Automatic Recognition of Facial Expressions Using Hidden Markov Models and Estimation of Expression Intensity

Jenn-Jier James Lien

CMU-RI-TR-98-31

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

April 14, 1998

COMMITTEE SIGNATURE PAGE

This dissertation was presented

by

Jenn-Jier James Lien

It was defended on

April 14, 1998

and approved by

(Signature) _____
Committee Co-Chair
Dr. Ching-Chung Li, Professor of Electrical Engineering

(Signature) _____
Committee Co-Chair
Dr. Takeo Kanade, Professor and Director of the Robotics Institute,
School of Computer Science, Carnegie Mellon University

(Signature) _____
Committee Co-Chair
Dr. Jeffrey F. Cohn, Professor of Psychology

(Signature) _____
Committee Member
Dr. Henry Y.H. Chuang, Professor of Computer Science

(Signature) _____
Committee Member
Dr. Richard W. Hall, Professor of Electrical Engineering

(Signature) _____
Committee Member
Dr. Morton Kanefsky, Professor of Electrical Engineering

(Signature) _____
Committee Member
Dr. Marwan A. Simaan, Professor and Chairman of Electrical Engineering

ACKNOWLEDGEMENTS

I am greatly indebted to Professor Takeo Kanade and Professor Ching-Chung Li, my co-advisors and my mentors, for not only providing invaluable guidance, advice, criticism and encouragement but also giving me the latitude I have needed to develop as a researcher. I thank Professor Jeffrey Cohn, co-advisor, for his support and teaching with the Facial Action Coding System (FACS). I would also like to thank the other members of my thesis committee: Professors Richard Hall, Morton Kanefsky, Marwan Simaan and Henry Chuang for their valuable suggestions and feedback.

My years at the Vision and Autonomous Systems Center of the Robotics Institute, Carnegie Mellon University have been priceless. I consider myself lucky to be a part of this research center and spend many nights with friends discussing technical issues. I would like to thank Jie Yang and Michael Nechyba for sharing their experiences with Hidden Markov Models; Conrad Poelman, Richard Madison and Yalin Xiong for exchanging knowledge on optical flows; Peter Rander, Henry Rowley, Shumeet Baluja, Teck Khim, Wei Hua, Mei Han, Mei Chen, Dongmei Zhang, Michael Smith, Daniel Morris and Farhana Kagalwala for much needed help; and Adena Zlochower for her help with FACS on the facial expression analysis project. My thanks also go to Chung-Hui Anne Lin for her support and encouragement. I would especially like to thank David LaRose and Yu-Te Wu, who have been tremendous fun to work and play with, and who have provided countless hours of invaluable discussions on the topics presented here.

Special thanks to Matthew Turk, Steve Shafer, P. Anandan, Richard Szeliski and Harry Shum at Microsoft Vision group for their valuable comments and suggestions.

Finally, I would like to thank my parents, Chin-Chuan Lien and Wen-Hua Shih, as well as my other family members: Jenn-Ren Lien, Jenn-Yueh Lien, Shu-Hua Chien, Hui-Yin Christia Tien for their constant love, support and encouragement. Without them, none of this would have been possible. I cannot begin to thank them enough, and they will always have my respect and love.

ABSTRACT

Signature _____
Professor Ching-Chung Li

Signature _____
Professor Takeo Kanade

Signature _____
Professor Jeffrey F. Cohn

AUTOMATIC RECOGNITION OF FACIAL EXPRESSIONS USING HIDDEN MARKOV MODELS AND ESTIMATION OF EXPRSSION INTENSITY

Jenn-Jier James Lien, Ph.D.

Facial expressions provide sensitive cues about emotional responses and play a major role in the study of psychological phenomena and the development of nonverbal communication. Facial expressions regulate social behavior, signal communicative intent, and are related to speech production. Most facial expression recognition systems focus on

only six basic expressions. In everyday life, however, these six basic expressions occur relatively infrequently, and emotion or intent is more often communicated by subtle changes in one or two discrete features, such as tightening of the lips which may communicate anger. Humans are capable of producing thousands of expressions that vary in complexity, intensity, and meaning. The objective of this dissertation is to develop a computer vision system, including both facial feature extraction and recognition, that automatically discriminates among subtly different facial expressions based on Facial Action Coding System (FACS) action units (AUs) using Hidden Markov Models (HMMs).

Three methods are developed to extract facial expression information for automatic recognition. The first method is facial feature point tracking using the coarse-to-fine pyramid method, which can be sensitive to subtle feature motion and is capable to handle large displacements with subpixel accuracy. The second is dense flow tracking together with principal component analysis, where the entire facial motion information per frame is compressed to a low-dimensional weight vector for discrimination. And the third is high gradient component (*i.e.*, furrow) analysis in the spatio-temporal domain, which exploits the transient variance associated with the facial expression.

Upon extraction of the facial information, non-rigid facial expressions are separated from the rigid head motion components, and the face images are automatically aligned and normalized using an affine transformation. The resulting motion vector sequence is vector quantized to provide input to an HMM-based classifier, which addresses the time warping problem. A method is developed for determining the HMM topology optimal for our recognition system. The system also provides expression intensity estimation, which has significant effect on the actual meaning of the expression.

We have studied more than 400 image sequences obtained from 90 subjects. The experimental results of our trained system showed an overall recognition accuracy of 87%, and also 87% in distinguishing among sets of three and six subtly different facial expressions for upper and lower facial regions, respectively.

DESCRIPTORS

Action Unit (AU)

Computer Vision and Pattern Recognition

Dense Flow

Eigenflow

Expression Intensity Estimation

Facial Action Coding System (FACS)

Facial Expression Recognition

Feature Point Tracking

Furrow Extraction

Hidden Markov Model (HMM)

Human-Computer Interaction (HCI)

Motion Line/Edge Extraction

Optical Flow

Principal Component Analysis (PCA)

Rigid and Non-Rigid Motion

Wavelet-Based Motion Estimation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

Human face is a rich and powerful source of communicative information about human behavior. Facial expression provides sensitive cues about emotional response and plays a major role in human interaction and nonverbal communication. It can complement verbal communication, or can convey complete thoughts by itself. It displays emotion [(35)*], regulates social behavior [(23)], signals communicative intent [(38)], is computationally related to speech production [(70)], and may reveal brain function and pathology [(81)]. Thus, to make use of the information afforded by facial expressions, automated reliable, valid, and efficient methods of measurement are critical.

Computer-vision based approaches to facial expression analysis discriminate among a small set of emotions [(12,13,36,68,82,103)]. This focus follows from the work of Darwin [(32)] and more recently Ekman [(35)], who proposed "six basic emotions" (*i.e.*, joy, surprise, anger, sadness, fear, and disgust), each of which has a prototypic facial expression involving changes in facial features in multiple regions of the face. These basic expressions, however, occur relatively infrequently in everyday life and emotion expression is far more varied. Facial action (detailed facial motion) more often is communicated by subtle changes in one or several discrete features, such as tightening the lips which may communicate anger. In reality, humans are capable of producing thousands of expressions that vary in complexity, intensity, and meaning.

There is a standard anatomically based Facial Action Coding System (FACS) [(34)] developed by psychologists for use in coding facial expressions. With FACS, observers can manually code all possible discrete movements of the face, which are referred to as action units (AUs). AUs individually or in combination can represent all visibly discriminable expressions. Considering the complication of the movements involved and

* Parenthetical references placed superior to the line of text refer to the bibliography.

discrimination of the subtle changes, there is a need to develop an automated system for efficient and quantitative measurement of facial expressions based on FACS, which will make the standardized facial expression measurements more accessible for research in various fields including cognitive and behavior science, psychology, biomedical engineering, teleconferencing and human-computer interface or interaction (HCI).

At the present time, most active communication between human and computer is still in one direction: from human to computer, even though computers may hear human speeches through the use of special audio and speech recognition equipments. Allowing computers to understand human operators through vision will bridge the gap of active communication from the direction of computer to human, which will make computers more active, smart, and friendly. Human face is the richest source of nonverbal communication and the most accessible interface displaying human emotion. To automatically analyze and recognize facial expressions using computers will revolutionize fields which rely on human-computer interaction so that computers will be able to understand whether users feel excited or bored, agrees or disagrees. It will be a great challenge and of practical significance to develop a computer vision system which can automatically recognize a variety of facial expressions and estimate expression intensity.

1.1 Related Works

An increasing number of researchers in computer vision have developed various techniques in providing capabilities for automatic facial expression recognition. We will briefly review the strengths and weaknesses of some major paradigms.

Three-dimensional geometric wireframe (or mesh) face models have been used by Aizawa, Harashima and Saito [1]; Choi, Harashima and Takebe [21]; Essa and Pentland [36]; and Terzopoulos and Waters [91] for facial expression analysis, synthesis and recognition. Essa and Pentland [36] developed two methods to recognize expressions. The first method is to recognize 5 expressions: smile, surprise, raised brow, anger, and disgust by scoring

the dot-product similarity based on 36 peak muscle actuations in comparison to the standard training expression templates but the temporal affect is ignored. The overall recognition rate was 97.8% on 6 subjects with 23 and 48 image sequences for training and testing, respectively. The second method uses the temporal-template matching for two-dimensional gray value images. The time warping is an important consideration which improves the recognition accuracy since the temporal-template matching measures the correlation between testing and standard template image sequences.

In contrast to the use of the complex three-dimensional geometric models, Himer, Schneider, Kost, and Heimann [46]; and Kaiser and Wherle [48] proposed a method for automated detection of facial actions by tracking the positions of attached dots on the face as appeared in an image sequence. Since the shape of dots will deform due to muscle movement during facial expression, it is difficult to locate accurately the corresponding central positions for the deformed dots, and thus affects the tracking accuracy.

Optical flow in two dimensions has been used to track motion and classify basic emotion expression (Black, Yacoob, Jepson and Fleet [12,13]; Mase and Pentland [67,68]; Rosenblum, Yacoob and Davis [82]; and Yacoob and Davis [103]). In work by Mase [68], motions of facial muscles were computed rather than those of facial features. Muscle regions were manually selected by referring to major feature points in the face. Optical flow was computed to extract 12 of the 44 facial muscle movements, which in combination with feature positions were interpreted as appropriate AUs. Mase's approach relies heavily on accurate tracking of the manually selected muscle regions; flow directions within each individual region is averaged to represent the flow direction of that region. However, when the selected area corresponds to a smooth, featureless surface in the face, the optical flow estimation will be unreliable, leading to tracking error. Some selected muscle regions may be difficult to locate manually since they are small and highly mobile. In essence, Mase built a model that is appropriate for synthesizing facial expressions but remains uncertain in analyzing facial expressions. He computed mean and covariance of the optical flow in each local region, and then, based on the highest ratio of between-class

to within-class variability to classify various expressions; the k-nearest-neighbor rule was applied for recognition. His experiments indicated an accuracy of approximately 86% in recognizing five expressions (happiness, anger, surprise, disgust, and unknown) on 1 subject with 20 and 30 training and testing image sequences, respectively.

The work of Yacoob and Davis [103]; and Rosenblum, Yacoob and Davis's [82] are related closely to Mase's in that they used optical flow to track the motion of the surface regions of facial features: brows, eyes, nose and mouth, but not that of the underlying muscle groups. In each facial feature region, the flow magnitude was thresholded to reduce the effect of small computed motions which may be either produced from textureless parts or affected by illumination. The overall flow direction of each region is to conform with the plurality in the neighborhood. The direction of any flow in this region is quantized to one of eight main directions to give a mid-level representation (to match with the dictionary or lookup table of the motion direction for each region of the basic facial action) so as to permit the high-level classification of facial expressions. Yacoob and Davis [103] used this mid-level representation to classify the six basic facial expressions as well as eye blinking. The recognition rate was 88% (except eye blinking for which it was 65%) among 32 subjects with 46 image sequences. Rosenblum, Yacoob and Davis [82] extended Yacoob and Davis's [103] work, based on the similar mid-level representation to recognition of facial expressions of smiling and surprise, using an artificial neural networks with radial basis function (RBF). The recognition rate achieved was 88% for 32 subjects.

Black and Yacoob [12] used a local parameterized model of the image motion to separate and recognize the non-rigid facial expression from the rigid head motion. Their high-level recognition approach was similar to that of Yacoob and Davis's technique [103], which is based on the mid-level index of the motion direction of each facial feature region (brows, eyes and mouth). The mid-level representation was predicted, however, by taking the difference of the motion parameter estimation and a threshold value. Thresholding motion parameters would filter out some subtle motion. Furthermore, different threshold

were used for different motion parameters in the experiment [12]: some were between 0.5 ~ -0.5, and others were between 0.00005 ~ -0.00005. This thresholding method for motion parameters, in effect, reduced reliability and accuracy of the recognition. In their studies of recognition of six basic facial expressions, the average recognition rate was 92% in 40 subjects with 70 image sequences.

Principal component analysis (PCA) has been used previously in gray-value base for recognition expressions of on the forehead and brows (Bartlett, Viola, Sejnowski, Golomb, Larsen, Hager, and Ekman [4]), for face recognition (Kirby and Sirovich [54]; and Turk and Pentland [93]), and for object recognition with varying poses (rigid motion) and illumination (Murase and Nayar [73]). It has also been used in optical flow base for recognition of smile and mouth motion (Black, Yacoob, Jepson, and Fleet [13]), and of lip-reading (Mase and Pentland [67]). Black, et. al. [13] assigned thresholds for motion parameters of linear combination in PCA for their classification paradigm. Mase, et. al. [67] considered the averaged flow direction at each of four rectangular feature regions around the mouth for lip-reading recognition. Both of these constraints introduced some degree of insensitivity to the extracted motion information and thus limited the recognition ability and accuracy

Mase and Pentland's lip-reading approach [67] uses a template matching that minimizes the sum of squared differences (SSD) between the projected flow curve of the testing word and that of the word templates in the two-dimensional eigenspace. The work of Black, Yacoob, Jepson, and Fleet on smile and mouth motion recognition [13] uses a similar approach by comparing similarities of the parameters of linear combination in PCA between the training and testing image sequences. In both cases, time warping is an essential preprocessing for comparison purpose. This becomes impractical when the lengths of image sequences are arbitrary (say, from 9 to 47 frames) and the projected flow curves are in a higher dimensional eigenspace.

Kobayashi and Hara [55,56,57] used three sets of artificial neural networks to recognize six basic facial expressions, mixed facial expressions (combinations of 2 or 3 basic

components), and the intensity of each facial expression, respectively. Inputs to these neural networks are the movements of sixty facial characteristic points which are manually selected. The recognition rate for six basic facial expressions was 88.7% from 15 subjects, and 70% for recognizing mixed facial expressions from 10 subjects.

Other studies in Japan [33,44,80,85,89,94] have used approaches similar to that of Kobayashi and Hara based on the displacement of manually selected facial characteristic points. Ding, Shimamura, Kobayashi, and Nakamura [33] used three sets of artificial neural networks to recognize brows, eyes and mouth expressions. They assumed symmetrical facial expressions, so they performed recognition only on the left half of the face. Others used fuzzy logic (Hashiyama, Furuhashi, Uchikawa and Kato [44], Ralescu and Hartani [80], and Ushida, Takagi and Yamaguchi [94]) or chaos (Sato and Yamaguchi [85]) combined with artificial neural networks to recognize six basic facial expressions.

Bartlett, Viola, Sejnowski, Golomb, Larsen, Hager, and Ekman [4] used three methods (PCA of difference images, optical flow with correlation coefficients, and high gradient component, *i.e.*, wrinkle, detection) to extract information on upper facial expressions (six upper face FACS AUs: AU1, 2, 4, 5, 6 and 7), and employed artificial neural networks for recognition. To deal with the time warping problem, they proposed to manually pick up six frames from each image sequence to form a new sequence for further processing: neutral expression for the first frame, low magnitude expressions for the second frame, medium magnitude expressions for the third and forth frames, and high magnitude expressions for the last two frames. Considering the relative geometric correspondence of face images, they took care of only rotation and horizontal scaling based on the location of both eyes, which was insufficient for aligning face images accurately because the vertical scaling was missing and the sizes of faces could be very different among subjects. The information they used for the PCA is the differences in images obtained by subtracting the gray values of the neutral expression (first frame) from those of the subsequent images for each image sequence. Such a simple subtraction process is not adequate to take care of the differences among individual faces. For high gradient component detection, they did

not discriminate that some wrinkles may be produced by facial expressions while others may be a permanent characteristic of the individual's face. Also, they proposed to wrinkles along several lines where some subjects may and other subjects may not appear wrinkled with the same expression. The best recognition rate was 91% from 20 expert subjects with 80 image sequences and 400 images.

Other methods of recognition have been applied to face or facial expression analysis. Beymer [9] proposed a method to normalize face images across different subjects, he analyzed and synthesized face images by interleaving shape and texture computations using optical flow and PCA in gray-value base [10]. Bregler and Konig [15] employed PCA and Hidden Markov Model (HMM) for speech recognition (eigenlips) and other applications. Kanade [49], one of the pioneers in face identification, used geometrical features of the face, such as the length of facial features, distance between features, and chin shape, to identify a face. Samaria and Young [84] converted each two-dimensional static and mono-shot face image into a concatenated one-dimensional gray-value vector for use in a continuous HMM to identify a face. These are indirectly related to our study but provide valuable references to this research.

