ROBUST GESTURE RECOGNITION

A Dissertation
Presented to
The Academic Faculty

By

You-Chi Cheng

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2015

ROBUST GESTURE RECOGNITION

Approved by:

Dr. Biing-Hwang Juang, Committee Chair
Professor, School of ECE
Georgia Institute of Technology

Dr. Gee-Kung Chang
Professor, School of ECE
Georgia Institute of Technology

Dr. Chin-Hui Lee, Advisor
Professor, School of ECE
Georgia Institute of Technology

Dr. Yao Xie
Asst. Professor, School of ISyE
Georgia Institute of Technology

Dr. Mark A Clements
Professor, School of ECE
Georgia Institute of Technology

Date Approved: December 2014

To my parents, Yih-Nen Jeng and Ying-Hwa Hsu.

ACKNOWLEDGMENT

My deepest gratitude goes first and foremost to Professor Chin-Hui Lee , my research advisor, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

Second, I would like to express my heartfelt gratitude to Professors Biing-Hwang (Fred) Juang and Mark A Clements for their precious comments and advice since my proposal examination. I also highly appreciate Professors Gee-Kung Chang and Yao Xie for serving on my committee. I owe a great acknowledgement to Dr. Ville Hautamäki for his great advice and assistance on the theoretical parts of my research work. And other great acknowledgements will go to my colleagues in Robert BOSCH research and technology center, Dr. Fuliang Weng, Dr. Zhongnan Shen, Dr. Zhe Feng, Dr. Kui Xu, and Dr. Zhengyu Zhou for their support and guidance during this Ph.D. study.

Third, I would like to take this opportunity to express my gratitude to a number of people for their support over this Ph.D. study. I sincerely acknowledge my previous colleagues- Dr. Yu Tsao, Dr. Chengyuan Ma, Dr. Jinyu Li, Dr. Jeremy Reed, Prof. Marco Siniscalchi, Prof. Juanjuan Zhu, Dr. Byung Ki Byun, Dr. Ilseo Kim, and Dr. Aleem Mushtaq, Dr. Chen-Yu Chiang and current colleagues- I-Fan Chen, Zhen Huang, Kehuang Li for their fruitful suggestions and support during this Ph.D. study.

In addition, I would like to express my thanks to current and previous local friends here in Atlanta- Jen-Cheng Huang, Hwa-You Hsu, Jessie Lee, Chi-Ti Hsieh, Jen-Feng Li, Cheng-Lin Tsao, Pauline Yu, Ruoju Liu, Guomei Zhu, Ran Zhou, Chao Weng, Zhenhua Yu, Huicong Qiao, Zhen Wang, and Sungkap Yeo; and friends in other areas in the United states- Yi Gai, Fei Ye, Bing He, Feng-Hao Liu, Ji Wang, Chenliang Xu, and Chao Yang. Their supports and encouragements are highly appreciated during this period of Ph.D. study.

Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. Their continuous affectionate support and sacrifices are a very crucial factor for the successful completion of this Ph.D. study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

It is a challenging problem to make a general hand gesture recognition system work in a practical operation environment. In this study, it is mainly focused on recognizing English letters and digits performed near the steering wheel of a car and captured by a video camera. Like most human computer interaction (HCI) scenarios, the in-car gesture recognition suffers from various robustness issues, including multiple human factors and highly varying lighting conditions. It therefore brings up quite a few research issues to be addressed. First, multiple gesturing alternatives may share the same meaning, which is not typical in most previous systems. Next, gestures may not be the same as expected because users cannot see what exactly has been written, which increases the gesture diversity significantly. In addition, varying illumination conditions will make hand detection trivial and thus result in noisy hand gestures. And most severely, users will tend to perform letters at a fast pace, which may result in lack of frames for well-describing gestures. Since users are allowed to perform gestures in free-style, multiple alternatives and variations should be considered while modeling gestures. The main contribution of this work is to analyze and address these challenging issues step-by-step such that eventually the robustness of the whole system can be effectively improved. By choosing color-space representation and performing the compensation techniques for varying recording conditions, the hand detection performance for multiple illumination conditions is first enhanced. Furthermore, the issues of low frame rate and different gesturing tempo will be separately resolved via the cubic B-spline interpolation and i-vector method for feature extraction. Finally, remaining issues will be handled by other modeling techniques such as sub-letter stroke modeling. According to experimental results based on the above strategies, the proposed framework clearly improved the system robustness and thus encouraged the future research direction on exploring more discriminative features and modeling techniques.

CHAPTER 1

INTRODUCTION

Because of the rapid development of computing technologies, systems can now simultaneously acquire data from multiple sources of information for a wide range of existing and potential applications [1, 2]. Contemporary human computer interfaces may involve many different types of sensors; how to use the information collected by these sensors is important for most human machine interfaces and their induced applications. The possible information sources or sensors may involve microphones, cameras, and even thermal and pressure sensors. To understand the intention of users by their speech contents or their explicit body actions in a robust manner, proper feature extraction and modeling schemes followed by statistical learning tools for various user intentions should be carefully designed.

The main purpose for this study is to provide alternative information sources besides human speech to facilitate a natural controlling environment. Before going into more details, it is important to note that user interfaces for in-vehicle instrument control require not only good user experience but also minimal distraction due to safety reasons [3]. In such hands-busy and eyes-busy input conditions, gesture input plays a key role compared to speech input due to their complimentary nature that is able to provide side information that cannot be obtained by solely using speech input. However, to ensure enough reliability, the robustness issues for the gesture recognition should be studied.

In this chapter, the robustness concept will be briefly introduced first. And the target gesture of our interest is also defined. Next, by dealing with practical in-car robustness issues for gestures, the main contribution can be summarized. And finally, the thesis organization will be presented.

1.1 Robustness

For the purpose of building a reliable human computer interaction application, the system robustness under highly varying real-world environments is the most crucial factor to be considered. Many recent research efforts on automatic speech recognition (ASR) have enabled some real world applications with smart phones and hand-held devices [4].

The robustness framework [5], as shown in Figure 3, was proposed to deal with the robust ASR problem in signal, feature, or model domain such that the system is less sensitive to these distortion sources. By borrowing this concept, one can discuss the potential robustness problem for the automatic gesture recognition (AGR) problem as well. This will be discussed in depth in Chapter 2.4.

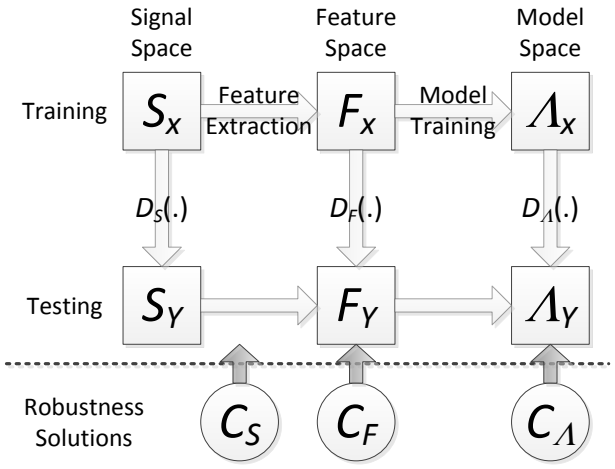


Figure 1: Framework of mismatches modeling and corresponding solutions.

1.2 Gesture of Interest

Contemporary gestures can be categorized into several different categories, based on their motion and posture variations. For this study, the "gesture" of our interest is the **i**solated in-car English letter or digit gesture recognition which belongs to the motion or trajectory-based gestures. As for the application environment, the in-car applications is the main focus. This choice is to relax the constrained in-door scenario to a more general one.

Recently, for in-vehicle applications, while most systems were done by control knobs or touch screens with proper display systems [6, 7] for user interactions, multiple modalities, such as speech and gesture recognition [4], also draws researchers' attentions [8, 4]. Although some technologies have been demonstrated to be quite useful, more research efforts are still needed for practical applications.

Most traditional gesture recognitions which may have variations on both trajectory and motion involve three important modules:

1. Moving object (hand in our application) detection and tracking.

2. Feature extraction to summarize its motion trajectory.

3. Modeling the gesture for future recognition.

However, unlike most studies for gesture recognition via image processing, which focus on well controlled experimental conditions such as controlled illumination, properly focused target, non-oscillating camera, well-constrained gesture writing styles, fixed gesture stroke orders, etc., many system constraints should be relaxed to match natural user experiences. Thus, in addition to the original complex variability of gesturing English letters, several new difficulties due to the various noise sources from the in-car recording environments should be overcome, which is our main research interest.

1.3 Robustness Issues for AGR

To understand the robustness issues, it is good to make reference to indoor state-of-the-art systems and see how far a naïve system is compared to these systems. Chen et al. [9] devised an in-office control gesture recognition, which can achieve 6.5% to 15% error rate. Recently, Chen et al. [10] devised a 40-control-gesture recognition system to achieve 3.1% error rate for set of data collected in a laboratory. And a previous designed English letter recognition system by Elmezain et al. [11] can also achieve 7.7% error rate. However, when directly applying a naïve setup on in-car gesture problem, a much worse error rate at

24.62% was observed. Therefore, the robustness issues are definitely required to be studied carefully. It is good to start from the nature of the gesture collected in a car.

Since our in-car gesture recognition is aimed at tackling realistic situations instead of the well-controlled conditions, several practical challenges may need to be taken into account:

1. Abnormal illumination conditions and the various shadow regions.
2. Automobile vibration.
3. Fast movement may result in insufficient frame rate.
4. User style variations.

In addition to above difficulties, users are prone to make wrong and ambiguous gestures compared to a similar task, hand-written character recognition. This is because users are not able to see the frames and the gesture trajectories while gesturing. Meanwhile, it is possible for a user to hesitate during the action or to write in different stroke orders. This possibility makes the problem even more complicated. With these difficulties, it is obvious that this study is facing very noisy data.

1.4 Main Contributions

According to the author's knowledge, this is the first work to explore the unconstrained in-car gesture recognition task for challenging environments. Several practical challenges and solutions that could be generalized to deal with other similar practical problems are proposed and verified. Although gesture recognition has many similarities compared with speech recognition, several challenges such as frame rate and other sparse issues are quite difficult and not observed in speech-related tasks. To sum up, by tackling robustness issues that will be discussed in Chapter 2.4, the main contributions of the work can be summarized into four major aspects:

1. Enhancing the hand detection robustness via adaptive modeling and block detection. The detail will be presented in Chapter 3.

2. Generalizing the model capability by stroke features; the details will be discussed in Chapter 4
3. Addressing data sparse issue with interpolation; the details will be shown in Chapter 4.
4. Casting the trajectory features to high level statistical features using i-vector; the discussion will be done in Chapter 5.

First, the traditional hand detection can involve background subtraction [12] and color or texture model [13]. However, for in-car hand detection, it is much easier to gain many false alarms, due to illumination changes, insufficient image contrast, and shadowing. It was proposed to use model adaptation to enhance the capability of detecting figure. Moreover, traditional detections are mainly pixel-based, block-based, or even blob or cluster-based. This work also went beyond these assumptions and explored alternative blocks based on local similarities instead of regular size grids.

Second, as mentioned previously, the performed gestures may involve high variation due to the user style and the fact that no trajectory feedback is shown to the user. Direct modeling the existing stroke order is the most obvious and intuitive option, but may restrict the generalization capability. Besides, there may be some meaningless strokes during moving-in and moving-out. The sub-unit group called "stroke" was defined to take care of these variabilities and "noisy part of gesture" issues. Moreover, multi-scale and multi-pass modeling techniques were discussed as well.

Last but not least, a proper feature set for different gesture recognition tasks was explored. Most previous gesture recognition systems utilized trajectory and object contour information to describe motion and shape changes overtime, mostly using hidden Markov model (HMM) [14] or finite state machine (FSM) [15]. This work studies the feasibility of different feature types, properly modifying the feature under low frame rate conditions by several interpolation strategies and their theoretical properties, and performing high dimensional projection of the original trajectories to further enhance the robustness.

1.5 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, the background and related tools for this study is discussed. In Chapter 3, the article illustrates the hand detection problem under realistic conditions and proposes adaptive scheme and data categorization strategies to enhance the robustness for hand detection. In Chapter 4, gesture trajectory modeling strategies, such as feature extraction and whole symbol and sub-symbol modelings, are discussed. The choice of interpolation strategies for low frame rate issues will also be presented. In Chapter 5, the enhanced feature and corresponding modeling schemes to tackle real-world challenges is presented. Finally, all the findings and future works are listed in Chapter 6. All the detailed hardware setups and more in-depth derivations are described in Appendices 7.

CHAPTER 2

BACKGROUND REVIEW

In this chapter, the system architecture of a traditional gesture recognition system is first introduced. After that, typical feature extraction techniques used for gesture recognition systems, which may or may not contain the object detection step for different applications, are reviewed. Next, several ways of feature extraction related to the present work are checked. Then, since the practical modeling methodologies will depend highly on the extracted feature procedures, several modeling technologies for using different types of extracted features are studied. These methods may include well-known hidden Markov model (HMM) and support vector machine (SVM). Finally, the article checks several feature space transformations as well as model space refinement techniques that are usually used for most gesture recognition systems under challenging recording conditions.

2.1 System Architecture for Gesture Recognition System

A gesture recognition is a commonly used technique for human computer interface related tasks. However, this term is usually understood as a broader concept, which can contain several categories. Depending on the desired subjects, they can categorized as follows:

- Posture recognition: Understand a hand or body posture for a single frame only.
- Gesture recognition: Understand the meaning of one or more moving objects, which may or may not contain shape and spatial variations.

For these systems, a schematic diagram is shown in Fig. 2.

Figure 2: The scheme for a general gesture recognition system

First, an image acquisition device may involve one or more cameras. Traditional camera or infrared camera can be used for this purpose. Next , depending on the system nature, the feature extraction may need some pre-processing, such as object detection, image denoising, or background subtraction. And finally, once the feature vectors are extracted, a recognizer that usually involves some statistical modeling technologies for dynamic or static data can be applied. The content will involve details of each module in the following paragraphs. First of all, different types of gesture will be discussed.

2.1.1 Gesture Categories

Gesture is a general concept that includes the posture and dynamic gesture. The former one is identified by a single frame, while the later may involve dynamic spatial and shape variations.

2.1.1.1 Posture

These types of gesture recognition systems are static, which work on data of single frame; no time-varying dynamic is taken into account. And mostly, the discrimination among postures is usually represented by different object shapes, such as different hand shapes [16, 17, 18]. Therefore, descriptors such as Fourier descriptor [19], neural network based descriptor[17], or contour descriptors [20] can be applied for the desired hand or body parts, which are frequently obtained by hand or body detectors based on color and texture clues. Since no temporal dynamic is taken into account, linear or non-linear classifier such as support vector machine (SVM) are usually applied.

2.1.1.2 Dynamic Gesture Recognition

A gesture involves temporal dynamics and is much more computationally expensive than the previously mentioned posture. In addition to the shape changes, frame-to-frame relationship as well as the varying components should be taken into account. Different strategies should be emphasized according to different applications.

2.1.1.3 Gesture with Time-Varying Location

The underlying semantic meanings of these gestures are embedded in their trajectories. Two commonly seen examples are the hand written text [21] and some pre-defined trajectories. For this type of gesture, the location and the angular measurements in the extracted trajectory are usually used as the feature vector once the gesturing hand or object is identified. To get the gesturing hands or body parts, frame-by-frame detection or tracking technologies such as particle filter [22] are frequently used. To do the recognition, the modeling tool that can handle temporal dynamics such as HMM or finite state machine (FSM) is often applied. This research focusing on the hand-written English letters and digits falls into this category.

2.1.1.4 Gesture with Time-Varying Object Shape

These gestures involve time-varying gesturing objects, usually hands. The most well-known one is the sign language recognition. And most likely, the gesturing hand can also involve the previously mentioned spatial variation. Generally, the feature used for this application should involve both the shape descriptors mentioned above and the spatial information such as the location and angular values from frame to frame. This application is the most complicated one compared to above-mentioned applications. Thus it is often studied under in-door application scenarios [22, 9, 23]. As for the recognizer building, these state-of-the-art systems still stick to the HMM or FSM based systems. The conditional random field (CRF) has also been proved to be useful.

Since the interest of the present study is the in-car English letter and digit recognition in the second category, in Section 2.2, the contemporary feature extraction tools on this category will be briefly reviewed.

2.2 Feature Extraction

Due to the computational issues as well as to enhance the discriminative power, most recognition systems will need to extract feature vectors from raw data. For gesture recognition,

the first step is usually the gesturing object detection. Unlike the automatic speech recognition (ASR), which performed transformations on raw data, the object detection is quite crucial. There are two main reasons for this problem. First, the speech signal cannot be easily decoupled from noise due to its nature of being a one dimensional temporal sequence. In order to tackle this issue, the most commonly used technique for speech recognition is the spectral subtraction or cepstral mean normalization (CMN). Conversely, the gesture recognition usually used the background subtraction, which is a similar technique, but proved to be of limited help for more challenging application scenarios, such as moving camera [24]. Second, the gesturing object or hand only occupies limited area in most image frames. That is, most raw data contribute nothing to the classification or recognition purposes. These are the main reasons why most systems will require the detection or tracking of the hand or body regions before doing the feature extraction.

2.2.1 Without Hand Detection and Tracking

Under a clean background scenario, it seems that hand detection is not required [25]. Scaling down the video frame and doing principal component analysis (PCA) on these raw data has been proved to be useful in this situation. However, for cluttered background and in-car application, since the background is quite complicated and not fixed, this variation may turn out to dominate the feature space. In the present internal testing, this type of method cannot generate meaningful recognition result, but only 10 % accuracy can be achieved. Therefore, the requirement of the hand detection in this problem is obvious. Next, how to perform the hand detection will be discussed.

2.2.2 Hand Detection and Tracking

The hand detection and tracking procedures are the crucial parts to determine the motion trajectory of a gesture. Their related techniques will be briefly reviewed.

2.2.2.1 Skin Color Model

Instead of directly working on the raw data, object detection is usually served as the first stage of these types of strategies. Since most of these works are targeted at hand gestures, skin and texture color analysis is the most commonly used method.

To determine skin and non-skin blocks or pixels, several color spaces are proposed, such as RGB, normalized RGB, LSV, and CIE Lab, etc. However, there is no globally optimal choice among the color spaces [26]. The choice of the color spaces are mainly based on the nature of the application and are mostly chosen empirically.

One of the most commonly used methods is to model skin and non-skin classes as Gaussian mixture models (GMMs) [27, 28], which model skin and non-skin patches as two sets of GMMs and comparing the conditional likelihood scores so that the likelihood ratio test can be used to determine if a patch belong to a skin patch or not. Given an image patch $\boldsymbol{O}$, and skin and non-skin color model parameter sets $\boldsymbol{\Lambda}_{skin}, \boldsymbol{\Lambda}_{non-skin}$, the likelihood scores can be computed by

$$P(\boldsymbol{O}|\boldsymbol{\Lambda}_{skin}) = \sum_{k=1}^{K} \omega_{skin,k} \mathcal{N}(\boldsymbol{O}|\mu_{skin,k}, \Sigma_{skin,k}), \tag{1}$$

$$P(\boldsymbol{O}|\boldsymbol{\Lambda}_{non-skin}) = \sum_{k=1}^{K} \omega_{non-skin,k} \mathcal{N}(\boldsymbol{O}|\mu_{non-skin,k}, \Sigma_{non-skin,k}), \tag{2}$$

where $\omega_{skin,k}$ and $\omega_{non-skin,k}$ are mixture weight within the range $[0, 1]$ such that $\sum_{k=1}^{K} \omega_{skin,k} = 1$ and $\sum_{k=1}^{K} \omega_{non-skin,k} = 1$.

Obviously, to enable the GMM for these applications involves the estimation of some latent variables which cannot be obtained directly from the data. That is, these parameters should be first estimated even if some of them are latent variables. This parameter estimation process for these models is usually done via expectation-maximization (EM) algorithm [29, 30] in the sense of maximum likelihood estimation (MLE).

EM algorithm

The standard EM algorithm is to iteratively approximate the MLE solution with latent variables or missing data in the sense of expectation on latent variables. Assume data set $\boldsymbol{O}$

is given with latent variable $\boldsymbol{S}$ and density parameter set $\boldsymbol{\Lambda}$, the EM algorithm will converge to MLE results [29, 30]. How this algorithm works in the following paragraph will be discussed.

Conceptually, EM algorithm includes two steps: the step of computing an expectation (E) of the likelihood by including the latent variables; and the step of computing the maximum likelihood estimations of the parameters by maximizing the expected log likelihood found in the E step. In fact, this procedure can be verified by using the following three steps.

1. Use the Jensen's inequality to draw a lower bound.
2. Find the best lower bound using an auxiliary function so that the lower bound will touch the objective function at the current guess.
3. Maximize the auxiliary function to obtain the new guess, then go to Step 2 until converge.

They will be sequentially described in the following content.

Draw a Lower Bound

Consider the total likelihood function

$$\begin{aligned} \log P(\boldsymbol{O}|\boldsymbol{\Lambda}) =& \log \sum_{t=1}^{T} P(\boldsymbol{O}|\boldsymbol{S}, \boldsymbol{\Lambda}) \\ =& \log\Big[\sum_{t=1}^{T} p^{\boldsymbol{\Lambda}}(\boldsymbol{S}) \frac{P(\boldsymbol{O}|\boldsymbol{S}, \boldsymbol{\Lambda})}{p^{\boldsymbol{\Lambda}}(\boldsymbol{S})}\Big] \end{aligned} \tag{3}$$

in which $p^{\boldsymbol{\Lambda}}(\boldsymbol{S})$ is an arbitrary probability distribution of the latent variable $\boldsymbol{S}$, and $\boldsymbol{\Lambda}$ is the whole parameter set. After applying the Jensen's inequality, due to the concavity of *log* function, the following lower bound function $g(\boldsymbol{\Lambda})$ of the objective function $F(\boldsymbol{\Lambda})$ can be obtained.

$$\begin{aligned} F(\boldsymbol{\Lambda}) = \log P(\boldsymbol{O}|\boldsymbol{\Lambda}) \geq & \sum_{t=1}^{T}\Big[p^{\boldsymbol{\Lambda}}(\boldsymbol{S}) \log \frac{P(\boldsymbol{O}|\boldsymbol{S}, \boldsymbol{\Lambda})}{p^{\boldsymbol{\Lambda}}(\boldsymbol{S})}\Big] \\ =& g(\boldsymbol{\Lambda}) \end{aligned} \tag{4}$$

Find the Best Lower Bound

Then, people find the arbitrary function $p^{\mathbf{\Lambda}}(\boldsymbol{S})$ that makes the lower bound function touch the object function at the current guessed point $\mathbf{\Lambda} = \overline{\mathbf{\Lambda}}$. This is equivalent to find the local maxima of the object function with respect to $p^{\mathbf{\Lambda}}(\boldsymbol{S})$. To obtain the best arbitrary function, one can use the fact that $\sum_{\mathbf{\Lambda}} p^{\mathbf{\Lambda}}(\boldsymbol{S}) = 1$ and use a Lagrange multiplier η to define the following objective function.

$$\eta(1 - \sum_{S} p^{\mathbf{\Lambda}}(\boldsymbol{S})) + \sum_{S} p^{\mathbf{\Lambda}}(\boldsymbol{S}) log[P(\boldsymbol{O}, \boldsymbol{S}|\overline{\mathbf{\Lambda}})] - \sum_{S} p^{\mathbf{\Lambda}}(\boldsymbol{S}) log(p^{\mathbf{\Lambda}}(\boldsymbol{S})) \tag{5}$$

Take the derivative with respect to $p^{\mathbf{\Lambda}}(\boldsymbol{S})$ and let it be zero will give the following equation:

$$-\eta + log(P(\boldsymbol{O}, \boldsymbol{S}|\overline{\mathbf{\Lambda}})) - log(p^{\mathbf{\Lambda}}(\boldsymbol{S})) - 1 = 0 \tag{6}$$

Use some algebraic manipulations to obtain the result.

$$p^{\mathbf{\Lambda}}(\boldsymbol{S}) = \Big[e^{-\eta} P(\boldsymbol{O}, \boldsymbol{S}|\overline{\mathbf{\Lambda}})\Big]/e \tag{7}$$

$$\sum_{S} p^{\mathbf{\Lambda}}(\boldsymbol{S}) = e^{-\eta}\Big[\sum_{S} P(\boldsymbol{O}, \boldsymbol{S}|\overline{\mathbf{\Lambda}})\Big]/e = 1 \tag{8}$$

Divided (7) by (8), the best $p^{\mathbf{\Lambda}}(\boldsymbol{S})$ is get as

$$p^{\mathbf{\Lambda}}(\boldsymbol{S}) = \frac{P(\boldsymbol{O}, \boldsymbol{S}|\overline{\mathbf{\Lambda}})}{\sum_S P(\boldsymbol{O}, \boldsymbol{S}|\overline{\mathbf{\Lambda}})} = \frac{P(\boldsymbol{O}, \boldsymbol{S}|\overline{\mathbf{\Lambda}})}{P(\boldsymbol{O}|\overline{\mathbf{\Lambda}})} = P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}}) \tag{9}$$

Thus the lower bound function can be re-written as:

$$\begin{aligned} g(\mathbf{\Lambda}, \overline{\mathbf{\Lambda}}) &= \sum_{S} P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}}) log\Big(\frac{P(\boldsymbol{O}, \boldsymbol{S}|\mathbf{\Lambda})}{P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}})}\Big) \\ &= \sum_{S} P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}}) log(P(\boldsymbol{O}, \boldsymbol{S}|\mathbf{\Lambda})) - \sum_{S} P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}}) log(P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}})). \end{aligned} \tag{10}$$

The only term that depends on the "to-be-estimate" model set $\mathbf{\Lambda}$, $\sum_S P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}}) log(P(\boldsymbol{O}, \boldsymbol{S}|\mathbf{\Lambda}))$, is then defined as the auxiliary function, denoted by $Q(\mathbf{\Lambda}|\overline{\mathbf{\Lambda}})$.

Assume one has $\mathbf{\Lambda}$ and probability $P(\boldsymbol{O}, \boldsymbol{S}|\mathbf{\Lambda})$ for each $\boldsymbol{S}$ occurred in the generation of $\boldsymbol{O}$. Then he can find a new $\mathbf{\Lambda} = \hat{\mathbf{\Lambda}}$ that maximizes the expectation $\sum_S P(\boldsymbol{S}|\boldsymbol{O}, \overline{\mathbf{\Lambda}}) log[P(\boldsymbol{O}|\boldsymbol{S}, \mathbf{\Lambda})]$.

Subsequently, following the above derivations (see Eq's. (3), (9), and (10)), it can be shown that

$$P(\boldsymbol{O}|\hat{\boldsymbol{\Lambda}}) \geq P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}}). \tag{11}$$

Basically, the EM algorithm can discover parameters of model $\boldsymbol{\Lambda}$ to maximize the log-likelihood of the incomplete data, $logP(\boldsymbol{O}|\boldsymbol{\Lambda})$, by iteratively maximizing the expectation of the log-likelihood of the complete data, $logP(\boldsymbol{O}, \boldsymbol{S}|\boldsymbol{\Lambda})$.

For the GMM parameter estimation, after initializing the parameter set $\boldsymbol{\Lambda}$, do

$$Estep: Q(\boldsymbol{\Lambda}|\boldsymbol{\Lambda}^{old}) = \sum_{\boldsymbol{S}} p(\boldsymbol{S}|\boldsymbol{O}, \boldsymbol{\Lambda}^{old}) ln\left[p(\boldsymbol{O}, \boldsymbol{Z}|\boldsymbol{\Lambda})\right], \tag{12}$$

$$Mstep: \boldsymbol{\Lambda}^{new} = arg\ \max_{\boldsymbol{\Lambda}} Q(\boldsymbol{\Lambda}|\boldsymbol{\Lambda}^{old}), \tag{13}$$

and then check the parameter convergence by comparing $\boldsymbol{\Lambda}^{new}$ and $\boldsymbol{\Lambda}^{old}$.

For GMM with K mixtures, the latent variable for its parameter learning is the mixture membership. Therefore, to evaluate the auxiliary function Q during E step, one should first evaluate the membership function with old parameters for each sample $\boldsymbol{o}_n$:

$$\gamma(s_{n,k}) = \frac{\omega_k \mathcal{N}(\boldsymbol{o}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \omega_j \mathcal{N}(\boldsymbol{o}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{14}$$

where $s_{n,k}$ is the membership variable of the n-th data for k-th mixture. Note this term is also the term $p(\boldsymbol{S}|\boldsymbol{O}, \boldsymbol{\Lambda}^{old})$ in EM formulation. During the M step, the parameter can be updated by tacking derivative of the Q function with Lagrange multiplier using the constraint that $\sum_{j=1}^{K} \omega_j = 1$ and set its derivative to zero. Subsequently, one obtains the following relations.

$$\boldsymbol{\omega}_k^{new} = \frac{\sum_{n=1}^{N} \gamma(s_{n,k})}{N}, \tag{15}$$

$$\boldsymbol{\mu}_k^{new} = \frac{\sum_{n=1}^{N} \gamma(s_{n,k}) \boldsymbol{o}_n}{\sum_{n=1}^{N} \gamma(s_{n,k})}, \tag{16}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{\sum_{n=1}^{N} \gamma(s_{n,k})(\boldsymbol{o}_n - \boldsymbol{\mu}_k^{new})(\boldsymbol{o}_n - \boldsymbol{\mu}_k^{new})^T}{\sum_{n=1}^{N} \gamma(z_{n,k})}, \tag{17}$$

EM algorithm can be applied on both hand and background skin pixels to train two separate sets of GMMs. Moreover, it uses the likelihood ratio test to judge if a specific pixel block belongs to a skin block.

2.2.2.2 Object Tracking

Contemporary gesture recognition systems may also add object tracking module. The objective of the object tracking is to robustly and efficiently detect the desired object in video sequences. The most commonly used method is the particle filtering technology [31] or Kalman filtering [32], which can be denoted conceptually as follows:

$$\boldsymbol{x}(t + \Delta t) = f(\boldsymbol{x}(t)) + \boldsymbol{\epsilon}, \tag{18}$$

$$\boldsymbol{y}(t + \Delta t) = h(\boldsymbol{x}(t)) + \boldsymbol{\delta}, \tag{19}$$

where $\boldsymbol{x}$ is the desired state variable such as the object location, $\boldsymbol{y}$ is the observed vector, the function $f(.)$ is the dynamic motion or state transition function, and $h(.)$ is the observation function. These functions form the core of the image tracking techniques because they describe how the object may move and how the potential motion can be observed. The difference between Kalman filter and particle filter is mainly on the relaxation of the linear model assumption and the probabilistic viewpoint. The particle filter can take nonlinear transition and observation functions and the whole model is viewed as a probabilistic model. Since the distribution after transition may be intractable, sampling is required to approximate the transitioned and observed data distribution. This requirement of the sampling procedure is the reason why in the statistical literature, particle filter was originally called "sequential Monte Carlo".

It goes without saying that the particle filter requires a good design on the transition and observation model. In fact, a rough dynamic or transition model will make the sampled particle die quickly and re-sampling is required. When this occurs too often, the tracking become inefficient. In the present problem, the cost of sampling particle and evaluating the

observation probability will be too inefficient due to this frame-rate issue. Therefore, this study will just use the frame-by-frame detection instead of doing tracking to achieve the real-time performance.

2.2.3 Feature

With detection and tracking modules, one can identify and extract features from the detected object sequence. Several following features are most commonly adopted:

- Position
  - Center of mass: position, velocity, and acceleration of object centroid.
  - Center of several tracked feature points.
- Trajectory
  - Angles with reference points.
- Image patch
  - Haar like features.
  - Texture features.
- Statistical Feature
  - Histogram of velocity

With one or more of above mentioned features, one can make them as input for the modeling tool so that the gesture can be recognized.

2.3 Gesture Recognizer

When a posture is the desired recognition target, static modeling tools such as support vector machine (SVM) or artificial neural network (ANN) are usually used. While working on the dynamic gesture, the hidden Markov model (HMM) or finite state machine (FSM) are usually used. Their details will be presented in the following paragraphs.

2.3.1 Support Vector Machine

The SVM [33] is widely used in information retrieval problems [34, 35]. Its soft margin version with regularized penalty term is most widely used, which is shown in the following:

$$\min_{\boldsymbol{w},\xi,b}\{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i\}$$
$$\text{subject to } y_i(\boldsymbol{w}\cdot\boldsymbol{x} - b) \geq 1 - \xi_i, \forall i = 1, ..., N, \tag{20}$$

where $y_i \in \{-1, 1\}$ is the class label of the i^{th} sample for a specific target class. Generally, multiple SVMs are used for a multi-class problem. Once the feature vector $\boldsymbol{x}$ was extracted, it could be applied following the regular optimization problem with gradient descent. This study applied this linear SVM classifier implemented by the LIBSVM tool [36]. Note there are various kernelized SVMs that can be also adopted because they can be directly applied on kernelized space.

2.3.2 Artificial Neural Network

Recently, a set of artificial neural networks (ANN) drew high attention due to its good performance. These modified ANN are generally called deep neural networks (DNN). As compared to the traditional artificial neural networks, the biggest difference is its initialization procedure called pre-training [37]. Also, when the classification is the target, the cross-entropy is the mostly adopted cost function for DNN training [37]. The main drawback of these kinds of "deep learning" is the requirement of large training data. Therefore, the current research focuses mostly on how to change the training strategy when limited training data are available, such as freezing some sub-networks and updating only parameters of several layers [37].

ANN is usually formulated by defining the network structure believed to be able to describe the desired feature space very well. Then the parameter is updated via "back-propagation", which is based on chain-rules. Assume we are working on an augmented feature space formed by $\boldsymbol{\chi} = [1\ \boldsymbol{x}]^T$. Each neuron i is formed by weighting the input vector

$\boldsymbol{\chi}_i$ with weight vector $\boldsymbol{w}_i$, and applying activation function, which is usually done with sigmoid-shape function such as logistic sigmoid,

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \tag{21}$$

where a is a function of $\boldsymbol{\chi}$ typically done by an affine transform $\boldsymbol{W}^T\boldsymbol{\chi}$. The weight vector is used to specify the connection between each successive layers. The back-propagation is done by using chain-rule to do the gradient descent algorithm. Assume the sub-network node with weight w to be estimated via the cost function or measure $\mathcal{E}$, it typically takes the form as the mean-squared error (MSE) compared to the target output or cross entropy. If the sub-network output vector is z_j, $j = 1, \ldots, J$, and the next layer has output $y_k, k = 1, \ldots, K$, the gradient is evaluated as

$$\frac{\partial \mathcal{E}}{\partial w} = \sum_{j=1}^{J} \sum_{k=1}^{K} \frac{\partial \mathcal{E}}{\partial y_k} \frac{\partial y_k}{\partial z_j} \frac{\partial z_j}{\partial w}, \tag{22}$$

where each term can be computed by considering the derivative of the output of each neuron with respect to its input by using an analytic cost function $\mathcal{E}$. Also, the network output layer is usually scaled with a softmax function to approximate the posterior probability for each target class.

2.3.3 Hidden Markov Model

Due to the temporal sequence nature of dynamic gestures, HMM [14] is a convenient tool to serve this purpose. The left-to-right topology is the most commonly adopted one. While the state-jump configuration is not allowed, it is required to visit each state in a left to right order at least once. Just as how it was used in automatic speech recognition (ASR) applications, one does not know and cannot directly observe which "parts" of these gesture trajectories or motions should belong to which state. For gesture trajectory modeling, each state may represent part of the trajectory components, such as a straight line, a curve, or near the transition of two curves. It is generally believed that a finer HMM can be built to characterize gestures better when enough training samples are at hand. Unlike ASR task,

however, an insufficient acquisition frame rate will usually make it very difficult to build such kinds of models.

