HIDDEN MARKOV MODEL BASED VISUAL SPEECH RECOGNITION

DONG LIANG

(M. Eng.)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

To Mom and Dad…

with forever love and respect

Acknowledgements

I would like to thank my advisor, Associate Professor Foo Say Wei, for his vision and encouragement throughout the years, for his invaluable advice, guidance and tolerance.

Thanks to Associate Professor Lian Yong, for all the support, understanding and perspectives throughout my graduate study.

Thanks are also due to my friends in DSA Lab, Gao Qi, Mao Tianyu, Xiang Xu, Lu Shijian, Shi Miao . . . , for the happy and sad time we had been together.

My special thanks to my mother and father. Without their steadfast support, under circumstances sometimes difficult, this research would not have been possible. I am also indebted to my little niece, Pei Pei, who brightened my mind with her smile.

Dong Liang

July 2004

Contents

Summary

It is found that speech recognition can be made more accurate if other than audio information is also taken into consideration. Such additional information includes visual information of the lip movement, emotional contents and syntax information. In this thesis, studies on lip movement are presented.

Classifiers based on Hidden Markov Model (HMM) are first explored for modeling and identifying the basic visual speech elements. The visual speech elements are confusable and easily distorted by their contexts, and a classifier to distinguish the minute difference among the different categories is desirable. For this purpose, new methods are developed that focus on improving the discriminative power and robustness of the HMM classifiers. Three training strategies for HMM, referred to as two-channel training strategy, Maximum Separable Distance (MSD) training strategy and HMM Adaptive Boosting (AdaBoosting) strategy, are proposed. The two-channel training strategy and the MSD training strategy adopt a criterion function called separable distance to improve the discriminative power of an HMM while HMM AdaBoosting strategy applies AdaBoost technique to HMM modeling to build a multi-HMM classifier to improve the robustness of HMM. The proposed training methods are applied to identify context-independent, context-dependent visual speech units and confusable visual words. The results indicate that higher recognition accuracy can be attained than using traditional training approaches.

The thesis also covers the investigation of recognition of words and phrases in visual speech. The approach is to partition words and phrases into the basic visual speech models. Level building on AdaBoost-HMM classifiers is studied for this purpose. The proposed method employs a specially designed probability trellis to decode a sequence of best-matched AdaBoost-HMM classifiers. A Viterbi matching algorithm is also presented, which facilitates the process of sequence partition with the application of specially tailored recognition units and transition units. These methods, together with the traditional level building method, are applied to recognize/decompose words, phrases and connected digits. The comparative results indicate that the proposed approaches outperform the traditional approach in recognition accuracy and processing speed.

Two other research topics covered in the thesis are strategies of extending the applicability of a visual speech processing system to unfavorable conditions such as when the head of the speaker moves during speech or the visual features of the speaker are greatly unknown. A 3D lip tracking method is proposed that 3D deformable templates and a template trellis are adopted to capture lip dynamics. Compared with the traditional 2D deformable template method, this approach can well compensate the deformation caused by the movement of the speaker's head during speech. The strategy of mapping visual speech between a source speaker and a destination speaker is also proposed with exploration of HMMs with special mapping terms. The mapped visual speech elements can be accurately identified by the speech models of the destination speaker. This approach may be further studied for eliminating the speaker-dependency of a visual speech recognition system.

List of Tables

List of Figures

Chapter 1

Introduction

Visual speech processing, which is more generally referred to as automatic lip reading, is the technique of decoding speech content from visual clues such as the movement of the lip, tongue and facial muscles. In recent years, investigation in this area has become an attractive aspect of multimedia. The experience of lip reading, however, is not new to us. When language came into being, speech perception by lip reading had also started. In our daily communication, lip reading is widely used whether consciously or unconsciously. In noisy environments such as bus stop, stock market or office, much of the speech information is retrieved from the visual clues. For the hearing-impaired people, lip reading plays an even more important role for them to understand conversation.

1.1 Human lip reading

The time of the first study on human lip reading cannot be traced. It is believed that the ability of lip reading is mastered when a man begins to learn a language. Scientific studies on human lip reading have been carried out since 1900s. Sumby and Pollack [1] found that visual information can lead to significant improvement of human's perception of speech especially in a noisy environment. They also showed that the incorporation of visual signals gives rise to 12dB gain in SNR. In 1956, Neely *et al* [2] studied the factors that affect human lip reading, which include illumination, distance from the speaker, detection of teeth and tongue. They also found that the accuracy of lip reading using frontal views of the speaker is better than that using other view angles. The contribution of visual information to speech perception has been demonstrated in a wide variety of conditions: in noisy environments [3], with highly complex sentences [4], with conflicting auditory and visual speech [5][6], and with asynchronous auditory and visual speech information [7]. Under all these conditions, improvement to speech perception was observed.

The reasons that underlie the improvement of speech perception by lip reading were also investigated. Visual speech predominantly provides information about the place of articulation of the spoken sounds. Human observers may thus pay attention to the correct signal source [3]. Besides this, movements of the articulators naturally accompany the production of speech sound. Human observers use these two sources of speech information from an early age and thus they can fuse the two types of information quickly and accurately [8][9].

A comprehensive study on the relationship between visual speech and acoustic speech was carried out by McGurk and his colleagues [10]. The famous "McGurk effect" indicates that human perception of speech is bimodal in nature. When human observers were presented with conflicting audio and visual stimuli, the perceived sound may exist in either modality. For example, when a person heard the sound /ba/ but saw the speaker saying /ga/, the person might not perceive either /ga/ or /ba/. Instead, what he perceived was /da/. Table 1.1 gives some examples of the McGurk effect.

The McGurk effect stimulated further investigations on the relationship between visual speech and acoustic speech. Psychologists studied the McGurk effect that

Table 1.1: Examples of the McGurk effect

Audio	Visual	Perceived
ba	ga	da
pa	ga	ta
ma	ga	na

occurred across different languages [11] and the robustness of the McGurk effect [12]. They also proposed the "reverse McGurk effect" [13], i.e. the results of visual speech perception can be affected by the dubbed audio speech.

Since 1980s, the developments on human lip reading have attracted the attention of researchers on multimedia. Since then, computed-based visual speech processing became a branch of speech processing and much research work has been carried out.

1.2 Machine-based lip reading

The ability to perform lip reading was long regarded as the privilege of human beings because of the complexity of machine recognition. To convert the captured videos to speech information, the following processing must be undertaken: image processing, feature extraction, sequence modelling/identification, speech segmentation, grammar analysis and context analysis. If any of the composite processing modules malfunctions, the overall performance of lip reading becomes unreliable. In addition, the above mentioned processing units are inter-dependent. The individual processing units should have the ability to respond to the feedback from the other units. The difficulties involved in machine-based lip reading are even more enormous if the distinct features of lip dynamics are considered. First, the movement of the lip is slow compared with the corresponding acoustic speech signal. The low frequency feature of the lip motion indicates that the amount of information conveyed by the visual speech is very much smaller than that by the speech sound. Second, the variation between consecutive frames of visual images is small while such variation is important for recognition because they serve as the discriminative temporal features of visual speech. Third, the visual representations of some phonemes are confusable. For example, phonemes /f/ and /v/ are visually confusable as both of them have very similar sequence of mouth shapes where the upper teeth are touching the lower lip. It is commonly agreed that the basic visual speech elements in English, which are called visemes (the concepts about viseme are explained in detail in Section 2.2), can be categorized into 14 groups, while there are 48 phonemes used in acoustic speech. For example, phonemes /s/ and /z/ belong to the same viseme group. As a result, even if a word is partitioned into the correct viseme combination, it is still not guaranteed that the correct word can be decoded. Fourth, the visemes are easily distorted by the prior viseme and posterior viseme. The temporal features of a viseme can be very different under different contexts. As a result, the viseme classifiers have stricter requirement on the robustness than the phoneme classifiers.

Although there are many difficulties in machine-based lip reading, it does not mean that efforts made in this area are not worthwhile. First, many experiments proved that even if a slight effort was made toward incorporation of visual signal, the combined audio-visual recognizer would outperform the audio-only recognizer [14]-[17]. Second, some speech sounds which are easily confused in the audio domain such as “b” and “v”, “m” and “n”, are distinct in the visual domain [18]. These facts indicate that the information hidden in visual speech is valuable. In addition, the many potential applications of visual speech such as in computer-aided dubbing, speech-driven face animation, visual conferencing and tele-eavesdropping stimulate the interest of researchers. With the aid of modern signal processing technologies and computing tools, lip reading became a feasible research area and much inspiring work has been done on the theoretical aspects and applications of automatic lip reading. According to the order of the implementation of lip reading, the previous work concentrated on the following three aspects: 1) Lip tracking, 2) Visual features processing, 3) Language processing. These are elaborated in the following sections.

1.2.1 Lip tracking

The purpose of lip tracking is to provide an informative description of the lip motion. The raw input data to the lip reading system are usually video clips that indicate the production of a phoneme, word or sentence. The most direct means is to gather the color information of all the pixels of the image and feed them into the recognition modules. Actually, this was done by Yuhas *et al* [19]. The advantage of this approach is that there is no information loss during recognition. However, the disadvantage of the method is evident. First, the computations involved in processing the entire frame are intolerable. Second, this method is very sensitive to the change of illumination, position of the speaker's lips and camera settings.

The initial attempts on lip feature extraction were chiefly individual-image-oriented methods. By analyzing the color distribution of the image, the lip area was segmented by some image processing techniques. To improve the accuracy of image segmentation, image smoothing, Bayes thresholding, morphological image processing and "eigenlip" method were all used [20]-[22]. These approaches treated the video as a series of independent images. The geometric measures extracted from one frame were not relevant to the other frames. The individual-image-oriented approaches had the advantage of easy implementation and many mature image processing techniques could be adopted. However, the features obtained in this way might not be accurate enough and the continuity was not good.

Much of the recent work in visual analysis has centered on deformable models. The snake-based methods fit into this category. Snake was first proposed by Kass *et al* [23]. It allows one to parameterize a closed contour by minimizing an energy function that is the sum of the internal energy and external energy. The internal energy acts to keep the contour smooth while the external energy acts to attract the snake to the edges of the image. The curves used as "snakes" can be B-splines [24][25], single-span quadratics and cubic splines, e.g. Bezier curves [26][27]. Further researches were carried out to improve the performance of the snakes such as the robustness, continuity or viscosity. For example, surface learning [28][29] and flexible appearance models [30] were adopted in snake-fitting.

Deformable template algorithm is another deformable model approach. The method was proposed by Yuille *et al* [31] and was applied to capture lip dynamics by Hennecke *et al* [32]. Like snakes, the deformable templates also give an energy function for parameter adjustment. Besides this, it provides a parameterized model that imposes some constraints on the tracking process. The prior knowledge about the tracked object is revealed by the initial settings of the template. When applied to lip tracking, the templates that describe the lip contour may be simple, e.g. several parabolas [33]. Many researchers have used deformable templates to achieve good results in visual speech recognition. Several extensions to the method have also been studied. Kervrann *et al* suggested incorporating Kalman filtering techniques into deformable templates [34]. The method was also extended from 2D to 3D by Lee *et al* [35].

The two deformable model approaches mentioned above are continuous-image-oriented methods. In the tracking process, the relation between continuous frames is taken into consideration. As a result, the geometric features obtained demonstrate good continuity for continuous flow of images.

Other lip tracking approaches include Active Shape Models (ASMs) and Active Appearance Models (AAMs). The ASM was first formulated in 1994 [36] and was introduced to lip reading by Luettin *et al* [37]. The ASM is a shape-constrained iterative fitting algorithm. The shape constraint comes from the use of a statistical shape model which is called point distribution model. In the ASM tracking process, the conventional iterative algorithm [36], simplex algorithm [38] or multi-resolution image pyramid algorithm [39] could be applied. The AAM was proposed by Cootes *et al* [40]. It is a statistical model of both shape and gray-level appearance. The fitting process of AAM is largely similar to that of the ASM where iterations were implemented to minimize the difference between the target image and the image synthesized by the current model parameters.

Like the deformable models, ASM and AAM approaches also focus on the changes between consecutive images. As a result, the features extracted also demonstrate good continuity.

The ultimate goal of visual speech processing is to decode speech content from the lip motion. Lip tracking accomplishes the first half of the task, in which the raw image sequence is converted into tractable feature vector sequence. Subsequent processing will be carried out to extract the information conveyed by the decoded feature vectors.

1.2.2 Visual features processing

The literature on automatic lip reading is fairly limited compared with that on speech recognition. However, because visual speech and acoustic speech have much in common, some techniques that have achieved success in acoustic speech recognition can be applied to visual speech recognition with some modifications. These techniques/tools include Time Warping, Neural Network, Fuzzy Logic and Hidden

Markov Models. Early lip reading systems only used some simple pattern recognition strategies as the designer might face severe hardware speed limitations. In some cases, a major goal of the research was simply to demonstrate the feasibility of the concept. Some scholars consider Petajan as the first researcher that systematically investigated machine-based lip reading. In his design, linear time warping and some distance measures were used for recognition [20]. Later, Mase and Pentland also applied linear time warping approach to process the feature vector sequences [41]. Although these studies laid emphasis on the time warping aspect of visual speech, the linear time warping is not an appropriate technique to process natural speech because the temporal features of natural speech are far from linear.

Dynamical time warping was used in a later version of Petajan's lip reading system [42]. With further consideration on the non-linear features of visual speech, some improvement on the recognition accuracy was observed.

The Neural Network (multi-layer perceptron, MLP) was first applied to lip reading by Yuhas *et al* [43]. However, the MLP is not flexible enough for processing time sequences. In 1992, Time-Delayed Neural Network (TDNN) was explored by Stork *et al* [44]. The inputs to Stork's system were dots of the raw image as introduced in Section 1.2.1. Such a design made full use of the information conveyed by the video and was computationally expensive. The recognition results of Stork's system were better than that of time warping but were sensitive to the changes of the environment. Some improved TDNN designs were proposed and further experiments were conducted by Cosi *et al* [45] and Movellan [46].

Neural Network (NN) is a classical tool of pattern recognition. It has been intensively studied for more than half a century. From primitive McCulloch-Pitts's neuron model to today's MLP with millions of neurons, the theoretical aspects and applications of NN developed very fast. There are many types of NN and training strategies available for various requirements such as MLP [51][65], Support Vector Machines [49][50], Radial Basis Function (RBF) [54], TDNN [47][48] and Self-Organizing Feature Maps (SOFMs) [53]. As a result, NN-based lip reading is also a promising research area.

Another powerful tool for visual speech recognition is Hidden Markov Models (HMMs). The basic theory of HMM was published in a series of papers by Baum and his colleagues in the late 1960s and early 1970s. The process generated by HMMs has been widely studied in statistics. It is basically a discrete-time bivariate parametric process: the underlying process is a finite-state Markov chain; the explicit process is a sequence of conditionally independent random variables for a given state chain. HMM was first applied to lip reading by Goldschen in 1993 [55]. In Goldschen's system, HMM classifiers were explored for recognizing a closed set of TIMIT sentences. Because of its good performance and speed of computation, HMM was extensively applied to the subsequent lip reading systems for recognizing isolated words or non-sense words, consonant-vowel-consonant (CVC) syllables [56], digit set [57][58] and *AVletters* [59]. In the mean time, HMM-related techniques have advanced greatly. Tomlinson *et al* suggested a cross-product HMM topology, which allows asynchronous processing of visual signals and acoustic signals [60]. Luettin *et al* used HMMs with an early integration strategy for both isolated digit recognition and connected digit recognition [61]. In recent years, coupled HMM, product HMM and factorial HMM are explored for audio-visual integration [62]-[65]. Details of the HMM-based visual speech processing techniques can be found in [66] and [67].

In addition to the techniques mentioned above, fuzzy logic was also applied to visual speech processing. In 1996, Silsbee presented a system that combined an acoustic speech recognizer and a visual speech recognizer with fuzzy logic [57].

Another example is the Boltzmann zippers that were used in Stork's lip reading system [68]. The recognition results indicated that fuzzy logic is a practical tool for processing visual speech and works well for small-vocabulary cases.

There are more mathematical and computational techniques/tools that can be applied to automatic lip reading than those listed above. In summary, the prospective techniques for visual speech processing should have the following characteristics in common: 1) Being sensitive to the temporal features of the investigated sequences. 2) Offering time-warping to the investigated sequences and 3) Showing tolerance to the erratic observations (good generalization).

1.2.3 Language processing

Language processing is the last stage of visual speech processing. It is considered the most intelligent part of a lip reading system because it imitates the complex mechanism of language perception of the human brain. In this step, lexical, syntactic, semantic and pragmatic information is incorporated to interpret the captured image sequences. Some preliminary knowledge of the machine-based language processing can be obtained if we observe what our brain does when we attempt to lipread: when we see the lip motion of a speaker, we will first check our memory to find the words that correspond to the lip shapes. Usually, there are many possible words that match a particular lip shape. Next, the brain selects the optimal word combination based on the context and syntax rules. This "optimal" word sequence is the one that has meaning in a certain linguistic environment. The micro-processing of lip reading is principally the interactions and information-exchanges among neurons. However, after decades of research, modern neuroethology and neurophysiology still cannot reveal the underlying mechanism of the interaction among neurons.

In the field of speech processing, language analysis chiefly focuses on the macroscopic process of the human brain rather than the behavior of the individual neurons. From the linguistic perspective, scientists have found that there are three factors that determine the performance of a language analyzer. One is linguistic rule, which governs how a word is built and how a sentence is constructed. For example, the *adj.*-ly morphology and the *Subject + Predicate + Object* sentence frame are valid in English. The second is the context, by which a listener can conclude the meaning of new language information and evaluate the previous interpretations. For example, if we hear the sentence "A ??g is barking", we can deduce that the uncertain part is "dog" even if the sound is ambiguous. The third factor is vocabulary. A poor vocabulary greatly limits the number of word combinations that match the input speech. For example, we cannot understand what a Japanese speaks if we have not learnt the language before.

To date, some advanced artificial intelligence systems can make simple sentences and correct syntax errors for an input sentence. These techniques was applied to acoustic speech processing such as the Dragon system and the IBM large-vocabulary speech processing system [69][70][71] with high degree of success. Regretfully, language analysis is not intensively used in visual speech processing because some problems involving lip tracking and visual feature processing still remain unresolved. In the experimental lip reading systems developed so far, the language processing was either neglected or very simple algorithm is used. For example, the visual speech processing system proposed by Matthews *et al* can recognize a number of *AVletters* [59]. Luettin's HMM-based classifier can identify digits [61]. In such small vocabulary cases and when the individual words are the targets of recognition, the use of language analysis is not warranted. Some simple language processing is adopted when there is only minor consideration given to the relationship between consecutive words. For example, Cosi's recognizer applied some grammar rules on the identification of continuous digits such as post codes or telephone numbers [72]. Cisar's limited-vocabulary lip reading system could recognize some sentences whose structures observed a set of predefined laws [73].

In brief, to apply the language processing techniques to visual speech processing, the lip tracking module and visual feature processing module should provide accurate word combinations for language processing. Only when the problems in lip tracking and visual features processing are solved, will language processing in automatic lip reading be fully developed.

1.2.4 Other research directions

The success of a lip reading system relies on, but is not limited to, the solutions of lip tracking, visual features processing and language processing. Other factors can also influence the robustness, accuracy and application scope of a lip reading machine. The researches carried out on audio-visual signal incorporation and speaker-dependency elimination may extend the applicability of a lip reading system.

Since human speech is bimodal, it is natural to associate the acoustic information and visual information while interpreting the speech. Classical approaches of audio-visual speech processing include early integration and late integration. In an early-integration design, the acoustic features, which are extracted from an acoustic decoder, are synthesized with visual features to build a macro feature vector. The macro feature vectors that indicate certain audio-visual production are processed by a joint-feature classifier. The early integration is so called because the information comprising visual and audio channels is integrated before identification. Methods described in [17][61][74][75] are all examples of early integration.

Late integration strategy, on the other hand, applies two independent recognition engines to identify the audio signals and video signals of the same audio-visual production, respectively. The identity of the input production is formulated by synthesizing the decisions from the two engines according to certain rules. The name “Late integration” is used because the decisions from the visual recognition channel and audio recognition channel are integrated after identification is implemented on both channels. Designs given in [66][75][76][77] are based on the late integration approach.

Most of the proposed lip reading designs are speaker-dependent. These systems can only be trained to analyze the visual speech of certain speaker(s). It is well-known that the shape and the movement of the lip can be very different across genders, races and ages. If a different speaker uses such a speaker-dependent system, the recognition accuracy will drop drastically. Literature on speaker-dependency of automatic lip reading is very limited. Only some preliminary studies were conducted in this area. For example, Luettin developed an HMM-based lip reading system with an early integration strategy for both speaker-independent digits recognition and speaker-independent connected-digit recognition in 1996 [61].

The problems involved in automatic lip reading are more than those enumerated above. Since lip reading is essentially a video-to-text conversion, some work has been carried out on the video end to provide smooth video streams or easy-to-tackle image sequences. The researches on this aspect include video capture [78], video sampling [79] and lip synchronization [80][81].

1.3 Contributions of the thesis

From the above discussion on the development of machine-based lip reading, it can be seen that some basic problems in lip tracking and visual features processing still exist. In this thesis, we focused on classifier design and training, sequence decomposition, lip tracking and speaker-dependency elimination.

1.) Viseme recognition: Studies on viseme recognition were first carried out. To achieve this goal, conventional single-HMM classifier was first constructed. The single-HMM classifiers were configured according to the temporal features of the visemes and were trained using the Baum-Welch method. This approach has the advantage of easy implementation. However, as the visemes are confusable and distorted by their contexts, the single-HMM classifiers sometimes cannot identify them with sufficient accuracy. To improve the discriminative power of the HMMs to separate similar observations and to enhance the robustness of the HMMs to better cover the erratic observations, the following three kinds of HMM-based classifiers: two-channel HMM classifiers, maximum separable distance (MSD) HMM classifiers and adaptively boosted (AdaBoost) HMM classifiers were adopted.

The two-channel HMM classifier and MSD HMM classifier were specially trained to improve the discriminative power of the HMMs. Both classifiers adopted a novel criterion function called separable distance for parameter adjusting. The separable distance indicates the difference between a pair of confusable sequences measured by an HMM. Greater value of separable distance means better chances of discriminating the confusable sequences. For the two-channel training strategy, the separable distance was adjusted by dividing the symbol output matrix of the HMM into two channels: one static channel to maintain the validity of the HMM and one dynamic channel to be modified to maximize the separable distance between the training pair. Because the static channel is usually obtained from a pre-trained HMM of the target process, such management might improve the discriminative power of the HMM and at the same time, maintain the goodness-of-fit of the trained two-channel HMM. The MSD training method, on the other hand, does not work on a pre-trained HMM. The parameters in the symbol output matrix were updated directly to maximize the separable distance. This approach is much simplified compared with the two-channel training strategy. The two-channel HMM classifiers were applied to identify the visemes and a hierarchical multi-HMM classifier was proposed for the application. The MSD HMM classifiers were applied to separate confusable words in visual speech. The decisions made by the respective HMMs were synthesized with eliminating series strategy. In both experiments, the discriminative power of the proposed HMM classifiers was improved compared with that of the conventional single-HMM classifier. The recognition rates of visemes and selected words were also higher than that by using single-HMM classifiers.

The improvement made to the HMM-based classifiers for identification of visemes in various contexts was also studied. Because the visemes demonstrate polymorphism under different contexts, traditional single-HMM classifiers may lack the ability to cover the erratic samples of a viseme. In the proposed design, adaptive boosting (AdaBoosting) was applied to HMMs. By calling the biased Baum-Welch estimation and weight adjusting strategy in the boosting iterations, a multi-HMM classifier was trained with the composite HMMs highlighting different groups of samples. Such a system was employed to identify the visemes in various contexts and the recognition accuracy was compared with that of the single-HMM classifiers. The comparative results showed that the AdaBoost HMM classifiers might better cover the spread-out samples of the visemes than single-HMM classifiers, and the recognition accuracy improved by about 16%.

2.) Recognition of continuous visual speech: This is a higher-level recognition because connected viseme units such as words, phrases, connected-digits are to be identified. In the field of HMM-based continuous speech processing, level building strategy is commonly adopted to link different HMMs to model words and sentences. The strategy of level building on single-HMM classifiers was presented first. Following that, the strategy of level building on AdaBoost HMM classifiers was proposed. Both approaches were applied to decompose selected word and phrase productions into visemes and the results were compared with one another. It was observed that the connected AdaBoost HMM classifiers could better recognize/decompose the target words and phrases than connected single-HMM classifiers. Because the computations involved in level building are enormous and sometimes intolerable, a Viterbi matching algorithm was also proposed in this study to facilitate the process of sequence partition. This method employed specially tailored recognition units and transition units to decode the target sequence into a chain of component units. The Viterbi approach had been applied to decode words, phrases and connected digits in visual speech. The results indicated that the recognition/decomposition accuracy using the proposed Viterbi matching algorithm was close to that using the conventional level building method while the computational load was less than one fourth of the latter.

3.) 3D lip tracking: In the lip tracking stage, a 3D deformable template algorithm was proposed to capture lip dynamics from natural speech. This method employed 3D templates to compensate for the variations caused by the rotation of the speaker's head and maintained a template trellis to track the movement of the lips. The tracking results of the 3D approach were compared with those using the 2D template tracking method and it was found that the deformation to the lip shapes caused by the movement of the speaker's head was better recovered with the proposed 3D template approach.

4.) Cross-speaker viseme mapping: The problems associated with speaker-dependency were also addressed in the thesis. HMM classifiers with mapping terms were proposed to mapping viseme productions among different speakers. This method provides an indirect approach of eliminating speaker-dependency of a visual speech processing system because the viseme production of an unknown speaker could be generated using the samples of a known speaker. Experiments conducted in the thesis indicated that the mapped visemes could be accurately identified by the true models of the unknown speaker.

The studies reported in the thesis followed a bottom-to-top thread for the construction of a visual speech processing system, in which the recognition of the basic visual speech elements (visemes) was first performed, and then the recognition of the continuous visual speech units (words and phrases) was investigated. Viseme classifiers/models, which would be the core recognition engine for future lip reading machine, were highlighted in the thesis and three training methods were proposed to improve their discriminative power and robustness. Recognition of words and phrases in visual speech was realized by partitioning them into viseme models. The proposed strategies were based on exhaustive searching methods such as level building and Viterbi matching. These approaches played an important role on associating the recognition of visemes and the recognition of connected viseme units in visual speech. The 3D template lip tracking method and the strategy of mapping visemes across speakers were two minor research topics covered in the thesis. These approaches extended the applicability of a visual speech processing system to unfavorable conditions such as when the head of the speaker was moving during speech or when the visual features of a speaker, e.g. the shape of the lips, remained greatly unknown.

1.4 Organization of the thesis

The reminder of the thesis is organized as follows. An introduction of the preprocessing of visual speech signals and the structure of a single-HMM classifier are given in Chapter 2. Training strategies based on separable-distance, which include two-channel training strategy and the MSD training strategy, are presented in Chapter 3. This is followed by discussion of HMM AdaBoosting technique in

Chapter 4. The level building strategy on HMM-based classifiers and the Viterbi matching algorithm for sequence partition are detailed in Chapter 5. The lip tracking strategy based on 3D deformable template and the method of mapping visual speech among different speakers are described in Chapter 6. The last chapter, Chapter 7, is the concluding chapter. Recommendations for future research are also presented in this chapter.

Chapter 2

Pre-processing of Visual Speech Signals and the Construction of Single-HMM Classifier

Some preliminary knowledge about visual speech processing is presented in this chapter, which include data acquisition, definition of the basic visual speech elements, image processing, feature extraction and the construction of HMM classifiers. The results of identifying visual speech elements under various experimental conditions are presented. These results demonstrate the power of the single-HMM classifier and are also used for comparison with other approaches that will be discussed in later chapters.

2.1 Raw data of visual speech

In visual speech domain, the raw data of visual speech are video clips that capture the movement of the lips, facial muscles, tongue and teeth during the productions of visual speech elements, words and sentences. It is proved by previous experiments

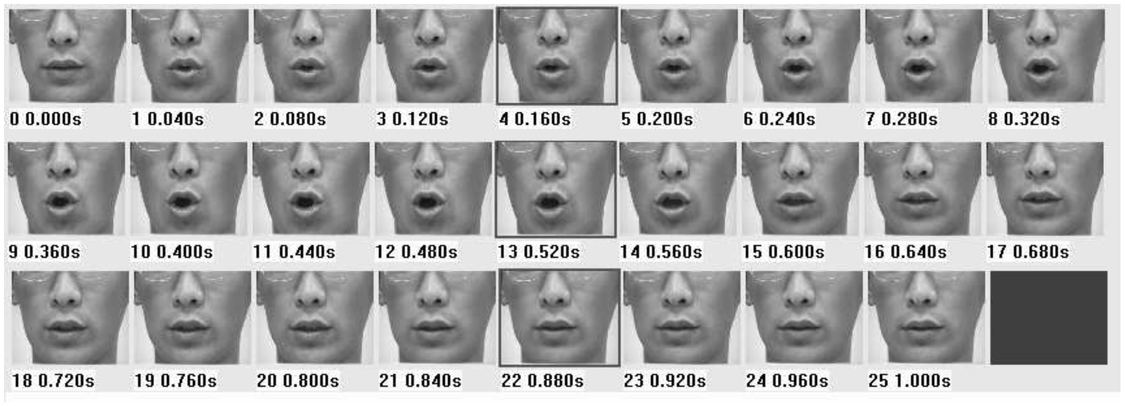

Figure 2.1: The video clip indicating the production of the word *hot*

that the frontal view of the speaker reveals much information of visual speech [2]. As a result, the frontal view of the speakers is adopted in our system. The speakers are asked to articulate the given text for a number of times. The movement of the facial area is captured as avi files at 25 frames per second and the video clips that indicate speech productions are manually segmented out of the video. The frames of the video clips are saved as 24-bit bitmap images, which are depicted in Fig.2.1.

It should note that for the segmented video clips, the prior and posterior images corresponding to the long silence of word productions are cut so that the main portion of the image sequence corresponds with the voiced phase.

In the acoustic domain, a number of speech databases such as TIMIT, YOHO, Switchboard, ELRA have been developed. These databases are widely used in speech recognition and become the benchmark for measuring the performance of a speech recognizer. In visual speech processing, although some audio-visual speech databases have been proposed [82][83][84], they are not widely accepted by the multimedia community. The data used in most visual speech experiments are independently recorded as visual speech recognition is very much speaker-dependent.

The samples used in our experiments are also self-recorded video clips. Several

English speakers (native and non-native English speakers, females and males) are invited to visually clearly produce some given texts for a number of times. The distance between the speaker and camera is fixed at 1 meter; white background is used; illumination condition is six fluorescent light sources fixed at the ceiling and the viewing angle is a frontal view. The heads of the speakers are fixed with props during speech. The video clips filmed under such controlled conditions are collected to build a database. Studies on the video clips in the database reveals the following features of visual speech:

1.) The movement of the lips varies slowly over time. Compared with the speech signal, which has significant frequency components up to 4kHz, the lip motion is a very low-frequency signal. Information conveyed by lip movement is thus limited.

2.) Visual speech is context-sensitive. The same sound may have different visual representations when it appears in different contexts. In statistical jargon, samples of certain sound production demonstrate spread-out distribution.

3.) Visual speech is also speaker-dependent. The facial features of speakers demonstrate great difference across race, sex, skin color, etc. Adaptation to such variance should also be considered while building a recognition system.

The properties mentioned above are actually the difficulties that may be encountered while carrying out visual speech recognition. To solve these problems, a bottom-to-top study is carried out in this thesis.

2.2 Viseme

In visual speech domain, the smallest visibly distinguishable unit is commonly referred to as viseme. A viseme is a short period of lip movement that can be used to describe a particular sound. Like phonemes which are the basic building blocks of

Table 2.1: Visemes defined in MPEG-4 Multimedia Standards

Viseme No.	Phonemes	Examples	Viseme No.	Phonemes	Examples
1	p, b, m	push, bike, milk	8	n, l	note, lose
2	f, v	find, voice	9	r	read
3	T, D	think, that	10	A:	jar
4	t, d	teach, dog	11	e	bed
5	k, g	call, guess	12	I	tip
6	tS, dZ, S	check, join, shrine	13	Q	shock
7	s, z	set, zeal	14	U	good

sound of a language, visemes are the basic constituents for the visual representations of words. The relationship between phonemes and visemes is a many-to-one mapping. For example, although phonemes /b/, /m/, /p/ are acoustically distinguishable sounds, they are grouped into one viseme category as they are visually confusable, i.e. all are produced by similar sequence of mouth shapes.

An early viseme grouping was suggested by Binnied *et al* in 1974 [85] and was applied to some identification experiments such as [86]. Viseme groupings in [87] are obtained by analyzing the stimulus-response matrices of the perceived visual signals. The recent MPEG-4 Multimedia Standards adopted the same viseme grouping strategy for face animation, in which fourteen viseme groups are included [88]. Unlike the 48 phonemes in English [100], the definition of viseme is not uniform in visual speech. In the respective researches conducted so far, different groupings may be adopted to fulfill specific requirements [89]. This fact may cause some confusion on evaluating the performance of viseme classifiers.

MPEG-4 is an object-based multimedia compression standard and plays an important role in the development of multimedia techniques. As a result, the viseme

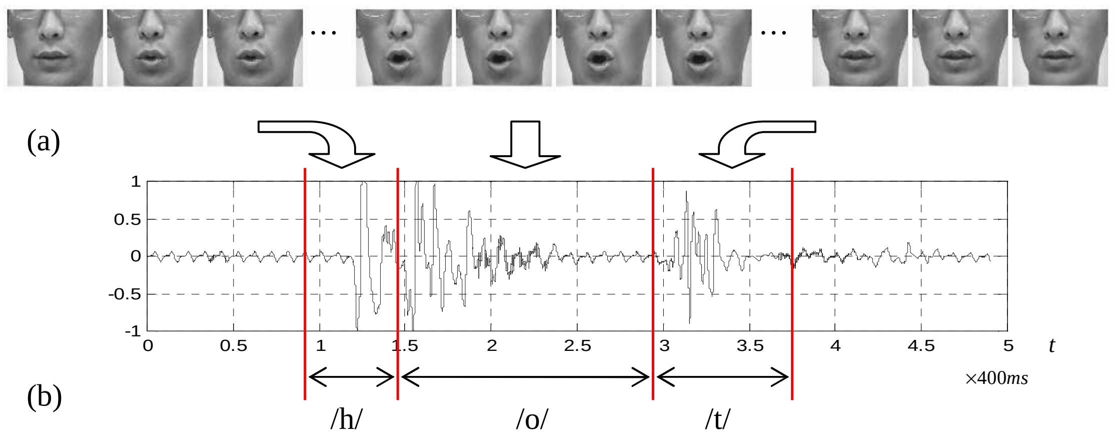

Figure 2.2: Segmentation of a viseme out of word production

(a) video clip and (b) acoustic waveform of the production of the word *hot*

categorization proposed in MPEG-4 is adopted in our experiments. The fourteen visemes defined in MPEG-4 are illustrated in Table 2.1. It is observed that each viseme corresponds to several phonemes or phoneme-like productions. Note that some consonants are not included in the table as their visual representations are chiefly determined by their adjoining phonemes and diphthongs are also not included as their visual representations are assumed to be combinations of the visemes illustrated in the table.

The visemes are liable to be distorted by their context. For example, the visual representations of the vowel /ai/ are very different when extracted from the words *hide* and *right*. A viseme thus demonstrates polymorphism under different contexts. For this reason, the samples of visemes are obtained in two approaches.

1.) The speaker is asked to produce an isolated viseme, starting with closed mouth and ending with closed mouth too. This kind of samples is referred to as context-independent viseme sample because the temporal features of a viseme are not affected by the context factors.

2.) The speaker is asked to produce some words that contain the target viseme. The video clips of the viseme are segmented from the word productions using the image sequences and the corresponding acoustic waveform, which is exemplified in Fig.2.2. The samples obtained in this way are referred to as context-dependent viseme samples because the adjoining sounds/visemes may greatly affect the temporal features of the viseme.

For each viseme, 140 text-dependent samples and 200 text-independent samples are collected for training and testing the viseme classifiers.

2.3 Image processing and feature extraction of visual speech

Most classifiers can only take feature vectors of visual speech as the input patterns rather than the video clips (however, there are exceptions such as [43][44][90]). Geometric features of the lips are commonly adopted because they may reveal the physical aspects of the mouth and are also invariant to the changes of the viewing angle, position and illumination. The procedures of extracting the geometric features include lip segmentation, template matching and feature extraction.

2.3.1 Lip segmentation

The position of the lips is first located and the approximate lip region is segmented from the image. Human lips are of darker color than the color of the other part of the facial area. There are distinguishable borders between the lips and the surrounding skin. A proper threshold can therefore be set for segmentation based on, say the red, green, blue (RGB) factors or the hue, saturation, value (HSV) factors of the image. Hue-saturation factors are used in our system because they are relatively insensitive to changes in the absolute brightness. The RGB to HSV

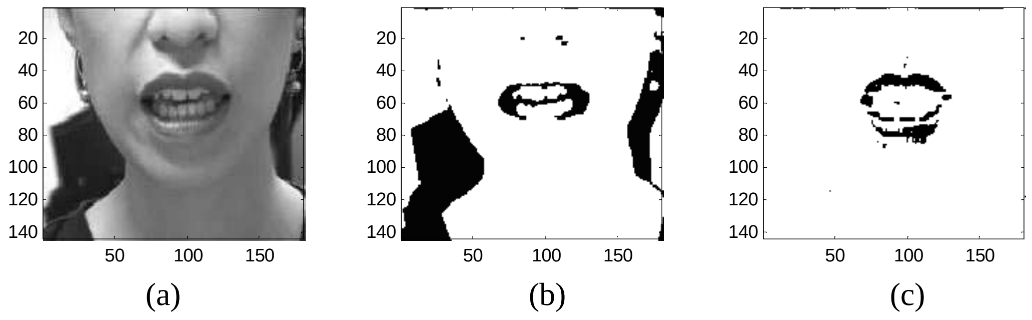

Figure 2.3: (a) original image (b) lip localization (c) segmented lip area

conversion algorithms for lip region detection proposed by [90][92] are adopted. First, the red-dominant area of the image is detected by means of red exclusion [92]. The approximate lip region is located as illustrated in Fig.2.3(b). Next, the raw image is transformed to a modified hue image with RGB to HSV transformation proposed in [90][91]. As depicted in Fig.2.4, the hue factors of the lip region and the remaining lip-excluded image account for different portions of the histogram. A proper threshold is extracted from the hue histogram of the entire image, which usually corresponds to a local minimum point (valley) in the histogram. The hue image is then compared with the chosen threshold and a binary image is obtained. Finally, logical AND operation of the binary image and the approximate lip region is performed. The lip region is then segmented as shown in Fig.2.3(c).

2.3.2 Edge detection using deformable template

Further processing is based on the deformable templates algorithm [31][32][93]. This approach maintains a geometric template with dynamic contours to fit an elastic object. The template is controlled by some parameters and is drawn near to the boundaries of the target in the tracking process. The contours of the lips

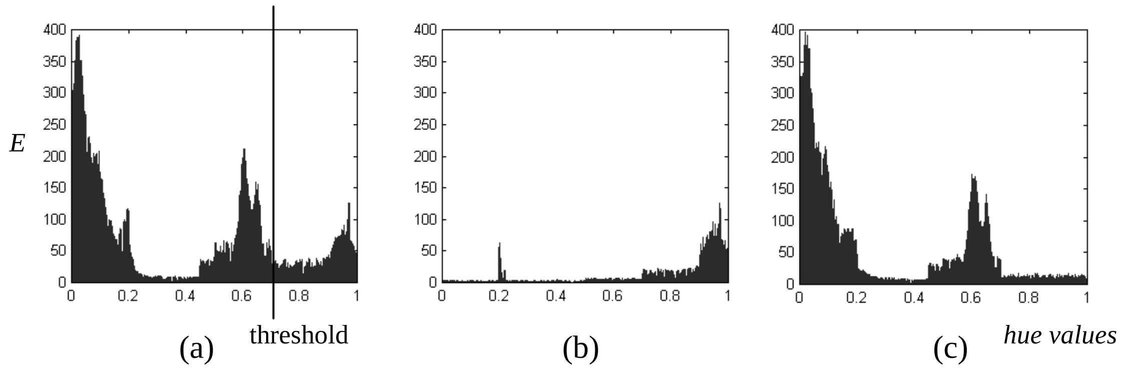

Figure 2.4: Separation of the lip region out of the image using hue distribution.