1.2 Problem Statement

As reviewed in the previous section, most research in facial expression recognition is limited to six basic expressions and several combinations [12,13,36,68,82,103]. These stylized expressions are classified into emotion categories rather than facial actions. It is insufficient to describe all facial expressions because, in everyday life, six basic expressions occur relatively infrequently. Emotion is often communicated by small changes in one or two discrete features; on the other hand, the same facial expression may be involved in more than one emotion. The presence or absence of one or more facial actions may change its interpretation. For example, as shown in Figure 1, different smile expressions have an action unit AU12 (lip corners pulled obliquely) and emotional simile which may

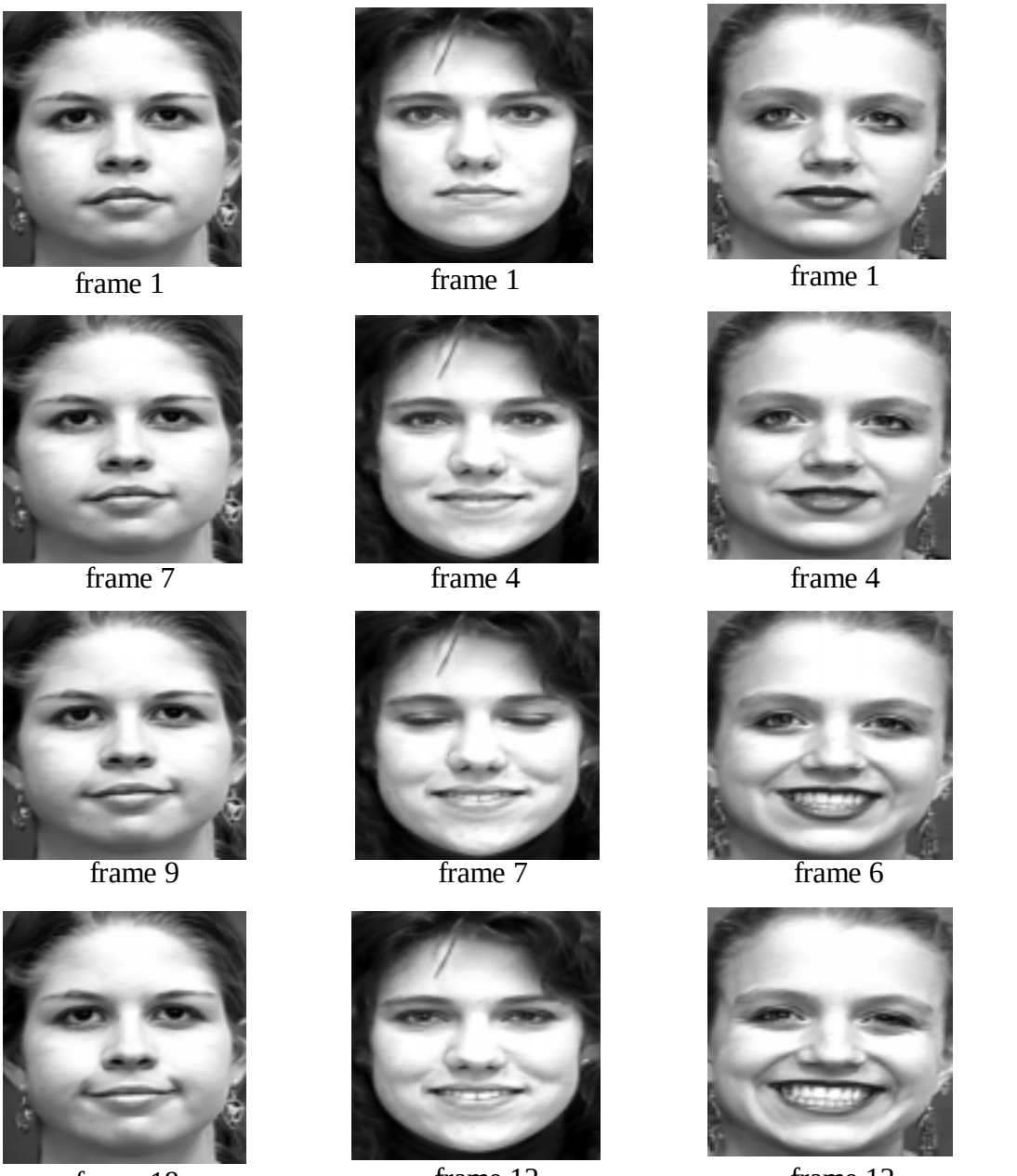


Figure 1 Comparison of different smile expressions with different expression intensities. The presence or absence of one or more facial actions can change their interpretations.

indicate an anxious or concealed emotion, a grin or a genuine (AU6+12+25: lip corners pulled obliquely for AU12, cheek raised for AU6, and subtly exposed for AU25, with or without crow-feet wrinkles near corners of the eyes). The degree of smiling is communicated by the intensity of raising the cheek and lip corners, and having the wrinkles. So it is important to be able not only to recognize basic expressions but also to discriminate subtly different expressions and estimate their expression intensities, which have a similar gross morphology but indicate varied meanings.

The Facial Action Coding System (FACS) [34] is so far the most comprehensive method of coding facial expressions, and provides a guideline for discrimination among closely related expressions. Manually encoding all action units (AUs) for various facial expressions is a laborious process. It takes approximately 100 hours to train a technician to have acceptable levels of coding experiences and up to 10 hours to code one-minute video tape of facial behavior [4,26]. Thus, it is desirable to automate the extraction and coding process, capable of delineating the temporal dynamics and intensity of facial expressions. Feature points are to be tracked in pixel base instead of averaging flow directions in a feature region. Optical flow in a larger region is to be efficiently represented for discriminating expressions. Image sequences ought to be preprocessed to separate the non-rigid motion of facial expression from any rigid head motion as much as possible and to geometrically normalize (align) corresponding face images in a sequence to ensure the assessment of correct motion information.

As the facial motion information are encoded into symbol sequences, it is natural to consider an HMM for automatic recognition of facial expressions where the maximum likelihood decision is assigned to an observable expression symbol sequence. HMM is capable of taking care the problem of variable expression length. HMM topology is referred to a particular network of states and state transitions. The best model is one with as few parameters as possible that can capture the behavior of the training data set. There exists no unified method to determine the optimum topology for an HMM. It will be a

great challenge to develop a strategy to do so for the underlying facial expression recognition problem.

1.3 Objective of the Research

The objective of this dissertation research is to develop a computer vision system, including both facial feature extraction and facial expression recognition based on FACS AUs, that is capable of automatically discriminating among subtly different facial expressions.

For facial feature extraction, we will consider three approaches in parallel. We will apply a pixel-based feature point tracking method, based on the coarse-to-fine pyramid approach, so as to make it sensitive to subtle feature motion as well as to handle large displacements; it will produce facial expression descriptions corresponding to each individual AU or AU combinations. We will also develop a method by using the dense flow to track motion vectors over a large facial region and applying the principle component analysis for data compression yet yielding the entire facial motion information. In addition, we will extract and analyze the motion of high gradient components (furrows) in the spatio-temporal domain to exploit their transient variances associated with facial expression.

Upon extraction of the facial expression information, each motion vector sequence will be vector quantized to a symbol sequence to provide an input to the facial expression classifier. An HMM-based classifier will be designed to deal with varies of facial expressions which are to be recognized in the context of motion sequences of variable length. Furthermore, different methods based on different types of the extracted expression information will be developed for expression intensity estimation which will be useful to segment facial expression sequences, measure the meaning of the expression, and analyze and synthesize facial expression for MPEG-4 applications in teleconferencing.

1.4 Organization of the Dissertation

The dissertation is organized into nine chapters. Chapter 1 gives the motivation of this research and reviews briefly the related works on facial expression recognition systems. The objectives of this dissertation are discussed. Chapter 2 introduces the framework of our computer vision system for facial expression recognition where FACS is used as a basis and where feature tracking is contemplated. Under certain limitations, the rigid head motion is removed from non-rigid facial expressions, and a geometric normalization is prescribed to ensure that optical flows or gray values of face images have the close geometric correspondence.

Chapters 3 through 5 present three methods to extract detail information of facial expressions and to give expression intensity estimation. Chapter 3 describes the pixel-based facial feature point tracking method using the pyramid approach for extracting subtle as well as large movements of facial features in subpixel accuracy. The critical role of the window function in motion estimation is analyzed. The motion is vector quantized into a symbol sequence representing the facial expression. Chapter 4 employs the wavelet-based motion estimation technique for dense flow tracking in order to include information of the entire range of facial motion. Flow-based principal component analysis is presented to compress the high-dimensional dense flows to a low-dimensional weight vector for each frame in a video sequence, which is encoded for recognition processes and expression intensity estimation. Chapter 5 presents a technique for high gradient component (*i.e.*, furrows) analysis. Motion line and edge detectors are designed to extract high gradient components in the spatio-temporal domain and to distinguish furrows from the noise. The high gradient components are encoded to mean and variance vectors as inputs to the recognition process.

Chapters 6 and 7 present the HMM for facial expression recognition. Chapter 6 analyzes the HMM technique and its associated computational issues. Chapter 7 presents a method of determining a special HMM topology for applications to facial expression recognition.

Chapter 8 describes our experimental results. A large database has been tested, subjects ranged in gender, age and ethnicity. We analyzed the performances of the recognition system using three feature tracking methods and demonstrated its high accuracy in comparison to the ground truth by human observation.

Chapter 9 discusses our major contributions and suggestions for further research.

2.0 FACIAL EXPRESSION RECOGNITION SYSTEM OVERVIEW

Humans are capable of producing thousands of facial actions during communication that vary in complexity, intensity, and meaning. Emotion or intention is often communicated by subtle changes in one or several discrete features. The addition or absence of one or more facial actions may alter its interpretation. In addition, some facial expressions may have a similar gross morphology but indicate varied meaning for different expression intensities. In order to capture the subtlety of facial expression in nonverbal communication, we propose to develop a computer vision system with a user interface (Figure 2) that automatically extract features and their motion information, discriminate subtly different facial expressions, and estimate expression intensity. The system contains two components: extraction and recognition as shown in Figure 3. Three methods are developed for feature and motion extraction yielding symbol sequences to represent observed expressions. These symbol sequences are input to the recognition process, which is an HMM computation to give the maximum likelihood decision.

2.1 Three Methods of Feature Motion Extraction

Facial expression is produced by the activation of facial muscles, which are triggered by the nerve impulses. Facial muscle actions cause the movement and deformations of facial skin and facial features. In the interpretation of facial expression, it is these deformations which we observe, and from which we must deduce the underlying emotion. Three convergent approaches are used to extract expression information (Figure 3): (1) facial feature point tracking using the pyramid method, (2) dense flow tracking with principal component analysis (PCA), and (3) high gradient component analysis in the spatio-temporal domain. In order to allow recognition, this extracted expression

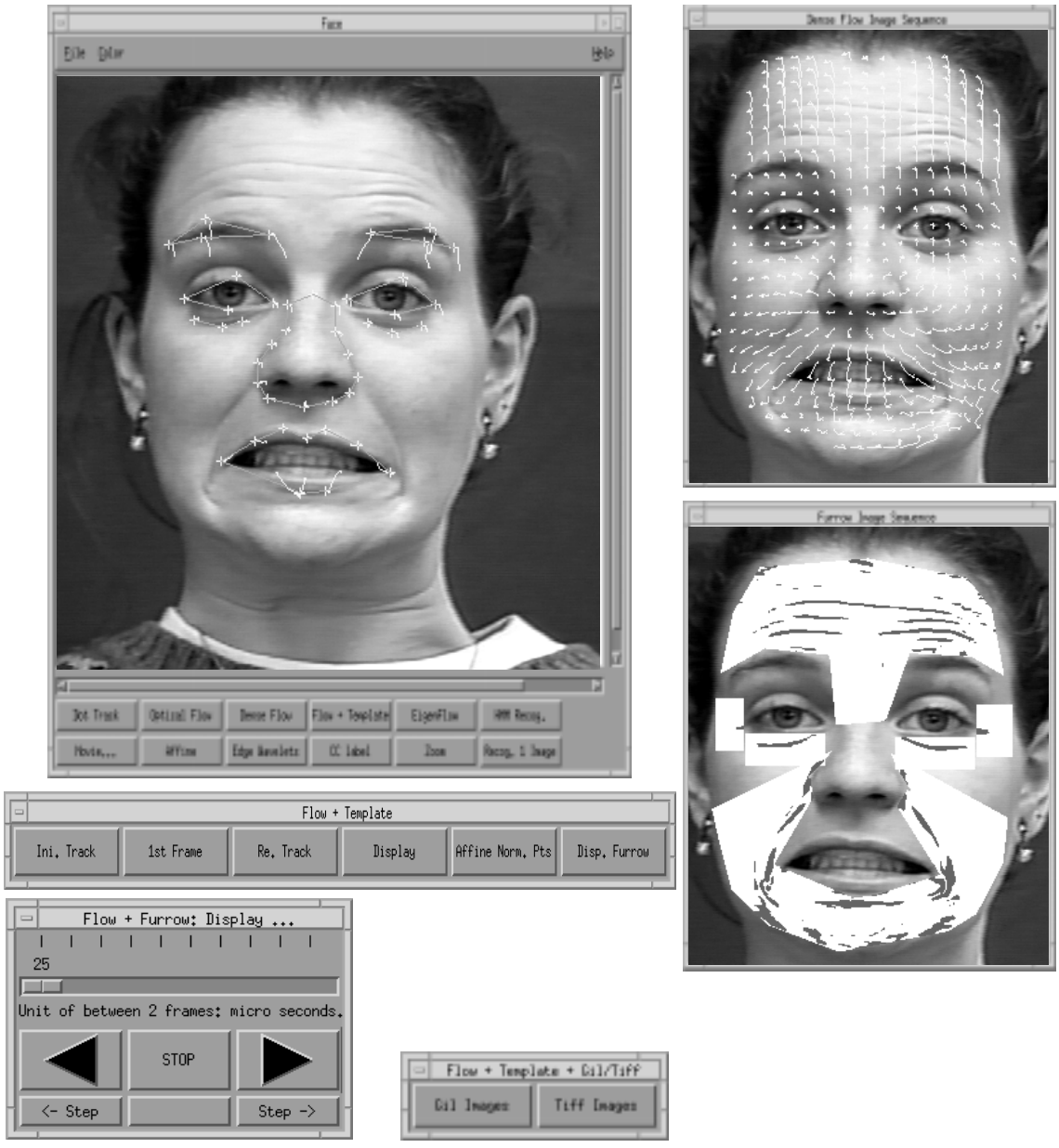


Figure 2 The user interface created by programming in C, Motif, X Toolkit and Xlib.

information must be converted into motion vectors so they may be passed to the recognition process (Figure 3).

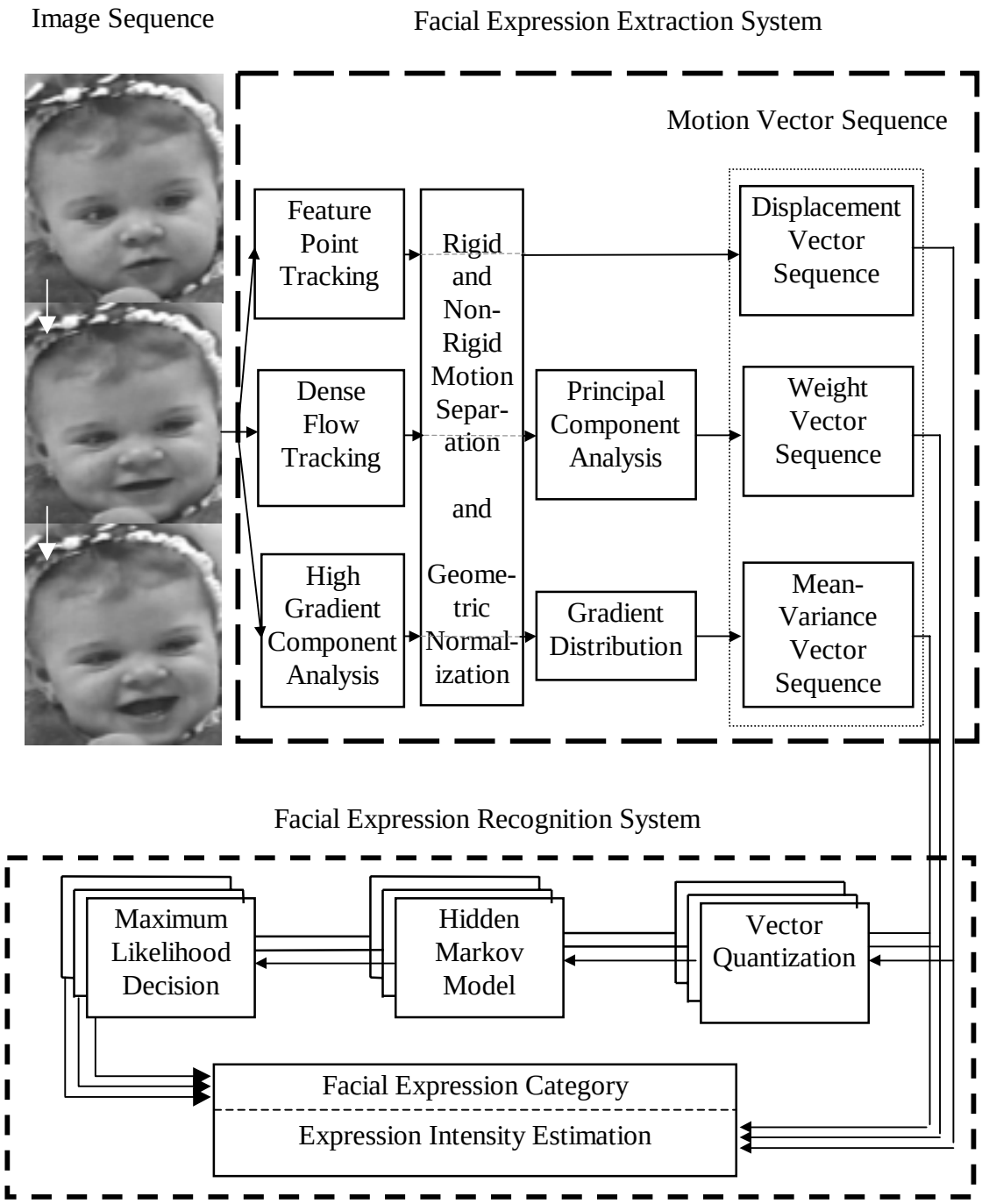


Figure 3 Block diagram of a facial expression recognition system.