The main parameters for HMM are composed of three parts, namely the state initial probability, transition probability, and state-emission probability. There are several kinds of HMM categorized by their state-emission probabilities, such as continuous density HMM (CDHMM), discrete HMM, or semi-continuous HMM. Due to the nature of the noisy recording conditions, there will be an infinite number of possibilities for the detected object location. Therefore, CDHMM was chosen to model letter or digit gestures. For a better description purpose, the following terminologies are used:

- t: Time index, which takes integer values: $0, 1, \ldots, T$.
- N: Number of state for each HMM, all HMMs are assumed to share the same number of states.
- s_t: The state at time t, which is "hidden", we denote the state sequence by $S = \{s_1, s_2, \ldots, s_T\}$ in the following discussion.
- $\boldsymbol{o_t}$: The observed feature vector at time t, we denote the observation sequence by $\boldsymbol{o}_1^T = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T\}$ in the following discussion.
- M: Number of symbols to be modeled in the system.
- $\Lambda = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}\}$: Set of HMM parameters, include the following discussion:
  - $\boldsymbol{\pi} = \pi_i, i = 1, 2, \ldots, N$: Initial state probability, that is, $\pi_i = P(s_1 = i)$.
  - $\boldsymbol{A} = a_{i,j}, i, j = 1, 2, \ldots, N$: The state transition probability matrix, where $a_{i,j}$ is the probability of a transition from state i to state j, such that matrix $\boldsymbol{A}$ is $N \times N$ square matrix.
  - $\boldsymbol{B} = b_j(\boldsymbol{o}_t), j = 1, 2, \ldots, N$: The state observation probability, also known as state emission probability, which is the probability of observing feature vector $\boldsymbol{o}_t$ given state j, $P(\boldsymbol{o}_t | s_t = j)$.

To use the HMM for recognition purposes, there are three fundamental problems, namely:

1. How to compute the probability $P(\boldsymbol{o}_1^T|\Lambda)$.
2. How to decode the best state sequence $\max_S P(\boldsymbol{o}_1^T, S|\Lambda)$.
3. How to train the parameter set Λ.

From a practical viewpoint, one needs to solve the problem 3 so that HMM can be used as a modeling tool. To do so, the Baum-Welch algorithm, which is a kind of EM algorithm, is usually used. In order to use the EM algorithm, he would need to know how to evaluate $P(\boldsymbol{o}_1^T|\Lambda)$ for its E-step. After the model is trained to converge, he will need to find the model that most likely generated the observed data. This can be done approximately by solving problem 2, where the probability of the most possible state sequence is used to approximate the likelihood of the model generating the data. Finally, problem 2 can be solved by using the Viterbi algorithm; the corresponding probability for the best path is used for recognition purposes. Details of these problems will be described in following subsections. Now the article will first discuss how to solve problem 1 and assume the HMM follows the left-to-right configuration without state skipping.

2.3.3.1 Left to Right HMM without State Skip

Assume $\pi_1 = 1$ and $\pi_2 = \pi_3 = \ldots = \pi_N = 0$ and $a_{i,j}$ are initially guessed following the rule stated below. Assume HMM only allows the left to right transition without skipping any state, which is denoted as:

$$a_{i,j} = \begin{cases} 0 & j \neq i \wedge j \neq i+1 \\ a_{i,j} & j = i \vee j = i+1, \ s.t. \ a_{i,i} + a_{i,i+1} = 1 \end{cases} \tag{23}$$

Then the observed data can be evenly divided into N segments (also known as "flat start") to train the N sets of Gaussian mixture models (GMMs) with these corresponding segments so as to estimate parameters of b_j. Thus the initial parameters $\boldsymbol{\Lambda}$ are roughly estimated.

2.3.3.2 *Evaluation Problem of HMM*

To evaluate the probability $P(\boldsymbol{o}_1^T|\Lambda)$, given a set of HMM $\boldsymbol{\Lambda}$, the most intuitive way is to sum up all possible state sequences S by using the rule of total probability:

$$P(\boldsymbol{o}_1^T|\Lambda) = \sum_{\forall S} P(\boldsymbol{o}_1^T|S,\Lambda)P(S|\Lambda), \tag{24}$$

where

$$P(\boldsymbol{o}_1^T|S,\Lambda) = \prod_{t=1}^{T} P(\boldsymbol{o}_t|s_t,\Lambda) = \prod_{t=1}^{T} b_{s_t}(\boldsymbol{o}_t), \tag{25}$$

$$P(S|\Lambda) = P(s_1|\Lambda)\prod_{t=2}^{T} P(s_t|s_{t-1},\Lambda) = \pi_{s_1}\prod_{t=2}^{T} a_{s_{t-1},s_t}, \tag{26}$$

After combining Eq. (25) and (26), one has

$$P(\boldsymbol{o}_1^T|\Lambda) = \sum_{s_1,s_2,\ldots,s_T} \pi_{s_1} b_{s_1}(\boldsymbol{o}_1)a_{s_1,s_2}b_{s_2}(\boldsymbol{o}_2)a_{s_2,s_3}b_{s_3}(\boldsymbol{o}_3)\ldots a_{s_{T-1},s_T}b_{s_T}(\boldsymbol{o}_T). \tag{27}$$

It is obvious that to directly compute this probability with all possible considerings is too expensive. The forward-backward algorithm is used to overcome this difficulty.

To run the forward algorithm, previous researchers first define the forward variable:

$$\alpha_t(i) = P(\boldsymbol{o}_1^t, s_t = i|\Lambda), \tag{28}$$

which is the probability of observing first t feature vectors and the state at time t being i given the HMM $\boldsymbol{\Lambda}$. The forward iterative procedure is done by visiting from $t = 1$ all the way to $t = T$:

- Initialization:

$$\alpha_1(i) = \pi_i b_i(\boldsymbol{o}_1), 1 \le i \le N. \tag{29}$$

- Induction, for $t = 2, \ldots, T$

$$\alpha_t(i) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i)a_{i,j}\right] b_i(\boldsymbol{o}_t), 1 \le i \le N. \tag{30}$$

- Termination:

$$P(\boldsymbol{o}_1^T|\Lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{31}$$

And similarly, a backward procedure can be done from $t = T$ all the way back to $t = 1$, with backward variable defined as:

$$\beta_t(i) = P(\boldsymbol{o}_{t+1}^T|s_t = i, \Lambda), \tag{32}$$

which is the probability of observing the remaining vectors after time t, given the state at time t being i and HMM $\boldsymbol{\Lambda}$. The backward procedure is summarized as follows:

- Initialization:

$$\beta_T(i) = 1, 1 \leq i \leq N. \tag{33}$$

- Induction, for $t = T, \ldots, 1$

$$\beta_t(i) = \sum_{j=1}^{N} a_{i,j} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(i), 1 \leq i \leq N. \tag{34}$$

- Termination:

$$P(\boldsymbol{o}_1^T|\Lambda) = \sum_{i=1}^{N} \beta_0(i). \tag{35}$$

2.3.3.3 Decoding Problem of HMM

As mentioned above, the training procedure requires the evaluation of the probability $P(\boldsymbol{o}_1^T|\Lambda)$. For the recognition purposes, investigators usually use the probability corresponding to best path $\max_S P(\boldsymbol{o}_1^T, S|\Lambda)$ to make the decision, which can be computed more efficiently with Viterbi decoding algorithm. This is also viewed as an approximation of the forward algorithm mentioned above. Then, they define the variables for the most likely state sequence probability which has generated the observation vector $\boldsymbol{o}_1^t$ and ends in the state i at time t given the model, $\delta_t(i)$. Also, the tracking variable for this most likely

sequence $\psi_t(i)$ can be defined as:

$$\delta_t(i) = \max_{s_1^{t-1}} P(\boldsymbol{o}_1^t, s_1^{t-1}, s_t = i|\Lambda), \tag{36}$$

$$\psi_t(i) = arg \max_{s_1^{t-1}} P(\boldsymbol{o}_1^t, s_1^{t-1}, s_t = i|\Lambda). \tag{37}$$

Then the Viterbi algorithm with the following recursion equations can be done:

- Initialization:

$$\delta_1(i) = \pi_i b_i(\boldsymbol{o}_1), 1 \leq i \leq N, \tag{38}$$

$$\psi_1(i) = 0. \tag{39}$$

- Recursion:

$$\delta_t(i) = \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{i,j} b_j(\boldsymbol{o}_t)\right], 2 \leq t \leq T, 1 \leq i, j \leq N, \tag{40}$$

$$\psi_t(i) = arg \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{i,j}\right], 2 \leq t \leq T, 1 \leq i, j \leq N. \tag{41}$$

- Termination:

$$P^* = \max_{1 \leq i \leq N} \left[\delta_T(i)\right], \tag{42}$$

$$s_T^* = arg \max_{1 \leq i \leq N} \left[\delta_T(i)\right]. \tag{43}$$

- Backtracking:

$$s_t^* = \psi_{t+1}(s_{t+1}^*), t = T-1, T-2, \ldots, 1. \tag{44}$$

After running the Viterbi algorithm, the best state sequence $S^* = \{s_1^*, s_2^*, \ldots, s_T^*\}$ and best score P^* can be obtained so that the decoding problem is solved.

2.3.3.4 Training Problem of HMM

To train the HMM for practical application, since both actual state sequence and mixture membership are unknown, the EM algorithm should be applied to enable the estimation of HMM parameters. In the subsequent sub-sections, this study will discuss how to use the EM algorithm for the CDHMM (HMM with GMM state emission probability, also known as GMM-HMM) to solve its parameter estimation problem.

2.3.3.5 The EM Algorithm of HMM

Define the auxiliary function as:

$$Q(\boldsymbol{\Lambda}, \overline{\boldsymbol{\Lambda}}) = \sum_{\boldsymbol{S}} [P(\boldsymbol{S}|\boldsymbol{O}, \overline{\boldsymbol{\Lambda}}) log(P(\boldsymbol{O}, \boldsymbol{S}|\boldsymbol{\Lambda}))] \tag{45}$$

$$= \sum_{\boldsymbol{S}} \Big[\frac{P(\boldsymbol{O}, \boldsymbol{S}|\overline{\boldsymbol{\Lambda}})}{P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}})} log(P(\boldsymbol{O}, \boldsymbol{S}|\boldsymbol{\Lambda})) \Big],$$

and use the following definition:

$$P(\boldsymbol{O}, \boldsymbol{S}|\overline{\boldsymbol{\Lambda}}) = \overline{\boldsymbol{\pi}}_{s_1} \Big[\boldsymbol{\Pi}_{t=1}^{T-1} \overline{a}_{s_t s_{t+1}} \Big] \Big[\boldsymbol{\Pi}_{t=1}^{T} \overline{b}_{s_t}(\boldsymbol{o}_t) \Big] \tag{46}$$

$$= \overline{\boldsymbol{\pi}}_{s_1} \Big[\boldsymbol{\Pi}_{t=1}^{T-1} \overline{a}_{s_t s_{t+1}} \Big] \Big[\boldsymbol{\Pi}_{t=1}^{T} \sum_{m_t=1}^{C} \overline{\omega}_{s_t, m_t} \overline{b}_{s_t, m_t}(\boldsymbol{o}_t) \Big], \tag{47}$$

where $\overline{b}_{s_t, m_t}$ is the corresponding current estimate at time t given by mixture m_t of state s_t, and $\overline{\omega}_{s_t, m_t}$ is the corresponding mixture weight. Considering a specific mixture sequences $\boldsymbol{M} = m_1, m_2, \ldots, m_T$, one can make the following definition:

$$P(\boldsymbol{O}, \boldsymbol{S}, \boldsymbol{M}|\overline{\boldsymbol{\Lambda}}) = \overline{\boldsymbol{\pi}}_{s_1} \Big[\boldsymbol{\Pi}_{t=1}^{T-1} \overline{a}_{s_t s_{t+1}} \Big] \Big[\boldsymbol{\Pi}_{t=1}^{T} \overline{\omega}_{s_t, m_t} \overline{b}_{s_t, m_t}(\boldsymbol{o}_t) \Big], \tag{48}$$

such that

$$\sum_{\boldsymbol{S}} \sum_{\boldsymbol{M}} P(\boldsymbol{O}, \boldsymbol{S}, \boldsymbol{M}|\overline{\boldsymbol{\Lambda}}) = P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}}). \tag{49}$$

Then he has

$$logP(\boldsymbol{O}, \boldsymbol{S}, \boldsymbol{M}|\boldsymbol{\Lambda}) = log\boldsymbol{\pi}_{s_1} + \Big[\sum_{t=1}^{T-1} loga_{s_t s_{t+1}} \Big] + \sum_{t=1}^{T} log\omega_{s_t, m_t}$$
$$+ \Big[\sum_{t=1}^{T} logb_{s_t, m_t}(\boldsymbol{o}_t) \Big]. \tag{50}$$

The auxiliary function becomes

$$Q(\boldsymbol{\Lambda}, \overline{\boldsymbol{\Lambda}}) = \sum_{\boldsymbol{S}} \sum_{\boldsymbol{M}} \frac{P(\boldsymbol{O}, \boldsymbol{S}, \boldsymbol{M}|\overline{\boldsymbol{\Lambda}})}{P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}})} \Big\{ log\boldsymbol{\pi}_{s_1} + \Big[\sum_{t=1}^{T-1} loga_{s_t s_{t+1}} \Big] +$$
$$\sum_{t=1}^{T} log\omega_{s_t, m_t} + \Big[\sum_{t=1}^{T} logb_{s_t, m_t}(\boldsymbol{o}_t) \Big] \Big\}$$
$$= Q_{\boldsymbol{\pi}}(\boldsymbol{\pi}, \overline{\boldsymbol{\pi}}) + Q_{\boldsymbol{a}}(\boldsymbol{a}, \overline{\boldsymbol{a}}) + Q_{\boldsymbol{\omega}}(\boldsymbol{\omega}, \overline{\boldsymbol{\omega}}) + Q_{\boldsymbol{b}}(\boldsymbol{b}, \overline{\boldsymbol{b}}), \tag{51}$$

where

$$Q_{\pi}(\boldsymbol{\pi}, \overline{\boldsymbol{\pi}}) = \sum_{i=1}^{N}\Big[\frac{P(\boldsymbol{O}, \boldsymbol{s}_1 = i|\overline{\boldsymbol{\Lambda}})}{P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}})} log\boldsymbol{\pi}_i\Big] = \sum_{i=1}^{N}\Big[\overline{\gamma}_1(i) log\pi_i\Big], \tag{52}$$

$$\begin{aligned} Q_{\boldsymbol{a}}(\boldsymbol{a}, \overline{\boldsymbol{a}}) &= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T-1}\Big[\frac{P(\boldsymbol{O}, \boldsymbol{s}_t = i, \boldsymbol{s}_{t+1} = j|\overline{\boldsymbol{\Lambda}})}{P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}})} log a_{i,j}\Big] \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T-1}\Big[\overline{\xi}_t(i, j) log a_{i,j}\Big], \end{aligned} \tag{53}$$

$$\begin{aligned} Q_{\omega}(\omega, \overline{\omega}) &= \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{C}\Big[\frac{P(\boldsymbol{O}, s_t = i, m_t = k|\overline{\boldsymbol{\Lambda}})}{P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}})} log\omega_{i,k}\Big] \\ &= \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{C}\Big[\overline{\gamma}_t(i, k) log\omega_{i,k}\Big], \end{aligned} \tag{54}$$

$$\begin{aligned} Q_{\boldsymbol{b}}(\boldsymbol{b}, \overline{\boldsymbol{b}}) &= \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{C}\Big[\frac{P(\boldsymbol{O}, s_t = i, m_t = k|\overline{\boldsymbol{\Lambda}})}{P(\boldsymbol{O}|\overline{\boldsymbol{\Lambda}})} log b_{i,k}(\boldsymbol{o}_t)\Big] \\ &= \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{k=1}^{C}\Big[\overline{\gamma}_t(i, k) log b_{i,k}(\boldsymbol{o}_t)\Big]. \end{aligned} \tag{55}$$

where ξ_t and γ_t are parameters that will be defined latter in Eq.(62) and Eq.(61), respectively. The last equations show that the auxiliary function is separated into four terms, each respectively named as $\pi_i, a_{i,j}, \omega_{i,k}$ and $b_{j,k}(\boldsymbol{o}_t)$. For convenience, the maximization procedure on $Q(\boldsymbol{\Lambda}, \overline{\boldsymbol{\Lambda}})$ is done by maximizing the individual terms separately subject to probability constraints, say,

$$\begin{aligned} \sum_{i=1}^{N}\pi_i &= 1, \\ \sum_{j=1}^{N} a_{i,j} &= 1, \quad \forall i, \\ \sum_{k=1}^{M}\omega_{j,k} &= 1, \quad \forall j. \end{aligned} \tag{56}$$

After careful inspecting the above relations and followed by a suitable proof procedure, it can be shown that all these terms involving the above Lagrange multipliers have the following form:

$$F(\boldsymbol{y}) = g(y_1, y_2, \ldots, y_N) = \sum_{j=1}^{N} w_j log(y_j), \text{ where } \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{y_j} = 1, \ \& \ \mathrm{y_j} \geq 0$$

$$F(\boldsymbol{y}) \text{ has a maximum value when } \mathrm{y_j} = \left[\mathrm{w_j} / \sum_{\mathrm{n=1}}^{\mathrm{N}} \mathrm{w_n}\right] \tag{57}$$

Also note for $Q_{\boldsymbol{b}}(\boldsymbol{b}, \overline{\boldsymbol{b}})$ with $\boldsymbol{b}$'s modeled by GMM, one has:

$$b_{i,k}(\boldsymbol{o}_t) = \frac{1}{(2\pi)^{\frac{D}{2}} det|\boldsymbol{\Sigma}_{i,k}|^{\frac{1}{2}}} exp\left[-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_{i,k})^T \boldsymbol{\Sigma}_{i,k}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{i,k})\right], \boldsymbol{o}_t \in \mathbb{R}^D, \tag{58}$$

$$log(b_{i,k}(\boldsymbol{o}_t)) = -\frac{D}{2} log(2\pi) - \frac{1}{2} log(det|\boldsymbol{\Sigma}_{i,k}|) - \frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_{i,k})^T \boldsymbol{\Sigma}_{i,k}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{i,k}). \tag{59}$$

Now, define the following variables:

$$\begin{aligned}
\gamma_t(i) &= P(s_t = i|\boldsymbol{o}_t, \boldsymbol{\Lambda}) \\
&= \frac{P(\boldsymbol{o}_t, s_t = i|\boldsymbol{\Lambda})}{P(\boldsymbol{o}_t|\boldsymbol{\Lambda})} \\
&= \frac{\alpha_t(i)\beta_t(i)}{P(\boldsymbol{o}_t|\boldsymbol{\Lambda})} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)},
\end{aligned} \tag{60}$$

$$\begin{aligned}
\gamma_t(i,k) &= P(s_t = i|\boldsymbol{o}_t, \boldsymbol{\Lambda}) P(m_t = k|s_t = i) \\
&= \gamma_t(i) \frac{\omega_k \mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{i,k}, \boldsymbol{\Sigma}_{i,k})}{\sum_{h=1}^{C} \mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{i,h}, \boldsymbol{\Sigma}_{i,h})}
\end{aligned} \tag{61}$$

$$\begin{aligned}
\xi_t(i,j) &= P(s_t = i s_{t+1} = j|\boldsymbol{o}_t, \boldsymbol{\Lambda}) \\
&= \frac{P(s_t = i s_{t+1} = j, \boldsymbol{o}_t|\boldsymbol{\Lambda})}{P(\boldsymbol{o}_t|\boldsymbol{\Lambda})} = \frac{\alpha_t(i) a_{i,j} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_t(m) a_{m,n} b_n(\boldsymbol{o}_{t+1} \beta_{t+1}(n)},
\end{aligned} \tag{62}$$

where $\gamma_t(i,k)$ considers both the state membership at time t as $\gamma_t(i)$ and the corresponding Gaussian mixture membership at mixture k, so obviously $\sum_{k=1}^{C} \gamma_t(i,k) = \gamma_t(i)$. Then, taking the local extreme points by setting the derivative/gradient result to zero, the HMM

parameters are re-estimated as:

$$\hat{\boldsymbol{\pi}}_i = \overline{\gamma}_1(i) \tag{63}$$

$$= \text{expected number of times in state i at time t = 1}$$

$$\hat{a}_{i,j} = \left[\sum_{t=1}^{T-1} \overline{\xi}_t(i,j)\right] / \left[\sum_{t=1}^{T-1} \overline{\gamma}_t(i)\right] \tag{64}$$

$$= \frac{\text{expected number of transitions from state i to state j}}{\text{expected number of transitions from state i}},$$

$$\hat{\omega}_{i,k} = \frac{\sum_{t=1}^{T} \overline{\gamma}_t(i,k)}{\sum_{m=1}^{C} \sum_{t=1}^{T} \overline{\gamma}_t(i,m)} \tag{65}$$

$$= \frac{\text{expected number of times in state i and mixture k}}{\text{expected number of times in state i}},$$

$$\hat{\boldsymbol{\mu}}_{i,k} = \frac{\sum_{t=1}^{T} \overline{\gamma}_t(i,k)\boldsymbol{o}_t}{\sum_{t=1}^{T} \overline{\gamma}_t(i,m)} \tag{66}$$

$$= \text{weighted average of observations at state } i \text{ and mixture } k,$$

$$\hat{\boldsymbol{\Sigma}}_{i,k} = \frac{\sum_{t=1}^{T} \overline{\gamma}_t(i,k)(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{i,k})(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_{i,k})^T}{\sum_{t=1}^{T} \overline{\gamma}_t(i,m)} \tag{67}$$

$$= \text{weighted covariance of observations at state } i \text{ and mixture } k,$$

2.3.4 Finite State Machine

FSM [15] is similar to HMM, but the modeling target is not the direct gesture sequence, some parts of gesture components are used to build the gesture. This technique is usually applied as a language model state space constraint. Several state components that may have semantic meaning such as “words” are defined and how these words should be connected are fed into a rule-based automaton. In this work, single letter or digit can thus be modeled individually. Therefore, instead of explicitly defining a set of states and the corresponding rule based constraints, this study simply discusses the stroke model and uses the dictionary on how these words should be combined.

2.3.5 Other Tools

There are still other modeling tools for dynamic gesture recognition. For example, the Markov random field with shared hidden states can be used. But so far, for most dynamic recognition tasks, most existing systems use HMM [38, 9] with position related feature as their benchmarks.

2.4 Practical Gesture Recognition Issues

In previous sections, the existing gesture recognition features and models as well as the most commonly used technologies were reviewed. However, most of above mentioned works had their data recorded under indoor environments, which did not consider highly diversified environments. To apply a system under practical conditions, there are some existing issues. In this section, the illumination and cluttered background issue are first visited. After that, the low frame rate issues are reviewed. Finally, practical gestures may allow more gestures as well as different gesturing orders. This issue is seldom discussed in contemporary gesture recognition systems, but should be quite crucial for practical application, so it will be briefly introduced before ending this chapter.

2.4.1 Robust recognition framework

The robustness framework was introduced in [5, 39] to model the mismatches between training and testing conditions, where mismatches can be observed at signal, feature, and model spaces, as shown in Figure 1. First, suppose the signal spaces for training and testing data, S_X and S_Y, respectively, have mismatch or distortion function $D_S(.)$. In the stage of feature extraction, the feature spaces for training and testing data, F_X and F_Y, respectively, have corresponding distortion function $D_F(.)$. And the model spaces for training and testing data, Λ_X and Λ_Y, respectively, have distortion function $D_\Lambda(.)$. To compensate for these distortions, people may apply one of the algorithms C_S, C_F, or C_Λ on either signal space, feature space, or model space.

From the aforementioned robustness scheme, one can now discuss the robustness framework for the gesture recognition system. To facilitate the understanding of the robustness issue, especially for the in-car gesture recognition system, this study reviews the gesture acquisition process by the conceptual diagram shown in Figure 3. As seen in this Figure, there are several transformations between the intended letter trajectory $\mathbf{V}_{train}$ and the extracted trajectory $\mathbf{V}_{3,train}$. If one carefully reviews the whole procedure as successive mathematical processes, he may find solutions to compensate for the potential mismatches within the training data and those between training and testing data.

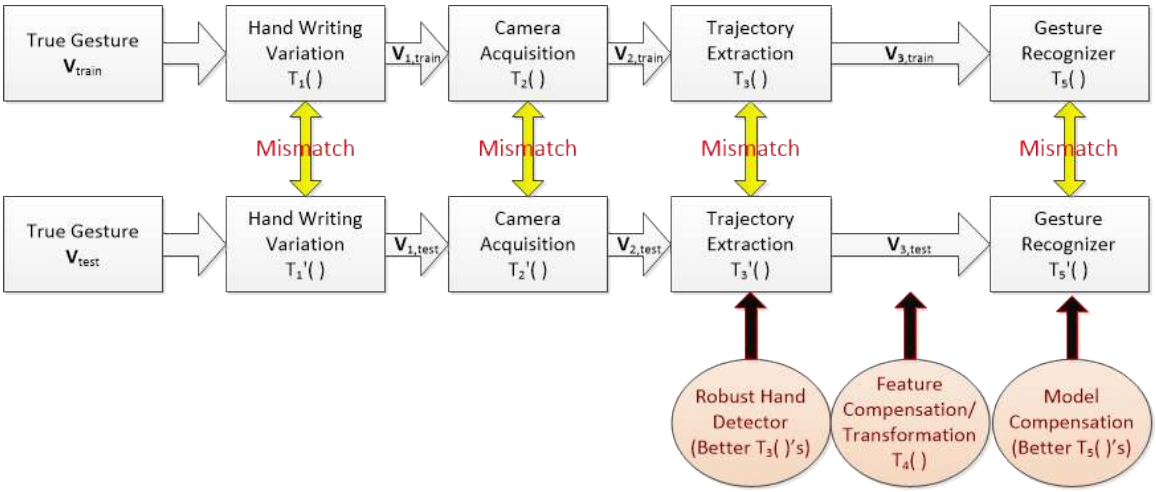


Figure 3: The scheme for robust general gesture recognition system.

Firstly, it is certain that the same letter may be defined by a set of different letter forms. To write a printed letter, actual writing forms mainly depend on writing habits that differ from user to user. Consequently, even the standard letter in a person's mind can still have a certain degree of variation. For the sake of simplicity, let us assume that these variations do not disturb the clear differences between different letters.

For convenience, let the data points $f(x_0, y_0), f(x_1, y_1), \ldots, f(x_n, y_n)$, form an original vector set $\boldsymbol{V}$. This study uses many successive parameter space transformations to modify this vector, say $\boldsymbol{V}_1 = \boldsymbol{T}_1(\boldsymbol{V})$, $\boldsymbol{V}_2 = \boldsymbol{T}_2(\boldsymbol{V}_1)$, $\boldsymbol{V}_3 = \boldsymbol{T}_3(\boldsymbol{V}_2), \ldots$. If the $i-$th transformation is linear, it takes the form $\boldsymbol{V}_i = \widetilde{T}_i \cdot \boldsymbol{V}_{i-1}$. Each transformation defined as $\boldsymbol{T}_i$ may or may not be properly designed so that the resulting model probability distribution obtained from the

training data may have some mismatches in feature and model parameter spaces.

To be more specific, how the mismatch will occur during the gesturing procedure will be shown as follows. Before a person writes a letter, the assigned letter form and stroke order were presented to him or her. Since the data acquisition system does not involve a feedback, the person does not know precisely what gestures are performed and the resulting outputs are correct or not. Moreover, the data acquisition procedure required two hours of a person's time. It seems that after 30 minutes or less, the person usually became tired so that he may have forgotten the letter form and stroke order. Therefore, the hand-written letter is transformed from $\boldsymbol{V}$ to be $\boldsymbol{V}_1 = \boldsymbol{T}_1(\boldsymbol{V})$. This output involves many uncertainties and may not be easily improved in the post processing stage.

The in-car camera system was used to acquire many hand-written letters over periods in different day time slots. Namely, the illumination conditions are subject to change. Since the person cannot see the resulting letter, the size, shape, and stroke are also variable. Consequently, the letter becomes $\boldsymbol{V}_2 = \boldsymbol{T}_2(\boldsymbol{V}_1)$. Again, the probability distribution of $\boldsymbol{V}_2$ is assumed to be non-adjustable, as the produced trajectories were not fed back to users doing those gestures.

After the data acquisition process was completed, several image recognition models were applied to read the letter from a series of frames captured by the camera. At first, the hand region was identified and then its center is considered as the trajectory point of the letter. The resulting trajectory is $\boldsymbol{V}_3 = \boldsymbol{T}_3(\boldsymbol{V}_2)$. One can try to adjust $\boldsymbol{T}_3$ to make the difference term $d(\boldsymbol{V}_3, \boldsymbol{V}_2)$ and difference between their probability distributions are as small as possible, where $d(.)$ can be arbitrary difference measurement such as Euclidean distance, log likelihood ratio, or KullbackLeibler (KL) divergence.

After the previous procedure, the recognition model built with the processed training data set was used to recognize a letter of the test data set. For convenience, $\boldsymbol{V}_{3t}$ and $\boldsymbol{V}_{3r}$ are the letters taken from the training and test sets, respectively. A recognition process imposes a new transformation so that these vectors become $\boldsymbol{V}_{4t} = \boldsymbol{T}_4(\boldsymbol{V}_{3t})$ and $\boldsymbol{V}_{4r} = \boldsymbol{T}_4(\boldsymbol{V}_{3r})$. In

order to have a good result of letter recognition, one should design a transformation $\boldsymbol{T}_4$ to make $d(\boldsymbol{V}_{4t}, \boldsymbol{V}_{4r})$ and differences between their probability distributions as small as possible.

In general, in the decision stage, there are four possible results: a letter is precisely recognized as the assigned letter, a letter is recognized as another letter (false rejection) or cannot be recognized, and a different letter is recognized as the assigned letter (false alarm). Except for the first case, the rest of the three cases are certainly wrong decisions. Under the aforementioned framework of robustness in different domains, this study will try to impose a series of transformations, $\boldsymbol{T}_i$ so that the resulting $d(\boldsymbol{V}_{it}, \boldsymbol{V}_{ir})$ is properly reduced. To fulfill this goal, each of the above mismatched sources are examined individually.

2.4.2 Illumination and Background Challenges

For the indoor environment, the illumination conditions did not change drastically. Generally, the background can be cluttered for some application scenarios such as "in an office". It is believed this part will mainly affect the hand detection but should only slightly impact gesture recognition as long as we can guarantee a robust hand/object detector.

Under these circumstances, previous researchers used solutions such as different color space representations [22], sophisticated skin regions versus non-skin regions classification or thresholding rules [40, 9], Besides, instead of enhancing the feature, prior information or other side information has been proved to be useful in many applications. For instance, infrared or depth cameras [41, 42] may be applied to get three-dimensional information. In addition, simultaneously detecting several body parts, such as face and limbs, has been proved to be beneficial [43, 44] with prior geometrical constraints. Other information, such as motion and edge, has also been proved to be useful for better detection results [9]. Other types of solutions involve hardware solution including automatic white balance calibration [18] and the usage of active light sources. For example, in an in-car gesture recognition designed in [40], LED light is used to minimize the effect of different illumination conditions.

For in-car environments for the proposed research problem, these challenges are even

more severe. Some of the above mentioned techniques which require strong prior information and assumption did not work in the preliminary tests due to strong environmental noises and variations. Therefore, a reliable approach without such strong requirements should be studied carefully.

2.4.3 Gesture Variation

In this work, the main research target is English letter-type gestures. Unlike previous approaches, there are move-in and move-out components which contain no discrimination information at all. Moreover, different writing orders should be allowed. A velocity histogram based method [45] may be adopted, but it made a strong assumption on the writing style. The proposed method for this issue, however, is to explicitly define a set of basic units called strokes. The test experiments show that it worked reasonably well for gestures performed with different stroke orders. Instead, it should be reasonable to explicitly regulate how different strokes could be used to perform the desired letters. This issue will be discussed further in Chapter 4.

2.4.4 Low Frame Rate

Gesture sequences are usually sparsely acquired at a low frame rate of 15 to 30 frames per second (fps). It is well known that the speech signal is usually sampled at 16KHz and processed at 100 fps, with each frame composed of a 20 to 30 ms data weighted by a Hamming window, to obtain a sufficient data resolution. As a consequence, the relatively low frame rate acquisition process would result in too few available samples to represent each gesture, especially when a user performs a gesture sequence very quickly.

Under this low frame rate condition, traditional methods may not be applied properly. For example, the HMM-based systems can only afford a small number of states. This issue was seldom addressed in previous research efforts. Fortunately, a good example on the object tracking algorithm was found. Since the moving object will be often unpredictable, extra observers at different scales and re-weighting algorithms are required [46] so that

HMM-like recognizers [47] can be used properly. This work discusses an alternative such as interpolation techniques.

Since all the practical devices cannot use an infinite number of data points to describe a letter, it is reasonable to use a series of finitely many points, $f(x_0, y_0), f(x_1, y_1), \ldots, f(x_n, y_n)$, to approximate the overall $f(x, y)$. Now assume that there exists an number N such that, for all $n \geq N$, the probability of making a wrong decision between the whole $f(x, y)$ and these data points is negligible. Conversely, if $n < N$, the error probability is certainly positive. As to the magnitude of N, it depends on the above mentioned independent variable together with the stroke numbers to define the letter. Note that every stroke needs at least 2 points to specify a straight stroke and roughly 5 points for a curvy stroke. Consider the letter B, which involves a straight stroke and 2 curved strokes, as an example. One needs at least $M = 2 \times 2 + 2 \times 5$ points to specify all the strokes. The practical tests show that, for the sake of precisely defining a well written letter, the total number of points for specifying all the strokes is generally larger than M. These extra data points are used to cover many variations in the hand-written procedure, image data acquisition, image to actual written trajectory transformation, and letter modeling and recognition. Therefore, to clearly define a letter from every stroke point to every other point, all the position vectors are theoretically required to take these variations into account. The detailed discussion will be presented in the last part of Chapter 4.

Moreover, due to these variations, to develop an automatic recognition system, one can also form reasonable statistics from general training data sets. Obviously, the researcher hopes that the probability distributions describing these variations of the training data match those of the test sets. This issue will be further discussed in Chapter 5.

2.5 Summary

In this chapter, the baseline gesture recognition system and related feature extraction, modeling techniques, and potential robustness issues are briefly reviewed. For practical application under environments such as in-car, several challenges such as illumination variations, frame rates, and user variability certainly make the problem even more complicated. These challenges may result in many robustness issues that can deteriorate system performance. In fact, all of the test runs of this study showed disappointed overall performance of the system built solely basing on these existing techniques. Therefore, to alleviate these practical robustness issues will be the key to improving the system robustness. The following chapters will discuss some robustness techniques that are mainly targeted at these issues to enhance system reliability and performance. These techniques were mainly designed to achieve good robustness to illumination variation, low acquisition frame rate, model and feature representation.

CHAPTER 3

HAND DETECTION

In this chapter, the role of hand detection for contemporary gesture recognition as well as the in-car gesture recognition systems is reviewed. First, recognition systems based on regular block detection and spatial-chromatic clustering strategies are discussed, respectively. Second, two mainstream aspects of approaches based on color model and foreground-background segmentation are used. Next, a model adaptation scheme based on skin color distribution statistics and several useful foreground-background statistics are proposed to improve the detection rate. And finally, the identification of different scenes based on several sequence-level statistics is studied.