(a) Histogram of the hue component of the entire image

(b) Histogram of the hue component of the lip region

(c) Histogram of the hue component of the lip-excluded image

are simple compared with other moving objects such as human body or sign language. Many polynomial curves such as parabolas or elliptical curves can well match the boundaries of the lip [31][66]. In our experiments, the parameterized template is built up with ten Bezier curves and is proved effective. As depicted in Fig.2.5(a), eight curves characterize the lip contours and the other two curves describe the tongue when it is visible. The process of fitting the template to the lip image is to adjust the parameters (usually the control points) of the template to minimize certain energy function [26][27]. As depicted in Fig.2.5(a), the points in the small circles are control points of the template. The energy function takes four components as defined in Eq.(2.1)-(2.4).

$$E_{lip} = -\frac{1}{R_1}\int_{R_1} H(z)dz \tag{2.1}$$

$$E_{edge} = -\frac{1}{C_1 + C_2}\int_{C_1+C_2} |H^+(z) - H(z)| + |H^-(z) - H(z)|dz \tag{2.2}$$

$$E_{hole} = -\frac{1}{R_2 - R_3}\int_{R_2-R_3} H(z)dz \tag{2.3}$$

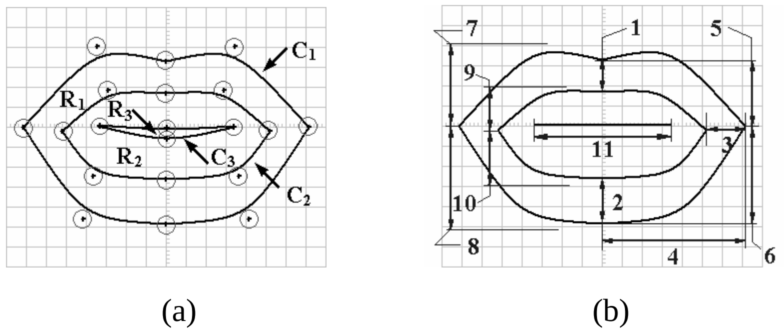

Figure 2.5: Deformable lip template for edge detection

(a) Parameterized lip template and the control points (b) Geometric measures extracted from the template. Thickness of the 1) upper bow, 2) lower bow, 3) lip corner. Position of the 4) lip corner, 5) upper lip, 6) lower bow. Curvature of the 7) upper-exterior boundary, 8) lower-exterior boundary, 9) upper-interior boundary, 10) the lower-interior boundary. 11) Width of the tongue (when it is visible).

$$E_{inertia} = ||\Gamma_{t+1} - \Gamma_t||^2 \tag{2.4}$$

where R_1, R_2, R_3, C_1 and C_2 are areas and contours as illustrated in Fig.2.5(a). $H(z)$ is a function of the hue of a given pixel z; $H^+(z)$ is the hue function of the closest right-hand side pixel and $H^-(z)$ is that of the closest left-hand side pixel of z. Γ_{t+1} and Γ_t are the matched templates at time $t+1$ and t. $||\Gamma_{t+1} - \Gamma_t||$ indicates the Euclidean distance between the two templates. Mathematically,

$$||\Gamma_{t+1} - \Gamma_t|| = \sqrt{\sum_{i=1}^{n}[(x_i^{t+1} - x_i^t)^2 + (y_i^{t+1} - y_i^t)^2]} \tag{2.5}$$

where (x_i^{t+1}, y_i^{t+1}) is the coordinate of the i-th control point of template Γ_{t+1}, (x_i^t, y_i^t) is that of template Γ_t, n is the number of control points and $n = 16$ for the lip template used in our experiments, The overall energy of the template E is the linear combination of the components as defined in Eq.(2.6).

$$E = c_1 E_{lip} + c_2 E_{edge} + c_3 E_{hole} + c_4 E_{inertia} \tag{2.6}$$

The energy terms of the tongue template are defined in a similar manner.

The template is matched against the lip region of the image sequence by adopting different values of the parameters $c_i (i = 1, 2, \cdots, 7)$ in a number of searching epochs (a detailed discussion is given in [31][32][93]). An example of the matched lip template and tongue template is given in Fig.2.5(b). It is manifested that the matched template is symmetric and smooth, and is therefore easy to process.

Eleven geometric measures are finally extracted from the matched template to build the feature vector of a specific lip shape. As depicted in Fig.2.5(b), these measures indicate the thickness of various parts of the lips, the positions of some key points and the curvatures of the lips. They are chosen as they uniquely determine the shape of the lips and best characterize the movement of the lips.

The collected feature vectors of different lip shapes are put through Principal Component Analysis (PCA) to reduce the dimensionality from 11D to 7D and are clustered into groups using the *K-means* algorithm. In the experiments conducted, 128 clusters (code words) are used in the vector database (code book). The means of the clusters compose the symbol set $O^{128} = \{O_1, O_2, \cdots, O_{128}\}$ of the HMM. They are also applied to encode the vector sequences that are presented to the system.

2.4 Single-HMM viseme classifier

The basic visual speech elements such as visemes are to be identified in this section. For HMM-based classifiers, a viseme is usually modelled by one HMM. The flow chart of the proposed viseme recognition system is depicted in Fig.2.6.

The input image sequence indicating certain viseme production is converted into vector sequence with the processing mentioned in Section 2.3. The subsequent model matching and viseme indexing are realized by HMM classifiers.

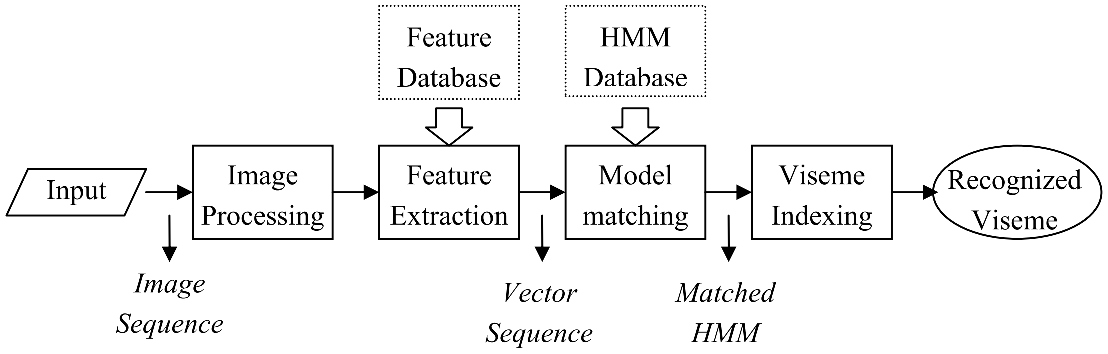

Figure 2.6: Flow chart of the viseme recognition system

2.4.1 Principles of Hidden Markov Model (HMM)

Hidden Markov Model (HMM) is more technically referred to as Hidden Markov Process (HMP) as the latter emphasizes the stochastic process rather than the model itself. HMM was first introduced by Baum *et al* in 1966 [94]. The basic theories/properties of HMM were introduced in full generality in a series of papers by Baum and his colleagues [95]-[98], which included the convergence of the entropy function of an HMM, the computation of the conditional probability, and the local convergence of the Maximal Likelihood (ML) parameter estimation of HMM. HMM is essentially a division of a process into a number of discrete states. While using an HMM to analyze a stochastic process, the observation sequence, say $x^T = (o_1, o_2, \cdots, o_t, \cdots, o_T)$ where o_t is the t-th symbol appearing in the sequence, is assumed to be output from a sequence of hidden states $s^T = (s_1, s_2, \cdots, s_T)$, where $s_i \in S^N$ and $S^N = \{S_1, S_2, \cdots, S_N\}$ is the state set of the HMM. Each state maintains a probability function or probability density function to indicate the likelihood of emitting certain symbols. The states are interconnected with each other by some state transition coefficients, which indicate the likelihood of a state repeating itself or transiting to another state. The relationship between the explicit process (observations), the hidden process (states) and the HMM is

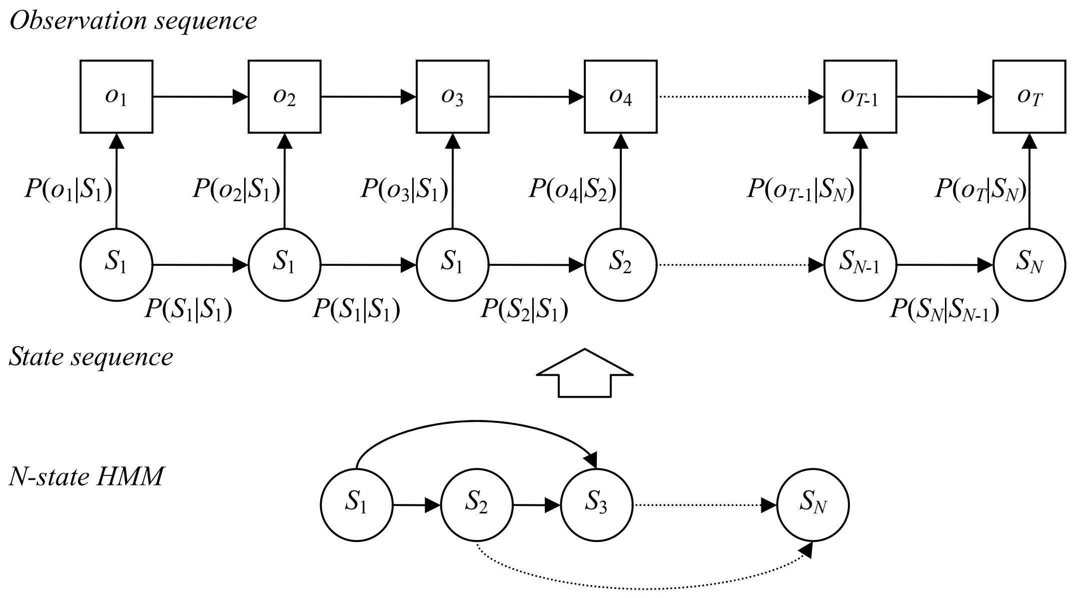

Figure 2.7: The relation between the observation sequence and the state sequence of an HMM with N states

depicted in Fig.2.7, where $P(S_j|S_i)(i,j=1,2,\cdots,N)$ is the likelihood of transition from S_i to S_j and $P(o_t|S_i)(t=1,2,\cdots,T)$ is the likelihood of emitting o_t in state S_i. If the output of an HMM takes discrete and finite values, e.g. from a finite symbol set $O^M=\{O_1,O_2,\cdots,O_M\}$, the HMM is called discrete; if the output takes continuous values, the HMM is called continuous.

Assume that $O^M=\{O_1,O_2,\cdots,O_M\}$ is the set of discrete symbol alphabet and $S^N=\{S_1,S_2,\cdots,S_N\}$ is the set of states. An N-state M-symbol HMM $\theta(\pi,A,B)$ is determined with the following three components [99][100]:

1.) $\pi=[\pi_1,\pi_2,\cdots,\pi_N]_{1\times N}$, where $\pi_i=P(s_1=S_i)$ indicates the probability that the first state s_1 is S_i.

2.) $A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{12} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}_{N\times N}$, where $a_{ij}=P(s_{t+1}=S_j|s_t=S_i)$ is the probability of transition from state S_i to S_j.

3.) $B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{12} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NM} \end{bmatrix}_{N \times M}$, where $b_{ij} = P(o_t = O_j | s_t = S_i)$ denotes the probability of outputting symbol O_j at state S_i.

If the observation space is continuous, the symbol emission matrix can be modeled using Gaussian mixtures [99], i.e.

$$b_i(o) = \sum_{q=1}^{Q} c_{iq} G(o, \mu_{iq}, \sigma_{iq}), \qquad \sum_{q=1}^{Q} c_{iq} = 1 \tag{2.7}$$

where $G(o, \mu_{iq}, \sigma_{iq})$ is a Gaussian distribution function with mean vector μ_{iq} and variance σ_{iq}, Q is the number of the Gaussian mixtures and $c_{iq}(q = 1, 2, \cdots, Q)$ is a non-negative coefficient.

For the input sequence x^T, the likelihood $P(x^T|\theta)$ can be computed by summing up all the states at each moment or can be more easily computed using the forward variables and backward variables [99][100]. The forward variables $\alpha_t(i)$ and the backward variables $\beta_t(i)$ for x^T, are defined in Eq.(2.8) and (2.9). For simplification, θ is assumed a discrete HMM in this section.

$$\alpha_t(i) = P(o_1, o_2, \cdots, o_t, s_t = S_i|\theta) \tag{2.8}$$

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \cdots, o_T|s_t = S_i, \theta) \tag{2.9}$$

where s_t is the t-th state variable of the state sequence s_T. $\alpha_t(i)$ and $\beta_t(i)$ are computed in a recursive manner, which is referred to as the forward process and backward process [99]. Given $\alpha_t(i)$ and $\beta_t(i)$, $P(x^T|\theta)$ can be computed via Eq.(2.10),

$$P(x^T|\theta) = \sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} \beta_1(i) b_i(o_1) \tag{2.10}$$

Given an observation sequence, the optimal state sequence s^T is revealed using the Viterbi searching algorithm [99][100]. The key issue of HMM is the training strategy, which is the process of updating the parameters of an HMM to optimize the performance that is measured by certain objective function. The most popular objective function adopted in the training strategies is the one given in Eq.(2.11).

$$\theta_{ML} = \arg\max_{\theta}[P(x^T|\theta)] \tag{2.11}$$

θ_{ML} is the maximum likelihood (ML) estimation as it has the largest likelihood of generating the observation x^T. Eq.(2.11) leads to the well-known Baum-Welch estimation, which is detailed in [98][99].

For the Baum-Welch estimation, the goodness-of-fit of the HMM has much to do with the initial values. For the discrete HMM, parameters in Matrix A can take arbitrary or uniform values subject to the probability constraints given below.

$$\sum_{j=1}^{N} a_{ij} = 1 \quad \text{and} \quad a_{ij} \geq 0, \quad i, j = 1, 2, \cdots, N \tag{2.12}$$

The initial values of the elements in Matrix B can also take arbitrary or uniform non-negative values as they satisfy Eq.(2.13).

$$\sum_{m=1}^{M} b_{jm} = 1 \quad \text{and} \quad b_{jm} \geq 0, \quad j = 1, 2, \cdots, N, m = 1, 2, \cdots, M \tag{2.13}$$

In specific identification task, however, the parameters of an HMM should be set according to the statistical/temporal features of the training samples. In Section 2.4.2, parameters of Matrices A and B are configured in a special manner so that the states of the HMM are segmented and physically associated with the phases of visemes production. In continuous case, such initialization is essential for the performance of the trained HMM. The detailed initialization steps will also be discussed in Section 2.4.2.

The decision about the identity of an input sequence is made by comparing the probabilities scored by different HMMs. In a K-class recognition problem, assume that $d_1, d_2, \cdots, d_K$ are class labels and $\theta_1, \theta_2, \cdots, \theta_K$ are HMMs trained for the

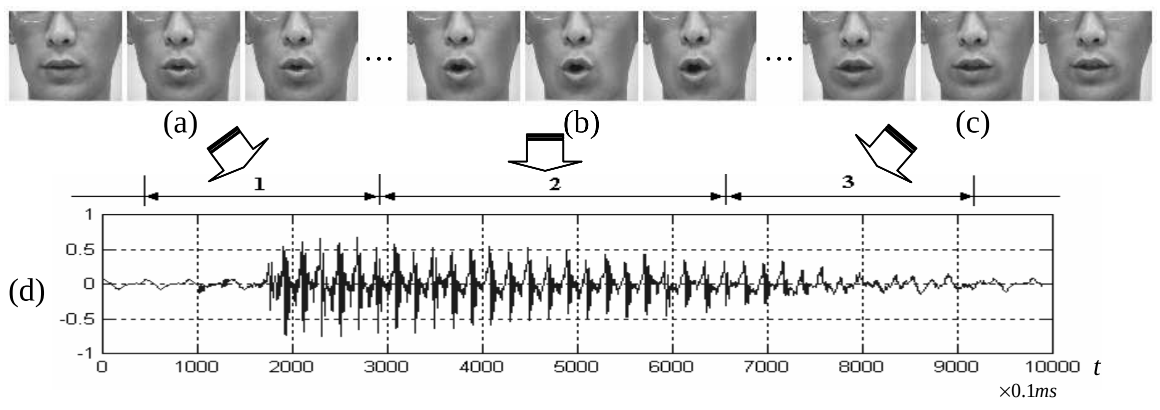

Figure 2.8: The three phases of viseme production
(a) initial phase (b) articulation phase (c) end phase (d) waveform of the sound produced

K classes. For an unknown input y^T, the probabilities scored by the K HMMs are compared with one another. The one that gives the maximum probability is chosen as the identity of y^T. Mathematically,

$$ID(y^T) = \arg\max_k P(y^T|\theta_k), \quad 1 \le k \le K \tag{2.14}$$

2.4.2 Configuration of the viseme models

Both discrete HMMs and continuous HMMs are applied to model and identify the visemes. Because each viseme is modeled by one HMM, the viseme models are also referred to as single-HMM classifiers or ML HMM classifiers. For our experiments, the motion of the lips is partitioned into three phases during viseme production: the first is initial phase, starting from a closed mouth or the end of the last viseme production to the beginning of sound production of the viseme investigated. During this interval, the sound of the target viseme is not fully produced and the lips are characterized with sharp changes. The next phase is the articulation phase, which is the time when sound is produced. The change of the lip shape is not so abrupt

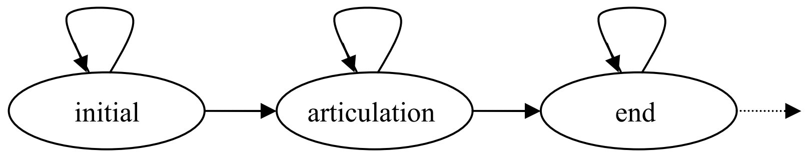

Figure 2.9: The three-state left-right viseme model

during this phase compared with the change in the other phases, and there is usually a short stable moment. The third phase is the end phase when the mouth restores to the relaxed state or transits to the production of the next viseme. The three phases and the corresponding acoustic waveform are depicted in Fig.2.8 when the phonetic sound /o/ is uttered. Among the three phases, the articulation phase is the most important for recognition because the difference among visemes chiefly lies there and it is relatively independent to the context. The initial phase and end phase are transitional phases and may be greatly distorted by their contexts. To align the states of an HMM with the phases of viseme production, three-state left-right HMM as shown in Fig.2.9 is adopted as the viseme model.

With this frame, the state transition matrix A has the form of Eq.(2.15),

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15}$$

where the 4th state, S_4, is a null state indicating the end of the viseme production. Given the video clip of a viseme, the approximate initial phase, articulation phase and end phase are manually segmented from the image sequence and the acoustic waveform, which is exemplified in Fig.2.8. The number of the frames (durations) of the three phases are also counted. If the duration of the i-th phase is T_i, the initial value of $a_{i,i}$ is $\frac{T}{T_i+1}$ and the initial value of $a_{i,i+1}$ is $\frac{1}{T_i+1}$ as they maximize the probability $a_{i,i}^{T_l} a_{i,i+1}$. In discrete case, the parameters in Matrix B are also

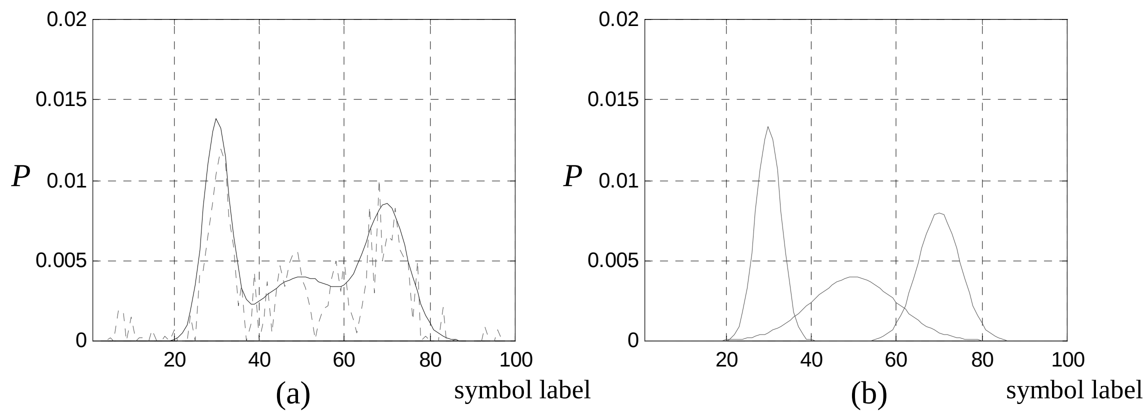

Figure 2.10: (a) Gaussian mixtures are matched against the actual symbol output pdf of State S_i. (b) The matched Gaussian mixtures

initialized according to the number of the frames of each phase. For example, if symbol O_j appears $T(O_j)$ times in the i-th phase, the initial value of b_{ij} is $\frac{T(O_j)}{T_i}$. In continuous case, the symbol output pdf of certain state is obtained by averaging the observed symbols within the state. The Gaussian mixtures (smooth contour) are matched against the actual symbol output pdf (jagged contour) as depicted in Fig.2.10. This process is carried out in a similar manner as that of the discrete case mentioned above. It is easy to prove that with such arrangement, the i-th phase of the viseme production is physically associated with State S_i of the HMM. The three states of the HMM are therefore referred to as initial state, articulation state and end state.

2.4.3 Training of the viseme classifiers

The 140 context-independent samples and 200 context-dependent samples of a viseme as mentioned in Section 2.2 are divided into two groups: 40 context-independent samples or 100 context-dependent samples are used for training the HMMs and the left 100 samples are applied to test the performance of the obtained

HMMs. For each viseme, four kinds of single-HMM classifiers with different initial settings are trained using the Baum-Welch algorithm.

Classifier 1 (θ^1): The symbol set of θ^1 is discrete, say O^M. The training samples and testing samples are also encoded by O^M. Matrix A of θ^1 is initialized according to the procedures given in Section 2.4.2 while the elements in Matrix B are initialized with uniform values, i.e.

$$b_{ij} = \frac{1}{M}, \quad i = 1, 2, \cdots, N, j = 1, 2, \cdots, M \tag{2.16}$$

Classifier 2 (θ^2): θ^2 is also a discrete HMM as θ^1 while both Matrix A and Matrix B are configured according to the procedures given in Section 2.4.2.

Classifier 3 (θ^3): The symbol set of θ^3 is continuous. For each state, only one Gaussian mixture is applied to model its symbol output pdf. Matrices A and B of θ^3 are configured according to the procedures mentioned in Section 2.4.2.

Classifier 4 (θ^4): θ^4 is also a continuous HMM as θ^3. However, the symbol output pdf of each state is modeled by three or four Gaussian mixtures. Matrices A and B of θ^4 are also configured according to the procedures given in Section 2.4.2.

After implementing the Baum-Welch training, the HMMs are put through parameter smoothing to prevent the occurrence of zero probability while measuring an input sequence. Parameter smoothing is the simple management that b_{ij} is set to be equal to some minimum value, e.g. $\epsilon = 10^{-3}$, if the estimated conditional probability $b_{ij} = 0$ [99]. In this way, even though symbol O_j never appears in the training samples, there is still a non-zero probability of its occurrence in the HMM when scoring an unknown observation.

The performance of a single-HMM classifier is evaluated by its recognition rate. By counting the number of correct and incorrect classifications of the testing samples, the recognition rate of a classifier, say θ_i, is computed via Eq.(2.17).

$$R_r(\theta_i) = \frac{\text{number of the correctly identified samples of Class } d_i}{\text{number of the testing samples of Class } d_i} \tag{2.17}$$

The recognition rate is associated with the false rejection rate (FRR) by Eq.(2.18).

$$FRR(\theta_i) = 1 - R_r(\theta_i) \tag{2.18}$$

FRR is also referred to as Type II error. It indicates the samples of Class d_i being misclassified into some wrong category $d_j (j \neq i)$. Another source of error indicates the samples of other classes $d_j (j \neq i)$ are erroneously accepted by θ_i. This portion of error is referred to as false acceptance rate (FAR) or Type I error. However, FAR is not an important performance indicator of the HMM classifiers proposed in this thesis (the reason will be given in Section 4.3.5). As a result, the recognition rate computed via Eq.(2.18), which is directly associated with FRR, is computed for the viseme classifiers in this section.

2.4.4 Experimental results

Experiments of viseme recognition are conducted on two speakers to verify the performance of the single-HMM classifiers $\theta^1, \theta^2, \theta^3$ and θ^4. Speaker 1 is a female native English speaker and Speaker 2 is a male non-native English speaker. In the first experiment, the training data and testing data are context-independent samples. The recognition rates of the 100 testing samples are given in Table 2.2. For the context-independent samples, the average classification accuracy is usually above 60%, especially for the vowels, where the average accuracy is nearly 90%. Both continuous HMM classifiers and discrete HMM classifiers can well identify the target visemes. In discrete case, the recognition rate of θ^2 is normally higher than that of θ^1. It indicates that the configuration of HMM based on the three phases of viseme production may improve the performance of an HMM classifier. In continuous case, the HMMs with multiple Gaussian mixtures (θ^4) provide better accuracy than those with only one mixture (θ^3). It manifests that a complex symbol output pdf (composed by several mixtures) may also improve the performance of

Table 2.2: Recognition rates of the context-independent viseme samples

Viseme Number	Speaker 1				Speaker 2			
	θ^1	θ^2	θ^3	θ^4	θ^1	θ^2	θ^3	θ^4
1 p, b, m	65%	87%	54%	82%	55%	63%	60%	72%
2 f, v	90%	96%	87%	82%	76%	72%	43%	67%
3 T, D	48%	89%	56%	85%	72%	81%	55%	68%
4 t, d	70%	85%	71%	74%	68%	66%	69%	78%
5 k, g	73%	85%	71%	72%	54%	60%	62%	74%
6 tS, dZ, S	87%	90%	76%	92%	73%	75%	62%	63%
7 s, z	94%	96%	80%	98%	75%	81%	88%	85%
8 n, l	61%	81%	69%	82%	37%	46%	15%	50%
9 r	59%	82%	24%	85%	26%	36%	8%	16%
10 A:	84%	99%	80%	89%	78%	78%	47%	79%
11 e	87%	92%	74%	88%	58%	70%	66%	73%
12 I	92%	99%	93%	90%	89%	90%	74%	75%
13 Q	91%	93%	93%	96%	89%	89%	79%	88%
14 U	97%	93%	69%	96%	91%	93%	83%	89%

the HMM classifier.

In the second experiment, the training data are 40 context-dependent samples and the testing data are the remaining 100 context-dependent samples of a viseme. The recognition rates of the four HMMs of the two speakers are given in Table 2.3.

It is observed the recognition rates of identifying context-dependent samples are much lower than that of the context-independent samples as depicted in Table 2.2. For the vowels, the average accuracy is below 60% for Speaker 2. For the consonants, the accuracy is discouragingly low, e.g. less than 30%. Although the recognition accuracy can be raised by selecting better initial values for the HMM, the improvement is limited. For further improvement of the ability of the HMM classifiers, especially to identify visemes under various contexts, the use of novel

Table 2.3: Recognition rates of the context-dependent viseme samples

Viseme Number	Speaker 1				Speaker 2			
	θ^1	θ^2	θ^3	θ^4	θ^1	θ^2	θ^3	θ^4
1 p, b, m	65%	80%	72%	70%	21%	29%	30%	43%
2 f, v	54%	73%	47%	58%	23%	32%	33%	42%
3 T, D	38%	66%	30%	56%	19%	35%	18%	30%
4 t, d	52%	50%	29%	30%	3%	10%	10%	14%
5 k, g	66%	83%	54%	60%	7%	10%	15%	4%
6 tS, dZ, S	72%	79%	70%	80%	27%	25%	33%	60%
7 s, z	61%	55%	66%	68%	56%	51%	28%	25%
8 n, l	22%	21%	7%	9%	5%	32%	0%	31%
9 r	25%	44%	12%	19%	5%	14%	0%	0%
10 A:	71%	78%	66%	63%	58%	70%	46%	66%
11 e	53%	66%	43%	42%	19%	48%	56%	69%
12 I	80%	90%	85%	88%	66%	66%	68%	71%
13 Q	79%	65%	55%	79%	69%	62%	49%	55%
14 U	78%	91%	60%	80%	62%	69%	73%	77%

training strategy or complex HMM classifier would be explored.

Chapter 3

Discriminative Training of HMM Based on Separable Distance

The Baum-Welch training strategy mentioned in Chapter 2 optimizes an HMM to maximize the *a priori* probability $P(x^T|\theta)$. The trained HMM may stand better chance of generating the true samples of the stochastic process investigated. In this chapter, the performance of an HMM is evaluated by its discriminative power. An HMM (θ_1) is assumed "better" than another HMM (θ_2) if θ_1 can separate the true samples of a class out of the samples of another class with higher credibility than θ_2 does. To improve the discriminative power of an HMM, two training strategies, which are based on a novel separable distance function, are proposed.

3.1 Separable distance

In the K-class identification problem mentioned in Section 2.4.1, decision about the identity of an input sequence is made by comparing the probabilities scored by the K HMMs. If the samples of different class are confusable, the HMMs must have good discriminative power so as to separate them. The maximum mutual information (MMI) estimation is a popular discriminative training strategy for HMM. It leads to the largest *a posteriori* probability corresponding to the training data and thus has good overall discriminative power[102]. However, the MMI estimation is difficult to implement as the analytical solutions to its criterion function are difficult to realize [102].

To improve the discriminative ability of HMM over confusable samples, and at the same time, facilitate the training of the HMM, a new metric is proposed to measure the difference between the samples given an HMM. Assume that in a two-class identification problem, $\{x_1^T : d_1\}$ and $\{x_2^T : d_2\}$ are a pair of training samples, where $x_1^T = (o_1^1, o_2^1, \cdots, o_T^1)$ and $x_2^T = (o_1^2, o_2^2, \cdots, o_T^2)$ are T-length observation sequences and d_1 and d_2 are their identity labels. The observed symbols in x_1^T and x_2^T are encoded by the symbol set O^M. $P(x_1^T|\theta)$ and $P(x_2^T|\theta)$ are the scored likelihood for x_1^T and x_2^T given θ, respectively. The pair of training samples x_1^T and x_2^T must be of the same length so that their probabilities $P(x_1^T|\theta)$ and $P(x_2^T|\theta)$ can be suitably compared (because the shorter sequence may give larger probability than the longer one even if it is not the true sample of θ). Define a new function $I(x_1^T, x_2^T, \theta)$, called the separable-distance function, as follows.

$$I(x_1^T, x_2^T, \theta) = \log P(x_1^T|\theta) - \log P(x_2^T|\theta) \tag{3.1}$$

A large value of $I(x_1^T, x_2^T, \theta)$ would mean that x_1^T and x_2^T are more distinct and separable. The strategy then is to determine the HMM θ_{MSD} (MSD for maximum separable-distance) that maximizes $I(x_1^T, x_2^T, \theta)$. Mathematically,

$$\theta_{MSD} = \arg\max_{\theta} I(x_1^T, x_2^T, \theta) \tag{3.2}$$

For the proposed training strategy, the parameters in matrix B are adjusted to maximize the separable-distance while matrix A and π are kept unchanged. With this assumption and if only the probability constraint $\sum_{j=1}^{M} b_{ij} = 1 (i = 1, 2, \cdots, N)$

is considered, maximizing Eq.(3.1) is equivalent to maximizing the auxiliary function $F(x_1^T, x_2^T, \theta, \lambda)$, which is defined in Eq.(3.3),

$$F(x_1^T, x_2^T, \theta, \lambda) = I(x_1^T, x_2^T, \theta) + \sum_{i=1}^{N} \lambda_i (1 - \sum_{j=1}^{M} b_{ij}) \tag{3.3}$$

where λ_i is the Lagrange multiplier for State S_i. Differentiate $F(x_1^T, x_2^T, \theta, \lambda)$ with respect to b_{ij} and set the result to 0, we have,

$$\frac{\partial \log P(x_1^T|\theta)}{\partial b_{ij}} - \frac{\partial \log P(x_2^T|\theta)}{\partial b_{ij}} = \lambda_i \tag{3.4}$$

Since λ_i is positive, the optimum value obtained for $I(x_1^T, x_2^T, \theta)$ is a maximum as solutions for b_{ij} must be positive. In Eq.(3.4), $\log P(x_1^T|\theta)$ and $\log P(x_2^T|\theta)$ may be computed by summing up all the probabilities over time T as follows.

$$\log P(x_1^T|\theta) = \sum_{\tau=1}^{T} \log \sum_{i=1}^{N} P(s_\tau^T = S_i) b_i(o_\tau^1) \tag{3.5}$$

where s_τ^T is the τ-th state of the state sequence decoded for x_1^T. Note that the state transition coefficients a_{ij} do not appear explicitly in Eq.(3.5), they are included in the term $P(s_\tau^T = S_i)$. The partial derivatives in Eq.(3.4) may be evaluated separately as follows,

$$\frac{\log P(x_1^T|\theta)}{\partial b_{ij}} = \sum_{\tau=1, o_\tau^1 = O_j}^{T} P(s_\tau^T = S_i|\theta, x_1^T) = \sum_{\tau=1}^{T} \frac{P(s_\tau^T = S_i, o_\tau^1 = O_j|\theta, x_1^T)}{b_{ij}} \tag{3.6}$$

By defining

$$E(S_i, O_j|\theta, x_1^T) = \sum_{\tau=1}^{T} P(s_\tau^T = S_i, o_\tau^1 = O_j|\theta, x_1^T) \tag{3.7}$$

$$D_{ij}(x_1^T, x_2^T, \theta) = E(S_i, O_j|\theta, x_1^T) - E(S_i, O_j|\theta, x_2^T) \tag{3.8}$$

where $E(S_i, O_j|\theta, x_2^T)$ is obtained in the same way as in Eq.3.7, we have,

$$\frac{E(S_i, O_j|\theta, x_1^T) - E(S_i, O_j|\theta, x_2^T)}{b_{ij}} = \frac{D_{ij}(x_1^T, x_2^T, \theta)}{b_{ij}} = \lambda_i, \quad 1 \le j \le M \tag{3.9}$$

By making use of the fact that $\sum_{j=1}^{M} b_{ij} = 1$, it can be shown that

$$b_{ij} = \frac{D_{ij}(x_1^T, x_2^T, \theta)}{\sum_{j=1}^{M} D_{ij}(x_1^T, x_2^T, \theta)} \tag{3.10}$$

The set $\{b_{ij}\}(i = 1, 2, \cdots, N, j = 1, 2, \cdots, M)$ so obtained gives the maximum value of $I(x_1^T, x_2^T, \theta)$.

An algorithm for the computation of the values may be developed by using standard Expectation-Maximization (EM) technique [101]. By considering x_1^T and x_2^T as the observed data and the state sequence $s^T = (s_1^T, s_2^T, \cdots, s_T^T)$ as the hidden or unobserved data, the estimation of $E_\theta(I) = E[I(x_1^T, x_2^T, s^T|\tilde{\theta})|x_1^T, x_2^T, \theta]$ from incomplete data x_1^T and x_2^T is then given by Eq.(3.11):

$$\begin{aligned} E_\theta(I) &= \sum_{s^T \in S} I(x_1^T, x_2^T, s^T|\tilde{\theta}) P(x_1^T, x_2^T, s^T|\theta) \\ &= \sum_{s^T \in S} [\log P(x_1^T, s^T|\tilde{\theta}) - \log P(x_2^T, s^T|\tilde{\theta})] P(x_1^T, x_2^T, s^T|\theta) \end{aligned} \tag{3.11}$$

where θ and $\tilde{\theta}$ are the HMM before training and the HMM after training respectively, and S denotes all the state combinations with length T. The purpose of the E-step of the EM estimation is to calculate $E_\theta(I)$. By using the auxiliary function $Q_x(\tilde{\theta}, \theta)$ proposed in [102] and defined as follows,

$$Q_x(\tilde{\theta}, \theta) = \sum_{s^T \in S} \log P(x_1^T, s^T|\tilde{\theta}) P(x_1^T, s^T|\theta) \tag{3.12}$$

Eq.(3.11) can be written as

$$E_\theta(I) = Q_x(\tilde{\theta}, \theta) P(x_2^T|s^T, \theta) - Q_y(\tilde{\theta}, \theta) P(x_1^T|s^T, \theta) \tag{3.13}$$

$Q_x(\tilde{\theta}, \theta)$ and $Q_y(\tilde{\theta}, \theta)$ may be further analyzed by breaking up the probability $P(x_1^T, s^T|\theta)$ as in Eq.(3.14) below,

$$P(x_1^T, s^T|\tilde{\theta}) = \tilde{\pi}(s_0) \prod_{\tau=1}^{T} \tilde{a}_{s_{\tau-1}, s_\tau} \tilde{b}_{s_\tau}(o_\tau^1) \tag{3.14}$$

where $\tilde{\pi}$, $\tilde{a}$ and $\tilde{b}$ are parameters of $\tilde{\theta}$. Here, we assume that the initial distribution starts at $\tau = 0$ instead of $\tau = 1$ for notational convenience. $Q_x(\tilde{\theta}, \theta)$ then becomes

$$Q_x(\tilde{\theta}, \theta) = P(x_1^T, s^T|\theta) \sum_{s^T \in S} [\log \tilde{\pi}(s_0) + \sum_{\tau=1}^{T} \log \tilde{a}_{\tau-1,\tau} + \sum_{\tau=1}^{T} \log \tilde{b}_\tau(o_\tau^1)] \quad (3.15)$$

The parameters to be optimized are now separated into three independent terms. From Eq.(3.13) and (3.15), $E_\theta(I)$ can also be divided into the following three terms,

$$E_\theta(I) = E_\theta(\tilde{\pi}, I) + E_\theta(\tilde{a}, I) + E_\theta(\tilde{b}, I) \quad (3.16)$$

where

$$E_\theta(\tilde{\pi}, I) = \sum_{s^T \in S} \log \tilde{\pi}(s_0)[P(x_1^T, x_2^T, s^T|\theta) - P(x_1^T, x_2^T, s^T|\theta)] = 0 \quad (3.17)$$

$$E_\theta(\tilde{a}, I) = \sum_{s^T \in S} \sum_{\tau=1}^{T} \log \tilde{a}_{\tau-1,\tau}[P(x_1^T, x_2^T, s^T|\theta) - P(x_1^T, x_2^T, s^T|\theta)] = 0 \quad (3.18)$$

$$E_\theta(\tilde{b}, I) = \sum_{s^T \in S} [\sum_{\tau=1}^{T} \log \tilde{b}_\tau(o_\tau^1) - \sum_{\tau=1}^{T} \log \tilde{b}_\tau(o_\tau^2)] P(x_1^T, x_2^T, s^T|\theta) \quad (3.19)$$

$E_\theta(\tilde{\pi}, I)$ and $E_\theta(\tilde{a}, I)$ are associated with the hidden state sequence s^T. It is assumed that x_1^T and x_2^T are drawn independently and emitted from the same state sequence s^T, hence both $E_\theta(\tilde{\pi}, I)$ and $E_\theta(\tilde{a}, I)$ become 0. $E_\theta(\tilde{b}, I)$, on the other hand, is related to the symbols that appear in x_1^T and x_2^T and contributes to $E_\theta(I)$. By enumerating all the state combinations, we have,

$$E_\theta(\tilde{b}, I) = \sum_{i=1}^{N} \sum_{\tau=1}^{T} [\log \tilde{b}_\tau(o_\tau^1) - \log \tilde{b}_\tau(o_\tau^2)] P(x_1^T, x_2^T, s_\tau = S_i|\theta) \quad (3.20)$$

If $\sum_{\tau=1}^{T} [\log \tilde{b}_\tau(o_\tau^1) - \log \tilde{b}_\tau(o_\tau^2)]$ is arranged according to the order of appearance of the symbols (O_j) within x_1^T and x_2^T, we have,

$$E_\theta(\tilde{b}, I) = \sum_{i=1}^{N} \sum_{j=1}^{M} \log \tilde{b}_{ij}[E(S_i, O_j|\theta, x_1^T) - E(S_i, O_j|\theta, x_2^T)] P(x_1^T, x_2^T|\theta) \quad (3.21)$$

where $E(S_i, O_j|\theta, x_1^T) = \sum_{\tau=1, s.t. o_\tau^1 = o_\tau^2 = O_j}^{T} P(o_\tau^1 = O_j, s_\tau = S_i|\theta, x_1^T)$. In the M-step of the EM estimation, b_{ij} is adjusted to maximize $E_\theta(\tilde{b}, I)$ or $E_\theta(I)$. Since $\sum \tilde{b}_{ij} = 1$ and Eq.(3.21) has the form $K\sum_{j=1}^{M} w_j \log v_j$, which attains a global maximum at the point $v_j = \frac{w_j}{\sum_{j=1}^{M} w_j} (j = 1, 2, \cdots, M)$, the re-estimated value of $\tilde{b}_{ij}$ of $\tilde{\theta}$ that lead to the maximum $E_\theta(I)$ is given by

$$\tilde{b}_{ij} = \frac{E(S_i, O_j|\theta, x_1^T) - E(S_i, O_j|\theta, x_2^T)}{\sum_{j=1}^{M}[E(S_i, O_j|\theta, x_1^T) - E(S_i, O_j|\theta, x_2^T)]} \tag{3.22}$$

This equation, compared with Eq.(3.10), enables the re-estimation of the symbol emission coefficients $\tilde{b}_{ij}$ from expectations of the existing HMM. The above derivations strictly observe the standard optimization strategy [101], where the expectation of the value of the separable-distance function, $E_\theta(I)$, is computed in the E-step and the coefficients b_{ij} are adjusted to maximize $E_\theta(I)$ in the M-step. The convergence of the method is therefore guaranteed. However, b_{ij} may not be estimated by applying Eq.(3.22) alone, other considerations shall be taken into account such as when $D_{ij}(x^T, y^T, \theta)$ is less than or equal to 0. Further discussion on the determination of values of b_{ij} is given in the subsequent sections.

3.2 Two-channel discriminative training

The training strategy discussed in this section is essentially an HMM structure that the symbol output matrix of the HMM is split into two channels (sub-matrices): a static channel to maintain the validity of the HMM and a dynamic channel that is modified to maximize the separable distance. For the specific application to viseme recognition, a hierarchical identification system is designed as depicted in Fig.3.1. For the K visemes to be recognized, R (usually $R < K$) ML HMM classifiers are employed for preliminary recognition. The output of the preliminary recognition is a coarse identity, which may include L (usually $1 < L < K$) viseme classes. Fine recognition is then performed using a bank of two-channel HMMs.