Feature point tracking and dense flow tracking are used to track facial motion for recognition of expressions varying in intensity in the spatio-temporal domain. Frontal views of subjects (none wears eyeglasses) are videotaped under constant illumination, although lighting may vary across subjects particularly when we videotape on different days. These constraints are imposed to prevent significant degradation in optical flow calculation.

Facial feature point tracking using the pyramid method is especially sensitive to subtle feature motion and is also able to track a large displacement of feature motion in subpixel accuracy. Facial feature point is based on facial features in regions of brows, eyes, nose, and mouth. However, the forehead, cheek and chin regions also have important expression information. Dense flow tracking is used to include motion information from the entire face. The use of optical flow to track motion is advantageous because facial features and skin naturally have a great deal of texture. Using the principal component analysis, a low-dimensional weight vector in eigenspace can be obtained to represent the high-dimensional dense flows of each frame. Based on the displacement and weight vectors, the motion information is converted to symbol sequences from which we can recognize facial expressions, and is applied to estimate the expression intensity.

High gradient component analysis is also used to recognize expressions by the presence of furrows. Facial motion produces transient wrinkles and furrows perpendicular to the motion direction of the activated muscles. The facial motion associated with a furrow produces gray value change in the face image, which can be extracted by the use of high gradient component (motion line or edge) detectors in the spatio-temporal domain.

2.2 Recognition Using Hidden Markov Models

Modeling facial expression needs to take into account the stochastic nature of human facial expression involving both the human mental state, which is hidden or immeasurable, and the human action, which is observable or measurable. For example, different people

with the same emotion may exhibit very different facial actions, expression intensities and durations. Individual variations notwithstanding, a human observer can still recognize what emotion is being expressed, indicating that some common element underlies each motion. Therefore, the purpose of facial expression modeling is to uncover the hidden patterns associated with specific expressions from the measured (observable) data. Facial expression modeling requires a criterion for measuring a specific expression. It is desirable to analyze a sequence of images to capture the dynamics [5]. Expressions are recognized in the context of an entire image sequence of arbitrary length. We will develop a recognition system based on the stochastic modeling of the encoded time series describing facial expressions, which should perform well in the spatio-temporal domain, analogous to the human performance.

In order to model subtly different facial expressions having different durations (arbitrary length of image sequence), the Hidden Markov Model (HMM) is developed to recognize expressions based on the maximum likelihood decision criterion. A key problem is to determine the HMM topology for the facial expressions under consideration. Some other advantages of using HMMs are: HMM computations converge quickly making it practical for real time processing, it may evaluate an input sequence of uncertain category to present a low output probability, and a multi-dimensional HMM may be developed to integrate individual HMMs to give a robust and reliable recognition. The correspondence between facial expressions and elements of the HMM is shown in Table 1.

Facial expression and speech represent human visual and audio actions, respectively [88]. The HMM technique has been successfully applied to model all known phonemes (the basic units of speech). Elementary HMMs of phonemes have then combined to represent words, and then sentences [59,76,79]. Speech may be considered as two- or three-dimensional signals: frequency and amplitude change with time. Facial expressions may be considered as three (or four)-dimensional signals: a time sequence of images. So a set of elementary HMMs will be developed to model various “expression units” of individual

Table 1 Correspondence between facial expressions and elements of the Hidden Markov Model.

	Facial Expression	**Hidden Markov Model**
Hidden Process	Mental State	Model State
Observable	Expression (Facial Action)	Symbol Sequence
Temporal Domain	Dynamic Behavior	A Network of State Transition
Characteristics	Expression	State Transition Probability and Symbol Probability
Recognition	Expression Similarity	The Confidence of Output Probability

AUs or AU combinations, such as illustrated in Figure 4. Based on combinations of elementary HMMs, we will be able to recognize continuously varying facial expressions. A comparison of modeling facial expressions with modeling speech using HMMs is listed in Table 2.

2.3 Facial Action Coding System and "Expression Units"

The proposed automatic of facial expression analysis follows the anatomically based Facial Action Coding System (FACS) [(34)], which is the most comprehensive method for coding facial expressions by psychologists. With FACS, observers can manually code discrete deformations of the face (movements of the facial muscle and skin) which are referred to as action units (AUs). Basically, FACS divides the face into upper and lower facial expressions and subdivides motion AUs. FACS consists of 44 basic AUs, with 14 additional AUs for head and eye positions as shown in Table 3. AUs are the smallest visibly discriminable muscle actions that individuate or combine to produce characteristic facial expressions which can be recognized from the image. More than 7000

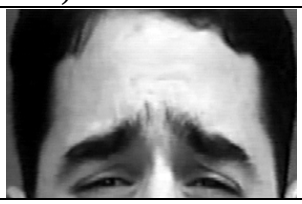
Upper Facial Expressions		
AU4: Brows are lowered and drawn together.	AU1+4: Medial portion of the eyebrows is raised (AU1) and pulled together (AU4).	AU1+2: Inner (AU1) and outer (AU2) portions of the brows are raised.
		
Lower Facial Expressions		
AU12: Lip corners are pulled up and backward.	AU6+12+25: Cheek raised (the lower-eye and infra-orbital furrows are raised and deepened, and the eye opening is narrowed) (AU6), and AU12 with mouth opening (AU25).	AU20+25: Lips are parted (AU25), pulled back laterally, and may be slightly raised or pulled down (AU20).
		
AU9+17: The infra-orbital triangle and center of the upper lip are pulled upwards (AU9), and the chin boss and lower lip are pulled upwards (AU17).	AU17+23+24: The chin boss is raised, which pushes up the lower lip (AU17); the lips are tightened, narrowed (AU23), and pressed together (AU24).	AU15+17: Lip corners are pulled down and stretched laterally (AU15), and chin boss is raised which pushes up the lower lip (AU17).
		

Figure 4 "Expression units" of subtly different facial expressions in our study (taken from [34]).

Table 2 Comparison of modeling facial expressions with modeling speech using HMMs.

	Speech	**Facial Expressions**
Human Action	Audio Action	Visual Action
Dimension (Including Time Series)	2-dimensional Signals	3 or 4-dimensional Signals
Action Unit	Phoneme	Expression Unit: Individual AUs or AU Combinations
HMM Unit	1st-order 3-state HMM	2nd-order 3-state HMM for Upper Facial Expression and 3rd-order 4-state HMM for Lower Facial Expression *.
HMM Unit Combinations	One Word	One Basic Facial Expression (*e.g.*, joy)
Concatenated HMM Unit Combinations	Sentences	Continuously Varying Basic Facial Expressions

* Obtained in this research.

have been observed. According to FACS, each AU corresponds to an activity in a distinct muscle, with the exception of AU4 [34,107]. Even though the one-to-one mapping of individual AUs to distinct muscle activities is a basic assumption of the FACS, AUs enable discrimination between closely related expressions. By discriminating "expression units (individual AUs or AU combinations)", we can simulate and understand individual mechanics of the facial muscles. In the present study we consider, three upper facial "expression units" and six lower facial "expression units" which are shown in Figure 4. They are frequently occurring facial expressions containing subtle differences. They will be studied for automatic recognition and estimation of their intensities.

Table 3 Action Units (AUs) in the Facial Action Coding System (FACS) [34].

Upper Face

AU	Label
1	Inner Brow Raise
2	Outer Brow Raise
4	Brow Lower
5	Upper Lid Raise
6	Cheek Raise
7	Lids Tight
41	Lids Droop
42	Lids Slit
43	Lids Closed
44	Squint
45	Blink
46	Wink

Lower Face

AU	Label
9	Nose Wrinkle
10	Upper Lip Raise
11	Nasolabial Deepen
12	Lip Corner Pull
13	Sharp Lip Pull
14	Dimple
15	Lip Corner Depress
16	Lower Lip Depress
17	Chin Raise
18	Lip Pucker
20	Lip Stretch
22	Lip Funnel
23	Lip Tight
24	Lip Press
25	Lips Part
26	Jaw Drop
27	Mouth Stretch
28	Lip Suck

Miscellaneous

AU	Label
8	Lips Toward
19	Tongue Show
21	Neck Tighten
29	Jaw Thrust
30	Jaw Sideways
31	Jaw Clench
32	Bite (Lip)
33	Blow
34	Puff
35	Cheek Suck
36	Tongue Bulge
37	Lip Wipe
38	Nostril Dilate
39	Nostril Compress

Head Position

AU	Label
51	Turn Left
52	Turn Right
53	Head Up
54	Head Down
55	Tilt Left
56	Tilt Right
57	Forward
58	Back

Eye Position

AU	Label
61	Left
62	Right
63	Up
64	Down
65	Walleye
66	Cross-eye

2.4 Rigid and Non-Rigid Motion Separation and Geometric Normalization

For facial expression recognition, two main issues in image processing will affect the recognition results: separation of non-rigid facial expression from rigid head motion, and facial geometric correspondence to keep face size constant across subjects. Both processes are necessary in order to ensure that these variables do not interfere with expression recognition. Though all subjects are viewed frontally in our current research, some out-of-plane head motion (*e.g.*, yaw rotations or less than ±10 degree pitch rotations) may occur with facial expressions. Furthermore, face size varies among individuals. For elimination of the above-mentioned rigid head motion from non-rigid facial expression, an affine transformation (which includes translation, scaling and rotation factors) is adequate to normalize the face geometric position and maintain face magnification invariance. Face images are automatically normalized with the affine transformation to ensure that optical flows or gray values of individual frames have close geometric correspondence in order to achieve consistent recognition performance.

In the first frame of each image sequence, we manually select three facial feature points for image normalization: medial canthus of both eyes and the uppermost point on the philtrum as shown in Figure 5. These three points will carry only rigid motion components accompanied with the head motion. Each of these points forms the center of a 13 x 13 pixel flow window, and they are automatically tracked in the remaining frames of each image sequence. Based on these three facial feature points, the original 490 x 640 (row x column) pixel display is cropped to 417 x 385 pixels for each frame to keep the foreground face and remove the unnecessary background. The positions of all tracking facial feature points, dense flows, or image gray values for each frame are then normalized by warping them onto a standard two-dimensional face model based on the affine transformation $\mathfrak{I}$ (Figure 5) given as follows:

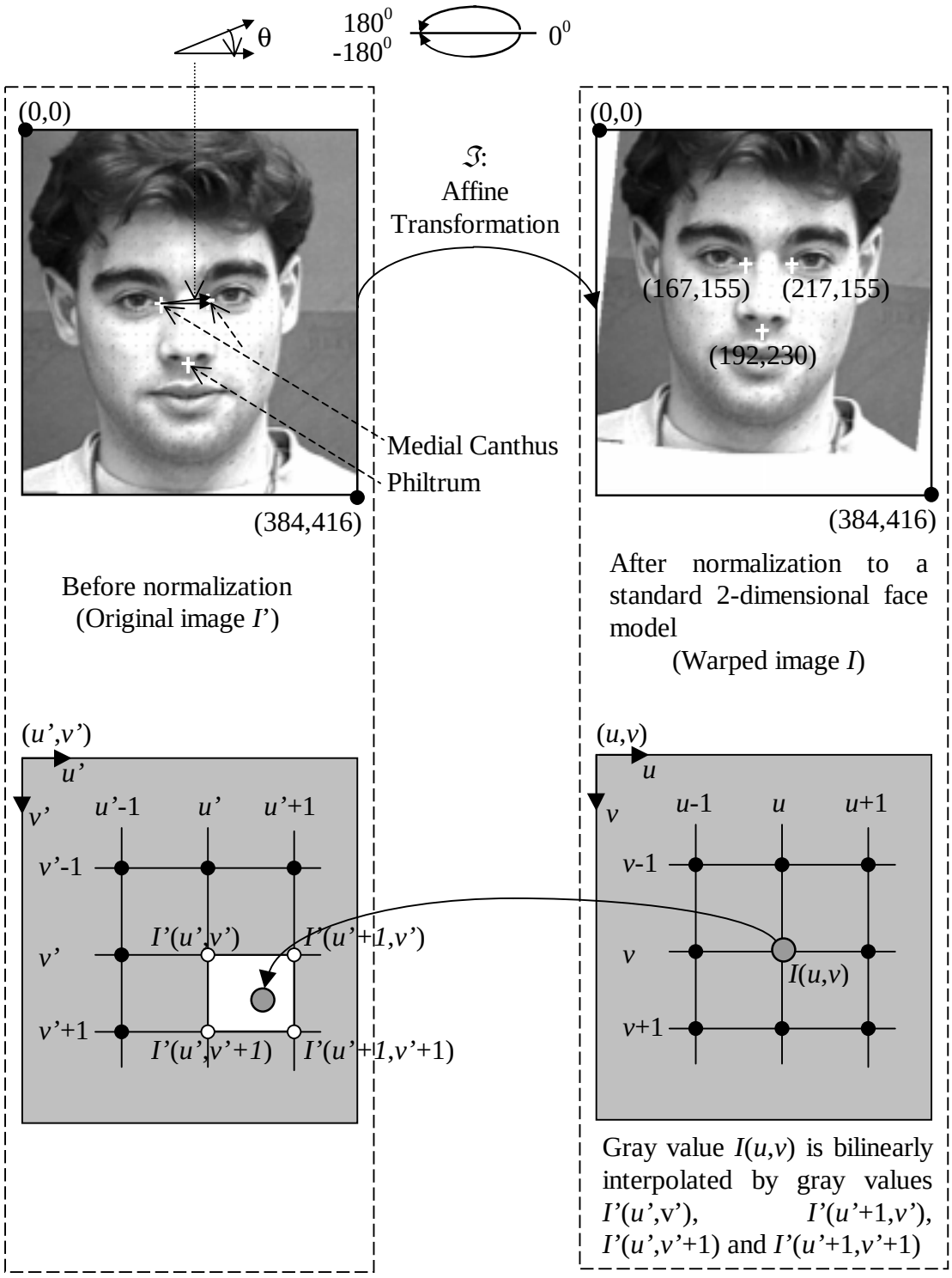


Figure 5 Normalization of each face image to a standard 2-dimensional face model.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} S_u & 0 \\ 0 & S_v \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix} + \begin{bmatrix} D_u \\ D_v \end{bmatrix} \quad (2\text{-}1)$$

where

$$S = \begin{bmatrix} S_u & S_v \end{bmatrix} = \begin{bmatrix} \frac{w}{w'} & \frac{h}{h'} \end{bmatrix} \quad (2\text{-}2)$$

$$D = \begin{bmatrix} D_u & D_v \end{bmatrix} = \begin{bmatrix} d_u - d'_u & d_v - d'_v \end{bmatrix} \quad (2\text{-}3)$$

Here, u and v are the horizontal and vertical positions of the two-dimensional face model coordinates, and u' and v' are the horizontal and vertical positions of the original image coordinates. The upper-left corner of each frame, including the face model image is denoted as (0,0). In the standard face model, the top point of the philtrum is the rotation center whose position is (d_u,d_v) = (192,230), and the position of the medial canthus of the right eye is (167,155) and that of the left eye is (217,155); the width w between the medial canthi of both eyes is 50 pixels and the height h from the level of the medial canthi to the top point of the philtrum is 75 pixels. The horizontal scaling is given by the parameter S_u which is computed as the ratio of the distance w at the face model to that distance w' at the original face image. The vertical scaling given by S_v is computed as the ratio of the distance h at the face model to that distance h' at the original face image. The horizontal and vertical displacements (translations) are represented by D_u and D_v, respectively, and are measured from the top point of the philtrum in the original face image (d'_u,d'_v) to that in the face model (d_u,d_v). The angle of rotation of the line connecting the medial canthi of both eyes in the original face image from the corresponding horizontal line in the face model is represented by θ, where the clockwise rotation is negative and the counterclockwise rotation is positive. The pixel positions of each image are integer-valued, but the warped positions after the affine transformation are, in general, not integer-valued. So the gray value at each integer-valued pixel of the warped image needs to be estimated by bilinear interpolation based on the gray values of its four nearest neighbor pixels in the original image as shown in Figure 5.