3.1 Background and Related Work

It is well known that detecting interesting body parts from single images or frames in video clips provides highly salient information for many human-centric applications, such as sign language recognition [43], biometric authentication [48], and other human computer interaction (HCI) systems [49]. Once the desired body parts are extracted, further post processing can be applied to these applications. It is obvious that the precision of these detections is of crucial importance.

Due to illumination changes, reflections, shadows, and occlusions, it is often challenging to detect human body parts in a robust manner under both normal and adverse operational conditions. Although most existing generic gesture recognition systems are built for in-door environments [18, 22, 9, 23, 21], several solutions have been proposed to tackle issues related to cluttered backgrounds or to address detection issues. First of all, enhanced features or detection rules may be applied to enhance system performance. As mentioned in Chapter 2, previous researchers used solutions such as different color space representations [22], sophisticated skin regions versus non-skin regions classification or thresholding rules

[40, 9]. Next, instead of enhancing the feature, prior knowledge or other side information is proved to be useful in many applications. For instance, infrared or depth cameras [41, 42] can be utilized to get three-dimensional information. In addition, simultaneously detecting several body parts, such as face and limbs, has been proved to be beneficial [43, 44] with geometrical or structural constraints. Other side information, such as motion and edges, is also useful for better detection results [9]. Other solutions involve additional components, such as hardware included automatic white balance calibration [18] and active light sources. For example, in an in-car gesture recognition system designed in [40], LED light was used to minimize the effect of different illumination conditions.

In-car recording scenarios have similar but much more severe issues when compared to most indoor systems mentioned above. For in-car scenarios, the hardware setup such as aperture and shutter speed may be improperly set by a participant who was not familiar with the camera. The collected images are certainly not adapted to the irradiance of the light source well because the hand movement and induced shadow can drastically and dynamically affect the illumination conditions. Consequently, the image brightness and contrast may be insufficient and produce low contrast frames that are too dim or too bright. Moreover, the automatic gain control for sensors may actually result in lots of noisy salt-and-pepper artifacts and constant illumination level changes. This work chooses to compensate these situations via post-processing.

As far as locating individual body parts, skin color detection and foreground-background segmentation are two mainstream approaches. These two aspects need to be combined to get more reliable detection results. Also, as suggested above, it is useful to include more side information to help the detection.

3.2 Baseline Hand Detector with Skin Color Modeling

This work starts with a baseline system introduced by the above mentioned algorithms. The

baseline hand detector was built solely using color information. For clean recording conditions, only doing naïve background subtraction will be able to identify the skin regions easily, but this is far from the truth for skin detection from in-car recordings. Therefore, it is required to build models for skin and non-skin pixels or blocks. To model skin versus non-skin pixels or blocks, several color space representations such as RGB, normalized RGB, HSV, YCbCr, and CIE-Lab, etc. are all possible candidates to be applied to each pixel, rectangular blocks, or irregular blobs. The effect of different color space representations and skin color models are discussed in [50] for hand detection, and it turns out that different color spaces are recommended in different application domains. This is because, in practical situations, skin regions can have dramatically different color space distributions in different backgrounds and under various illumination conditions [26].

In addition to the traditional modeling ways, color tracking using the expectation-maximization (EM) algorithm is shown to be useful [51] while adaptive techniques based on refining the model with false detection samples have also been proposed in [52] to enhance skin detection performance. Therefore, instead of finding a robust feature set or color space representation, this work studies the skin color detection problem from an adaptive model-based perspective. In the following paragraphs, both the regular block and irregular block based detection strategies are discussed.

3.2.1 Regular Block Detection

To use more local information, this study analyzes image frame with a block of pixels and takes the average of the RGB pixel values to represent the block in a three-dimensional RGB feature space. This is because other color representations cannot give statistically meaningful improvement in the testing results. Average RGB values within a block of size 4×4, 16×16, and 32×32 are chosen as a feature vector. This low-level RGB feature vector is used to design our baseline detectors by modeling skin and non-skin classes as Gaussian mixture models (GMMs) [27, 28]. This modeling strategy enables us to do hypothesis testing for the skin and non-skin blocks as shown below.

Assume that likelihood score vectors from K modalities for class C and $\widetilde{C}$ are denoted by $\boldsymbol{f}(x|\boldsymbol{\theta}_C) = [f_1(x|\theta_C^1), \ldots, f_K(x|\theta_C^K)]^T$ and $\boldsymbol{f}(x|\boldsymbol{\theta}_{\widetilde{C}}) = [f_1(x|\theta_{\widetilde{C}}^1), \ldots, f_K(x|\theta_{\widetilde{C}}^K)]^T$, respectively. When $K = 1$, the log likelihood ratio is

$$LLR(x|\theta_C, \theta_{\widetilde{C}}) = \log f(x|\theta_C) - \log f(x|\theta_{\widetilde{C}}), \tag{68}$$

which is produced by the skin and non-skin models. Note that it is also possible to use statistic scores as features to design skin color detectors, because the hypothesis testing by itself is equivalent to a linear classifier on a transformed one-dimensional space. This study will further use it as a high-level, discriminative binary-class feature.

Moreover, the skin color model can be applied at several different block scales. Then, with a simple heuristic, the following two constraints are used to combine the detection results from these different scales.

1 Detection consistency constraint: Small block and large block results should be consistent. Small blocks that do not agree with the large blocks will be thought of as outliers and discarded.
2 Group size constraint: During block agglomeration, connected regions that are smaller than a pre-defined threshold will be removed.

It is interesting to note that this model effectively suppresses the previously mentioned oscillation problem caused by car vibration.

3.2.2 Irregular Block with Spatial-Chromatic Clustering

As discussed in previous sections, all detections can be done on uniformly sampled blocks. The main issue of this strategy is that it might give granular detection result as well as false alarm because the within frame local similarity is not considered. That is, an alternative scheme that is able to consider the local similarities of each image frame quickly and agglomerate them into meaningful sub-blocks properly matching with original image edges may be beneficial.

To fulfill this purpose, a jointly spatial and chromatic clustering is required. This work applied the simple linear iterative clustering (SLIC) [53] algorithm, with default 200 superpixels and 20 regularization parameters, on the studied dataset. It is interesting to find the two most important factors for robust local region clustering are the choice of color space and the neighborhood selection. The SLIC is basically a regular clustering algorithm on an image with distance measure used for the clustering being:

$$D = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2}, \tag{69}$$

where d_c, d_s, m, S are color space Euclidean distance, spatial space Euclidean distance, compactness regularization parameter, and grid size, respectively. From the follow-up work of [53], it is found choosing the parameter m that manually can be quite tricky and may produce irregular superpixels and it is better to use adaptive parameters for different configurations of neighborhoods. Since the parameter m also means the maximal color distance within each cluster, it can be chosen adaptively by directly computing from each cluster. Thus the parameter can be automatically chosen, which is called SLIC Zero or SLIC0. As seen in the Fig. 5, the SLIC0 produces many less irregular regions.

From test experiments, it is found the SLIC is indeed sensitive to the illumination condition even with the color space transformation. This is where compensation technique such as the histogram equalization (HEQ) can be helpful, as shown in Fig. 4. It is very important to match the edge well since it will give more accurately detected hand regions.

Meanwhile, it is quite important to choose a good color space as well. The most intuitive way is to do the RGB space SLIC. However, the image after HEQ can have some granular noises and the resulting superpixels are quite scattered and noisy. On the contrary, the use of the Lab color space will give more compact and reasonable regions, as shown in Fig. 6

Here, the Lab color space is based on CIE D65 standard [54] to mimic human perception, which can be done as follows. First, scale RGB by $\frac{1}{255}$, i.e. $\{r, g, b\} = \frac{\{R,G,B\}}{255}$, then

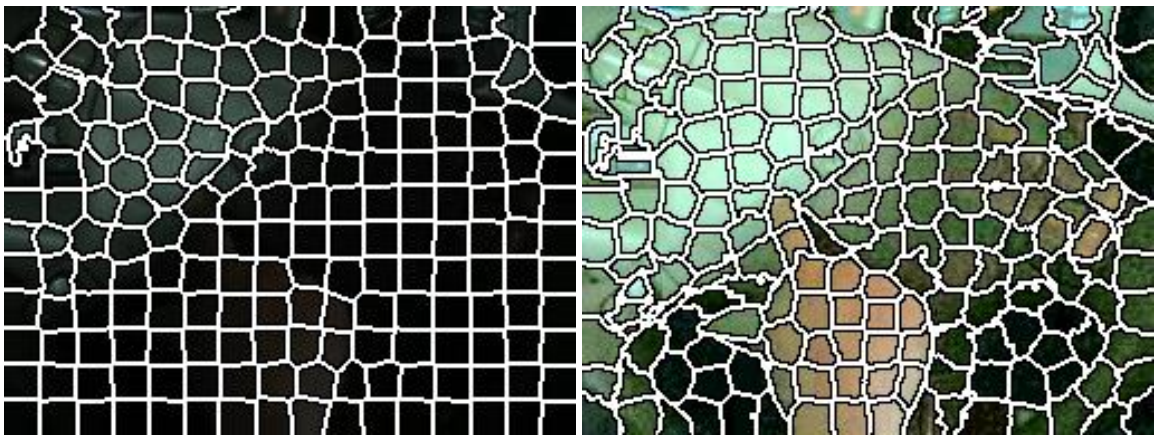


Figure 4: SLIC results before (left) and after (right) HEQ, it is evident that after HEQ, the image edges can be tracked well while they are mostly missed in original image.

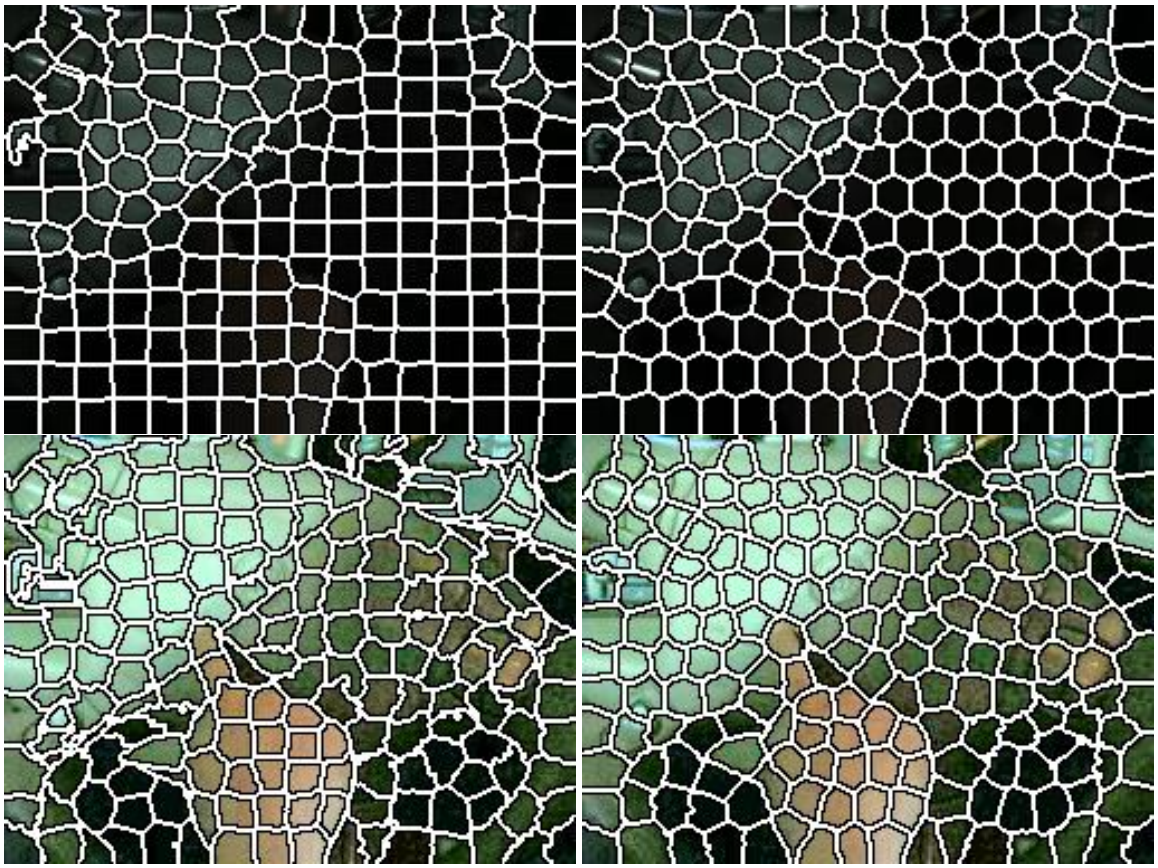


Figure 5: SLIC (upper) and SLIC0 (bottom) results before (left) and after (right) HEQ, The SLIC0 can give more homogeneous superpixels before and after HEQ and can match the edge well.

compute intermediate variables:

$$\{R,G,B\}_{linear} = \begin{cases} \frac{\{r,g,b\}}{12.92} & ,\{r,g,b\} \leq 0.04045 \\ \left(\frac{\{r,g,b\}+0.055}{1+0.055}\right)^{2.4} & ,\{r,g,b\} > 0.04045, \end{cases} \tag{70}$$

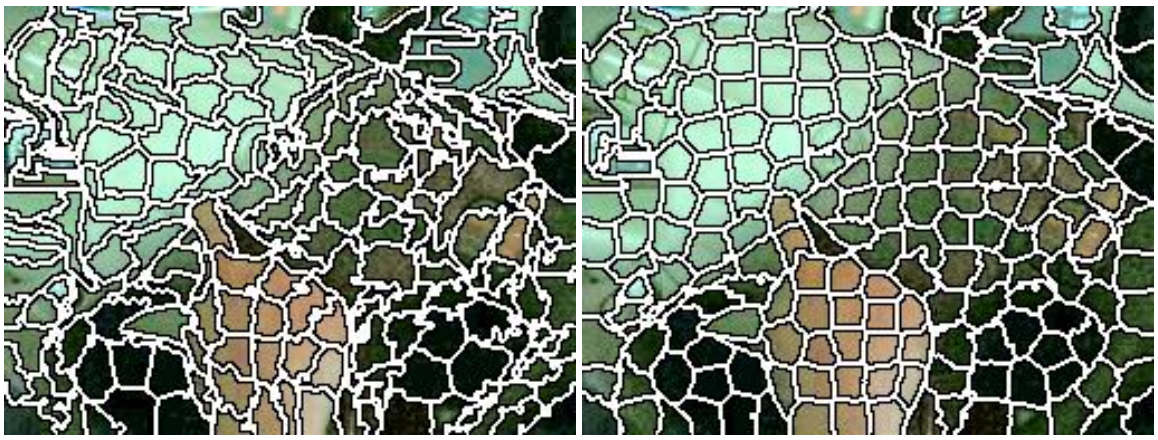


Figure 6: RGB (left) and Lab (right) SLIC, the RGB result are quite noisy and many small clusters are observed, while the Lab gave more compact and reasonable regions.

Next, convert them to CIE XYZ color space:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126719 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} R_{linear} \\ G_{linear} \\ B_{linear} \end{bmatrix}, \tag{71}$$

where when $[R_{linear}\, G_{linear}\, B_{linear}]^T = [1\ 1\ 1]^T$, one has the reference white point $[X_n\, Y_n\, Z_n]^T = [0.950456\ 1\ 1.088754]^T$, and finally convert XYZ to Lab space:

$$L = 116\left(\frac{Y}{Y_n}\right) - 16 \tag{72}$$

$$a = 500\left[\frac{X}{X_n} - \frac{Y}{Y_n}\right] \tag{73}$$

$$b = 200\left[\frac{X}{X_n} - \frac{Z}{Z_n}\right], \tag{74}$$

where

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & t > \left(\frac{6}{29}\right)^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29}, & otherwise. \end{cases} \tag{75}$$

As shown in above equations, this is a highly nonlinear empirical transformation. Although using it for GMM-based detector did not work better than RGB, it is quite useful here for clustering purposes.

In addition, the neighborhood search region was found to be quite important. When the region size is too small, the resulting blocks did not match the image edge well, as in Fig. 7. After several trials and errors, it was confirmed that twice the grid size gave the overall best qualitative results.

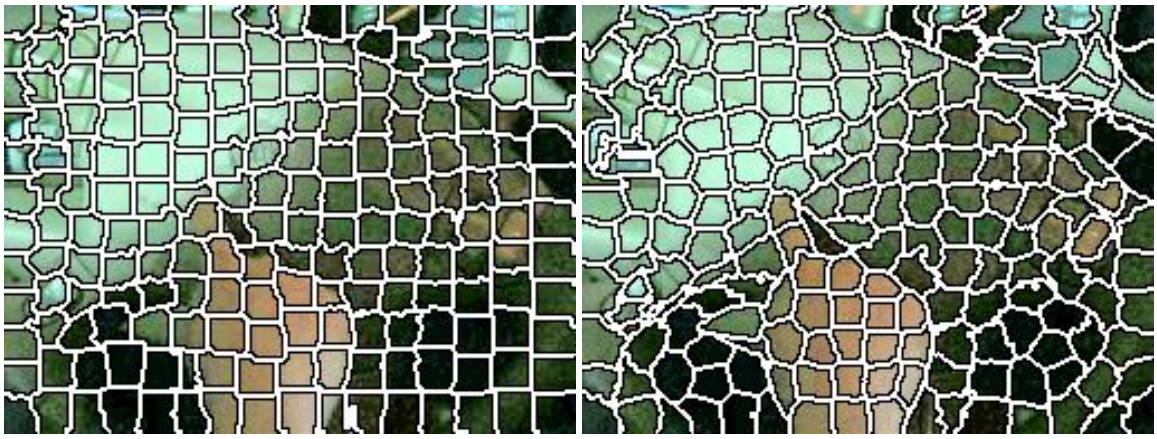


Figure 7: When search region is reduced by 40% (left), the resulting blocks do not match edges well.

Finally, it was found the compactness parameter can be quite tricky in complex images as what encountered here.

Instead of doing jointly clustering over spatial and chromatic domain, one can include the higher order clustering by including the temporal variation besides the previously mentioned two domains. The previous algorithm will be applicable as well, but the search space should include the temporal domain. The rest of the work is still the same. However, for low frame rate scenarios, directly applying the clustering in 3D space will not give a reasonable result because the fast movement and illumination changes will make the clustering inefficient. Moreover, the time complexity for the current implementation seriously blocks the efficient use of this 3D clustering, so this study will not apply this extended algorithm in the following application.

3.3 Background-Foreground Information

When images are given by video clips instead of with individual pictures, one can get some information by comparing the background (non-skin) and foreground (skin) regions. Motivated by voice activity detection (VAD) [55] in the speech community, this work designed an energy-based algorithm to identify background or reference frames so that it can construct background and foreground relationship.

Assume the first frame of each clip is always a background frame, and the oscillation is NOT severe enough to give strong global background movement. Then all other frames in the same clip are examined using a given threshold η, $0 < \eta < 1$. For the i^{th} frame, if its sample standard deviation σ_i satisfies the following inequality, it is considered as a background frame,

$$\frac{\sigma_i - \sigma_{min}}{\sigma_{max} - \sigma_{min}} < \eta \tag{76}$$

Contiguous background frames are then averaged to form an averaged background for future comparisons. Having identified background frames, blocks that potentially belong to foreground objects are compared with their corresponding background blocks to check if they are in the same class, i.e. skin or non-skin block.

A straightforward way to test if two blocks in similar images belong to the same class is to examine their sums of squared errors, as has been done in the Kanade-Lucas-Tomasi (KLT) optical flow algorithm [56]. However, this method can be problematic if the block pixels are under affine transformation, which is similar to the illumination model proposed in [57]. Therefore, instead of using deterministic matching tricks, this work proposed to use other statistical features to measure the background-foreground similarities.

The similarity between two sets of data samples is a general question. There have been many statistical methods designed for this purpose. This study used several most commonly seen methods to deal with this issue as stated below.

3.3.1 Pearson's Correlation Coefficient

One potential useful statistical quantity is the Pearson's correlation coefficient. The linear correlation coefficient ρ is relatively robust to color space affine transformation, which is defined as:

$$\rho = \frac{Cov(f, g)}{\sqrt{Var(f)Var(g)}} \tag{77}$$

Under the jointly Gaussian and uncorrelated assumptions of the pixel vectors in a local image foreground block f and its corresponding background block g, the test statistic $t = \sqrt{\rho^2(M-2)/1-\rho^2}$ has the t-distribution [58] , where M is the number of samples in each block being analyzed, say, 256 samples in a 16×16 block.

Since $|t|$ is often used in t-test and is also a strictly increasing function of $|\rho|$ in the range of $[0, 1]$, which is in a normalized form. So $|\rho|$ is chosen as a feature to measure the background-foreground similarity at each image block. It is believed that this is an effective high-level discriminative binary-class feature to be used for skin color detection.

3.3.2 Kolmogorov-Smirnov Statistic

The most commonly used measurement for measuring distribution similarity is the Kullback-Leiber divergence and related amounts. However, to use it for general measurement without the exact distribution form, the Monte Carlo sampling [59] is needed to approximate the integration computation. Therefore, this work adopts the the Kolmogorov-Smirnov (KS) statistics [60] for non-parametric similarity computation to measure the similarity of two objects.

It is quite straightforward to compute the KS statistics between two sets of data, or so called two-sample KS statistics. First, the empirical cumulative distribution function (CDF) for the two sets are computed, then the two-sample KS statistics are computed as follows.

$$KS_{F_m,G_n} = \sup_x | F_m(x) - G_n(x) |, \tag{78}$$

where F_m and G_n are CDF of two sets with cardinalities m and n, respectively. An example is shown in Fig. 8 in which the upper curve is F_m, the lower curve represents G_n, and the

two sided arrow shows KS_{F_m,G_n}.

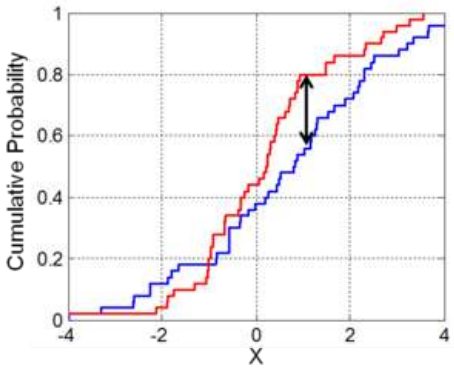


Figure 8: The two sided KS statistics are shown as double sided arrow between two empirical CDF.

To test the hypothesis that F and G followed the same distribution, one can use a scaled version of the statistics [61].

$$D_{F_m,G_n} = \sqrt{\frac{mn}{m+n}} \sup_x | F(x) - G(x) |, \tag{79}$$

which can then be shown to follow the KS distribution [60], whose CDF is in the form of:

$$Pr(K \leq x) = 1 - 2 \sum_{k=1}^{\infty} (-1)^{k-1} \exp\left(-2k^2x^2\right) \tag{80}$$

With the above distribution at hand, one can use the KS statistics for the hypothesis testing. Practically, a lot of samples are needed to perform the KS test , but in the color value histogram comparisons, it is found that the KS test is fairly strict. For example, with images of size $256 \times 192 = 49,152$, the significant level [60] at $\alpha = 0.05$ is around 0.0087, which is a quite strict hypothesis testing. Therefore, in the proposed application, instead of doing the significant test, KS statistics are served as a feature to measure how similar the color value histograms are for different blocks. Note that the KS statistics do not consider the case when pixel intensities are under affine transformation, this can be their weakness and may lead to bad detection results.

3.4 Adaptive Skin Modeling

The main advantage of utilizing the adaptive model-based method is the possibility of feeding back the evidences from the observed data into previously trained models. These adaptation techniques are most commonly used in speech community [62, 63]. For instance, online adaptation with observation contributions adjusted by exterior cues may enable the adaptation of the HMM for gestures [64]. These type of concept may be extended to hand detection, modeling, and even feature learning.

The adjusted parameters of the trained models cannot only achieve a good fit for the observed data but also maintain good generalization capability. One commonly used method is a Bayesian model estimation which is known as the maximum a posteriori (MAP) adaptation [65].

When training and testing conditions are different, parameters obtained from the training phase may not be able to properly describe the actual distribution of the testing data. Bayesian model adaptation is often adopted under this circumstance in the speech community. MAP adaptation assumes the form of the prior density to be a conjugate prior of the probability density function of the feature vectors. Therefore, the posterior density will not only include the observed data information but also have the same form as the prior density [65]. The information of the observed data and the prior density are effectively combined under this framework. Assume the likelihood function for class C is denoted by a K-mixture GMM: $f(x|\theta_c) = \Sigma_{i=1}^{K} \omega_i \mathcal{N}(x|\mu_i, \Sigma_i)$, as shown in [65], given a set of adaptation data, $x_n, n = 1, ...N, x_n \in \mathcal{R}^D$, parameter adaptation can be done as follows:

$$\omega_i^{MAP} = \frac{\nu_i - 1 + \sum_{n=1}^{N} \gamma_{in}}{N - K + \sum_{j=1}^{K} \nu_j}, \tag{81}$$

$$\mu_i^{MAP} = \frac{\tau_i m_i + \sum_{n=1}^{N} \gamma_{in} x_n}{\tau_i + \sum_{n=1}^{N} \gamma_{in}}, \tag{82}$$

$$\begin{aligned} \Sigma_i^{MAP} = & \frac{u_i + \tau_i (m_i - \mu_i^{MAP})(m_i - \mu_i^{MAP})^T}{\alpha_i - D + \sum_{n=1}^{N} \gamma_{in}} + \\ & \frac{\sum_{n=1}^{N} \gamma_{in} (x_n - \mu_i^{MAP})(x_n - \mu_i^{MAP})^T}{\alpha_i - D + \sum_{n=1}^{N} \gamma_{in}}, \end{aligned} \tag{83}$$

where

$$\gamma_{i,n} = \frac{\omega_i \mathcal{N}(x_n|\mu_i, \Sigma_i)}{\sum_{j=1}^{K} \omega_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{84}$$

and v_i is the i^{th} hyperparameter of the prior density of mixture weights and τ_i, m_i, u_i, α_i are hyperparameters of the prior density for i^{th} Gaussian mixture:

$$p(\omega_1, ...\omega_K) \propto \prod_{i=1}^{K} \omega_i^{v_i - 1}, \tag{85}$$

$$\begin{aligned} p(\mu_i, \Sigma_i^{-1}) \propto & |\Sigma_i^{-1}|^{\frac{\alpha_i - D}{2}} exp[-\frac{1}{2} tr(u_i \Sigma_i^{-1})] \times \\ & exp[-\frac{\tau_i}{2}(\mu_i - m_i)^T \Sigma_i^{-1}(\mu_i - m_i)]. \end{aligned} \tag{86}$$

This study will not adapt the covariance matrix to save computational cost. Also, the adaptation is performed in an unsupervised manner such that the current model set is first used to decode the test samples to a specific class. Such labels are then treated as supervision information for that class and are then used to perform the above MAP adaptation algorithm. Practically, MAP adaptation can be viewed as the weighted sum of the parameter prior information and the sampled statistics.

3.5 Information Fusion

Successful experiences reported in [27] showed that scores generated by several modalities can serve as features for training a fused-feature model and give beneficial impact to the detection performance. This late fusion technique is usually done in two different ways. One is to find a simple transform to combine the outputs of several classifiers online, the other is to treat scores generated by competing classifiers as features to train higher level classifiers. This study investigates the latter one by fusing the regularized log likelihood ratio (LLR) feature and the foreground-background statistic into a single feature vector.

To make sure the model will not favor a specific feature, the log likelihood ratio $LLR(x|\theta_C, \theta_{\overline{C}})$ is transformed into a reasonable dynamic range. Here this work chooses a logistic sigmoid

function defined as follows:

$$\sigma(LLR(x|\theta_C, \theta_{\bar{C}})) = \frac{1}{1 + exp[-\xi(LLR(x|\theta_C, \theta_{\bar{C}}) - \delta)]} \tag{87}$$

with empirically chosen parameters $(\xi, \delta) = (0.5, 0)$ to transform LLR scores into the range of $[0, 1]$. Also, as stated previously, this study chooses $|\rho|$ instead of ρ to make it also in the range of $[0, 1]$. The KS statistic is always within this range so one can directly use it as well. Thus the early fusion feature for a 16×16 block used to train the fusion GMMs is composed of $\{\sigma(LLR(x|\theta_C, \theta_{\bar{C}}))\ |\rho|\ KS_{C,\bar{C}}\}$. This feature set will be shown to have enhanced discriminative powers over the original color features.

The detection performance can be further enhanced if one replaces the LLR features by MAP-adapted LLR features. The preliminary experimental results indicate that the proposed joint framework can effectively alleviate the model parameter mismatch problem and achieve an F_1 score as high as almost 90% across a wide range of lighting conditions. Skin and non-skin models can thus be trained on the above-mentioned feature vector to obtain an improved performance over the low-level RGB feature vector.

3.6 Robust Hand Detection

With the above mentioned algorithms, it is found that there are some cases that are subject to more misclassification than other cases. In this section, therefore, two sets of methods such as data categorization and feature compensation are studied.

3.6.1 Data Categorization

The previous methods assume the skin and background patches can be well described solely by two sets of GMMs for both foreground and background. This is, however, in general, not quite true. Patches of different tones and illuminations, in actual scenarios, are not very easy to be modeled only by color features. For example, recent progress on doing ego-centric hand detection [24] showed the benefit of including prior models in a flavor similar to topic model framework. Under these circumstances, one can modified the LLR score to

be under scenario S:

$$LLR(x|S, \theta_{C,S}, \theta_{\widetilde{C},S}) = \log f(x|\theta_C, S) - \log f(x|\theta_{\widetilde{C}}, S) \tag{88}$$

In a practical situation, the scene model is not given, so this work partitions the color space into several candidate subsets and then makes a fusion afterward. Several potential strategies which may involve other modalities are discussed below to facilitate the recording scene categorization.

3.6.1.1 Multi-Condition

The multi-condition is a strategy often used for automatic speech recognition. Basically, it is to pool together all data available in the training set and those used for training and validation purposes. The underlying assumption is, one has enough data for most of the conditions so that the training procedure will not favor a specific data condition and thus result in the over-training. This condition, however, may not be always true for unbalanced data sets.

3.6.1.2 Sequence Histogram Based Categorization

Although good results for skin block detection have been reported with adaptation schemes [66], as will be discussed in the experimental result, bad center of mass trajectories can still happen if the "false-alarm blocks" are located in a scattered way. After checking the existing literature [67, 68], it is shown that complicated features such as scale-invariant feature transformation (SIFT) based features or oriental histogram based features such as histogram of oriental gradient (HoG) will give better averaged precision (AP) in most cases. But these methods are mostly for the off-line analysis, which could be too time consuming to be used for this application.

After reviewing the present data set, it is found that most bad detections occur while the image frames are too dark/low in the contrast of brightness or suffered from the shadow. Illumination histogram based features are designed to take care of this issue. Assume each sequence is relatively stationary, if the illumination distribution of all frames of each

sequence are given, defining the illumination variable as X, the histogram can then be expressed by $P((n+1)b > X \geq nb)$ for the n-th bin with bin size b. How to use this information will be discussed now.

3.6.1.3 K-Means Clustering

K-means clustering via the above mentioned illumination histogram will be directly run. This is usually applied in many works such as those presented in [69].

3.6.1.4 Percentile Features

To represent the illumination, the inverse of the cumulative distribution of x will be used. At first, this study defines Y as an inverse variable of X, that is, $F(x) = y, y \in [0, 1]$. Then, the median $F^{-1}(0.5)$ and 95% percentile $F^{-1}(0.95)$ are picked as the feature for preliminary studies. As shown in Fig. 9, this inverse cumulative distribution feature (ICDF) can clearly identify different image conditions in our training data. This work chose the samples in dotted line rectangle as category one, and the rest as category two to separately extract trajectories. Note the direct clustering may not be able to identify the samples in such a clear way.

3.6.1.5 Kolmogorov-Smirnov Statistic

As mentioned in previous paragraphs, the KS statistic is usually used to measure the data when no underlying distribution is given. This work also uses them for the categorization purposes.

3.6.2 Image Compensation

Instead of working on the data categorization, techniques that can compensate for the chromatic distribution mismatches are also worth studying. Two strategies are discussed below.

3.6.2.1 Histogram Equalization

This is served as an alternative method for detection domain robustness. As shown in Fig. 10. The detection result did not show quite dramatic difference. But this slight difference can affect the overall system quite crucially. For example, less false alarm frames are

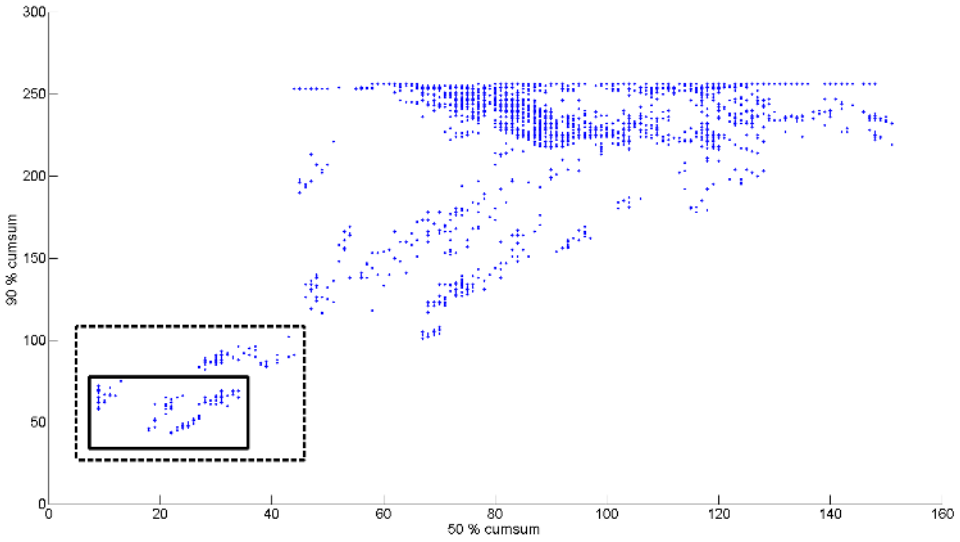


Figure 9: Inverse CDF features for illumination on training data. Note that samples in the the solid rectangle are clearly different from others, and manually checking samples showed that the samples in dotted line rectangle are very similar in their illumination conditions, while the rest in solid rectangle are not as dark and should be considered as transition between two illumination conditions.

observed in the starting and ending frames. These observations verify the requirement to make a better robustness for better detected trajectories than that in the current stage.

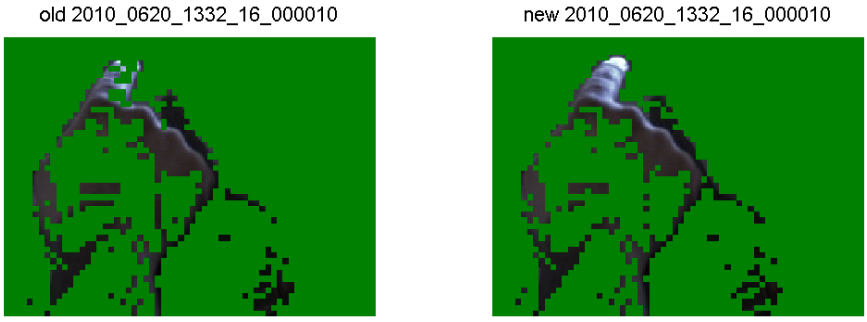


Figure 10: Detection example: before and after illumination categorization. An only slightly better result is observed.