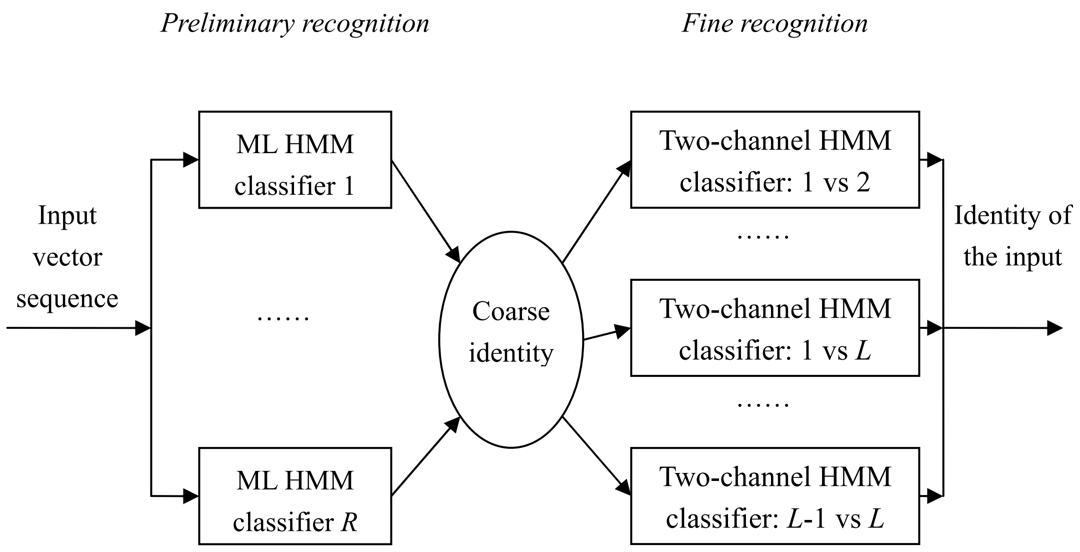

Figure 3.1: System architecture

3.2.1 Structure of the two-channel HMM

To modify the parameters according to Eq.(3.22) and simultaneously ensure the validity of the model, a two-channel structure as depicted in Fig.3.2 is proposed. The elements (b_{ij}) of Matrix B of the HMM are decomposed into two parts as

$$b_{ij} = b^s_{ij} + b^d_{ij}, \quad \forall i = 1, 2, \cdots, N, j = 1, 2, \cdots, M \tag{3.23}$$

b^s_{ij} for the static-channel and b^d_{ij} for the dynamic-channel. The symbol output coefficients of State S_i in Fig.3.2, $\{b_{i1}\ b_{i2}\ \cdots\ b_{iM}\}$, are decomposed as in Eq.(3.23).

$$\{b_{i1}\ b_{i2}\ \cdots\ b_{iM}\} = \underbrace{\{b^s_{i1}\ b^s_{i2}\ \cdots\ b^s_{iM}\}}_{\text{static channel}} + \underbrace{\{b^d_{i1}\ b^d_{i2}\ \cdots\ b^d_{iM}\}}_{\text{dynamic channel}} \tag{3.24}$$

The dynamic-channel coefficients b^d_{ij} are the key source of the discriminative power and is estimated via Eq.(3.22) while the static-channel coefficients b^s_{ij} are computed using parameter-smoothed ML HMM and weighted. To avoid the occurrence of zero or negative probability, b^s_{ij} should be kept greater than 0 and at the same time,

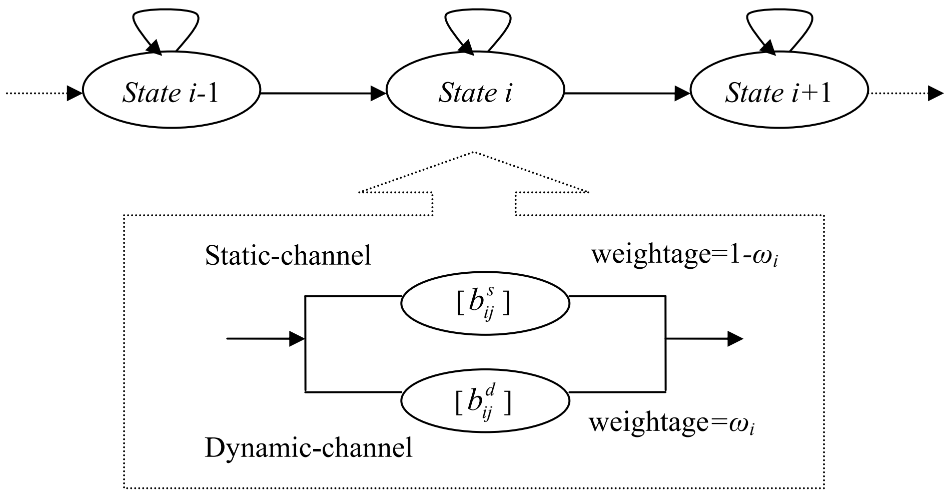

Figure 3.2: The two-channel structure of the i-th state of a left-right HMM

the dynamic-channel coefficient b_{ij}^d should be non-negative. Thus the probability constraint $b_{ij} = b_{ij}^s + b_{ij}^d \geq b_{ij}^s > 0$ is met. In addition, the relative weightage of the static-channel and the dynamic-channel may be controlled by the credibility weighing factor ω_i. The scaling of the coefficients in dynamic-channel and static-channel are subject to the weightage constraint given in Eq.(3.25).

$$\sum_{j=1}^{N} b_{ij}^d = \omega_i \quad \text{or} \quad \sum_{j=1}^{N} b_{ij}^s = 1-\omega_i, \quad 0 \leq \omega_i < 1, \forall i = 1, 2, \cdots, N \quad (3.25)$$

3.2.2 Step 1: Parameter initialization

The parameter-smoothed ML HMM of x_1^T, $\tilde{\theta}_{ML}^1$, which is trained using the Baum-Welch estimation, is referred to as the base HMM. b_{ij}^s of the static-channel HMM is derived from the base HMM after applying the scaling factor as follows.

$$\{b_{i1}^s \; b_{i2}^s \; \cdots \; b_{iM}^s\} = (1-\omega_i)\{\tilde{b}_{i1} \; \tilde{b}_{i2} \; \cdots \; \tilde{b}_{iM}\}, \quad i = 1, 2, \cdots, N, 0 \leq \omega_i < 1 \quad (3.26)$$

where $\tilde{b}_{ij}$ is the symbol output probability of $\tilde{\theta}_{ML}^1$. As for the dynamic-channel coefficients b_{ij}^d, uniform values equal to ω_i/M are assigned to b_{ij}^ds as initial values.

The selection of credibility weighing factor of the i-th state, ω_i, is flexible and problem-dependent. A large value of ω_i means large weightage is assigned to the dynamic-channel and the discriminative power is enhanced. However, the probability of the correct observation $P(x_1^T|\theta)$ will normally decrease. It is undesirable because the HMM obtained is unlikely to generate even the correct samples.

A guideline for the determination of the value of ω_i is as follows. Given the base HMM $\tilde{\theta}_{ML}^1$, the optimal state chains are searched for x_1^T and x_2^T using the Viterbi algorithm [99]. If $\tilde{\theta}_{ML}^1$ is a left-right model and the expected (optimal) duration of State S_i of x_1^T is from t_i to $t_i+\tau_i$, $P(x_1^T|\tilde{\theta}_{ML}^1)$ is then written as Eq.(3.27).

$$P(x_1^T|\tilde{\theta}_{ML}^1) = \prod_{i=1}^{N} P(o_{t_i}^1, o_{t_i+1}^1, \cdots, o_{t_i+\tau_i}^1|\tilde{\theta}_{ML}^1) \tag{3.27}$$

Let $P_{dur}(x_1^T, S_i, \tilde{\theta}_{ML}^1) = P(o_{t_i}^1, o_{t_i+1}^1, \cdots, o_{t_i+\tau_i}^1|\tilde{\theta}_{ML}^1)$. It may be computed using the forward variables $\alpha_t^1(i)$ and backward variables $\beta_t^1(i)$ [99]. In case that $\tilde{\theta}_{ML}^1$ is not a left-right model but an ergodic model, the expected duration of a state will be the multiplication of a number of separated time slices as manifested in Eq.(3.28),

$$P_{dur}(x_1^T, S_i, \tilde{\theta}_{ML}^1) = \prod_{j=1}^{k} P(o_{t_{ij}}^1, o_{t_{ij}+1}^1, \cdots, o_{t_{ij}+\tau_{ij}}^1|\tilde{\theta}_{ML}^1) \tag{3.28}$$

where the jth slice is from t_{ij} to $t_{ij}+\tau_{ij}$. The value of ω_i is derived by comparing the corresponding $P_{dur}(x_1^T, S_i, \tilde{\theta}_{ML}^1)$ and $P_{dur}(x_2^T, S_i, \tilde{\theta}_{ML}^1)$. If $P_{dur}(x_1^T, S_i, \tilde{\theta}_{ML}^1) \gg P_{dur}(x_2^T, S_i, \tilde{\theta}_{ML}^1)$, ω_i should be set to a small value to preserve the original ML configurations. If $P_{dur}(x_1^T, S_i, \tilde{\theta}_{ML}^1) < P_{dur}(x_2^T, S_i, \tilde{\theta}_{ML}^1)$ or $P_{dur}(x_1^T, S_i, \tilde{\theta}_{ML}^1) \approx P_{dur}(x_2^T, S_i, \tilde{\theta}_{ML}^1)$, ω_i must be set to a large value. In practice, ω_i can be manually selected according to the conditions mentioned above (which is preferred), or they can be computed using the following expression.

$$\omega_i = \frac{1}{1+Cv^D} \tag{3.29}$$

where $v = \frac{P_{dur}(x_1^T, S_i, \tilde{\theta}_{ML}^1)}{P_{dur}(x_2^T, S_i, \tilde{\theta}_{ML}^1)}$. $C (C > 0)$ and D are constants that jointly control the smoothness of ω_i with respect to v. For example, if the range of v is $10^{-3} \sim 10^5$, a typical setting is $C = 1.0$ and $D = 0.1$. Once the values of ω_i are determined, they shall not be changed in the training process.

3.2.3 Step 2: Partition of the observation symbol set

Let θ denote the HMM with the above initial configurations. $E(S_i, O_j|\theta, x_1^T)$ is computed via the counting process [100]. Mathematically,

$$E(S_i, O_j|\theta, x_1^T) = \sum_{\tau=1, s.t. o_\tau^1=O_j}^{T} \sum_{j=1}^{N} \frac{\alpha_\tau^1(i)a_{ij}b_j(o_{\tau+1}^1)\beta_{\tau+1}^1(j)}{\sum_{m=1}^{N}\sum_{n=1}^{N}\alpha_\tau^1(m)a_{mn}b_n(o_{\tau+1}^1)\beta_{\tau+1}^1(n)} \tag{3.30}$$

$E(S_i, O_j|\theta, x_2^T)$ is computed in a similar manner. For certain symbol, e.g. O_p, the expectation $D_{ip}(x_1^T, x_2^T, \theta)$ may be less than 0. To find out these symbols, the symbol set O^M is partitioned into the subset $V = \{V_1, V_2, \cdots, V_K\}$ and its complement set $U = \{U_1, U_2, \cdots, U_{M-K}\}(O^M = U \cup V)$ according to Eq.(3.31).

$$\{V_1, V_2, \cdots, V_K\} = \arg_{O_j}[\frac{E(S_i, O_j|\theta, x_1^T)}{E(S_i, O_j|\theta, x_2^T)} > \eta], \quad (\eta \geq 1, \forall j = 1, 2, \cdots, M) \tag{3.31}$$

where η is the threshold. With $\eta \geq 1$, $E(S_i, O_j|\theta, x_1^T) - E(S_i, O_j|\theta, x_2^T) > 0$. As an illustration, the distributions of the values of $E(S_i, O_j|\theta, x_1^T)$ and $E(S_i, O_j|\theta, x_2^T)$ for different symbol labels are shown in Fig.3.3(a). The filtered symbols in set V when η is set 1 are shown in Fig.3.3(b).

3.2.4 Step 3: Modification to the dynamic-channel

The symbol set is partitioned for each state in Step 2. Consider State S_i, for symbols in the set U, the coefficient $b_i(U_j)(U_j \in U)$ should be set as small as possible. Let $b_i^d(U_j) = 0$, and so $b_i(U_j) = b_i^s(U_j)$. For symbols in the set V, the corresponding dynamic-channel coefficient $b_i^d(V_k)$ is computed according to Eq.(3.32), which is derived from Eq.(3.22).

$$b_i^d(V_k) = P_D(S_i, V_k, x_1^T, x_2^T)[\omega_i + \sum_{j=1}^{k} b_i^s(V_j)] - b_i^s(V_k), \quad k = 1, 2, \cdots, K \tag{3.32}$$

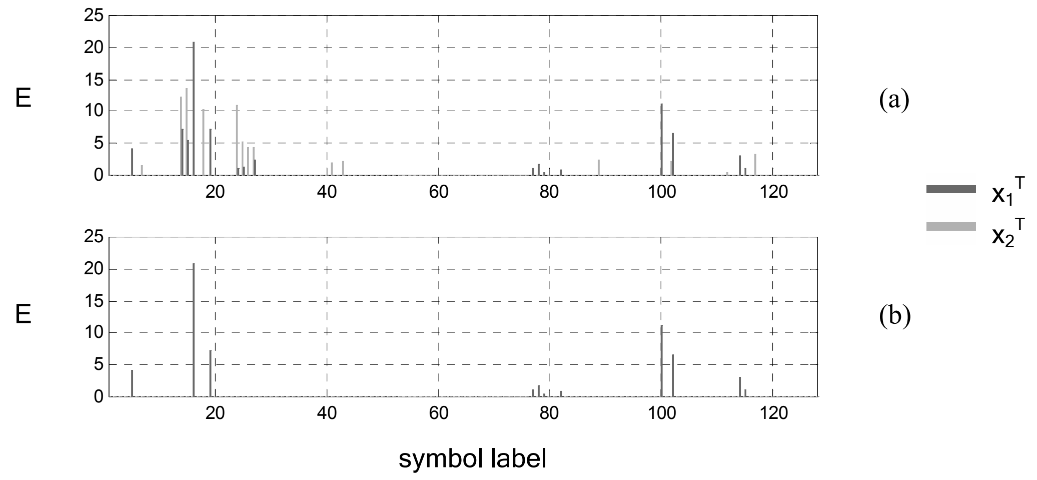

Figure 3.3: (a) Distributions of $E(S_i, O_j|\theta, x_1^T)$ and $E(S_i, O_j|\theta, x_2^T)$ for various symbols (b) Distribution of $E(S_i, O_j|\theta, x_1^T)$ for the symbols in V

where $P_D(S_i, V_k, x_1^T, x_2^T) = \frac{E(S_i,O_j|\theta,x_1^T)-E(S_i,O_j|\theta,x_2^T)}{\sum_{j=1}^{K}[E(S_i,O_j|\theta,x_1^T)-E(S_i,O_j|\theta,x_2^T)]}$. However, some coefficients obtained may still be negative, e.g. $b_i^d(V_l) < 0$ because of large value of $b_i^s(V_l)$. In this case, the symbol V_l is transferred from V to U and $b_i^d(V_l)$ is set to 0. The coefficients of the remaining symbols in V are re-estimated using Eq.(3.32) until all $b_i^d(V_k)$s are greater than 0. This situation (some $b_i^d(V_l) < 0$) usually happens at the first few epochs of training and it is not conducive to convergence because there is steep jump in the surface of $I(x_1^T, x_2^T, \theta)$. To relieve this problem, a larger value of η in Eq.(3.31) shall be used.

Optimization is done through repeating the training epoch described in Sections 3.2.3 and 3.2.4. After each epoch, the separable distance $I(x_1^T, x_2^T, \tilde{\theta})$ of the HMM $\tilde{\theta}$ obtained, is calculated and compared with that obtained in the last epoch. If $I(x_1^T, x_2^T, \tilde{\theta})$ does not change more than a predefined value, training is terminated and the target two-channel HMM is established.

3.3 Properties of the two-channel training

3.3.1 State alignment

One of the requirements for the proposed training strategy is that the state durations of the training pair, say x_1^T and x_2^T, are comparable. Otherwise, $D_{ij}(x_1^T, x_2^T, \theta)$ will become meaningless. For example, if $E(S_i|\theta, x_1^T) \ll E(S_i|\theta, x_2^T)$, even symbol O_j takes much greater portion in $E(S_i|\theta, x_1^T)$ than in $E(S_i|\theta, x_2^T)$, the computed $D_{ij}(x_1^T, x_2^T, \theta)$ may also be less than 0. The outcome is that b_{ij} is always set to b_{ij}^s rather than adjusted to increase $I(x_1^T, x_2^T, \theta)$. As a result, the following state duration validation procedure is added to make the training strategy complete. After each training epoch, $E(S_i|\theta, x_1^T)$ is obtained as follows.

$$E(S_i|\theta, x_1^T) = \sum_{\tau=1}^{T} \frac{\alpha_\tau^1(i)\beta_\tau^1(i)}{\sum_{i=1}^{N} \alpha_\tau^1(i)\beta_\tau^1(i)}, \quad i = 1, 2, \cdots, N \tag{3.33}$$

$E(S_i|\theta, x_2^T)$ is computed in the same way. If $E(S_i|\theta, x_1^T) \approx E(S_i|\theta, x_2^T)$, e.g. $1.2E(S_i|\theta, x_2^T) > E(S_i|\theta, x_1^T) > 0.8E(S_i|\theta, x_2^T)$, training continues; otherwise, training stops even $I(x_1^T, x_2^T, \tilde{\theta})$ keeps on increasing. If the $I(x_1^T, x_2^T, \tilde{\theta})$ of the final HMM $\tilde{\theta}$ does not meet certain discriminative requirement, i.e. $I(x_1^T, x_2^T, \tilde{\theta})$ is less than a desired value, a new base HMM or smaller ω_i should be used instead.

3.3.2 Speed of convergence

The convergence of the training strategy proposed in Eq.(3.22) is discussed in Section 3.1. For the two-channel training, only some of the symbol output coefficients in the dynamic-channel are modified according to Eq.(3.22) while the others remain unchanged. However, the convergence is still assured because firstly the surface of $I(x_1^T, x_2^T, \theta)$ with respect to b_{ij} is continuous, and also adjusting the dynamic-channel elements according to the two-channel training strategy leads to increased

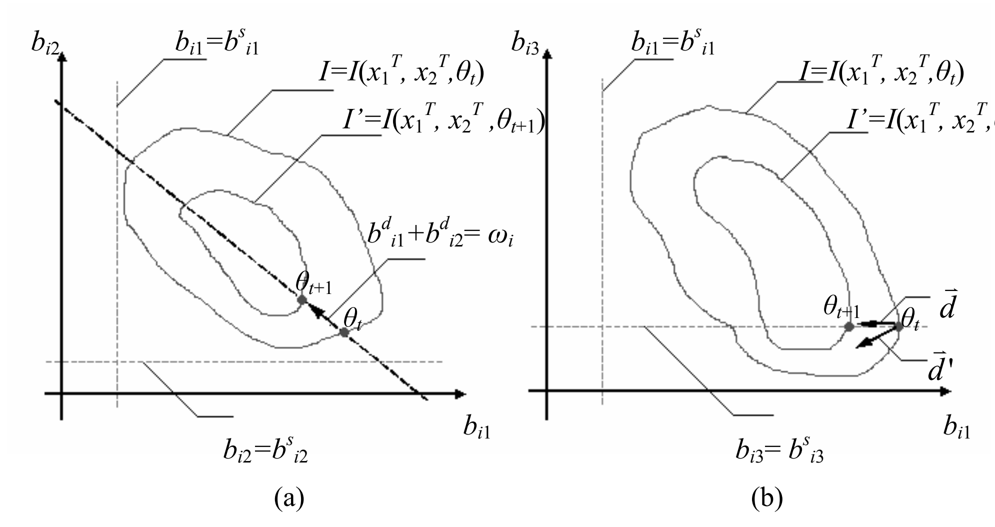

Figure 3.4: The surface of I and the direction of parameter adjustment.

$E_\theta(I)$. A conceptual illustration is given in Fig.3.4 on how b_{ij} is modified when the symbol set is divided into subsets V and U. For ease of explanation, we assume that the symbol set contains only three symbols O_1, O_2 and O_3 with $O_1, O_2 \in V$ and $O_3 \in U$ for State S_i. Let θ_t denote the HMM trained at the t-th round and θ_{t+1} denote the HMM obtained at the $t+1$-th round. The surface of the separable distance (I surface) is denoted as $I' = I(x_1^T, x_2^T, \theta_{t+1})$ for θ_{t+1} and $I = I(x_1^T, x_2^T, \theta_t)$ for θ_t. Clearly $I' > I$. The I surface is mapped to $b_{i1} - b_{i2}$ plane [Fig.3.4(a)] and $b_{i1} - b_{i3}$ plane [Fig.3.4(b)]. In the training phase, b_{i1} and b_{i2} are modified along the line $b^d_{i1} + b^d_{i2} = \omega_i$ to reach a better estimation θ_{t+1}, which is shown in Fig.3.4(a). In the $b_{i1} - b_{i3}$ plane, b_{i3} is set to the constant b^s_{i3} while b_{i1} is modified along the line $b_{i3} = b^s_{i3}$ with the direction $\vec{d}$ as shown in Fig. 3.4(b). The direction of parameter adjustment given by Eq.(3.22) is denoted by $\vec{d'}$. In the two-channel approach, since only b_{i1} and b_{i2} are modified according to Eq.(3.22) while b_{i3} remains unchanged, $\vec{d}$ may lead to lower speed of convergence than $\vec{d'}$ does.

3.3.3 Improvement to the discriminative power

The improvement to the discriminative power is estimated as follows. Assume that $\tilde{\theta}$ is the two-channel HMM obtained. The lower bound of $P(x_2^T|\tilde{\theta})$ is given by

$$P(x_2^T|\tilde{\theta}) \geq (1-\omega_{max})^T P(x_2^T|\tilde{\theta}_{ML}^1) \tag{3.34}$$

where $\omega_{max} = \max(\omega_1, \omega_2, \cdots, \omega_N)$. Because the base HMM is the parameter-smoothed ML HMM of x_1^T, it is reasonable to assume that $P(x_1^T|\tilde{\theta}_{ML}^1) \geq P(x_2^T|\tilde{\theta}_{ML}^1)$. The upper bound of the separable distance is given by the following expression

$$I(x_1^T, x_2^T, \tilde{\theta}) \leq \frac{P(x_1^T|\tilde{\theta}_{ML}^1)}{(1-\omega_{max})^T P(x_2^T|\tilde{\theta}_{ML}^1)} = -T\log(1-\omega_{max}) + I(x_1^T, x_2^T, \tilde{\theta}_{ML}^1) \tag{3.35}$$

In practice, the gain of $I(x_1^T, x_2^T, \tilde{\theta})$ is much smaller than the theoretical upper bound. It depends on the resemblance between x_1^T and x_2^T, and the setting of ω_i.

3.4 Extensions of the two-channel training algorithm

3.4.1 Training samples with different lengths

Up to this point, the training sequences are assumed to be of equal length. This is necessary as we cannot properly compare the probability scores of two sequences of different lengths. To extend the training strategy to sequences of different lengths, linear adjustment is carried out as follows. Given the training pair $x_1^{T_1}$ of length T_1 and $x_2^{T_2}$ of length T_2, the objective function Eq.(3.9) is modified as follows.

$$\frac{1}{b_{ij}}[\sum_{\tau=1}^{T_1} P(s_\tau^{T_1} = S_i, o_\tau^1 = O_j|\theta, x_1^{T_1}) - \frac{T_1}{T_2}\sum_{\tau=1}^{T_2} P(s_\tau^{T_2} = S_i, o_\tau^2 = O_j|\theta, x_2^{T_2})] = \lambda_i \tag{3.36}$$

Because $E(S_i, O_j|\theta, x_1^{T_1}) = \sum_{\tau=1}^{T_1} P(s_\tau^{T_1} = S_i, o_\tau^1 = O_j|\theta, x_1^{T_1})$, parameter estimation is then carried out as follows.

$$b_{ij} = \frac{E(S_i, O_j|\theta, x_1^{T_1}) - \frac{T_1}{T_2} E(S_i, O_j|\theta, x_2^{T_2})}{\sum_{j=1}^{M} [E(S_i, O_j|\theta, x_1^{T_1}) - \frac{T_1}{T_2} E(S_i, O_j|\theta, x_2^{T_2})]} \tag{3.37}$$

$E(S_i, O_j|\theta, x_2^{T_2})$ is normalized using the scale factor $\frac{T_1}{T_2}$. This approach is easy to implement, however, it does not consider the nonlinear variance of signal such as local stretch or squash. If the training sequences demonstrate obvious nonlinear variance, some nonlinear processing such as sequence truncation or symbol prune may be carried out to adjust the training sequences to the same length [103].

To verify the state durations in the training process, the duration of state S_i for $x_1^{T_1}$, $E(S_i|\theta, x_1^{T_1})$,is then compared with $\frac{T_1}{T_2} E(S_i|\theta, x_2^{T_2})$. If $E(S_i|\theta, x_1^{T_1}) \approx \frac{T_1}{T_2} E(S_i|\theta, x_2^{T_2})$, the training continues; otherwise, the training terminates.

3.4.2 Multiple training samples

In order to obtain a reliable model, multiple observations must be used to train the HMM. The extension of the proposed method to include multiple training samples may be carried out as follows. Consider two labeled sets: $X_1 = \{x_1^{(1)}, x_2^{(1)}, \cdots, x_{R_1}^{(1)} : d_1\}$ and $X_2 = \{x_1^{(2)}, x_2^{(2)}, \cdots, x_{R_2}^{(2)} : d_2\}$, where X_1 has R_1 number of samples and X_2 has R_2 number of samples, the separable distance function that takes care of all these samples is given by

$$I(X_1, X_2, \theta) = \frac{1}{R_1} \sum_{i=1}^{R_1} \log P(x_i^{(1)}|\theta) - \frac{1}{R_2} \sum_{i=1}^{R_2} \log P(x_i^{(2)}|\theta) \tag{3.38}$$

For simplicity, if we assume that the observation sequences in X_1 and X_2 have the same length T, then Eq.(3.9) may be rewritten as

$$\frac{1}{b_{ij}} [\frac{1}{R_1} \sum_{m=1}^{R_1} E(S_i, O_j|\theta, x_m^{(1)}) - \frac{1}{R_2} \sum_{n=1}^{R_2} E(S_i, O_j|\theta, x_m^{(2)})] = \lambda_i \tag{3.39}$$

Table 3.1: The 18 visemes selected for recognition

/a:/,	/ai/,	/ae/,	/ei/,	/i/,	/j/,	/ie/,	/o/,	/oi/
/th/,	/sh/,	/tZ/,	/dZ/,	/eu/,	/au/,	/p/,	/m/,	/b/

The probability coefficients are then estimated using Eq.(3.40).

$$b_{ij} = \frac{\frac{1}{R_1}\sum_{m=1}^{R_1} E(S_i, O_j|\theta, x_m^{(1)}) - \frac{1}{R_2}\sum_{n=1}^{R_2} E(S_i, O_j|\theta, x_n^{(2)})}{\sum_{j=1}^{M}[\frac{1}{R_1}\sum_{m=1}^{R_1} E(S_i, O_j|\theta, x_m^{(1)}) - \frac{1}{R_2}\sum_{n=1}^{R_2} E(S_i, O_j|\theta, x_n^{(2)})]} \tag{3.40}$$

While validating the state durations, the term $\frac{1}{R_1}\sum_{m=1}^{R_1} E(S_i, O_j|\theta, x_m^{(1)})$ is computed and compared with $\frac{1}{R_2}\sum_{n=1}^{R_2} E(S_i, O_j|\theta, x_n^{(2)})$. If

$$\frac{1}{R_1}\sum_{m=1}^{R_1} E(S_i, O_j|\theta, x_m^{(1)}) \approx \frac{1}{R_2}\sum_{n=1}^{R_2} E(S_i, O_j|\theta, x_n^{(2)}) \tag{3.41}$$

the training cycle is repeated; otherwise, the training terminates. In a more complex situation where the samples in X_1 and X_2 have different lengths, they can be first linearly scaled using Eq.(3.36) and then put through parameter estimation.

3.5 Application of two-channel HMM classifiers to lip reading

The proposed two-channel HMM method is applied to speaker-dependent viseme recognition. To highlight the discriminative power of the two-channel HMMs, 18 context-independent phonemes in Table 3.1 are chosen for recognition as some of them bear close similarity to another. For notational convenience, we still use the term "viseme" to indicate the visual representation of these phonemes. Note that the visemes used in this chapter are the one-to-one mappings of the phonemes.

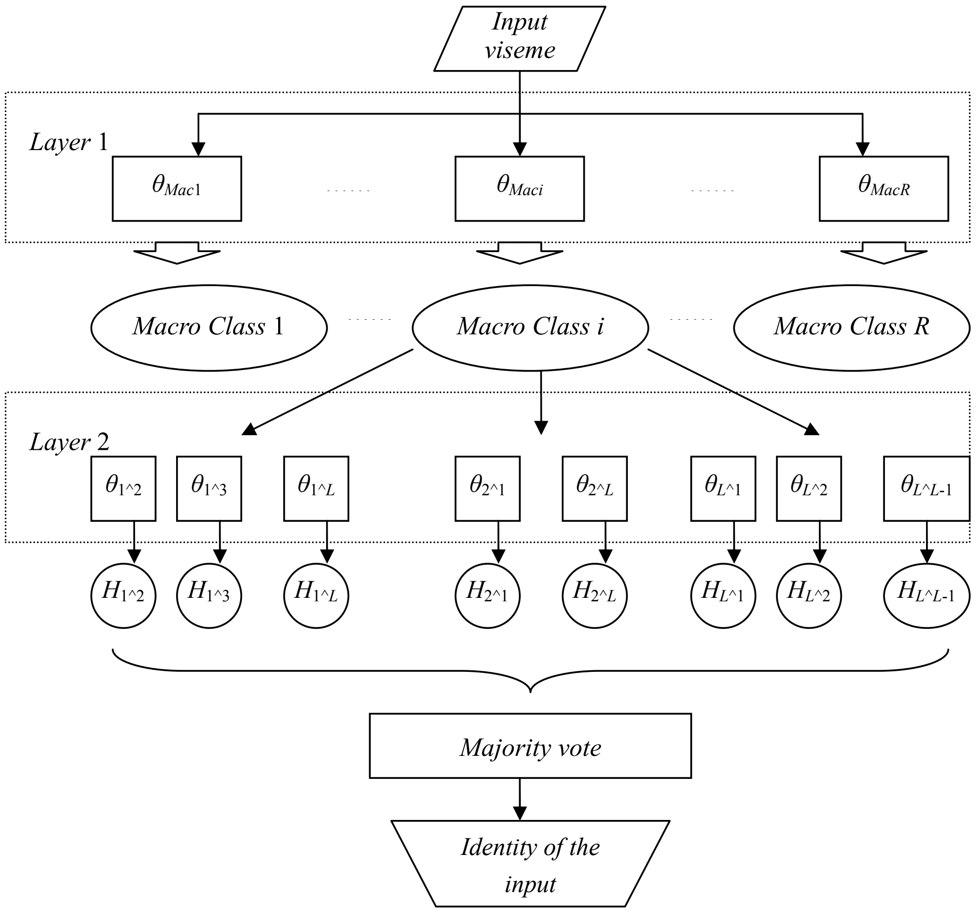

Figure 3.5: Flow chart of the hierarchical viseme classifier

3.5.1 Viseme classifier

The block diagram of the proposed hierarchical viseme classifier is given in Fig.3.5. The similarity between the visemes is measured first. Assume that $X_i = \{x_1^i, x_2^i, \cdots, x_{l_i}^i : d_i\}$ are the training samples of viseme d_i and l_i is the number of the samples, the joint probability scored by an HMM θ_j is computed as follows.

$$P(X_i|\theta_j) = \prod_{l=1}^{l_i} P(x_l^i|\theta_j) \tag{3.42}$$

A viseme model θ_i is able to separate visemes d_i and d_j if Eq.(3.43) applies,

$$\log P(X_i|\theta_i) - \log P(X_i|\theta_j) \geq \rho l_i, \quad \forall j \neq i \tag{3.43}$$

where ρ is a positive threshold. Otherwise, visemes d_i and d_j are categorized into a macro class and the training samples of d_i and d_j are jointly used to train the ML HMM of the macro class. In our experiments, the length of training samples is 25 and we set $\rho = 2$. In this way, the 18 visemes are clustered into 6 macro classes and the HMMs are denoted as $\theta_{Mac1}, \theta_{Mac2}, \cdots, \theta_{MacR}$ with $R = 6$ in Fig.3.5.

For an input viseme y^T to be identified, the probabilities $P(y^T|\theta_{Mac1})$, $P(y^T|\theta_{Mac2})$, $\cdots$, $P(y^T|\theta_{MacR})$ are computed and compared with one another. The macro identity of y^T is determined by the HMM that gives the largest probability.

A macro class may consist of several similar visemes. Fine recognition within a macro class is carried out at the second layer. Assume that Macro Class i comprises L visemes: $V_1, V_2, \cdots, V_L$, $L(L-1)$ two-channel HMMs are trained to separate each pair of them, denoted as $\theta_{j\wedge k}, (j, k = 1, 2, \cdots, L, j \neq k)$ in Fig.3.5. Note that the parameter-smoothed ML HMM of V_1 is adopted as the base HMM for $\theta_{j\wedge k}$ with viseme V_j being the true class and viseme V_k being the false class.

For an input viseme y^T to be identified, the following hypothesis is made,

$$H_{i\wedge j} = \begin{cases} i, & \text{if} \quad \log P(y^T|\theta_{i\wedge j}) - \log P(y^T|\theta_{j\wedge i}) > \rho \\ 0, & \text{otherwise} \end{cases} \tag{3.44}$$

where ρ is defined in Eq.(3.43) and $\rho = 2$ in this instance. $H_{i\wedge j} = i$ indicates a vote for V_i. The decision about the identity of y^T is made by majority vote of all the two-channel HMMs. The viseme class that has the maximum number of votes is chosen as the identity of y^T, denoted by $ID(y^T)$. Mathematically,

$$ID(y^T) = \max_i[\text{number of } H_{i\wedge j} = i], \quad \forall i, j = 1, 2, \cdots, L, i \neq j \tag{3.45}$$

If visemes V_i and V_j receive the same number of votes, the decision about the

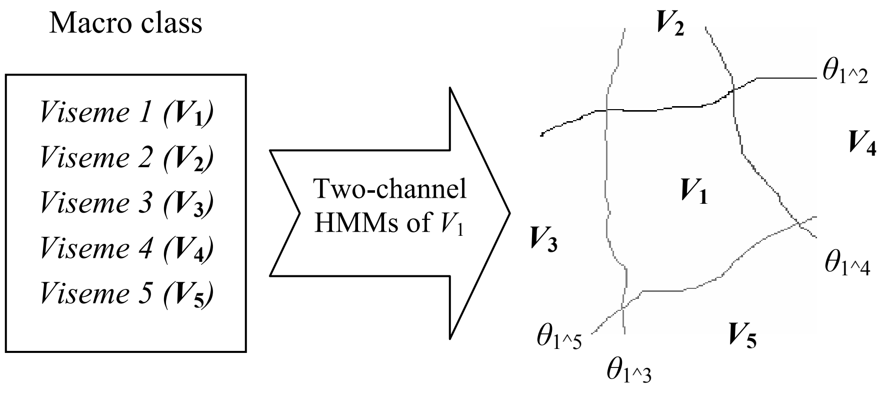

Figure 3.6: Viseme boundaries formed by the two-channel HMMs

identity of y^T is made by comparing $P(y^T|\theta_{i\wedge j})$ and $P(y^T|\theta_{j\wedge i})$ via Eq.(3.46).

$$ID(y^T) = \begin{cases} i, & \text{if} \quad \log P(y^T|\theta_{i\wedge j}) > \log P(y^T|\theta_{j\wedge i}) \\ j, & \text{otherwise} \end{cases} \tag{3.46}$$

The decision is based on pairwise comparisons of the hypotheses. It may greatly reduce the computational load because pairwise comparisons are carried out within each macro class, which comprises much fewer candidate classes than the entire set. If coarse identification is not performed, the number of classes increases and the number of pairwise comparisons goes up rapidly.

The two-channel HMMs act as the boundary functions for the viseme they represent as they serve to separate the correct samples from the samples of another viseme. A conceptual illustration is given in Fig.3.6 where the macro class comprises five visemes $V_1, V_2, \cdots, V_5$. $\theta_{1\wedge 2}, \theta_{1\wedge 3}, \cdots, \theta_{1\wedge 5}$ build the decision boundaries for V_1 to delimit it from the similar visemes. The proposed two-channel HMM model is specially tailored for the target viseme and its "surroundings". As a result, it is more accurate than the traditional modeling method that uses single ML HMM.

Table 3.2: The macro classes for coarse identification

Macro classes	Visemes	Macro classes	Visemes
1	/a:/, /ai/, /ae/	4	/o/, /oi/
2	/ei/, /i/, /j/, /ie/	5	/th/, /sh/, /tZ/, /dZ/
3	/eu/, /au/	6	/p/, /m/, /b/

Table 3.3: The average values of probability and separable distance function of the ML HMMs and two-channel HMMs

Viseme pair		$\tilde{\theta}_{ML}$		θ_1		θ_2				
$\boldsymbol{x}_1$	x_2	$\bar{P}$	$\bar{I}$	$\bar{P}$	$\bar{I}$	$\bar{P}$	$\bar{I}$	ω_1	ω_2	ω_3
/a:/	/ai/	-14.1	1.196	-17.1	5.571	-18.3	6.589	0.5	0.5	0.5
/ei/	/i/	-14.7	2.162	-19.3	5.977	-20.9	7.008	0.6	0.8	0.6
/au/	/eu/	-15.6	2.990	-18.1	5.872	-18.5	6.555	0.6	0.5	0.6
/o/	/oi/	-13.9	0.830	-17.5	2.508	-18.7	3.296	0.5	0.5	0.5
/th/	/sh/	-15.7	0.602	-19.0	2.809	-18.5	2.732	0.4	0.4	0.4
/p/	/m/	-16.3	1.144	-19.0	3.102	-17.1	2.233	0.4	0.5	0.4

3.5.2 Experimental results

Experiments are carried out to assess the performance of the proposed system. For each viseme in Table 3.1, 100 samples are drawn with 50 for training and the remaining 50 for testing. The samples have uniform length of 25 frames. The 6 macro classes obtained in Section 3.5.1 are illustrated in Table 3.2.

The results of fine recognition of some confusable visemes are listed in Table 3.3. Each row in Table 3.3 shows the two similar visemes that belong to the same macro class. $\boldsymbol{x}_1$ (in boldface) is the target (correct) viseme and x_2 is the incorrect viseme. $\tilde{\theta}_{ML}$ denotes the parameter-smoothed ML HMMs of $\boldsymbol{x}_1$. θ_1 and θ_2 are two-channel HMMs with different credibility factors ($\omega_1, \omega_2, \omega_3$ for the three states). For θ_1,

ω_1, ω_2 and ω_3 are set according to Eq.(3.29), with $C = 1.0$ and $D = 0.1$. For θ_2, ω_1, ω_2 and ω_3 are manually selected. $\bar{P}$ is the average log probability that is computed via Eq.(3.47).

$$\bar{P} = \frac{1}{l} \sum_{i=1}^{l} \log P(x_i^1 | \theta) \tag{3.47}$$

where x_i^1 is the i-th testing sample of viseme $\boldsymbol{x}_1$ and l is the number of the testing samples. $\bar{I}$ is the average separable distance that is computed via Eq.(3.48).

$$\bar{I} = \frac{1}{l^2} \sum_{i=1}^{l} \sum_{j=1}^{l} I(x_i^1, x_i^2, \theta) \tag{3.48}$$

It is seen that θ_1 and θ_2 give a much larger value of $\bar{I}$ than $\tilde{\theta}_{ML}$, which indicates that better discrimination capability is attained with the two-channel approach. In addition, $\bar{I}$ can be adjusted by tuning the credibility factors. However, θ_1 and θ_2 score smaller $\bar{P}$ than $\tilde{\theta}_{ML}$ does. It indicates that the two-channel HMMs are not good at modeling the visemes.

The change of $I(x_1, x_2, \theta)$ with respect to the training epochs in the two-channel training is depicted in Fig.3.7. For the three-state left-right HMMs and 25-frame-length training samples adopted in the experiment, the separable distance may become stable after 10$\sim$20 epochs. Such speed of convergence shows that the two-channel training is not computationally intensive for viseme recognition. It is also observed that $I(x_1, x_2, \theta)$ may drop at the first few training epochs. This is because that some symbols in subset V are transferred to U during training, which is explained in Section 3.3.2. Fig.3.7(d) also illustrates the situation of early termination. The training process stops because the state alignment condition mentioned in Section 3.3.1.is violated.

The performance of the proposed hierarchical system is compared with that of the traditional recognition system where ML HMMs (parameter-smoothed) are used as the viseme classifiers. The False Rejection Rates (FRR) of the two types of

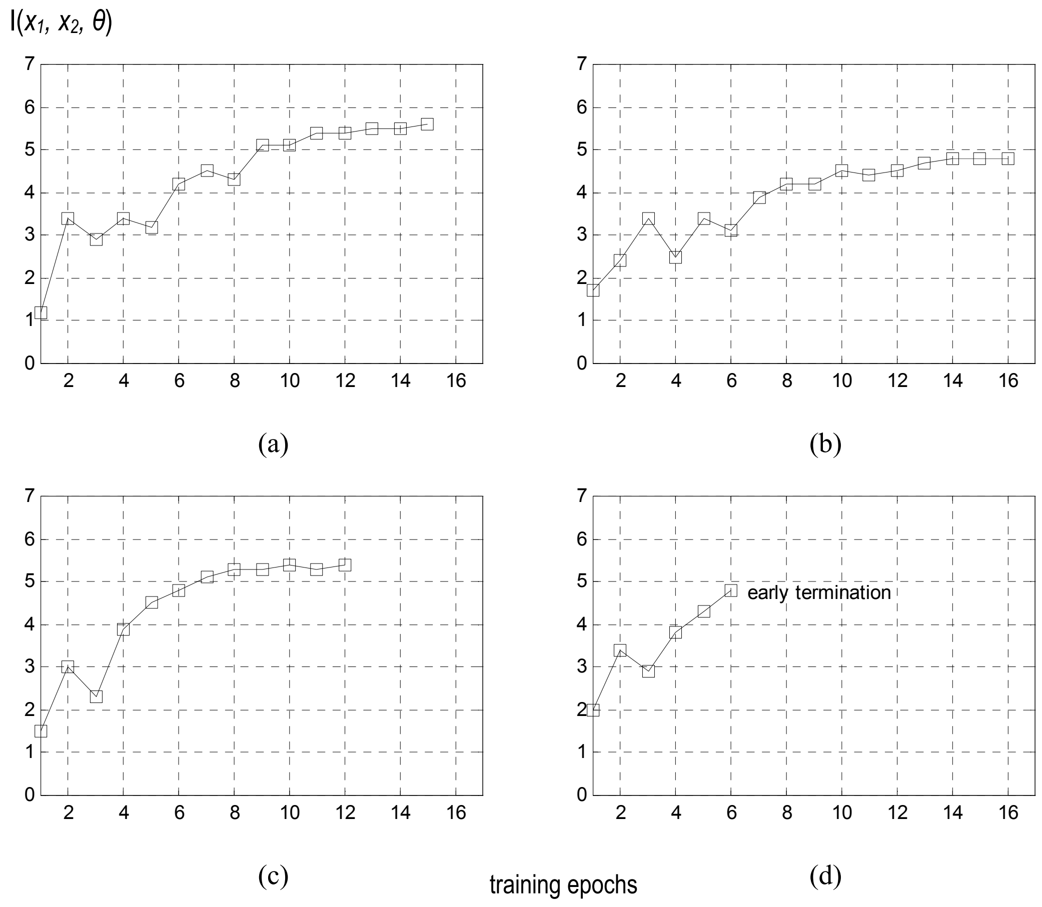

Figure 3.7: Change of $I(x_1, x_2, \theta)$ during the training process.

classifiers are computed for the 50 testing samples of each of the 18 visemes. Among them, 6 visemes can be accurately identified by the ML HMMs with FRRs less than 10%. To highlight the improvement resulting from the two-channel training, only the FRRs of the left 12 visemes are listed in Table 3.4.