3.0 FACIAL FEATURE POINT TRACKING

The face is an interface of nonverbal communication, which can represent a subject's social feeling or reveal his brain function through the use of expressions. People activates facial action mainly by controlling the individual or combined motions of four facial features: brows, eyes, nose and mouth. These are the most attractive features on the facial surface because they have high textures, and symbolize the underlying muscle activations. An observer may recognize easily and directly the messages transmitted from the movement of facial features. Optical flow in an image sequence has been used to track highly textured regions reliably for extracting the motion information of facial features to be used in further recognition process. Optical flow provides an estimate of the movement of facial feature points. Since our goal is to discriminate subtly different facial expressions and to estimate the expression intensity, the tracking algorithm must have high accuracy, be sensitive to subpixel motion, and be able to deal with relatively large facial movements.

3.1 Dot Tracking and Reliability of Feature Point Selection

Since FACS specifies that each AU corresponds to the movement of a single muscle, we design an automatic feature point tracking to extract motion information of facial feature actions based on the movement of facial feature points (which represents the underlying muscle activations) across an image sequence. This will allow realization of the AU actions of facial expressions for encoding "expression units" constructed by individual AUs or AU combinations.

It is important to make sure that the locations and movement of feature points can exactly reflect the AU activation. The following method is used to track the facial features. We attach black dots to specific points on faces of the subjects; a black dots

have the same radius equivalent to 15 pixels on a face image of the size of 490 x 640. We use the template matching (the correlation coefficient ρ) method to track dot movements during the facial action by considering the highest value of the correlation coefficient between the dot template F and the gray values in a target region $I(x)$ centered at position x,

$$\rho(x) = \left| \frac{\sum_{r\in R}[F(r)-\overline{F}][I(x+r)-\overline{I}(x)]}{\sqrt{\sum_{r\in R}[F(r)-\overline{F}]^2}\sqrt{\sum_{r\in R}[I(x+r)-\overline{I}(x)]^2}} \right| \qquad 0.0 \le \rho \le 1.0 \qquad (3\text{-}1)$$

where r denotes a position within the circular region of the dot template whose area is R (radius = 15 pixels), $I(x+r)$ denotes the gray value of the image at position $x+r$, $\overline{F}$ and $\overline{I}(x)$ are the average gray values in circle regions of the dot template and the image, respectively, and x is in a search region m of 60 x 60 pixels, which is large enough to reach the maximum dot movement but not too large to reach any neighboring dot template. If the correlation coefficient ρ is closer to 1.0, there is very strong similarity between the dot template and the target region; otherwise, they are less correlated. Utilizing the relations

$$\sum_{r\in R} F(r) = R\overline{F} \qquad (3\text{-}2)$$

and

$$\sum_{r\in R} I(x+r) = R\overline{I}(x) \qquad (3\text{-}3)$$

the above equation is simplified to achieve the efficient computation,

$$\rho(x) = \left| \frac{\sum_{r\in R}F(r)I(x+r) - \overline{I}(x)\sum_{r\in R}F(r) - \overline{F}\sum_{r\in R}I(x+r) + \sum_{r\in R}\overline{F}\overline{I}(x)}{\sqrt{\sum_{r\in R}[F(r)-\overline{F}]^2}\sqrt{\sum_{r\in R}I^2(x+r) - 2\overline{I}(x)\sum_{r\in R}I(x+r) + \sum_{r\in R}\overline{I}^2(x)}} \right|$$

$$= \left| \frac{\sum_{r\in R}F(r)I(x+r) - R\overline{F}\overline{I}(x)}{\sqrt{\sum_{r\in R}[F(r)-\overline{F}]^2}\sqrt{\sum_{r\in R}I^2(x+r) - R\overline{I}(x)\overline{I}(x)}} \right| \qquad (3\text{-}4)$$

The first term in the denominator and the average gray value of dot template $\overline{F}$ can be calculated first and assigned as constant values for quick processing.

For the first frame of each image sequence, we used the computer user interface which we designed to initially locate the center of each dot marked by a cross (+) on the screen. By observing or automatically tracking the motion of the dots in the remaining frames, we realized the AU actions of facial expressions and understood the underlying muscle activations (Figure 6). This allowed us to correctly locate controlled positions of facial feature points and measure the reliability of the selection of feature points based on which the automatic tracking using the optical flow will begin.

Using the template matching method to track dots is inaccurate for two reasons: the dots on screen may become deformed and are sometimes affected by the reflections due to lighting (especially if red or white dot templates are used) on subjects with dark skin color. The method is also much slower when compared to selected optical flow tracking discussed before, especially when the number of dots and search regions are increased.

For the optical flow tracking, a set of feature points will be initially selected on the first frame of each image sequence (from the neutral expression to the peak expression in an arbitrary length of time). We suggest to select 4 facial feature points around the contour of each brow, 4 points around each eye, 14 points around the nose contour, 10 points surround the lip contour, and 3 points along each cheek bone (below each lower eyelid) as shown in Figure 7. This can be done by an operator using a computer, mouse, and user interface as shown in Figure 2. The manually selected feature points are then automatically tracked using optical flow in the remainder of the sequence. The motion of these facial feature points simulates the facial muscular actions corresponding to AUs. The displacements of these feature points are directly proportional to the AU expression intensities.

The reliability of the feature point selection has been confirmed in experiments by two experienced operators. The first frames of 80 image sequences are independently selected by two different operators. The optical flow method was used to automatically track those selected feature points and inter-observer reliability was evaluated. Our

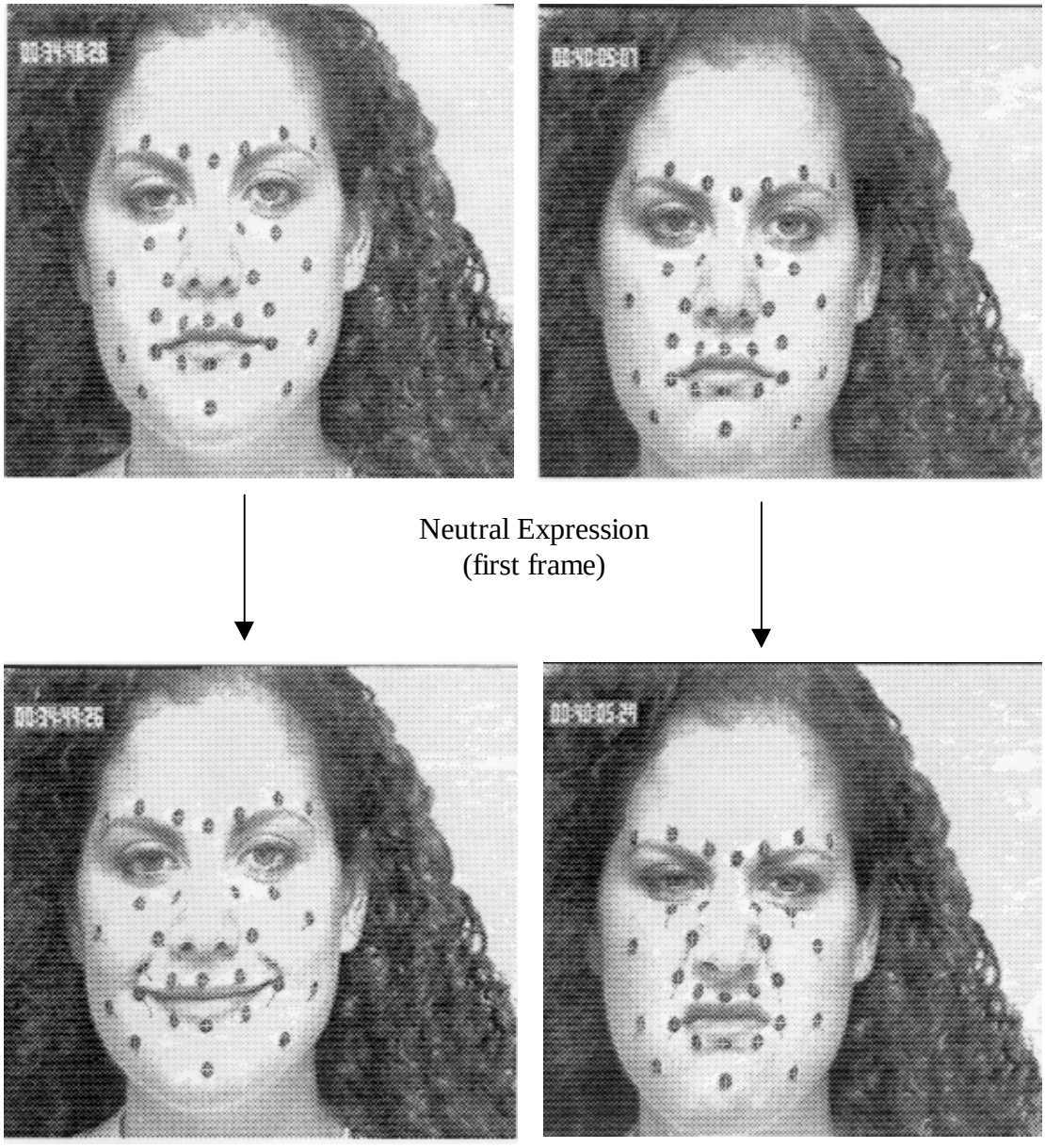


Figure 6 Dot tracking: each dot is marked by a cross (+) at its center, lines trailing from the dots represent changes in the location of dots due to facial expression.

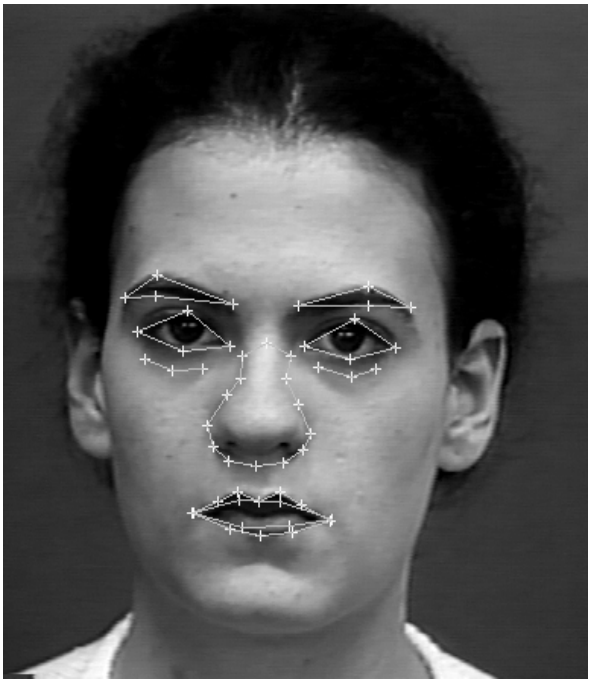


Figure 7 Locations of selected facial feature points (marked by a cross '+') which reflect the muscle motion of facial features.

results showed a very high correlation and the identical recognition performance between two operators [28]. This selection process can be easily taught to new operators. An operator can learn this after a training of about 5 minutes session, which is substantially shorter than learning the FACS (which may require 100 hours).

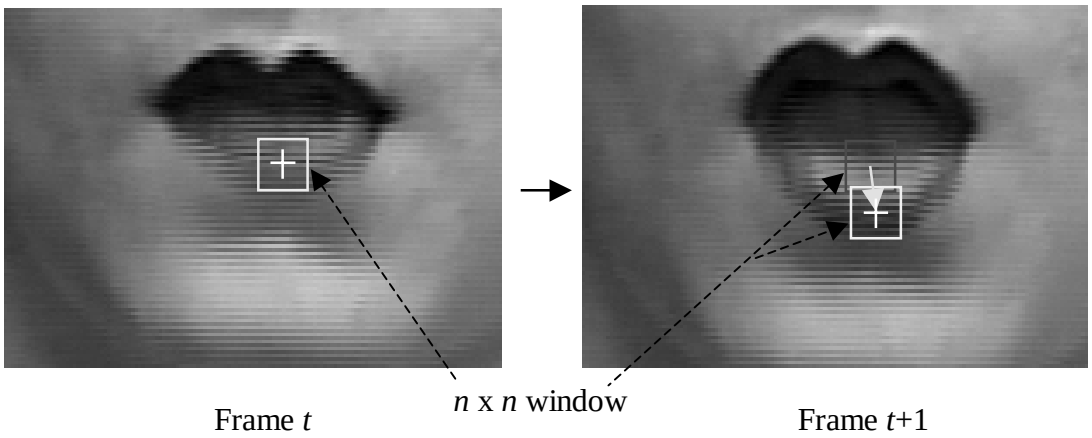


Figure 8 Feature point tracking based on tracking the movement of an *n* x *n* feature window between two consecutive frames (here, *n* is 13 pixels).

3.2 Motion Estimation and Flow Window

The motion estimation method used here is based on the optical flow algorithm developed by Lucas and Kanade [66], and implemented by including the pyramid approach used by Poelman and Kanade [77] to track large motion. This method assumes that the gray values in any image feature region (*n* x *n* feature window) do not change between two consecutive frames, but only shift from one position to another (Figure 8). We can track the motion of high gradient points, such as facial feature points, with subpixel accuracy by iterative computation. Its convergence is very fast.

Let us consider an *n* x *n* region *R* in the reference image at time *t*, where $I_t(x)$ denotes the gray value of the pixel position *x* in *R*. Let us find the best matching (registration) position of this region in the following frame at time *t*+1, where $I_{t+1}(x)$ denotes the gray value in the region, by minimizing a cost function *E* of the sum of squared differences (SSD) defined as

$$E(d(x)) = \sum_{x \in R} [I_t(x - d(x)) - I_{t+1}(x)]^2 w(x) \qquad (3\text{-}5)$$

where $d(x)$ is the displacement of x of region R between two consecutive frames and $w(x)$ is a window function for weighting the squared differences in E. This minimization for finding the motion vector $d(x)$ can be done in iterations. Let

$$d(x) = d^i(x) + \Delta d(x) \tag{3-6}$$

at the ith iteration and $\Delta d(x)$ is the incremental displacement at the ith iteration. We want to robustly estimate the incremental displacement $\Delta d(x)$ with a subpixel accuracy. Let us expand the term

$$I_t(x-d) = I_t(x-(d^i + \Delta d)) \tag{3-7}$$

by the first order Taylor' expansion:

$$I_t(x-d^i-\Delta d) \approx I_t(x-d^i) - I_t'(x-d^i)^T \Delta d \tag{3-8}$$

where $I'_t(x)$ denotes the gradient of the gray value $I_t(x)$. The incremental change in the SSD cost function is given by

$$\begin{aligned} E(\Delta d) &= E(d^i + \Delta d) - E(d^i) \\ &\approx \sum_{x\in R}[I_t(x-d^i) - I_t'(x-d^i)^T \Delta d - I_{t+1}(x)]^2 w(x) - \sum_{x\in R}[I_t(x-d^i) - I_{t+1}(x)]^2 w(x) \\ &= \sum_{x\in R}[I_t'(x-d^i)^T \Delta d]^2 w(x) - 2\sum_{x\in R}[I_t(x-d^i) - I_{t+1}(x)] I_t'(x-d^i)^T \Delta d w(x) \\ &= \Delta d^T G \Delta d - 2e^T \Delta d \end{aligned} \tag{3-9}$$

where

$$G = \sum_x I_t'(x-d^i) I_t'(x-d^i)^T w(x) \tag{3-10}$$

is the Hessian matrix of the gradients of I_t with a window function $w(x)$, and

$$e^T = \sum_x [I_t(x-d^i) - I_{t+1}(x)] I_t'(x-d^i)^T w(x) \tag{3-11}$$

is a difference-gradient row vector which is the product of the difference (or error) between the regions in the two consecutive images and the gradient of the gray-value I_t together with a window function $w(x)$. The maximum decrement $E(\Delta d)$ occurs when its gradient with respect to Δd is zero,

$$\frac{\partial E(\Delta d)}{\partial(\Delta d)} = G\Delta d + (\Delta d^T G)^T - 2e = 2(G\Delta d - e) = 0 \qquad (3\text{-}12)$$

Hence,

$$\Delta d(x) = G^{-1}e \qquad (3\text{-}13)$$

Initializing $d^{(0)}(x) = [0,0]^T$ and following equations (3-8), (3-10), (3-11) and (3-13), the optical flow $d(x)$ can be robustly estimated through iterations yielding the subpixel accuracy.