The original histogram equalization is a technique for image contrast enhancement for gray-scale images. The key idea is to make the transformed probability density function of its pixel intensity as uniform as possible. Assume image pixels are a random variable X taking the value from $[0, L-1]$, the resulting transformed pixel value Y takes the value

from transform $T(.)$, which is assumed to be invertible:

$$Y = T(X) \tag{89}$$

Then from probability theory and fundamental theorem of calculus:

$$\int_0^y f_Y(\psi)d\psi = \int_0^{T^{-1}(y)} f_X(\chi)d\chi, \tag{90}$$

$$\frac{d}{dy}\left(\int_0^y f_Y(\psi)d\psi\right) = f_Y(y) = f_X(x)\frac{dx}{dy}, \tag{91}$$

when transform $T(X)$ is in the form of scaled cumulative distribution function

$$T(X) = (L-1)\int_0^X f_X(\chi)d\chi, \tag{92}$$

one has

$$f_Y(y) = f_X(x) \times \frac{1}{(L-1)f_X(x)} = \frac{1}{L-1}, \tag{93}$$

which is a uniform distribution taking the value in $[0, L-1]$.

3.6.2.2 Histogram Matching

A more general problem is to match a histogram to another one rather than to a uniform distribution. This can be done in a straight-forward way by using the above fact. Assume Z has the desired pixel histogram, one has a transform $W = G(Z)$ to transform it into a uniform distribution in $[0, L-1]$. Since the distribution of W and Y are both uniform in $[0, L-1]$, it is obvious that

$$Z = G^{-1}(T(X)) \tag{94}$$

An adaptive equalization technique [70] also exists for equalizing local regions instead of global regions in which multiple matching algorithms for multiple blocks are also explored for template matching [71]. In this work, the main focus is on the baseline algorithm of Eq. (92) and its impact to the whole system. Moreover, matching the nominated distribution provided by sampling the training data is also studied.

3.7 Experiments

This section will show some experimental results based on the methods proposed in this chapter.

3.7.1 Enhanced Color Detection for Hand Detection

The examined dataset was recorded using Dragonfly 2 color camera with the setup of 15 fps 1024×768 YUV422 and 10 ms shutter speed to remove blurring artifacts caused by possibly fast hand motion. All frames were resized to 256×192 to save storage space. 16 users were recorded at 3 time slots in a day: morning, noon, and evening, resulting in roughly 4,000 clips with about 16,000 frames.

In order to study the performance under different recording conditions, this work first used frame-average RGB values to cluster the whole dataset into three categories. After a brief data analysis, it was found Categories I, II, and III were mostly recorded in the morning, noon, and evening, respectively. And the ratio of frame numbers in Categories I, II, and III is roughly 4:2:1. The present work randomly selected 1/3 of the data from each category for manual labeling with 16×16 blocks and randomly divided the labeled data evenly for training and testing. GMMs with mixture numbers determined by [72] were used for modeling. Apparently the category with most number of samples, i.e., Category I, would dominate the training. Based on their different illumination conditions, Categories I, II, and III were respectively marked as well-matched, slightly-mismatched, and highly-mismatched conditions.

Besides, due to unbalanced number of samples between skin and non-skin blocks, which has the ratio of 1:4, accuracy ends up being a bad performance index. To fairly evaluate the performance, indices measuring correctly classified skin blocks should be used. To do so, let (TP, TN, FP, FN) denote numbers of correctly classified skin blocks, correctly classified non-skin blocks, non-skin blocks classified as skin blocks, and skin blocks classified as non-skin blocks, respectively. The precision, recall, and F_1 are defined as $TP/(TP + FP)$, $TP/(TP + FN)$, and $2 \times \text{precision} \times \text{recall}/(\text{precision} + \text{recall})$ which is a

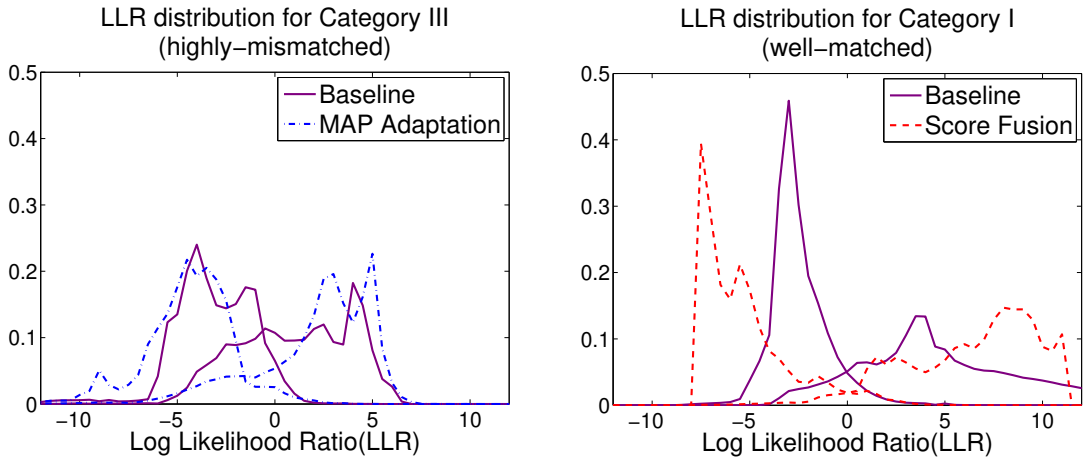
Table 1: The Performance Evaluation for threshold=0

Category I	Precision	Recall	F_1	F_1^{best}
Baseline	79.51	89.01	83.99	85.88
MAP adaptation	94.50	86.55	90.35	90.36
Score fusion	84.83	95.91	90.03	91.77
Proposed framework	90.98	92.85	91.91	92.56
Category II	Precision	Recall	F_1	F_1^{best}
Baseline	69.16	86.90	77.02	78.69
MAP adaptation	90.98	84.73	87.74	87.90
Score fusion	82.15	95.62	88.37	89.82
Proposed framework	91.79	91.85	91.82	91.98
Category III	Precision	Recall	F_1	F_1^{best}
Baseline	77.16	61.74	68.59	69.12
MAP adaptation	87.15	78.77	82.75	83.04
Score fusion	71.83	85.95	78.26	78.90
Proposed framework	86.69	89.25	87.95	89.03

harmonic mean of the precision and recall, respectively.

The class separation plots shown in Figure 11 were first checked to examine the advantages of applying MAP and fusion of features. Note that the misclassification rate can be visualized by the overlapping area under the curves of distributions of 2 classes. In a highly-mismatched scenario, applying the MAP adaptation, as shown in the dash-dot lines in Figure 11(a), the reduction of between-class overlap is clear. In a well-matched condition, applying feature fusion can still enhance the discriminative power, as shown in the dashed lines in Figure 11(b).

F_1 scores with LLR decision threshold 0 are listed in column 3 of Table 1. One can see that the MAP adaptation with 5 iterations outperformed the baseline score by 6.4 to 14.1% absolutely for Categories I to III. Moreover, applying fusion scheme on baseline score can beat the MAP adaptation only in Category II by 0.63% and degraded by 0.31% and 5% in Categories I and III, respectively. Although 0.63% and -0.31% are within the range of statistical error, the 5% degradation in Category III clearly shows the difficulty of the original fusion scheme under mismatched conditions. The proposed joint framework, however, consistently outperforms the MAP adaptation and the original fusion scheme by



(a) MAP adaptation (b) Fusion of features

Figure 11: Separation: (a) before (solid line) and after (dash-dot line) adaptation for Category III data; and (b) before (solid line) and after (dashed line) score feature fusion for Category I data. The misclassification rates (intersection areas) were halved for both scenarios.

1.5 to 5 % and 1.8 to 9.7 %, respectively. Next, the best F_1 scores for each algorithm were checked, as listed in column 4 of Table 1. The MAP adaptation is shown to be better than the baseline by 4.5 to 13.9%, the score fusion outperformed MAP by 1.4 to 2% except in Category III, and the proposed framework, again, outperformed MAP adaptation and the original fusion scheme by 2.2 to 6.0%, and 0.8 to 10.1%, respectively. Obviously, the proposed method effectively combined the advantages of both methods and obtained an F_1 as high as 89.0 to 92.6%. Especially, it can effectively compensate the mismatch between the dark environment (Category III) and the training condition.

A clear picture can be found in Figure 12. The curve trend indicates that in well-matched cases, i.e., Category I, score fusion can outperform MAP adaptation in general, and was not good in slightly-mismatched Category II scenarios, and became much worse in highly-mismatched Category III conditions. But the proposed method could outperform all algorithms when the precision exceeds 90%. In highly-mismatched conditions, it can compensate for the mismatch very well.

Adding KS statistics, however, will make the performance difference within 0.2%. In

fact, the correlation coefficient itself is already strong enough to give reasonable performance. This is possibly due to the issue of affine transformed pixel intensities.

3.7.2 Detection with Spatial-Chromatic Clustering

This study manually labelled the extracted SLIC clusters and did the same configuration as using regular blocks. The overall F_1 score is degraded by 5% absolutely. This is due to the fact that the irregular blocks can make the feature vectors, that involve the blob variance such as auto correlation, not robust. So this suggests that one can only use SLIC as a tool to help validate the detection result. Due to the extra time required on the computation, it will not be used in the following content.

3.7.3 Effect of Scene Categorization

By calculating the KS statistics from the histograms of the image sequence pixel values or desired probability distribution, three ways were compared:

1. Do standard K-means on pixel value histograms of training data.
2. Do K-means clustering on KS statistics. While merging clusters, clusters that can minimize the maximal (minimax) KS statistics between two clusters are chosen.
3. Do K-means clustering on KS statistics. After merging clusters, we used the cumulative distribution function provided by whole clusters for further iterations.

After trying several parameters, it was found 4 clusters can better describe the population, the clustering results are summarized in Fig. 13. It is obvious that the first two strategies gave much more balanced results than the third one. So this work is doing further experiments based on the first two strategies in order to identify the better strategies to categorize the training data.

After comparing with an AGR system built by enhanced hand detector without scene categorization, it turns out applying the regular K-means clustering gives the worst result while the percentile feature gave the best result. However, the impact of scene categorization on our gesture recognition task is not very strong; the percentile based categorization

is the best one (0.25% absolutely, 1.5% relative gesture recognition error reduction), and the direct clustering on the sequence color histogram is the worst, which has no gain over the original AGR system. This result is very likely due to the fact that one has not enough data for different categories. However, the positive performance impact indicates that the categorization could be useful for more general systems with more data collected under different recording scenarios.

3.7.4 Histogram Compensation

Next, the effect of doing histogram compensation techniques on all the datasets will be discussed. As shown in Figure 14, doing HEQ and re-training the skin-color model can enhance gesture recognition quality, but may also give some false alarms. This can be seen more clearly in Figure 15.

Table 2: The Performance for Different Hand Detection Strategies on Dataset I

Strategies	Naïve	Enhanced Hand Detection	+HEQ	+Histogram Matching
Letter Error Rate	24.62 %	14.86 %	14.65 %	15.77%

Now, based on these observations, the performance of doing image compensation is checked. As shown in Table 2, HEQ cannot directly help much although it is useful to reduce the recognition error. It is good to imply the high contrast one still performs reasonably well while the low contrast data which are much less than the high contrast data can take some advantage by using HEQ. The histogram matching, however, will bring more unnatural artifacts as well as more false alarms. Therefore, when compensation is required, it still works on the uniform equalization.

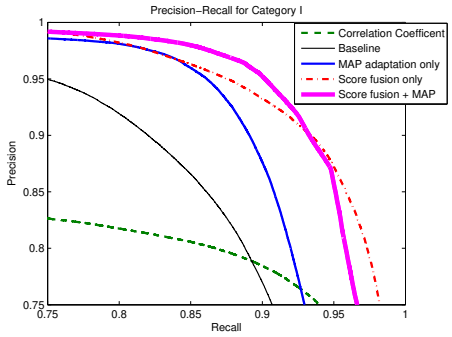
3.8 Summary

In this chapter, methods of doing hand detection for the gesture recognition system were studied. The applications of color skin and foreground-background information have been proved to be highly competitive. By fusing these information sources with properly maximizing a posteriori adaptation, one can further improve the detection reliability. With the

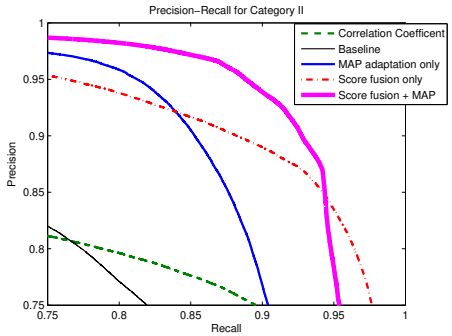
proposed scheme, the detection F_1 score can be enhanced and the final gesture recognition relative error rate can be reduced by around 45 to 49% compared with the gesture recognizer with the original hand detector. Specifically, the error rate is closed to the commonly satisfactory level of 10%.

This study also tried to use the SLIC superpixel for detection and found the final recognition performance did not gain much by using the irregular units. Data categorization and histogram equalization have been proved to be useful for mismatched cases but is not quite obvious, due to the currently datasets which were acquired in a biased manner. But its positive impact indicated that when the data distribution are biased, these kinds of transformations can be helpful to maintain the system robustness.

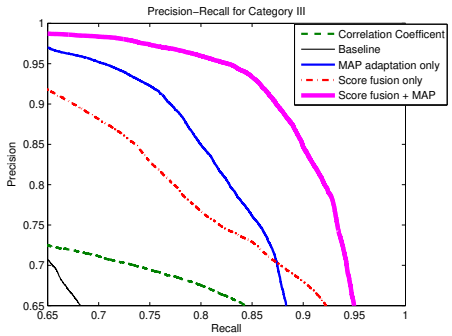
In summary, a hand detector that jointly considered both foreground-background correlation and adaptive skin model on multiple resolution square blocks was proved to be useful to maintain robustness to various illumination conditions. It can not only achieve around 90% F_1 score over multiple conditions but also enhance the gesture recognition performance almost by a factor of 2. Therefore, it can maintain good robustness to the in-car illumination variations, as shown in the very left side of Figure 24 in Chapter 6.



(a) Category I: Normal illumination, well matched



(b) Category II: Bright illumination, slightly mismatched



(c) Category III: Dark illumination, highly mismatched

Figure 12: Precision-recall curves for 3 different testing categories. We can see that the purple lines have the maximum F_1 scores when precision is required to be more than 90%. Applying high-level feature fusion (also known as score fusion in this work) scheme is in general better than MAP adaptation in well-matched conditions (Category I), but deteriorates in slightly-mismatched conditions (Category II), degrades severely when the condition is highly-mismatched (Category III). The proposed framework can solve this difficulty and make the dark illumination condition (Category III) achieve the biggest improvement.

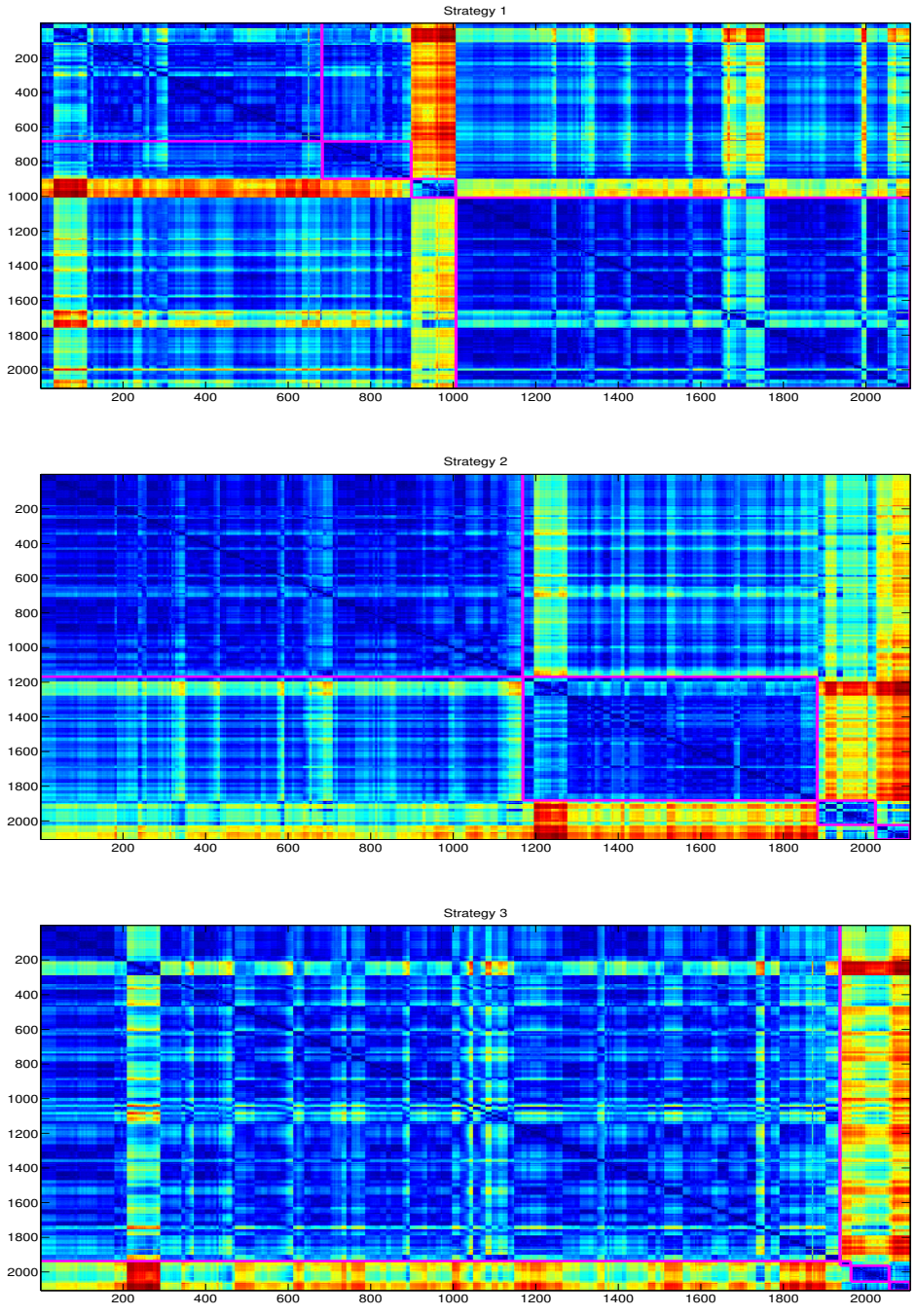


Figure 13: Comparisons of different categorization strategies. Standard K-means on histogram gave the most balanced clusters

Figure 14: Detection example: before and after uniform histogram equalization. Clearly, the false rejection is reduced greatly, but some false alarms are also observed.

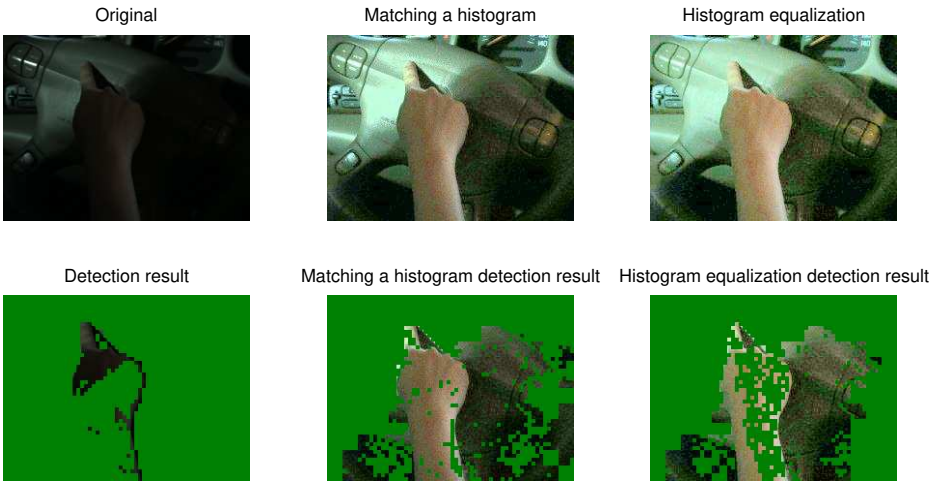


Figure 15: False alarm regions caused by applying histogram equalization; note the right regions are transformed to be almost surely skin color while they are actually not, which is equivalent to adding more of the wrong samples during the skin color modeling and degrading the detection.

CHAPTER 4

FEATURE EXTRACTION AND MODELING

In this chapter, the main focus is to address the issues of the whole gesture recognition system with previously mentioned hand detection results. As mentioned in the previous chapters, the gesture trajectory conveys the identity to be recognized for applications similar to our gesture recognition system. Therefore, the trajectory of the English letter should be properly determined to ensure a trust-worthy recognition result. Humans frequently trace the tip of the forefinger to build up the writing trajectory. Since people may bend their forefinger during the writing, the three-dimensional image information might be needed, which is out of the scope of the present study. To simplify the study for the complicated in-car environment, the geometric center of the two-dimensional identified hand region will be considered to avoid the noisy detection made in fingertip detection.

As for the modeling strategies, most generic gesture recognition systems use sequential modeling techniques, such that a whole gesture is modeled by a single HMM [9, 38] or FSM[15], and the gesture that gives the highest model score will be chosen as the recognized gesture. Typical input features for these applications may involve position, velocity, and orientation [21], which can also be smoothed over several frames. The main challenge for in-car environment is its gesturing diversities. Since a user may put more attention on driving instead of looking at his or her own gesturing hands, these gestures are expected to be more noisy and may have meaningless gesture components, such as making a long starting stroke before actually beginning to perform the intended gesture. Directly normalizing the whole gesture like that proposed in [73] may cause the meaningless strokes to be misunderstood as part of a gesture.

In this chapter, with the raw features determined by the hand detection results, that is the position, window smoothed velocity and acceleration across frames, the following topics on modeling strategies are discussed. First, modeling with whole-letter and strokes are

discussed. Second, the multi-pass modeling that removed the starting and ending strokes is discussed. Third, the confusion pairs are identified to enhance the system robustness. Then, to enhance the system robustness to low frame rate issues, several interpolation strategies are discussed. The experimental results are presented in the final section.

4.1 Modeling with Hidden Markov Model

As mentioned in the introduction and background review chapters, the most intuitive way is to use the hidden Markov model (HMM) as a tool for modeling gesture trajectories and shape changes. This is a common choice even for the state of the art systems. As for the feature vector, both horizontal and vertical coordinates were first normalized to make sure the position vectors on both axes are within the range of $[0, 1]$. Next, corresponding velocity and acceleration on both directions were smoothed every 3 frames. Totally a 6 dimensional feature vector was produced for each frame. In the following discussion, this feature vector is the default feature to be used for the gesture recognition system. Next, detailed modeling strategies which can extend this intuitive configuration are discussed.

4.1.1 Whole-Letter Modeling

The most intuitive baseline system used the normalized position, velocity, and acceleration information to form an input vector sets $\boldsymbol{o}$. As the HMM with continuous state observation probability is used, the log likelihood score for the $i-th$ English letter can be computed as:

$$log[P(o|\Lambda_i^{whole\ word})]. \tag{95}$$

where the terminologies $\boldsymbol{o}$ and $\boldsymbol{\Lambda}$ are defined in Section 2.3.3. So if M gestures are modeled, there will be M sets of HMMs $\boldsymbol{\Lambda}_1, \cdots \boldsymbol{\Lambda}_M$, the one gives the highest log likelihood score will be the recognition result.

4.1.2 Stroke Modeling

In the present preliminary studies, a whole gesture was modeled by a single HMM, and the model that gave the highest score was chosen as the recognized gesture. For recognition

inside a car, however, a gesture can be noisy because the user didn't actually look at the gesture during writing. Furthermore, for gestures, such as English letters, one gesture may have several different ways of writing. Thus it will need more effort to collect the data for several different stroke orders even for the same letter. This motivates the study of stroke modeling.

The idea of stroke modeling is similar to the stroke modeling for hand-written Chinese character recognition [74, 75, 73] or phoneme modeling for speech recognition [76]. This is due to its infeasibility to individually collect and model thousands of commonly used characters. Therefore, a lexicon is defined to regulate the way to compose a character with strokes similar to [77] that stroke modeling may be able to take care of gestures with different stroke orders. This strategy gives the possibility of increasing the discriminative power by explicitly modeling the intra-stroke correlation with geometric properties as a post processing stage, and the potential to remove noisy strokes. The experimental results showed these advantages are verified that both issues, insufficient data and multiple user variabilities, are effectively addressed.

To deal with the difference between in-car stroke modeling and written text stroke modeling, this work treated the intermediate strokes as meaningful strokes while building the lexicon. With this framework, the meaningless beginning and ending strokes of the gesture can be handled and the re-normalization of all the meaningful strokes can be done accordingly. In addition, this study also incorporated stroke relationship modeled by some geometric properties. Finally, fusion with the original whole-letter model can also be applied.

So far, using the detection scheme proposed in Chapter 3, the proposed framework over prior arts are as follows:

1. Use background and foreground information and adaptation to produce good hand detection.
2. The strokes can be trained with more effective training data.

3. It can potentially increase the following discriminative power:
   (a) By removing meaningless strokes.
   (b) By explicit modeling the relationship among strokes.
4. Give flexibility to future data with different stroke orders.

To be more specific, a lexicon is first defined for each English letter gesture. For example, letter P is defined as ►↓↑ ❩ ■, as shown in Table 3. Multiple stroke sequences are also possible, but this work only used one stroke sequence per letter in current study. With each stroke modeled by an HMM with the same set of features but with a smaller number of states, the log likelihood score for the i-th letter can be computed as:

$$log[P(o|\Lambda_i^{strokes})] = \sum_{k \in D_i} log[P(o|\Lambda_k^{stroke} \cap \Lambda_i)], \tag{96}$$

where D_i is the stroke set allowed for the $i-th$ letter defined in the lexicon. An example segmentation result is shown in Fig. 16.

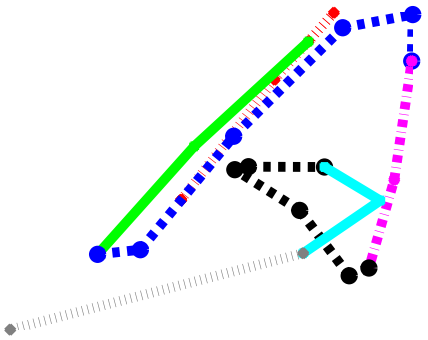


Figure 16: An example of segmenting a letter with stroke HMM, lines with different colors are strokes decoded.

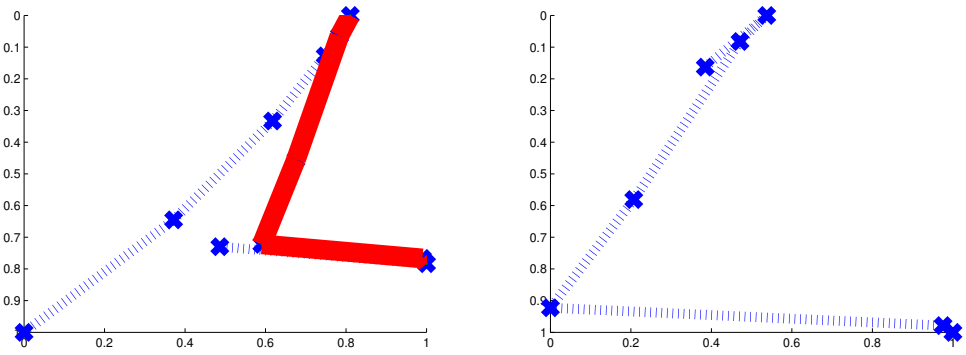
4.2 Multi-Pass Modeling: Removing Move-in and Move-out Strokes

As mentioned before, in-car gesture may include irrelevant starting and ending strokes, as shown in the left sub figure of Fig. 17. These improper strokes need to be removed and the meaningful gesture parts need to be re-scaled afterward because these varying length

Table 3: Most Commonly Observed Stroke Orders in Our Dataset

A	▶↙↗↘↖→■	J	▶→←↓◡■	S	▶ᔓ■
B	▶↓↑ᔑᔑ■	K	▶↓↗↙↘■	T	▶→←↓■
C	▶ᔕ■	L	▶↓→■	U	▶◡■
D	▶↓↑ᔐ■	M	▶↓↑↘↗↓■	V	▶↘↗■
E	▶↓→↖→↙→■	N	▶↓↑↘↑■	W	▶↘↗↘↗■
F	▶↓↑→↙→■	O	▶↻■	X	▶↘↑↙■
G	▶ᔕ↓↖→■	P	▶↓↑ᔑ■	Y	▶↘↗↙↓■
H	▶↓↑→↑↓■	Q	▶↻↘■	Z	▶→↙→■
I	▶→←↓←→■	R	▶↓↑ᔑ↘■		

strokes can make the gesture model biased. In order to cope with this issue, the HMM is used in a two-pass manner. The HMM with $START$(▶) and END(■) models is used for decoding, and then these strokes and re-normalize the remaining position vectors are removed, and finally the re-normalized vectors are passed to new set of HMMs without $START$ and END. As shown in the right part of Fig. 17, this strategy works reasonably well.



(a) Original Trajectory (b) Re-Normalized Trajectory

Figure 17: An example of applying two-pass decoding on the trajectory. The starting and ending part of the trajectory are effectively removed. Note in (a), the red solid lines are the meaningful strokes

All previous configurations did not consider the explicit relation among strokes. By

considering relationship between any two strokes, with the assumption of uniform stroke distribution, and considering the fact of probability scale difference, the modified score function for $i-th$ letter can be written as:

$$g(o|\Lambda_i) = \sum_{k \in D_i} log[P(o|\Lambda_k^{stroke} \cap \Lambda_i)] + \gamma \sum_{\substack{k,h \in D_i \\ k \neq h}} log[P(o|\Lambda_{k,h}^{geometric} \cap \Lambda_i)], \quad (97)$$

where γ is a positive constant with a typical value in the range of $[0, 1]$. Now, the final two-pass stroke modeling algorithm is summarized as follows:

Algorithm 1: Two-pass letter recognition

input : sequence of position vectors o
letter models with starting/ending strokes Λ
letter models without starting/ending strokes Ξ
output: recognized English letter
for $\Lambda_i \in \Lambda$ **do**
compute $log[P(o|\Lambda_i)]$
end
$stroke\_sets \leftarrow \underset{\lambda \in \Lambda}{argmax}\{log[P(o|\lambda)]\}$
// $\mathcal{N}$: scale normalization function
$o\_ \leftarrow \mathcal{N}\{o - starting\ \&\ ending\ strokes\}$
for $\Xi_i \in \Xi$ **do**
compute $log[P(o\_|\Xi_i)]$
end
return $\underset{\xi \in \Xi}{argmax}\{log[P(o\_|\xi)]\}$

In practical application, instead of picking up the maximum score over all candidates, this study only chose two candidates with the first and second highest stroke scores. According to previous results, these candidates will give relatively big counts in the confusion matrix, so they are the most competitive confusion pairs to be considered. Furthermore, this study also observed if geometric properties are not strong enough, fusing the whole letter score can be helpful, as will be discussed soon in Section 4.5.

4.3 Enhancing Modeling with Confusion Pairs

In the previous section, the modeling strategy using a basic unit called stroke was presented. However, it was seldom discussed that the modeling strategy can also focus on how strong the competition is for each candidate and their competitors. This idea was similar to those studied in the automatic speech recognition community under the area called discriminant training (DT) [78]. But it turns out not all competitors are worthy in this application.

Following the motivation of modeling this kind of competition, this study will further model the stroke by another set of features and models that can improve both the description of the trajectory shapes and trajectory similarities. According to test results, a commonly used shape descriptor family called Fourier descriptor could be quite useful for this purpose. But directly using this feature may not be effective because it is too sensitive to some local variations so that one may falsely reject the correct candidates. By applying the two-pass decoding method presented previously, the noisy effect can be reduced significantly. Thus the resulting trajectories can be directly applied to the two-pass processed trajectory. Subsequently, another pass to those easily confused gesture pairs called ”confusion pairs” is applied.

To identify the confusion pairs, it is natural to check the confusion matrix during modeling training and validation processes. Once it was done, one can proceed to model the easily confused gesture pairs. Due to the stroke sequence similarity, it is not always trivial to identify which strokes are more crucial than the rest strokes for the easily confused gestures. However, with the Cosine distance used as the distance metric, one can identify the most critical differences between these gestures. After that, one can further model them with Fourier descriptors. Since the periodic condition for Fourier descriptors must be enforced, the even data mapping was applied.

4.4 Enhancing Frame Rate Robustness with Interpolation

Although the previous modeling strategies have been proved to be useful, these strategies may not be able to model the whole gesture well due to the lack of image frames. So the main focus of this section is to discuss the robustness to low frame rate issues. Note that gesture sequences are usually sparsely acquired at a low frame rate of 15, 30, or 60 frames per second (fps). Since the speech signal is sampled at 16KHz and processed at 100 fps, and weighted by a Hamming window of length 20 to 30 ms. The 100 fps frame rate for speech signal is much higher than that of the gesture sequences, the techniques used in speech technologies cannot be directly applied. In addition, gestures mostly rely on the raw trajectory indices, such as position and velocity sequences [38, 9, 79], and may not be as strong as speech feature vectors in terms of their discriminative power. More specifically, the main focus of this section is to enhance the feature representation that is capable of effectively describing the detailed gesture structure and reducing the impact of data sparseness on the gesture recognition system performance, caused by the lack of raw samples.

To ensure an adequate number of samples for whole-gesture modeling, this study adopts feature interpolation as a tool of "missing data reconstruction". Several possible interpolation algorithms are proposed to ensure the good use of limited data with only a small number of samples acquired at the rate of 15 fps. Interpolation strategies up to third order polynomials or second order continuous derivatives [16, 80] are discussed.

In addition to the problems induced by insufficient frame number, there exist abnormal elements caused by the missing of the critical frame element(s) and seriously scattered frame due to the noisy nature of hand detection. In fact, if one element of an observed vector is seriously scattered to and from its neighbors, the forward and backward estimation procedures of the HMMs are corrupted. The reason comes from the fact that the objective function $F(\mathbf{\Lambda})$ becomes highly non-smoothing. Consequently, the optimization may be trapped in a local minimum. This critical case is due to the abnormal frames caused by

non-uniform illumination conditions, which can be unpredictable once a hand first enters an active region of the camera. It is believed the effect of shot noise can be smoothed with reasonable number of data.

4.4.1 HMM under Low Frame Rate Conditions

Practically, the low frame rate acquisition condition may prevent one from properly training HMMs with the limited number of states suggested above. Consider an ℓ-th gesture with T samples that are used to train an HMM of N states with their observation probability density functions modeled as Gaussian mixture models (GMMs). T can be thought of as the number of points for training a whole-gesture model or the number of points that is aligned to train a stroke HMM. Several critical cases may occur under low frame rate conditions:

1. A state involves no data point when $T \leq N$.
2. A state involves only one or two points when $N \leq T \lesssim 2N$.
3. A state involves interior/junction points between two successive strokes.

Obviously, the first two cases cause a failure or instability of evaluating the associated HMM due to the small sample size. The HMM with "state jump" will not resolve the problem because it will sacrifice the model discriminative power due to the blurred observation densities caused by sharing samples.