Compared with the conventional ML HMM classifier, the classification error of the proposed hierarchical viseme classifier is reduced by about 20%. Thus the two-channel training algorithm is able to increase the discriminative ability of HMM significantly for identifying confusable visemes.

Table 3.4: Classification error ϵ_1 of the conventional classifier and classification error ϵ_2 of the two-channel classifier

Visemes	ϵ_1	ϵ_2	Visemes	ϵ_1	ϵ_2
/a:/	64%	12%	/o/	46%	28%
/ai/	60%	40%	/oi/	36%	8%
/ei/	46%	22%	/th/	18%	16%
/i/	52%	32%	/sh/	20%	12%
/au/	30%	18%	/p/	36%	12%
/eu/	26%	16%	/m/	32%	32%

3.6 The MSD training strategy

The two-channel training strategy mentioned in previous sections employs an ML model as the base model. This arrangement maintains the validity of the HMM but may be computationally expensive. The maximum separable distance (MSD) training presented in this section is an approach that training is performed on a non-demanding base model rather than a pre-trained ML HMM. For the training set $\{x_1^T : d_1\}$ and $\{x_2^T : d_2\}$ as mentioned in Section 3.1 and for the N-state M-symbol discrete HMM $\theta_{1,2}$, the steps of MSD estimation are given below.

3.6.1 Step 1: Parameter initialization

The selection of the initial values of MSD estimation is much easier than that in the two-channel training algorithm. Parameters in Matrix A and Matrix B can take arbitrary or uniform values provided that the probability constraints given in Eq.(2.12) and (2.13) are met. For the viseme classifiers discussed in this section, the parameters are initialized according to the procedures given in Section 2.4.2.

3.6.2 Step 2: Compute the expectations

The expectations $E(S_i, O_j|\theta_{1,2}, x_1^T)$ and $E(S_i, O_j|\theta_{1,2}, x_2^T)$ are computed in the same manner as mentioned in Section 3.2.3. By counting the states with designated output, we have,

$$E(S_i, O_j|\theta_{1,2}, x_1^T) = \sum_{\tau=1, s.t. o_\tau^1 = O_j}^{T} \sum_{j=1}^{N} \frac{\alpha_\tau^1(i) a_{ij} b_j(o_{\tau+1}^1) \beta_{\tau+1}^1(j)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_\tau^1(m) a_{mn} b_n(o_{\tau+1}^1) \beta_{\tau+1}^1(n)} \tag{3.49}$$

$$E(S_i, O_j|\theta_{1,2}, x_2^T) = \sum_{\tau=1, s.t. o_\tau^2 = O_j}^{T} \sum_{j=1}^{N} \frac{\alpha_\tau^2(i) a_{ij} b_j(o_{\tau+1}^2) \beta_{\tau+1}^2(j)}{\sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_\tau^2(m) a_{mn} b_n(o_{\tau+1}^2) \beta_{\tau+1}^2(n)} \tag{3.50}$$

and

$$D_{ij}(x_1^T, x_2^T, \theta_{1,2}) = E(S_i, O_j|\theta_{1,2}, x_1^T) - E(S_i, O_j|\theta_{1,2}, x_2^T) \tag{3.51}$$

where $\alpha_\tau^1(i)$ and $\beta_{\tau+1}^1(j)$ are forward and backward variables of x_1^T, $\alpha_\tau^2(i)$ and $\beta_{\tau+1}^2(j)$ are forward and backward variables of x_2^T.

3.6.3 Step 3: Parameter modification

For symbol O_j, if the corresponding $D_{ij}(x_1^T, x_2^T, \theta_{1,2}) > 0$, b_{ij} is set proportional to $D_{ij}(x_1^T, x_2^T, \theta_{1,2})$ using Eq.(3.10). For certain symbol, say O_p, if the expectation $D_{ip}(x_1^T, x_2^T, \theta_{1,2}) \leq 0$, a small value ϵ, e.g. $\epsilon = 10^{-3}$, is assigned to the corresponding b_{ip}. As a result, if there are L occurrences of $D_{ip}(x_1^T, x_2^T, \theta_{1,2}) \leq 0$, i.e.

$$L = \text{number of } [D_{ip}(x_1^T, x_2^T, \theta_{1,2}) \leq 0], \quad p = 1, 2, \cdots, M \tag{3.52}$$

b_{ij} is then estimated as via Eq.(3.53).

$$\tilde{b}_{ij} = \begin{cases} \frac{D_{ij}(x_1^T, x_2^T, \theta_{1,2})}{\Sigma_D}(1 - L\epsilon), & \text{if } D_{ij}(x_1^T, x_2^T, \theta_{1,2}) > 0 \\ \epsilon, & \text{otherwise} \end{cases} \tag{3.53}$$

where Σ_D is the sum of $D_{ij}(x_1^T, x_2^T, \theta_{1,2})$ provided that $D_{ij}(x_1^T, x_2^T, \theta_{1,2}) > 0$.

$$\Sigma_D = \sum_{j=1}^{M} D_{ij}(x_1^T, x_2^T, \theta_{1,2}), \quad \text{if} \quad D_{ij}(x_1^T, x_2^T, \theta_{1,2}) > 0 \tag{3.54}$$

Parameter smoothing is automatically considered in Eq.(3.53). As a result, $\theta_{1,2}$ trained in this way will not generate zero probability for a non-zero input sequence.

3.6.4 Step 4: Verification of state duration

The proposed MSD estimation also requires that the state durations of the training pair x_1^T and x_2^T are comparable. The expected durations $E(S_i|\theta_{1,2}, x_1^T)$ and $E(S_i|\theta_{1,2}, x_2^T)$ are computed using the method mentioned in Section 3.3.1. The MSD training continues if $E(S_i|\theta_{1,2}, x_1^T) \approx E(S_i|\theta_{1,2}, x_2^T)$, for example,

$$1.2 > E(S_i|\theta_{1,2}, x_1^T)/E(S_i|\theta_{1,2}, x_2^T) > 0.8, \quad i = 1, 2, \cdots, N \tag{3.55}$$

otherwise training stops even $I(x_1^T, x_2^T, \theta_{1,2})$ still shows the trend of increasing.

Step 2 and 3 are repeated in each training cycle. After that, Step 4 is implemented to verify the state durations. The procedures are repeated until either premature termination occurs, i.e. $E(S_i|\theta_{1,2}, x_1^T)$ and $E(S_i|\theta_{1,2}, x_2^T)$ differ too much, or the difference of $I(x_1^T, x_2^T, \theta_{1,2})$ between consecutive training cycles is smaller than a predefined threshold, or the specified number of training cycles is met.

The MSD training can also be extended to the cases of non-uniform training samples and multiple training samples. If the training pair has different lengths, linear scaling is performed as discussed in Section 3.4.1. If the training data are two sample sets with each of them consisting multiple samples, the symbol output coefficients are then estimated via the approach presented in Section 3.4.2.

3.6.5 Decision strategy

In a two-class identification problem, say Class d_1 vs. Class d_2, the identity of an unknown input y^T can be relatively easily determined by comparing the probabilities $P(y^T|\theta_{1,2})$ and $P(y^T|\theta_{2,1})$ using Eq.(3.56), where $\theta_{2,1}$ is the MSD estimation

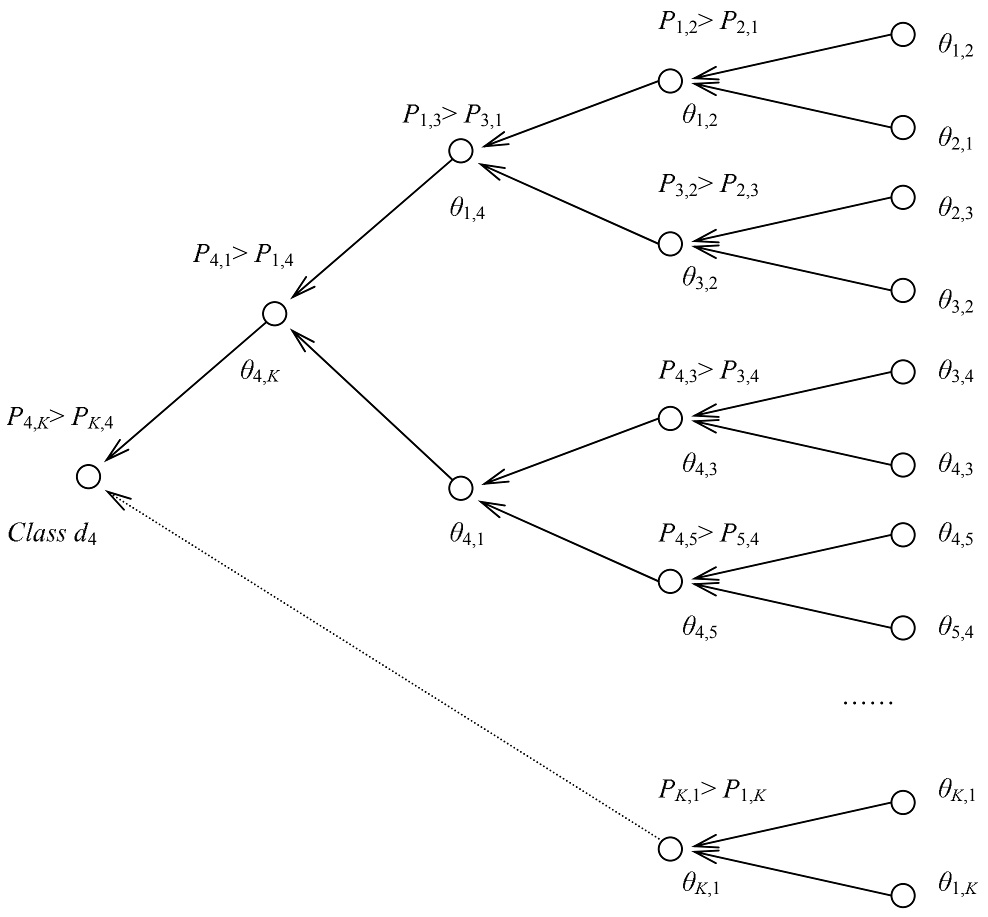

Figure 3.8: The eliminating series for determining the identity of the input sample in multi-class identification

of the HMM (abbreviated as MSD HMM) with Class d_2 being the true class and Class d_1 being the false class.

$$ID(y^T) = \begin{cases} d_1, & \text{if} \quad P(y^T|\theta_{1,2}) > P(y^T|\theta_{2,1}) \\ d_2, & \text{otherwise} \end{cases} \tag{3.56}$$

In a multi-class case, say Class $d_1, d_2, \cdots, d_K$, $K(K-1)$ MSD HMMs, $\theta_{i,1}, \theta_{i,2}, \cdots,$ $\theta_{i,j}, \cdots, \theta_{i,K} (i, j = 1, 2, \cdots, K, i \neq j)$, are trained using the MSD estimation, where $\theta_{i,j}$ indicates Class d_i is the true class and Class d_j is the false class. For an input sequence y^T that belongs to one of the K classes, its identity is determined by an elimination series described below:

1.) $2K$ MSD HMMs $\theta_{1,2}, \theta_{2,1}, \theta_{2,3}, \theta_{3,2}, \theta_{3,4}, \theta_{4,3}, \cdots, \theta_{K,1}, \theta_{1,K}$ are grouped into K pairs and the probabilities of y^T are computed, i.e. $P(y^T|\theta_{1,2})$ and $P(y^T|\theta_{2,1})$, $P(y^T|\theta_{2,3})$ and $P(y^T|\theta_{3,2})$,$\cdots$, $P(y^T|\theta_{i,j})$ and $P(y^T|\theta_{j,i})$,$\cdots$, $P(y^T|\theta_{K,1})$ and $P(y^T|\theta_{1,K})$. They are denoted as $P_{i,j} = P(y^T|\theta_{i,j})$ in Fig.3.8.

2.) The probabilities $P(y^T|\theta_{i,j})$ and $P(y^T|\theta_{j,i})$ are compared. If $P(y^T|\theta_{i,j}) > P(y^T|\theta_{j,i})$, Class d_i is more likely to be the identity of y^T than Class d_j and d_i is selected as the winning class; otherwise, Class d_j is selected as the winning class.

3.) The MSD HMMs of the winning classes are regrouped. If the winning classes of two neighboring groups are Class d_i and Class d_p, respectively, $\theta_{i,p}$ and $\theta_{p,i}$ are then chosen as the HMMs for the next round of comparison.

The number of winning classes halves after each decision round. Step $1-3$ are repeated until only one winning class remains. Under such management, it is usually unnecessary to compute all the probabilities that are scored by the $K(K-1)$ MSD HMMs.

3.7 Application of MSD HMM classifiers to lip reading

The proposed MSD estimation is applied to recognize several groups of words. The performance is compared with that of the ML HMM mentioned in Section 2.4.

3.7.1 Data acquisition for word recognition

Five groups of words are selected for recognition, which are shown in Table 3.5. Each group consists of one true word and five false words. The false words may not have meaning in English and they are denoted by sequences of phonemes that build

Table 3.5: The words to be identified by the HMM classifiers

Group No.	True word	False word 1	False word 2	False word 3	False word 4	False word 5
1	**zoo** /z/+/U/	/h/+/U/	/l/+/o/	/tr/+/ue/	/s/+/ue/	/zh/+/U/
2	**hot** /h/+/o/+/t/	/r/+/o/ +/t/	/l/+/oi/ +/d/	/n/+/eu/ +/t/	/l/+/u/ +/k/	/m/+/u/ +/d/
3	**deck** /d/+/e/+/k/	/h/+/A/ +/k/	/n/+/e/ +/g/	/t/+/e/ +/d/	/d/+/ei/ +/t/	/l/+/ei/ +/k/
4	**sleep** /s/+ /l/+/I/+/p/	/s/+/n/ +/ei/+/b/	/sh/+/l/ +/e/+/p/	/s/+/k/ +/I/+/p/	/z/+/l/ +/I/+/k/	/ch/+/r/ +/ei/+/b/
5	**transit** /tr/+/an/+ /s/+/i/+/t/	/tr/+/A/ +/z/+/i/ +/t/	/dr/+/A/ +/s/+/ei/ +/d/	/ch/+/an/ + /sh/+/i/ +/d/	/tr/+/o/ +/s/+/i/ +/k/	/dr/+/an/ +/sh/+/e/ +/t/

them. The true word and the false words in each group are visually similar with each other such that they are difficult to be separated with normal ML HMMs.

The speaker is asked to clearly articulate each true word for 100 times and each false word for 50 times. The samples (video clips) of the words in the same group are manually truncated so that they have the same length (number of frames). 50 samples of a true word are used for training the HMM while the left 50 samples for testing. The 50 samples of the false word are all applied for training.

3.7.2 Experimental results

Decision strategy proposed in Section 3.6.5 is adopted to identify the words. Note that each group consists of $K = 6$ words, thus $6 \times (6 - 1) = 30$ MSD HMMs are trained using the 50 training samples, denoted as $\theta_{1,2}, \cdots, \theta_{1,6}, \theta_{2,1}, \cdots, \theta_{2,6}, \cdots,$ $\theta_{6,1}, \cdots, \theta_{6,5}$. Take $\theta_{1,2}$ as an example, it indicates that *hot* is the true class while */r/+/o/+/t/* is the false class. At the mean time, for each true/false word in Table 3.5, an ML HMM is also trained using the 50 training samples.

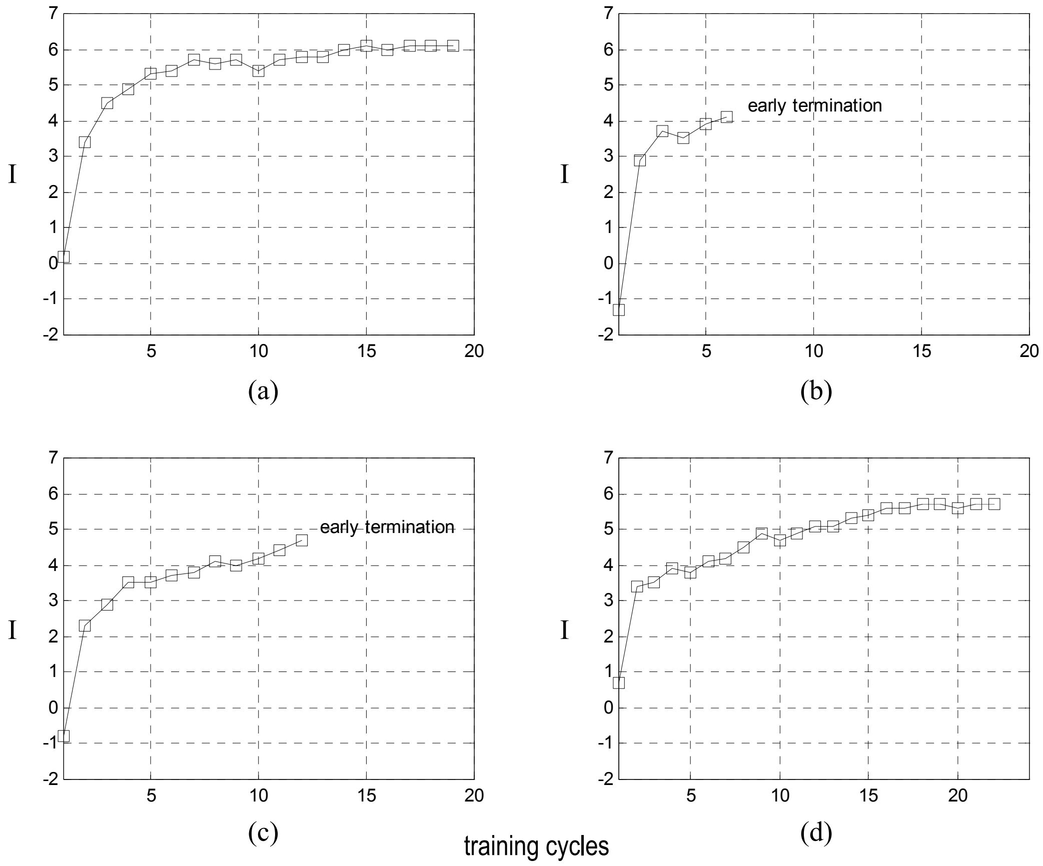

Figure 3.9: The change of the separable distance with respect to the training cycles (a) $\theta_{1,2}$: true class - *hot*, false class - */r/+/o/+/t/* (b) $\theta_{1,3}$: true class - *hot*, false class - */l/+/oi/+/d/* (c) $\theta_{1,4}$: true class - *hot*, false class - */n/+/eu/+/t/* (d) $\theta_{1,5}$: true class - *hot*, false class - */l/+/u/+/k/*

The separable distance varies while implementing the MSD estimation. In Fig.3.9, the separable distances defined in Eq.(3.38) of four MSD HMMs with respect to the training cycles are depicted. It is observed that the separable distance shows the trend of increasing in the process of MSD estimation and becomes stable after about 20 cycles, i.e. the difference of the separable distances between consecutive cycles is less than 0.05. The case of early termination is also depicted in Fig.3.9(b)

Table 3.6: The separable distances measured by the two types of HMMs

Model type	Separable distance (*true word* vs. *false word*)			
	hot vs. /r/+/o/+/t/	*hot* vs. /l/+/oi/+/d/	*hot* vs. /n/+/eu/+/t/	*hot* vs. /l/+/u/+/k/
ML HMMs	1.27	0.74	1.05	1.79
MSD HMMs	6.13	4.09	4.71	5.77

Table 3.7: Classification rates of the ML HMMs and the MSD HMMs

	Group 1	Group 2	Group 3	Group 4	Group 5
ML HMMs	48%	36%	60%	64%	72%
MSD HMMs	70%	54%	68%	82%	82%

and (c), where the state alignment condition discussed in Section 3.6.4 is violated. The separable distances measured for the same training samples by the ML HMMs are given in Table 3.6, together with those scored by the MSD HMMs. It is seen that the separable distances scored by the MSD HMMs are much greater than those scored by the ML HMMs. As a result, the MSD HMMs can tell apart the true word out of the confusable false words more easily than the ML HMMs do.

The 50 testing samples of each true word are identified by ML HMMs and MSD HMMs, respectively. Note that recognition is carried on a closed set, i.e. six words in each group. For the ML HMMs, the classification rates are computed via Eq.(2.17) while for the MSD HMMs, the rates are computed by counting the true decisions made by the eliminating series described in Section 3.6.5.

The classification rates depicted in Table 3.7 show that the MSD HMMs improve the classification accuracy by about 15% than the ML HMMs. It further proves that the MSD estimation is able to improve the discriminative ability of HMM.

3.8 Summary

The two-channel training strategy and the MSD estimation discussed in this chapter are discriminative training methods of HMM. Both approaches employ the separable distance as the criterion function while they differ in implementation. The two-channel training algorithm applies a static-channel to maintain the validity of the HMM. If the static-channel is derived from an ML estimation of HMM, the two-channel training algorithm can be looked as a one-step improvement to the conventional Baum-Welch estimation. Compared with the two-channel training, the MSD training is more easily implementable because the requirement on the base model is not stringent, and the parameter estimation process of the MSD training is based on some simple manipulations to the criterion function.

In the application to visual speech processing, both two-channel HMMs and MSD HMMs can improve classification rates by about 20% (for visemes) or 15% (for words) over the ML HMMs after 10∼20 training epochs. It manifests that both approaches excels the traditional Baum-Welch estimation while identifying confusable observations and also has fast speed of convergence.

On the other hand, being discriminative training for HMM, two-channel method and MSD estimation have some limitations in common. First, the trained classifiers are not good models of the target sequence. They should be used in conjunction with other classifiers for fine recognition after coarse recognition is performed. Second, both classifiers principally work on the individual recognition units rather than long sequences. Third, the state alignment condition of the training strategies is a rather strict condition that is sometimes impractical for real world scenarios. Fourth, the individual two-channel HMM classifier and MSD HMM classifier can only perform binary classification. To extend the approaches to multi-class case, the implementation of the recognition may be based on the decision tree strategy, majority vote or the eliminating series that are adopted in this chapter.

Chapter 4

Recognition of Visual Speech Elements Using Adaptively Boosted HMMs

Traditional approaches for optimizing an HMM classifier focuses on improving the performance of the individual HMM, e.g. setting proper HMM parameters, maximizing the probabilities of specific samples, minimizing the cross-entropy and so on. The separable-distance-based training strategies presented in Chapter 3 also belong to the class of traditional training approaches. On the other hand, an HMM-based classifier is more than a single HMM. It may be a multiple-HMM classifier, a hybrid neural network/HMM classifier and so on. For such a complex classifier, configuring and training of the individual HMM is not so critical as in the single-HMM classifier. The key point of designing a complex HMM-based classifier is usually the incorporative application of multiple identification techniques. The AdaBoost-HMM classifier presented in this chapter is a kind of multiple-HMM classifier and the highlight of the classifier is the incorporation of the Adaptive Boosting technique and HMM modeling.

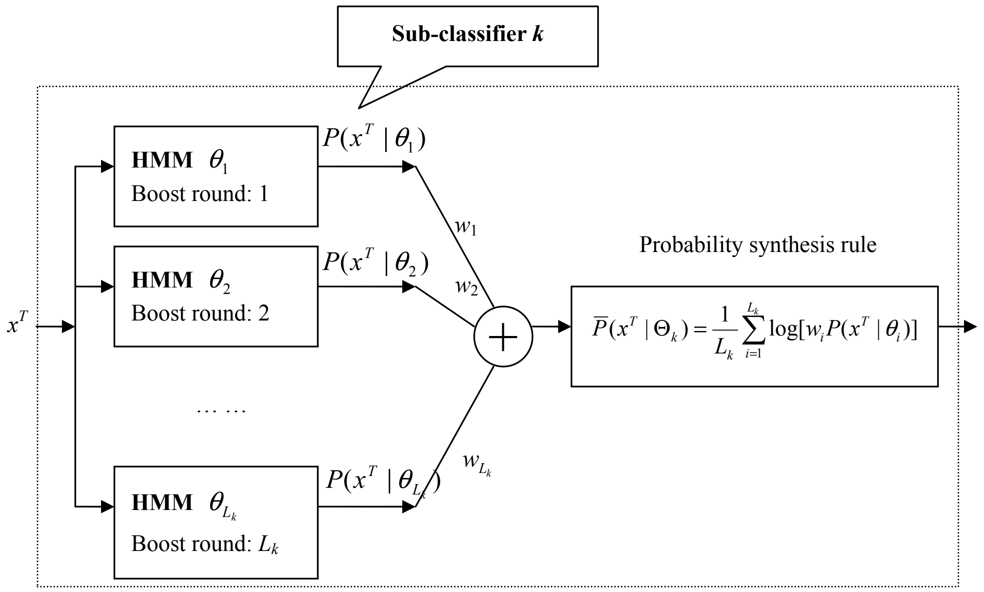

Figure 4.1: Block diagram of an Adaptively Boosted multiple-HMM sub-classifier.

4.1 An overview of the proposed system

The proposed system employs a bank of HMMs that are connected in parallel as the basic recognition units. For the K-class identification problem, the block diagram of the structure of a typical sub-classifier of the proposed system is presented in Fig.4.1. The sub-classifier of the k-th class, Θ_k, is comprised of L_k HMMs, where L_k is the number of Boosting rounds (or the number of the composite HMMs). Each composite HMM computes the probability for an input sequence x^T. The outputs from the L_k HMMs, $P(x^T|\theta_1), P(x^T|\theta_2), \cdots, P(x^T|\theta_{L_k})$ are processed according to a desired probability synthesis rule. The synthesized likelihood $\bar{P}(x^T|\Theta_k)$ is then submitted to the decision module to determine the identity of the input x^T.

A conceptual interpretation of the proposed multi-HMM classifier is given in Fig.4.2. It is expected that by synthesizing the decisions made by the multiple HMMs, a more complex decision boundary can be formulated than using single HMM.

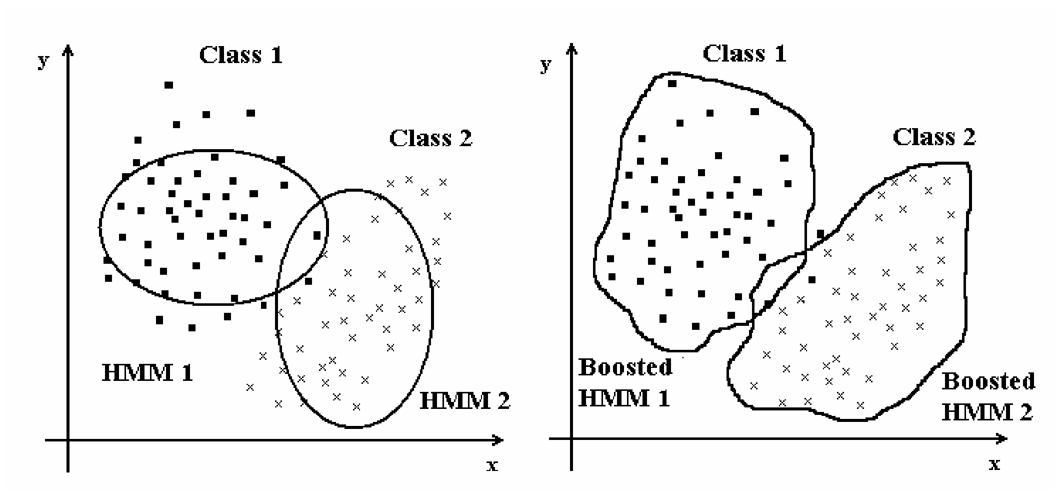

Figure 4.2: A conceptual illustration: the decision boundaries formed by single HMM (left) and multiple HMMs (right)

4.2 Review of Adaptive Boosting

Adaptive Boosting (AdaBoost) algorithm is a method for converting a weak classifier into a strong one. This method maintains a distribution of weights over the training set. A new classifier is trained by weighted samples and the final decision is made by summing up the sub-decisions of all the sub-classifiers. Assume that in the two-class identification problem (Class 1 vs. Class 2), the training set is comprised of R samples $\{y_1, d_1\}, \{y_2, d_2\}, \cdots, \{y_R, d_R\}$. y_i is the observed data and $d_i = \{-1, +1\}$ is the label identifier where $d_i = -1$ denotes Class 1 and $d_i = +1$ denotes Class 2. The process of Adaptive Boosting involves of a series of rounds ($t = 1, 2, \cdots, T$) of weight-adjusting and classifier-training [104]. Let $D_t(i)$ stand for the weight that is assigned to the i-th training sample in the t-th Boosting round and D_t denote the set of the weights $\{D_t(1), D_t(2), \cdots, D_t(R)\}$. The steps of Adaptive Boosting are presented below:

1. Initially, assign the weight $D_1(i) = 1/R, (i = 1, 2, \cdots, R)$ to the R training

samples $\{y_1, d_1\}, \{y_2, d_2\}, \cdots, \{y_R, d_R\}$. Note that $D_1(i)$ follows a uniform distribution.

2. Train the t-th classifier θ_t in the t-th Boosting round using the distribution D_t, starting with $t = 1$.

3. Formulate the hypothesis, $h_t(y_i) \rightarrow \{-1, +1\}, (i = 1, 2, \cdots, R)$ and calculate the error $\varepsilon_t = P[h_t(y_i) \neq d_i]$ for θ_t.

4. Calculate $w_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$, which weights the importance of the classifier θ_t.

5. Update the distribution: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-w_t}, & \text{if } h_t(y_i) = d_i \\ e^{w_t}, & \text{if } h_t(y_i) \neq d_i \end{cases}$, where Z_t is a normalization factor to make D_{t+1} a distribution.

The procedures of Adaptive Boosting are also presented as a flowchart in Fig.4.3. Step 2 to Step 5 are repeated until the error rate of the classifier exceeds 0.5 or after a given number of training epochs. A series of classifiers $\theta_1, \theta_2, \cdots, \theta_T$ and weights $w_1, w_2, \cdots, w_T$ are obtained at the end of the above procedures. In this paper, the T sub-classifiers, together with the weights assigned to them, are looked as an integral entity and are referred to as an AdaBoost-classifier. For an unknown input y, the sub-decisions made by the T composite classifiers are synthesized using $H(y) = \text{sign}[\sum_{t=1}^{T} w_t h_t(y)]$. If $H(y) < 0$, $y \in$Class 1, otherwise $y \in$Class 2.

The objective of boosting is to minimize the training error $\varepsilon(H) = \frac{1}{R}\sum_{i=1}^{R}[i : H(y_i) \neq d_i]$ of the final hypothesis. Schapire and Singer proved that $\varepsilon(H)$ is bounded as follows [105]:

$$\varepsilon(H) \leq \frac{1}{R}\sum_{i=1}^{R} \exp[-d_i f(y_i)] \tag{4.1}$$

where $f(y_i) = \sum_{t=1}^{T} w_t h_t(y_i)$. By unraveling the recursive definition of D_t, we have

$$\frac{1}{R}\sum_{i=1}^{R} \exp[-d_i f(y_i)] = \prod_{t=1}^{T} Z_t \tag{4.2}$$

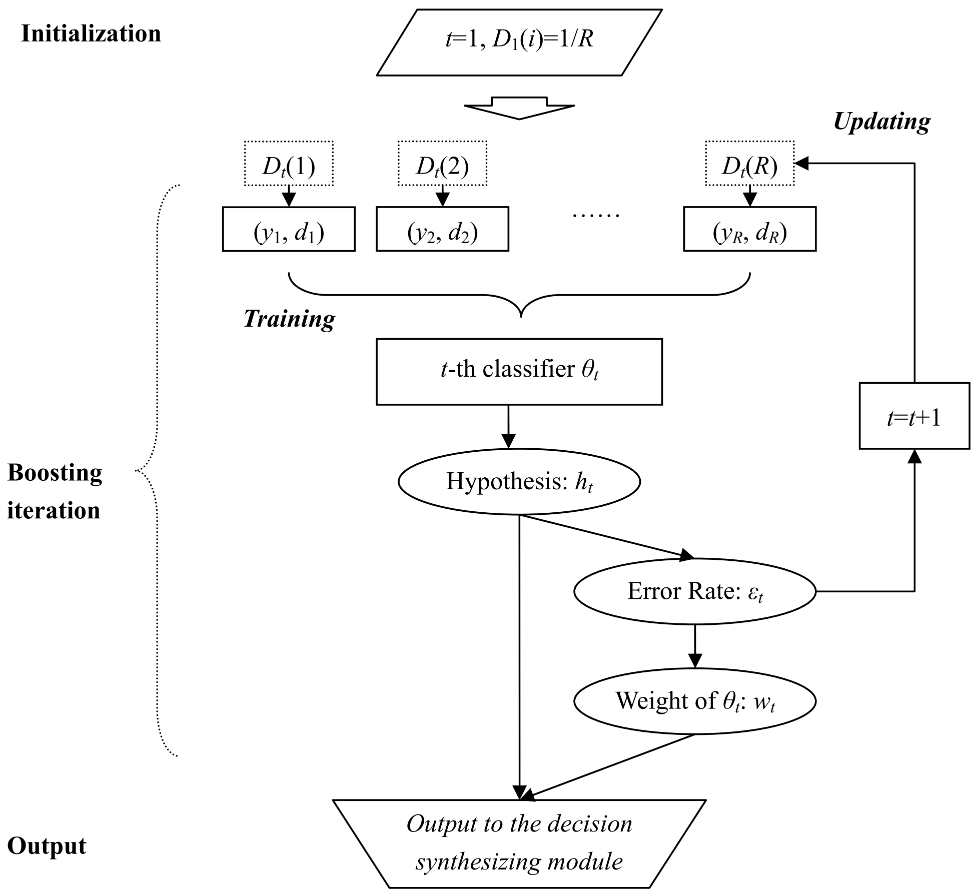

Figure 4.3: Steps of Adaptive Boosting algorithm.

where

$$Z_t = \sum_{i=1}^{R} D_t(y_i)\exp[-w_t d_i h_t(y_i)] \tag{4.3}$$

Eq.(4.1) suggests that the training error can be reduced most rapidly by choosing w_t and h_t at each round to minimize Z_t. In the case of binary hypotheses, this leads to the choice of w_t in Eq.(4.4), which is adopted in Step 4 of the above mentioned boosting steps [106].

$$w_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t}) \tag{4.4}$$

If the training error of θ_t (the classifier obtained at the t-th Boosting round) is less than 0.5, say $\varepsilon_t = \frac{1}{2} - \gamma_t (\gamma_t > 0)$, it is also proved in [107] that the training error is bounded as in Eq.(4.5).

$$\varepsilon(H) \leq 2^T \prod_{t=1}^{T} \sqrt{\varepsilon_t(1-\varepsilon_t)} = \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2} \leq \exp(-2\sum_{t=1}^{T} \gamma_t^2) \tag{4.5}$$

Eq.(4.5) shows that if the individual classifier has a classification error less than 0.5 (or equivalently a classification rate greater than 0.5), the overall error rate should decrease exponentially [107]. The boosted classifier may generate a hypothesis with an arbitrary low rate of error as boosting continues.

4.3 AdaBoosting HMM

Boosting is carried out during the training phase of the composite HMMs. For the K class identification problem, the purpose of HMM AdaBoosting is to train a set of HMMs that represent or span the distribution of the training samples. In the application of AdaBoosting strategy to the construction of a multi-HMM classifier, the following two important issues arise: 1) The choice of base training algorithm and 2) the measurement of classification error. These are discussed in detail in the following paragraphs.

4.3.1 Base training algorithm

The most popular training algorithm for HMM is the Baum-Welch algorithm mentioned in Section 2.4.1. The Baum-Welch method makes use of Maximum Likelihood (ML) estimation. The parameters are adjusted toward the direction of maximizing the probability $P(x^T|\theta)$ where x^T is the training sample. Since the Expectation-Maximization (EM) iterations are applied in the training procedures, the method has the advantage of ease of implementation and speed of convergence.

However, the parameters of the HMM obtained from the Baum-Welch estimation are solely determined by the "correct" samples. The trained HMM is therefore not guaranteed to discriminate similar samples well.

An alternative to the Baum-Welch training algorithm is the Maximum Mutual Information (MMI) estimation mentioned in Section 3.1. This method is a discriminative training method because it increases the *a posteriori* probabilities of the model corresponding to the training data. However, the analytical solution to the MMI criterion function is difficult to realize, the implementation of MMI estimation is hence tedious.

As mentioned in Section 4.2, Adaptive Boosting has loose requirements on the selection of base classifiers. As long as the training error of the individual classifier is less than 0.5, the training error of the AdaBoost-classifier will drop. The same requirements apply to HMMs. If the training error of the composite HMMs is forced to be less than 0.5, the error rate of the AdaBoost-HMM classifier will also decrease as boosting continues. Besides this, since each of the multiple HMMs has to be individually trained, reduction of the computational load per HMM is also an important consideration. Considering these two factors, the Baum-Welch algorithm, which is less computationally intensive, is adopted.

Assume that $X_k = \{x_1^k, x_2^k, \cdots, x_{R_k}^k : d_k\}$ are R_k training samples (sequences) of Class d_k, where $x_l^k = (o_1^{k,l}, o_2^{k,l}, \cdots, o_{T_l}^{k,l}), (l = 1, 2, \cdots, R_k)$ is a T_l-length observation sequence and $o_i^{k,l} (i = 1, 2, \cdots, T_l)$ is the i-th symbol appeared in the sequence. For the N-state M-symbol HMM $\theta(\pi, A, B)$, we define the forward variables $\alpha_t^{k,l}(i) = P(o_1^{k,l}, o_2^{k,l}, \cdots, o_t^{k,l}, s_t = S_i|\theta)$ and backward variables $\beta_t^{k,l}(i) = P(o_{t+1}^{k,l}, o_{t+2}^{k,l}, \cdots, o_{T_k}^{k,l}|s_t = S_i, \theta)$ for x_l^k, the parameters of the HMM are then estimated through the following EM recursion [99].

$$\bar{a}_{ij} = \frac{\sum_{l=1}^{R_k} \frac{1}{P(x_l^k|\theta)} \sum_{t=1}^{T_l-1} \alpha_t^{k,l}(i) a_{ij} b_j(o_{t+1}^{k,l}) \beta_{t+1}^{k,l}(j)}{\sum_{l=1}^{R_k} \frac{1}{P(x_l^k|\theta)} \sum_{t=1}^{T_l-1} \alpha_t^{k,l}(i) \beta_{t+1}^{k,l}(j)} \tag{4.6}$$

$$\bar{b}_{jm} = \frac{\sum_{l=1}^{R_k} \frac{1}{P(x_l^k|\theta)} \sum_{t=1,s.t.o_t^{k,l}=O_m}^{T_l-1} \alpha_t^{k,l}(j)\beta_t^{k,l}(j)}{\sum_{l=1}^{R} \frac{1}{P(x_l^k|\theta)} \sum_{t=1}^{T_l-1} \alpha_t^{k,l}(j)\beta_t^{k,l}(j)} \tag{4.7}$$

where O_m is the m-th symbol in the symbol set. The terms used in Eq.(4.6) and (4.7) are discussed in Section 2.4.1. In the above mentioned strategy, all the samples are treated equally. If weight D_l is assigned to the l-th sample x_l^k, then Eq.(4.6) and (4.7) become,

$$\bar{a}_{ij} = \frac{\sum_{l=1}^{R_k} \frac{D_l}{P(x_l^k|\theta)} \sum_{t=1}^{T_l-1} \alpha_t^{k,l}(i) a_{ij} b_j(o_{t+1}^{k,l})\beta_{t+1}^{k,l}(j)}{\sum_{l=1}^{R_k} \frac{D_l}{P(x_l^k|\theta)} \sum_{t=1}^{T_l-1} \alpha_t^{k,l}(i)\beta_{t+1}^{k,l}(j)} \tag{4.8}$$

$$\bar{b}_{jm} = \frac{\sum_{l=1}^{R_k} \frac{D_l}{P(x_l^k|\theta)} \sum_{t=1,s.t.o_t^{k,l}=O_m}^{T_l-1} \alpha_t^{k,l}(j)\beta_t^{k,l}(j)}{\sum_{l=1}^{R} \frac{D_l}{P(x_l^k|\theta)} \sum_{t=1}^{T_l-1} \alpha_t^{k,l}(j)\beta_t^{k,l}(j)} \tag{4.9}$$

For the above equations, Arslan and Hansen proved that weighting of the training samples does not violate the convergence property of the maximum likelihood training [108]. A local maximum point of $P(X_k|\theta)$ will be attained after a sufficient number of training epochs. Since different samples are treated differently in estimating the parameters, Eq.(4.8) and (4.9) have the potential to be applied to HMM AdaBoosting. In this thesis, the above mentioned training strategy shall be referred to as the biased Baum-Welch estimation.

4.3.2 Cross-validation for error estimation

Unlike Neural Network or other classifiers, HMM gives a probabilistic measure rather than a definite Boolean result. The decision about the identity of the input is usually obtained by comparing the probabilities measured of all HMMs.