The motion estimate $d(x)$ is more accurate when the gradients of both $I_t(x)$ and $I_{t+1}(x)$ are large and nearly equal as illustrated in Figure 9 [66].

$$\begin{aligned} I_t'(x) - I_{t+1}'(x) &= I_{t+1}'(x+d) - I_{t+1}'(x) \\ &\approx [I_{t+1}'(x) + I_{t+1}''(x)d] - I_{t+1}'(x) \\ &= I_{t+1}''(x)^T d \end{aligned} \qquad (3\text{-}14)$$

and $I_{t+1}''(x)$ denotes the second derivatives. The first order Taylor's linear approximation is more likely to give an accurate estimate d when both the difference of the gradients I_t' and I_{t+1}', and the second derivatives of I_{t+1} are small [66]. Otherwise, it is prove to have a large error. Thus the window function $w(x)$ should be small when the difference of the I_t' and I_{t+1}' is large, and large when the difference is small (Figure 9) [66]. If the window function is unity, $w(x) = 1$, over the n x n region R such as used in Lucas and Kanade's flow estimation [66], the local minimum of SSD is considered which maintains the high frequency information in the region but may yield a noisy result. This optical flow can accurately track the highly textured local region and the computation converges very fast. It is good for use in the facial feature point tracking and for real time processing. But it will be less accurate when used to track a less textured region or a region with high reflection where there is no high gradient pixels for tracking, *i.e.*, the region is not trackable. In such a case, more global information instead of local minimization may be needed, such as using the regularization-based or global smoothness approach [11,47] to estimate the optical flow in textureless region, but it is more time consuming since a large

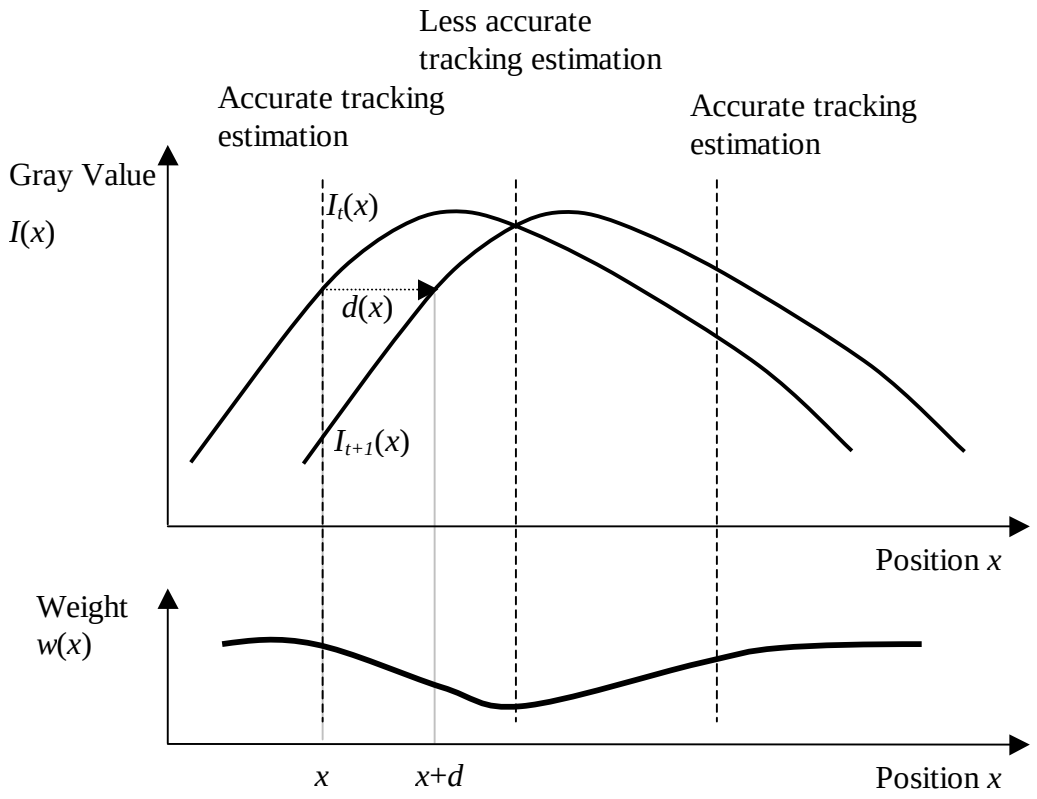
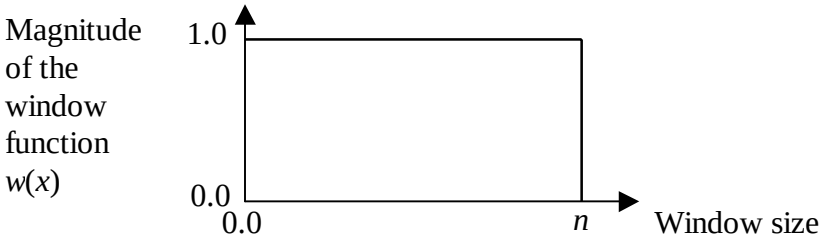
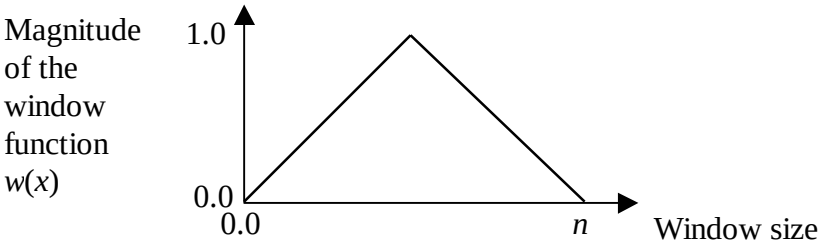


Figure 9 Window (weight) function $w(x)$ can be used to control the accuracy of motion estimation based on the gradient varying from point to point [66].

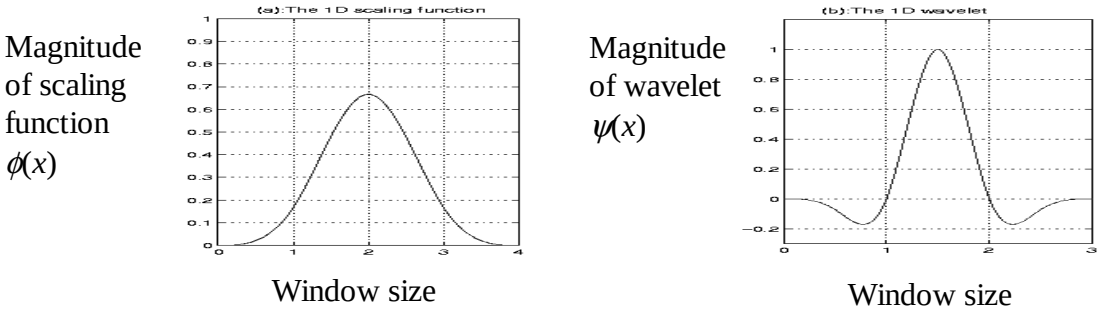
window is involved and it may also smooth out high frequency components and thus reduce the tracking accuracy. Two methods have been recently developed to overcome these local noise and global smoothness problems: the spline base method [90] and the wavelet base method [101], both use the pyramid approach and multiple flow windows (Figure 10).



Window function $w(x)$ used in the Lucas-Kanade's method [66].



Window function $w(x)$ used in Szeliski's spline-based method [90].



Low-pass scaling function $\phi(x)$ and high-pass wavelet $\psi(x)$ used in the wavelet-based method of Wu [101].

Figure 10 Comparison of Lucas-Kanade, spline-based and wavelet-based window functions.

3.3 Motion Confidence Estimation

Tomasi and Kanade [92]; and Poelman and Kanade [77] have shown that the eigenvalues of the Hessian matrix G can be used to estimate the confidence of whether the n x n feature region R is trackable or not. We illustrate it by four features regions shown in Figure 11 with window function $w(x) = 1$ in each region, thus

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{12} & G_{22} \end{bmatrix} = \begin{bmatrix} \sum_{x \in R} \left(\frac{\partial I_t}{\partial x_1} \right)^2 \Big|_{x - d^i} & \sum_{x \in R} \left(\frac{\partial I_t}{\partial x_1} \right) \left(\frac{\partial I_t}{\partial x_2} \right) \Big|_{x - d^i} \\ \sum_{x \in R} \left(\frac{\partial I_t}{\partial x_1} \right) \left(\frac{\partial I_t}{\partial x_2} \right) \Big|_{x - d^i} & \sum_{x \in R} \left(\frac{\partial I_t}{\partial x_2} \right)^2 \Big|_{x - d^i} \end{bmatrix} \tag{3-15}$$

$$e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} \sum_{x \in R} \left(\frac{\partial I_t}{\partial x_1} \right) \Big|_{x - d^i} [I_t(x - d^i) - I_{t+1}(x)] \\ \sum_{x \in R} \left(\frac{\partial I_t}{\partial x_2} \right) \Big|_{x - d^i} [I_t(x - d^i) - I_{t+1}(x)] \end{bmatrix} \tag{3-16}$$

In Figure 11.a, the feature region R is textureless or very smooth. This region will be difficult to track since it has zero gradient in all directions and, hence, both eigenvalues of the Hessian matrix G are equal to zero. If a feature region R contains a line or edge which has high spatial gradient values in at least one direction as shown in Figure 11.b or 11.c, or which has highly correlated spatial gradients in both horizontal and vertical directions as in Figure 11.c (the Hessian matrix G has one large eigenvalue and one small eigenvalue), then this region will be difficult to track. Only when the feature region R has high spatial gradients in two orthogonal directions (horizontal- and vertical-gradients are weakly correlated), and hence both eigenvalues of G are large, can this feature region be localized and easily tracked as shown in Figure 11.d. From the mathematical point of view, if the Hessian matrix G has one or more small eigenvalues, then computation of its inverse is an ill-conditioned problem because it is close to be singular.

To estimate the confidence of whether a selected n x n feature region R is trackable or not, we use the confidence value defined below.

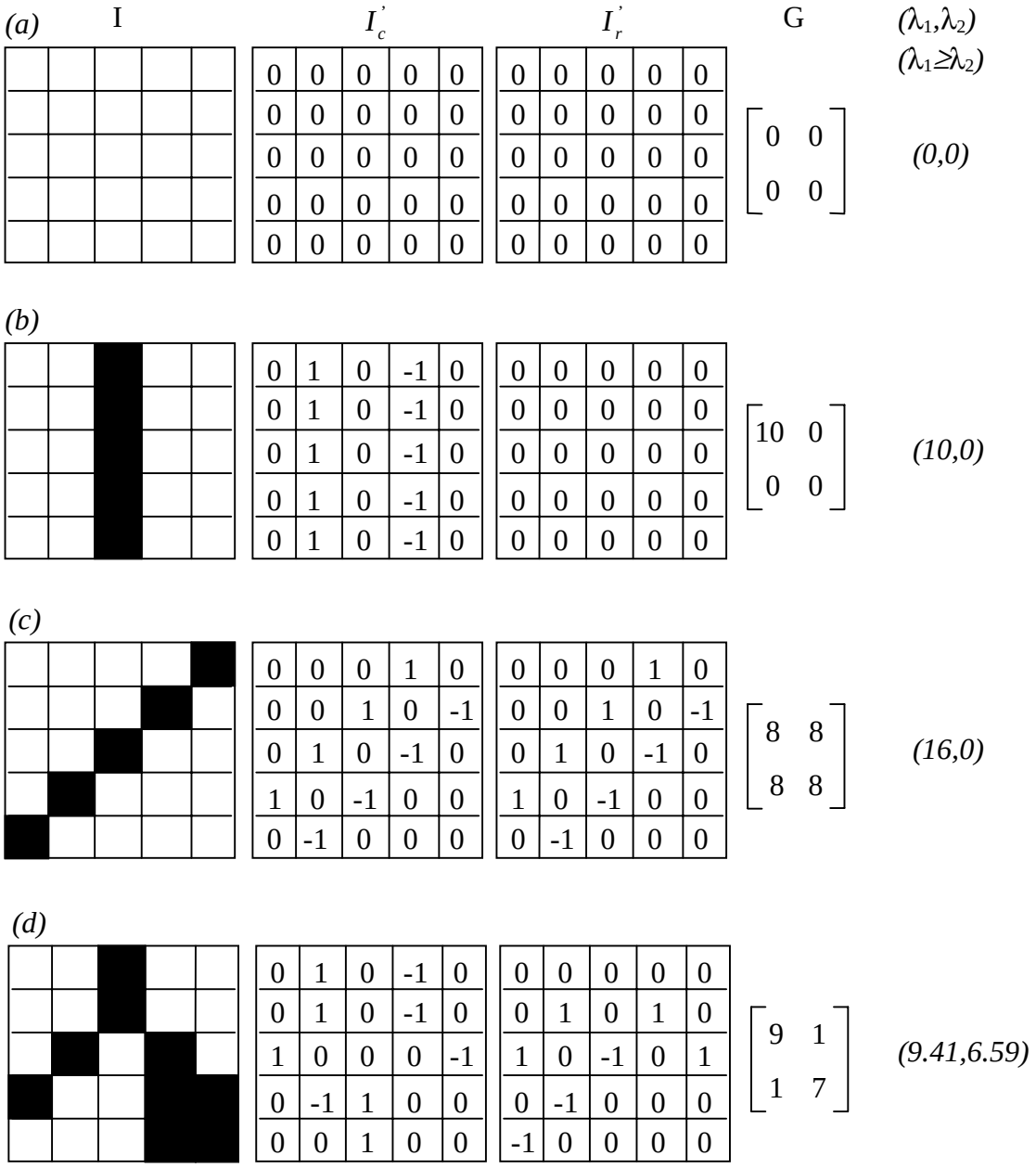


Figure 11 Trackability of various feature regions in a binary image: (a) contains no image texture so it would make a poor feature; (b) and (c) have high gradients locally either in one direction, or the horizontal- (I_c'-) and vertical- (I_r'-) gradients are highly correlated with each other as in (c), they also are not trackable; only feature (d) can be used as a trackable feature [77].

$$C = \frac{\lambda_{\min} * (n * n)}{1 + \sum_{x \in R} [I_t(x - d) - I_{t+1}(x)]^2} \tag{3-17}$$

Here, λ_{min} is the minimum eigenvalue of the Hessian matrix G. The constant 1 is used to avoid a zero value in the denominator in case of a perfect matching. The trackability is proportional to the confidence value, (or is proportional to λ_{min}), and inversely proportional to the difference (or residue) between two matching regions. A feature region with high λ_{min} contains high-frequency textured patterns and can be localized accurately even though there is noise in the image. A feature region with a very low value of λ_{min} contains smooth areas, so image noise is more likely to cause shifts along the lower gradient direction, such as along the line or edge in Figure 11.b and 11.c.

3.4 Tracking Subpixel and Large Motion

Since we want to discriminate subtly different facial expressions by extracting the movement of feature points, it is necessary to use optical flow to accurately track motions of feature points in subpixel accuracy. Initially a 5 x 5 Gaussian filter is used to smooth out the noise in order to enhance the flow computation convergence. Selecting the size of the feature region is an important trade-off. It should be large enough to include sufficient texture in it, while small enough so that the computation of the inverse Hessian matrix G will not become ill-conditional. Also, a larger region will require more computation in order to perform feature tracking. We choose 13 x 13 pixels to be the feature region. That is, each selected feature point in the first frame of each image sequence (image size 490 x 640 pixels but cropped to 417 x 385 pixels) is the center of a 13 x 13 flow region. A window function $w(x)$ = 1 over the feature region is chosen because the area surrounding each feature point is full of texture. The movement of facial feature points is then automatically tracked with subpixel accuracy in translation [3,102] via optical flow in the remaining frames of the image sequence.

The selected feature point location x is integer-valued, but $x+d$ is generally not integer-valued in order to accommodate for subpixel flows. We estimate the image gray value at an non-integer-valued pixel by using the bilinear interpolation,

$$
\begin{aligned}
I(x+d) &= I(x_1, x_2) \\
&= I(i_1, i_2) * (i_1 + 1 - x_1, i_2 + 1 - x_2) + I(i_1, i_2 + 1) * (i_1 + 1 - x_1, x_2 - i_2) + \\
&\quad I(i_1 + 1, i_2) * (x_1 - i_1, i_2 + 1 - x_2) + I(i_1 + 1, i_2 + 1) * (x_1 - i_1, x_2 - i_2)
\end{aligned} \tag{3-18}
$$

where $i_1 = \lfloor x_1 \rfloor$ and $i_2 = \lfloor x_2 \rfloor$, and $\lfloor x \rfloor$ represents the largest integer smaller than or equal to x. Since the movement of feature points will be tracked for an entire sequence in subpixel accuracy, the ending position of each tracked feature point in the first pair of frames (I_t,I_{t+1}) will be used as the starting position of the tracked point for the next pair of frames (I_{t+1},I_{t+2}), and so on. The bilinear interpolation method is applied to interpolate gray values of the non-integer-valued pixels of both the starting and ending positions for each tracked feature point in each consecutive pair of frames in the sequence.

