL/" for more details.
REFERENCES


Appendix A Data Collection

In our experiment, $D_1$ was captured from [83]. $D_4$ was captured and shared by [78]. Fig. A.1 shows us the capturing process of $D_2$ and $D_3$. We only used the data from Camera G in the experiment shown in this work.

While capturing the videos, the signer was standing at the center. 2 500 watt white bulbs were used with a white umbrella to sustain illumination. A side view camera was set up at the main-hand side of the signer. $E$ was a dragonfly camera for the face sequence, which was set up at the eye level of the signer, with a small angle. $G$ was another dragonfly camera for the front view sequence, which was also set up at the eye level of the signer. $F$ was setup just in front of $G$, which was a stereo bumblebee camera. This camera was setup at the neck level of the signer.

The videos were streamed into an SCSI drive using an MPEG 4 encoder with 2M bits/s bitrate, the frame rate was set to be 30, and the resolution of the camera was
set to 640x480 pixels with 24 bit depth. A PGR synchronizer was used to synchronize the 2 dragonfly cameras and the 2-view stereo bumblebee camera.
Appendix B Groundtruth Tools

We provided a semi automatic annotation tool to groundtruth ASL image sequences. A candidate hand generator was applied by using the mean shift image segmentation algorithm and a greedy seeds growing algorithm. After a number of hand candidates are generated, the user can reduce the number of candidates by simple interaction (mouse click). The tool also provided a hand tracking function for faster processing and a face detection function for non-manual signal groundtruthing purposes. In addition, we provided a two-pass groundtruthing scheme unlike other groundtruthing tools that only do one-pass. Our first pass processing was automatic, and the second pass was semi automatic based on the first pass’s result.

We were aware of many other video annotation tools for groundtruthing purposes. However, most of them focus on scene segmentation or key frame detection, (e.g. IBM EVA [84], ESP Game [85]). Only few of them focus on local feature extraction and temporal tracking together. For example, The VIPER annotation Tool proposed by Pfund [86] provided image segmentation, temporal segmentation and even annotation together. ViPer-GT proposed by Doermann [87] can detect multiple objects and track them using bounding box automatically, Marcotegui et al. proposed VOGUE [88], where a number of image and video segmentation techniques were incorporated for object annotation purpose. All of these tools are stand alone applications providing semi automatic groundtruthing function with friendly user interface.

Our annotation tools (SignGT) were a side-product of our vision-based ASL recognition system. It also provided a semi automatic scheme for efficient groundtruthing purposes. However, its main purpose was to segment the hand pixels frame by frame. Instead of using the existing segmentation and tracking algorithm as in the existing tools, we advocated a candidate hand generator approach that was more reliable dur-
Appendix B (Continued)

ing hand shape change, and hand crossing face situation. Unlike the existing tools
where only one-pass is conducted, we offered a two-pass scheme for faster processing,
where the first pass generated the candidate hands automatically.

Fig. B.1 illustrates the two-pass scheme. In Fig. B.1 (a), ASL video frames were
firstly segmented into seed primitives. These primitives were grouped by a grouping
engine to generate overlapped candidate hand groups, where each group may consist
of one or more primitives. This step was automatic and no user interaction was
involved. In Fig. B.1 (b), where the second pass was taken, the grouped results was
loaded back for the user’s examination. Note the number of generated groups could
be huge. Hence, we allowed the user to mouse click the hand region and reduce the
number of candidates to be examined. At the same time, a tracking method was also
incorporated among adjacent frames to improve efficiency.

After the candidate groups and their links were generated, user interaction was
needed to select the best group and guard the tracking result. We provided a set
of functionalities which specifically work with sign sentences. These functionalities
were built upon the candidate generator discussed in Chapter 5, a simple tracking
technique that works with the links between the candidate groups, a face detector for
non-manual information analysis, a glossing tool, and various elements that facilitates
the hand groundtruthing.