Assume that at a certain Boosting round, Class $d_l (l = 1, 2, \cdots, K)$ has L_l sub-classifiers (HMMs) - $\Theta_l = \{\theta_1^l, \theta_2^l, \cdots, \theta_{L_l}^l\}$. As mentioned in Section 2.4.1, for a given training sample $x_i^k \in X_k$, the identity of x_i^k is determined via Eq.(4.10).

$$\theta^* = \arg\max_{\theta} P(x_i^k|\theta_j^l), \quad \forall l = 1, 2, \cdots, K, j = 1, 2, \cdots, L_l \tag{4.10}$$

The decision made in this way is a *one versus the rest* classification. If the correct model scores greater likelihood than the others, the result is correct; otherwise, an error occurs. As a result, the following hypothesis is made upon an HMM classifier in Class d_k, e.g. $\theta_p^k, (1 \leq p \leq L_k)$:

$$h_p^k(x_i^k) = \begin{cases} 1, & \text{if } P(x_i^k|\theta_p^k) > P(x_i^k|\theta_q^j), \quad \forall j \neq k, q = 1, 2, \cdots, L_j \\ -1, & \text{otherwise} \end{cases} \tag{4.11}$$

The training error of θ_p^k is estimated by summarizing the weighted hypotheses over all the training samples in X_k.

$$\varepsilon_p^k = D(x_i^k)E(h_p^k(x_i^k) = -1) = \sum_{i=1, s.t. h_p^k(x_i^k)=-1}^{R_k} D(x_i^k) \tag{4.12}$$

It is shown in Eq.(4.11) that the error rate not only depends on the classifier itself but also relates to the other classifiers. The HMM obtained at Boosting round t, θ_t^k, will influence the error rate of all the HMMs trained at the previous Boosting rounds, e.g. $\theta_\tau^j (j = 1, 2, \cdots, K, \tau < t)$. AdaBoosting requires that the composite classifiers have an error rate less than 0.5. As a result, not only the recently boosted HMM but also all the existing HMMs have to be validated. Clearly the computations involved in calculating and comparing the probabilities are intensive. In our system, the following measures are taken to facilitate the processing.

As illustrated in Fig.4.4, each class, say d_k, maintains a maximum probability array. The elements in the array are the R_k greatest probabilities of the training samples $X_k = \{x_1^k, x_2^k, \cdots, x_{R_k}^k\}$ that are scored by the HMMs of Class $d_j (\forall j \neq k)$. These maximum probabilities are denoted as $P_{max}(x_1^k|\bar{\theta}), P_{max}(x_2^k|\bar{\theta}), \cdots, P_{max}(x_{R_k}^k|\bar{\theta})$ in the figure, where $\bar{\theta}$ denotes any HMM of the class other than d_k.

At Boosting round $L_k + 1$, after a new HMM of Class d_k is trained, say $\theta_{L_k+1}^k$, the probabilities of all the training samples of Classes $d_1, d_2, \cdots, d_K$, given $\theta_{L_k+1}^k$ are computed. For Class d_l as an example, if $P(x_i^l|\theta_{L_k+1}^k) > P_{max}(x_i^l|\bar{\theta}), (i =$

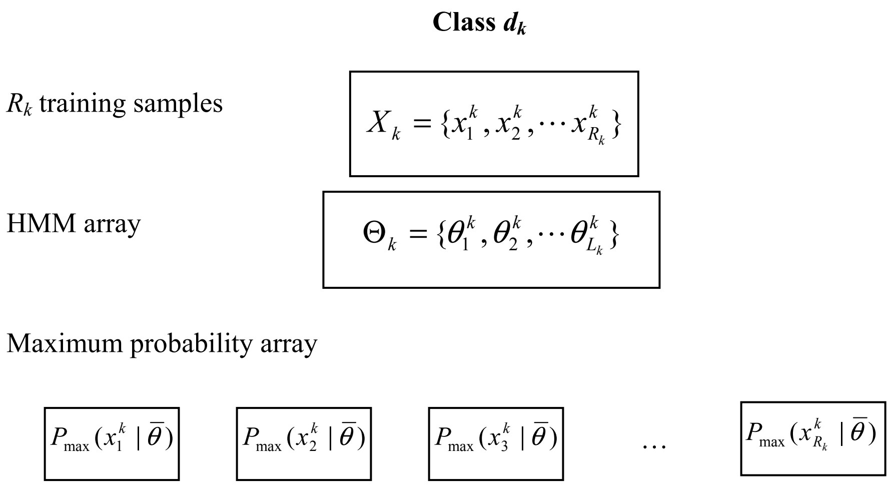
Class d_k
R_k training samples
$X_k = \{x_1^k, x_2^k, \cdots x_{R_k}^k\}$
HMM array
$\Theta_k = \{\theta_1^k, \theta_2^k, \cdots \theta_{L_k}^k\}$
Maximum probability array
$P_{\max}(x_1^k \mid \bar{\theta})$
$P_{\max}(x_2^k \mid \bar{\theta})$
$P_{\max}(x_3^k \mid \bar{\theta})$
$\cdots$
$P_{\max}(x_{R_k}^k \mid \bar{\theta})$

Figure 4.4: Data structure for implementing error estimation.

$1, 2, \cdots, R_l$), $P_{max}(x_i^l|\bar{\theta})$ is replaced with $P(x_i^l|\theta_{L_l+1}^k)$. The following hypothesis, which is concluded from Eq.(4.11), is made for a training sample of Class d_k,

$$h_{L_k+1}^k(x_i^k) = \begin{cases} 1, & \text{if } P(x_i^k|\theta_{L_k+1}^k) > P_{max}(x_i^k|\bar{\theta}), \quad i = 1, 2, \cdots, R_k \\ -1, & \text{otherwise} \end{cases} \tag{4.13}$$

The training error of $\theta_{L_k+1}^k$ is then computed using Eq.(4.12). In this way, the training error of any composite HMM, say $\theta_t^l, (t = 1, 2, \cdots, L_l)$, can be easily obtained by comparing with the corresponding value in its maximum likelihood array - $P_{max}(x_i^l|\bar{\theta})$.

If the error rates of $\theta_{L_k+1}^k$ and the existing HMMs are all less than 0.5, $\theta_{L_k+1}^k$ is retained as a qualified boosted classifier; otherwise, $\theta_{L_k+1}^k$ is discarded. The above mentioned strategy provides an easily programmable approach for evaluating and computing the training error of the AdaBoost-HMM classifiers. Because the error rates are computed by comparing the probabilities scored by the individual HMMs, the above mentioned procedure shall be referred to as cross-validation.

4.3.3 Steps of the HMM AdaBoosting algorithm

The step-by-step procedures of HMM AdaBoosting for a K-class problem are given below:

1. For training set of Class d_k with R_k samples $X_k = \{x_1^k, x_2^k, \cdots, x_{R_k}^k\}$, initially, assign a uniform distribution $D_1(x_j^k) = 1/R_k, (\forall k = 1, 2, \cdots, K, j = 1, 2, \cdots, R_k)$ to $x_1^k, x_2^k, \cdots, x_{R_k}^k$. The boosting token k is initialized to be equal to 1.

2. Train a new HMM for the k-th class θ_t^k using the biased Baum-Welch algorithm with the distribution $D_t(x_j^k)$.

3. Formulate the binary hypothesis $h_t^k(x_j^k) \rightarrow \{-1, +1\}$ for θ_t^k using Eq.(4.11). The error rate of θ_t^k - ε_t^k, and the error rates of all the existing HMMs of other classes $\theta_l^j, (j = 1, 2, \cdots, K, j \neq k, l = 1, 2, \cdots, R_j)$ are estimated and verified using the cross-validation and Eq.(4.12). If the new model is valid, go on to Step 4; otherwise, the boosting token is passed to the next class $k = \begin{cases} k+1, & \text{if } k < K \\ 1, & \text{if } k = K \end{cases}$ and then Step 2 is repeated.

4. Calculate $w_t^k = \frac{1}{2}\ln(\frac{1-\varepsilon_t^k}{\varepsilon_t^k})$ for θ_t^k, where ε_t^k is the error rate. If the error rate of some existing HMM, say θ_l^j, changes, the corresponding $w_l^j = \frac{1}{2}\ln(\frac{1-\varepsilon_l^j}{\varepsilon_l^j})$ is also recomputed.

5. Update the distribution: $D_{t+1}(x_j^k) = \frac{D_t(x_j^k)}{Z_t} e^{-w_t^k h_t^k(x_j^k)} (h_t^k(x_j^k) = 1 \text{ or } -1)$, where Z_t is the normalization factor.

The above procedures terminate if the HMMs obtained for every class are invalid (see Step 3) or after a given number of boosting rounds. The convergence rate of an AdaBoost-HMM classifier is thus set equal to the number of boosting rounds. It should be noted that for different applications, the convergence rate may be very different, which depends on the distribution of the training samples and the similarities between different classes.

It can be seen from Step 4 that the smaller the value of ε_t^k, the larger the classifier weight w_t^k. Thus the AdaBoosting algorithm places more importance or weight on decisions made by more accurate HMMs. It is also observed in Step 5 that for a correctly classified sample, the sample weight $D_{t+1}(x_j^k)$ is reduced by a factor of $e^{-w_t^k}$ where as for an incorrectly classified sample, the sample weight increase by a factor of $e^{w_t^k}$. The "hard" samples are thus highlighted in the AdaBoost-HMM classifier.

4.3.4 Decision formulation

For an unknown input sequence y^T, its identity is determined by summing up the sub-decisions made by the composite HMMs of an AdaBoost-HMM classifier. Assume that Θ_k is the AdaBoost HMM classifier for Class $d_k, (k = 1, 2, \cdots, K)$ that has been "boosted" for L_k rounds. Then Θ_k consists of L_k HMMs - $\theta_1^k, \theta_2^k, \cdots, \theta_{L_k}^k$ and L_k weights - $w_1^k, w_2^k, \cdots, w_{L_k}^k$. The decision is formulated with one of the following three strategies:

Decision strategy 1: The hypothesis made on the identity of y^T: $h_t^k(y^T) \rightarrow \{+1, -1\}, (t = 1, 2, \cdots, L_k)$ is formulated by each composite HMM using Eq.(4.13). The hypotheses are synthesized via Eq.(4.14).

$$H(y^T|\Theta_k) = \frac{1}{L_k}\sum_{t=1}^{L_k} w_t^k h_t^k(t^T) \tag{4.14}$$

The decision is made by comparing $H(y^T|\Theta_k)$ for all the classes $d_k, (k = 1, 2, \cdots, K)$. The one that gives the maximum value is chosen as the identity of y^T.

$$ID(y^T) = \arg\max_k H(y^T|\Theta_k), \qquad 1 \leq k \leq K \tag{4.15}$$

The hypothesis $h_t^k(y^T) \rightarrow \{+1, -1\}$ is based on the *one versus the rest* dichotomy. This approach is featured with ease of implementation and is able to be applied to most recognition tasks. However, the $\{+1, -1\}$ hypothesis adopted in the strategy is not a fuzzy indicator. Sometimes we not only want to know the decision about a candidate pattern, but also hope to know the credibility of the decision made. To provide such information, decisions should be based on probabilities.

Decision strategy 2: The normalized log probability of y^T given Θ_k is defined as follows:

$$\bar{P}(y^T|\Theta_k) = \frac{1}{L_k}\sum_{t=1}^{L_k}\log[w_t^k P(y^T|\theta_t^k)] \tag{4.16}$$

where L_K is a normalization factor as different AdaBoost-HMM classifiers may have different number of composite HMMs. The final decision is made by comparing the $\bar{P}(y^T|\Theta_k)$ for all the classes ($k = 1, 2, \cdots, K$). The one that gives the maximum probability is chosen as the identity of y^T.

$$ID(y^T) = \arg\max_k \bar{P}(y^T|\Theta_k), \qquad 1 \le k \le K \tag{4.17}$$

This decision strategy synthesizes the probabilities scored by the composite HMMs rather than the sub-decisions as adopted in Strategy 1. As a result, the probability metric $\bar{P}(y^T|\Theta_k)$ can better indicate the credibility of the decision made. This decision strategy can be applied to most recognition tasks and is adopted in our viseme recognition experiments. An important property of the approach is that the synthesized probability can be accumulated to analyze a process that is modeled by a sequence of AdaBoost-HMM classifiers. The continuous visual speech modeling discussed in Chapter 5 is based on this decision strategy.

Decision strategy 3: The probabilities of y^T scored by the composite HMMs of Θ_k can also be synthesized using Eq.(4.18).

$$\bar{P}'(y^T|\Theta_k) = \frac{1}{\Sigma_k}\sum_{t=1}^{L_k}[w_t^k P(y^T|\theta_t^k)] \tag{4.18}$$

where $\Sigma_k = \sum_{t=1}^{L_k} w_t^k$. The identity of y^T is obtained by comparing $\bar{P}'(y^T|\Theta_k)$ for all the classes ($k = 1, 2, \cdots, K$).

$$ID(y^T) = \arg\max_k \bar{P}'(y^T|\Theta_k), \qquad 1 \le k \le K \tag{4.19}$$

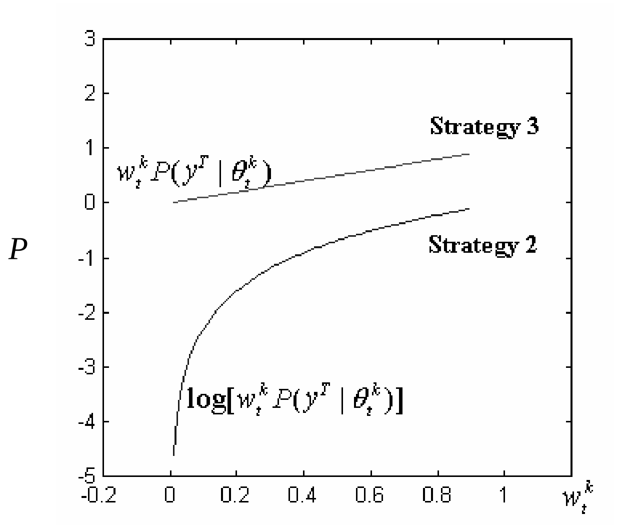

Figure 4.5: Change of the composite probabilities of Strategy 2 and Strategy 3

This decision strategy differs from Strategy 2 in two aspects. First, the log operation is not carried out in Eq.(4.18). Because $w_t^k > 0$ (which corresponds to $\varepsilon_t^k < 0.5$) and $P(y^T|\theta_t^k) \ll 1$, $\bar{P}(y^T|\Theta_k)$ may increase more sharply with respect to w_t^k than $\bar{P}'(y^T|\Theta_k)$ does, which is depicted in Fig.4.5. It thus concludes that in Strategy 3, the composite HMM with smaller weight (w_t^k) takes larger portion in the final decision than that in Strategy 2. In applications, the composite HMMs with smaller weights are usually trained by the "hard" samples. Strategy 3 thus highlights the hard samples and works better than Strategy 2 in the situation when the distribution of the samples is very much spread-out. Strategy 2, on the other hand, outperforms Strategy 3 when the samples are relatively clustered.

Second, the normalization factor adopted in Strategy 3 is Σ_k while L_k is used in Strategy 2. If boosting is carried out for only one round, the decision made by Strategy 3 is the same as that made by the single-HMM classifier, while this is not applicable to Strategy 2.

Strategy 3 also generates a probability $\bar{P}'(y^T|\Theta_k)$ for the input y^T. Like Strategy

2, it can also be applied to analyze a long process that is modeled by connected AdaBoost-HMM classifiers. In our other experiments concerning AdaBoost-HMM, Strategy 1 and Strategy 3 have been applied to speaker verification and acoustic speech recognition with some success. However, studies on these topics are not presented in this chapter as they are beyond the scope of this thesis.

4.3.5 Properties of HMM AdaBoosting

The principal idea of AdaBoosting is the improvement to the recognition accuracy of the “hard” samples or “outlier” samples of a class. For the proposed HMM AdaBoosting strategy, the probability scored for such hard samples increases as boosting continues, which in turn improves the acceptance rate of the hard samples. Because the hard samples usually indicate the decision boundaries of the classier, it may conclude that the AdaBoost-HMM classifiers can formulate more complex decision boundaries than using single-HMM classifiers.

While boosting HMMs, the decrease in error rate after each round may not be as fast as that of boosting other types of classifiers such as Neural Network. The reason is that the error rate of an HMM is obtained by comparing the likelihood value determined by HMM in the current round of boosting and those in the previous rounds, as explained in Section 4.3.2. With the propagation of HMMs, the classification error of the composite HMM will normally increase and hence the error of the AdaBoost-HMMs decreases at a low speed. Our experiments also reveal that the error surface with respect to the boosting round is generally smooth but sometimes may have bumps on it. The error rate, however, demonstrates the tendency to decrease as boosting continues.

For the AdaBoost-HMM classifiers of different classes, the number of the composite HMMs may be different because boosting for one class may terminate earlier than another class due to the cross-validation. However, such inequality does not affect the final decision because $H(y^T|\Theta_k)$, $\bar{P}(y^T|\Theta_k)$ and $\bar{P}'(y^T|\Theta_k)$ in Eq.(4.14), (4.16) and (4.18) are normalized with the number of HMMs in a class.

Another aspect of the proposed strategy that should be stressed is that the error rate computed by Eq.(4.12) only accounts for part of the classification error. It is the False Rejection Rate (FRR) that is discussed in Section 2.4.3, which indicates the samples in X_k being misclassified into some wrong category $d_j (j \neq k)$. Another source of the error is referred to as False Acceptance Rate (FAR), which indicates the samples of other classes $X_j (j \neq k)$ are erroneously accepted by Θ_k. However, FAR is not used in the proposed HMM AdaBoosting algorithm because it cannot work with the biased Baum-Welch estimation. As illustrated in Eq.(4.8) and (4.9), the biased Baum-Welch algorithm only uses the correct training samples for parameter estimation. An erroneously accepted sample cannot be applied to train the parameters of the HMMs. FRR can weight the correct training samples in the parameter estimation equations as it is an indicator of the goodness-of-fit of the correct samples, while FAR, which is a statistical measure for the incorrect samples (irrelevant to the correct samples), cannot be applied for the proposed method. This is why FAR is not computed in the boosting steps given in Section 4.3.3. If other base training algorithm, such as MMI estimation, is applied in HMM boosting, where the erroneously accepted samples are also used to train the parameters of the HMM, both FRR and FAR may be considered.

4.4 Performance of the AdaBoost-HMM classifier

Experiments are carried out to access the performance of the AdaBoost-HMM classifier in recognition of visemes defined in MPEG-4. Comparison is also made with the performance of the single-HMM classifiers.

4.4.1 Experiment 1

In the first experiment, 40 context-independent samples of a viseme are used as the training samples and the left 100 are used as the testing samples. Note that the training samples and testing samples are the same as those used in the single-HMM classifier discussed in Section 2.4.4. The HMMs are initialized with the approach given in Section 2.4.3. For each context-independent viseme, an AdaBoost-HMM classifier that consists of 15 to 20 HMMs is trained with the strategy given in Section 4.3.3. According to Eq.(4.17), for a testing sample y^T of Class d_k, a correct classification is made if $ID(y^T) = d_k$. The classification error (FRR) of the AdaBoost-HMM classifier Θ_k is then computed using Eq.(4.20), which is originated from Eq.(2.17) and (2.18).

$$FRR(\Theta_k) = 1 - \frac{\text{number of correctly classified samples of } d_k}{\text{number of all the testing samples of } d_k} \tag{4.20}$$

The classification errors of the AdaBoost viseme classifiers are listed in Table 4.1. The FRRs of the single-HMM classifiers (θ^2 of Speaker 1) trained in Section 2.4.4 are also listed in Table 4.1 for comparison.

From Table 4.1, it can be seen that both the single-HMM classifiers and AdaBoost-HMM classifiers can identify the context-independent visemes with reasonable accuracy. An average classification error below 20% is obtained with either approach. The classification error is lower for the vowels as the movement of the lips is more. The performance of the AdaBoost-HMM classifiers and that of the single-HMM classifiers are not significantly different. The samples obtained for the context-independent visemes demonstrate good homogeneity as they are independently produced.

Table 4.1: Classification errors (FRR) of the single-HMM classifier and the AdaBoost-HMM classifier in recognition of context-independent visemes

Viseme	Classification Error		Viseme	Classification Error	
No.	Single-HMM	AdaBoost-HMM	No.	Single-HMM	AdaBoost-HMM
1 p, b, m	13%	8%	2 f, v	4%	5%
3 T, D	11%	4%	4 t, d	35%	17%
5 k, g	24%	9%	6 tS, dZ, S	10%	10%
7 s, z	4%	4%	8 n, l	19%	20%
9 r	18%	7%	10 A:	1%	4%
11 e	8%	11%	12 I	1%	4%
13 Q	7%	10%	14 U	7%	9%
Average	11.6%	8.7%			

4.4.2 Experiment 2

In the second experiment, 100 context-dependent samples of a viseme are used to train the AdaBoost-HMM classifier and the left 100 context-dependent samples are used to test the performance of the classifier.

As mentioned in Section 4.3.5, the training error of an AdaBoost-HMM classifier [which is computed via Eq.(4.12)] shows tendency of decreasing during boosting. To illustrate such change, the training errors of four AdaBoost-HMM classifiers and the *t*-th HMM (the HMM trained in the *t*-th Boosting round) of the classifiers are depicted in Fig.4.6. Take Fig.4.6(a) as an example, the training error of the AdaBoost-HMM classifier decreases from 0.27 to 0.13 for the 20 rounds while the training error of the *t*-th HMM increases from 0.27 to 0.45. This is so as the composite HMM biases more and more to the outlier samples as boosting continues. For the experiments carried out, the error rate of the composite HMM increases

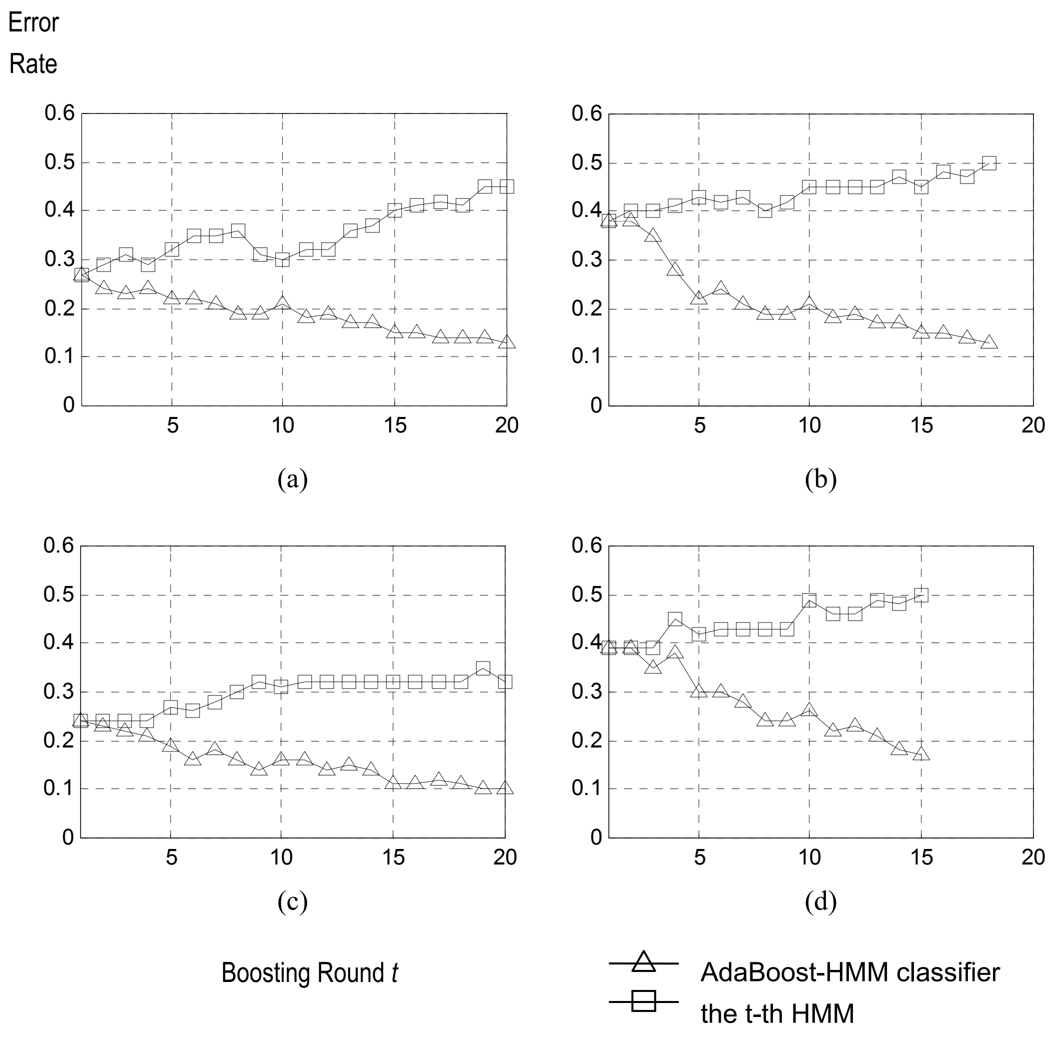

Figure 4.6: Rate of training error versus boosting round.
Viseme classifiers of (a) /e/ (b) /s, z/ (c) /T, D/ (d) /t, d/

to 0.5 after about 12 to 20 Boosting rounds. It may thus conclude that AdaBoost viseme classifier converge after 12∼20 rounds. Such speed of convergence is acceptable as the biased Baum-Welch estimation is carried out for 12∼20 times. The training errors and classification errors (FRR) of the testing samples computed

Table 4.2: Training errors and classification errors (FRR) of the single-HMM classifiers and the AdaBoost-HMM classifiers

Viseme No.	Single-HMM Classifier		AdaBoost-HMM Classifier	
	Training Error	Classification Error	Training Error	Classification Error
1 p, b, m	14%	20%	6%	15%
2 f, v	15%	27%	11%	25%
3 T, D	24%	34%	10%	18%
4 t, d	39%	50%	17%	19%
5 k, g	17%	17%	16%	16%
6 tS, dZ, S	17%	21%	5%	9%
7 s, z	39%	45%	13%	17%
8 n, l	40%	79%	22%	33%
9 r	21%	54%	22%	37%
10 A:	14%	18%	5%	5%
11 e	27%	33%	13%	7%
12 I	9%	10%	0%	2%
13 Q	10%	35%	4%	11%
14 U	4%	9%	1%	7%
Average	21%	32%	10%	16%

for the AdaBoost-HMM classifiers are listed in Table 4.2 together with the corresponding errors using the single-HMM classifiers (θ^2 of Speaker 1 as mentioned in Table 2.3). It is observed that the accuracy of recognition is significantly improved (smaller error rate) using AdaBoost-HMM classifier.

Compared with the classification errors listed in Table 4.1, the classification accuracy of the single-HMM classifiers decreases dramatically in identifying context-dependent visemes. The classification rates of some consonants are even less than 50%. The reason underlying the high identification error is the distribution of the samples. As the samples of a context-dependent viseme are extracted from various words (contexts), the "shapes" of the samples of even the same viseme are different. In statistics jargon, the samples of a context-dependent viseme demonstrate a spread-out distribution. The traditional single-HMM classifiers cannot cover such a distribution well. However, the classification accuracy can be greatly improved with the application of AdaBoost-HMM classifiers. As illustrated in Table 4.2, although the classification errors are larger compared with those listed in Table 4.1, an average recognition accuracy of 70%∼80% is still attainable, which is about 16% better than the single-HMM classifiers. The improvement can be attributed to the fact that a more complex decision boundary is formulated using the AdaBoost-HMM classifier than using the single-HMM classifier. Therefore, the AdaBoost-HMM classifiers can better cover the spread-out distribution for both the testing samples and the training samples. This is validated by the experimental results. If the context-dependent visemes are looked as isolated visemes distorted by adjoining visemes, it is concluded that the AdaBoost-HMM classifiers provide better robustness on identifying visemes than single-HMM classifiers.

4.4.3 Computational load

The complexity of computation of the HMM AdaBoosting strategy is closely associated with the number of boosting rounds. If an AdaBoost-HMM classifier is boosted for t rounds, the computations involved are approximately t-times of that training a ML HMM classifier.

Take the 3-state 128-symbol discrete HMM applied in the experiments as an example. If the number of training samples for each viseme is 100 and the samples range from 15∼50 frames, about 10^6 computations (computations include multiply and division) are required to train an HMM with the biased Baum-Welch estimation. Most of the computations are for calculating the forward variables and backward variables. For example, if the length of a training sample is 30 frames (average length), about 1200 multiplies are required to build a probability trellis to compute the forward variables and backward variables. Assume that all the 100 training samples have the same length of 30 frames, about 1.2×10^5 computations are required to compute these variables. The forward variables and backward variables have to be computed more than once because EM iterations are taken in the Baum-Welch estimation. If ten iterations are used (in our experiments, the probability scored for the training samples becomes stable after about 10 iterations), 1.2×10^6 computations are required. The number of computations involved in estimating the state transition coefficients and symbol emission probabilities is small compared with this quantity. As a result, it is reasonable to conclude that computations of the order of 10^6 are required to train a single-HMM classifier. The computations can be completed in less than 10 seconds using Pentium III-900MHz.

For AdaBoost-HMM classifier comprising 20 HMMs, approximately 2×10^7 computations are required for training the classifier. The additional number of computations including error estimation using cross-validation and calculation of weights is small compared with the number of computations required for the Baum-Welch estimation. The total number of computations is thus approximately 2×10^7. This is a modest amount of computations for the modern computers or signal processor chips. The average time for training the AdaBoost-HMM classifier is about 2 minutes using Pentium III-900MHz.

The computational load in the recognition phase is far less than that in the training phase. For the single-HMM classifiers, the likelihood of an input sequence is computed using the forward process. If the input sequence is 30-frame in length, about 1200 multiplies are performed. As the identity of the input sequence is determined out of fourteen viseme categories, the total number of computations will be approximately 1.7×10^4. It only takes 1 second using Pentium III-900MHz. For the AdaBoost-HMM classifiers, the input sequence is evaluated by a total of 280 (20×14) HMMs, where about 3.4×10^5 multiplies are carried out. The number of computations involved in probability synthesis is small compared with this quantity. It takes 4$\sim$5 seconds to determine the identity of the input sequence using the Pentium III 900 computer.

4.5 Summary

The Adaptively Boosted HMM classifier proposed in this chapter is a type of multi-HMM classifiers. It is specially suited to improve the accuracy of identifying time series with spread-out distribution. The Adaptively Boosting technique serves to train a group of undemanding HMMs, in which the training error is less than 0.5. The decisions made by the composite HMMs are synthesized and a more complex decision boundary is formulated than using single HMM.

In this thesis, the AdaBoost-HMM classifier is applied to identify visemes in visual speech processing. Experimental results show that such a classifier is able to achieve an average improvement of 16% on recognition accuracy over the traditional single HMM classifier in identifying context-dependent visemes. The complexity of computational load to train the boosted viseme classifier is approximately twenty times that of the load of training a single-HMM viseme classifier when the Baum-Welch estimation is used.

The proposed method can readily be extended to many other situations especially when the observed data have spread-out distribution, for example, speech recognition, handwriting recognition and speaker identification.

Chapter 5

Visual Speech Modeling Using Connected Viseme Models

Visual speech processing can be considered at two levels: the first level processing is to identify the basic visual speech elements such as visemes; the second level processing is to recognize connected-viseme units. The approaches mentioned in Chapter 2, 3 and 4 are first level processing where the individual visemes are recognized. In this chapter, the strategies of connecting HMM-based classifiers are proposed to model connected-viseme units in visual speech.

5.1 Constituent element and continuous process

Recall the properties of HMM modeling presented in Chapter 2, it is seen that an HMM can be trained to model a stochastic process of arbitrary length. For our system, however, an HMM is trained to model relatively short sequence such as a viseme. The long sequences, on the other hand, are to be modeled by connected-HMM models. For ease of description, the short sequences that may be repeated in the long sequences are referred to as constituent elements, and the long sequences are referred to as continuous processes in this thesis.

For the specific application of visual speech processing, the visemes are looked as the constituent elements while words, phrases and sentences are treated as continuous processes. In visual speech processing, it is also desirable to model the constituent elements (visemes) that may be repeated in the continuous processes (words, phrases and sentences) using HMM rather than modeling the continuous processes directly. The advantages of such management are described below:

1.) The number of constituent elements is usually limited while the number of continuous processes may be large because the possible number of combinations of the constituent elements is large. It thus takes many more classifiers to model the continuous processes than the constituent elements. For example, in speech, the subword elements such as phonemes, consonant-vowel-consonant syllables (CVC) and vowel-consonant-vowel syllables (VCV) are usually treated as the constituent elements [65]. The number of these speech elements is of the order of hundred while the vocabulary of the spoken words and phrases that are made up of these constituent elements is huge.

2.) It is impractical to find sufficient number of training samples for the continuous processes while it is much easier to do so for the constituent elements. For example, in some speech recognition systems, the speaker is asked to read the training text that contains hundreds of words, several times. The samples drawn from the text may be insufficient to train the word models appearing in the text but are usually sufficient to train models of the constituent elements such as phonemes.

3.) In HMM-based speech processing, building models for the constituent elements is also necessary for text-independent applications and applications involving a large vocabulary. By modeling the subword elements, the speech recognition system is able to make a good assessment of any words that the systems were not previously trained.

For the proposed methods, the constituent elements are modeled by HMM-based classifiers and continuous processes are modeled by connected classifiers.

The approaches proposed in this chapter are chiefly based on the level building algorithm and the classifiers adopted to model the constituent elements include the ML HMM classifiers mentioned in Chapter 2 and the AdaBoost-HMM classifiers mentioned in Chapter 4. The two-channel HMM classifier and the MSD HMM classifier, however, are not studied for modeling continuous processes because both classifiers are specially tailored to differentiate confusable samples. While dealing with a long sequence in visual speech, it is difficult to model the sequence using a chain of two-channel HMMs or MSD HMMs. For this reason, the discussion of HMM-based continuous visual speech processing focuses on the ML HMM classifiers and AdaBoost-HMM classifiers.

5.2 Level building on ML HMM classifiers

The approaches of connecting HMMs are based on dynamic programming, which include level-building algorithm [99] and frame-synchronous search algorithm [109]. Level building on HMM is a traditional method of searching/determining a sequence of HMMs to model a long process. It employs a probability trellis to synchronize the HMMs with the segments of the target sequence. In the history of dynamic time warping technique for speech processing, the level building method plays an important role because it is theoretically sound and easily programmable. In recent years, however, the level building method is gradually substituted by frame-synchronous method. The frame-synchronous approach applies a grammar network to search for the best-matched paths/models that account for sub-sequences terminating at any frame. This method is capable of synthesizing complex syntactic constrains (grammar) to the statistical framework of HMM modeling and is thus more efficient at realizing large-vocabulary speech recognition than the level building method. However, the implementation of frame-synchronous searching is tedious because some cost measures such as distances, likelihood and penalty terms have to be computed. In this thesis, the discussion is focused on the level building method since only a small vocabulary is involved in the recognition task and grammatical rules are not mandatory. The probability accumulation strategy of level building method is also fundamental to frame-synchronous approaches. In the paragraphs that follow, the principles of level building are given.

Let $\Theta = \{\theta_1, \theta_2, \cdots, \theta_K\}$ denote the set of HMMs of K constituent elements (one constituent element is modeled by one HMM). For T-length observation sequence of a continuous process, say $y^T = (o_1, o_2, \cdots, o_T)$, the purpose of level building is to decode an η-HMM chain $\Theta^\eta = (\theta_{1,t_1}, \theta_{2,t_2}, \cdots, \theta_{\eta,t_\eta}), \theta_{i,t_i} \in \Theta(i = 1, 2, \cdots, \eta)$ to maximize the likelihood $P(y^T|\Theta^\eta)$, where θ_{i,t_i} is the i-th sequence with starting frame $t_i, (i = 1, 2, \cdots, \eta)$. The number of constituent elements in an observation sequence is usually unknown. In level building algorithm, this problem is solved by partitioning the target sequence into different "levels". The level corresponds with the location of the constituent element in the sequence and the number of the levels equals to the number of constituent elements. Level building on HMM is carried out with the following steps.

5.2.1 Step 1: Construct the probability trellis

From Level 1 to Level η_{max}(the maximum number of the levels that may appear in y^T), a probability trellis is built as illustrated in Fig.5.1. At the node at Frame t on Level η, denoted as Node(t, η), $P_a(t_\eta, t, \eta, \theta_k)$, the accumulated probability along the best path (also called Viterbi path, which will be discussed in Step 2) for reference model $\theta_k, (k = 1, 2, \cdots, K)$, is computed.

$$P_a(t_\eta, t, \eta, \theta_k) = P_{best}(t_\eta - 1, \eta - 1) \times P(o_{t_\eta}, o_{t_\eta+1}, \cdots, o_t|\theta_k) \tag{5.1}$$

where t_η is the starting frame of Level η, $P_{best}(t_\eta - 1, \eta - 1)$ is the accumulated probability of the previous $\eta - 1$ levels, terminating at Frame $t_\eta - 1$, and $P(o_{t_\eta}, o_{t_\eta+1}, \cdots, o_t|\theta_k)$ is the probability of the sub-sequence $(o_{t_\eta}, o_{t_\eta+1}, \cdots, o_t)$ scored by θ_k at Level η. Because t_η is the starting frame of Level η, $t_\eta - 1$ is the end frame of Level $\eta - 1$. The computation of $P_{best}(t_\eta - 1, \eta - 1)$ is given in Step 2. The starting frame of $P_a(t_\eta, t, \eta, \theta_k)$ is retained as $F(t, \eta, \theta_k) = t_\eta$ for back-tracking the HMMs. The probability $P(o_{t_\eta}, o_{t_\eta+1}, \cdots, o_t|\theta_k)$ can be computed via the standard forward process [99]. If $\eta = 1$ in Eq.(5.1), which means no previous level accounts for the accumulated probability, we have,

$$P_a(t_1 = 1, t, \eta = 1, \theta_k) = P(o_1, o_2, \cdots, o_t|\theta_k) \tag{5.2}$$

This term is also computed via the forward process.

5.2.2 Step 2: Accumulate the probabilities

If Frame t is the end frame of Level η, maximization over Θ is performed to get the best-matched HMM at Level η, starting from Frame t_η and terminating at Frame t.

$$P_{best}(t_\eta, t, \eta) = \max_{\theta_k \in \Theta} P_a(t_\eta, t, \eta, \theta_k), \qquad k = 1, 2, \cdots, K \tag{5.3}$$

The starting frame t_η is also a variable. For Level η, it ranges within the scope $t_{\eta-1} < t_\eta < t$, where $t_{\eta-1}$ is the starting frame of Level $\eta - 1$. Maximization over all the possible starting frames t_η is performed to obtain $P_{best}(t, \eta)$, which indicates the largest probability accumulated to Frame t, Level η, regardless of what the composite HMMs $\theta_{1,t_1}, \theta_{2,t_2}, \cdots, \theta_{\eta,t_\eta}$ are and where the starting frames are.

$$P_{best}(t_\eta, \eta) = \max_{\forall t_\eta} P_{best}(t_\eta, t, \eta), \qquad t_{\eta-1} < t_\eta < t \tag{5.4}$$

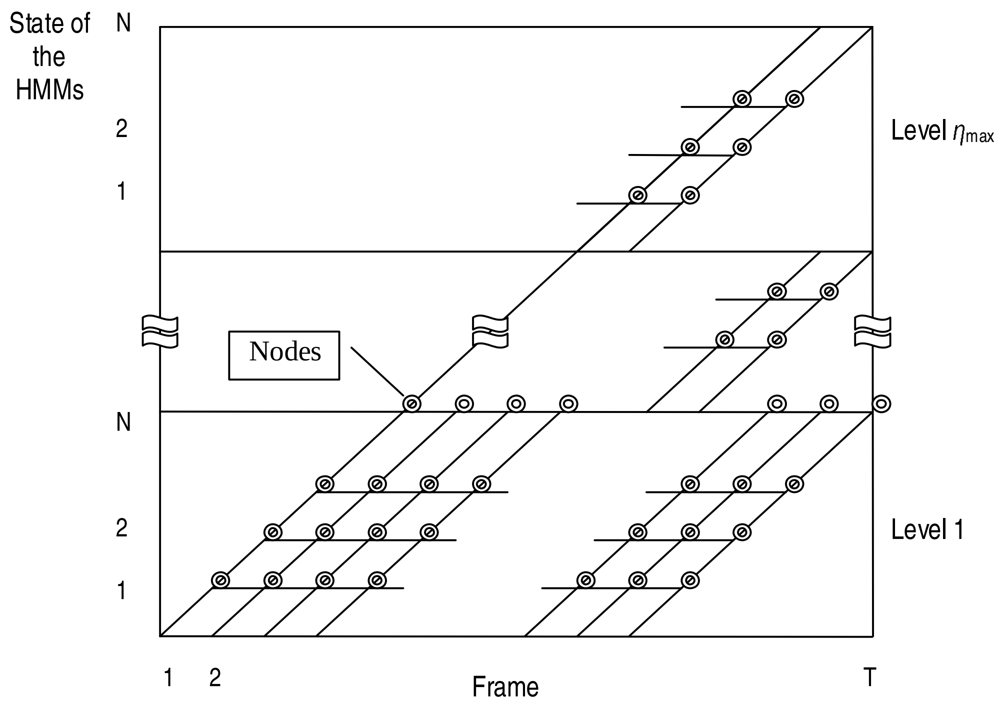

Figure 5.1: The probability trellis in level building on HMM

The index of the best-matched HMM is also computed as in Eq.(5.5).

$$\theta_{best}(t,\eta) = \arg\max_{\forall\theta_k\in\Theta} P_{best}(t,\eta), \qquad k = 1,2,\cdots,K \tag{5.5}$$

And the starting frame of the best-matched HMM is

$$F_{best}(t,\eta) = F(t,\eta,\theta_{best}(t,\eta)) \tag{5.6}$$

While at Level 1, $P_{best}(t,\eta=1)$ is obtained via Eq.(5.7) because $t_1 = 1$ is a constant value.

$$P_{best}(t,1) = \max_{\forall\theta_k\in\theta} P_a(1,t,1,\theta_k), \qquad k = 1,2,\cdots,K \tag{5.7}$$

In the probability trellis given in Fig.5.1, the indexes and the starting frames of the best-matched HMMs build the best path, which is referred to as Viterbi path. In Step 1 mentioned above, $P_{best}(t_\eta - 1, \eta - 1), (\eta \geq 2)$ that appears in Eq.(5.1) is computed using Eq.(5.4).