In this study, an interpolation scheme is proposed to alleviate the problem of sample sparseness so that the first two extreme cases can be overcome. The third case, however, can usually happen while performing stroke modeling. Since the exact labels for the stroke junctions are not available, when the Viterbi alignment makes mistakes, it can be quite vital especially for a data set containing a lot of stroke order variations. This is because making a segmentation error can include points that are actually quite far away from the true strokes, and greatly bias the model.

A careful inspection of the HMM and its related definitions and the forward-backward algorithm leads to the following well-known fact: "*To properly evaluate an N-state left-to-right HMM without state skipping, all the observed vectors must be at least N frames.*" This property can be further explained as follows.

Assume an observed vector $\boldsymbol{o}_t$ is abnormal such that its number of element T is less than N. The corresponding $a_{i,j}$ and many parameters cannot be evaluated because i and j must be transited from 1 to N with step size 1. Then, the forward-backward algorithm cannot be completed due to lake of samples. Therefore, for an N states HMM, the observed vector must contain at least N elements. Otherwise the corresponding parameters in the above sequence cannot be properly evaluated. This is obvious by checking the HMM evaluation process, such as those shown in the following Eq. (98) and Eq. (99).

$$\alpha_{t+1}(j) = \sum_{s_1,s_2,\ldots,s_t=1}^{N} \pi_{s_1} a_{s_1,s_2} a_{s_2,s_3} \ldots a_{s_t,j} b_{s_1}(\boldsymbol{o}_1) b_{s_2}(\boldsymbol{o}_2) \ldots b_{s_t}(\boldsymbol{o}_t) b_j(\boldsymbol{o}_{t+1}). \tag{98}$$

$$\begin{aligned} \beta_t(i) &= \sum_{j=1}^{N} a_{i,j} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j), \quad 1 \le t \le T-1, \ 1 \le j \le N \\ &= \sum_{s_{t+1},\ldots,s_T=1}^{N} a_{i,s_{t+1}} a_{s_{t+1},s_{t+2}} \ldots a_{s_{T-1},s_T} b_{s_{t+1}}(\boldsymbol{o}_{t+1}) \ldots b_{s_T}(\boldsymbol{o}_T) \end{aligned} \tag{99}$$

$$\begin{aligned} P(\boldsymbol{O}|\boldsymbol{\Lambda}) &= \sum_{i=1}^{N} P(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T, \boldsymbol{s}_{i=1}|\boldsymbol{\Lambda}) \\ &= \sum_{i=1}^{N} \beta_1(i) b_i(\boldsymbol{o}_1) \boldsymbol{\pi}_i, \end{aligned} \tag{100}$$

where all the $a_{i,j}$s within the serial mulitplications should have finite values.

The induced fact above can be further generalized from a different point of view. To properly evaluate an N-state HMM, some critical elements to properly describe an objective should not be missed. Consequently the frame number should be larger than N. Moreover, the worst case occurs when the observed vector involves seriously and isolatedly scattered elements or outliers such that the differences between the element and neighbors are much larger than the overall averaged distance between two adjacent elements. It leads to the

result that target probabilistic parameters may not be properly evaluated. Note that several abnormal observed vectors (i.e. those corrupted by shot noise) will ruin the corresponding parameters and mislead the optimization procedure. Consequently, since all the HMM parameters involve the observed vector, the corrupted observations will bias the whole HMM procedure so that the resulting estimation or probability can be misled.

If a state incorrectly covers too many elements, which is equivalent to an $\boldsymbol{o}_t$ involves two or more strokes of an English letter, the probability model may concentrates on elements of one of the strokes only. Consequently, the important elements of the rest strokes are incorrectly modeled and are eventually ignored by the erroneous probabilistic likelihood model.

In these situations, some elements to calculate $P(\boldsymbol{O}|\boldsymbol{\Lambda})$ are incorrect so that the resulting $P(\boldsymbol{O}|\boldsymbol{S},\boldsymbol{\Lambda})$ is misled. Thus, the parameters a's, b's, α's, β's, γ's, and ξ's are incorrect that they may run out of their range of definition or cannot be properly evaluated.

Moreover, according to the error analysis theory [81], since the mean mixture update (Eq. (101)) listed below as well as other parameter updates are in the form of A/B, the estimation error can be at the same scale of $|||A||/||B|||$ when both terms are very small and thus can make the trained HMM biased and unstable.

$$\hat{\boldsymbol{\mu}}_{s_t,k}^{(new)} = \frac{\sum_{t=1}^{T} \gamma_t^{(old)}(s_t,k)\boldsymbol{o}_t}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t^{(old)}(s_t,k)}. \tag{101}$$

As some critical elements are not fine enough such that the differences between the element and neighbors are much larger than the overall averaged distance between two adjacent elements, it will be difficult to properly describe the objective of modeling. The reason comes from the non-robustness nature of the MLE, which is a generalized form of the least squares method. In a least square type optimization problem, any significantly scattered data shall introduce a prominent bias of the resulting estimation [82]. The last situation means more than one element is significantly contaminated by noise with a large

noise level, which leads to a small sign-to-noise ratio . If the number of these abnormal elements increases, the corresponding variance and hence standard deviation increases, too. Consider the case of two variables, say to recognize the letter B that may be confused with the letter D because of several frames to describe the junction of two curved strokes of B are missed. An analogy of this can be seen in 1D cases. Since after proper projection, one can always make classification in a 1D projected space. Assume that two 1D random variables X, Y each have respectively means m_1 and m_2 with the same variance σ^2, denote the cumulative distribution function of standard Gaussian distribution as $\Phi()$, the misclassification error is $2\Phi(\frac{m_1-m_2}{\sigma})$. Note the intersection point of two probability density functions is at the value of $\frac{(m_1+m_2)}{2}$. when both variables are corrupted with a zero mean noise $\mathcal{N}(0, s^2)$, the misclassification rate will end up being $2\Phi(\frac{m_1-m_2}{\sigma+s})$. Also note the Φ is a monotonic increasing function due to its cumulative distribution function definition, and since $m_1 < m_2$, those within the parentheses are a negative value, that is, $2\Phi(\frac{m_1-m_2}{\sigma}) < 2\Phi(\frac{m_1-m_2}{\sigma+s})$. Therefore, after being corrupted by noise, the misclassification error will increase.

These discussions indicate that, for an observed vector $\boldsymbol{o}_t$ whose number of states T is less than $2N$, techniques such as interpolation should be applied to add reasonable elements to exclude the possibility of failed GMM-HMM evaluation. For the present study, the insufficient states are just the case of insufficient frame resolution to properly resolve an English letter. Also, this study applies the Gaussian model on $\mathbb{R}^D$ space with diagonal covariance matrix, thus $2D$ parameters are to be determined for each Gaussian. So one has to ensure at least this number of vectors are aligned to each state. Fortunately, for the examined dataset, the interpolation method can provide necessary additional frames. Current tests showed that a frame number around 5 times larger than N is a safe guidance for this purpose. In some sense, the factor 5 reflects the requirement that a curve should use at least 5 points to properly capture its curvature. *Therefore, the interpolation method is always helpful to achieve this goal.* Now, to derive the interpolation strategy, a mathematical discussion is given as follows:

First, a stroke is defined as follows:

Definition 1. *A stroke is a non-self-intersect curve.*

With this definition, one can define a set of strokes as basic units for gestures. In this work, to model English alphabetical gestures, several basic stroke categories can be defined as:

- **Line stroke**: A stroke that can be represented by a straight line at several orientations. In this work, 8 orientations are used.
- **Curvy stroke**: An arc that goes clockwise or counter-clockwise.
- **Move-in and Move-out strokes**: Model the starting and ending strokes of a gesture. It can be composed of any of the above two items.

With these properly defined strokes, each of the 26 English letters can then be modeled as a sequence of strokes, as shown in Table 3. For example, self intersecting gestures such as "X" can be decomposed into several non-intersecting units, "►, ↘, ↑, ↙, ■ ", where starting and ending strokes are denoted as ► and ■, respectively.

Note that the above definition is different from our impressions that the letter "X" is represented by two not connected straight lines. However, this impressions misses three necessary elements, say "►, ↑, ■". Since this study does not carry out the three-dimensional image processing, it is almost impossible to properly exclude these three units. Therefore, they are included to properly define an English letter. In fact, this approach ultimately increases the complexity and difficulty of our gesture recognition.

To distinguish a curvy stroke from a straight stroke, this work assumes that it follows the conic section assumption, such as an ellipse, circle, parabola, one branch of a hyperbolic curve, or a straight line for the domain of gesture trajectories. Here the straight line is added to include the degenerated case from a curvy stroke to a straight stroke. This study chooses this mathematical model due to its simplicity and the possibility of covering most simple

physical movements with constant velocity, angular speed, or acceleration. The general form of conic section in the Cartesian coordinate system is:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{102}$$

In order to find the parameter for this general curve, at least 5 points are required for both $f \neq 0$ and $f = 0$. That is, this equation only has 5 unknowns, and one can always scale other unknowns by $1/f$ if $f \neq 0$. Note that when all the second order terms become 0, this is just a straight line. Under this assumption, one needs at least 5 points to represent each stroke such that any two strokes can be disambiguated.

By the above curve-fitting model, 2 and 5 points seem to be adequate to represent a straight and a curvy stroke, respectively, as shown in the topological view mentioned in Eq. (102). However, in practical situation, a user usually cannot make a perfect straight line. Thus it is better to have more than 2 points to cover a reasonably small deviation from a straight line. For a curve, henceforth, people need to get more than 5 points to cover potential deviation from a perfect curve or part of gestures (PoGs) from adjacent strokes. Therefore, in general, it is suggested that one should make a stroke model containing at least 6 nodes. That is, each stroke should be modeled with at least 6 states.

This idea can be extended to the whole-gesture modeling. However, it is not guaranteed that the decoded HMM state boundary will coincide with the stroke junction. Therefore enough states must be used to ensure the discriminative power to identify curvy PoG from several piecewise continuous straight PoGs.

Eventually, taking the maximal number of strokes among all gestures can resolve this issue. In the present case, it will be 8 strokes, as shown in Table 3, if each stroke has 5 states as suggested above, there are a total of $8\times5-(8-1) = 33$ or $8\times5 = 40$ states when junction points are assumed to share or not share the same state, respectively. So a number from 33 to 40 states should be a good choice to build a whole-letter HMM for each gesture. Having a little-bit more states than these numbers is not harmful unless large amount of samples

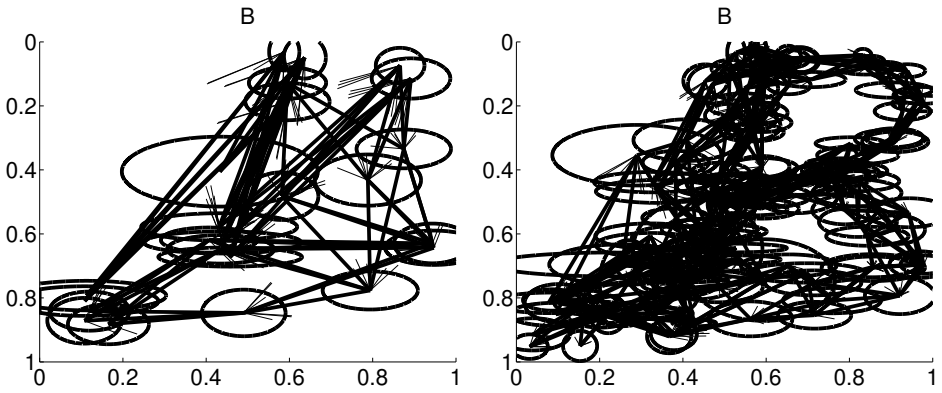
are not given to train desired models. Unfortunately, gestures sampled at low frame rate make this requirement not easy to meet. As will be discussed in Section 4.5.5.1, where sample points are randomly removed from good data with more than 33 sample points to simulate the low frame rate issue. Again, the reason is that the hand movement speed is subject to change under this scenario. Next, different existing interpolation strategies from previous literature are discussed in order to choose the most appropriate one for this work.

4.4.2 Interpolation Theory

Previously, interpolation techniques are widely used in the domain of data analysis and image resizing [83, 84] or reconstruction [85, 86]. Although interpolation may introduce aliasing effect due to side ripples of the interpolation window, it is even more harmful to apply low pass filtering because the removed high-frequency components frequently contain the stroke transition, which is a key to distinguishing curvy and piecewise straight parts of gestures. Therefore, unlike pure data reconstruction problems, reasonable interpolation techniques may provide more flexibility to properly model the desired gestures.

When a sufficiently large number of points are added by a reasonable interpolation strategy, in this case, 5 points, the extreme issues caused by insufficient frame number, say $T \leq N$ and $N < T \leq 2N$, can be resolved. This is because interpolation can provide a way to use the low frame rate data along with the sufficiently sampled data for a robust model training. As seen in Fig. 18, in the very beginning, the 8-state model cannot fully represent the letter “B”, but after interpolation, it can be clearly visualized. However, when too many points are added, the weights of the artificially added points will increase. This will make the resulting model biased and the performance will likely drop.

Among all the classical interpolations, the linear and cubic spline interpolations are widely applied to provide piecewise continuous lines and/or curves between the data points. Besides these two interpolations, there exists a set of interpolation families called splines [87, 88, 89] which put major emphasis on the continuity of the derivatives and are also shown to have smaller frequency-domain ripples compared to the cubic spline interpolation



(a) no interpolation (b) with interpolation

Figure 18: Examples for model visualization with covariance ellipses: (a) An example of HMM "B" with 8-state 3-mixture trained by original data. (b) An example of HMM "B" with 40-state 3-mixture trained by adding 5 points between every two original points with linear interpolation. Letter cannot be easily visualized in (a) but is clearly revealed in (b), the value of doing interpolation is obvious.

[90, 91]. Generally, these interpolation techniques make each interpolated point a function of the two or more adjacent original points [87, 88, 89]. Now, how the linear, cubic spline, and cubic B-spline interpolation techniques [92, 93, 94, 95] impact the proposed gesture recognition system will be checked in the subsequent paragraphs.

Consider a two-dimensional line described by a series of data points (x_0, y_0), (x_1, y_1), . . ., (x_m, y_m). An interpolation formula is used to generate a series of continuous curvy lines from these data points. This study uses the interpolation methods to add one or more data points to rescue the problem induced by the insufficient frame rate. For example, the linear interpolation between two successive points gives a series of piecewise straight lines connecting to each other:

$$y = \frac{x_{k+1} - x}{x_{k+1} - x_k} y_k + \frac{x - x_k}{x_{k+1} - x_k} y_{k+1}, \; x \in [x_k, x_{k+1}], \; k = 0, 1, \ldots, m-1 \tag{103}$$

For data points describing a curve, the deviation from these piecewise continuous lines to the curve monotonically decays as the number of data points m increase. Conversely, as m decreases the deviation may be significant but is still finite. In general, the linear interpolation is free from the numerical oscillation but may be too rough to approximate

a curve when m is too small. Obviously, if the actual line is straight, $m \geq 1$ is enough to describe the geometrical details.

If one uses the following cubic polynomial to approximate the curve between two successive points and requires that these piecewise curve and their first and second order derivatives are continuous, the following cubic polynomial with proper relations between them can be found in the book [82] to satisfy the continuous constraints for their derivatives.

$$\begin{aligned} y_k(x) =& a_k(x - x_k)^3 + b_k(x - x_k)^2 + c_k(x - x_k) + d_k, \\ & x \in [x_k, x_{k+1}],\ k = 0, 1, 2, \ldots, m-1 \\ a_k =& \frac{1}{\Delta x_k^2}\left(-2\frac{\Delta y_k}{\Delta x_k} + y'_k + y'_{k+1}\right) \\ b_k =& \frac{1}{\Delta x_k}\left(3\frac{\Delta y_k}{\Delta x_k} - 2y'_k - y'_{k+1}\right) \\ c_k =& y'_k, \quad d_k = y_k \end{aligned} \tag{104}$$

Note that, if the requirements of the continuity of the first and second order derivatives are dropped, the equation is reduced to the linear interpolation. Since the curve is only piecewise continuous, a sufficiently high order derivative (third) is discontinuous. In other words, the cubic spline interpolation suffers from the spurious oscillation or Gibbs' phenomenon [96] when the first and second order derivatives of the curve are not smoothly distributed. The requirement for less oscillation is important because the oscillated trajectory, which can form a false stroke, will be falsely modeled. Define the normalized derivative parameters α_k and β_k for consecutive frames as follows [94].

$$\begin{aligned} m_k =& \Delta y_k / \Delta x_k = (y_{k+1} - y_k)/(x_{k+1} - x_k) \\ \alpha_k =& y'_k/m_k,\ \ \beta_k = y'_{k+1}/m_k,\ \ s = \left|\sqrt{\alpha_k \beta_k}\right| \end{aligned} \tag{105}$$

The necessary monotonic condition of $y(x)$ in the interval (x_k, x_{k+1}) is that [94]:

$$sgn(\alpha_k) = sgn(\beta_k) = sgn(m_k) \tag{106}$$

The monotone constraint can be satisfied as follows.

- [1.] If $\alpha_k + \beta_k - 2 \leq 0$, if and only if the necessary conditions Eq.(106) are satisfied.
- [2.] If $\alpha_k + \beta_k - 2 > 0$, if and only if one of the following conditions and the necessary conditions Eq. (106) are satisfied.

  a. $2\alpha_k + \beta_k - 3 \leq 0$;

  b. $\alpha_k + 2\beta_k - 3 \leq 0$;

  c. $\alpha_k^2 + \alpha_k(\beta_k - 6) + (\beta_k - 3)^2 < 0$.

The last inequality is derived from the condition that the quadratic algebraic equation of $y'(x) = 0$ has no real root. Also, the boundary of $y'(x) = 0$ having repeated real roots is an ellipse. The ellipse has a tangent point with $\alpha_k = 0$ at $(\alpha_k, \beta_k) = (0, 3)$ and has another tangent point with $\beta_k = 0$ at $(\alpha_k, \beta_k) = (3, 0)$ and passes through the point $(\alpha_k, \beta_k) = (3, 3)$. The whole four conditions generate the monotone region on the (α_k, β_k) space enclosed by the lines of $\alpha_k = 0$ and $\beta_k = 0$ and the upper boundary of the ellipse. It is interesting to note that the monotone region on the (α_k, β_k) space outside the ellipse corresponds to the cases with their local maximum and/or minimum points located in region(s): $x < x_k$ and/or $x > x_{k+1}$. The monotone condition was summarized as follows by Fritsch and Carlson [94]:

$$\begin{aligned} 0 <& \alpha_k + \beta_k < 3 + s, \quad \text{if } 0 \leq \beta_k \leq 3 \\ 3 - s <& \alpha_k + \beta_k < 3 + s, \quad \text{if } 3 < \beta_k \leq 4 \end{aligned} \tag{107}$$

For the sake of clarity, the above inequalities of defining the monotone region are plotted in Fig.(19). Wolberg and Alfy [97] pointed out that the monotonic condition cannot always be satisfied. Therefore, the curve could oscillate in the neighborhood of an outlier.

Next, an alternative interpolation strategy will be discussed.

4.4.3 Constrained cubic B spline interpolation

A cubic B-spline curve on the (x, y) plane is a piecewise continuous cubic curve defined by $n + 1$ control points, say $\vec{P}_0, \vec{P}_1, \ldots \vec{P}_n$, where $\vec{P}_0 = x_0\vec{i} + y_0\vec{j}, \ldots$. Namely, these control points define the shape of the piecewise continuous curve. For a clamped cubic B-spline

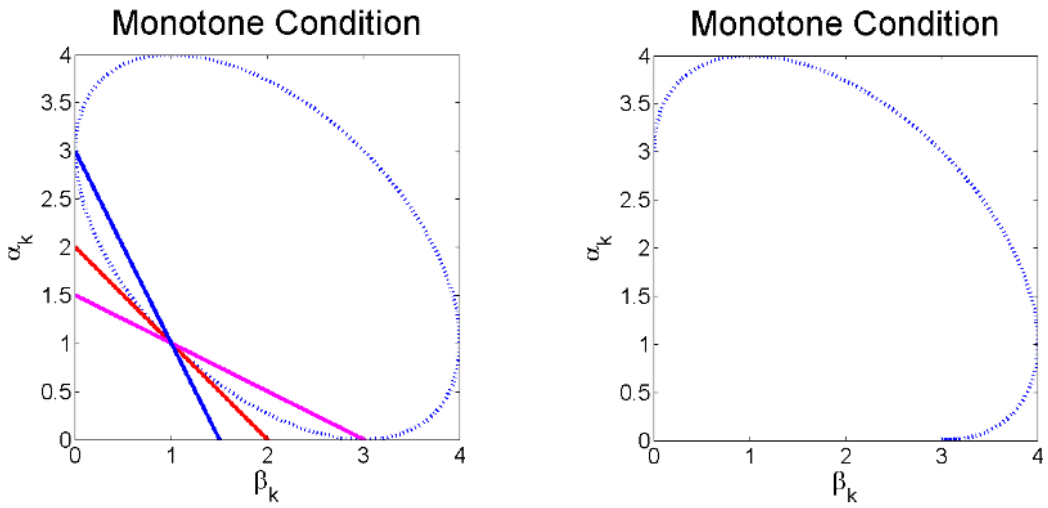


Figure 19: The region of the derivatives that satisfy the monotone condition.

curve, these control points constitute $p = n + 4$ knot vector, $u_0, u_1, \ldots, u_{n+4}$, with the first four and last four knots clamping the curve. Then, the curve passes two end control points. In this study, the curve is not clamped, the two ends of the curve do NOT match the two end control points.

The B-spline blending or spline functions are defined by the following de Boor recurrent formulas.

$$B_{i1} = \begin{cases} 1, & \text{if } t_i \le t \le t_{i+1}; \\ 0, & \text{otherwise.} \end{cases} \tag{108}$$

$$B_{ik} = \omega_{ik} B_{i,k-1} + (1 - \omega_{i+1,k}) B_{i+1,k-1} \tag{109}$$

$$\omega_{ik} = \begin{cases} \frac{(t-t_i)}{(t_{i+k-1}-t_i)}, & \text{if } t_k < t < t_{i+l-1} \\ 0, & \text{otherwise} \end{cases}$$

where the knot vector u_i's are defined by parameters t_i. An example of the cubic B-spline interpolation curve and interpolation relations are respectively defined as follows.

$$C(t) = \sum_{i=0}^{n+2} B_{i,3}(t)\vec{P}_i, \quad t \in [0, n] \tag{110}$$

$$C(i) = \sum_{i=0}^{n+2} B_{i,3}(i)\vec{P}_i = \vec{D}_i \tag{111}$$

in which $\vec{D}_i = x_i\vec{i} + y_i\vec{j}$ and denotes the data points. After solving Eq.(111) to obtain $\vec{P}_i$s, we have the B-spline interpolation curve. In the computer-aided design field, people give and properly modify the control points to generate the desired curve. Since the expression of a B-spline curve is a linear combination of many products $B_{i,k}\vec{P}_i$, it is easy to prove the following relation.

$$L\{C(t)\} = L\{\sum_{i=0}^{n+2} B_{i,3}(t)\vec{P}_i\} \tag{112}$$

$$= \sum_{i=0}^{n+2} L\{B_{i,3}(i)\vec{P}_i\} \tag{113}$$

$$= \sum_{i=0}^{n+2} B_{i,3}(i)L\{\vec{P}_i\} \tag{114}$$

where the operator L is a linear transformation. Consider a B-spline curve defined by a series of control points. de Boor proved that one can properly use more and more control points to represent the same curve. As the number of inserting control points increase to infinite, the control polygon formed by the control points asymptotically approaches the curve. Based on this theorem, he further showed that a cubic B-spline curve has the following important and convenient properties.

1. The curve is contained in the convex hull of control poly-line formed by all the control points.

2. Changing the position of P_i only affects the curve on intervals defined by (u_i, $u_{i+1}, u_{i+2}, u_{i+3}, u_{i+4}$).

3. If the control points do not repeat themselves, the piecewise curve is C^3 continuous at a knot point such that the curve is continuous up to the third order derivative.

4. The B-spline has the shape preservation property that any straight line crosses the spline curve no more than it crosses the control polygon. Thus the curve has the "variation diminishing" property.

5. If an affine transformation is applied to the B-spline curve, the resulting curve can be constructed from the affine images of its control points.

Eq.(110) indicates that the $i-$sgement piecewise cubic B-spline curve depends only on a finite number of control points. This important property is the fundamental of the shape preservation of item 4; the curve is contained in the convex hull of the control polygon in the first item. These five properties make a B-spline curve have a well-shaped configuration [98, 99] and follows the topology of the control poly-line. Therefore, it is widely applied in the field of computer-aided design. Note the last point (item 5) is equivalent to the desired "shape preserving" corollary, which was proved in [98]:

Corollary 1. ***Shape preservation*** *A spline crosses any straight line no more often than does its control polygon. In particular, if the control polygon is monotone (convex), then so is the spline.*

This corollary implies the small oscillation property of the cubic B-spline curve compared to the cubic spline curve interpolation. For the sake of completeness, degrees of oscillatory phenomenon of the cubic spline curve and B-spline curve are discussed in the following content.

For convenience, consider the following segments in $x \in [-1, 0]$ and $x \in [0, 1]$ where two cubic polynomials are generated subject to the continuous conditions of y, y', and y'' at $x = 0$ node.

$$\begin{aligned} x &= -1, y = -\epsilon, y' = s_{-1} \\ x &= 0, y = 0, \\ x &= 1, y = \Delta, y' = s_1 \end{aligned} \tag{115}$$

The solution of these conditions give

$$y_{-1}(x) = s_0 x + (s_{-1} + 2s_0 - 3\epsilon)x^2 + (s_{-1} + 2s_0 - 3\epsilon)x^3 \tag{116}$$

$$y_0(x) = s_0 x + (-s_1 - 2s_0 + 3\Delta)x^2 + (s_1 + s_0 - 2\Delta)x^3$$

$$s_0 = [3(\Delta + \epsilon) - (s_{-1} + s_1)]/4 \tag{117}$$

This typical example denotes the situation that a small varied segment is in the neighborhood of a large varying region. The following property can be easily proven: "*The Gibbs oscillatory phenomenon exists in the small varied side of a threshold point where a curve changes from a region with very small variation to a rapid varied segment. That the monotone condition of the typical example is always violated in the region where* $x \in (-1, 0)$ *if* $\epsilon << \Delta$ *and/or if* $\epsilon << s_1$."

In fact, according to the two cubic polynomials shown above, their parameters take the following forms:

$$m_{-1} = \epsilon,\ \alpha_{-1} = \frac{s_1}{\epsilon},\ \beta_{-1} = \frac{3(\Delta + \epsilon) - (s_{-1} + s_1)}{4\epsilon} \tag{118}$$

$$m_1 = \Delta,\ \alpha_1 = \frac{3(\Delta + \epsilon) - (s_{-1} + s_1)}{4\Delta},\ \beta_1 = \frac{s_1}{\Delta}, \tag{119}$$

Obviously, $\epsilon << \Delta$ and $\epsilon << s_1$ lead to $\beta_{-1} >> 1$ and $\alpha_{-1} >> 1$, respectively, such that the monotone conditions of Eq.(107) are violated in the smooth segment where $x \in (-1, 0)$. On the other hand, in the segment $x \in (0, 1)$, the monotone conditions are satisfied except that $s_1 >> \Delta$ or $s_1 < 0$ with $|s_1| >> 1$. Thus, the property is verified.

Numerical results show that both a cubic spline curve and cubic B-spline curve have similar problems with respect to the Gibbs phenomenon. However, the degrees of oscillation and oscillatory range of the cubic spline curve are always worse than that of a cubic B-spline curve. One reason is that the cubic B-spline curve is always located within the convex hull defined by the control polygon while the cubic spline interpolation cannot guarantee this property to restrict the Gibbs oscillation. Another reason will be discussed below.

Consider a series of data points which have the following properties: $\ldots, |y_{i-2}-y_{i-3}|, |y_{i-1}-y_{i-2}|, |y_i - y_{i-1}| \leq \epsilon$ and $|y_{i+1} - y_i|, |y_{i+2} - y_{i+1}|, |y_{i+3} - y_{i+2}|, \ldots > N\epsilon$, where N is a large positive number. This is a curve involving a smooth segment next to a segment with a large variation. The $i-$th piecewise cubic B-spine curve remote from the two ends can be written to be

$$C_i(t) = \sum_{j=-1}^{2} B_{i+j,3}(t)\vec{P}_{i+j}, \quad t \in [0,1] \tag{120}$$

Namely, all the information of $C_i(t)$ is only defined by limited number of control points P_{i-1}, P_i, P_{i+1}, and P_{i+2}. After solving the relations between P_i's and $\vec{D}_i$'s, $C_i(t)$ is principally defined by $\vec{D}_{i-1}, \vec{D}_i, \vec{D}_{i+1}$, and $\vec{D}_{i+2}$.

For a cubic spline curve, one cannot clearly define the control points since it does not have the properties of Eqs.(111) and (120). The resulting piecewise cubic spline curve is solved from the requirements that the first, second, and third order derivatives of the two segments on the two sides of a data point should be continuous. Although the dependence of the piecewise cubic spline curve on $\vec{D}_i$ is similar to the cubic B-spline curve, the dependence attenuation of the former is always slower than that of the latter. In other words, the segmental cubic spline curve depends on a wider range of data points than that of a cubic B-spline segment. These dependencies are just the serial effect of (x_i, x_{i+1}) upon (x_{i-1}, x_i), (x_{i-1}, x_i) upon (x_{i-2}, x_{i-1}), ... Therefore, the degree and range of the Gibbs' phenomenon of the cubic spline curve is larger than that of the corresponding cubic B-spline curve.

The above discussions show that as the three interpolation methods are applied to a series of points defining the trajectory, they have the following properties.

1. The linear interpolation curve faithfully follows the trajectory points as the frame rate is fast enough to resolve the writing. Conversely, the curve deviates from the expected trajectory of the letter whenever the frame rate is insufficient.

2. The cubic spline interpolation curve gives continuous and smooth shape when the

trajectory points are smoothly distributed. Unfortunately, if the frame rate is insufficient and/or there is one or more unexpected abnormal trajectory points, the curve certainly suffers from significant numerical oscillation.

3. A cubic B-spline curve always provides continuous and smooth shape. Even in the extreme case of insufficient frame rate, the curve still roughly follows the shape defined by the control poly-line. It is certain that, if there is one or more unexpected abnormal trajectories, the curve is subject to oscillation. However, the degree of oscillation is smaller than that produced by the cubic spline interpolation [98, 99]

In the following experiments, these three interpolation methods will be compared and the advantage of the cubic B-spline interpolation will be numerically verified. Among these three strategies, cubic B-spline is the most robust one.

4.5 Experiments

Using the image data collected by a PointGray Dragonfly 2 camera, the post processing of stroke modeling with enhanced features was implemented. To verify the proposed framework, the following experiments were conducted:

1. Testing the performance of adaptive hand detector over naïve one for gesture recognition.

2. Comparing the stroke HMM, and whole-letter HMM.

3. Checking the impact of stroke normalization, and adding simple geometric information.

4. Examining different interpolation strategies for gesture recognition.

Within this set of 4,211 sequences, the most commonly seen stroke orders used for each of 26 letter gestures were picked for evaluation. Half of the data for each letter and recording condition were randomly chosen and used for training and testing separately. After trials

Table 4: Summary of Different Modeling Strategies

Configuration	Error Rate(%)
Whole-letter HMM (naïve skin model, baseline)	24.62
Whole letter HMM (MAP-adapted skin model)	14.86
Stroke HMM	12.73
Stroke HMM (+Re-normalization)	9.91
Stroke HMM (+Geometric)	9.17
Stroke HMM (+Whole letter fusion)	8.36
Stroke HMM (2-best)	5.54

and errors on the baseline system, 16-state single-mixture or 8-state 2-mixture and 2-state 32-mixture HMMs were used for whole-letter and stroke modeling, respectively. To make a fair comparison among different configurations, this study only evaluated the systems on a subset of 1,877 gesture sequences with more than 16 frames of hand motion. Input to all models were sets of vectors composed of position with both axes normalized into the range of [0, 1] and corresponding velocity and acceleration.

Therefore, the raw features consisted of 6-dimension data: the scaled position, corresponding smoothed velocity and acceleration. The position coordinates were scaled into a square region with both axes normalized to be within [0, 1]. A smoothing window of size 3 was used for computing both the velocity and acceleration in both axes.

4.5.1 Whole-letter Modeling

With the recommended parameter setting and the baseline system mentioned in the previous section, this work characterized the skin and non-skin models by Gaussian mixture models (GMMs) [100] with mixture counts determined by the technique in [72]. Consequently, the proposed adaptive algorithm based on the previous work significantly reduces the recognition error rate by 39.64% relatively, as shown in the first two rows of Table 4.

4.5.2 Sub-letter Stroke Modeling

Again, the recommended parameter setting was used, and testing was done on the same subset of data used in testing whole-letter models. One can see that the error rate for stroke modeling was 12.73%, which was better than 14.86%, produced by whole-letter modeling, as shown in the 2^{nd} and 3^{rd} rows of Table 4.

One reason for this slight degradation in performance could come from the fact that the current stroke modeling was based on the same feature set as those used in whole-letter modeling. For example, if both arcs in letter B were used to model the arc stroke that appears in B, P, R without considering their relative positions to the whole letter, this would make the resulting model unstable. To enhance the stroke modeling, this study adopted the position re-normalization and added simple geometric information in the subsequent work.

4.5.3 Position Normalization and Letter Geometry

With the gesture modeled by strokes defined in a lexicon and applying the two-pass re-normalization scheme, the error rate was reduced to 9.91%, a 22.15% relative error rate reduction over the original position-independent stroke models as shown in the 3^{rd} and 4^{th} rows of Table 4.

In addition, the present study applied the following geometric indices and used Eq.(97) and Algorithm 1 for letter recognition, as shown in Figure 20.

- Distances between endpoints from either stroke.
- Distances of stroke endpoints to the line connected by endpoints of the other stroke.
- Distance of the middle point of a stroke to the line connected by endpoints of the other stroke.
- Cosine value of two lines formed by endpoints of either strokes.

This work only considered the letter candidates with the first and second highest stroke model scores. One can see that this setup with $\gamma = 0.08$ can further improve the performance by 7.47% and 27.97% relatively over the re-normalized and original stroke models,

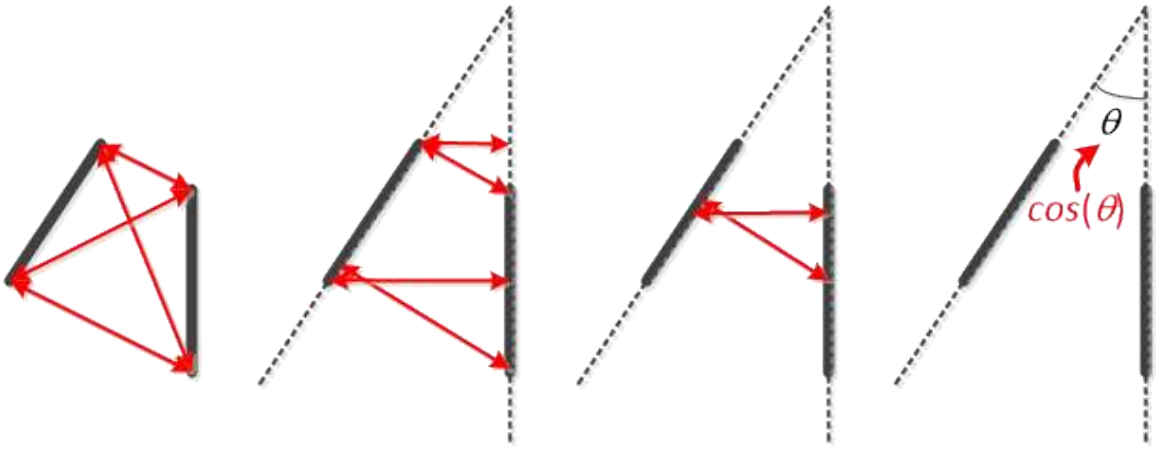


Figure 20: The geometric indices used for stroke modeling, distances are marked with red arrows. Also, the cosine value of the angle formed between two strokes was also applied.

respectively, as shown in the 3^{rd}, 4^{th}, and 5^{th} rows of Table 4. Moreover, by adding corresponding scores from the original whole letter models with weight $(\alpha, \beta, \gamma) = (1, 0.08, 0.5)$, one can further reduce the error rate from stroke models with geometric properties to the best performance of a 8.36% error rate. That represents a 66.04% relative error reduction from the baseline whole-letter system with naïve hand detection strategy.