The application was coded under the Microsoft Visual Studio Environment, using
MFC class, OpenCV libraries and related windows APIs. Fig. B.2 illustrates us the
GUI. Important functions related to ASL were supported as:

1. 3 views of current frame being processed: the first is the dominant hand view,
    the second is the non-dominant hand view, and the third is the view for both.
Appendix B (Continued)

Figure B.1 Overview of the two-pass approach.
Appendix B (Continued)

2. Click to select: one can click on the hand area to give the direction of the groups to be shown.

3. Missing hand checkbox: one can choose the current hand as missing if the hand is out of scene.

4. Glossing textbox: one can input the gloss for the current frame. The gloss was propagated to the next frame automatically.

5. 2 hand listboxes: the two listboxes below show the list of the candidate dominant hands and candidate non-dominant hands. The list was ranked by their boundary smoothness and the tracking result.

6. Face detector: automatic face detector, shown as the blue bounding box.

7. Play, stop, step button: pressing the play button will automatically track all of the hands and save the result. Stop button will stop the tracking. Step button allowed the application track one frame and wait for the user’s response, which was most often used.

8. Redo checkbox: redo checkbox allows one to re-detect the current sequence.

We ran our SignGT with 2 data sets with different parameter settings. Both of them consisted of ASL sign sentences. The first data set had simple background, the resolution was 460x290 and there were 10675 frames. On average there were 100 candidate hands generated for this data set. And it took us less than 8 hours to finish groundtruthing both hands. The second data set had a complex background with 640x480 resolution. There were 500 candidate hands generated. We took 500
frames and groundtruthed it within 1 hour. Note the time we refer to here is the user interaction time. That is, the time of the second pass.

In Fig. B.3 (a), we show the performance over the 2 data sets of the two passes. We used a P4 2.4G CPU with 4G memory. The number shown is the time taken per frame. Our first pass took relatively longer since we incorporated automatic segmentation, face detection, and the greedy seeds growing algorithm. However, it was completely taken off-line. The second pass was done by reloading the candidate result. It took much shorter time.

On the other hand, Fig. B.3 (b) shows us the time taken for a different configuration of the tool. We chose 500 frames from the simple background set to do the experiment. Here we refer "A" as the method to only use the generated result, "B" refers to using the click-to-select method to reduce the candidate set, "C" refers to us-
Figure B.3 Performance of the annotation tool. (a) Performance of first pass and second pass. (b) Performance of different method sets.

ing the candidate with tracking method, "D" refers to the method of using candidate with both click-to-select and tracking methods. Tracking contributed a lot because it exploited the temporal relationship. The click-to-select method did help especially when tracking failed. For example when large motion happened, hand shape changed drastically and occlusion happened.

Fig. B.4 shows us some visual result of the generated candidate hands. We can see our method can get the hand shape out even when there is significant occlusion and overlaps. In particular, Fig. B.4 (c) shows us the result where face crosses hand, and Fig. B.4 (f) shows us where the two hands cross each other.
Figure B.4 Illustration of the multiple candidates. (a) Original frame 1. (b) Segmented frame 1. (c) List of candidate groups in frame 1. (d) Original frame 2. (e) Segmented frame 2. (f) List of candidate groups in frame 2.
Appendix C Text Corpus

Here we show the text sequence we used when we recognized the continuous sentence on the data set $D_1$, $D_2$, $D_3$. Table C.1 and C.2 shows us the extended 150 sentences we used as the text corpus corresponding to $D_1$. "Extended" means we had extra number (125) of sentence compared to the original (25) number of test sentences. Table C.3 and C.4 shows us the non-extended sentences we used as the text corpus corresponding to $D_2$ and $D_3$. 
Table C.1 Sequences used as the text corpus in $D_1$.