Consecutive frames of an image sequence may contain large feature-point motion caused by gross movement of the subject between frames, such as sudden head movements, brow raised or mouth opening of the surprise expression, which may cause missing or lost tracking (Figure 12). In Figure 12, lines trailing along feature points denote their movements across image frames in the sequence. In order to recover these large motions without losing subpixel accuracy, we use a pyramid method with reduced resolution (spatial smoothing) [77]. Each image is decomposed into 5 levels from level 0 (the original finest resolution image) to level 4 (the coarsest resolution image). The image sizes are 490 x 640 (row x column), 125 x 160, 62 x 80, 31 x 40, and 15 x 20 pixels, respectively (Figure 13). We use 13 x 13-pixel window regions for each level. From level 4 to level 1 (from coarse to fine levels), we consider window-wise 1, 4, 16 and 64 flow regions for the whole image at each level, respectively. Each window center in the first frame is used as the starting position for motion estimation at that level. From level 1 to level 0, we only consider those 13 x 13 feature regions whose centroids are the locations of the previously tracked feature points. Flow computation proceeds from the

Figure 12 Feature point tracking excluding the pyramid method: it is sensitive to subtle motion such as eye blinking, but it loses tracking for large motion such as mouth opening and suddenly raising eye brows. Lines trailing along feature points (marked by a cross '+') denote their movements across image frames in the sequence.

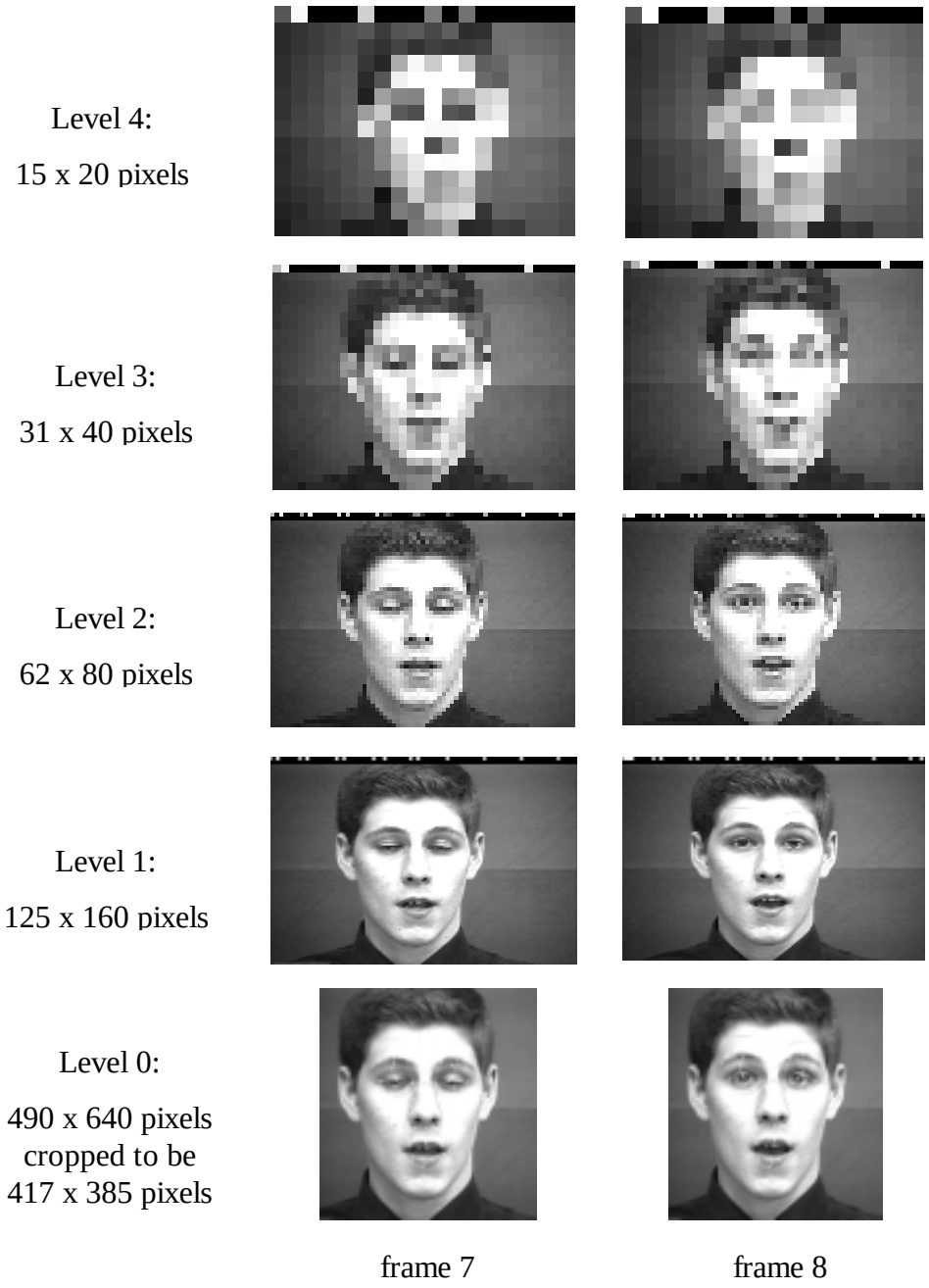


Figure 13 A 5-level pyramid for feature point tracking.

lowest resolution level (level 4) to the highest level (level 0) of the pyramid.

For the iterative computation at level l, we use the estimated motion vector d_{l+1} and confidence value C_{l+1} obtained at the coarser level l+1. At the coarsest resolution level 4, the motion vector is initialized at $(0,0)^T$ and iterated to obtain a solution d_4; confidence value C_4 is then computed. There are four feature regions at level 3, $2*d_4$ is taken as the initial motion vector of each region center and iteration proceeds to obtain a motion vector $\tilde{d}_3$ in each region with confidence value $\tilde{C}_3$. $\tilde{d}_3$ and d_4 are weighted as described below to give a new estimate d_3; so is a new estimate of C_3 obtained by weighting $\tilde{C}_3$ and C_4. In general,

$$d_l = \frac{C_{l+1} * (2 * d_{l+1}) * K + \tilde{C}_l * d_l * (1-K)}{C_{l+1} * K + \tilde{C}_l * (1-K)} \tag{3-19}$$

$$C_l = \frac{C_{l+1} * C_{l+1} * K + \tilde{C}_l * \tilde{C}_l * (1-K)}{C_{l+1} * K + \tilde{C}_l * (1-K)} \tag{3-20}$$

where $0.0 \le K \le 1.0$ *and* $1 \le l \le 3$

where K is a constant weighting factor that determines how much confidence is given to that obtained from the coarser level in the pyramid. $2*d_l$ will be used as the initial motion vector and C_l as new confidence value for iterative estimation at the next finer resolution level l-1. If K = 1.0, all flow estimates at the current level (l) are derived from the previous level (l+1) without regard to the flow estimate computed at the current level. If K = 0.0, it provokes each level's computation to use the previous level's flow estimate only as an initial value. This inter-level confidence base is used for combining the effect of spatial smoothing. In order to have a reliable flow estimation when the tracking region contains insufficient texture, the flow at the high-resolution level is mainly inherited from the flow in the previous coarse level of the pyramid, that is, K is close to but not equal to 1.0. If the tracking region has high texture, then it can be reliably tracked so as to be less involved with the flow estimated at the previous level, that is, K is close to but not equal to 0.0. We choose K=0.5 for confidence estimation. From level 1 to level 0, we first identify the locations at level 1 corresponding to the feature points at level 0, which in

general will not coincide with the region centers at level 1. At each of these locations, its motion vector will be estimated by the bilinear transformation from its four neighboring centers. This estimated vector multiplied by 4 will be used as the initial motion vector at the corresponding feature point for iterative estimation at level 0 to obtain the estimate d_0 in the feature point tracking. Then the process repeats for the next consecutive point of frames.

Using this pyramid method for optical flow computation, we initially enable the gradient descent method at low resolution levels to avoid local minima in the search for the optimal solution of feature point displacement. This allows us to recover any large motion (up to 100-pixel displacement) of the feature point while maintaining its sensitivity to subtle (subpixel) facial motion (as shown in Figure 14), and the flow computation converges quickly (less than 20 seconds for tracking 70 feature points between two consecutive frames using the interface under SUN Sparc 5). Point tracking method deals very well with large feature point movement between two 490 x 640-pixel frames.

3.5 Analysis of Feature Point Tracking Problems

It has been noted in our experiments that an error may occurred when some facial feature points located at the edge of the brows or mouth had large movements between two consecutive frames. Those selected feature points were tracked along the edge direction of the brows or mouth (Figure 15). This is due to the fact that the tracking was sensitive along the low gradient direction when this region contains a high gradient line.

There are three ways to correct these errors. One way is to locate those feature points away from edges instead of along the edges. The error is reduced for brows raised but still occurs along edges when mouth opens larger (Figure 16), because the mouth motion causes more facial deformation than that by brow motion. Another solution is to have a larger n x n feature region R to include more motion information (Figure 17), but it requires more computation time for tracking. Still another method, which we use, is to

increase the value of the weighting factor *K* of the inter-level confidence base so as to include more global information from the previous low-resolution level processing in the pyramid method (Figure 18).

3.6 Data Quantization and Conversion for the Recognition System

Three upper facial expressions are to be recognized based on displacements of 6 feature points at the upper boundaries of both brows, and six lower face expressions are to be recognized based on displacements of 10 feature points around the mouth. Feature points are numbered from left to right for brows region, and from the left corner point of lip clockwise around the mouth. The displacement of each feature point is calculated by subtracting its normalized position in the first frame from its current normalized position. Each feature point has the horizontal displacement component and vertical displacement component. The displacement vector is 12-dimensional in the upper face and 20-dimensional in the lower face (Figure 19). Facial expressions are characterized by these two vector sequences. These displacement vectors in upper and lower facial regions are vector-quantized separately into 16 and 32 symbols, respectively, as discussed in section 6.1 and section 8.3.1. Table 4 shows sample symbol sequences for nine facial expressions under consideration. Such symbol sequences are used as inputs to the HMMs of upper facial expressions and lower facial expressions, respectively, for automatic recognition.

Figure 14 Feature point tracking including the pyramid method: it is sensitive to subtle motion such as eye blinking and also tracks accurately for large motion such as mouth opening and suddenly raising eye brows.

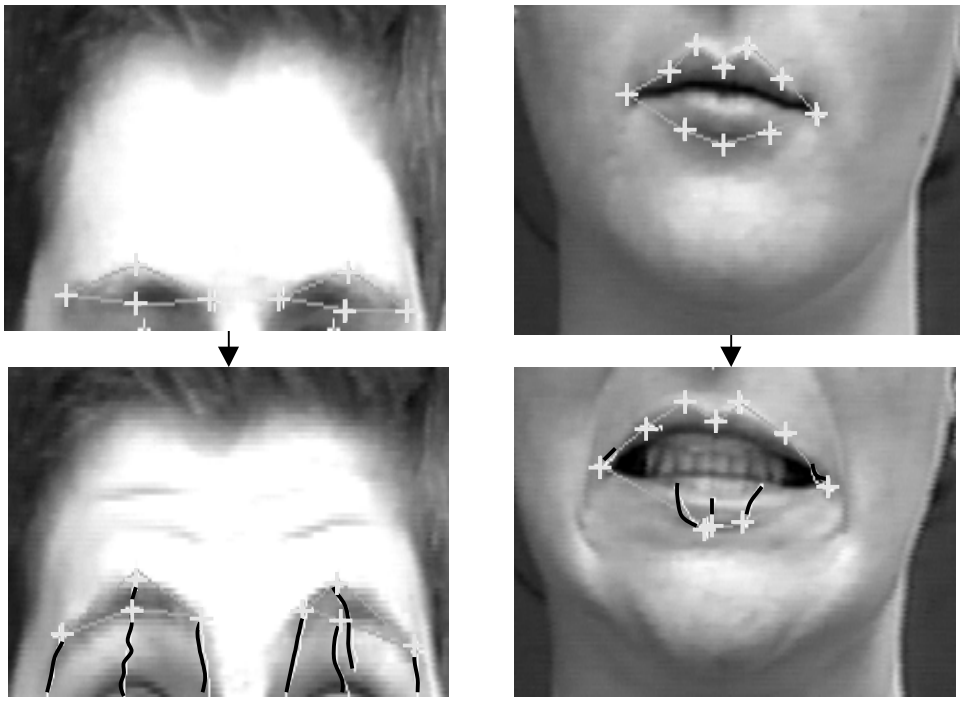


Figure 15 The feature point tracking error due to violation of the feature region's trackability condition: tracking along the edge direction at both brows and mouth regions with deformed shapes.

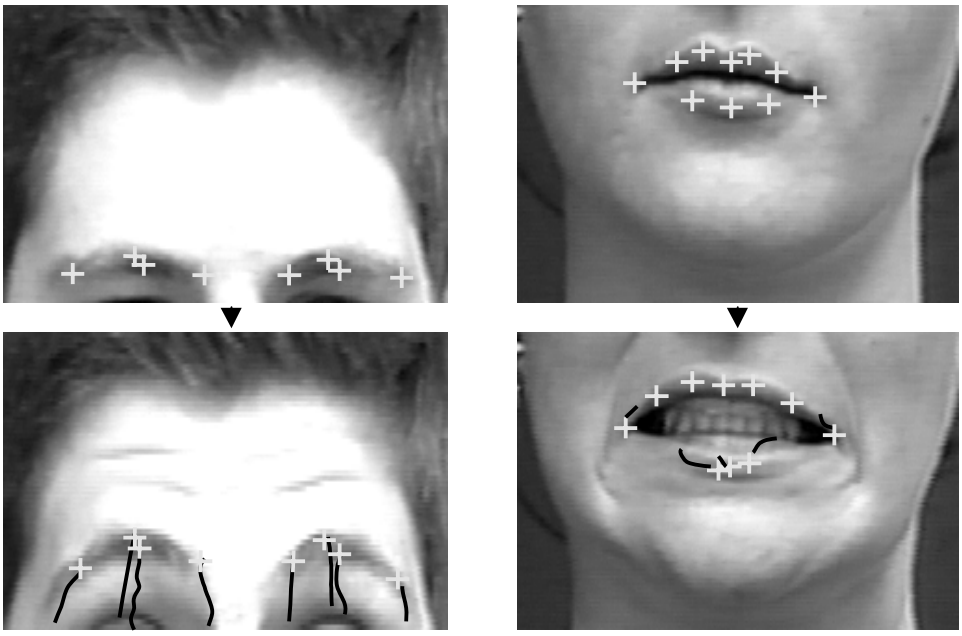


Figure 16 Reducing the tracking error by locating feature points away from edges of facial features: the tracking for brow region is improved, but is still erroneous in mouth region with large mouth opening.

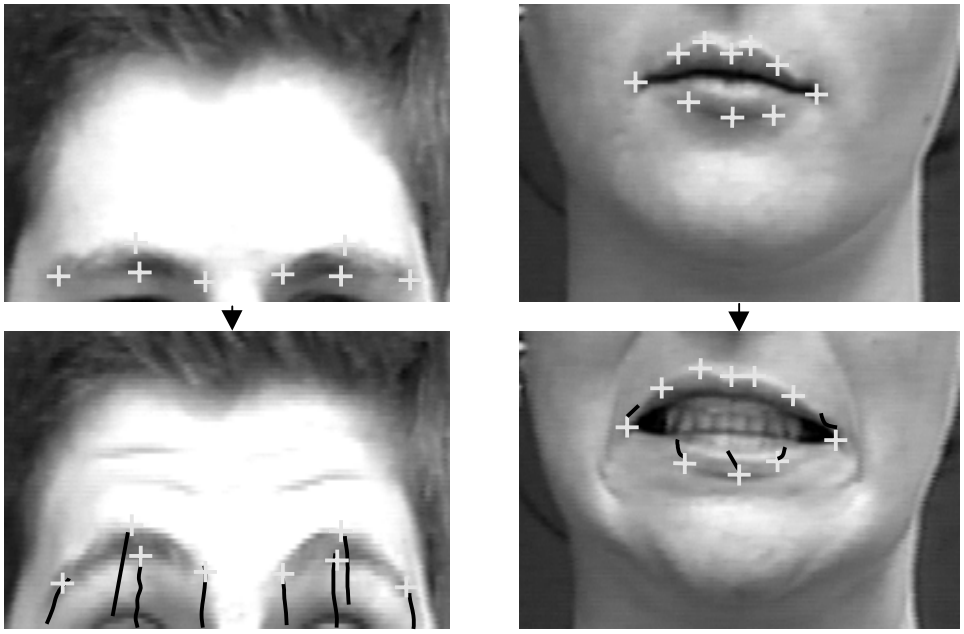


Figure 17 Reducing the tracking error by using a large window size which requires more processing time.