5.2.3 Step 3: Backtrack the HMM sequence

If the observation sequence y^T is decomposed into $\eta, (\eta \leq \eta_{max})$ levels, $P_{best}(T, \eta)$ is the greatest probability for the HMM chain with η reference HMMs. The best-matched HMM $\theta_{\eta,t_\eta} = \theta_{best}(T, \eta)$ at Level η is determined using Eq.(5.5), and the starting frame of θ_{η,t_η}, $F_{best}(T, \eta)$, is located using Eq.(5.5). Thus the end frame of Level $\eta - 1$ is $F_{best}(T, \eta) - 1$. By repeating Eq.(5.5) and Eq.(5.6) until Level 1, the optimal HMM chain $\Theta^\eta = (\theta_{1,t_1}, \theta_{2,t_2}, \cdots \theta_{\eta,t_\eta})$ with η HMMs is decoded and the positions of the HMMs are located.

The level building on HMM method has been applied to identify a selected number of words and phrases in visual speech. The experimental results are presented later in Section 5.4.3 and are compared with the connected AdaBoost-HMM classifiers.

5.3 Level building on AdaBoost-HMM classifiers

The experiments conducted in Chapter 3 indicate that AdaBoost-HMM classifiers provide better accuracy than the conventional ML HMM classifiers on recognizing individual visemes. In this section, the application of AdaBoost-HMM classifier is extended by means of level building algorithm to model continuous processes. It is expected that the connected AdaBoost-HMM classifiers may outperform connected ML HMM classifiers while modeling and identifying connected-viseme units in visual speech. Before going into the details of the proposed method, the process of computing the probability of a sub-sequence by an AdaBoost-HMM classifier is briefed.

For an AdaBoost-HMM classifier Θ_k that comprises L_k composite HMMs, the underlying process of computing the probabilities is depicted in Fig.5.2, where $x^\tau = (o_{t+1}, o_{t+2}, \cdots o_{t+\tau})$ is a sub-segment of the sequence $y^T = (o_1, o_2, \cdots o_T)$. Note that y^T is an observation of a continuous process and x^τ is looked as an observation of a constituent element of y^T. S_0 is a null state indicating the beginning of x^τ and S_e is another null state indicating the end of x^τ. The composite HMMs $\theta_1^k, \theta_2^k, \cdots \theta_{L_k}^k$, which are affiliated to Class d_k, are connected in parallel as illustrated in Fig.5.2. For θ_l^k, the probability scored from Frame $t+1$ to $t+t'$, say $P_l^k(t') = P(o_{t+1}, o_{t+2}, \cdots o_{t+t'}|\theta_l^k)$, is computed using the forward variables.

$$P_l^k(t') = \sum_{i=1}^{N} \alpha_{t'}(i) \tag{5.8}$$

where $\alpha_{t'}(i) = P(o_{t+1}, o_{t+2}, \cdots o_{t+t'}, s_{t+t'} = S_i|\theta_l^k)$ and $s_{t+t'}$ is the t'-th state in $(s_{t+1}, s_{t+2}, \cdots s_{t+\tau})$, which is the state sequence decoded for x^τ. The probability of the sub-sequence x^τ given the HMM θ_l^k is computed by

$$P(x^\tau|\theta_l^k) = P_l^k(\tau+1) = \sum_{i=1}^{N} \alpha_\tau(i) P(S_e|S_i) \tag{5.9}$$

where $P(S_e|S_i)$ is the probability of transiting from State S_i to the end state S_e. To connect the AdaBoost HMM classifiers to model a continuous process, the critical problem is to align the composite HMMs so that they begin and end at the same frame.

Level building of HMMs is carried out to search a chain of AdaBoost-HMM classifiers $\Theta^\eta = (\Theta_{1,t_1}, \Theta_{2,t_2}, \cdots \Theta_{\eta,t_\eta})$ to match the target sequence y^T, where $\Theta_{i,t_i} \in \{\Theta_1, \Theta_2, \cdots \Theta_K\}$. The composite HMMs of an AdaBoost HMM classifier are synchronized with some special processing. Before implementing level building, the number of constituent elements that appear in y^T, or equivalently the number of levels, denoted by η , is first estimated. We assume that $\eta_{min} \leq \eta \leq \eta_{max}$. In addition to η, the durations of the constituent elements are also estimated. For the constituent element of Class d_k, which is modeled by an AdaBoost HMM classifier Θ_k, the duration (or number of frames) is assumed to be within the range $[d_{min}^k, d_{max}^k]$. Such an assumption is reasonable in most applications. For example, if a phoneme is treated as a constituent element, its duration is most likely to be

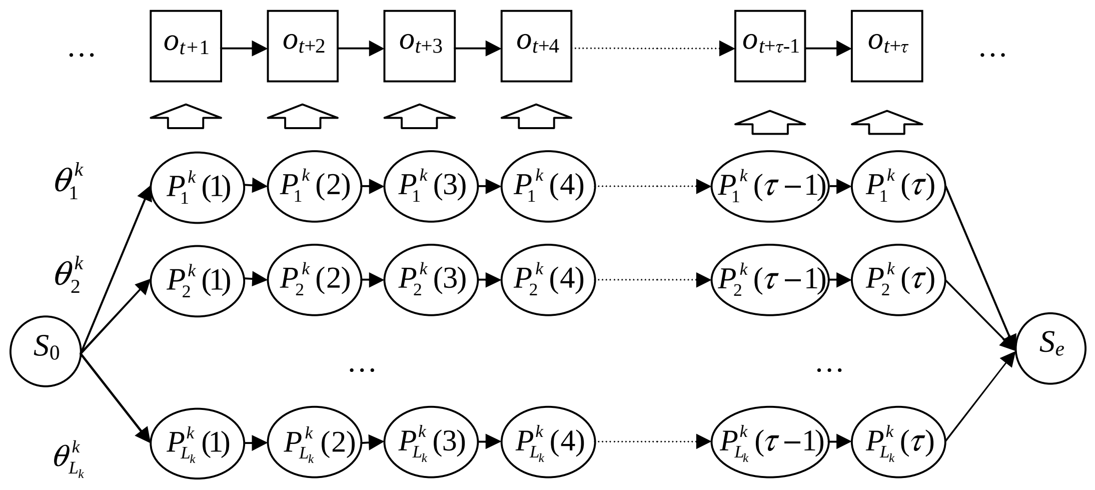

Figure 5.2: The underlying process of computing the probabilities by the composite HMMs of an AdaBoost-HMM classifier

within [0.2s, 0.8s]. It is observed that some prior knowledge about the number of the levels and the durations of the constituent elements greatly facilitates the computation.

5.3.1 Step 1: Probabilities computed at the nodes

A probability trellis as shown in Fig.5.3 is constructed. The topological lattice of the trellis is similar to that given in Fig.5.1. However, the underlying process of computing the probabilities is different from level building on single HMM. In Fig.5.3, the nodes at the end of a level are called end nodes because some additional processing is performed at these locations. The probability trellis is constructed as follows: starting from Node$(1, 1)$, the accumulated log probabilities of each composite HMM are computed at each node (including end node). For example, at Node(t, η), the probability $P_a(t_\eta, t, \eta, \theta_l^k)$ is computed for θ_l^k, where t_η is the starting frame of Level η. $P_a(t_\eta, t, \eta, \theta_l^k)$ is the sum of two entries as follows.

$$P_a(t_\eta, t, \eta, \theta_l^k) = \bar{P}_{best}(t_\eta - 1, \eta - 1) + \log P(t_\eta, t, \eta, \theta_l^k) \tag{5.10}$$

where $\bar{P}_{best}(t_\eta - 1, \eta - 1)$ is the optimal synthesized accumulated log probability of the previous $\eta - 1$ levels, terminating at Frame $t_\eta - 1$. Note that the term "synthesized" is applied to describe $\bar{P}_{best}(t_\eta - 1, \eta - 1)$, which is different from $P_{best}(t_\eta - 1, \eta - 1)$ in the level building on single HMMs discussed in Section 5.2. Computation of $\bar{P}_{best}(t_\eta - 1, \eta - 1)$ is given in Step 2. The second term in Eq.(5.10), $P(t_\eta, t, \eta, \theta_l^k)$, is the probability of the sub-sequence $(o_{t_\eta}, o_{t_\eta+1}, \cdots o_t)$ scored by θ_l^k.

$$P(t_\eta, t, \eta, \theta_l^k) = P(o_{t_\eta}, o_{t_\eta+1}, \cdots o_t | \theta_l^k), \qquad t - d_{max}^k \leq t_\eta \leq t - d_{min}^k \tag{5.11}$$

$P(t_\eta, t, \eta, \theta_l^k)$ is computed using Eq.(5.8). The starting frame of the sub-sequence t_η may vary between $t - d_{max}^k$ to $t - d_{min}^k$. As a result, $(d_{max}^k - d_{min}^k + 1)$ probabilities have to be computed for θ_l^k at Node(t, η).

5.3.2 Step 2: Probability synthesizing and HMM synchronizing at the end nodes

If Node(t, η) corresponds to an end node, the probabilities scored by the composite HMMs are synthesized using Eq.(5.12), which is similar to Decision Strategy 2 presented in Section 4.3.4.

$$\bar{P}(t_\eta, t, \eta, \Theta_k) = \frac{1}{L_k} \sum_{l=1}^{L_k} \log\left(w_l^k P(t_\eta, t, \eta, \theta_l^k)\right), \quad t - d_{max}^k \leq t_\eta \leq t - d_{min}^k \tag{5.12}$$

where w_l^k is the weight assigned to θ_l^k. In Eq.(5.12), t_η also ranges from $t - d_{max}^k$ to $t - d_{min}^k$. As a result, $(d_{max}^k - d_{min}^k + 1)$ synthesized log probabilities for Θ_k are computed at Node(t, η). The composite HMMs thus start from the same frame t_η and terminate at the same frame t. For each AdaBoost HMM classifier, say $\Theta_k, (k = 1, 2, \cdots, K)$, such an array of probabilities are retained.

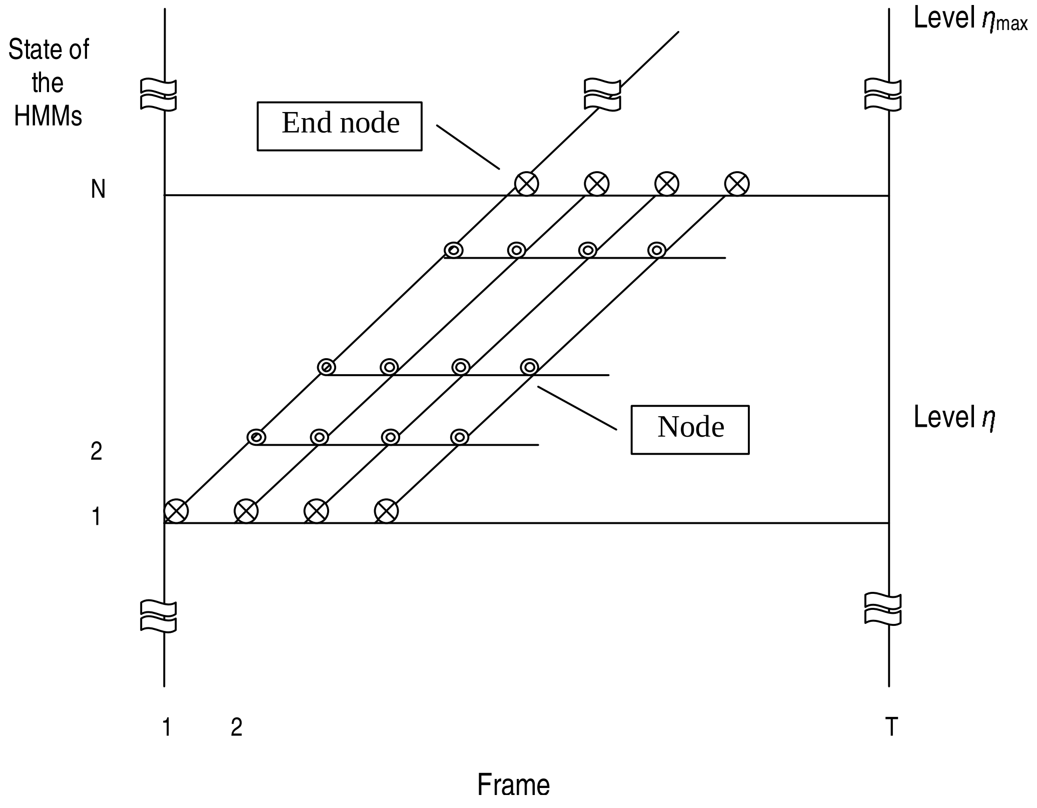

Figure 5.3: Topological lattice of level building on AdaBoost-HMM classifiers

Let $\bar{P}_a(t, \eta, \Theta_k)$ denote the accumulated synthesized log likelihood at Node(t, η), with the reference model Θ_k. This entry is computed using Eq.(5.13).

$$\bar{P}_a(t, \eta, \Theta_k) = \max_{t_\eta}(\bar{P}_{best}(t_\eta - 1, \eta - 1) + \bar{P}(t_\eta, t, \eta, \Theta_k)) \tag{5.13}$$

where $t - d^k_{max} \leq t_\eta \leq t - d^k_{min}$. $\bar{P}_{best}(t_\eta - 1, \eta - 1)$ is the optimal synthesized log likelihood accumulated to Node$(t_\eta - 1, \eta - 1)$, regardless of what the decoded AdaBoost-HMM classifiers are and where the starting frames of the classifiers are. A general method for computing $\bar{P}_{best}(t, \eta)$, which is the largest synthesized log likelihood accumulated to Node(t, η), is the maximization of $\bar{P}_a(t, \eta, \Theta_k)$ over all the reference models $\{\Theta_1, \Theta_2, \cdots \Theta_K\}$.

$$\bar{P}_{best}(t,\eta) = \max_{\Theta_k} \bar{P}_a(t,\eta,\Theta_k), \qquad k = 1, 2, \cdots, K \tag{5.14}$$

At Level 1, we have

$$\bar{P}_{best}(t,1) = \max_{\Theta_k} (\bar{P}(1,t,\eta=1,\Theta_k)), \qquad k = 1, 2, \cdots, K \tag{5.15}$$

At any node, $\bar{P}_a(t,\eta,\Theta_k)$ and $\bar{P}_{best}(t,\eta)$ can be computed in a recursive manner using Eq.(5.10),(5.13),(5.14),(5.15). Let $F(t,\eta,\Theta_k)$ denote the starting frame of $\bar{P}_a(t,\eta,\Theta_k)$ at Level η, we have,

$$F(t,\eta,\Theta_k) = \arg\max_{t_\eta}(\bar{P}_{best}(t_\eta - 1, \eta - 1) + \bar{P}(t_\eta, t, \eta, \Theta_k)), \quad t_{min} \leq t_\eta \leq t_{max} \tag{5.16}$$

$F(t,\eta,\Theta_k)$ is retained as the backpointer for path finding in Step 3. The index of the best-matched reference model is also retained at Node(t,η).

$$\Theta_{\eta,t_\eta} = \Theta_{best}(t,\eta) = \arg\max_{\Theta_k} \bar{P}_a(t,\eta,\Theta_k), \qquad \Theta_k \in \{\Theta_1, \Theta_2, \cdots, \Theta_K\} \tag{5.17}$$

5.3.3 Step 3: Path backtracking

With the above probability trellis, backpointers and classifier indices, the optimal sequence of the AdaBoost-HMM classifiers is backtracked for y^T with some simple processing. At a possible level number $\eta, (\eta_{min} \leq \eta \leq \eta_{max})$ and Frame T being the end frame of Level η, the best-matched reference model at Level η, $\Theta_{\eta,t_\eta} = \Theta_{best}(T,\eta)$, is decoded from Eq.(5.17). Using Eq.(5.16), the optimal starting frame of Level η, $F(T,\eta,\Theta_{best}(T,\eta))$ is obtained. The end frame of Level $\eta - 1$ thus is $F(T,\eta,\Theta_{best}(T,\eta)) - 1$ and the best-matched reference model at Level $\eta - 1$, $\Theta_{\eta-1,t_{\eta-1}}$, is decoded by applying Eq.(5.17) again. The above backtracking iteration continues until Level 1, Frame 1 is reached. A sequence of best-matched AdaBoost HMM classifiers $\Theta^\eta = (\Theta_{1,t_1}, \Theta_{2,t_2}, \cdots \Theta_{\eta,t_\eta})$ is determined for y^T. The log probability of y^T scored by this best sequence of classifiers is computed using Eq.(5.14) and denoted as $\bar{P}_{best}(T, \eta)$.

Because the number of levels η varies within the range $[\eta_{min}, \eta_{max}]$, a sequence of reference models $\Theta^\eta = (\Theta_{1,t_1}, \Theta_{2,t_2}, \cdots \Theta_{\eta,t_\eta})$ is decoded at every possible value of η. Thus $(\eta_{max} - \eta_{min} + 1)$ sequences are obtained. Some systems take all these sequences as the possible pattern candidates while others only require one sequence. The best sequence of AdaBoost HMM classifiers is obtained by maximizing over all the possible level numbers as in Eq.(5.18),

$$\Theta^\eta_{best} = \arg\max_{\Theta^\eta} \bar{P}_{best}(T, \eta), \qquad \eta_{min} \leq \eta \leq \eta_{max} \tag{5.18}$$

And the optimal number of levels η_{best} is obtained from Eq.(5.19)

$$\eta_{best} = \arg\max_{\eta} \Theta^\eta_{best} \tag{5.19}$$

5.3.4 Simplifications on building the probability trellis

It is observed from the above steps that only the accumulated probabilities at the end nodes are necessary for the construction of the optimal reference models. As a result, the procedures of constructing the probability trellis can be simplified if we focus on the properties of the end nodes. For example, at the end node Node(t, η), the probability of the sub-sequence $(o_{t_\eta}, o_{t_\eta+1}, \cdots o_t)$ given the composite HMM $\theta^k_l \in \Theta_k$ can be computed using the backward variables. For θ^k_l, the backward variable at Frame t_η is defined in Eq.(5.20).

$$\beta_{t_\eta}(i) = P(o_{t_\eta}, o_{t_\eta+1}, \cdots o_t | s_{t_\eta} = S_i, \theta^k_l) \tag{5.20}$$

Thus we have,

$$P(t_\eta, t, \eta, \theta^k_l) = \sum_{i=1}^{N} (P(s_{t_\eta} = S_i | s_{t_\eta - 1} = S_0) b_i(o_{t_\eta}) \beta_{t_\eta}(i)) \tag{5.21}$$

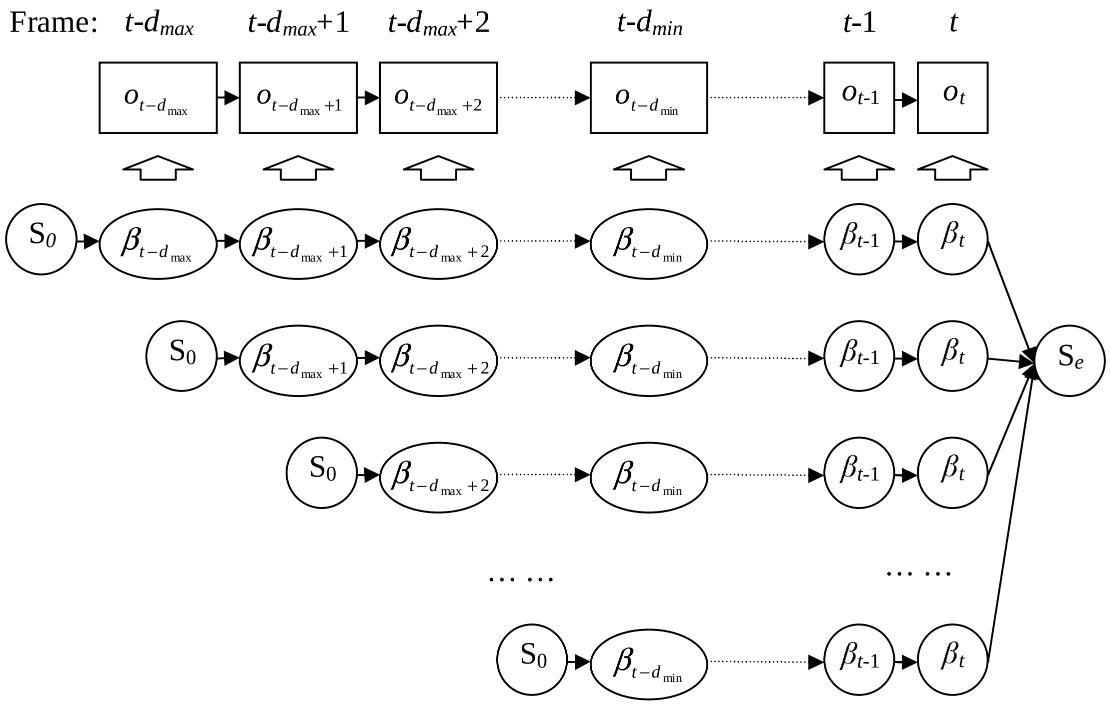
Frame: t-d_max t-d_max+1 t-d_max+2 t-d_min t-1 t
$o_{t-d_{max}}$ $o_{t-d_{max}+1}$ $o_{t-d_{max}+2}$ $o_{t-d_{min}}$ o_{t-1} o_t
S_0 $\beta_{t-d_{max}}$ $\beta_{t-d_{max}+1}$ $\beta_{t-d_{max}+2}$ $\beta_{t-d_{min}}$ β_{t-1} β_t
S_e
…… ……

Figure 5.4: Computation of accumulated probabilities using the backward variables

where S_0 is a null state that indicates the start of the sub-sequence. The starting frame t_η of Level η ranges within $[t - d^k_{max}, t - d^k_{min}]$. The probabilities of the sub-sequences terminating at Frame t, starting at $t - d^k_{max}$,$t - d^k_{max} + 1, \cdots, t - d^k_{min}$ given θ^k_l are computed as depicted in Fig.5.4. In the figure, S_e is another null state indicating the end of the sub-sequence and $\beta_\tau(i)$ is abbreviated as $\beta_\tau, (\tau = t - d_{max}, t - d_{max} + 1, \cdots, t - d_{min})$.

This approach facilitates the computation of the probability trellis in two aspects:

1.) Instead of computing $(d^k_{max} - d^k_{min} + 1)$ probabilities for each node, it is only necessary to compute the same number of accumulated probabilities at the end nodes.

2.) The computational load of getting $P(t_\eta, t, \eta, \theta^k_l)$ in Step 1 is greatly lessened as the backward variable $\beta_{t_\eta}(i)$ is computed in a recursive manner [99]. As manifested in Eq.(5.22),

$$\beta_{t_\eta-1}(i) = \sum_{j=1}^{N} P(s_{t_\eta} = S_j | s_{t_\eta-1} = S_i) b_j(o_{t_\eta}) \beta_{t_\eta}(j) \tag{5.22}$$

$\beta_{t_\eta}(j)$ is used in the computation of $\beta_{t_\eta-1}(i)$. Only a small number of additional computations are necessary to obtain $\beta_{t_\eta-1}(i)$.

The application of the backward variables facilitates the computation of $P(t_\eta, t, \eta, \theta_l^k)$ at the end nodes. Other probabilities, such as $\bar{P}_a(t, \eta, \Theta_l^k)$ and $\bar{P}_{best}(t, \eta)$, are computed in a similar manner as in Step 2, and the optimal sequence of AdaBoost classifiers is decoded using the strategy given in Step 3.

The probability synthesizing rule applied to the proposed method is Decision Strategy 2 presented in Section 4.3.4. Decision Strategy 3 can also be adopted to build the probability trellis. The procedures of level building using Strategy 3 are similar to those using Strategy 2, and are thus not reiterated in the thesis.

5.4 Word/phrase modeling using connected viseme models

5.4.1 Connected viseme models

Theoretically speaking, a word, phrase and sentence can be partitioned into a sequence of visemes. However, it is not easy to automate the process because the transitions between the visemes are always ambiguous in natural speech and the durations of the composite visemes of a word are difficult to determine. For the proposed level building methods, some probabilistic measures are applied to deal with the uncertainties such as the durations of the constituent elements and the transitions between the constituent elements. It thus provides a fuzzy approach of decomposing the words in visual speech.

The viseme models applied to analyze the target words/phrases are the ML viseme models (θ^2 of Speaker 1) mentioned in Section 2.4.3 and the AdaBoost viseme models trained in Section 4.4.2. For both kinds of viseme models, context-dependent samples are adopted as the training samples. Note that the state transition matrix of a composite HMM if of the form that given in Eq.(2.15) and State S_4 indicates the end of viseme production. While level building the HMMs, S_4 is replaced with the initial state (S_1) of another HMM. The underlying assumption is that every HMM has equal probability, $a_{34} = P(S_4|S_3)$, to transit to any other HMMs. For the limited number of words/phrases to be identified in our experiments, this assumption does not add much error to the recognition results. In natural visual speech, however, this assumption does not hold because some viseme transitions, e.g. /b/ to /A:/ and /k/ to /e/, are more likely to occur than other viseme transitions such as /b/ to /b/ and /s/ to /tS/. The transitions between the viseme models in large-vocabulary cases should thus be adjusted according to the phonetic, lexical and semantic rules.

For the experiment, ten words and phrases listed in Table 5.1 are selected for recognition/decomposition. These words/phrases consist of different visemes, which are also given in the table. The same speaker (Speaker 1) as mentioned in Section 2.4 is asked to produce each word/phrase 100 times. RGB to HSV conversion, template matching, feature extraction and vector quantization as mentioned in Section 2.3 are carried out for the videos to obtain the corresponding sequence of codes.

The number of the composite visemes, or equivalently the level number, of a word or phrase can be estimated based on the duration of the word/phrase and the probable duration of visemes shown in Table 5.1. For example, if a word production lasts for 1.2 second, empirically the number of the composite visemes, η, is assumed to be from 2 to 5. Level building on the AdaBoost viseme models is then carried

Table 5.1: Words and phrases selected for recognition

Words/ Phrases	Viseme Spelling	Level No. $[\eta_{min}, \eta_{max}]$
zoo	/z/+/U/	[1, 5]
right	/r/+/A:/+/t/	[2, 5]
deck	/d/+/e/+/k/	[1, 5]
smith	/s/+/m/+/I/+/T/	[2, 7]
banana	/b/+/e/+/n/+/A:/+/n/+/A:/	[3, 9]
we are	/w/+/I/+/A:/	[1, 6]
use up	/U/+/z/+/A:/+/p/	[2, 7]
on my way	/Q/+/m/+/A:/+/w/+/e/	[3, 9]
around the world	/A/+/r/+/Q/+/z/+/U/+/e/+/d/	[4, 10]

Table 5.2: The estimated durations of the visemes

Viseme No.	Duration(sec.) $[d_{min}, d_{max}]$	Viseme No.	Duration(sec.) $[d_{min}, d_{max}]$
1 p, b, m	[0.1, 0.6]	2 f, v	[0.1, 0.7]
3 T, D	[0.2, 0.8]	4 t, d	[0.1, 0.7]
5 k, g	[0.1, 0.7]	6 tS, dZ, S	[0.3, 0.8]
7 s, z	[0.2, 0.8]	8 n, l	[0.1, 0.6]
9 r	[0.1, 0.6]	10 A:	[0.2, 0.8]
11 e	[0.1, 0.6]	12 I	[0.1, 0.8]
13 Q	[0.2, 0.8]	14 U	[0.2, 0.8]

out within the range $\eta_{min} \leq \eta \leq \eta_{max}$. In addition to the level number η, the durations of the constituent elements (visemes) are also used in the computation of the accumulated probabilities. They are estimated and presented in Table 5.2.

Using the above information, a sequence of vectors indicating word/phrase production is decomposed into several sequences of visemes .

5.4.2 Performance measures

The accuracy of recognition is evaluated using the following two performance measures:

Measure 1 (μ_1): It is assumed that the number of visemes appearing in the target word/phrase is known, say η. A correct decomposition is made if the decoded viseme sequence Θ_{best}^{η} is identical to the actual viseme spelling of the target word/phrase. For the 100 samples of a given word/phrase, the number of correct decomposition is counted, say U_1. The accuracy rate is computed as in Eq.(5.23).

$$\mu_1 = \frac{U_1}{100} \tag{5.23}$$

Measure 2 (μ_2): This measure is used when the number of visemes appearing in the target word/phrase is not known. As the level number η ranges from η_{min} to η_{max}, (η_{max}-η_{min}+1) sequences of visemes are decoded by selecting different values for η. The best Θ_{best}^{η} is then identified using Eq.(5.18). Θ_{best}^{η} is regarded as a correct decomposition if it is identical to the true viseme spelling of the target word/phrase. The number of correct decomposition is counted for the 100 samples, say U_2, and the accuracy rate μ_2 is computed using Eq.(5.24).

$$\mu_2 = \frac{U_2}{100} \tag{5.24}$$

Clearly, the information available for the second performance measure is less than that for the first measure. In practice, both measures are useful. Some systems require only one definite decomposition result while others may require several decoded sequences, i.e. the sequences of visemes that are decoded with different level numbers.

Table 5.3: Decomposition accuracy of the two viseme models

Word/Phrases	ML viseme model		AdaBoost viseme model	
	μ_1	μ_2	μ_1	μ_2
zoo	63%	35%	67%	57%
right	39%	28%	43%	40%
deck	27%	17%	31%	17%
smith	32%	20%	30%	25%
transit	25%	15%	33%	26%
banana	34%	24%	42%	25%
we are	59%	40%	58%	51%
use up	32%	26%	41%	30%
on my way	18%	15%	24%	18%
around the world	10%	4%	16%	8%

5.4.3 Experimental results

Comparison of the performance of level building on AdaBoost viseme models and lon ML viseme models (traditional approach) is made. The accuracies of the two approaches using Measure 1 and Measure 2 are summarized in Table 5.3.

The results indicate that words/phrases decomposition using connected AdaBoost HMMs gives better accuracy than that using connected single HMMs for both measures. An AdaBoost viseme model is capable of better covering the distribution of a viseme than an ML viseme model. When applying level building algorithm to model words/phrases, the connected AdaBoost viseme models may also better cover the samples of a given word/phrase than the connected ML viseme models.

It is shown in Table 5.3, even the correct level number is selected, the accuracy of recognition is still low, the highest accuracy attained is 67%. The accuracy of recognition also decreases with increasing duration of the word/phrase to be recognized. This is due to the accumulation of error of viseme decoding. The accuracy of recognition may be increased by using a combination of methods including natural language processing.

5.4.4 Computational load

The computational load involved is estimated as a measure of complexity of the proposed method. An estimate is made based on the following set of parameter values. The composite viseme models are assumed to be 3-state 128-symbol left-right HMMs. Typical length of target words are assumed to be 70 frames (or 1.4s of duration), level number of words are within 2 to 7, and length of visemes within 5 to 35 frames (or equivalently 0.1-0.7s of duration).

The 14 visemes listed in Table 5.2 are modeled by AdaBoost HMM classifiers. If each AdaBoost viseme model comprises 20 HMMs, level building is carried out with a total of $20 \times 14 = 280$ HMMs. At a possible end node, say Node(t, η), $2 \times 4 = 8$, (2×state number) multiplications are carried out to compute the backward variable $\beta_{t_\eta}(i)$ using Eq.(5.22). Note that the state number is 4 because a null state S_0 indicating the beginning of the sub-sequence is also involved. The starting frame of the sub-sequence may range from $t - d_{max}$ to $t - d_{min}$. Therefore, about $8 \times (35 - 5 + 1) = 248$, $(8 \times (d_{max} - d_{min} + 1))$ multiplications are carried out to compute the probabilities of the sub-sequences given a composite HMM using the simplified approach as mentioned in Section 5.3.4. Thus, about 248×280(total number of HMMs) $\approx 7.0 \times 10^4$ multiplications are needed for all the composite HMMs. Computational loads required for other processes such as tracking the best path, matching the reference models and finding the starting frames are light compared to this number and they are neglected. The computational load of building the probability trellis is thus about

70(frame number) $\times$ $7.0 \times 10^4 \approx 4.9 \times 10^6$ multiplications. Such computational load can be considered as modest for modern processors.

It should be noted that the computational load is very much dependent on the duration of the target word/phrase. A target sequence of longer duration requires more computation as a larger probability trellis has to be built. For a long sequence, such as the production of a sentence, it is a good practice to partition the long sequence into several short sequences with each of them comprising fewer visemes. Level building is then performed on the short sequences. This proves to be faster and more accurate than level building on the long sequence. Another factor that influences the computational load is the durations of the constituent elements. In the example mentioned above, if the lengths of the visemes are assumed to have a range of 1-70 frames (or equivalently 0.02-1.4s of duration) instead of 5-35 frames, approximately $8 \times (70 - 1 + 1) = 560$ multiplications would be required to compute the probabilities of the sub-sequences given a composite HMM. The computational load of building the probability trellis is then about $560 \times 280 \times 70 \approx 1.1 \times 10^7$ multiplications, which is twice the number of computations for the example given above. As such, a good estimation of the durations of the constituent elements will greatly lessen the computational load.

5.5 The Viterbi Matching Algorithm for Sequence Partition

For the level building algorithms presented in Section 5.2 and 5.3, a longer sequence is decompose into a chain of constituent elements and each constituent element is modeled by an ML HMM classifier or an AdaBoost-HMM classifier. Such processing may be computationally expensive if an HMM consists of several states. In this section, a simplified approach for sequence partition is proposed with exploration of the Viterbi method.

5.5.1 Recognition units and transition units

Recall the training of the viseme models presented in Section 2.4.2 and 2.4.3, the states of an HMM are configured to align with the initial phase, articulation phase and end phase of viseme production. After implementing the Baum-Welch estimation, the states of the trained ML viseme model also demonstrate good consistency with the three phases. As mentioned in Section 2.4.2, the articulation state of an HMM, which corresponds to the articulation phase of the viseme it models, is the key factor for recognition because the discriminative features of the viseme chiefly lie here. On the other hand, the initial state and the end state of an HMM, which correspond to the initial phase and end phase of the target viseme, are not so prominent for discrimination because they are transitional states and may be distorted by adjoining visemes. For the strategy presented in this section, the articulation states of the HMMs of fourteen visemes are referred to as recognition units and are segmented out of the ML viseme models.

The ML viseme models mentioned in Section 2.4.3 are 3-state 128-symbol discrete HMMs and the second state (S_2) is the articulation state. A recognition unit is thus built up with the following components:

1.) The symbol output array:

$$B = [b_1 \quad b_2 \quad \cdots \quad b_M]_{1\times M} \tag{5.25}$$

where $b_i, (i = 1, 2, \cdots, M)$ is the probability of outputting symbol O_i in the articulation state and M is the number of symbols. In this instance, $b_i = P(O_i|S_2)$ and $M = 128$.

Table 5.4: Transitions between the visemes

Mouth shapes (*starting with*)	Mouth shapes (*end with*)			
	mouth opened	mouth closed	lips stretched	mouth stretched forward
mouth opened	/h/+/ai/	/ai/+/k/	/h/+/I/	/A:/+/u/
mouth closed	/t/+/A:/	/d/+/e/	/t/+/ei/	/d/+/au/
lips stretched	/l/+/ai/	/ei/+/t/	/l/+/I/	/th/+/o/
mouth stretched forward	/p/+/A:/	/u/+/k/	/o/+/I/	/b/+/u/

2.) The state transition matrix:

$$A = \begin{bmatrix} a & 0 \\ 0 & 1-a \end{bmatrix}_{2\times 2}, \qquad 0 \le a \le 1 \tag{5.26}$$

A recognition unit has the probability a to repeat itself and the probability $1-a$ to transit to another unit. Because the state transition matrix of an ML viseme model has the form $A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$, thus $a = a_{22}$.

The transitions between visemes are also studied for the proposed strategy. A number of adjoining visemes are segmented from the word productions. For example, /l/+/I/ is extracted from the production of *lip*. The approximate transition phase is manually extracted from the image sequence in the same manner as viseme segmentation mentioned in Section 2.4.2. The transitions between visemes are roughly clustered into 16 categories as illustrated in Table 5.4.

Four basic mouth shapes are used to describe the viseme productions. As a result, there are totally 16 transitions between them. The examples given in Table 5.4

indicate the transitions between two visemes with designated lip features. For example, /l/+/ai/ in column 2, row 4, is the transition from /l/ (with lips stretched) to /ai/ (with mouth opened). For each transition state, a number of samples are collected to learn its statistical features. For example, the training samples of the transition from state *mouth-opened* to state *lips-stretched* include the transition phases between /l/ and /ai/, /n/ and /au/, /l/ and /A:/. The statistical features of the transitions are obtained by counting and averaging the symbols appeared in the training samples. If there are a total number of M' symbols appeared in the training samples and symbol O_j appears $T(O_j)$ times, the probability distribution coefficient of O_j is computed via Eq.(5.27).

$$b'_j = \frac{T(O_j)}{\sum_{j=1}^{M'} T(O_j)} \tag{5.27}$$

The probability function obtained in this way is applied to model the transition between visemes and are referred to as transition units. In addition to b'_j, a parameter a' is also employed to describe the probability of the transition unit repeats itself and $1 - a'$ indicates the probability of the transition unit transits to another unit. Like the recognition unit mentioned above, a transition unit is also comprised of two components.

1.) The symbol output matrix:

$$B' = [b'_1 \quad b'_2 \quad \cdots \quad b'_M]_{1\times M} \tag{5.28}$$

where $b'_i, (i = 1, 2, \cdots, M)$ is the probability of outputting symbol O_i in the transition unit and $M = 128$ in this instance.

2.) The state transition matrix:

$$A = \begin{bmatrix} a' & 0 \\ 0 & 1 - a' \end{bmatrix}_{2\times 2}, \qquad 0 \le a' \le 1 \tag{5.29}$$

The entries a' and $1 - a'$ are introduced to build state chains to model continuous processes. Setting of the value of a' is problem-dependent. If the duration of the

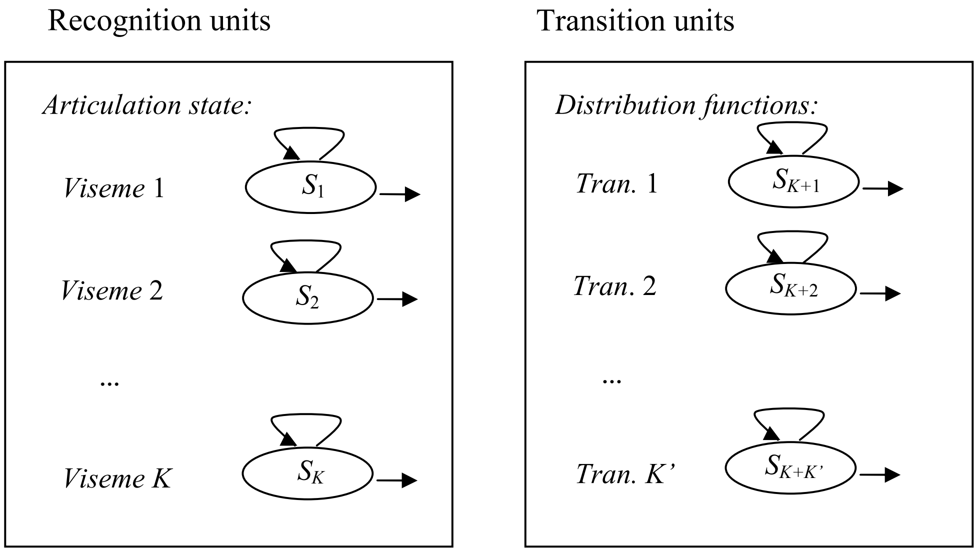

Figure 5.5: The recognition units set and transition units set of the database Θ

transition state of the target process is long, greater value of a' should be used, otherwise, smaller value is chosen. For the experiments conducted in this thesis, we set $a' = 0.5 \sim 0.8$.

The K recognition units are collected to build the recognition unit set and the K' transition units are collected to build the transition unit set as depicted in Fig.5.5. The two sets build the database Θ. The recognition units in Θ are numbered as θ_1 to θ_K and the transition units are numbered as θ_{K+1} to $\theta_{K+K'}$. With such a database, recognition of a continuous process is the process of decoding a chain of states for the target process that the recognition units and transition units appear alternately, which is depicted in Fig. 5.6.

Assume that $x^T = (o_1, o_2, \cdots, o_T)$ is a T-length observation sequence indicates the production of an unknown word or phrase in visual speech. A probability trellis as depicted in Fig. 5.7 is built for x^T with the approach described below.