<table>
<thead>
<tr>
<th>LIPREAD CAN I</th>
<th>LIPREAD AGAIN CAN I</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNDERSTAND CAN I</td>
<td>WAIT CAN I</td>
</tr>
<tr>
<td>WAIT AGAIN CAN I</td>
<td>MY TICKET GIVE CAN I</td>
</tr>
<tr>
<td>MY TICKET GIVE AGAIN CAN I</td>
<td>MY ID PAPER GIVE CAN I</td>
</tr>
<tr>
<td>MY ID PAPER GIVE AGAIN CAN I</td>
<td>MY PHONE GIVE CAN I</td>
</tr>
<tr>
<td>MY PHONE GIVE AGAIN CAN I</td>
<td>TICKET BUY AGAIN CAN I</td>
</tr>
<tr>
<td>TICKET BUY CAN I</td>
<td>SUITCASE PACK AGAIN CAN I</td>
</tr>
<tr>
<td>SUITCASE PACK CAN I</td>
<td>FINISH CAN I</td>
</tr>
<tr>
<td>PHONE BUY AGAIN CAN I</td>
<td>PHONE BUY CAN I</td>
</tr>
<tr>
<td>LIPREAD CANNOT I</td>
<td>LIPREAD AGAIN CANNOT I</td>
</tr>
<tr>
<td>UNDERSTAND CANNOT I</td>
<td>WAIT CANNOT I</td>
</tr>
<tr>
<td>WAIT AGAIN CANNOT I</td>
<td>MY TICKET GIVE CANNOT I</td>
</tr>
<tr>
<td>MY TICKET GIVE AGAIN CANNOT I</td>
<td>MY ID PAPER GIVE CANNOT I</td>
</tr>
<tr>
<td>MY ID PAPER GIVE AGAIN CANNOT I</td>
<td>TICKET BUY AGAIN CANNOT I</td>
</tr>
<tr>
<td>TICKET BUY CANNOT I</td>
<td>SUITCASE PACK AGAIN CANNOT I</td>
</tr>
<tr>
<td>SUITCASE PACK CANNOT I</td>
<td>FINISH CANNOT I</td>
</tr>
<tr>
<td>PHONE BUY AGAIN CANNOT I</td>
<td>PHONE BUY CANNOT I</td>
</tr>
<tr>
<td>SUITCASE SUITCASE</td>
<td>MEAN WHAT</td>
</tr>
<tr>
<td>I UNDERSTAND</td>
<td>YOU UNDERSTAND</td>
</tr>
<tr>
<td>YOU UNDERSTAND ME</td>
<td>I UNDERSTAND YOU</td>
</tr>
<tr>
<td>I UNDERSTAND THAT</td>
<td>YOU UNDERSTAND THAT</td>
</tr>
<tr>
<td>SUITCASE I PACK FINISH</td>
<td>WAIT I FINISH</td>
</tr>
<tr>
<td>MY TICKET JUST GIVE I FINISH</td>
<td>MY TICKET GIVE FINISH</td>
</tr>
<tr>
<td>TICKET I GIVE FINISH</td>
<td>MY PHONE I JUST GIVE FINISH</td>
</tr>
<tr>
<td>I GIVE MY TICKET FINISH</td>
<td>I JUST GIVE MY TICKET FINISH</td>
</tr>
<tr>
<td>I JUST GIVE TICKET FINISH</td>
<td>I JUST GIVE MY PHONE FINISH</td>
</tr>
<tr>
<td>SUITCASE MOVE I FINISH</td>
<td>I FINISH MOVE SUITCASE</td>
</tr>
<tr>
<td>SUITCASE I MOVE FINISH</td>
<td>SUITCASE I FINISH MOVE</td>
</tr>
<tr>
<td>DONT KNOW I</td>
<td>DONTKNOW WHERE</td>
</tr>
<tr>
<td>DONTKNOW WHERE</td>
<td>DONTKNOW ID PAPERS WHERE</td>
</tr>
<tr>
<td>I DONTKNOW TABLE WHERE</td>
<td>I DONTKNOW KEY WHERE</td>
</tr>
<tr>
<td>I DONTKNOW PHONE WHERE</td>
<td>I DONTKNOW SUITCASE WHERE</td>
</tr>
<tr>
<td>I DONTKNOW AIRPLANE WHERE</td>
<td>I DONTKNOW TICKET WHERE</td>
</tr>
<tr>
<td>I DONTKNOW PEOPLE WHERE</td>
<td>I DONTKNOW KEY WHERE</td>
</tr>
<tr>
<td>DONTKNOW WHERE I</td>
<td>DONTKNOW ID PAPERS WHERE I</td>
</tr>
<tr>
<td>DONTKNOW TABLE WHERE I</td>
<td>DONTKNOW KEY WHERE I</td>
</tr>
<tr>
<td>DONTKNOW PHONE WHERE I</td>
<td>DONTKNOW SUITCASE WHERE I</td>
</tr>
<tr>
<td>DONTKNOW AIRPLANE WHERE I</td>
<td>DONTKNOW TICKET WHERE I</td>
</tr>
<tr>
<td>DONTKNOW PEOPLE WHERE I</td>
<td>DONTKNOW KEY WHERE I</td>
</tr>
<tr>
<td>I NOT HAVE KEY</td>
<td>I NOT HAVE SUITCASE</td>
</tr>
</tbody>
</table>