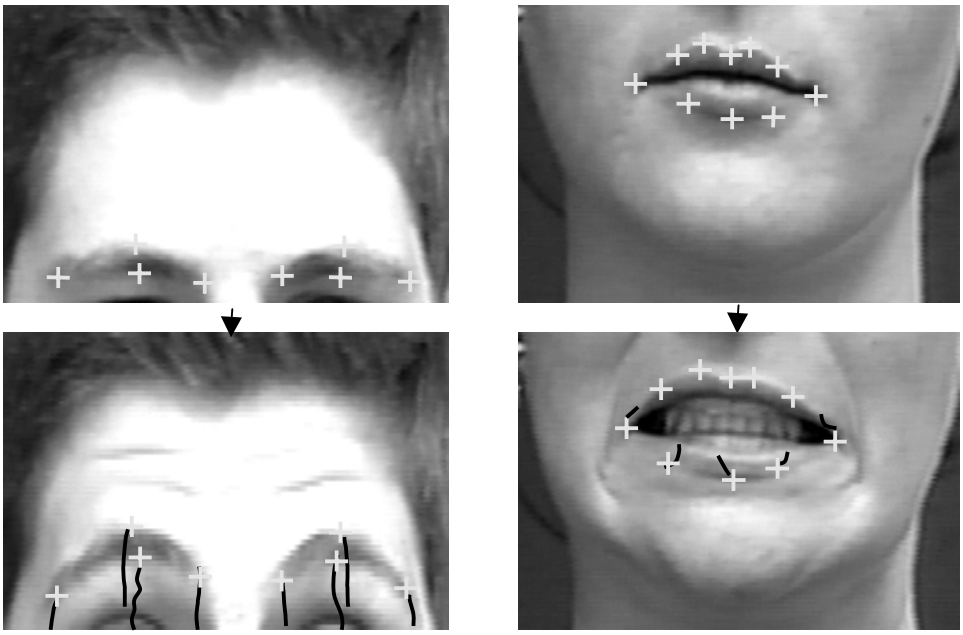
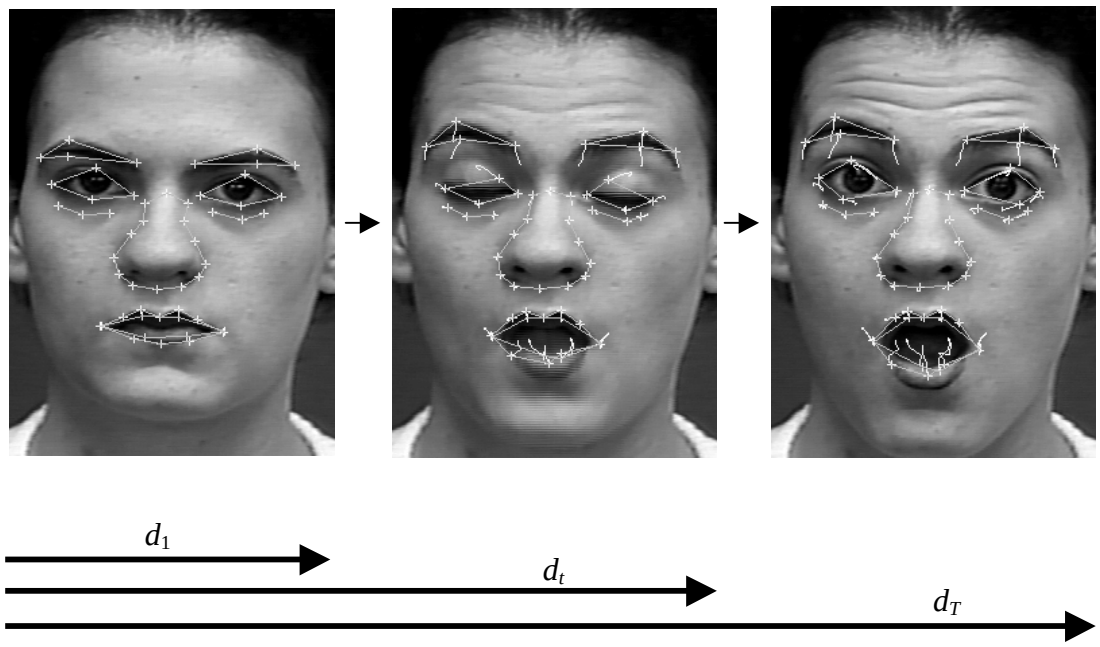


Figure 18 Reducing the tracking error by increasing the value of the weighting factor K (K=0.5) of the inter-level confidence base in order to include more global information from the previous low-resolution level processing in the pyramid method.



Displacement vector $d_t = (d_{t,1}, d_{t,2}, \ldots, d_{t,j}, \ldots, d_{t,i})$

where $d_{t,j} = (d_{t,j,horizontal}, d_{t,j,vetrical})$ is the pair of horizontal and vertical displacements for feature point j at frame t,

$i = 6$ for brow region (upper facial expression)

$i = 10$ for mouth region (lower facial expression)

and feature points are numbered from left to right for brow region, and from the left corner point of lip clockwise around the mouth.

Displacement vector sequence $D = (d_1, d_2, \ldots, d_t, \ldots, d_T)$ for each of the upper and lower facial expressions where T is the length of an image sequence.

Figure 19 Displacement vector of the facial feature point tracking to be encoded for input to a Hidden Markov Model.

Table 4 Sample symbol sequences for three upper facial expressions and six lower facial expressions under consideration.

AUs	**Feature Point Tracking: Upper Facial Expressions (Symbol Sequence)**
4	3 3 3 3 *5* 5 5 5 5 *5* 5 5
1+4	3 3 3 3 *3* 3 3 1 1 *1* 1 1 1 1 *1* 1 1
1+2	3 3 3 3 *3* 6 6 6 6 *6* 6 6
AUs	**Feature Point Tracking: Lower Facial Expressions (Symbol Sequence)**
12	2 2 2 2 *2* 2 2 9 1 *1* 1
6+12+25	2 2 2 2 *10* 5 5 5 11 *11* 11 11 11 11 *11*
20+25	2 10 10 10 *13* 13 13 13 13 *13* 13
9+17	2 2 3 3 *3* 14 14 14 14 *14*
15+17	2 2 2 2 *6* 6 6 6 6 *6* 6 6 6
17+23+24	2 2 2 2 *4* 4 4 4 4

d	0	1	3	5	7	10	14	20
η	0.00	0.14	0.39	0.58	0.71	0.82	0.89	0.94

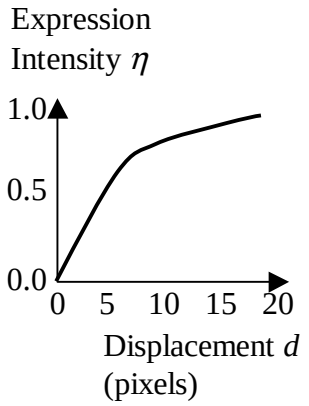


Figure 20 An example illustrating the expression intensity estimation by following the non-linear mapping (k = 7) of the constrained feature point displacement for the case of AU2.

3.7 Expression Intensity Estimation

Since displacements of feature points correspond to AU intensities of facial expressions and indicate the underlying muscle motion, we can quantify the expression intensity based on the displacements of feature points. Figure 20 and Table 5 show the logic of expression intensity estimation and the specified constraints for displacement measurement of each feature point which corresponds to the expression intensity of individual AU. We propose to estimate the expression intensity η by using following non-linear mapping of the feature point displacement

$$\eta = \frac{d}{\sqrt{d^2 + k^2}} \qquad \textit{where } 0 \le \eta \le 1.0 \tag{3-21}$$

where d is the measured normalized displacement of a feature point under constraint for a AU as specified in Table 5. Since the motion of a feature point during a facial expression is described, in general, initial as gradual motion (starting from the neutral expression) followed by quick motion then gradually slowing down until the peak expression is reached, the value of constant k is determined empirically according to individual AU

Table 5 The dictionary for expression intensity estimation (image: 417 x 385 pixels).

Action Unit	**Constraint: direction** 	**Measured Displacement *d***	***k*** **(Pixel)**
	Brows		
1	Inner point of brow moves vertical up	Vertical displacement	7
2	Outer point of brow moves vertical up	Vertical displacement	7
4	1. Inner point of brow moves vertically down 2. Inner points of both brows move horizontally toward one another	1.Vertical displacement 2.Horizontal distance between both points	4
	Eyes		
5	Middle point of upper eyelid moves up	Vertical displacement	2
7	Middle point of lower eyelid moves up	Vertical displacement	2
41~46	Middle points of eyelids move vertically together (narrow eyes or blinking)	Vertical displacement between both points	4
	Nose		
9	Side point at nostril moves up	Vertical displacement	5
	Mouth		
12	1. Left corner point of lip moves up between 100 and 170 degrees 2. Right corner point of lip moves up between 10 and 80 degrees	1. Euclidean distance 2. Euclidean distance	6
20	1. Left corner point of lip moves horizontally between 170~180 and -170~-180 degrees 2. Right corner point of lip moves horizontally between 0~10 and 0~-10 degrees	1. Euclidean distance 2. Euclidean distance	6
15	1. Left corner point of lip moves down between -90 and -170 degrees 2. Right corner point of lip moves down between -10 and -90 degrees	1. Euclidean distance 2. Euclidean distance	3
18	Both corner points of lip move horizontally toward one another	Horizontal distance between both lip corners	8
23	Two points on upper lip move horizontally together	Horizontal distance between both points	2
24	Both center points on lips move vertically together	Vertical distance between both points	5
25~27	Both center points on lips move vertically away from baseline	Vertical distance between both points	20

Table 5 (Continued)
The dictionary for expression intensity estimation (image: 417 x 385 pixels).

Action Unit	**Constraint: direction** 180^0 -180^0 0^0	**Measured Displacement *d***	***k*** **(Pixel)**
	Mouth		
17	Existence of furrow or wrinkle on the chin	Not measured	-

(listed in Table 5) to give a genuine expression intensity time course fit to the velocity of the facial motion (like the oscillation of a spring between compression and release) as shown in Figure 21.

The expression intensity estimation given above is one first attempt to quantify AU expression measurement. In reality, it is difficult to measure the expression of an individual AU by tracking a single feature point which indicates a single muscle movement. For example, both AU15 and AU20 involve downward motions of feature points at lip corners. Although the FACS system assumes that there is a one-to-one mapping between an AU and a single muscle motion, the expression intensity of an individual AU may be composed of the coordinated movements of multiple muscles or movements of multiple feature points. It would be desirable to consider a set of feature points corresponding to an "expression unit," and measure their simultaneous motion for quantifying the expression intensity. This will be a challenging aspect for future research.

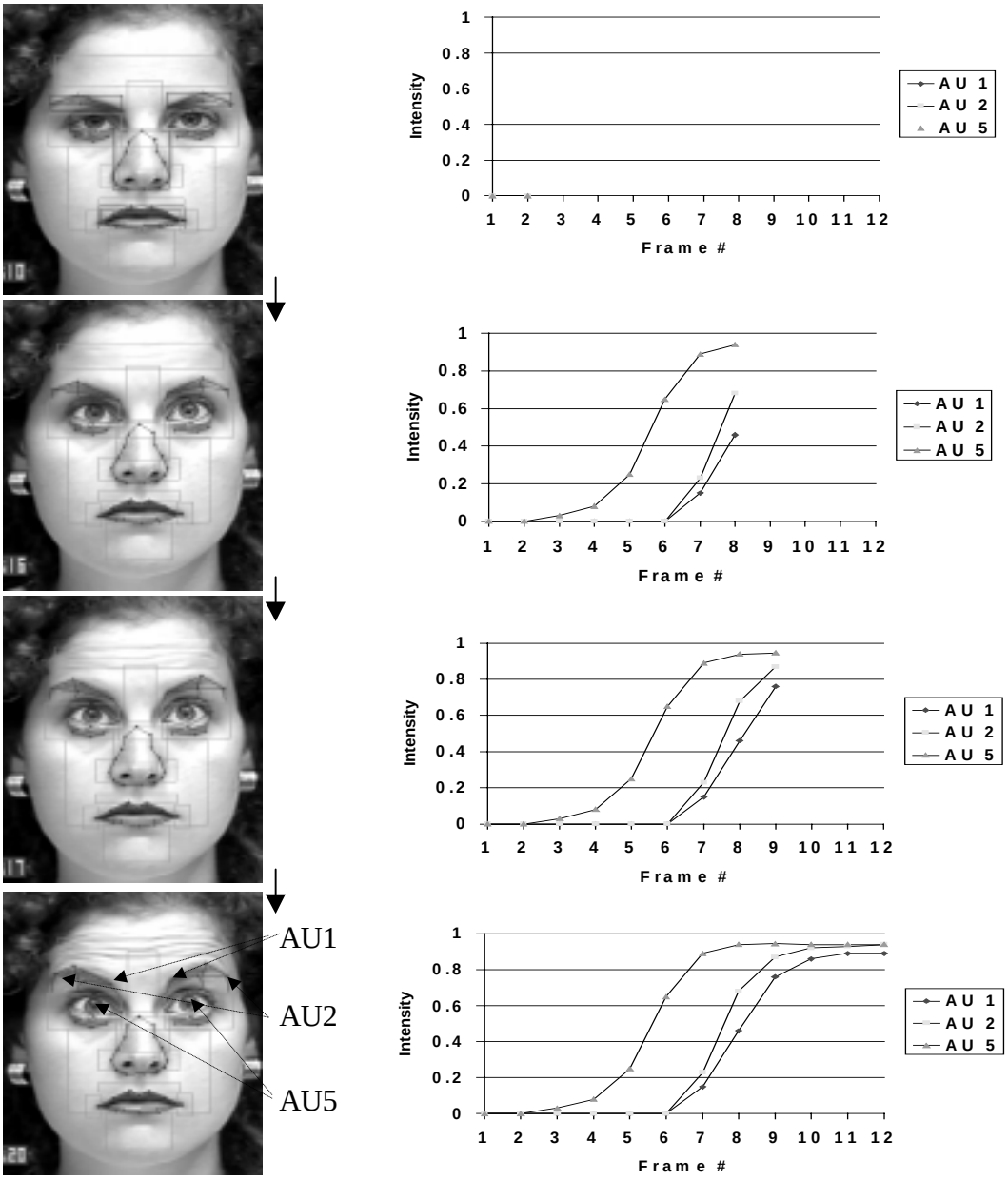


Figure 21 Expression intensity time course of AU1, 2 and 5 fit to the displacement changes of facial feature points based on the non-linear mapping [107].

4.0 DENSE FLOW TRACKING AND EIGENFLOW COMPUTATION

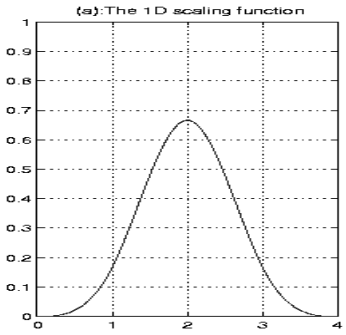
Feature point tracking of eyebrows and mouth presented in Chapter 3 is sensitive to subtle feature motions and is capable to track large displacements. The forehead, cheek and chin regions also contribute important facial expression information. Since the actions of individual facial muscles are interdependent, and the activation of fibers in one muscle may influence movements of adjacent muscles. Single facial expression may be the result of movements and deformation of not only facial features but also facial skins which are caused by facial muscle actions triggered by nerve impulses. To enhance the realism of an automatic recognition system, it is desirable to capture more detailed motion information. This leads to the consideration dense flow which describes the motion of each pixel on the entire face image.

4.1 Wavelet-Based Motion Estimation

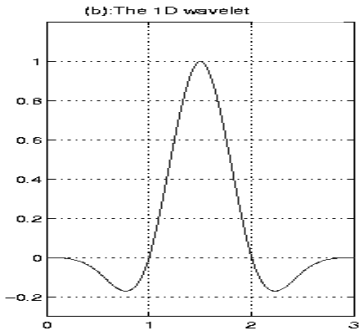
Wu's approach of dense flow estimation [101] is employed to estimate the entire facial motion. The flow window functions are based on scaling function and wavelet of Cai and Wang [18] (Figure 22.a), which provide wavelet coefficients from the coarse-to-fine resolution levels. This approach estimates image motion in large regions with large window support at the coarse resolution level and motions in small regions with small window support at the fine resolution level. It is different from the traditional coarse-to-fine pyramid method by taking into account the detail information decomposing image resolutions at various resolution levels. Both coarser and finer level motions can be simultaneously estimated to correct the error produced by the previous motion estimation at the coarse level. It will yield more accurate motion estimation.