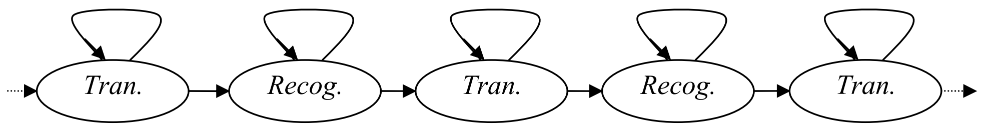

Figure 5.6: The state chain decoded for the target process

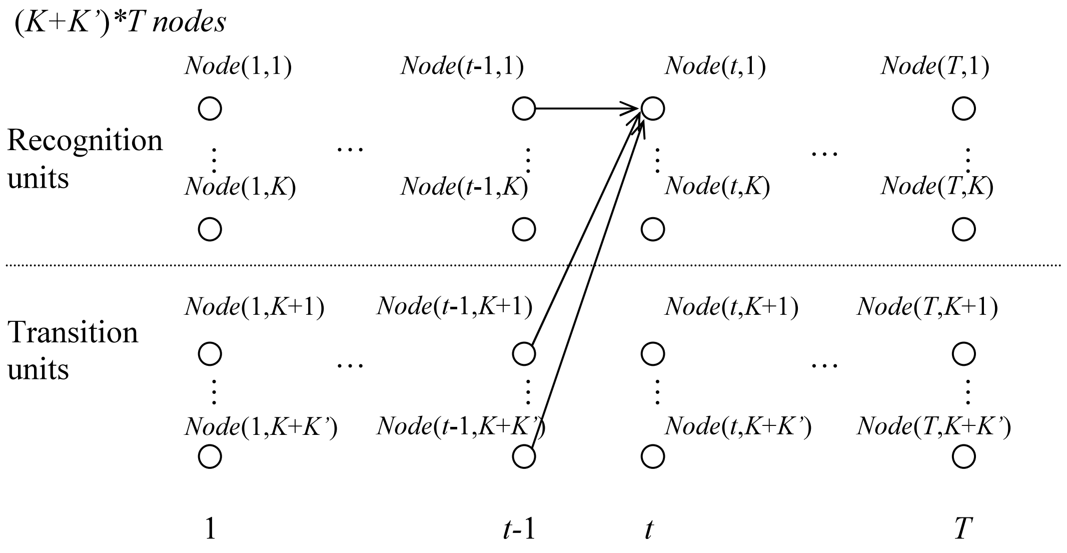

Figure 5.7: The probability trellis for implementing the Viterbi matching algorithm

5.5.2 Initialization

Assume that the sequence starts from a transition state, e.g. from the mouth is closed, the first column of the probability trellis, Node(1,1), Node(1,2), ..., Node$(1, K + K')$, is computed via Eq.(5.30),

$$P_a(1,i) = \begin{cases} 0, & 1 \le i \le K \\ \pi_i b'_i(o_1), & K < i \le K + K' \end{cases} \tag{5.30}$$

where $b'_i(o_1), (1 < i \le K + K')$ is the probability of outputting o_1 given the i-th transition unit and π_i is the probability that the i-th unit (recognition unit or transition unit) is the first state in the state chain. Because x^T is assumed to start from a transition state, values of uniform distribution are applied to configure π_i for the transition units. Thus,

$$\pi_i = \begin{cases} 0, & 1 \leq i \leq K \\ 1/K', & K < i \leq K + K' \end{cases} \tag{5.31}$$

5.5.3 Forward process

In such a probability trellis, a unit will either repeats itself or transits to a unit of another set. If the unit at Node(t, i) is a recognition unit $\theta_i, (1 \leq i \leq K)$, the $(t-1)$-th state is either the same recognition unit or a transition unit. In the case of recognition unit, the accumulated probability to Node(t, i), $P_a^1(t, i)$, is computed via Eq.(5.32).

$$P_a^1(t, i) = P_{best}(t-1, i) a_i b_i(o_t), \qquad 1 \leq i \leq K \tag{5.32}$$

where $P_{best}(t-1, i)$ is the largest accumulated probability to Node$(t-1, i)$ and $b_i(o_t)$ is the probability of outputting o_t given $\theta_i, (1 \leq i \leq K)$. The $(t-1)$-th state can also be a transition unit $\theta_j, (K < j \leq K + K')$. In this case, the accumulated probability $P_a^2(t, j)$ is computed via Eq.(5.33).

$$P_a^2(t, i) = P_{best}(t-1, j)(1 - a'_j) b_i(o_t), \qquad K < j \leq K + K' \tag{5.33}$$

where $P_{best}(t-1, j)$ has the same meaning as in Eq.(5.32) and a'_j is the state transition coefficient of $\theta_j, (K < j \leq K + K')$. Maximization over θ_i and θ_j is performed to get the largest accumulated probability at Node(t, i).

$$P_{best}(t, i) = \max[P_a^1(t, i), P_a^2(t, j)], \qquad 1 \leq i \leq K, K < j \leq K + K' \tag{5.34}$$

Similarly, if the unit at Node(t, i) is a transition unit $\theta_i, (K < i \leq K + K')$, the $(t-1)$-th state may be a recognition unit θ_j and the accumulated probability is computed via Eq.(5.35).

$$P_a^1(t, j) = P_{best}(t-1, j)(1 - a_j) b'_i(o_t), \qquad 1 \leq j \leq K \tag{5.35}$$

where $b_i'(o_t)$ is the probability of outputting o_t given $\theta_i, (K < i \leq K + K')$. The $(t-1)$-th state may also be the same transition unit. In this case, the accumulated probability is computed via Eq.(5.36).

$$P_a^2(t,i) = P_{best}(t-1,i)a_i'b_i'(o_t), \qquad K < i \leq K + K' \tag{5.36}$$

The largest accumulated probability at Node(t,i) is obtained via Eq.(5.37).

$$P_{best}(t,i) = \max[P_a^1(t,j), P_a^2(t,i)], \qquad 1 \leq j \leq K, K < i \leq K + K' \tag{5.37}$$

At each node of the probability trellis, the accumulated probabilities are computed by repeating Eq.(5.32)-(5.37). This process is referred to as the forward process.

5.5.4 Unit backtracking

The probability trellis is finalized at Node$(T,1)$, Node$(T,2)$, $\ldots$, Node$(T,K+K')$. The best-matched state sequence is obtained by backtracking the probability trellis. At time T, the optimal unit is decoded using Eq.(5.38).

$$\theta_{best}(T) = \arg\max_{\theta_i}[P_{best}(T,i)], \qquad 1 \leq i \leq K + K' \tag{5.38}$$

where $\theta_{best}(T) \in \{\theta_1, \theta_2, \cdots, \theta_{K+K'}\}$. If the target sequence is assumed to end with a transition unit, the reference unit is decoded within the range of $K < i \leq K+K'$, i.e.

$$\theta_{best}(T) = \arg\max_{\theta_i}[P_{best}(T,i)], \qquad K \leq i \leq K + K' \tag{5.39}$$

Similarly, if the target sequence is assumed to end with a recognition unit, the reference unit $\theta_{best}(T)$ is searched within the set of recognition units $\{\theta_1, \theta_2, \cdots, \theta_K\}$. If the decoded unit at time T is a recognition unit, i.e. $\theta_{best}(T) = \theta_i (1 \leq i \leq K)$, at time $T-1$, the reference unit $\theta_{best}(T-1)$ is searched within $\{\theta_i, \theta_{K+1}, \theta_{K+2}, \cdots, \theta_{K+K'}\}$. Let

$$P_{max}(T-1,j) = \max_j[P_{best}(T-1,j)], \qquad K < j \leq K + K' \tag{5.40}$$

we have,

$$\theta_{best}(T-1) = \begin{cases} \theta_i, & \text{if } P_{best}(T-1,i) > P_{max}(T-1,j) \\ \arg_{\theta_j}[P_{max}(T-1,j)], & \text{otherwise} \end{cases} \tag{5.41}$$

If the decoded unit at time T is a transition unit, i.e. $\theta_{best}(T) = \theta_i, (K < i \leq K+K')$,the reference unit at time $T-1$, $\theta_{best}(T-1)$, is searched within the set $\{\theta_1, \theta_2, \cdots, \theta_K, \theta_i\}$. Similarly, we define

$$P_{max}(T-1,j) = \max_j [P_{best}(T-1,j)], \qquad 1 \leq j \leq K \tag{5.42}$$

The optimal unit is also decoded using Eq.(5.42). Note that $K < i \leq K+K'$ and $1 \leq j \leq K$ in this case.

The above backtracking process iterates until the first column of the probability trellis is reached and a sequence of transition units and recognition units $\theta_{best}(T), \theta_{best}(T-1), \cdots, \theta_{best}(1)$ are decoded. If two adjoining units are the same, they are combined into one unit as they model the same viseme or transition state. In this way, η units $(\theta_{1,t_1}, \theta_{2,t_2}, \cdots, \theta_{\eta,t_\eta})$, $(\theta_{i,t_i} \in \{\theta_1, \theta_2, \cdots, \theta_{K+K'}\})$, as depicted in Fig.5.6 are decoded for x^T.

For the proposed method, the recognition units and transition units are accumulated to build the probability trellis. The process is similar to the Viterbi algorithm for decoding hidden states of HMM [99]. As a result, this method is referred to as Viterbi matching algorithm for sequence partition. The approach differs from the level building algorithm in two aspects. First, in level building algorithm, "level" is employed to delimit a viseme, i.e. to determine where a viseme production starts and where it stops, while in Viterbi algorithm, there is no "level" defined for the probability trellis. The start and end of viseme production are identified by the transitions between recognition units and transition units. Second, in the level building algorithm, several HMM chains can be decoded by selecting different level numbers, while in the proposed algorithm, only one sequence of speech units are decoded. The number of visemes of the target sequence is determined by the number of recognition units that appear in the decoded sequence. As a result, the proposed Viterbi algorithm should not be applied to the situation when the number of composite visemes of the target sequence is known.

The Viterbi algorithm has the advantages of simplification of computation and economy of data storage. At Node(t, i), for example, only $1 + K'$ (if θ_i is a recognition unit) or $1 + K$ (if θ_i is a transition unit) probabilities are computed with the proposed Viterbi approach while $N \times K$ probabilities, where N is the state number of the ML HMM classifiers, have to be computed with the level building method. In addition, the starting frames and model indices of the best-matched model are not necessary to be stored at each node for the proposed Viterbi method, while these values have to be saved for the level building method as they are used for the computation of the probabilities of subsequent nodes.

5.6 Application of the Viterbi approach to visual speech processing

The performance of the proposed Viterbi method is first assessed with experiments on word/phrase partition. A number of words/phrases as illustrated in Table 5.5 are selected for recognition/decomposition. While decoding the speech units with the Viterbi approach, the target word/phrase is assumed to start and end with transition units. Thus the recognition units are $(\theta_{2,t_2}, \theta_{4,t_4}, \cdots, \theta_{\eta-1,t_{\eta-1}})$ and η is an odd number. If the target word consists of k visemes, a correct decomposition is made if it satisfies: 1.) the number of the recognition units $\frac{\eta-1}{2} = k$ and 2.) the recognition units $(\theta_{2,t_2}, \theta_{4,t_4}, \cdots, \theta_{\eta-1,t_{\eta-1}})$ are identical to the k composite visemes of the target word. For the samples of the selected words and phrases, the number of correct recognition/decomposition is counted and the accuracy rate μ_2

Table 5.5: Recognition/decomposition accuracy of the words and phrases

Words/Phrases	Accuracy rate	
	Level building method: μ_2	Viterbi method: μ_2
zoo	35%	17%
lip	31%	32%
with	41%	35%
black	22%	27%
transit	15%	6%
use up	26%	33%
on my way	15%	3%
around the world	4%	0%

is computed via Eq.(5.24).

5.6.1 Experiment 1

In the first experiment, 100 samples are drawn for each testing word/phrase in Table 5.5. The accuracy rate (μ_2) is computed via Eq.(5.24). The accuracy rates of the ML viseme classifiers (μ_2) are also listed in Table 5.5 for comparison.

The experimental results indicate that the accuracy of the proposed Viterbi algorithm is lower than that of the level building on ML HMMs. The Viterbi approach focuses on the transitions between visemes, while for the word or phrase productions given in Table 5.5, the transition phases are usually ambiguous. Therefore, the recognition units are difficult to detect from the target words using the Viterbi method. The level building algorithm, on the other hand, is not so much dependent on the transitions for recognition/decomposition, and thus performs better than the proposed Viterbi approach.

Table 5.6: Recognition/decomposition accuracy of connected digits

Connected digits	Accuracy rate	
	Level building method: μ_2	Viterbi method: μ_2
1-3	88%	80%
2-4-6	74%	74%
5-9-7-9	62%	66%
0-0-8-8	70%	74%
1-2-3-4-5	42%	56%
9-8-7-6-5-4	28%	22%

5.6.2 Experiment 2

In the second experiment, the visual speech units to be identified are connected digits as illustrated in Table 5.6. The speaker is asked to produce each digit-sequence with clear intervals between adjoining digits. For example, 0-3-5 is articulated as "zero, three, five". 50 samples of each digit-sequence are drawn to be partitioned into a sequence of digit models using the level building algorithm and the proposed Viterbi algorithm.

Like the viseme models, the digit models used in the experiment are 3-state 128-symbol left-right discrete HMMs and are configured according the initial phase, articulation phase and end phase of digit productions. The training data are a number of context-dependent digit samples. After training the HMMs with the Baum-Welch estimation, the articulation phases are segmented to build the set of recognition units. For the closed-set of ten digits, it consists of ten recognition units. The set of the transition units comprises 16 units as given in Table 5.4. The accuracy rates of the Viterbi algorithm and level building method, which are computed via Eq.(5.24), are given in Table 5.6.

It is observed that the Viterbi approach performs almost equally well for digit partition as the level building approach. The sequences of digits, unlike the words/phrases in Experiment 1, have clear intervals between component digits. And the intervals may be well modeled by the transition units. As a result, the recognition accuracy in Experiment 2 is higher than that in Experiment 1 when the number of composite recognition units is the same for the target sequences. The accuracy of recognition/decomposition decreases as the number of digits in the target sequence increases. This situation applies to both the level building approach and the Viterbi approach because the error accumulates with the increment of the length of the target sequence.

5.6.3 Computational load

The computational load of the Viterbi approach is also compared with that of the level building approach. If a sample of the production of “2-4-6” lasts for 2.5sec and the video clip is sampled at 50 frames per second, the probability trellis has 125(frames) $\times$ 26(number of units) $=$ 3250 nodes. At each node, 11 (for a recognition unit) or 17 (for a transition unit) probabilities are computed. About $(11 + 17)/2 \times 3250 = 4.55 \times 10^4$ probabilities have to be computed to build such a probability trellis. For the level building on HMMs approach, the probability trellis is comprised of 125(frames) $\times$ 4(state number) $=$ 500 nodes. At each node, 4(state number) $\times$ 10(number of HMMs) $\times$ 10(state duration) $=$ 400 probabilities have to be computed. The total number of probabilities is $400 \times 500 = 2 \times 10^5$. The computational load of the Viterbi approach is less than one-fourth of the level building approach. As a result, the implementation of Viterbi matching is much faster than that of the level building method.

5.7 Summary

The approaches discussed in this chapter serve as a link between the recognition of constituent elements and the recognition of continuous processes. The level building strategy is studied for this purpose. Level building on HMM is an exhaustive means of searching a sequence of HMMs to match a target sequence. The key point of this method is the construction of a probability trellis that reveals the development of HMM modeling. In this chapter, level building on ML HMM classifiers is discussed first; following that, the procedures of level building on AdaBoost-HMM classifiers are given in detail.

The level building on ML HMM classifiers and AdaBoost-HMM classifiers have been applied to word/phrase partition in visual speech. For the experiments performed in this chapter, a number of word/phrases are decomposed into sequences of ML viseme models or AdaBoost viseme models with the proposed strategies. The accuracy of the two approaches is compared and the results indicate that the target word/phrases can be more accurately recognized/decomposed with the connected AdaBoost-HMM classifiers than using the connected ML HMM classifiers.

A simplified approach for partitioning continuous process called Viterbi matching algorithm is also presented in this chapter. This method employs some specially configured recognition units and transition units to decode the target sequence. If the transition units and recognition units are looked as HMMs with only one state, the Viterbi approach is essentially level building on these special HMMs. The Viterbi approach is featured with economy of computational load and data storage. The experiments of recognizing/decomposing connected digits indicate that this approach is almost equal effective as level building on ML HMM classifiers.

Chapter 6

Other Aspects of Visual Speech Processing

In previous chapters, the approaches of recognizing visemes and words/phrases in visual speech are presented. These are the main parts of our study. In this chapter, other problems of visual speech processing are discussed, which include tracking of lip features in 3D surface and mapping of viseme productions between different speakers. The solution to these problems may extend the applicability of a visual speech recognition system to unfavorable conditions.

6.1 Capture lip dynamics using 3D deformable template

The performance of a visual speech processing system relies heavily on the extracted features of visual speech. For the viseme classifiers mentioned in Chapter 2, 3 and 4, if the extracted geometric features cannot well reveal the actual lip shape, the performance of the classifiers will degrade greatly. In our experiment,

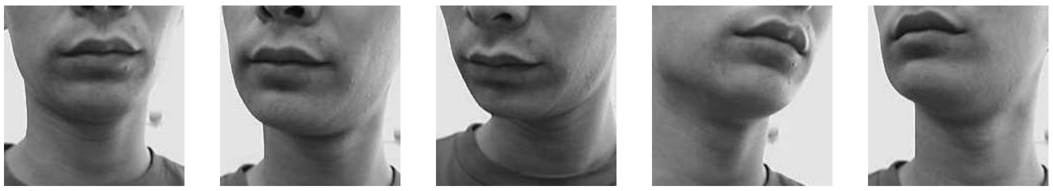

Figure 6.1: The head of the speaker may rotate during speech.

the video clips are processed by means of RGB to HSV conversion, image thresholding and template matching as mentioned in Section 2.3. In most cases, the lip area can be accurately segmented with these strategies and the extracted features can well indicate the lip shapes recorded in the video clips. However, as these approaches only attempt to extract information out of the raw images, they cannot correct the error caused by the raw image, such as the changes of the filming condition or the variations of the filming objects. In real world scenario, the head of the speaker may shift or rotate during speech, which is depicted in Fig.6.1. To guarantee the performance of a visual speech processing system, the influence caused by such variations should be eliminated.

Recall the data acquisition strategy mentioned in Section 2.1, the filming condition is under control such that the positions of the camera and the speaker are kept fixed and the head of the speaker is stable either. The speaker in this situation is a cooperative target/speaker and the data collected in this way are referred to as ideal data. The viseme classifiers and word classifiers presented in Chapter 2, 3, 4 and 5 can only work on such ideal data. The filming restrictions imposed to the speaker can eliminate the error caused by the relative movement between camera and the speaker. However, these measures are not realistic for real-world visual speech processing. The desired approach is to convert video samples that are filmed in any condition into ideal data. In this chapter, the approach for solving this problem is presented.

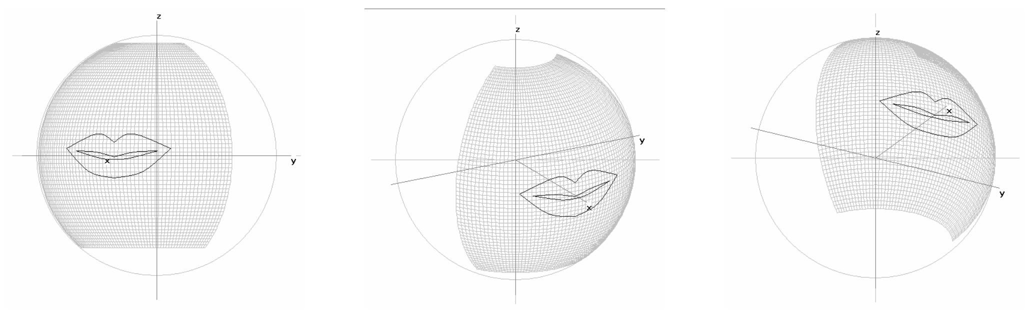

Figure 6.2: The 3D lip templates adopted in the system

For the proposed strategy, the filming condition is not as strict as that in filming a cooperative speaker. Only the distance between the camera and speaker is kept unchangeable while the head of the speaker is free to rotate during speech. The speaker in this situation is referred to as a non-cooperative target/speaker. The viseme productions of the non-cooperative speaker are recorded as video clips. The video clips are partitioned into sequences of images, the approximate lip region is segmented and the RGB factors of the image are converted to HSV factors following the strategies given in Section 2.3.

6.1.1 3D deformable template

In Section 2.3.2, deformable template is adopted to capture lip dynamics from the processed hue images. However, this is a 2D approach such that the images to be processed should be the exact frontal projection of the speaker. If the head positions are not uniform in the images, the 2D template is not effective even if the lip region is properly located and segmented. To solve this problem, 3D lip tracking approaches have been carried out as reported in [35][59][89]. In this thesis, a 3D deformable template and a novel tracking strategy are proposed.

As shown in Fig.6.2, the lip template is affixed to a 3D ball surface. The frontal

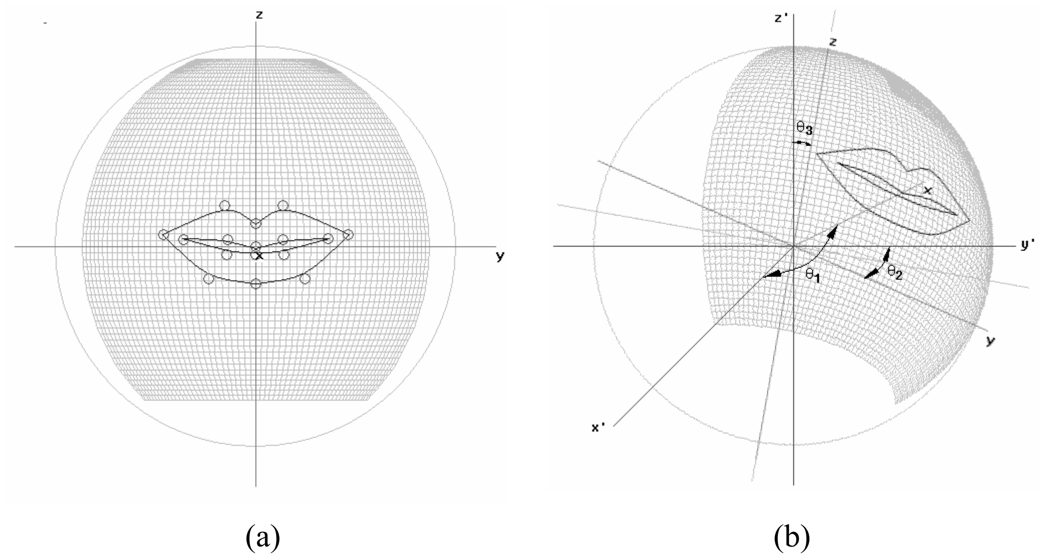

Figure 6.3: (a) Frontal view of the 3D lip template (standard template) (b) The rotation angles of the 3D lip template

projection of the 3D template is exactly the 2D template used in Section 2.3.2, which is also comprised of eight Bezier curves as shown in Fig.6.3(a). Note that the points in the small circles are the control points of the template. The 3D lip template may rotate around three axes. If the frontal view of the lip [Fig.6.3(a)] is defined as the standard template, the rotation angles of a template are defined as the angles between the axes of the target template with respect to the planes of the standard template. For example, θ_1 in Fig.6.3(b) is the angle between x-axis (target template) and $x'-o-z'$ plane (standard template), θ_2 is the angle between y-axis and $x'-o-y'$ plane and θ_3 is the angle between z-axis and $y'-o-z'$ plane. For the 3D lip template discussed in this section, θ_1, θ_2 and θ_3 are used as the position parameters of the template.

A 3D lip template thus consists of a 2D template, which is determined by 16 control points, and three rotation angles θ_1, θ_2 and θ_3. For simplification, the tongue template mentioned in Section 2.3.2 is not used in the 3D templates. The processed hue images indicating certain viseme production are put through the following lip tracking strategy to extract geometric features.

6.1.2 Lip tracking strategy

The energy function of the 3D template is defined the same as the energy function of the component 2D template. Mathematically,

$$E_{3D} = E_{2D} = c_1 E_{lip} + c_2 E_{edge} + c_3 E_{hole} + c_4 E_{inertia} \tag{6.1}$$

where E_{lip}, E_{edge}, E_{hole} and $E_{inertia}$ are component energy functions defined in Eq.(2.1)$\sim$(2.4). By selecting different values for c_1, c_2, c_3 and c_4, the control points of the template are adjusted to minimize the energy function E_{3D}. This template matching strategy is discussed in detail in [93] and is applied to 2D lip tracking in Section 2.3.2. To apply this algorithm to 3D lip tracking, the coordinates of the control points of 3D template are mapped to 2D plane using Eq.(6.2).

$$\begin{bmatrix} x_{2D} \\ y_{2D} \end{bmatrix} = \begin{bmatrix} -\cos\theta_1 \cos\theta_3 \sin\theta_2 & -\sin\theta_1 \cos\theta_3 \sin\theta_2 & \sin\theta_3 \sin\theta_2 \\ \cos\theta_1 \sin\theta_3 & \sin\theta_1 \sin\theta_3 & \cos\theta_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{6.2}$$

where (x, y, z) is the coordinates in 3D template and (x_{2D}, y_{2D}) is the corresponding coordinates in the frontal-view plane. Note that the frontal-view is defined as the plane parallel to $y-o-z$ plane of the standard template as depicted in Fig.6.3(a), the y-axis of the 3D template corresponds to the x_{2D}-axis and the z-axis of the 3D template corresponds to the y_{2D}-axis of the 2D template mentioned in Section 2.3.2. After carrying out coordinate transformation, the base template matching algorithm for lip tracking is then performed on the mapped 2D coordinates (x_{2D}, y_{2D}).

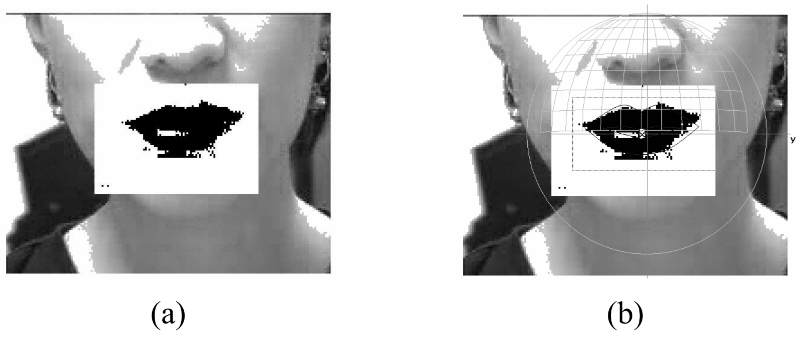

Figure 6.4: (a) The segmented lip region (b) A standard 3D template is configured to match the lip region.

The input hue image sequence is numbered as Frame 1, Frame 2, ..., Frame T. Initially, a 3D template with randomly configured rotation angles θ_1, θ_2 and θ_3 is selected to fit the segmented lip region at Frame 1 [see Fig.6.4(a)]. The control points of the template are adjusted to minimize the energy function E_{3D} using the base template matching algorithm. In Fig.6.4(b), for example, a standard template ($\theta_1 = \theta_2 = \theta_3 = 0$) is matched against the lip region. The initial rotation angle settings may not be correct but will be updated in subsequent tracking procedures.

Step 1: Compute the distance between 3D templates

The rotation angles of the 3D template take discrete values, for example, $\theta_1, \theta_2, \theta_3 = N \times \Delta\theta (N = 0, \pm 1, \pm 2, \cdots)$. At time t, if Γ_t^i is one of the fitted templates with rotation angles $\Theta_t^i = (\theta_1, \theta_2, \theta_3)$. At time $t+1$, K closest angle settings $\Theta_{t+1}^1 = (\theta_1^1, \theta_2^1, \theta_3^1)$, $\Theta_{t+1}^2 = (\theta_1^2, \theta_2^2, \theta_3^2)$,..., $\Theta_{t+1}^K = (\theta_1^K, \theta_2^K, \theta_3^K)$ are obtained from Eq.(6.3).

$$\Theta_{t+1}^i = \arg_{\Theta}[\|\Theta_{t+1}^i - \Theta_t\| < D_0] \tag{6.3}$$

where $\|\Theta_{t+1}^i - \Theta_t\|$ indicates the Euclidean distance between Θ_{t+1}^i and Θ_t, and D_0 is a predefined threshold. For each of the K angle settings at time $t+1$, the corresponding template is matched against the $(t+1)$-th hue image using the base template matching algorithm. At time t, if K matched templates are Γ_t^1, Γ_t^2, $\ldots$, Γ_t^K that have different rotation angles Θ_t^1, Θ_t^2, $\ldots$, Θ_t^K, there will be K^2 templates at time $t+1$ obtained via Eq.(6.3) (while some of the templates may have same rotation angles), denoted as $\bar{\Gamma}_{t+1}^1$, $\bar{\Gamma}_{t+1}^2$, $\ldots$, $\bar{\Gamma}_{t+1}^{K^2}$. The Euclidean distance of rotation angles between one of the K^2 templates at time $t+1$, say $\bar{\Gamma}_{t+1}^j$, and Θ_t^i is given in Eq.(6.4).

$$\|\Theta_{t+1}^j - \Theta_t^i\| = \sqrt{(\theta_1^j - \theta_1)^2 + (\theta_2^j - \theta_2)^2 + (\theta_3^j - \theta_3)^2} \tag{6.4}$$

where $\Theta_{t+1}^j = (\theta_1^j, \theta_2^j, \theta_3^j)$ is the rotation angles of $\bar{\Gamma}_{t+1}^j$ and $\Theta_t^i = (\theta_1, \theta_2, \theta_3)$. The Euclidean distances between Θ_{t+1}^j and $\Theta_t^i, (i = 1, 2, \cdots, K)$ are summed up via Eq.(6.5).

$$D(i, j, t+1, \Theta) = \sum_{i=1}^{K} \|\Theta_{t+1}^j - \Theta_t^i\|, \qquad j = 1, 2, \cdots, K^2 \tag{6.5}$$

The Euclidean distance between the 2D templates of $\bar{\Gamma}_{t+1}^j$ and that of Γ_t^i is also computed and summed up via Eq.(6.6).

$$D(i, j, t+1, \Gamma_{2D}) = \sum_{i=1}^{K} \|\bar{\Gamma}_{t+1}^j - \Gamma_t^i\|, \qquad j = 1, 2, \cdots, K^2 \tag{6.6}$$

The computation of the Euclidean distance $\|\bar{\Gamma}_{t+1}^j - \Gamma_t^i\|$ is given in Eq.(2.5) in Section 2.3.2. The overall distance between 3D templates $\bar{\Gamma}_{t+1}^j$ and $\Gamma_t^i, (i = 1, 2, \cdots, K)$, which is called 3D distance, is given in Eq.(6.7).

$$D(i, j, t+1, \Gamma_{3D}) = D(i, j, t+1, \Gamma_{2D}) + \mu D(i, j, t+1, \Theta) \tag{6.7}$$

where μ is a positive constant that weight the two portion of distances.

Step 2: Build the template trellis

The K^2 3D distances $D(i, j, t+1, \Gamma_{3D}), (j = 1, 2, \cdots, K^2)$ are computed for $\bar{\Gamma}_{t+1}^1$, $\bar{\Gamma}_{t+1}^2$, $\ldots$, $\bar{\Gamma}_{t+1}^{K^2}$ and are compared with one another. The K templates that have the smallest 3D distances and with different rotation angles are selected as the

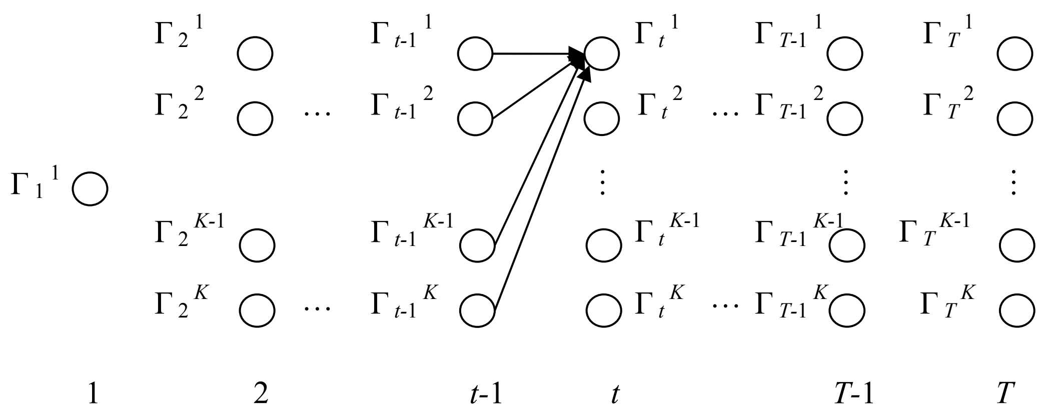

Figure 6.5: The trellis for searching the best-matched 3D templates

candidate templates at $t+1$. If two templates both have the distances within the K smallest 3D distances but their rotation angles are the same, only the template with smaller distance is selected as a candidate template. The K templates selected from $\bar{\Gamma}^1_{t+1}$, $\bar{\Gamma}^2_{t+1}$, $\ldots$, $\bar{\Gamma}^{K^2}_{t+1}$ are denoted as Γ^1_{t+1}, Γ^2_{t+1}, $\ldots$, Γ^K_{t+1}. In this way, a template trellis is built as depicted in Fig.6.5. Note that at $t = 1$, only one template is used in the trellis.

For each template at each time slot, the 3D distance is accumulated. For Γ^j_t, the accumulated 3D distance is computed via Eq.(6.8).

$$D_a(j,t) = \min[D_a(i,t-1) \times D(i,j,t,\Gamma_{3D})], \quad i = 1,2,\cdots,K \tag{6.8}$$

At $t = 2$, we have

$$D_a(j,2) = D(1,j,2,\Gamma_{3D}) \tag{6.9}$$

By repeating Eq.(6.8) from $t = 2$ to $t = T$, the accumulated 3D distance are computed for each matched template of the trellis.

Step 3: Search the best-matched templates

The path that leads to the smallest accumulated 3D distance is searched. For example, at $t = T$, the best-matched template $\Gamma_{best}(T)$ is obtained via Eq.(6.10).

$$\Gamma_{best}(T) = \arg\min_{\Gamma_T^j} D_a(j, T), \qquad j = 1, 2, \cdots, K \tag{6.10}$$

At time t, if $\Gamma_{best}(t) = \Gamma_t^j$, the best matched template at $t-1$ is searched via Eq.(6.11).

$$\Gamma_{best}(t-1) = \arg\min_{\Gamma_{t-1}^i}[D_a(i, t-1) \times D(i, j, t, \Gamma_{3D})], \quad i = 1, 2, \cdots, K \tag{6.11}$$

The procedures mentioned above are repeated until Frame 2 and a sequence of templates are decoded, denoted as $\Gamma_{best}(T)$, $\Gamma_{best}(T-1)$, ..., $\Gamma_{best}(2)$, Γ_1^1. However, the decoded templates may not be a good estimation for the lip movement because the rotation angles of Γ_1^1 are randomly set. To solve this problem, the template trellis is built again but in inverse order, i.e. the trellis starts from Frame T, $\Gamma_{best}(T)$, and ends at Frame 1, Γ_1^1, Γ_1^2,..., Γ_1^K. The best-matched sequence of templates is searched from Frame 1 to Frame T following the procedures given in Step 3. If the initial position of the head does not bias too much from the standard position ($\theta_1 = \theta_2 = \theta_3 = 0$), such management may draw the rotation angles of the templates of the first several frames to the actual head positions.

6.1.3 Properties of the tracking strategy

For the proposed strategy, the distance between consecutive 3D templates is minimized. Because the 3D distance indicates the difference between the lip shapes and head positions of the speaker in consecutive frames, the strategy is in favor of slow change of lip shape and slow movement of head. The video clips used in our experiments are sampled at 50 frames per second, the movement of the lips and the rotation of the head are minor in consecutive frames. As a result, the objective of the tracking strategy conforms to the dynamics of the captured lip motion.

In Eq.(6.7), the distance between 2D templates and rotation angles are balanced by a positive constant μ. Setting of the value of μ is problem-dependent. If greater μ

is chosen, the distance between rotation angles, say $D(i, j, t+1, \Theta)$, takes greater portion in the overall distance $D(i, j, t+1, \Gamma_{3D})$. The decoded sequence of lip templates may have smooth change of rotation angles, i.e. the rotation angles of consecutive frames do not vary too much, but the component 2D templates may change relatively abruptly. If smaller value of μ is chosen, the tracking algorithm allows relatively rapid variation of the rotation angles but smooth variation of the 2D templates (lip shapes). As a result, the value of μ is empirically selected for the speaker. If the head of the speaker rotates slightly during speech, greater value of μ should be adopted; otherwise, smaller value of μ is used. In our experiments, the value of the distance between the rotation angles, $D(i, j, t+1, \Theta)$, is of the order of 10^{-3}, while the Euclidean distance between the 2D templates, $D(i, j, t+1, \Gamma_{2D})$, is of the order of 10^3, μ is set to be equal to 1.0×10^5 for the speaker.

For the proposed tracking strategy, even if the rotation angles of template Γ_1^1 are not correctly initialized, the 3D templates of subsequent frames can still be "drawn" to the correct rotation angles with the development of the template trellis. This is validated by our experimental results. However, the rotation angles of Γ_1^1 cannot deviate too much from the true values. Otherwise the lip templates obtained may diverge. In our experiments, standard lip templates ($\theta_1 = \theta_2 = \theta_3 = 0$) are applied to match the lip shape at Frame 1 and the starting frame of the investigated speech is approximately a frontal view of the speaker, i.e. $\theta_1, \theta_2, \theta_3 < 3°$.

6.1.4 Experiments

The lip tracking algorithm based on 3D lip template is applied to capture lip dynamics from video clips of visual speech. The rotation angles take discrete values $\theta_1, \theta_2, \theta_3 = N \times \Delta\theta$,($N = 0, \pm 1, \pm 2, \cdots$), where $\Delta\theta$ is the quantization unit (step). In the experiments conducted in this thesis, we select $\Delta\theta = 3.49 \times 10^{-3} rad = 0.2°$. Our experimental results show that such a step value can describe head rotation

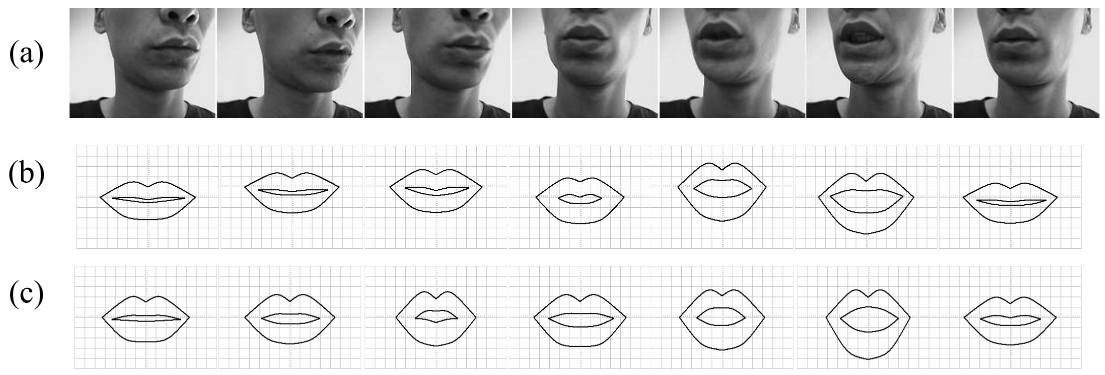

Figure 6.6: (a) Raw images (b) lip shapes decoded using the 3D template (c) lip shapes decoded using the 2D template

with good accuracy. If the rotation angle $\theta_1 = \theta$ at Frame t, the selection of θ_1 at Frame $t+1$ is $\theta - \Delta\theta$, θ or $\theta + \Delta\theta$. The same increments are also applicable to θ_2 and θ_3. As a result, for each frame, $3 \times 3 = 9$ templates with different rotation angles are searched with the approach proposed in Section 6.1.2.

The lip shapes extracted by means of 3D lip templates are depicted in Fig.6.6(b). The lip tracking results using the 2D method mentioned in Section 2.3.2 are also given in Fig.6.6(c) for comparison. Note that the frames depicted in Fig.6.6(a) are not consecutive frames but with intervals of about 5-20 frames. It can be observed that the deformation to the lip shapes caused by the rotation of the speaker's head is well compensated with the proposed 3D method. Take the second frame as an example. The head of the speaker rotates to the left side, which causes the width of 2D lip template to be shortened. However, such deformation is compensated with the application of the proposed 3D tracking algorithm. The width of the mouth is thus more accurately detected than using 2D template.

The lip tracking strategy based on 3D deformable template extends the applicability of the conventional 2D template method. A prominent feature of the strategy is the ease of implementation. The 2D template affiliated to the 3D template is the same as the one discussed in Section 2.3.2. As a result, the base 2D template matching algorithm can be adopted in 3D template matching without modification. Because the proposed tracking algorithm minimizes the accumulated distance between consecutive templates, this approach can be applied to the situation when the movement of the target is smooth over time, for example, gesture recognition, traffic monitoring and so on.

6.2 Cross-speaker viseme mapping using Hidden Markov Models

The viseme classifiers and word/phrase classifiers presented in Chapter 2, 3, 4 and 5 are speaker-dependent recognition systems. That is to say, such classifiers only work well for a specific speaker. If the visual speech of another speaker is presented to the system, the recognition accuracy will drop drastically. The reason underlying the speaker-dependency of the visual speech processing systems is the difference of facial features between different speakers. As mentioned in Section 1.2.4, the facial features demonstrate great variation from person to person. A system trained with the data of a specific speaker cannot be applied to process the visual speech data of another speaker. In visual speech processing domain, elimination of the speaker-dependency is an important aspect of building universal visual speech processing system. However, research on this topic is still fresh and only a very limited number of experiments have been conducted [112]. In this section, some preliminary research towards the goal of eliminating speaker dependency is reported. Our approach is to map the viseme produced by one speaker (referred to as the source speaker) to another speaker (referred to as the destination speaker). HMM is once again adopted as the viseme model. The proposed strategy is not sufficient to eliminate speaker-dependency of a visual speech processing system but may give an indirect approach for solving this problem.

6.2.1 HMM with mapping terms

The viseme models used in this section are HMMs described in Section 2.4.2. The HMMs have discrete symbol set and are trained with the Baum-Welch method. For ease of subsequent explanation, the viseme models (HMMs) of the source speaker are referred to as the source models and the viseme models of the destination speaker are referred to as the destination models.

Assume that $\{O_1^s, O_2^s, \cdots, O_M^s\}$ and $\{S_1^s, S_2^s, \cdots, S_N^s\}$ are the symbol set and state set of the source models, and $\{O_1^d, O_2^d, \cdots, O_{M'}^d\}$ and $\{S_1^d, S_2^d, \cdots, S_{N'}^d\}$ are the symbol set and state set of the destination models, where N is the state number and M is the symbol number of the source models, and N' and M' are those of the destination models. For a viseme in Table 2.1, say viseme $k, (k = 1, 2, \cdots, 14)$, a destination model θ_k^d (θ^2 as mentioned in Section 2.4.3) is trained using the context-independent samples of the destination speaker.