Furthermore, if one considered the same setup but allowed any hit from candidates with the 2 highest scores as a correct recognition, i.e., 2-best, can have a much lower error rate of 5.54% can be achieved which is a 77.50% relative error reduction. This makes application design easy when additional information such as a dictionary or a language model can be integrated into the figure-written letter recognition systems. It also implies potential improvement by using more discriminative geometric properties.

4.5.4 Confusion Pairs

The 8 most easily confused pairs are identified by using cross validation on the training data; these pairs include the following:

Table 5: Confusion Pairs Identified with 5-fold Cross Validation

confusion pair	(B,P), (D,P), (I,Z), (L,C), (N,M), (R,P), (R,K), (U,O)

Those pairs of training samples are then picked up for training a set of linear SVMs to

Table 6: Letter error rate under different interpolation strategies

Interpolation / Dataset	Baseline		Linear		Cubic spline		B-spline	
	stroke	whole	stroke	whole	stroke	whole	stroke	whole
Dataset *I*	12.73%	14.86%	10.79%	10.18%	11.93%	10.76%	10.65%	**9.00%**
Dataset *II*	13.16%	6.76%	11.59%	5.59%	12.06%	5.82%	10.13%	**4.25%**

do the confusion pair decision with 64-points Fourier descriptor on the trajectory magnitude of both horizontal and vertical axes, where real and imaginary parts are all used. When one has 2-best recognition results in these confusion pairs, he can apply these models. Doing so can further improve the stroke based system by reducing the error by 10%, relatively. But when only the confusion pairs are considered, one can improve the recognition rate on these systems by 40%, relatively.

4.5.5 Interpolation

This study conducted a series of experiments to understand the effect of different interpolation techniques, such as linear, cubic spline, and the third order B-spline interpolations, and the role of the number of added sample points of each gesture under this scenario.

As for the HMMs, 8 states and 2 mixtures per state are chosen because the configuration with large number of states may leave too few data points in each state. This study adopted this configuration and found the baseline system to be slightly worse than the configuration mentioned in the previous study [79] with the 16-state HMMs and 1 mixture per state on test set 1. However, even starting from this configuration, one can outperform the previously reported HMM-based system with the i-vector and SVM that will be discussed in the next chapter.

4.5.5.1 Effect on number of points

This study first did a simple experiment on a subset of Dataset *I* with 4 letters, each with more than 33 frames. For each letter, around 20 samples were used for both training and testing with 8-state HMMs and 2 mixtures per state. To examine the effect of different sampling rates under a comparable condition, random removal of 5% to 75% points with a step

size of 5% 10 times was done. It is a reasonable experiment to look into the performance of the low sampling rate because the hand movement is usually with varying velocity and acceleration. As shown in Fig. 21, as the data strings were extremely under-sampled, one shall not have good discriminative power. For example, when 50% of the samples were removed, most gestures having only 16 frames, the letter error rate would increase from 3.42% to 11.78%, which is 244% worse than that of the original one.

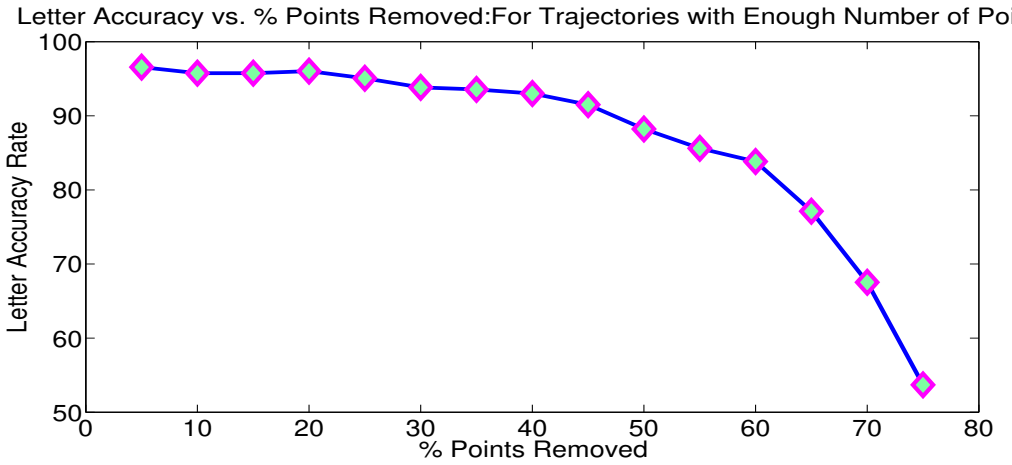


Figure 21: Effect of losing track on data samples, averaged over 10 repetitions. It is obvious when data are under-sampled, the discriminative power will decrease accordingly.

Similarly, this study checked the relationship between upper bound of the number of gesture samples and the letter accuracy, as shown in Fig. 22. When 5 points were added between each two original points to model a 40-state HMM, the performance almost always improved. Note that for those cases whose original data points are less than 11, the original baseline performed much worse than the interpolated version. The test clearly shows the positive effectiveness of feature interpolation.

4.5.5.2 Effect on Interpolation Configuration

Next, the effects of applying different numbers of linearly interpolated points as well as HMM configurations under different numbers of states and per-state mixtures are checked now. An averaged performance curve at 3 mixtures was shown in the upper part of Figure 23. As one can see, the performance would begin to saturate when the number of states

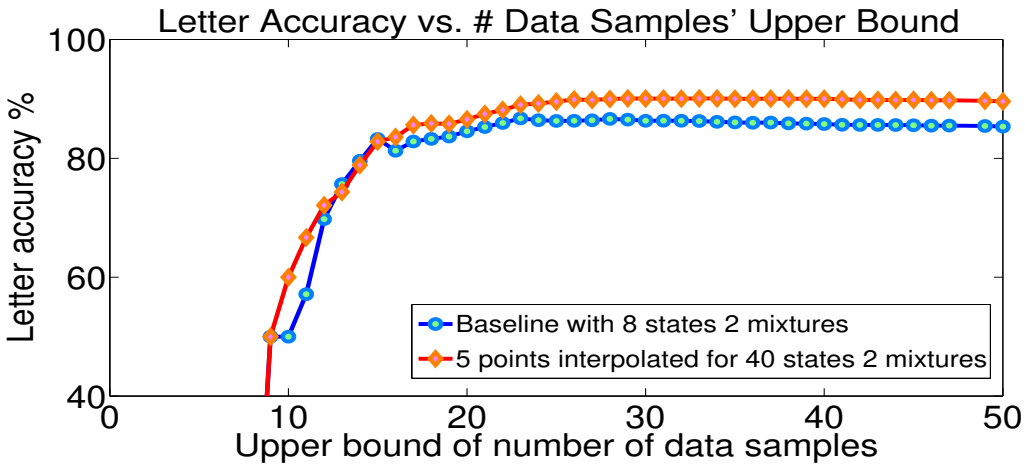


Figure 22: Letter recognition accuracy when the number of samples are up to a specific number, it is clear that when more samples are available for each gesture, we will have higher accuracy.

exceeded 30-40, which matched well with the mathematical model of strokes discussed earlier in this chapter.

Moreover, as shown in the lower part of Figure 23, when more than 5 points are added, the letter accuracy is going to drop in general. Note that the curve starts at 5 points because it is the minimum requirement to properly train whole-letter HMMs with 40 states.

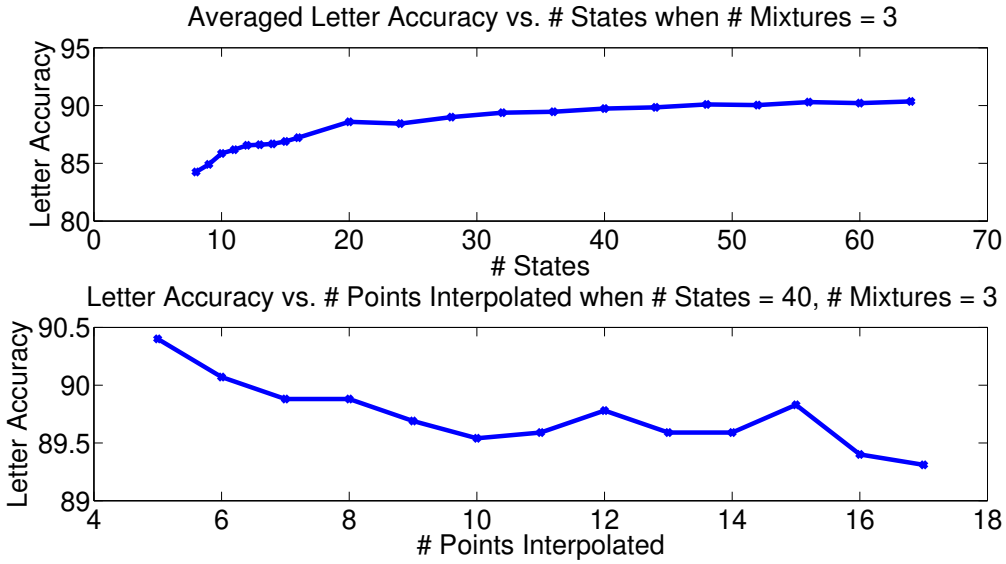


Figure 23: Upper part: Letter accuracy vs. number of states, averaged over number of points interpolated. Lower part: Letter accuracy vs. number of interpolated points, while mixture number is 3.

4.5.5.3 Different ways of interpolation

As noted in the above observations and discussions, the baseline system for whole-letter modeling was constructed with 8-state 3-mixture while 2-state 32-mixture is chosen for stroke modeling. After 5 points are interpolated with different interpolation algorithms, 40-state 3-mixture HMM and 6-state 2-mixture HMM configurations are chosen for whole-letter and stroke based modeling, respectively. The following interpolation techniques are discussed:

1. Linear interpolation.
2. Cubic spline interpolation.
3. Cubic B-spline interpolation.

The results of using these different interpolation techniques are shown in Table 6 for both whole-letter and stroke-based modeling. The stroke models (data columns 1, 3, 5, 7) are comparable to whole-letter models (data columns 2, 4, 6, 8) in set I, as shown in the first row for all interpolation techniques. This merit is not seen for set II in the second row because too large stroke spatial variations and too many possible stroke sequences made it difficult to align the strokes correctly due to low frame rate. For the whole-letter modeling, it did not suffer from such segmentation issues. Comparing results from different interpolation techniques reveal that the relative error reductions (columns 2 and 8) for set I and II with B-spline are 39.43% and 37.13%, respectively, while linear interpolation gives 31.49% and 17.31% (columns 2 and 4) and cubic spline gives 27.59% and 13.91% (columns 2 and 6) relative error reductions. From these results, it is found that cubic B-Spline is consistently the most effective among these interpolation strategies because it is not strictly required to pass all the original sample points and can give reasonable transitions between the original points and the smoothed points simultaneously. Notice that it outperforms lower order B-splines, which are not shown here. The cubic spline, however, is required to pass all the

original points and may cause the unnatural third order curves to be interpolated and thus the performance is even poorer than that of the linear interpolation.

4.5.5.4 Stroke Modeling with Interpolation

As discussed previously in Table 6, the stroke modeling strategy can still gain some advantages when the interpolation was applied. But the overall performance is still inferior to the whole-letter modeling strategy.

Now, when two-pass decoding strategy that removed the moving-in and moving-out strokes was applied, for the Dataset I, the error could be further reduced by 7.50% relatively. When the same geometrical indices were added, the relative error reduction over the baseline was 29.77 %, which gave a 8.94% error rate, which is now slightly better than the whole-letter modeling result.

4.6 Summary

In this chapter, this study dealt with challenges encountered under in-car environments for English letter hand gesture recognition. Using the hand detection strategy discussed in Chapter 3, the first challenge of varying lighting condition is alleviated by integrating background-foreground information and adaptive skin color modeling. For the challenge of proper gesture modeling, this work investigated the stroke modeling and designed a two-pass decoding algorithm and showed its effectiveness in removing the meaningless starting and ending strokes that are constantly seen for in-car hand gestures. Next, the geometric properties were integrated to further reduce the error rate. Finally, by further fusing this model with the original baseline score, a relative 66.04% error reduction compared to original HMM baseline is achieved. This performance definitely falls in the expected level of 10% error rate. Moreover, the performance gap still exists as compared to the 2-best scenario. Therefore, geometric properties among strokes with better discriminative power than the present jobs and other stroke properties should be further studied, especially for easily confused letter pairs.

Next, conic section was chosen as an approximation for the general strokes for hand gesture recognition of English letters. By this convention, this study inferred the number of states for appropriately modeling both stroke and whole-letter HMMs. The main reason why the low frame rate issue can be overcome by feature interpolation is that it enables a reasonable usage of under-sampled data along with good data. It was also found that stroke based models might suffer from false alignments under low frame rate conditions. Among several strategies, cubic B-spline interpolation with whole-letter models was shown to be the most promising one and achieved up to a 39% relative error reduction from the baseline system without feature interpolation. When geometrical information and two-pass decoding are applied to the stroke modeling strategies with interpolation, the performance can be slightly better than the whole-letter strategies, but are comparable.

In summary, as will be shown in Figure 24 in Chapter 6, for Dataset I, applying stroke model and removing meaningless strokes can achieve 9.91% error rate, with fusing whole letter model, it can achieve 9.17% error rate. Note this result is slightly worse than that of the whole-letter modeling with cubic B-spline interpolation (9.00%). When interpolation is applied on the stroke model, it can achieve even better performance at 8.94% error rate, which is roughly identical to the case of using Fourier descriptor on easily confused letters. These promising results showed that the robust to model representation and low frame rate was successfully achieved by using stroke model and cubic B-spline interpolation. However, the feature used in these strategies were still basically raw features without considering local statistical or structural information. In next chapter, several enhanced ways of feature extraction will be introduced to enhance robustness to feature representation.

CHAPTER 5

FEATURE AND MODEL ENHANCEMENT

In this chapter, by making reference to sets of clustered gesture segments, we developed a statistical feature representation that is capable of describing the detailed local gesture structure such that the discriminative power as well as the robustness can be enhanced for gestures of finger-written English letters . It is important to explore these kinds of features because most conventional gestural features rely only on the raw trajectory information, such as position and velocity sequences [38, 9, 79], and may not be as strong as speech feature vectors in terms of their discriminative power. Also, as mentioned previously, one should explore features that are robust to the low frame rate issues and user variations.

Conventional features for gesture recognition can be roughly categorized into two types. The first one contains the previously mentioned raw features of a trajectory. The second one includes the global transformation or statistics of the whole trajectory, such as Fourier descriptors (FD) [19, 101], histogram of trajectory orientations [102], or velocity profiles [45, 103]. Although some successful results are reported, these features seldom consider local information such as sub-unit relationships or constraints.

As mentioned previously, the existing feature vectors for gesture applications are mainly trajectory information such as position and velocity. In practical applications, these types of features are usually too simple and not robust enough to detect noise and user variations. In order to enhance robustness of these feature vectors, how to utilize the statistical features for practical issues should be discussed.

Moreover, since the recording scenarios are quite noisy, it is possible that part of the detected gesture trajectories are corrupted with noise sources as well. Unfortunately, it is in general trivial to recover it because it can come as a shot noise instead of additive noise that can be removed by applying subtraction techniques. These phenomena will make the resulting HMM problematic because they may lead into a wrong path in the decoding

trellis. In the speech recognition community, it is important to note that speech recognition systems usually include multiple samples with Hamming window to include short-time statistics to feed into the HMM. Also, doing splicing [104] (concatenating feature vectors from more than one frames) can achieve better robustness to potential short variations.

Also, in many computer vision researches, it is generally believed that multiple-feature fusion is a tool to enhance the description capability. In [105], a set of Haar-like filters was used together with multiple hand descriptors for the posture recognition task. Meanwhile, another similar idea was proposed in the area of information retrieval where the similarity measure of a sample with multiple training data was used to build the feature vector without using supervised information [106]. In other words, when multiple modalities are not available, using different ways to learn from the same data could also be an option.

Motivated by these ideas, the local statistics will be used in this work. Just as those discussed in previous chapter, which recommended B-spline due to its shape-preserving capability to reduce the effect of low sample rate and outlier data points, an alternative method for this will be using local statistical features. To use local statistics, in [102], velocity histograms of 8 directions were used to model the hand-written gestures. Instead of using the HMM as their modeling tool, they just used these histogram features to do the k-nearest neighbor search and claim to reach a good performance. This study reproduces this procedure in the examined system, and found it is comparable with the HMM system in a preliminary experiment. This observation implies that, as long as the statistical features are sufficient to cover the space, one need not model the gesture by HMM-like dynamic modeling tools.

However, this template technique can degrade the system performance during actual recording scenarios, since they did not directly model local structural similarity but just do search over examples. To go beyond the idea of doing joint comparisons on all samples, an unsupervised clustering can be used to categorize the motion templates. Therefore, it is believed that when enough training templates on multiple "examples" are given to cover

more possible parts of gesture variations, the feature robustness can be enhanced.

This idea can be done by comparing the data segment statistics with a finite set of templates or clusters that can represent the recorded gesture components. This is the motivation of the proposed method in this chapter. Traditional statistical modeling methods mostly rely on mixtures of Gaussian densities or Gaussian mixture models (GMM) to model all possible variations. The similarity to each "template" or "part of gesture" is only given by the joint likelihood computed from all "template" clusters. These variations can be considered jointly in a "super-vector" space. But this space is generally sparse and may need further post-processing so that the corresponding gesture variations can be well summarized. To carry this idea, a factor analysis method also known as "i-vector" is considered. This is induced by the factor analysis on the super-vector with respect to the universal background model (UBM) or sets of unsupervisedly clustered templates.

In the following paragraphs, the i-vector scheme and proposed modification for this research problem will be presented. Next, some further modification schemes are discussed. The frame rate issue is also an issue for i-vector, but can be resolved with the interpolation scheme mentioned in Chapter 4. Finally, experiments will be demonstrated followed by a short summary.

5.1 Enhancing Robustness with I-Vector

Now, the theoretical discussion and related works on i-vector technique are reviewed. As mentioned above, it is generally believed [107, 108, 109] that to jointly consider the feature space coverage is helpful in understanding the true factors governing the distinction among samples. This type of analysis was discussed in depth in the speaker verification community; they are usually referred to as joint factor analysis (JFA) [108, 109]. Basically, the logic is to measure the difference between the speaker feature statistics with an already existing universal background model (UBM) that covers the feature space with several clusters represented by a Gaussian mixture model (GMM). And then according to the

supervised information, the channel, speaker dependent factors, and other sources can be analyzed. Finally, those "factors" can be used as feature vectors for different purposes, as formulated below in Eq. (121).

To be more specific, assume that if the whole feature space can be roughly described by a Gaussian mixture model (GMM) with C mixture components, a super-vector of these C mixtures can be formed by concatenating mean vectors of the individual Gaussian densities according to a set of predefined mixture indices. This GMM is called UBM for the feature space [110]. The corresponding super-vector formed by concatenating the mean vectors of all mixtures is denoted by $\boldsymbol{m}$. The JFA is formulated as follows.

$$\boldsymbol{\mu} = \boldsymbol{m} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{D}\boldsymbol{z} + \boldsymbol{\epsilon}, \tag{121}$$

where $\boldsymbol{m}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ cover the speaker independent term, speaker dependent term, and environmental or channel term, respectively, and ϵ is a Gaussian noise term with zero mean and a diagonal covariance matrix. However, unlike the speaker verification problem, this work does not care who is doing the gesture, so the form will be reduced to only consider the whole factors all together. Note that $\boldsymbol{m}$ is considered to be speaker-independent in speaker tasks. Under the present gesture recognition scenarios, the identity subject will be the gesture, so that $\boldsymbol{m}$ is a gesture independent component. As long as one can find the projection matrix $\boldsymbol{V}$, all the discriminative power will be expressed by the vector y.

$$\boldsymbol{\mu} = \boldsymbol{m} + \boldsymbol{V}\boldsymbol{y} + \epsilon, \tag{122}$$

This is exactly the i-vector technique [109, 111, 112, 113] where all factors are considered together as a single "total variability factor" and then the linear analysis will make the resulting vector similar to the mathematical form of principal component analysis (PCA).

In this study, the i-vector is treated as an unsupervised learning technique. For the original i-vector scheme, supervised information for each speaker must be given so that one can extract i-vector for them. Here, again, it is not interesting to know the writer of

the gesture, but the motivation similar to the factor analysis mentioned above can be re-interpreted under the i-vector scheme as well. Since the speaker verification task did not care what is spoken, the temporal information is dropped during the super-vector extraction process. This assumption, however, is NOT true for the gesture modeling applications.

To be more specific, the number of speakers is frequently on the order of several hundreds which should be clearly identified via the JFA system of Eq.(121) . Conversely, in the present gesture issue, it has only 26 English letters which can be handled by the following simplification with the aid of the original HMM framework.

The i-vector and its similar techniques have been studied for about a decade in the speaker community [114, 111, 109]. Most of its applications are applied with GMM-based speaker applications so that a concise derivation for HMM was seldom given clearly. Therefore, this study considers the approach using a cosine explanation. In addition, due to its model similarity compared to the PCA and probabilistic PCA (PPCA), this study discusses how i-vector can be extracted under the EM algorithm framework by comparing it to PCA and PPCA, details can be found in Appendix.

Both PCA and PPCA used the following model with $\boldsymbol{m}$ being the mean of the observed vectors of size T, $\boldsymbol{W}$ being the unknown projection matrix to be determined, $\boldsymbol{x}_t$ being the low dimensional representation of data vector $\boldsymbol{o}_t$, and $\boldsymbol{\epsilon}$ being an isotropic Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$:

$$\boldsymbol{o}_t = \boldsymbol{m} + \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{\epsilon}. \tag{123}$$

PCA is to find a orthonormal matrix $\boldsymbol{W}$ that can maximize the covariance in the reduced space of $\boldsymbol{x}_t$, and the PPCA explicitly assumes the distribution of the reduced space representation as $\boldsymbol{x}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, and the following auxiliary function can be obtained with Bayes theorem. The i-vector scheme, however, has a different points of view. First of all, since the cluster membership is unknown, the vector can only be observed in an expected sense. Second, it has more general assumption on the noise term, which is diagonal but

non-isotropic with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. Considering these facts, the i-vector extraction under EM framework can be summarized as follows:

$$Q(\mathbf{\Lambda}; \overline{\mathbf{\Lambda}}) = \Bigg[\sum_{\ell=1}^{L} \sum_{s=1}^{N} \sum_{m=1}^{C} \sum_{t=1}^{T_\ell} \overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m) \mathcal{E}_{\boldsymbol{y}|\boldsymbol{o},\overline{\mathbf{\Lambda}}} \Big\{ log\Big(\frac{1}{(2\pi)^{\frac{D}{2}} det(\mathbf{\Sigma}_{s,m})^{\frac{1}{2}}} \Big) - \frac{1}{2} (\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m} - \boldsymbol{V}_{s,m} \boldsymbol{y}_\ell)^T \mathbf{\Sigma}_{s,m}^{-1} (\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m} - \boldsymbol{V}_{s,m} \boldsymbol{y}_\ell) \Big\} \Bigg] \tag{124}$$

where $\boldsymbol{V}_{s,m}$ is a sub-matrix of $\boldsymbol{V}$ corresponding to the state s and mixture m. Note it has D rows and the same number of columns as $\boldsymbol{V}$. Also, the membership probability $\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)$ has the same definition as that in HMM and the subscript $\boldsymbol{o}_t^\ell$ indicates the observed vector at time t for ℓ-th sample of the L-sample training data. The resulting i-vector $\langle \boldsymbol{y}_\ell \rangle$ and projection matrix component $\boldsymbol{V}_{s,m}$ can be estimated under the EM framework:

$$\langle \boldsymbol{y}_\ell \rangle = \Big[\boldsymbol{I} + \sum_{s,m} N_{s,m}(\ell) \boldsymbol{V}_{s,m}^T \mathbf{\Sigma}_{s,m}^{-1} \boldsymbol{V}_{s,m} \Big]^{-1} \Big[\sum_{s,m} \boldsymbol{V}_{s,m}^T \mathbf{\Sigma}_{s,m}^{-1} \boldsymbol{F}_{s,m}(\ell) \Big], \tag{125}$$

$$\langle \boldsymbol{y}_\ell \boldsymbol{y}_\ell^T \rangle = \boldsymbol{I} + \sum_{s=1}^{N} \sum_{m=1}^{C} \sum_{t=1}^{T_\ell} \overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m) \boldsymbol{V}_{s,m}^T \mathbf{\Sigma}_{s,m}^{-1} \boldsymbol{V}_{s,m} + \langle \boldsymbol{y}_\ell \rangle \langle \boldsymbol{y}_\ell \rangle^T, \tag{126}$$

where

$$N_{s,m}(\ell) = \sum_{t=1}^{T_\ell} \overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m), \tag{127}$$

$$\boldsymbol{F}_{s,m}(\ell) = \sum_{t=1}^{T_\ell} \overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m) (\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m}), \tag{128}$$

are also known as zero-th and first order Baum-Welch statistics, respectively. Then, one has

$$\begin{aligned} \boldsymbol{V}_{s,m} &= \Big[\sum_{\ell=1}^{L} \sum_{t=1}^{T_\ell} \overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m) (\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m}) \langle \boldsymbol{y}_\ell \rangle^T \Big] \Big[\sum_{\ell=1}^{L} \sum_{t=1}^{T} \overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m) (\langle \boldsymbol{y}_\ell \boldsymbol{y}_\ell \rangle^T) \Big]^{-1}, \\ &= \Big[\sum_{\ell=1}^{L} \boldsymbol{F}_{s,m}(\ell) \langle \boldsymbol{y}_\ell \rangle^T \Big] \Big[\sum_{\ell=1}^{L} N_{s,m}(\ell) (\langle \boldsymbol{y}_\ell \boldsymbol{y}_\ell \rangle^T) \Big]^{-1}. \end{aligned} \tag{129}$$

Therefore, under the EM framework, Eq.(125) and (129) are used iteratively to refine the projection matrix $\boldsymbol{V}$. Note the original HMM parameters serve as information of the

universe and only $\boldsymbol{y}$ and $\boldsymbol{V}$ are iteratively updated, so one can actually drop the bar above the membership function from above equations. After the EM algorithm converges (typically 10 iterations are chosen) , Eq.(125) is used to compute the i-vector $\boldsymbol{y}_\ell$.

5.2 I-Vector for Gesture Recognition

Despite its effectiveness reported in the speaker verification task, directly applying this GMM-based technique for all observed trajectories will not keep the structural information of the underlying gestures. The corresponding negative impact will be shown in the experimental results. In this section, how to modify the i-vector scheme to let it fit into our gesture recognition problem will be discussed.

5.2.1 One I-Vector Per State Approximation

To resolve the discrepancy between HMM and i-vector, the direct solution is inspired by an original work of i-vector [114] and other similar strategies [115, 116]. As shown in Eq. (170), the aforementioned scheme will need to consider all possible state alignments and mixture memberships. Instead of doing this complicated computation, considering only the best path found by Viterbi algorithm is an efficient approach. Therefore, this study only considers data samples aligned to the specific state instead of computing Baum-Welch statistics over the whole data points ($1 \sim T_\ell$).

That is, one can first use standard HMM recognition to get the state sequence given by the model with the maximal log likelihood score. Subsequently, one can apply i-vector extraction on segment aligned to each state, and eventually concatenate i-vectors generated by these states to form a "super-vector" of the i-vectors aligned to these states. It is designated as "one i-vector per state" strategy in the following paragraphs. In other word, this study treats data samples aligned to different states as if they are located in different universes. After the computations of the sub-spaces of i-vectors are performed, one simply concatenates them together, if all gestures share the same number of states. So if each

HMM has 8 states, for instance, 8 i-vectors will be extracted and concatenated. The approximation made here indicates that this work simply uses the most likely state sequence to approximate the i-vector computation for the HMM. Therefore, this strategy is not only a reasonable approximation but also preserves the local statistical information.

An induced problem is how to select the UBM for each state. One of the most intuitive ways is to pool all mixtures from the same state of different models and assume these mixtures are the only candidates to be chosen within this specific state. Or one can make a tied-mixture system and apply this tied-mixture pool as UBM mixtures. In this work, the former one was adopted, because it can be computed more efficiently than that of the later one and the results are still quite reasonable.

Besides doing HMM state alignment and extract i-vectors corresponding to these states, an alternative solution is to use a sliding window to extract i-vectors so that it has a set of i-vectors for each gesture. This method is more computationally expensive and unfortunately cannot provide extra gain, so it will not be discussed here. Therefore, the "one i-vector per state" strategy will be used as the primary feature to summarize the local statistics into a single vector to be fed into commonly used classifiers such as SVM classifiers.

Moreover, note that in Eq. (125), the column vectors of the projection matrix $\boldsymbol{V}_c$ are not guaranteed to form an orthonormal basis. As will be discussed in its experimental result, it is quite crucial to orthonormalize these vectors. It is believed this is because when these column vectors are not mutually independent, they may not be able to achieve the best representation of the original posterior variation in a low dimensional space. Moreover, since classifiers based on Euclidean distance are sensitive to the data scales, the normalization process is also believed to be able to make the resulting i-vector more robust to the estimation noise than the original one. For practical data, it is found the $\boldsymbol{V}_c$ matrix can have small norms for some of its column vectors and the inner product terms for these column vectors with other column vectors can sometimes be at scales larger than their norms. This makes it sensitive to the basis dependency, so to ensure that the projection matrix $\boldsymbol{V}_c$ has

orthonormal column vectors can be useful. This study used the Gram-Schmidt process to achieve the orthonormalization condition.

All in all, to apply the i-vector technique under the framework of our system, there are two major issues: (i) how to fit the i-vector technique into the local information described by the finite number of states in AGR, and (ii) how to overcome data sparseness caused by the low frame rate. The first issue can be resolved by the "one i-vector per state" approximation mentioned above via a Viterbi decoded sequence. The second issue, however, can be done via a proper interpolation scheme as proposed in Chapter 4. With these aforementioned strategies, once the UBM and the projection matrix $\boldsymbol{V}$ are ready for each state, one can compute i-vector for each testing segment with Eq. (125). In this study, ALIZE toolkit [117] or Kaldi [118] was used. Now, the conceptual differences between i-vector and HMM as well as the orthogonality issues are discussed in the following paragraphs.

5.2.1.1 Comparison with HMM

A careful inspection of the original HMM and i-vector algorithm reveals that they are ultimately different from each other. The HMM algorithm is principally a method to separately estimate the parameters of the underlying HMM structure for each letter. Because of the input uncertainty and noise of our data set, many writing data of a letter may be incorrectly recognized as another letter which cannot be explicitly remedied by the HMM algorithm. Noticed that, before the i-vector algorithm is applied, most of the probability parameters have been estimated by the Baum-Welch algorithm. Conceptually, the i-vector algorithm tries to simultaneously enhance these probability parameters of all the letters via suitable transformations. They systematically and implicitly manipulate the single matrix system of Eq.(122) under the super-vector structure. Thus the EM algorithm can easily handle the competition among letter mixtures since the required distances of decision of the final classification for all the letters are simultaneously evaluated.

To check the differences between HMM and i-vector technique carefully, one can start from checking their auxiliary functions. Now compare the i-vector auxiliary function of

the (Eq. (170)) with that for HMM as shown below.

$$Q(\mathbf{\Lambda}, \overline{\mathbf{\Lambda}}) = \sum_{\ell=1}^{L}\sum_{i=1}^{N}\overline{\gamma}_{\boldsymbol{o}_1^\ell}(i)log\pi_i + \sum_{\ell=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T_\ell-1}\overline{\xi}_{\boldsymbol{o}_t^\ell}(i,j)loga_{i,j}$$
$$+ \sum_{\ell=1}^{L}\sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T_\ell}\left[\overline{\gamma}_{\boldsymbol{o}_1^\ell}(s,m)log\omega_{s,m}\right] + \sum_{\ell=1}^{L}\sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T_\ell}\left\{\overline{\gamma}_{\boldsymbol{o}_1^\ell}(s,m)\left[log\left(\frac{1}{(2\pi)^{\frac{D}{2}}det(\mathbf{\Sigma}_{s,m})^{\frac{1}{2}}}\right)\right.\right.$$
$$\left.\left. - \frac{1}{2}(\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m})^T\mathbf{\Sigma}_{s,m}^{-1}(\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m})\right]\right\} \tag{130}$$

In each EM optimization step, for a Gaussian mixture state related to an observed element $\boldsymbol{o}_t^\ell$ (for a specific English letter), all the other states of the whole data set are frozen to simplify the problem. The HMM auxiliary function did not consider the likelihood of the current data with other gestures. Therefore, the iteration is inefficient and may be trapped by any undesired local maximum. Conversely, in Eq.(170), all the Gaussian mixtures for every $\boldsymbol{o}_t^\ell$ in the whole data set are simultaneously optimized. Comparing with that of HHM, the resulting $Q(\mathbf{\Lambda}, \overline{\mathbf{\Lambda}})$ is certainly more reliable, for it uses larger amount of data. The previous tests show that, no matter whether the corresponding $\boldsymbol{V}_{s,m}$ is orthogonalized or not, the efficiency of the EM algorithm applying the i-vector version in a classification manner is superior to that without applying the super-vector in a dynamic recognition manner.

In fact, the main goal of the i-vector or factor analysis is trying to measure the posterior differences after observing the data compared to each mixture. So "how far the observed data are from each estimated mixture" is the main target to be modeled while HMM only takes care of which model is the nearest one in the feature space. In fact, the classification performance highly depends on distances among different classes in the feature space. Unfortunately, the HMM is not actively targeted to achieve this purpose. The reason is that the HMM auxiliary function in Eq.(130) does not explicitly consider how to distinguish a specific letter from other letters. Although the i-vector does not target this aim, either, it uses the super-vector to actually connect all the observed vectors and the HMM mixtures via its infrastructure. Since the auxiliary function of Eq.(170) has the function of emphasizing

most of the critical terms, those observed elements corresponding to the confused competitions can be properly identified. Eventually, the performance of the final classification is better than that solely applying the HMM algorithm.

These discussions clearly lead to the following statement: " The proposed super-vector formulation of the i-vector strategy for the EM optimization has a better performance than that purely using HMM."

From these observations, if one replaces the optimization target, i.e. the auxiliary function with a objective function directly related to the performance indices, which is commonly used in discriminative training (DT) strategies, he will be able to find a discriminative projection and thus make discriminative feature extraction instead of doing it in an unsupervised manner. Currently, however, this cannot be done directly because of the unsupervised nature of the i-vector extraction, as will be discussed soon in the next sub-section.