130
Appendix C (Continued)

Table C.2 More sequences used as the text corpus in $D_1.$

<table>
<thead>
<tr>
<th>I NOT HAVE PHONE</th>
<th>I NOT HAVE ID PAPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>I NOT HAVE TICKET</td>
<td>THAT MY THAT</td>
</tr>
<tr>
<td>SUITCASE MY THAT</td>
<td>TICKET MY IT</td>
</tr>
<tr>
<td>PHONE MY IT</td>
<td>ID PAPER MY IT</td>
</tr>
<tr>
<td>I NEED THAT I</td>
<td>I NEED MY PHONE</td>
</tr>
<tr>
<td>I NEED MY SUITCASE</td>
<td>I NEED MY TICKET</td>
</tr>
<tr>
<td>I NEED MY ID PAPER</td>
<td>I NEED PHONE</td>
</tr>
<tr>
<td>I NEED SUITCASE</td>
<td>I NEED TICKET</td>
</tr>
<tr>
<td>I NEED ID PAPER</td>
<td>MY PHONE I NEED</td>
</tr>
<tr>
<td>MY SUITCASE I NEED</td>
<td>MY TICKET I NEED</td>
</tr>
<tr>
<td>MY ID PAPER I NEED</td>
<td>PHONE I NEED</td>
</tr>
<tr>
<td>SUITCASE I NEED</td>
<td>TICKET I NEED</td>
</tr>
<tr>
<td>ID PAPER I NEED</td>
<td>MY PHONE NEED</td>
</tr>
<tr>
<td>MY PHONE NEED I</td>
<td>MY SUITCASE NEED</td>
</tr>
<tr>
<td>MY TICKET NEED</td>
<td>MY ID PAPER NEED</td>
</tr>
<tr>
<td>PHONE NEED</td>
<td>SUITCASE NEED</td>
</tr>
<tr>
<td>TICKET NEED</td>
<td>ID PAPER NEED</td>
</tr>
<tr>
<td>WHY</td>
<td>MEAN</td>
</tr>
<tr>
<td>SUITCASE WHERE</td>
<td>TICKET WHERE</td>
</tr>
<tr>
<td>ID PAPER WHERE</td>
<td>PHONE WHERE</td>
</tr>
<tr>
<td>AIRPLANE WHERE</td>
<td>MY SUITCASE WHERE</td>
</tr>
<tr>
<td>MY TICKET WHERE</td>
<td>MY ID PAPER WHERE</td>
</tr>
<tr>
<td>MY PHONE WHERE</td>
<td>SUITCASE MOVE CAN I</td>
</tr>
<tr>
<td>TICKET GIVE CAN I</td>
<td>AIRPLANE POSTPONE AGAIN MAD I</td>
</tr>
<tr>
<td>AIRPLANE POSTPONE AGAIN I MAD</td>
<td>AIRPLANE AGAIN POSTPONE I MAD</td>
</tr>
<tr>
<td>AIRPLANE AGAIN POSTPONE MAD I</td>
<td>TICKET BUY FINISH</td>
</tr>
<tr>
<td>PEOPLE LONGLINE WAIT ANGRY</td>
<td>GATE WHERE</td>
</tr>
<tr>
<td>SUITCASE</td>
<td>YES</td>
</tr>
<tr>
<td>NO</td>
<td>MY TICKET JUST GIVE</td>
</tr>
<tr>
<td>MY ID PAPER JUST GIVE</td>
<td>TICKET JUST GIVE</td>
</tr>
<tr>
<td>ID PAPER JUST GIVE</td>
<td>ID PAPER WHERE</td>
</tr>
<tr>
<td>ID PAPER TABLE</td>
<td>PHONE THAT TABLE THAT</td>
</tr>
<tr>
<td>SUITCASE THAT TABLE THAT</td>
<td>ID THAT TABLE THAT</td>
</tr>
<tr>
<td>PAPER THAT TABLE THAT</td>
<td>PHONE THAT TABLE THAT</td>
</tr>
<tr>
<td>TICKET THAT TABLE THAT</td>
<td>ID PAPER THAT TABLE THAT</td>
</tr>
</tbody>
</table>
Appendix C (Continued)