For the source model of viseme k, θ_k^s, some mapping terms are introduced to maintain relationship between the states of the source model and the states of the destination model. Assume that $x_s^T = (o_1^s, o_2^s, \cdots, o_T^s)$ is a context-independent sample of viseme k of the source speaker, where o_i^s denotes the i-th observed symbol in the sequence. The source model θ_k^s is configured according to the three phases of viseme production (see Section 2.4.2). Given θ_k^s and x_s^T, the optimal state chain $s_s^T = (s_1^s, s_2^s, \cdots, s_T^s)$ is decoded using the Viterbi algorithm [99], where s_i^s stands for the i-th state in the decoded state chain. The source model is not trained by x_s^T alone but is also tuned to be related to the destination model. For this purpose, an observation sequence $x_d^T = (o_1^d, o_2^d, \cdots, o_T^d)$ with the same length T is selected from the training samples of viseme k of the destination speaker. The optimal

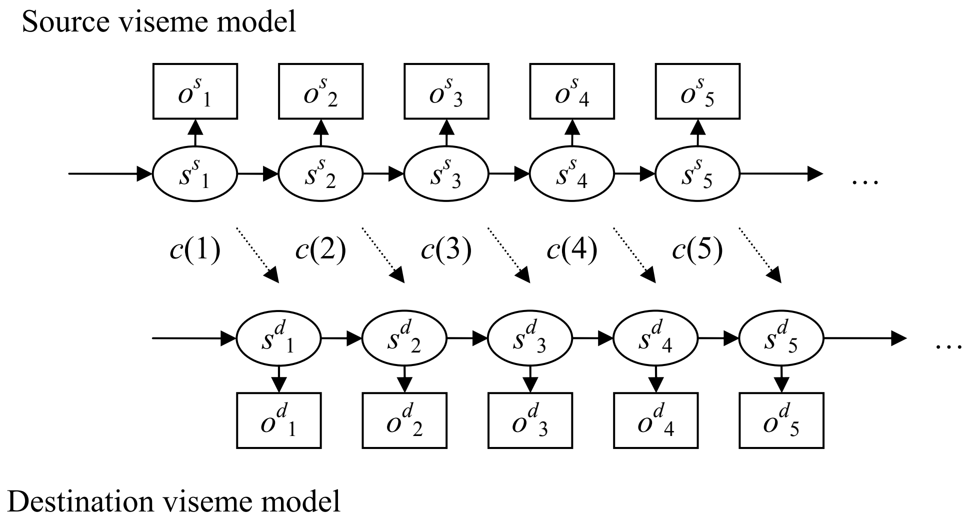

Figure 6.7: Mapping between the source model and the destination model

state chain given θ^d_k, denoted as $s^T_d = (s^d_1, s^d_2, \cdots, s^d_T)$, is decoded for x^T_d using the Viterbi algorithm, where o^d_i and s^d_i have the same meaning as in the source model. Note that θ^d_k is trained using the Baum-Welch method while θ^s_k is not trained. The states of the source model and destination model are associated with each other by the mapping terms $c(1), c(2), \cdots, c(T)$ as depicted in Fig.6.7.

These mapping terms come from the mapping matrix $C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N'} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N'} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N,1} & c_{N,2} & \cdots & c_{N,N'} \end{bmatrix}$, where $c_{i,j} = P(S^d_j | S^s_i)$. The coefficients in Matrix C are initialized with uniform values as given in Eq.(6.12).

$$c_{i,j} = 1/N, \qquad i = 1, 2, \cdots, N, j = 1, 2, \cdots, N' \tag{6.12}$$

The state chain $s^T_d = (s^d_1, s^d_2, \cdots, s^d_T)$ can be looked as the symbols output by $s^T_s = (s^s_1, s^s_2, \cdots, s^s_T)$. By combining o^s_t and s^d_t as the t-th observation symbol of the source sequence, training of the source model thus becomes the process of adjusting the parameters of θ^s_k to maximize the likelihood $P(s^d_1 + o^s_1, s^d_2 + o^s_2, \cdots, s^d_T + o^s_T | \theta^s_k)$.

Matrix B of θ_k^s is expanded from dimension $N \times M$ to dimension $N \times (M + N')$ as illustrated in Eq.(6.13).

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1M} & b_{1,M+1} & b_{1,M+2} & \cdots & b_{1,M+N'} \\ b_{12} & b_{22} & \cdots & b_{2M} & b_{2,M+1} & b_{2,M+2} & \cdots & b_{2,M+N'} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{NM} & b_{N,M+1} & b_{N,M+2} & \cdots & b_{N,M+N'} \end{bmatrix}_{N \times (M+N')} \tag{6.13}$$

where $b_{i,j+M} = c_{ij} = P(S_j^d|S_i^s), (i = 1, 2, \cdots, N, j = 1, 2, \cdots, N')$. The Baum-Welch estimation is carried out again to train the parameters in θ_k^s (see Section 2.4.1). After a number of EM iterations, the ML source model for d_k, θ_k^s, is obtained.

6.2.2 Viseme generation

A viseme production can be mapped from the source speaker to the destination speaker with the HMMs obtained in Section 6.2.1. Assume that $y = (y_1^s, y_2^s, \cdots, y_T^s)$ is a T-length sequence indicating the production of viseme k by the source speaker. An observation sequence $y' = (y_1^d, y_2^d, \cdots, y_T^d)$ indicating the production of the same viseme by the destination speaker is generated with the following steps. For simplification, y is referred to as the source sequence and y' is referred to as the destination sequence.

1.) Given y and θ_k^s, the optimal state chain $s_s^T = (s_1^s, s_2^s, \cdots, s_T^s)$ is decoded for the source speaker using the Viterbi search.

2.) Using s_s^T and Matrix C, a state chain $s_d^T = (s_1^d, s_2^d, \cdots, s_T^d)$ for the destination speaker is generated that maximizes the probability $P(s_d^T|s_s^T)$, which is defined in Eq.(6.14).

$$P(s_d^T|s_s^T) = \prod_{t=1}^{T} P(s_t^d|s_t^s) \tag{6.14}$$

3.) An observation sequence $y' = (y_1^d, y_2^d, \cdots, y_T^d)$ for the destination speaker is then generated by the state chain s_d^T and the symbol output matrix of θ_k^d.

The mapping of the source sequence to the destination sequence is thus realized. However, the approach mentioned above may generate lip shapes with abrupt change in consecutive frames. To solve this problem, some restrictions are imposed to the destination model. For θ_k^d, if the decoded state at time t is $s_t = S_i, (i = 1, 2, \cdots, N')$ and the symbol o_{t-1} is generated at time $t-1$, the symbol output coefficient is modified using Eq.(6.15).

$$b'_{ij} = \frac{b_{ij} e^{-\lambda D(o_{t-1}, O_j)}}{\omega} \tag{6.15}$$

where $b_{ij} = P(O_j|S_i)$ is the actual symbol output probability of θ_k^d, $D(o_{t-1}, O_j)$ indicates the Euclidean distance between o_{t-1} and O_j, ω is a normalization factor to make $b'_{ij}, (j = 1, 2, \cdots, M')$ a distribution. For the discrete symbol set used to characterize the lip shapes during viseme production, o_{t-1} and O_j are code words of the code book mentioned in Section 2.3.2, i.e. $o_{t-1}, O_j \in O^{128} = \{O_1, O_2, \cdots, O_{128}\}$. A viseme production is then generated for the destination speaker using the new symbol output coefficient b'_{ij}.

In Eq.(6.15), λ is a positive constant that controls the contribution of $D(o_{t-1}, O_j)$ to b'_{ij}. If greater value is selected for λ, b'_{ij} will be smaller and the generated sequence will be smoother. However, the modified value b'_{ij} will deviate further from the original value b_{ij}. The setting of θ_k^d is thus violated to some extent. If smaller λ is adopted, the setting of θ_k^d is better kept. However, the generated sequence may not be smooth. If the sequence of codes is mapped back into video frames, the movement of the lips may demonstrate sudden changes in consecutive frames. In application, the selection of λ is a tradeoff between the requested smoothness of the generated sequences and the fidelity of the destination process. For the experiments conducted in this section, the distance $D(o_{t-1}, O_j)$ in Eq.(6.15) is of the order of 10^3. λ is chosen within the range of $10^{-3} \sim 10^{-4}$.

Table 6.1: The recognition rates of the mapped visemes

Viseme	Source speaker			True samples of the
	Speaker 1	Speaker 2	Speaker 3	destination speaker
p, b, m	0.75	0.75	0.65	87%
tS, dZ, S	1	1	0.80	90%
f, v	0.80	0.55	0.75	96%
A:	0.85	1	1	99%
U	1	0.95	0.95	93%

6.2.3 Experimental results

Experiments are conducted to test the performance of the proposed viseme mapping strategy. A selected number of context-independent visemes produced by three speakers are mapped to a destination speaker. The accuracy of such mapping is studied. According to the decision strategy given in Section 2.4.3, if the mapped destination viseme can be correctly identified by the true destination viseme model, a correct classification is made; otherwise, an error occurs. For each source speaker (Speaker 1, Speaker 2 and Speaker 3), twenty samples are drawn for each viseme given in Table 6.1, and twenty mapped samples are obtained and identified by the viseme model of the destination speaker. The recognition rates of the mapped samples and the recognition rates of the true samples of the destination speaker (see Table 2.2, θ^2 of Speaker 1) are listed in Table 6.1 for comparison.

The recognition results indicate that the mapped viseme productions can be recognized by the viseme models of the destination speaker. The average recognition rate is about 85%, which is slightly lower than that of the true samples. It thus concludes that the mapped samples demonstrate similar temporal and statistical features as the true samples of the destination speaker.

6.2.4 Summary

The strategy proposed in this section is a simple method of mapping visemes between two speakers. By training some mapping terms for the HMM, the state chain of the source model is associated with that of the destination model. A viseme produced by a source speaker can then be mapped to a destination speaker with these mapping terms. The proposed method cannot eliminate the speaker-dependency of a visual speech processing system. However, it may give some useful clues for further research. To analyze the visual speech of an unknown speaker, a possible approach is to map the acquired visual speech signal to a known speaker. For HMM-based classifiers, such mapping can be performed on the states of the HMM. By bounding the states of the viseme models of the unknown speaker to those of a known speaker, much information about the unknown speaker can be learned. The proposed viseme mapping approach is only the first step toward the elimination of speaker-dependency, where a viseme is generated by the viseme models of both the source speaker and the destination speaker. In the next step, we attempt to analyze the connected viseme units of an unknown speaker using the HMMs with mapping terms.

Chapter 7

Conclusions and Future Directions

The studies reported in this thesis attempt to solve some basic problems of visual speech processing, which include the construction and training of HMM classifiers, recognition of the basic visual speech elements, and modeling and recognition of the continuous visual speech units. Two minor research topics about 3D lip tracking and mapping visual speech between different speakers are also covered in the thesis.

From an overall standpoint, the proposed visual speech processing system follows a bottom-to-top scheme, i.e. recognition of the basic visual speech elements is first performed; following that, recognition of the connected-viseme units such as words and phrases are implemented.

The approaches for recognizing the basic visual speech elements are based on Hidden Markov Model (HMM). Traditional single-HMM classifier that is trained using the Baum-Welch method is explored first. This kind of HMM classifier can be easily obtained and is able to identify context-independent visemes defined in MPEG-4 Standards with good accuracy. However, single-HMM classifiers cannot distinguish confusable visual speech units such as the visual representations of phonemes that are categorized into the same viseme group. To improve the discriminative power of the HMM, a new metric called separable distance is proposed to describe the discriminative power of an HMM. Based on the separable distance, two discriminative training strategies, referred to as two-channel training strategy and maximum separable distance (MSD) training strategy are proposed. These two approaches employ expectation-maximization (EM) iterations to modify the parameters of an HMM for greater separable distance. The experimental results on identifying confusable visual speech units and confusable words in visual speech indicate that the approaches can effectively improve the discriminative power of an HMM classifier. However, the proposed training strategies may decrease the probability of the true samples given the HMM. It indicates that the HMMs obtained in these ways may not provide a good fit for the models of the target signals. In application, the two-channel HMM classifiers or MSD HMM classifiers have to be used in conjunction with other classifiers and principally conduct fine recognition within a group of confusable patterns.

The single-HMM classifier is also not robust enough for identifying samples with spread-out distribution. This is validated by our experiments of identifying context-dependent visemes, where the temporal features of a viseme are distorted by its contexts. To improve the robustness of HMM classifiers, an adaptive boosting (AdaBoost) technique is applied to HMM modeling to construct a multi-HMM classifier. The composite HMMs of the multi-HMM classifier are trained using the biased Baum-Welch estimation, and the weights assigned to the training samples and the composite HMMs are modified with AdaBoost iterations. Such a multi-HMM classifier, which is referred to as AdaBoost-HMM classifier, is able to cover the erratic samples of a viseme by synthesizing the sub-decisions made by the composite HMMs. For the experiments carried out in the thesis, the samples of context-independent visemes, which are similar with one another, and context-dependent visemes, which demonstrate spread-out distribution, are recognized by

AdaBoost-HMM classifiers and traditional single-HMM classifiers. The comparative results indicate that the recognition accuracy of context-independent visemes using AdaBoost-HMM classifiers are close to that using the single-HMM classifiers, while for the context-dependent visemes, the average recognition rate of AdaBoost-HMM classifiers is 16% higher than that of the single-HMM classifiers. The cost for the improvement to the robustness is the increased computational load. Because multiple HMMs have to be trained in the HMM AdaBoosting strategy, the computations involved are many times that of building a single-HMM classifier.

The viseme classifiers mentioned above have laid a basis for further studies of visual speech recognition, based on which, recognition of connected viseme units such as words, phrases or connected digits in visual speech are carried out. The approaches reported in the thesis are to decompose the sequence indicating the production of a connected viseme unit into visemes. For this purpose, level building on single-HMM classifiers is first explored. The level building method applies a probability trellis to store the accumulated probabilities and positions of the reference models. The best-matched sequence of HMMs is searched by backtracking the probability trellis. For the AdaBoost-HMM classifiers, level building strategy faces the difficulty of synchronizing the composite HMMs of an AdaBoost-HMM classifier. This problem is solved by introducing special end nodes to the probability trellis, where the composite HMMs are aligned and the scored probabilities are synthesized. The strategies of level building on single-HMM classifiers and level building on AdaBoost-HMM classifiers are applied to recognize/decompose a number of words and phrases in visual speech. The experimental results indicate that the approach using AdaBoost-HMM classifiers has higher recognition/decomposition accuracy than that using single-HMM classifiers.

Level building method is an exhaustive searching algorithm to match the states of each reference models against each node of the probability trellis. As a result, the computational load is heavy whether the reference models are single-HMM classifiers or AdaBoost-HMM classifiers. To facilitate the process of sequence decomposition, a Viterbi matching algorithm is proposed in the thesis. This approach applies specially tailored recognition units and transition units to model the viseme productions and transitions between viseme productions. A sequence of recognition units and transition units are decoded using Viterbi algorithm for the target sequence. Although this approach does not work well for decomposing words/phrases in visual speech, where the transitions between visemes are ambiguous, it performs almost equally well as level building method for decomposing connected digits in visual speech, where the intervals between the productions of digits are distinct. Furthermore, the computational load of the Viterbi approach is less than one fourth of the level building method.

The ultimate goal of visual speech processing is to recognize continuous visual speech such as sentences or paragraphs. In this thesis, however, only the recognition of the basic visual speech elements and connected viseme units are realized. The extension of the proposed HMM techniques to process continuous visual speech is one of the important directions of our future research. For this purpose, the application of phonetic, lexical and semantic rules to HMM modeling should be explored. The prospective approaches may include frame synchronized methods that are popular in modern acoustic speech processing systems.

The development of visual speech processing is still at the early stage. As a result, much work has to be carried out to meet the requirement of building universal visual speech processing system. In this thesis, studies on lip tracking and eliminating speaker-dependency are presented. The proposed 3D lip tracking method applies 3D deformable template and template trellis to capture the movement of the lips. This approach excels the traditional 2D deformable template method as it can well compensate the deformation caused by the movement of the speaker's head. The proposed visual speech mapping strategy adopts HMMs with special mapping terms to map the viseme productions from a source speaker to a destination speaker. Experiments show that the mapped visemes can be accurately recognized by the true models of the destination speaker. These two approaches hold the potential of extending the applicability of a visual speech recognition system to unfavorable environments such as when the speaker's head is moving during speech or when the visual features, e.g. the shape of the lips, of a speaker are unknown. In this thesis, however, only preliminary researches are conducted.

The proposed visual speech processing system has limited applications by itself as only the visual aspect of speech is processed while the acoustic aspect is not considered. In most automatic lip reading systems reported in the literature and also the system proposed in this thesis, a huge amount of image data have to be processed first. Real-time recognition may then be a problem. Acoustic speech signals, on the other hand, do not involve pre-processing of such a huge amount of data and real-time acoustic speech recognition is already possible for some speech processing systems. To jointly process audio and visual signals, lip synchronization has to be investigated. Lip synchronization problem was put forward during 1990s and was considered partially solved with the "talking head" approaches that are included in MPEG-4 [88][110][111]. However, it is inadequate for automated lip reading and mapping between visual speech and acoustic speech at real time. Study on lip synchronization should be further carried out.

The speech information conveyed by the movement of the lips is far less than that of the acoustic signals. As a result, a reliable speech processing system cannot rely solely on the visual aspect of speech. Incorporation of audio recognition engine to the visual speech processing system is necessary. In our future work, the development of bimodal audio-visual speech recognition system will also be an important research direction.

Bibliography

[1] W. H. Sumby and I. Pollack. Visual contributions to speech intelligibility in noise. *Journal of the Acoustical Society of America*, 26:212–215, 1954.

[2] K. Neely. Effect of visual factors on the intelligibility of speech. *Journal of the Acoustical Society of America*, 28(6):1275–1277, 1956.

[3] C. Binnie, A. Montgomery and P. Jackson. Auditory and visual contributions to the perception of consonants. *Journal of Speech Hearing and Research*, 17:619–630, 1974.

[4] D. Reisberg, J. McLean and A. Goldfield. Easy to hear, but hard to understand: A lip-reading advantage with intact auditory stimuli. In B. Dodd and R. Campbell, editors, *Hearing by Eye*, pages 97–113. Lawrence Erlbaum Associates, 1987.

[5] K. P. Green and P. K. Kuhl. The role of visual information in the processing of place and manner features in speech perception. *Perception and Psychophysics*, 45(1):32–42, 1989.

[6] D. W. Massaro. Integrating multiple sources of information in listening and reading. In *Language perception and production.* Academic Press, New York.

[7] R. Campbell and B. Dodd. Hearing by eye. *Quarterly Journal of Experimental Psychology*, 32:85–99, 1980.

[8] B. Dodd. Lipreading in infants: Attention to speech presented in and out of synchrony. *Cognitive Psychology*, 11:478–484, 1979.

[9] P. K. Kuhl and A. N. Meltzoff. The bimodal perception of speech in infancy. *Science*, 218:1138–1141, 1982.

[10] H. McGurk and J. MacDonald. Hearing lips and seeing voices. *Nature*, 264:746–748, 1976.

[11] A. Fuster-Duran. Perception of conflicting audio-visual speech: An examination across spanish and german. In D. Stork and M. Hennecke, editors, *Speechreading by Human and Machines*, pages 103–114. Springer-Verlag, Berlin, Germany, 1996.

[12] K. P. Green. The use of auditory and visual information in phonetic perception. In D. Stork and M. Hennecke, editors, *Speechreading by Human and Machines*, pages 55–78. Springer-Verlag, Berlin, Germany, 1996.

[13] R. D. Easton and M. Basala. Perceptual dominance during lipreading. *Perception Psychophys*, 32:562–570, 1982.

[14] B. Dodd and R. Campbell, editors. *Hearing by Eye: The Psychology of Lipreading.* Lawrence Erlbaum, London, 1987.

[15] D. Stork and M. Hennecke, editors. *Speechreading by Human and Machines: Models, Systems, and Applications.* Springer-Verlag, Berlin, Germany, 1996.

[16] B. Dodd, R. Campbell and D. Burnham, editors. *Hearing by eye II : advances in the psychology of speechreading and auditory-visual speech.* Psychology Press, Hove, East Sussex, UK, 1998.

[17] M. Hennecke, D. Stork and K. Prasad. Visionary speech: Looking ahead to practical speechreading systems. In D. Stork and M. Hennecke, editors, *Speechreading by Humans and Machines: Models, Systems and Applications*, pages 331–349. Springer-Verlag, 1996.

[18] A. Q. Summerfield. Some preliminaries to a comprehensive account of audio-visual speech perception. In B. Dodd and R. Campbell, editors, *Hearing by Eye*, pages 3–51. Lawrence Erlbaum Associates, 1987.

[19] B. P. Yuhas, M. H. Goldstein and T. J. Sejnowski. Integration of acoustic and visual speech signals using neural networks. *IEEE Communication Magazine*, pages 65–71, 1989.

[20] E. D. Petajan. *Automatic lipreading to enhance speech recognition.* PhD thesis, University of Illinois at Urbana-Champaign, 1984.

[21] K. Prasad, D. Stork and G. Wolff. Preprocessing video images for neural learning of lipreading. Technical Report Technical Report CRC-TR-93-26, Ricoh California Research Center, 1993.

[22] S. Lucey, S. Sridharan and V. Chandran. Initialized eigenlip estimator for fast lip tracking using linear regression. In *International Conference on Pattern Recognition*, pages 182–185, Barcelona, Sep. 2000.

[23] M. Kass, A. Witkin and D. Terzopoulus. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[24] M. U. Ramos Sanchez, J. Matas and J. Kittler. Statistical chromaticity-based lip tracking with b-splines. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2973–2976, Munich, Germany, Apr. 1997.

[25] R. Cipolla and A. Blake. The dynamic analysis of apparent contours. In *3rd International Conference on Computer Vision*, pages 616–623, 1990.

[26] M. Vogt. Fast matching of a dynamic lip model to color video sequences under regular illumination conditions. In D. Stork and M. Hennecke, editors, *Speechreading by Humans and Machines: Models, Systems and Applications*, pages 399–407. Springer-Verlag, 1996.

[27] Hyewon Pyun, Hyun Joon Shin, Tae Hoon Kim and Sung Yong Shin. Real-time facial expression capture for performance-driven animation. Technical Report Technical Report CS-TR-2001-167, Computer Science Department, Korea Advanced Science and Technology, 2001.

[28] C. Bregler and S. M. Omohundro. Surface learning with applications to lipreading. In J. D. Cowan, G. Tesauro and J. Alspector, editors, *Advances in Neural Information Precessing Systems 6*. Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[29] C. Bregler and Y. Konig. Eigenlips for robust speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 669–672, 1994.

[30] A. Lanitis, C. Taylor and T. Cootes. Automatic tracking, coding and reconstruction of human faces using flexible appearance models. *IEE Electronic Letters*, 30:1578–1579, 1994.

[31] A. L. Yuille and P. Hallinan. Deformable templates. In Andrew Blake and Alan Yuille, editors, *Active Vision*, pages 21–38. The MIT Press, 1992.

[32] M. E. Hennecke, K. V. Prasad and D. G. Stork. Using deformable templates to infer visual speech dynamics. In *The 28th Asilomar Conf. on Signals, Systems and Computers*, pages 578–582, 1994.

[33] A. L. Yuill,e P. Hallinan and D. S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 2(8):99–112, 1992.

[34] C. Kervrann and F. Heitz. A hierarchical markov modeling approach for the segmentation and tracking of deformable shapes. *Graphical Models and Image Processing*, 60(3):173–195, May 1998.

[35] Mun Wai Lee and Surendra Ranganath. Pose-invariant face recognition using a 3D deformable model. *Pattern Recognition*, 36:1835–1846, 2003.

[36] T. F. Cootes, A. Hill, C. J. Taylor and J. Haslan. Use of active shape models for locating structures in medical images. *Image and Vision Computing*, 12(6):355–365, 1994.

[37] J. Luettin and N. A. Thacker. Speechreading using probabilistic methods. *Computer Vision and Image Understanding*, 65(2):163–178, Feb. 1997.

[38] J. A. Nelder and R. Mead. A simplex method for function optimization. *Computing Journal*, 7(4):308–313, 1965.

[39] T. F. Cootes, C. J. Taylor and A. Lanitis. Multi-resolution search using active shape models. In *The 12th International Conference on Pattern Recognition*, pages 610–612, 1994.

[40] T. F. Cootes, C. J. Taylor and G. J. Edwards. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, Jun. 2001.

[41] K. Mase and A. Pentland. Lip reading: Automatic visual recognition of spoken words. In *Optical Society of America Topical Meeting on Machine Vision*, pages 1565–1570, 1989.

[42] E. D. Petajan, B. J. Bischoff, D. A. Bodoff and N. M. Brooke. An improved automatic lipreading system to enhance speech recognition. Technical Report Technical Report TM 11251-871012-11, Bell Labs, 1987.

[43] B. P. Yuhas, M. H. Goldstein, T. J. Sejnowski and R. E. Jenkins. Neural network models of sensory integration for improved vowel recognition. *Proceedings of the IEEE*, 78(10):1658–1668, 1990.

[44] D. G. Stork, G. Wolff and E. P. Levine. Neural network lipreading system for improved speech recognition. In *International Joint Conference on Neural Network*, pages 285–295, 1992.

[45] P. Cosi, M. Dugatto, E. Magno Caldognetto, K. Vagges, G. A. Mian and M. Contolini. Bimodal recognition experiments with recurrent neural networks. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 553–556, 1994.

[46] J. R. Movellan. Visual speech recognition with stochastic networks. In D. Touretzky G. Tesauro and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 851–858. The MIT Press, Cambridge, MA, 1995.

[47] S. Choi, H. Hong, H. Glotin and F. Berthommier. Multichannel signal separation for cocktail party speech recognition: a dynamic recurrent network. *Neurocomputing*, 49(1-4):299–314, 2002.

[48] T. TobeIyt, N. Tsurutat and M. Amamiyat. On-line speech-reading system for japanese language. In *9th International Conference on Neural Information Processing*, volume 3, pages 1188–1193, 2002.

[49] M. Gordan, C. Kotropoulos and I. Pitas. A support vector machine-based dynamic network for visual speech recognition applications. *EURASIP Journal on Applied Signal Processing*, (11):1248–1259, 2002.

[50] G. Potamianos, C. Neti, J. Luettin and I. Matthews. Audio-visual automatic speech recognition: An overview,. In G. Bailly E. Vartikiotis-Bateson and P. Perrier, editors, *Audio-Visual Speech Processing*, pages 121–148. The MIT Press, 2003.

[51] A. Hagen and A. Morris. Comparison of HMM experts with MLP experts in the full combination multi-band approach to robust ASR. In *Int. Conf. Spoken Language Processing*, volume 1, pages 345–348, China, 2000.

[52] Y. Q. Chen, W. Gao, Z. Q. W, J. Miao and D. L. Jiang. Mining audio/visual database for speech driven face animation. In *Int. Conf. System, Man and Cybernetics*, pages 2638–2643, Oct. 2001.

[53] T. Chen J. S. Kim. Segmentation of image sequences using sofm networks. In *15th Int. Conf. Pattern Recognition*, volume 3, pages 3877–3880, 2000.

[54] K. Saenko, K. Livescu, J. Glass and T. Darrell. Production domain modeling of pronunciation for visual speech recognition. In *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, volume 3, US, March 2005.

[55] A. J. Goldschen. *Continuous Automatic Speech Recognition by Lipreading.* PhD thesis, Dept. of Electrical Engineering and Computer Science, George Washington University, 1993.

[56] A. Adjoudani and C. Benoit. Audio-visual speech recognition compared across to architectures. In *4th European Conference on Speech Communication and Technology*, volume 2, pages 1563–1566, Madrid, 1995.

[57] P. L. Silsbee and A. C. Bovic. Visual lipreading by computer to improve automatic speech recognition accuracy. Technical Report Technical Report TR-93-02-90, University of Texas Computer and Vision Research Center, Austin, TX, 1993.

[58] P. L. Silsbee and A. C. Bovic. Medium vocabulary audiovisual speech recognition. In *NATO ASI New Advances and Trends in Speech Recognition and Coding*, pages 13–16, 1993.

[59] I. Matthews, T. F. Cootes, J. A. Bangham, S. Cox and R. Harvey. Extraction of visual features for lipreading. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(2):198–213, Feb. 2002.

[60] M. Tomlinson, M. Russell and N. Brooke. Integrating audio and visual information to provide highly robust speech recognition. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 821–824, 1996.

[61] J. Luettin, N. A. Thacker and S. W. Beet. Speechreading using shape and intensity information. In *Int. Conf. on Spoken Language Processing*, pages 58–61, 1996.

[62] Xiaozheng Zhang, R. M. Mersereau and M. A. Clements. Audio-visual speech recognition by speechreading. In *14th Int. Conf. on Digital Signal Processing*, volume 2, pages 1069 –1072, 2002.

[63] G. Gravier, G. Potamianos and C. Neti. Asynchrony modeling for audio-visual speech recognition. In *Int. Conf. of Human Language Technology*, USA, 2002.

[64] A. V. Nefian, L. Liang X. Pi, X. Liu and K. Murphy. Dynamic bayesian networks for audio-visual speech recognition. *EURASIP Journal of Applied Signal Processing*, 2002(11):1274–1288, 2002.

[65] S. Dupont and J. Luettin. Audio-visual speech modeling for continuous speech recognition. *IEEE Trans. Multimedia*, 2(3):141–151, 2000.

[66] Tsuhan Chen. Audiovisual speech processing. *IEEE Signal Processing Magazine*, pages 9–21, Jan. 2001.

[67] J. J. Williams and A. K. Katsaggelos. An HMM-based speech-to-video synthesizer. *IEEE Trans. Neural Networks*, 13(4):900–915, Jul. 2002.

[68] D. G. Stork and H. L. Lu. Speech reading by boltzmann zippers. In *Machines that learn*, volume 7. Snowbird, UT, 1996.

[69] J. K. Baker. The dragon system - an overview. *IEEE Trans. on Acoustics, speech and Signal Processing*, 23:24–29, Feb. 1975.

[70] L. R. Bahl, S. B. Aiyer, J. R. Bellgarda, M. Franz, P. S. Gopalakrishnan, D. Nahamoo, M. Novak, M. Padmanabhan, M. A. Picheny and S. Roukos. Performance of the ibm large vocabulary speech recognition system on the arpa wall street journal task. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 41–44, 1995.

[71] P. D'Orta, M. Ferretti, A. Martelli, S. Melecrinis, S. Scarci and G. Volpi. Large-vocabulary speech recognition: A system for the Italian language. *IBM Journal of Research and Development*, 32(2):217–226, Mar. 1988.

[72] P. Cosi, J. P. Hosom and F. Tesser. High performance italian continuous digit recognition. In *International Conference on Spoken Language Processing*, pages 242–245, Beijing, Oct. 2000.

[73] Petr Cisar, Zdenek Krnoul, Jan Novak and Milos Zelezny. Approach to an audio-visual speech synthesis using concatenation-based method. In *Speech Processing of the 12th Czech-German Workshop*, pages 64–65, Prague, Czech Republic, 2002.

[74] P. Silsbee and A. Bovik. Computer lipreading for improved accuracy in automatic speech recognition. *IEEE Trans. Speech and Audio Processing*, 4(5):337–351, 1996.

[75] A. Adjoudani and C. Benoit. On the integration of auditory and visual parameters in an HMM-based asr. In D. Stork and M. Hennecke, editors, *Speechreading by Humans and Machines: Models, Systems and Applications*, pages 461–471. Springer-Verlag, 1996.

[76] A. Verma, T. Faruquie, C. Neti, S. Basu and A. Senior. Late integration in audio-visual continuous speech recognition. In *Proc. of Automatic Speech Recognition and Understanding*, Colorado, Dec. 1999.

[77] T. H. Chen and R. R. Rao. Audio-visual integration in mulimodal communication. *Proceedings of the IEEE*, 86(5):837–852, 1998.

[78] Available online, URL: http://www.intel.com/pressroom/archive/releases/20030428tech.htm.

[79] C. H. La and T. J. Hazen. Integration of audio and visual information for speech recognition. Technical report, MIT Computer Science and Artificial Intelligence Laboratory, 2004.

[80] R. Rodman, D. McAllister and D. Bitzer. Lip synchronization as an aid to the hearing disabled. In *Proc. of the American Voice Input/Output Society*, pages 233–248, 1997.

[81] K. Waters and T. M. Levergood. Decface: An automatic lip-synchronization algorithm for synthetic faces. Technical Report CRL-93-4, Digital Equipment Corporation, Cambridge Research Lab, Aug. 1993.

[82] C. C. Chibelushi, S. Gandon, J. S. D. Mason, F. Deravi and R. D. Johnston. Design issues for a digital audio-visual integrated database. In *IEE Colloquium on Integrated Audio-Visual Processing*, number 1996/213, pages 7/1–7/7, 1996.

[83] G. Potamianos, F. Cosatto, H. P. Graf and D. B. Roe. Speaker independent audio-visual database for bimodal asr. In *ESCA Workshop Audio-Visual Speech Processing*, pages 65–68, Nov. 1997.

[84] Available online, URL: http://www.tele.ucl.ac.be/PROJECTS/M2VTS/m2fdb.html.

[85] C. Binnie, A. Montgomery and P. Jackson. Auditory and visual contributions to the perception of consonants. *Journal of Speech Hearing and Research*, 17:619–630, 1974.

[86] G. W. Greenwood. Training partially recurrent neural networks using evolutionary strategies. *IEEE Trans. Speech and Audio Processing*, 5(2):192–194, 1997.

[87] E. Owens and B. Blazek. Visemes observed by hearing impaired and normal hearing adult viewers. *Journal of Speech Hearing and Research*, 28:381–393, 1985.

[88] A. M. Tekalp and J. Ostermann. Face and 2-D mesh animation in MPEG-4. *Signal Processing: Image Communication, Special Issue on MPEG-4*, 15:387–421, Jan. 2000.

[89] S. Morishima, S. Ogata, K. Murai and S. Nakamura. Audio-visual speech translation with automatic lip synchronization and face tracking based on 3-D head model. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 2117–2120, May 2002.

[90] Y. Pan, J. T. Wu, S. Tamura, H. Mitsumoto, H. Kawai, K. Kurosu and K. Okazaki. Neural network vowel-recognition jointly using voice features and mouth shape image. *Pattern Recognition*, 24(9):921–927, 1991.

[91] X. Z. Zhang, C. Broun, R. M. Mersereau and M. A. Clements. Automatic speechreading with applications to human-computer interfaces. *EURASIP Journal on Applied Signal Processing, Special Issue on Joint Audio-Visual Speech Processing*, pages 1228–1247, 2002.

[92] T. W. Lewis and D. M. W. Powers. Lip feature extraction using red exclusion. In *Selected papers from the Pan-Sydney Workshop on Visualization*, volume 2, pages 61–67, 2000.

[93] T. Coianiz, L. Torresani and B. Caprile. 2D deformable models for visual speech analysis. In D. Stork and M. Hennecke, editors, *Speechreading by Humans and Machines: Models, Systems and Applications*, pages 391–398. Springer-Verlag, 1996.

[94] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, pages 1554–1563, 1966.

[95] L. E. Baum and G. R. Sell. Growth functions for transformations on manifolds. *Pacific Journal of Mathematics*, (2):211–227, 1968.

[96] T. Petrie. Probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, (1):97–115, 1969.

[97] L. E. Baum, T. Petrie, G. Soules and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statist.*, pages 164–171, 1970.

[98] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, pages 1–8, 1972.

[99] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, (2):257–286, Feb. 1989.

[100] L. R. Rabiner and B. H. Juang. *Fundamentals of speech recognition.* Prentice Hall International Inc., 1993. Signal Processing Series.

[101] G. J. McLachlan and T. Krishnan. *The EM algorithm and extensions.* Wiley, New York, 1996.

[102] L. R Bahl, P. F. Brown, P. V. de Souza and R. L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 49–52, Apr. 1986.

[103] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches.* John Wiley and Sons Inc., 1992.

[104] R. E. Schapire. A brief introduction to boosting. In *16th Int. Joint Conf. on Artificial Intelligence*, pages 1401–1405, 1999.

[105] R. E. Schapire. Theoretical views of boosting and applications. In *10th Int. Conf. on Algorithmic Learning Theory*, pages 13–25, 1999.

[106] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *11th Annual Conf. on Computational Learning Theory*, pages 80–91, 1998.

[107] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Science*, (1):119–139, Aug. 1997.

[108] L. M. Arslan and J. H. L. Hansen. Selective training in hidden markov model recognition. *IEEE Trans. Speech and Audio Processing*, (1):46–54, Jan. 1999.

[109] C. H. Lee and L. R. Rabiner. A frame-synchronous network search algorithm for connected word recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, (11):1649–1658, Nov. 1989.

[110] Available online, URL: http://www.cs.rutgers.edu/ village/ruth/.

[111] T. Firsova, D. Ivanov, V. Kuriakin, E. Martinova, K. Rodyushkin and V. Zhislina. Life-like MPEG-4 3D "Talking Head" (beyond standard). In *5th Int. Conf. Computer Graphics and Artificial Intelligence*, France, May 2002.

[112] J. Hershey, H. Attias, N. Jojic and T. Kristjansson. Audio-visual graphical models for speech processing. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 5, pages 649–652, 2004.

Name: Dong Liang

Degree: Doctor of Philosophy

Department: Electrical and Computer Engineering

Thesis Title: Hidden Markov Model Based Visual Speech Recognition

Abstract

Speech recognition can be made more accurate if visual speech information such as the movement of the lips is taken into consideration. In this thesis, studies on visual speech processing are presented. Classifiers based on Hidden Markov Model (HMM) are first explored for modeling and identifying the basic visual speech elements. Considering that the temporal features of visual speech elements may be confusable and sensitive to their contexts, three novel training strategies, referred to as two-channel training strategy, Maximum Separable Distance training strategy and HMM Adaptive Boosting strategy, are proposed to improve the discriminative power and robustness of an HMM classifier. Following that, approaches for recognizing words, phrases and connected-digit units in visual speech are presented with exploration of level building method and Viterbi searching algorithm. The thesis also covers the studies of 3D lip tracking and visual speech mapping between different speakers. These approaches may extend the applicability of a visual speech processing system to unfavorable conditions.

Keywords:

visual speech recognition, Hidden Markov Model, separable distance, Adaptive Boosting, level building, lip tracking

Publications

Journal papers

1.) Say Wei Foo, Yong Lian and Liang Dong, "*Recognition of Visual Speech Elements Using Adaptively Boosted Hidden Markov Models,*" IEEE Trans. Circuits and Systems for Video Technology, Volume: 14, Issue: 5, Page(s): 693-705, May 2004

2.) Liang Dong, Say Wei Foo and Yong Lian, "*Level-building on AdaBoost HMM classifiers and the application to visual speech processing,*" IEICE Trans. Information and Systems, Volume: E87-D, No. 11, Page(s): 2460-2472, Nov. 2004.

3.) Liang Dong, Say Wei Foo and Yong Lian, "*Maximum separable distance estimation for hidden Markov model and its application to visual speech processing,*" GESTS Int'l Trans. Speech Science and Engineering, Volume: 2, No. 1, Feb. 2005

4.) Liang Dong, Say Wei Foo and Yong Lian, "*A two-channel training algorithm for Hidden Markov Model and its application to lip reading,*" Accepted by EURASIP Journal on Applied Signal Processing in Apr. 2004. To be published in June 2005.

Conference papers

5.) Say Wei Foo and Liang Dong, "*A boosted multi-HMM classifier for recognition of visual speech elements,*" IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03), Volume: 2, Page(s): 285-288, Apr. 2003

6.) Say Wei Foo, Yong Lian and Liang Dong, "*A two-channel training algorithm for hidden Markov model to identify visual speech elements,*" International Symposium on Circuits and Systems (ISCAS '03), Volume: 2, Page(s): 572-575, May 25-28, 2003

7.) Liang Dong, Say Wei Foo and Yong Lian, "*Cross-Speaker Viseme Mapping Using Hidden Markov Models,*" The 4th Int. Conf. Information, Communications and Signal Processing and the 4th IEEE Pacific-Rim Conf. Multimedia (ICICS-PCM 2003), Volume: 3, Page(s): 1384-1388, Dec. 2003

8.) Liang Dong, Say Wei Foo and Yong Lian, "*Modeling Continuous Visual Speech Using Boosted Viseme Models,*" The 4th Int. Conf. Information, Communications and Signal Processing and the 4th IEEE Pacific-Rim Conf. Multimedia (ICICS-PCM 2003), Volume: 3, Page(s): 1394-1398, Dec. 2003

9.) Say Wei Foo, Yong Lian and Liang Dong, "*Using 3D Deformable Template Trellis to Describe the Movement of the Lip,*" 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2003). Session 5, UK, 2003

10.) Say Wei Foo, Yong Lian and Liang Dong, "*A Simplified Viterbi Matching Algorithm for Word Partition in Visual Speech Processing,*" 4th European Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2003). Session 5, UK, 2003

11.) Say Wei Foo and Liang Dong, "*Recognition of Visual Speech Elements Using Hidden Markov Models,*" Advances in Multimedia Information Processing, Proceedings of 3rd IEEE Pacific Rim Conf. on Multimedia, Taiwan, Page(s): 607-614, 2002

12.) Liang Dong and Say Wei Foo, "*Boosting of Hidden Markov Models and the application to visual speech analysis,*" Recent Trends in Multimedia Information Processing, Proceedings of the 9th Int. Workshop on Systems, Signals and Image Processing, Page(s): 358-364. UK, 2002

13.) Say Wei Foo and Liang Dong, "*A supervised two-channel learning method for Hidden Markov Model and application to lip reading,*" IEEE Int. Conf. on Advanced Learning Technologies, Page(s): 334-338, Russia, 2002

HIDDEN MARKOV MODEL BASED VISUAL SPEECH RECOGNITION

DONG LIANG

NATIONAL UNIVERSITY OF SINGAPORE

2004

ODEN MARKOV MODEL BASED
SUAL SPEECH RECOGNITION DONG LIANG 2004