5.2.2 Stiefel Manifold Optimization for I-vector

Note that if one would like to enforce the V matrix to be orthogonal, a Lagrange multiplier $-\frac{1}{2}tr(\boldsymbol{\eta}\boldsymbol{V}^T\boldsymbol{V}-\boldsymbol{I})$ term can be added to Eq.(170), where $\boldsymbol{\eta}$ should be a symmetric matrix. Obviously the i-vector estimation in Eq.(125) will stay the same, but $\boldsymbol{V}_{s,m}$ shall be different. Taking the derivative with respect to $\boldsymbol{V}_{s,m}$ and set result to zero yields

$$\sum_{\ell=1}^{L}\sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)\boldsymbol{\Sigma}_{s,m}^{-1}(\boldsymbol{o}_t^\ell-\boldsymbol{\mu}_{s,m}-\boldsymbol{V}_{s,m}\boldsymbol{y}_\ell)\boldsymbol{y}_\ell^T-\boldsymbol{\eta}\boldsymbol{V}_{s,m}=\boldsymbol{0}, \tag{131}$$

Unfortunately, it is not easy to obtain a suitable Lagrange multiplier by applying the constraint $\boldsymbol{V}^T\boldsymbol{V}=\boldsymbol{I}$. This constraint is called the Steifel manifold. To do the optimization on this manifold, a feasible solution was proposed recently by Wen et al. [119]. With the help of matrix inversion lemma and Caley's transformation, a gradient descent algorithm will be able to get a reasonable estimate of $\boldsymbol{V}_{s,m}$ properly. This is a good strategy because the gradient descent method will convert the EM algorithm to be a generalized EM (GEM) algorithm that can iteratively optimize the target auxiliary function when a closed form of M-step is unavailable.

As mentioned previously, the i-vector scheme seems to be able to gain some advantages after performing the column orthonormalization. This observation motivates the research of the orthogonal constrained optimization.

Consider the M-step of the i-vector parameter estimation on GMM for simplicity. For GMM based i-vector scheme, one can simply drop the state index s in the previous derivations. Define the following matrices:

$$\boldsymbol{\Phi}_m = \sum_{\ell=1}^{L} \boldsymbol{F}_m(\ell)\boldsymbol{y}_\ell^T, m = 1, 2, \ldots, M \tag{132}$$

$$\boldsymbol{R}_m = \sum_{\ell=1}^{L} N_m(\ell)\boldsymbol{y}_\ell\boldsymbol{y}_\ell^T, m = 1, 2, \ldots, M \tag{133}$$

$$\boldsymbol{\Phi} = [\boldsymbol{\Phi}_1\boldsymbol{\Phi}_2 \ldots \boldsymbol{\Phi}_M]^T, \tag{134}$$

Then he can write the M-step as the following form:

$$Q(\boldsymbol{V}) = \sum_{m=1}^{M} \left\{ \text{tr}[\boldsymbol{V}^T\boldsymbol{S}_m^{-1}\boldsymbol{\Phi}] - \frac{1}{2}\text{tr}[\boldsymbol{S}_m^{-1}\boldsymbol{V}\boldsymbol{R}_m\boldsymbol{V}^T] \right\}, \tag{135}$$

where $\boldsymbol{S}_m$ denotes the block diagonal matrix of size $CD \times CD$ with only the m-th diagonal block element $\boldsymbol{\Sigma}_m$.

Optimizing this auxiliary function under the orthonormal $\boldsymbol{V}$ constraint is equivalent to optimizing the following problem:

$$\min_{\boldsymbol{V}} F(\boldsymbol{V}) = \sum_{m=1}^{M} \left\{ \frac{1}{2}\text{tr}[\boldsymbol{S}_m^{-1}\boldsymbol{V}\boldsymbol{R}_m\boldsymbol{V}^T] - \text{tr}[\boldsymbol{V}^T\boldsymbol{S}_m^{-1}\boldsymbol{\Phi}] \right\} \text{such that} \boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}. \tag{136}$$

This is the previously mentioned Stiefel manifold optimization [119]. According to the previous work, this optimization can be done in a straight-forward way. The basic idea behind this optimization problem is to search on a line on the tangent plane of the current estimation along a line where the orthogonality condition can be satisfied.

The line to be searched can be expressed in the following form:

$$\boldsymbol{Y}(\tau) = \boldsymbol{V} - \tau\boldsymbol{A}\frac{\boldsymbol{V} + \boldsymbol{Y}(\tau)}{2}, \tag{137}$$

After simple manipulations, one has:

$$\boldsymbol{Y}(\tau) = \left(\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{A}\right)^{-1}\left(\boldsymbol{I} - \frac{\tau}{2}\boldsymbol{A}\right)\boldsymbol{V}. \tag{138}$$

This is in the form of a Cayley transform, which will guarantee $\boldsymbol{Y}(\tau)$ to be orthonormal if $\boldsymbol{V}$ is orthonormal and $\boldsymbol{A}$ is an anti-symmetric matrix.

To verify this statement, without loss of generality, consider the following definition with $\boldsymbol{A}^T = -\boldsymbol{A}$:

$$\boldsymbol{B} = (\boldsymbol{I} + \boldsymbol{A})^{-1}(\boldsymbol{I} - \boldsymbol{A}), \tag{139}$$

then

$$\begin{aligned}\boldsymbol{B}^T =&(\boldsymbol{I} - \boldsymbol{A})^T[(\boldsymbol{I} + \boldsymbol{A})^{-1}]^T\\ =&(\boldsymbol{I} + \boldsymbol{A})[(\boldsymbol{I} + \boldsymbol{A})^T]^{-1}\\ =&(\boldsymbol{I} + \boldsymbol{A})(\boldsymbol{I} - \boldsymbol{A})^{-1}.\end{aligned} \tag{140}$$

Now, consider $\boldsymbol{B}^T\boldsymbol{B}$

$$\begin{aligned}\boldsymbol{B}^T\boldsymbol{B} =&(\boldsymbol{I} + \boldsymbol{A})(\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{I} + \boldsymbol{A})^{-1}(\boldsymbol{I} - \boldsymbol{A})\\ =&(\boldsymbol{I} + \boldsymbol{A})(\boldsymbol{I} + \boldsymbol{A})^{-1}(\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{I} - \boldsymbol{A}) = \boldsymbol{I}.\end{aligned} \tag{141}$$

This is because the middle two terms commute:

$$\begin{aligned}(\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{I} + \boldsymbol{A})^{-1} =&[(\boldsymbol{I} + \boldsymbol{A})(\boldsymbol{I} - \boldsymbol{A})]^{-1}\\ =&(\boldsymbol{I} + \boldsymbol{A}\boldsymbol{A})^{-1}\\ =&[(\boldsymbol{I} - \boldsymbol{A})(\boldsymbol{I} + \boldsymbol{A})]^{-1}.\end{aligned} \tag{142}$$

Therefore, the Cayley transform can guarantee the orthogonality constraint of $\boldsymbol{Y}(\tau)$, and searching along the line $\boldsymbol{Y}(\tau)\boldsymbol{V}$ for optimal $\boldsymbol{V}$ will satisfy the orthogonality constraint. For

this curve to represent a line on a tangent plane, one should have

$$\begin{aligned} \boldsymbol{G} &= \nabla_V F(\boldsymbol{V}) \qquad (143) \\ &= \sum_{m=1}^{M} \frac{1}{2}\left[\boldsymbol{S}_i^{-T}\boldsymbol{V}\boldsymbol{R}_m^T + \boldsymbol{S}_i^{-1}\boldsymbol{V}\boldsymbol{R}_m\right] - \boldsymbol{S}_i^{-1}\boldsymbol{\Phi} \\ &= \sum_{m=1}^{M} \boldsymbol{S}_i^{-1}\left(\boldsymbol{V}\boldsymbol{R}_m - \boldsymbol{\Phi}\right), \\ \boldsymbol{A} &= \boldsymbol{G}\boldsymbol{V}^T - \boldsymbol{V}\boldsymbol{G}^T. \qquad (144) \end{aligned}$$

It is quite easy to verify that $\boldsymbol{A}$ is an anti-symmetric matrix. Note the computation of $\boldsymbol{G}$ and $\boldsymbol{A}$ can be done with block diagonal matrix operations:

$$\boldsymbol{G}_m = \sum_{m=1}^{M} \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{V}_m\boldsymbol{R}_m - \boldsymbol{\Phi}_m\right), \qquad (145)$$

$$\boldsymbol{A}_m = \boldsymbol{G}_m\boldsymbol{V}_m^T - \boldsymbol{V}_m\boldsymbol{G}_m^T, \qquad (146)$$

$$\boldsymbol{G} = \left[\boldsymbol{G}_1, \boldsymbol{G}_2, \ldots, \boldsymbol{G}_M\right]^T, \qquad (147)$$

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \ldots & \boldsymbol{0} & \boldsymbol{A}_M \end{bmatrix}. \qquad (148)$$

To find an optimal τ, it is required to evaluate the derivative of the objective function over τ, which can be done as follows. First, compute the derivative of the new estimate $\boldsymbol{Y}(\tau)$ with respect to τ:

$$\begin{aligned} \frac{\partial \boldsymbol{Y}(\tau)}{\partial \tau} &= -\left[\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{A}\right]^{-1}\frac{1}{2}\boldsymbol{A}\left[\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{A}\right]^{-1}\left[\boldsymbol{I} - \frac{\tau}{2}\boldsymbol{A}\right]\boldsymbol{V} - \left[\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{A}\right]^{-1}\frac{1}{2}\boldsymbol{A}\boldsymbol{V} \\ &= -\left[\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{A}\right]^{-1}\boldsymbol{A}\left[\frac{\boldsymbol{V} + \boldsymbol{Y}(\tau)}{2}\right], \qquad (149) \\ \left.\frac{\partial \boldsymbol{Y}(\tau)}{\partial \tau}\right|_{\tau=0} &= -\boldsymbol{A}\boldsymbol{V}. \qquad (150) \end{aligned}$$

This can also be computed with each mixture component because the first two terms in the

last line of the above equations will result in a block diagonal matrix.

$$\begin{aligned}\frac{\partial F(\boldsymbol{Y}(\tau)}{\partial \tau} &= \mathrm{tr}\left\{\boldsymbol{G}^T \frac{\partial \boldsymbol{Y}(\tau)}{\partial \tau}\right\} \\ &= \sum_{m=1}^{M} \mathrm{tr}\left\{\boldsymbol{G}_m^T \frac{\partial \boldsymbol{Y}_m(\tau)}{\partial \tau}\right\},\end{aligned} \tag{151}$$

where the last equality holds due to the block diagonal nature, and the derivative term is:

$$\frac{\partial \boldsymbol{Y}_m(\tau)}{\partial \tau} = -\left[\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{A}_m\right]^{-1} \boldsymbol{A}_m\left[\frac{\boldsymbol{V}_m + \boldsymbol{Y}_m(\tau)}{2}\right], \tag{152}$$

$$\left.\frac{\partial \boldsymbol{Y}_m(\tau))}{\partial \tau}\right|_{\tau=0} = -\boldsymbol{A}_m \boldsymbol{V}_m. \tag{153}$$

In summary, the line search is done as follows, given $0 < \rho_1 < \rho_2 < 1$:

1. Initialize $\tau > 0$

2. do

   $\tau /= 2$;

   while$\Bigg($ not $\Big[$ $F(\boldsymbol{Y}(\tau)) \leq F(\boldsymbol{Y}(\tau))\big|_{\tau=0} + \rho_1 \tau \frac{\partial F(\boldsymbol{Y}(\tau)))}{\partial \tau}\big|_{\tau=0}$ *and*

   $\frac{\partial F(\boldsymbol{Y}(\tau)))}{\partial \tau} \geq \rho_2 \frac{\partial F(\boldsymbol{Y}(\tau)))}{\partial \tau}\big|_{\tau=0}$ $\Big]$ $\Bigg)$;

3. return converged $\boldsymbol{Y}(\tau)$.

The above two conditions for convergence are informally called sufficient decreasing condition and curvature condition. In fact, if the curvature condition cannot be satisfied, one can simply step out of the iteration when the function to be optimized decreases. This will make the M step only an approximation. This procedure is called generalized EM algorithm (GEM).

5.2.3 Discriminative Features for Super-Vector Space

Previous feature has more spatial information but did not include any discriminative optimization steps. That is because the i-vector extraction framework under the EM optimization structure is only to fit the observed data posterior distribution with the affine model

as assumed in Eq. (122) to find the projection matrix $\boldsymbol{V}$ in a maximal likelihood manner. The actual supervised classification information, however, is not involved. This section considers one of the most intuitive discriminative training algorithms, say the minimum classification error (MCE), and discusses the possibility of applying it to improve the feature.

In MCE framework, given observed data $\boldsymbol{o}$ and model parameter set $\boldsymbol{\Lambda}$ for a K-class classification problem, the following logistic sigmoid function with parameter ξ and δ is used to approximate the classification error.

$$\frac{1}{1+\exp\{-\xi[d_k(\boldsymbol{o},\boldsymbol{\Lambda})]\}}, \tag{154}$$

where d_k is a discriminative function for class k, which is usually defined as the minus scores of the difference of the target class k. It is an L_η norm ($\eta > 0$) computed by the scores of other non-k classes, that is

$$d_k(\boldsymbol{o},\boldsymbol{\Lambda}) = -g_k(\boldsymbol{o},\boldsymbol{\Lambda}) + \left[\frac{1}{K-1}\sum_{\forall i\neq k} g_i(\boldsymbol{o},\boldsymbol{\Lambda})^\eta\right]^{\frac{1}{\eta}}, \tag{155}$$

where g_k is the score from classifier, typically takes the form such as likelihood or regressor output. If any negative number causes an issue during the exponential computation, it can be done by using its absolute values in the exponential term and then multiplying its original sign afterward.

For linear classifier, g_k usually takes the form of the output from the model, that is, multiplying the augmented weight matrix with augmented data vector $[1\boldsymbol{o}^T]$. For GMM/HMM type classifiers, g_k is usually taken as the log likelihood from each GMM/HMM given by the corresponding model. However, in the i-vector extraction procedure, there is nothing that can be embedded to express supervised information. The likelihood is only used to describe the soft membership of each mixture component. Therefore, unlike most discriminative training algorithm with the discriminative function, it can be directly applied by using the model log likelihood. In i-vector extraction steps, again, the mixture likelihood has nothing to do with the discriminative power.

From the above discussion, instead of applying the discriminative training on the objective function for the i-vector scheme, it is more reasonable to apply it as a post processing tool. That is just to apply MCE as a tuning tool with linear discriminant analysis as a baseline system to enhance performance. This strategy is shown to be useful but not as effective as using SVM as multiple binary classifications. This is mainly due to the fact that some letter classes are easier to be confused with one another than other pairs of letter classes.

5.2.4 Stroke Modeling with Enhanced Features

To apply enhanced feature such as i-vector on stroke models, the most intuitive way is to mimic the procedure of the whole-letter modeling. As that mentioned in the i-vector extraction for HMM, one can use the Viterbi decoded result as the most likely sequence path and uses it to compute the i-vector state by state. This can also be done in a straightforward way as what had been done in the whole-letter modeling. This strategy will make each gesture a different number of i-vectors. Then, one can directly apply another set of HMMs to do the second pass recognition or use the resulting i-vector to make a decision between 1-best and 2-best results. This chapter will discuss both strategies as well as doing interpolation on these gestures in experimental results.

5.3 Interpolation for I-Vector

As discussed in Chapter 4, the interpolation technique is useful for enhancing the system performance. The i-vector, being a low dimensional representation of super-vector statistics, is also highly dependent on enough number of points to give a reasonable estimation.

Therefore, to take care of the data sparsity, as discussed in previous chapters, the interpolation techniques should be applied. Again, the cubic B-spline interpolation will give the most improvements over other strategies when whole-letter modeling strategy is adopted, as we will see in experimental results.

Table 7: Results for Preliminary Experiments

Feature Type	Dimension	Classifier	Set 1 (1,877) LER(%)	Set 2 (1,717) LER(%)
Raw Feature	6	HMM	14.86	7.16
FD (Global Feature)	64	SVM	26.52	13.80
One i-vector/gesture (Global Feature)	64	SVM	16.41	11.07
	100	SVM	16.46	11.01

5.4 Experiments

With the same setup as all previous experiments, this study did experiments on the same set of recognition tasks. To extract the i-vector, both the ALIZE [117] and KALDI [118] were utilized. The later one is much faster compared to the former one because of the parallelization and membership pruning. Their results, however, are comparable. This section will now discuss the experimental results on both whole-letter and stroke based modeling with i-vector.

5.4.1 Whole-Letter Modeling with I-Vector

The above mentioned tests for a subset of image data showed that several proposed techniques have a positive effect on improving performance. For the sake of illustrating its performance, the i-vector technique will be discussed in this section.

5.4.1.1 Preliminary Experiments

First, a set of preliminary experiments to compare global features with the raw trajectory features were performed. For the global feature, this work computed 64 dimensional FD features and only used a single i-vector to represent each gesture in both the 64 and 100 dimensional subspaces. Although these are fairly small numbers compared to those used in speaker recognition systems [109, 111, 112, 113, 120], it was found that both dimensions can contribute to more than 99% of the data covariance so it was adequate to represent

the potential data variation. These extracted features were then fed into SVM classifiers mentioned above. Compared to the baseline HMM system with raw features, unfortunately, these two global features are inferior to the baseline, as shown in rows 2 to 4 of Table 7, with sizes of two testing data sets given in the brackets shown in the columns 4 and 5 of the header row. This is because the FD features could capture and treat meaningless move-in and move-out strokes with random orientations as real gesturing parts. By removing them, the FD system can be comparable to the baseline. Moreover, although the single i-vector system seems to be better than FD as a global feature system, it is still inferior to the baseline for ignoring local structural information. The weakness of these global features and the importance of the local statistical information were now clearly verified. The remainder of the paper will concentrate on the local features.

5.4.1.2 Experiments with Local Features

In this subsection, the local statistical features with i-vector extracted at each state will be examined. The baseline HMM system was first run to determine the state boundaries of the most likely state sequences. During the training phase, data segments aligned to each state were pooled to train the $\boldsymbol{V}$ matrix for each state and UBM were formed by mixtures belonging to the corresponding state.

This study compared the projected i-vector with the original super-vector, estimated as $\boldsymbol{\mu} - \boldsymbol{m}$, and its simple dimensional reduction version based on PCA. As shown in the rows 2 and 3 of Table 8, the 64 and 100 dimensional i-vector per state setups can respectively give 24.70%-27.19% relative error reductions for test set 1, but increase the error rate by 2.51%-13.13% for test set 2 relatively. It is believed that these degradations were due to the non-uniform norm of the non-orthogonal column vectors of the projection matrix $\boldsymbol{V}$. In addition, the original super-vector in row 4 of Table 8 and the features reduced to 512 and 800 dimensions in rows 5 and 6 are comparable for both test sets and are consistently better than the i-vector results shown in rows 2 and 3. These results not only show that PCA can preserve the discriminative power of the super-vector but verify the importance

Table 8: AGR Results for Two Data Sets with Local Features

Feature Type	Dimension	Classifier	Set 1 (1,877) LER(%)	Set 2 (1,717) LER(%)
Raw Feature	6	HMM	14.86	7.16
One i-vector/state	64×8=512	SVM	11.19	8.10
	100×8=800	SVM	10.82	7.34
Super-Vector	2,496	SVM	8.79	5.24
Super-Vector(PCA)	512	SVM	8.79	5.36
	800	SVM	8.74	5.36
One i-vector/state with orthonormal$\boldsymbol{V}$	64×8=512	SVM	8.63	**4.54**
	100×8=800	SVM	**8.31**	4.83
One i-vector/state with orthonormal$\boldsymbol{V}$ + cubic B-spline	64×8=512	SVM	**7.51**	**4.08**
	100×8=800	SVM	7.83	4.78

of ensuring the column vectors of $\boldsymbol{V}$ to form an orthonormal basis. Therefore, as one can see in the last two rows of Table 8, the i-vector with an orthonormalized matrix $\boldsymbol{V}$, again, consistently outperformed all other systems with a relative error reduction of up to 44.08% and 36.59% respectively for test sets 1 and 2 compared to the baseline. Furthermore, when the interpolation scheme was applied, a further performance boost that gave 49.46 % and 43.02 % relative error reduction over the baseline can be obtained. The effectiveness of the proposed scheme and the advantage of the proper interpolation is thus verified.

5.4.1.3 Orthogonal Constraints

Experiments of applying the orthogonality constraint with the Stiefel manifold optimization and GEM show that the results are just the same or slightly worse than the previously mentioned i-vector with only unconstrained EM algorithm and the $\boldsymbol{V}$ matrix orthonormalized after each iteration (Rows 7 and 8 of Table 8). Indeed the auxiliary function did improve its value over the original one without running the constrained optimization procedure. This is because the EM algorithm used for the i-vector was only to "model" the posterior with the total variability model used by the i-vector scheme. This scheme, unfortunately, is not directly related to the recognition performance. Next, the application of discriminative information to the i-vector scheme will be discussed.

5.4.1.4 Discriminative Training on Classifier

It was found that taking the multiple class LDA on the i-vector will degrade the error rate by relatively 10% (which is around 1% absolutely). Doing MCE on the multiple class LDA will reduce the error by 0.21%, which is around a 2% error reduction. So doing multi-class MCE is not beneficial due to the problem nature, which makes it more important to focus on easily confused pairs instead of working on the joint optimization, as mentioned earlier in Chapter 4.

5.4.2 Stroke Modeling with I-Vector

With the 1-best result as segmentation to extract a series of 64 dimensional i-vectors, and apply 6-state and 4-state HMM on the resulting i-vectors, one can only achieve 70.82% letter accuracy with these feature sets; when first and second order derivatives are taken into account, 75.81% letter accuracy is obtained.

This is likely because of the effect of the small number of samples. For whole-letter modeling scheme, the data are always divided into the same number of segments. But for stroke modeling, when the first pass Viterbi decoding suggested the data to be segmenting into wrong number of segments, the resulting i-vectors may be extracted inappropriately. Therefore, the i-vector technique seems to be not quite appropriate for direct application on stroke modeling. Next, this study will discuss how it can be used to distinguish the 1 and 2 best candidates.

When i-vector is used to support the stroke model likelihood in a log linear combination manner, log likelihood $+\alpha\times$ i-vector SVM score, the performance can be further enhanced compared to the baseline stroke model. It improves the performance by absolutely 0.667%, which is a 4.74% relative error reduction. When interpolation strategy is applied on i-vector, the performance can further reduce the error by 1.95%, which is roughly 18.7% relative error reduction. However, this is not as effective as the i-vector technique of the whole letter modeling strategy.

5.5 Summary

Based on a statistical feature, an i-vector based feature extraction framework for gesture recognition of finger-written English letters was proposed. This study also showed that ensuring the column vectors of the projection matrix $\boldsymbol{V}$ of i-vector extraction to be orthonormal is the key to enhancing the discriminative power of the resulting i-vector over the original super-vector. With this proposed framework, experimental results confirm that the effectiveness of the proposed scheme can achieve up to a relative 36-49% relative reduction in letter error rates for two different data sets. Namely, error rate of the first data set is within 10%, while the second data set is within the expected 5% level.

In contrast to whole-letter modeling, stroke modeling cannot be combined with the i-vector as naturally as the whole-letter one. However, it can still provide some tools for validating the confusion pairs in the second pass refined recognition.

To sum up, working on the i-vector based feature with SVM gives the best performance. With the help of the cubic B-spline interpolation, it can be even more helpful to maintain the system robustness. As summarized in Figure 20 in Chapter 6, this strategy is the better among all proposed methods. The i-vector technique is basically an unsupervised feature extraction process. Discriminative training, however, cannot be applied directly. The post-processing type discriminative training can be helpful, but due to the nature of the problem, the multiple binary SVM classifier is still better.

It is believed that the i-vector features will open up new research opportunities in gesture recognition. Although it is designed to only extend the whole-letter models, its robustness and enhanced discriminative power are clearly verified. Different factors could also be studied under the JFA framework for letter modeling. Lighting conditions is one of the most critical sources of AGR errors. User modeling in AGR is also a subject that fits into the JFA framework similar to the speaker modeling in the speaker verification.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this study, a prototype of a practical in-car gesture recognition system concerning English letter type gestures was developed. The system properly modifies the current system for not-so-well-controlled environments by releasing several constraints. At first, the present work resolved the illumination and showing issues with both skin color modeling and background-foreground similarities to enhance the detection robustness with regular and irregular blocks. Then it tackled the issue of the user variation of the practical system with the stroke modeling, geometrical features, as well as the multi-pass modeling (i.e. removing the starting and ending strokes.) Next, the low frame rate issue was tackled by interpolation schemes. After comparing several interpolation schemes, it was found that the cubic B spline interpolation is the best one for whole letter modeling and stroke modeling. This work further improved the system robustness with the enhanced feature called i-vector. The main advantage of the i-vector is its ability to use the data statistics in the sense of maximum likelihood estimation so that the data lake of frames can gain some information from other gesture segments with similar spatial behaviors. Moreover, the combination of i-vector and interpolation can further improve the system performance. As summarized in Figure 24, this combined method achieved the best performance among all the proposed strategies. So the proposed robust techniques to illumination variation and feature representation were proved to be the most useful for boosting the system performance. To integrate the model representation such as i-vector technique, however, still needs further studies.

As to further works, the first topic is to go to the continuous gesture. As discussed in [10], the segmentation on the gesture sequence is critical for gesture understanding. Moreover, a protocol such as how to write gesture continuously should be defined. For example, some users prefer just do the gesture like normal writing from left to right; others

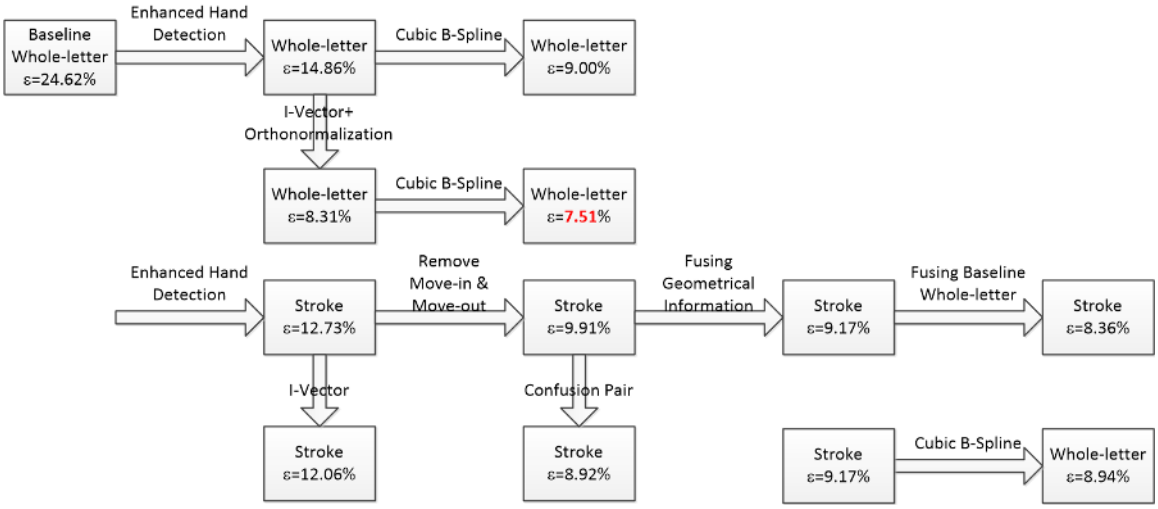


Figure 24: The summary of the proposed result in dataset I, the i-vector feature with cubic B-spline interpolation is the most effective one.

prefer doing each letter on the same location of the steering wheel.

Second, we can study the spatial-chromatic-temporal clustering method called super voxel with reduced complexity. Although directly applying super-pixel such as SLIC is not so useful, it can be used as a validation tool for hand detection. We believe that it is possible to derive new gesture features such as profiles upon the gestural volume formed by captured frames and the temporal axis. Third, with the experiences in i-vector and joint factor analysis, we believe it is possible to optimize its EM algorithm with different domain specific constraints. For example, based on what was found in the study of orthogonality, adding this constraint in the Stiefel manifold optimization was equivalent to doing Gram-Schmidt after each M-step, but both of them are beneficial to the system. In addition, directly using discriminative training as a post processing tool is only of limited help. So how to embed the supervised information into the i-vector extraction scheme and how to relax the affine model assumption will be also quite worth studying.

It is believed that this scheme can also be applied to other engineering problems such as speaker recognition, audio or video event tagging, and even big data problems. Specifically, by properly interpreting the tested variables and cooperating with big data strategies, the proposed system can be extended to study the following issues: human medical data,

brain neural information, and economic, political, and social security issues related to most human and social data. These possibilities might become true in future because of the huge progress of in computer technology on both hardware and software.

CHAPTER 7

APPENDICES

7.1 Data Acquisition Setup

The hand-written data acquisition was carried out in the auto front seat region. Basically, the image region covered a steering wheel where the hand-written actions are performed. The camera was fixed next to the dome light that is in the upper-right position behind the driver. The axial line of the camera lens was approximately pointed toward the central point of the steering wheel as shown in schematic diagram (see Figure 25). The type specifications of the Dragonfly 2 HICOL-CS-BOX camera, with Sony ICX204 CCD, 1/3", 4.65 μm sensor, are at 1024x768 YUV422 resolution and 15 fps and 10ms shutter speed, and other setup followed the default setup. Most experiments were taken in day-time and some results were taken in the evening when the sun light was dimming. There were 17 persons, including 10 men and 6 women, joined the data accumulation in the period from May to August in 2010. Another dataset was also acquired with the same setup but with different stroke order in 2013. For each one of the 26 English letters, users could write in their preferred stroke orders. Before doing the pre-processing, a manual inspection and a test run was done to examine the quality of the acquired image data. In this phase, several films which could not be properly recognized or did not match the specified letters were discarded so that the final data sets were about 4,211.

7.2 PCA, PPCA, and I-Vector

PCA assumed the observed data vectors $\boldsymbol{o}_t, t = 1, 2, \cdots, T$ are given. First of all, the observed vectors were projected onto a reduced space with the following model:

$$\boldsymbol{o}_t = \boldsymbol{m} + \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{\epsilon}, \tag{156}$$

where $\boldsymbol{m}$ is the mean of the observed vectors of size T, $\boldsymbol{W}$ is the unknown projection matrix to be determined, $\boldsymbol{x}_t$ is the low dimensional representation of data vector $\boldsymbol{o}_t$, and $\boldsymbol{\epsilon}$ is an

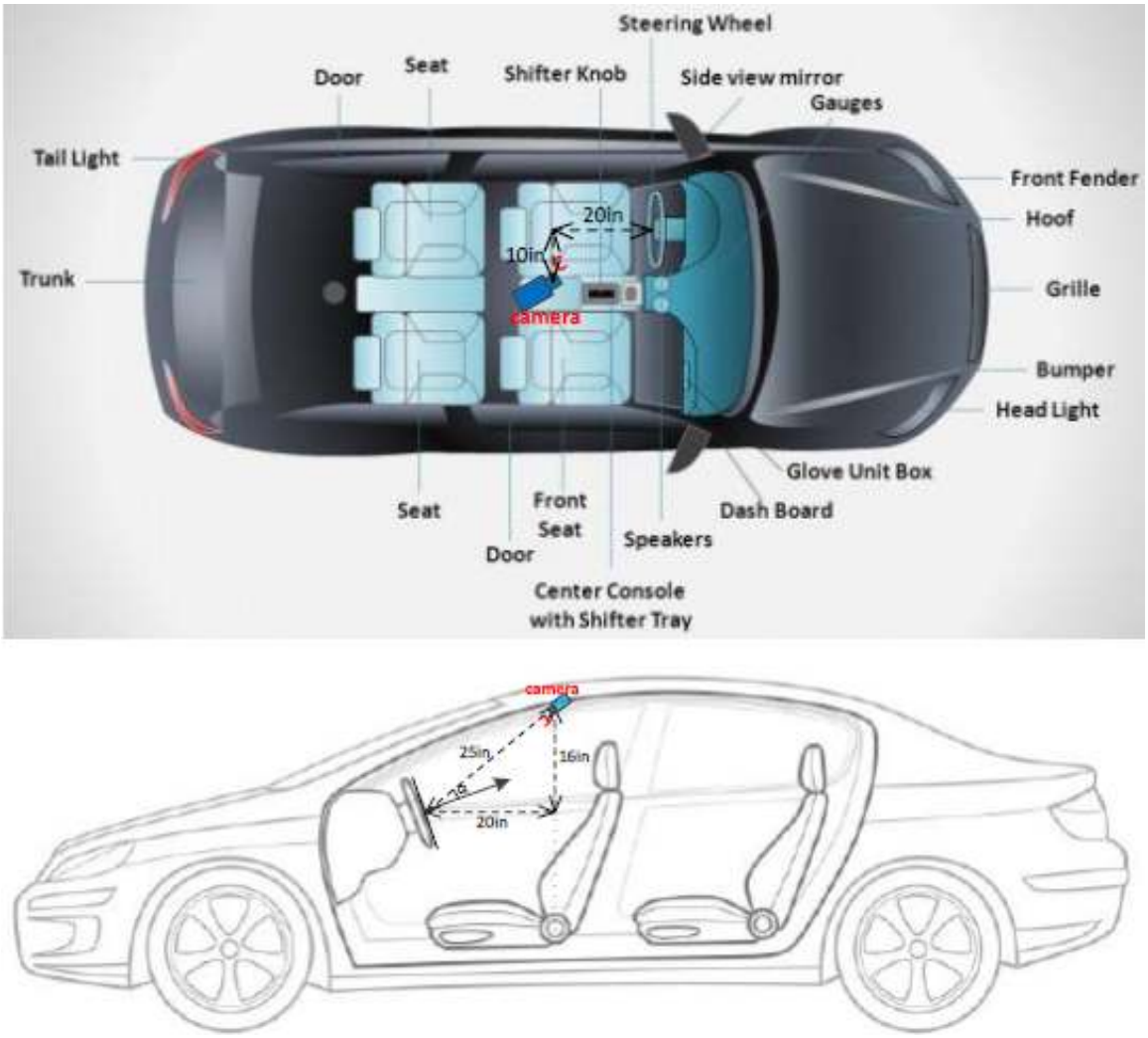


Figure 25: The in-car data acquisition setup.

isotropic Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$.

The goal was to optimize the orthonormal matrix $\boldsymbol{W}$ that could maximize the resulting covariance in its reduced space:

$$\hat{\boldsymbol{W}} = arg \max_{\boldsymbol{W}} \boldsymbol{W}\boldsymbol{S}\boldsymbol{W}^T \text{ such that } \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}, \tag{157}$$

where $\boldsymbol{S}$ is the sample covariance matrix of the data $\{\boldsymbol{o}_t, t = 1, 2, \cdots, T\}$. Applying Lagrange multiplier term $\boldsymbol{\lambda}(\boldsymbol{I} - \boldsymbol{W}^T\boldsymbol{W})$ to above objective function and taking derivative with respect to $\boldsymbol{W}$, when its derivative reach $\mathbf{0}$, one has:

$$2\boldsymbol{W}\boldsymbol{S} - \boldsymbol{\lambda}\boldsymbol{W} = 0, \tag{158}$$

which means PCA was to take the projection matrix from the eigen vector of the data sample covariance matrix.