Table C.3 Sequences used as the text corpus in $D_2$.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Text Corresponding to ( D_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOW I BUY TICKET FINISH</td>
<td>I WANT BUY TICKET I WHERE</td>
</tr>
<tr>
<td>WHY SHOULD I BUY TICKET WHY</td>
<td>WHY CANNOT THAT BUY TICKET WHY</td>
</tr>
<tr>
<td>FINISH BUY TICKET I</td>
<td>FINISH BUY TICKET FROM AGENT FINISH</td>
</tr>
<tr>
<td>WHERE CAN I BUY TICKET WHERE</td>
<td>NOW I BUY TICKET FINISH</td>
</tr>
<tr>
<td>FINISH BUY TICKET FROM AGENT</td>
<td>WHERE CAN I BUY TICKET WHERE</td>
</tr>
<tr>
<td>NOW I BUY TICKET FINISH</td>
<td>FINISH BUY TICKET NOW FINISH</td>
</tr>
<tr>
<td>YOU CAN BUY THAT FOR THAT</td>
<td>I CAN BUY TICKET FOR THAT I</td>
</tr>
<tr>
<td>THAT FINISH BUY TICKET THAT</td>
<td>WHY CANNOT THAT BUY TICKET THAT</td>
</tr>
<tr>
<td>BUY TICKET NOW FINISH</td>
<td>YOU CAN BUY ALSO FOR THAT</td>
</tr>
<tr>
<td>WHERE CAN I BUY TICKET WHERE</td>
<td>WANT BUY TICKET I WHERE Agent WHERE</td>
</tr>
<tr>
<td>WHY CANNOT I BUY TICKET I</td>
<td>THAT FINISH BUY TICKET THAT</td>
</tr>
</tbody>
</table>

Table C.4 Sequences used as the text corpus in $D_3$.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Text Corresponding to ( D_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN I BUY TICKET</td>
<td>FINISH BUY MY TICKET</td>
</tr>
<tr>
<td>I CAN UNDERSTAND YOU I</td>
<td>ID PAPER THAT TABLE THAT</td>
</tr>
<tr>
<td>ID PAPER THAT TABLE</td>
<td>I HAVE PAPER TICKET</td>
</tr>
<tr>
<td>I NOT UNDERSTAND YOU I</td>
<td>I UNDERSTAND YOU I</td>
</tr>
<tr>
<td>MY TICKET TABLE THAT</td>
<td>NEED BUY TICKET</td>
</tr>
<tr>
<td>NOT UNDERSTAND I</td>
<td>TABLE THAT</td>
</tr>
<tr>
<td>TABLE WHERE</td>
<td>TICKET THAT TABLE</td>
</tr>
<tr>
<td>UNDERSTAND I</td>
<td>WHERE MY ID PAPER WHERE</td>
</tr>
<tr>
<td>YOU NOT UNDERSTAND I YOU</td>
<td>YOU UNDERSTAND I</td>
</tr>
</tbody>
</table>
ABOUT THE AUTHOR

Ruiduo Yang received a Bachelor of Science Degree in Computer Science from Peking University, Beijing, China in 2001 and a Master of Philosophy Degree in Computer Science and Engineering from Hong Kong University of Science and Technology, Hong Kong, China in 2003. He is currently a PhD Candidate in the department of Computer Science and Engineering in University of South Florida. His research interests include Sign language/Gesture Recognition, Machine Learning, Sequence Recognition, Video Analysis and Video Coding. He has authored/coauthored more than 10 publications in the field of Pattern Recognition and Video processing. He served as a reviewer for International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) and Journal of Pattern Recognition.