As for PPCA, one still had the same dimensional reduction model as shown in Eq. (156), but now we assumed that the reduced space vectors are independently and identically distributed (i.i.d.) sampled from a standard Normal distribution, $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The orthogonality requirement was dropped and the only latent variable to be determined was $\boldsymbol{x}$. To find a proper projection matrix $\boldsymbol{W}$, the EM algorithm is required.

$$\boldsymbol{o}|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{m} + \boldsymbol{W}\boldsymbol{x}, \sigma^2\boldsymbol{I}), \tag{159}$$

$$\boldsymbol{o} \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}). \tag{160}$$

After some algebraic manipulations with the help of the Bayes theorem and "completing the square", it can be shown that

$$\boldsymbol{x}|\boldsymbol{o} \sim \mathcal{N}((\boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{W}^T(\boldsymbol{o} - \boldsymbol{m}), \sigma^2(\boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I})^{-1}), \tag{161}$$

Since $\boldsymbol{x}$ is a random variable, the MLE of $\boldsymbol{W}$ cannot be directly obtained. It can be done via doing EM algorithm with the following auxiliary function in E-step:

$$Q^{PPCA}(\hat{\boldsymbol{W}}|\boldsymbol{W}) = \sum_{n=1}^{N} \mathcal{E}_{\boldsymbol{x}|\boldsymbol{o}} \{\, log[P(\boldsymbol{o}|\boldsymbol{x})P(\boldsymbol{x})] \,\}, \tag{162}$$

where $P(\boldsymbol{o}|\boldsymbol{x})$ and $P(\boldsymbol{x})$ are evaluated with Eq. (159) and $\boldsymbol{x} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, respectively. The M-step is done by taking derivative with respect to $\boldsymbol{W}$ and setting the result to zero. The following relation is obtained.

$$\hat{\boldsymbol{W}} = \left[\sum_{n=1}^{N}(\boldsymbol{o}_n - \boldsymbol{m})\langle\boldsymbol{x}_n\rangle^T\right]\left[\sum_{n=1}^{N}\langle\boldsymbol{x}_n\boldsymbol{x}_n^T\rangle\right]^{-1} \tag{163}$$

where $\langle\boldsymbol{x}_n\rangle$ and $\langle\boldsymbol{x}_n\boldsymbol{x}_n^T\rangle$ can be done in E-step with $P(\boldsymbol{x}|\boldsymbol{o})$:

$$\langle\boldsymbol{x}_n\rangle = (\boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{W}^T(\boldsymbol{o}_n - \boldsymbol{m}), \tag{164}$$

$$\begin{aligned}\langle\boldsymbol{x}_n\boldsymbol{x}_n^T\rangle &= Var_{\boldsymbol{x}|\boldsymbol{o}}(\boldsymbol{x}|\boldsymbol{o}) + \langle\boldsymbol{x}_n\rangle\langle\boldsymbol{x}_n\rangle^T \\ &= \sigma(\boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I})^{-1} + \langle\boldsymbol{x}_n\rangle\langle\boldsymbol{x}_n\rangle^T.\end{aligned} \tag{165}$$

Therefore, the projection matrix and resulting reduced dimensional representations can be obtained by Eq. (163) and (164), respectively.

The i-vector extraction, however, utilizes the following model:

$$\boldsymbol{\mu} = \boldsymbol{m} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{\epsilon}, \tag{166}$$

Although their mathematical forms are similar, as shown in Eq's (156) and (166), there are two major differences between PCA/PPCA and i-vector. First, i-vector assumes a diagonal but non-isotropic noise sources, i.e. $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is a diagonal positive definite matrix. Second, the "observed" vector under i-vector scheme is the super-vector that is actually latent due to the fact that one did not know the exact Gaussian mixture membership. Therefore, one could only "observe" the super-vector in expected sense instead of "directly observing" them as in PCA/PPCA. In other words, the latent variables for i-vector schemes are mixture membership and the low dimensional representation $\boldsymbol{y}$ which is the i-vector itself. The rest assumption such as $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ will stay the same.

Following the derivation of PPCA, this study used the noise assumption, Bayes theorem, and the "completing the quadratic term". Then one can write the conditional distribution of $\boldsymbol{y}$ for given a set of observation data $\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_T$ with T samples and the current estimate of the projection matrix $\overline{\boldsymbol{V}}$ to be

$$p(\boldsymbol{y}|\{\boldsymbol{o}_1, \boldsymbol{o}_2, \cdots, \boldsymbol{o}_T\}, \overline{\boldsymbol{V}}) = \mathcal{N}\left(\langle \boldsymbol{y} \rangle, \boldsymbol{I} + \sum_{t=1}^{T} \overline{\boldsymbol{V}}\boldsymbol{\Sigma}^{-1}\overline{\boldsymbol{V}}\right) \tag{167}$$

$$\langle \boldsymbol{y} \rangle = \left[\boldsymbol{I} + \sum_{t=1}^{T} \overline{\boldsymbol{V}}\boldsymbol{\Sigma}^{-1}\overline{\boldsymbol{V}}\right]^{-1}\left[\sum_{t=1}^{T} \overline{\boldsymbol{V}}^{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{o}_t - \boldsymbol{m})\right] \tag{168}$$

$$\langle \boldsymbol{y}\boldsymbol{y}^{T} \rangle = \boldsymbol{I} + \sum_{t=1}^{T} \overline{\boldsymbol{V}}\boldsymbol{\Sigma}^{-1}\overline{\boldsymbol{V}} + \langle \boldsymbol{y} \rangle \langle \boldsymbol{y} \rangle^{T} \tag{169}$$

Using the fact that the super-vector with exact membership is unknown, the Q function should be modified. By considering the whole training data with L samples and the HMM instead of the GMM, the auxiliary function can be re-written as:

$$
\begin{aligned}
Q(\boldsymbol{\Lambda};\overline{\boldsymbol{\Lambda}}) =& \Bigg[\sum_{\ell=1}^{L}\sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T_\ell} P(\boldsymbol{S},\boldsymbol{M}|\boldsymbol{o}_t^\ell,\overline{\boldsymbol{\Lambda}})\mathcal{E}_{\boldsymbol{y}|\boldsymbol{o},\overline{\boldsymbol{\Lambda}}}\Big\{ logP(\boldsymbol{o}_t^\ell|\boldsymbol{S},\boldsymbol{M},\boldsymbol{\Lambda}\Big\}\Bigg]) \\
=& \Bigg[\sum_{\ell=1}^{L}\sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T_\ell} \overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)\mathcal{E}_{\boldsymbol{y}|\boldsymbol{o},\overline{\boldsymbol{\Lambda}}}\Big\{ log\Big(\frac{1}{(2\pi)^{\frac{D}{2}} det(\boldsymbol{\Sigma}_{s,m})^{\frac{1}{2}}}\Big) \\
& -\frac{1}{2}(\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m} - \boldsymbol{V}_{s,m}\boldsymbol{y}_\ell)^T\boldsymbol{\Sigma}_{s,m}^{-1}(\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m} - \boldsymbol{V}_{s,m}\boldsymbol{y}_\ell)\Big\}\Bigg] \qquad (170)
\end{aligned}
$$

where $\boldsymbol{V}_{s,m}$ is a sub-matrix of $\boldsymbol{V}$ corresponding to the state s and mixture m. Note it has D rows and the same number of columns as $\boldsymbol{V}$. Also, the membership probability $\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)$ has the same definition as that in HMM and the subscript $\boldsymbol{o}_t^\ell$ indicates the observed vector at time t for ℓ-th sample of the L-sample training data. In addition, the expectation $\mathcal{E}_{\boldsymbol{y}|\boldsymbol{o},\overline{\boldsymbol{\Lambda}}}$ is taken for all observed data vectors that are assumed to belong to the current given state alignments and mixtures. That is, when Eq.(166) is approximately valid as $\boldsymbol{y}$ is estimated, the expected value of the vector $\boldsymbol{y}_\ell$ could be done by taking the mean of the following modified distribution:

$$
\boldsymbol{y}_\ell \sim \mathcal{N}\Bigg(\langle\boldsymbol{y}_\ell\rangle, \boldsymbol{I} + \sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)\boldsymbol{V}_{s,m}^T\boldsymbol{\Sigma}_{s,m}^{-1}\boldsymbol{V}_{s,m}\Bigg) \qquad (171)
$$

$$
\begin{aligned}
\langle\boldsymbol{y}_\ell\rangle =& \Bigg[\boldsymbol{I} + \sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)\boldsymbol{V}_{s,m}^T\boldsymbol{\Sigma}_{s,m}^{-1}\boldsymbol{V}_{s,m}\Bigg]^{-1} \\
& \Bigg[\sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)\boldsymbol{V}_{s,m}^T\boldsymbol{\Sigma}_{s,m}^{-1}(\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m})\Bigg], \\
=& \Big[\boldsymbol{I} + \sum_{s,m} N_{s,m}(\ell)\boldsymbol{V}_{s,m}^T\boldsymbol{\Sigma}_{s,m}^{-1}\boldsymbol{V}_{s,m}\Big]^{-1}\Big[\sum_{s,m}\boldsymbol{V}_{s,m}^T\boldsymbol{\Sigma}_{s,m}^{-1}\boldsymbol{F}_{s,m}(\ell)\Big], \qquad (172)
\end{aligned}
$$

$$
\langle\boldsymbol{y}_\ell\boldsymbol{y}_\ell^T\rangle = \boldsymbol{I} + \sum_{s=1}^{N}\sum_{m=1}^{C}\sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)\boldsymbol{V}_{s,m}^T\boldsymbol{\Sigma}_{s,m}^{-1}\boldsymbol{V}_{s,m} + \langle\boldsymbol{y}_\ell\rangle\langle\boldsymbol{y}_\ell\rangle^T, \qquad (173)
$$

where

$$
N_{s,m}(\ell) = \sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m), \qquad (174)
$$

$$
\boldsymbol{F}_{s,m}(\ell) = \sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)(\boldsymbol{o}_t^\ell - \boldsymbol{\mu}_{s,m}), \qquad (175)
$$

are also known as zero-th and first order Baum-Welch statistics, respectively.

In M-step, this work takes derivative for the auxiliary function with respect to $\boldsymbol{V}_{s,m}$ and set the result to zero, It will give the following result:

$$\sum_{\ell=1}^{L}\sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)\boldsymbol{\Sigma}_{s,m}^{-1}\mathcal{E}_{\boldsymbol{y}|\boldsymbol{o},\overline{\boldsymbol{\Lambda}}}\{(\boldsymbol{o}_t^\ell-\boldsymbol{\mu}_{s,m}-\boldsymbol{V}_{s,m}\boldsymbol{y}_\ell)\boldsymbol{y}_\ell^T\}=\boldsymbol{0}, \tag{176}$$

thus

$$\begin{aligned}\boldsymbol{V}_{s,m}=&\Big[\sum_{\ell=1}^{L}\sum_{t=1}^{T_\ell}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)(\boldsymbol{o}_t^\ell-\boldsymbol{\mu}_{s,m})\langle\boldsymbol{y}_\ell\rangle^T\Big]\Big[\sum_{\ell=1}^{L}\sum_{t=1}^{T}\overline{\gamma}_{\boldsymbol{o}_t^\ell}(s,m)(\langle\boldsymbol{y}_\ell\boldsymbol{y}_\ell\rangle^T)\Big]^{-1},\\ =&\Big[\sum_{\ell=1}^{L}\boldsymbol{F}_{s,m}(\ell)\langle\boldsymbol{y}_\ell\rangle^T\Big]\Big[\sum_{\ell=1}^{L}N_{s,m}(\ell)(\langle\boldsymbol{y}_\ell\boldsymbol{y}_\ell\rangle^T)\Big]^{-1}.\end{aligned} \tag{177}$$

Therefore, under the EM framework, Eq.(172) and (177) are used iteratively to refine the projection matrix $\boldsymbol{V}$. Note the original HMM parameters are served as information of the universe and only $\boldsymbol{y}$ and $\boldsymbol{V}$ are iteratively updated, so one can actually drop the bar above the membership function from above equations. After the EM algorithm converged (typically 10 iterations are chosen) , Eq.(172) is used to compute the i-vector $\boldsymbol{y}_\ell$.

REFERENCES

[1] R. Sharma, V. I. Pavlovic, and T. S. Huang, "Toward Multimodal Human-Computer Interface," Proceedings of the IEEE, vol. 86, no. 5, pp. 853–869, 1998.

[2] A. Jaimes and N. Sebe, "Multimodal Human–Computer Interaction: A Survey," Computer Vision and Image Understanding, vol. 108, no. 1, pp. 116–134, 2007.

[3] L. Garay-Vega, A. Pradhan, G. Weinberg, B. Schmidt-Nielsen, B. Harsham, Y. Shen, G. Divekar, M. Romoser, M. Knodler, and D. Fisher, "Evaluation of Different Speech and Touch Interfaces to In-Vehicle Music Retrieval Systems," Accident Analysis & Prevention, vol. 42, no. 3, pp. 913–920, 2010.

[4] C. Muller and G. Weinberg, "Multimodal Input in the Car, Today and Tomorrow," Multimedia, IEEE, vol. 18, no. 1, pp. 98–103, 2011.

[5] A. Sankar and C.-H. Lee, "A Maximum-Likelihood Approach to Stochastic Matching for Robust Speech Recognition," Speech and Audio Processing, IEEE Trans. on, vol. 4, no. 3, pp. 190–202, 1996.

[6] M. Tonnis, V. Broy, and G. Klinker, "A Survey of Challenges Related to the Design of 3D User Interfaces for Car Drivers," in Proc. IEEE 3D User Interfaces 2006, pp. 127–134, IEEE, 2006.

[7] M. Jæger, M. Skov, N. Thomassen, et al., "You Can Touch, but You Can't Look: Interacting with In-Vehicle Systems," in Proc. ACM Conf. Human Factors in Computing Systems, pp. 1139–1148, ACM, 2008.

[8] S. Oviatt, P. Cohen, L. Wu, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, et al., "Designing the User Interface for Multimodal Speech and Pen-Based Gesture Applications: State-of-the-Art Systems and Future Research Directions," Human-Computer Interaction, vol. 15, no. 4, pp. 263–322, 2000.

[9] F.-S. Chen, C.-M. Fu, and C.-L. Huang, "Hand Gesture Recognition Using A Real-Time Tracking Method and Hidden Markov Models," Image and Vision Computing, vol. 21, no. 8, pp. 745–758, 2003.

[10] M. Chen, Universal Motion-Based Control and Motion Recognition. PhD thesis, Georgia Institute of Technology, August 2013.

[11] M. Elmezain and A. Al-Hamadi, "Gesture Recognition for Alphabets from Hand Motion Trajectory Using Hidden Markov Models," in Signal Processing and Information Technology, 2007 IEEE International Symposium on, pp. 1192–1197, IEEE, 2007.

[12] O. Barnich and M. Van Droogenbroeck, "ViBe: A Universal Background Subtraction Algorithm for Video Sequences," Image Processing, IEEE Transactions on, vol. 20, no. 6, pp. 1709–1724, 2011.

[13] T. Kurata, T. Okuma, M. Kourogi, and K. Sakaue, "The Hand Mouse: GMM Hand-Color Classification and Mean Shift Tracking," in Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 2001. Proceedings. IEEE ICCV Workshop on, pp. 119–124, IEEE, 2001.

[14] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[15] P. Hong, M. Turk, and T. Huang, "Gesture Modeling and Recognition Using Finite State Machines," in Proc. IEEE Conf. Automatic Face and Gesture Recognition, pp. 410–415, IEEE, 2000.

[16] W. T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition," in Proc. IEEE Conf. Automatic Face and Gesture Recognition, vol. 12, pp. 296–301, 1995.

[17] N. A. Ibraheem and R. Z. Khan, "Vision Based Gesture Recognition Using Neural Networks Approaches: A Review," International Journal of Human Computer Interaction (IJHCI), vol. 3, no. 1, pp. 1–12, 2012.

[18] T. Sim, S. Baker, and M. Bsat, "The CMU Pose, Illumination, and Expression (PIE) Database," in Proc. IEEE Automatic Face and Gesture Recognition 2002, pp. 46–51, IEEE, 2002.

[19] S. Impedovo, B. Marangelli, and A. Fanelli, "A Fourier Descriptor Set for Recognizing Nonstylized Numerals," Systems, Man and Cybernetics, IEEE Trans. on, vol. 8, pp. 640–645, 1978.

[20] Z. Ren, J. Yuan, and Z. Zhang, "Robust Hand Gesture Recognition Based on Finger-Earth Mover's Distance with A Commodity Depth Camera," in Proceedings of the 19th ACM International Conference on Multimedia, pp. 1093–1096, ACM, 2011.

[21] H. Yoon, J. Soh, Y. Bae, and H. Seung Yang, "Hand Gesture Recognition Using Combined Features of Location, Angle and Velocity," Pattern Recognition, vol. 34, no. 7, pp. 1491–1501, 2001.

[22] L. Bretzner, I. Laptev, and T. Lindeberg, "Hand Gesture Recognition Using Multi-Scale Colour Features, Hierarchical Models and Particle Filtering," in Proc. IEEE Automatic Face and Gesture Recognition 2002, pp. 423–428, IEEE, 2002.

[23] X. Liu and K. Fujimura, "Hand Gesture Recognition Using Depth Data," in Proc. IEEE Automatic Face and Gesture Recognition 2004, pp. 529–534, IEEE, 2004.

[24] C. Li and M. K. Kitani, "Pixel-Level Hand Detection in Ego-Centric Videos," in Proc. Computer Vision and Pattern Recognition, pp. 3570–3577, IEEE, 2013.

[25] P. Dreuw, T. Deselaers, D. Rybach, D. Keysers, and H. Ney, "Tracking Using Dynamic Programming for Appearance-Based Sign Language Recognition," in Proc. IEEE Conf. Automatic Face and Gesture Recognition, pp. 293–298, IEEE, 2006.

[26] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A Survey of Skin-Color Modeling and Detection Methods," Pattern Recognition, vol. 40, no. 3, pp. 1106–1122, 2007.

[27] K. Nandakumar, Y. Chen, S. Dass, and A. Jain, "Likelihood Ratio-Based Biometric Score Fusion," Pattern Analysis and Machine Intelligence, IEEE Trans. on, pp. 342–347, 2007.

[28] T. Jebara and A. Pentland, "Parametrized Structure from Motion for 3D Adaptive Feedback Tracking of Faces," in Proc. Computer Vision and Pattern Recognition (CVPR) 1997, pp. 144–150.

[29] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm," Journal of the Royal Statistical Society. Series B (Methodological), pp. 1–38, 1977.

[30] G. McLachlan and T. Krishnan, The EM Algorithm and Extensions, vol. 382. John Wiley & Sons, 2007.

[31] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe, "A Boosted Particle Filter: Multitarget Detection and Tracking," in Proc. European Conference on Computer Vision (ECCV), pp. 28–39, Springer, 2004.

[32] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Journal of Fluids Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[33] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[34] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in Proceedings of ECML-98, 10th European Conference on Machine Learning (C. Nédellec and C. Rouveirol, eds.), no. 1398, (Chemnitz, DE), pp. 137–142, Springer Verlag, Heidelberg, DE, 1998.

[35] Y. Chen, X. Zhou, and T. Huang, "One-Class SVM for Learning in Image Retrieval," in Proc. IEEE Intl. Conf. Image Processing (ICIP), IEEE, 2001.

[36] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," ACM Trans. on Intelligent Systems and Technology, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[37] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," Neural Computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[38] M. Chen, A. Kundu, and J. Zhou, "Off-Line Handwritten Word Recognition Using A Hidden Markov Model Type Stochastic Network," Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 16, no. 5, pp. 481–496, 1994.

[39] A. C. Surendran, C.-H. Lee, and M. Rahim, "Nonlinear Compensation for Stochastic Matching," Speech and Audio Processing, IEEE Trans. on, vol. 7, no. 6, pp. 643–655, 1999.

[40] M. Zobl, M. Geiger, B. Schuller, M. Lang, and G. Rigoll, "A Real-Time System for Hand Gesture Controlled Operation of In-Car Devices," in Proc. IEEE Multimedia and Expo 2003 (ICME 2003), vol. 3, pp. III–541, IEEE, 2003.

[41] G. Lu, D. Zhang, and K. Wang, "Palmprint Recognition Using Eigenpalms Features," Pattern Recognition Letters, vol. 24, no. 9-10, pp. 1463–1467, 2003.

[42] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images," in Proc. Computer Vision and Pattern Recognition (CVPR) 2011, pp. 1297–1304.

[43] K. Imagawa, S. Lu, and S. Igi, "Color-Based Hands Tracking System for Sign Language Recognition," in Proc. Automatic Face and Gesture Recognition 1998, pp. 462–467.

[44] L. Brethes, P. Menezes, F. Lerasle, and J. Hayet, "Face Tracking and Hand Gesture Recognition for Human-Robot Interaction," in Proc. International Conference on Robotics and Automation (ICRA) 2004, vol. 2, pp. 1901–1906.

[45] F. Hofmann, P. Heyer, and G. Hommel, "Velocity Profile Based Recognition of Dynamic Gestures with Discrete Hidden Markov Models," Proc. Gesture and Sign Language in Human-Computer Interaction: Intl. Gesture Workshop, pp. 81–95, 1998.

[46] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, "Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Life Spans," Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 30, no. 10, pp. 1728–1740, 2008.

[47] M. J. Black and A. D. Jepson, "A Probabilistic Framework for Matching Temporal Trajectories: Condensation-Based Recognition of Gestures and Expressions," in Proc. European Conference on Computer Vision (ECCV) '98, pp. 909–924, Springer, 1998.

[48] R. Rowe, U. Uludag, M. Demirkus, S. Parthasaradhi, and A. Jain, "A Multispectral Whole-Hand Biometric Authentication System," in Proc. Biometrics Symposium, 2007, pp. 1–6.

[49] V. Pavlovic, R. Sharma, and T. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 19, no. 7, pp. 677–695, 1997.

[50] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A Survey on Pixel-Based Skin Color Detection Techniques," in Proc. Graphicon, vol. 3, pp. 85–92, Moscow, Russia, 2003.

[51] Y. Wu and T. Huang, "Color Tracking by Transductive Learning," in Proc. IEEE Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 133–138, IEEE, 2000.

[52] Q. Zhu, K. Cheng, C. Wu, and Y. Wu, "Adaptive Learning of An Accurate Skin-Color Model," in Proc. IEEE Automatic Face and Gesture Recognition 2004, pp. 37–42, IEEE, 2004.

[53] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 34, no. 11, pp. 2274–2282, 2012.

[54] N. Ohta and A. Robertson, Colorimetry: Fundamentals and Applications. John Wiley & Sons, 2006.

[55] J. Ramirez, J. Górriz, and J. Segura, "Voice Activity Detection. Fundamentals and Speech Recognition System Robustness," Robust Speech Recognition and Understanding, pp. 1–22, 2007.

[56] J. Shi and C. Tomasi, "Good Features to Track," in Proc. Computer Vision and Pattern Recognition, pp. 593–600, IEEE, 1994.

[57] H. Haussecker and D. Fleet, "Computing Optical Flow with Physical Models of Brightness Variation," Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 23, no. 6, pp. 661–673, 2001.

[58] H. Walker and J. Lev, Statistical Inference. Henry Holt and Company, 1953.

[59] G. S. Fishman, Monte Carlo. Springer, 1996.

[60] G. W. Corder and D. I. Foreman, Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach. John Wiley & Sons, 2009.

[61] W. J. Conover and W. Conover, Practical Nonparametric Statistics. Wiley New York, 1980.

[62] S. Watanabe, A. Nakamura, and B.-H. Juang, "Model Adaptation for Automatic Speech Recognition Based on Multiple Time Scale Evolution," in Proc. Intl. Conf. Speech Communication and Technology (INTERSPEECH), pp. 1081–1084, ISCA, 2011.

[63] J. C. Kim, H. Rao, and M. A. Clements‘, “Formant Frequency Tracking Using Gaussian Mixtures with Maximum A Posteriori Adaptation,” in Proc. Intl. Conf. Speech Communication and Technology (INTERSPEECH), ISCA, 2013.

[64] A. D. Wilson and A. F. Bobick, “Realtime Online Adaptive Gesture Recognition,” in Proc. IEEE Intl. Conf. Pattern Recognition (ICPR), vol. 1, pp. 270–275, IEEE, 2000.

[65] J.-L. Gauvain and C.-H. Lee, “Maximum A Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains,” Speech and Audio Processing, IEEE Trans. on, vol. 2, no. 2, pp. 291–298, 1994.

[66] Y. Cheng, Z. Feng, F. Weng, and C. Lee, “Enhancing Model-Based Skin Color Detection: From Low-Level RGB Features to High-Level Discriminative Binary-Class Features,” in Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP), pp. 1401–1404, IEEE, 2012.

[67] K. E. Van De Sande, T. Gevers, and C. G. Snoek, “Evaluating Color Descriptors for Object and Scene Recognition,” Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 32, no. 9, pp. 1582–1596, 2010.

[68] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun Database: Large-Scale Scene Recognition from Abbey to Zoo,” in Computer Vision and Pattern Recognition (CVPR), IEEE conference on, pp. 3485–3492, IEEE, 2010.

[69] M. Celenk, “A Color Clustering Technique for Image Segmentation,” Computer Vision, Graphics, and Image Processing, vol. 52, no. 2, pp. 145–170, 1990.

[70] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, “Adaptive Histogram Equalization and Its Variations,” Computer Vision, Graphics, and Image Processing, vol. 39, no. 3, pp. 355–368, 1987.

[71] D. Shapira, S. Avidan, and Y. Hel-Or, “Multiple Histogram Matching.,” in Proc. International Conference on Image Processing (ICIP), pp. 2269–2273, 2013.

[72] M. Figueiredo and A. Jain, “Unsupervised Learning of Finite Mixture Models,” Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 24, no. 3, pp. 381–396, 2002.

[73] C. Liu, S. Jaeger, and M. Nakagawa, “Online Recognition of Chinese Characters: the State-of-the-Art,” Pattern Analysis and Machine Intelligence, IEEE Trans. on, vol. 26, no. 2, pp. 198–213, 2004.

[74] S. Lu, Y. Ren, and C. Suen, “Hierarchical Attributed Graph Representation and Recognition of Handwritten Chinese Characters,” Pattern Recognition, vol. 24, no. 7, pp. 617–632, 1991.

[75] H. Kim and J. Kim, "Hierarchical Random Graph Representation of Handwritten Characters and its Application to Hangul Recognition," Pattern Recognition, vol. 34, no. 2, pp. 187–201, 2001.

[76] Q. Li, B.-H. Juang, Q. Zhou, and C.-H. Lee, "Automatic Verbal Information Verification for User Authentication," Speech and Audio Processing, IEEE Trans. on, vol. 8, no. 5, pp. 585–596, 2000.

[77] M. Nakai, N. Akira, H. Shimodaira, and S. Sagayama, "Substroke Approach to HMM-Based On-Line Kanji Handwriting Recognition," in Proc. IEEE Conf. Document Analysis and Recognition, pp. 491–495, IEEE, 2001.

[78] R. C. Rose, "Discriminant Wordspotting Techniques for Rejecting Non-Vocabulary Utterances in Unconstrained Speech," in Proc. Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP), vol. 2, pp. 105–108, IEEE, 1992.

[79] Y.-C. Cheng, K. Li, Z. Feng, F. Weng, and C.-H. Lee, "Online Whole-Word and Stroke-Based Modeling for Hand-Written Letter Recognition in In-Car Environments," in Proc. Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP), 2013.

[80] T. H. H. Maung et al., "Real-Time Hand Tracking and Gesture Recognition System Using Neural Networks," World Academy of Science, Engineering and Technology, vol. 50, pp. 466–470, 2009.

[81] F. Hildebrand, Introduction to Numerical Analysis. Dover Books on Mathematics Series, Dover Publications, 1987.

[82] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing. New York, NY, USA: Cambridge University Press, 1992.

[83] A. Muñoz, T. Blu, and M. Unser, "Least-Squares Image Resizing Using Finite Differences," Image Processing, IEEE Trans. on, vol. 10, no. 9, pp. 1365–1378, 2001.

[84] A. Gotchev, K. Egiazarian, J. Vesma, and T. Saramaki, "Edge-Preserving Image Resizing Using Modified B-Splines," in Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP), vol. 3, pp. 1865–1868, IEEE, 2001.

[85] S. Lee, G. Wolberg, and S. Y. Shin, "Scattered Data Interpolation with Multilevel B-Splines," Visualization and Computer Graphics, IEEE Trans. on, vol. 3, no. 3, pp. 228–244, 1997.

[86] M. Arigovindan, M. Suhling, P. Hunziker, and M. Unser, "Variational Image Reconstruction from Arbitrarily Spaced Samples: A Fast Multiresolution Spline Solution," Image Processing, IEEE Trans. on, vol. 14, no. 4, pp. 450–460, 2005.

[87] C. Reinsch, "Smoothing by Spline Functions," Numerische Mathematik, vol. 10, no. 3, pp. 177–183, 1967.

[88] C. de Boor, A Practical Guide to Splines. Springer-Verlag, 1978.

[89] M. Unser, A. Aldroubi, and M. Eden, "B-spline Signal Processing. I. Theory," Signal Processing, IEEE Trans. on, vol. 41, no. 2, pp. 821–833, 1993.

[90] E. Maeland, "On the Comparison of Interpolation Methods," Medical Imaging, IEEE Trans. on, vol. 7, no. 3, pp. 213–217, 1988.

[91] Z. Mihajlovic, A. Goluban, and M. Zagar, "Frequency Domain Analysis of B-spline Interpolation," in Proc. Conf. Industrial Electronics, vol. 1, pp. 193–198, IEEE, 1999.

[92] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, Computer Methods for Mathematical Computations, vol. 8. Prentice-Hall Englewood Cliffs, NJ, 1977.

[93] L. Piegl and W. Tiller, The NURBS Book. 1997. Springer.

[94] F. N. Fritsch and R. E. Carlson, "Monotone Piecewise Cubic Interpolation," SIAM Journal on Numerical Analysis, vol. 17, no. 2, pp. 238–246, 1980.

[95] C. Kruger, "Constrained Cubic Spline Interpolation," Chemical Engineering Applications, 2003.

[96] J. W. Gibbs, "Fourier's Series," Nature, vol. 59, p. 200, 1898.

[97] G. Wolberg and I. Alfy, "Monotonic Cubic Spline Interpolation," in Computer Graphics International, 1999. Proceedings, pp. 188–195, IEEE, 1999.

[98] Carl and D. Boor, A Practical Guide to Splines, vol. 27. Springer-Verlag New York, 1978.

[99] L. A. Piegl, Fundamental Developments of Computer-Aided Geometric Modeling. Academic Pr, 1993.

[100] M. Jones and J. Rehg, "Statistical Color Models with Application to Skin Detection," in Proc. IEEE. Conf. Computer Vision and Pattern Recognition (CVPR), vol. 1, 1999.

[101] Ø. Due Trier, A. K. Jain, and T. Taxt, "Feature Extraction Methods for Character Recognition-A Survey," Pattern Recognition, vol. 29, no. 4, pp. 641–662, 1996.

[102] Z.-L. Bai and Q. Huo, "A study on the Use of 8-Directional Features for On-line Handwritten Chinese Character Recognition," in Proc. IEEE Conf. Document Analysis and Recognition, pp. 262–266, IEEE, 2005.

[103] M. Kherallah, L. Haddad, A. M. Alimi, and A. Mitiche, "On-Line Handwritten Digit Recognition Based on Trajectory and Velocity Modeling," Pattern Recognition Letters, vol. 29, no. 5, pp. 580–594, 2008.

[104] P. A. Torres-Carrasquillo, E. Singer, M. A. Kohler, R. J. Greene, D. A. Reynolds, and J. R. Deller Jr, "Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features.," in Proc. Intl. Conf. Speech Communication and Technology (INTERSPEECH), ISCA, 2002.

[105] Q. Chen, N. D. Georganas, and E. M. Petriu, "Real-Time Vision-Based Hand Gesture Recognition Using Haar-Like Features," in Proc. IEEE Conf. Instrumentation and Measurement Technology Conference, pp. 1–6, IEEE, 2007.

[106] A. Barla, F. Odone, and A. Verri, "Histogram Intersection Kernel for Image Classification," in Proc. International Conference on Image Processing (ICIP), vol. 3, pp. III–513, IEEE, 2003.

[107] W. M. Campbell, D. E. Sturim, D. A. Reynolds, and A. Solomonoff, "SVM Based Speaker Verification Using A GMM Supervector Kernel and NAP Variability Compensation," in Acoustics, Speech and Signal Processing, IEEE Trans. on, vol. 1, pp. I–I, IEEE, 2006.

[108] P. Kenny and P. Dumouchel, "Experiments in Speaker Verification Using Factor Analysis Likelihood Ratios," in Proc. ODYSSEY-The Speaker and Language Recognition Workshop, 2004.

[109] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," Audio, Speech, and Language Processing, IEEE Trans. on, vol. 19, no. 4, pp. 788–798, 2011.

[110] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models," Digital Signal Processing, vol. 10, no. 1, pp. 19–41, 2000.

[111] N. Dehak, R. Dehak, P. Kenny, N. Brummer, P. Ouellet, and P. Dumouchel, "Support Vector Machines Versus Fast Scoring in the Low-Dimensional Total Variability Space for Speaker Verification," in Proc. Intl. Conf. Speech Communication and Technology (INTERSPEECH), pp. 1559–1562, ISCA, 2009.

[112] D. Garcia-Romero and C. Y. Espy-Wilson, "Analysis of i-Vector Length Normalization in Speaker Recognition Systems," in Proc. Intl. Conf. on Speech Communication and Technology (INTERSPEECH), pp. 249–252, ISCA, 2011.

[113] V. Hautamäki, K. A. Lee, A. Larcher, T. Kinnunen, B. Ma, and H. Li, "Variational Bayes Logistic Regression as Regularized Fusion for NIST SRE 2010," in Proc. ODYSSEY-The Speaker and Language Recognition Workshop, 2012.

[114] P. Kenny, G. Boulianne, and P. Dumouchel, "Eigenvoice Modeling with Sparse Training Data," Speech and Audio Processing, IEEE Trans. on, vol. 13, no. 3, pp. 345–354, 2005.

[115] C.-Y. Lin and H.-C. Wang, "Language Identification Using Pitch Contour Information in the Ergodic Markov Model," in Proc. Intl. Conf. Acoustics, Speech, and Signal Processing (ICASSP), vol. 1, pp. 193–196, IEEE, 2006.

[116] Y. Zhang, Z.-J. Yan, and Q. Huo, "A New i-Vector Approach and Its Application to Irrelevant Variability Normalization Based Acoustic Model Training," in Proc. Machine Learning for Signal Processing (MLSP), pp. 1–6, IEEE, 2011.

[117] J.-F. Bonastre, N. Scheffer, D. Matrouf, C. Fredouille, A. Larcher, A. Preti, G. Pouchoulin, N. Evans, B. Fauve, and J. Mason, "ALIZE/SpkDet: A State-of-the-Art Open Source Software for Speaker Recognition," in Proc. ODYSSEY- the Speaker and Language Recognition Workshop, 2008.

[118] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi Speech Recognition Toolkit," in IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, IEEE Signal Processing Society, Dec. 2011. IEEE Catalog No.: CFP11SRW-USB.

[119] Z. Wen and W. Yin, "A Feasible Method for Optimization with Orthogonality Constraints," Mathematical Programming, vol. 142, no. 1-2, pp. 397–434, 2013.

[120] V. Hautamäki, Y.-C. Cheng, P. Rajan, and C.-H. Lee, "Minimax i-Vector Extractor for Short Duration Speaker Verification," in Proc. Intl. Conf. Speech Communication and Technology (INTERSPEECH), ISCA, 2013.