Let us consider the one-dimensional case first. The scaling function ϕ is the fourth-

Scaling function $\phi(x)$

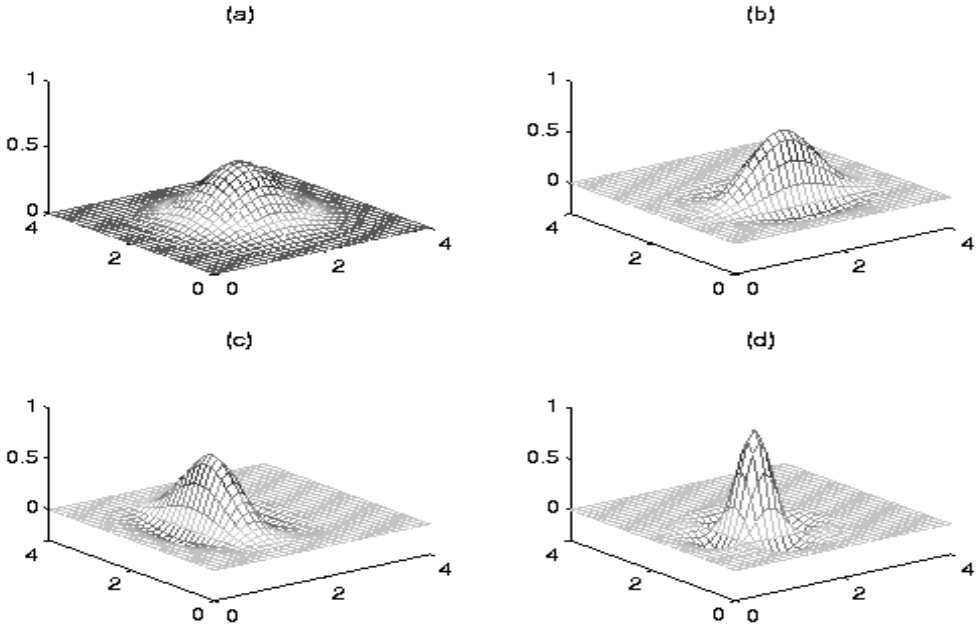


Wavelet $\psi(x)$



x x

1-dimensional scaling function and wavelet.



2-dimensional scaling function and wavelets: (a) $\phi(x,y) = \phi(x)\ \phi(y)$,
(b) $\psi^H(x,y) = \phi(x)\ \psi(y)$, (c) $\psi^V(x,y) = \psi(x)\ \phi(y)$, and (d) $\psi^D(x,y) = \psi(x)\ \psi(y)$.

Figure 22.a 1- and 2-dimensional scaling functions and wavelets [101].

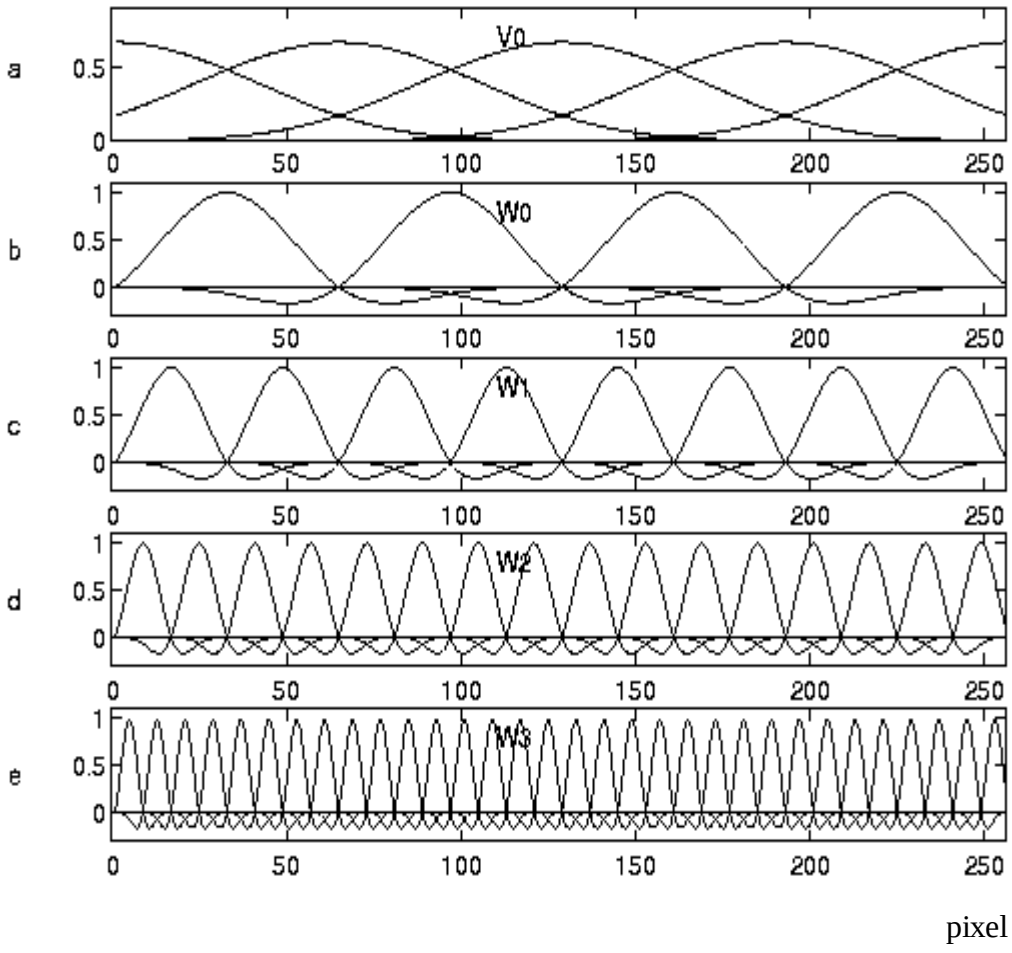


Figure 22.b Dilation and translation of 1-dimensional basis (scaling and wavelet) functions [101].

order (L = 4) B-spline function acting as a low-pass filter for smoothing the signals, this is represented by

$$\phi(x) = \frac{1}{6}\sum_{j=0}^{L}\binom{4}{j}(-1)^j (x-j)^3 \qquad \in H^2(I) \qquad \text{where } I = [0,L=4] \qquad (4\text{-}1)$$

where I denotes a finite interval and $H^2(I)$ represents the Sobolev space containing all continuous functions with the finite energy norm up to the second derivative. The wavelet $\psi(x)$ acting as a high-pass filter is derived from the scaling function through a 2-scale equation.

$$\psi(x) = \frac{-3}{7}\phi(2x) + \frac{12}{7}\phi(2x-1) - \frac{-3}{7}\phi(2x-2) \qquad (4\text{-}2)$$

The interval for $\psi(x)$ is [0,3]. The dilation 2^j and translation k of $\phi(x)$ and $\psi(x)$ are given by (Figure 22.a for j=0 and k=0)

$$\phi_{j,k}(x) = \phi(2^j x - k) \qquad j \geq 0, \quad k = \text{-}2,..., 2^j L\text{-}2 \qquad (4\text{-}3)$$

$$\psi_{j,k}(x) = \psi(2^j x - k) \qquad j \geq 0, \quad k = \text{-}1,..., 2^j L\text{-}2 \qquad (4\text{-}4)$$

A finite energy continuous function $d(x)$, which represents the motion at position x, can be decomposed into a scaling component $d_{-1}(x)$ at the coarsest resolution level -1 and many wavelet components $d_j(x)$'s at resolution levels $j \geq 0$,

$$d(x) \approx d_{-1}(x) + d_0(x) + d_1(x) + \ldots + d_j(x) \qquad (4\text{-}5)$$

where

$$d_{-1}(x) = \sum_{k=-2}^{L-2} c_{-1,k}\phi_{0,k}(x) \qquad (4\text{-}6)$$

$$d_j(x) = \sum_{k=-1}^{2^j L-2} c_{j,k}\psi_{j,k}(x) \qquad j \geq 0 \qquad (4\text{-}7)$$

In the motion estimation problem, $d(x)$ is unknown but satisfies the optical flow equation. Its solution is to be computed by the above multi-resolution approximation (Figure 22.b). By repeatedly estimating [101] the coefficients $c_{j,k}$, each component $d_j(x)$ is constructed from the linear combination of translated basis functions at level j.

For the case of a two-dimensional images, the two-dimensional scaling functions and wavelets may be constructed from the tensor products of two scaling function and wavelet (Figure 22.a for j=0, k_1=0 and k_2=0). Hence,

$$\phi_{j,k_1,k_2}(x,y) = \phi(2^j x - k_1)\phi(2^j y - k_2) \tag{4-8}$$

$$\psi^H_{j,k_1,k_2}(x,y) = \phi(2^j x - k_1)\psi(2^j y - k_2) \tag{4-9}$$

$$\psi^V_{j,k_1,k_2}(x,y) = \psi(2^j x - k_1)\phi(2^j y - k_2) \tag{4-10}$$

$$\psi^D_{j,k_1,k_2}(x,y) = \psi(2^j x - k_1)\psi(2^j y - k_2) \tag{4-11}$$

where j, k_1, and k_2 denote the resolution level, horizontal translation, and vertical translation, respectively.

Let $u(x,y)$ and $v(x,y)$ be the displacement of flow functions at the horizontal and vertical directions, respectively. They can be closely approximated by linear combinations of the scaling functions and wavelets (window functions) given in equations (4-8) through (4-11) from the coarsest motion-resolution level -1 to the fine motion-resolution level J.

$$u(x,y) = u_{-1}(x,y) + \sum_{j=0}^{J}[u^H_j(x,y) + u^V_j(x,y) + u^D_j(x,y)] \tag{4-12}$$

$$v(x,y) = v_{-1}(x,y) + \sum_{j=0}^{J}[v^H_j(x,y) + v^V_j(x,y) + v^D_j(x,y)] \tag{4-13}$$

where

$$u_{-1}(x,y) = \sum_{k_1=-2}^{L_1-2}\sum_{K_2=-2}^{L_2-2} c_{-1,k_1,k_2}\,\phi_{0,k_1,k_2}(x,y) \tag{4-14}$$

$$u^H_j(x,y) = \sum_{k_1=-2}^{2^j L_1-2}\sum_{K_2=-1}^{2^j L_2-2} c^H_{j,k_1,k_2}\psi^H_{j,k_1,k_2}(x,y) \tag{4-15}$$

$$u^V_j(x,y) = \sum_{k_1=-1}^{2^j L_1-2}\sum_{K_2=-2}^{2^j L_2-2} c^V_{j,k_1,k_2}\psi^V_{j,k_1,k_2}(x,y) \tag{4-16}$$

$$u^D_j(x,y) = \sum_{k_1=-1}^{2^j L_1-2}\sum_{K_2=-1}^{2^j L_2-2} c^D_{j,k_1,k_2}\psi^D_{j,k_1,k_2}(x,y) \tag{4-17}$$

and

$$v_{-1}(x,y) = \sum_{k_1=-2}^{L_1-2} \sum_{K_2=-2}^{L_2-2} d_{-1,k_1,k_2} \phi_{0,k_1,k_2}(x,y) \tag{4-18}$$

$$v_j^H(x,y) = \sum_{k_1=-2}^{2^j L_1-2} \sum_{K_2=-1}^{2^j L_2-2} d_{j,k_1,k_2}^H \psi_{j,k_1,k_2}^H(x,y) \tag{4-19}$$

$$v_j^V(x,y) = \sum_{k_1=-1}^{2^j L_1-2} \sum_{K_2=-2}^{2^j L_2-2} d_{j,k_1,k_2}^V \psi_{j,k_1,k_2}^V(x,y) \tag{4-20}$$

$$v_j^D(x,y) = \sum_{k_1=-1}^{2^j L_1-2} \sum_{K_2=-1}^{2^j L_2-2} d_{j,k_1,k_2}^D \psi_{j,k_1,k_2}^D(x,y) \tag{4-21}$$

The four window functions are used for the following representations. $\phi_{0,k_1,k_2}(x,y)$ is used initially to represent the flow at the coarsest level so as to achieve a fast convergence. Wavelets $\psi_{j,k_1,k_2}^H(x,y)$, $\psi_{j,k_1,k_2}^V(x,y)$ and $\psi_{j,k_1,k_2}^D(x,y)$ are used to represent the high gradient components of the optical flow at the *j*th resolution level in the vertical, horizontal and diagonal directions, respectively. Based on these four basis window functions, we can estimate the Hessian matrix *G* and difference-gradient vector *e* by [101]

$$G = \sum_{x,y} g(x,y) g(x,y)^T w(x,y) \tag{4-22}$$

$$e = \sum_{x,y} [I_t(x - u(x,y), y - v(x,y)) - I_{t+1}(x,y)]\, g(x,y) w(x,y) \tag{4-23}$$

where

window (weighted) function $w(x,y)$ is unity, (4-24)

$$g(x,y) = \begin{bmatrix} \phi_{0,-2,-2} I_x' \cdots \phi_{0,L_1-2,L_2-2} I_x' & \psi_{0,-2,-1}^H I_x' \cdots \psi_{j,2^j L_1-2,2^j L_2-2}^H I_x' \\ \psi_{0,-1,-2}^V I_x' \cdots \psi_{j,2^j L_1-2,2^j L_2-2}^V I_x' & \psi_{0,-1,-1}^D I_x' \cdots \psi_{j,2^j L_1-2,2^j L_2-2}^D I_x' \\ \phi_{0,-2,-2} I_y' \cdots \phi_{0,L_1-2,L_2-2} I_y' & \psi_{0,-2,-1}^H I_y' \cdots \psi_{j,2^j L_1-2,2^j L_2-2}^H I_y' \\ \psi_{0,-1,-2}^V I_y' \cdots \psi_{j,2^j L_1-2,2^j L_2-2}^V I_y' & \psi_{0,-1,-1}^D I_y' \cdots \psi_{j,2^j L_1-2,2^j L_2-2}^D I_y' \end{bmatrix}^T \tag{4-25}$$

$$I_t^{'}(x,y) = (I_x^{'}, I_y^{'})$$

$$= (\frac{\partial I_t(x-u(x,y), y-v(x,y))}{\partial x}, \frac{\partial I_t(x-u(x,y), y-v(x,y))}{\partial y}) \qquad (4\text{-}26)$$

Since the feature of Cai and Wang's basis functions is to use wavelet coefficients from coarse-to-fine levels to represent any given function, the flow functions $u(x,y)$ and $v(x,y)$ can be determined by estimating the wavelet coefficient vectors $c^T=(\ldots, c_{j,k_1,k_2},\ldots)$ and $d^T=(\ldots, d_{j,k_1,k_2},\ldots)$ from the coarsest level –1 to current level j using iterations of both coefficient vectors,

$$(c^T, d^T)^T = ([\ldots c_{-1,k_1,k_2} \ldots c^H_{0,k_1,k_2} \ldots c^V_{0,k_1,k_2} \ldots c^D_{0,k_1,k_2} \ldots c^H_{j,k_1,k_2} \ldots c^V_{j,k_1,k_2} \ldots c^D_{j,k_1,k_2} \ldots],$$

$$[\ldots d_{-1,k_1,k_2} \ldots d^H_{0,k_1,k_2} \ldots d^V_{0,k_1,k_2} \ldots d^D_{0,k_1,k_2} \ldots c^H_{j,k_1,k_2} \ldots c^V_{j,k_1,k_2} \ldots c^D_{j,k_1,k_2} \ldots])^T$$

$$= G^{-1}e \qquad (4\text{-}27)$$

where

$-2 \le k_1 \le L_1 - 2$ and $-2 \le k_2 \le L_2 - 2$ for c_{-1,k_1,k_2} and d_{-1,k_1,k_2}

$-2 \le k_1 \le 2^j L_1 - 2$ and $-1 \le k_2 \le 2^j L_2 - 2$ for c^H_{j,k_1,k_2} and d^H_{j,k_1,k_2}

$-1 \le k_1 \le 2^j L_1 - 2$ and $-2 \le k_2 \le 2^j L_2 - 2$ for c^V_{j,k_1,k_2} and d^V_{j,k_1,k_2}

$-1 \le k_1 \le 2^j L_1 - 2$ and $-1 \le k_2 \le 2^j L_2 - 2$ for c^D_{j,k_1,k_2} and d^D_{j,k_1,k_2}

and

$L_1 = L_2 = 4$, and $j = 0, 1, \ldots, J$

4.2 Dense Flow Tracking

In this method, the wavelet-based dense flow is used to automatically track a large region, for example, the entire 417 x 385-pixel face image (cropped from the original 490 x 640-pixel image) for each image sequence of a certain length (from the neutral to the peak expression) so as to include the whole motion information of a facial expression (Figure 23). Since the movement of each pixel is estimated between two consecutive frames, the ending position of each tracked pixel at the previous motion estimation (between images I_{t-1} and I_t) is the beginning position at the current motion estimation (between images I_t and I_{t+1}). The motion of subpixel accuracy is estimated, and the gray value at the non-integer-valued ending position for each tracked pixel is bilinearly interpolated for further processing.

Because using the wavelet-based dense flow method is very time consuming at the present time, taking more than 2 hours for three-level (-1, 0 and 1) or 20 minutes for two-level (-1 and 0) computation between two 417 x 385-pixel frames using a SGI-Irix workstation, we use the two-level wavelet-based dense flow computation to save time. When less levels are used, it will restrict how small the window size can be at the finest level and, hence, may be less sensitive to subtle motions (less than 2 pixels). It will also miss tracking large displacements (more than 15 pixels). In spite of these, our experimental work has shown better overall recognition rate in comparison to the feature point tracking in 5 levels. Since facial expressions are produced by movements of interdependent muscles over a region, the dense flow tracking has the advantage that it can include the entire motion information and so may be more effective to capture the facial motion.

The scaling function and wavelets at multi-resolution levels provide window functions of multiple support sizes to capture both local and global characteristics in the optimization process. This makes the wavelet-based motion estimation stable and accurate, especially for the low texture regions where the large window size at the coarse level may include sufficient higher gradient information in the neighboring regions to

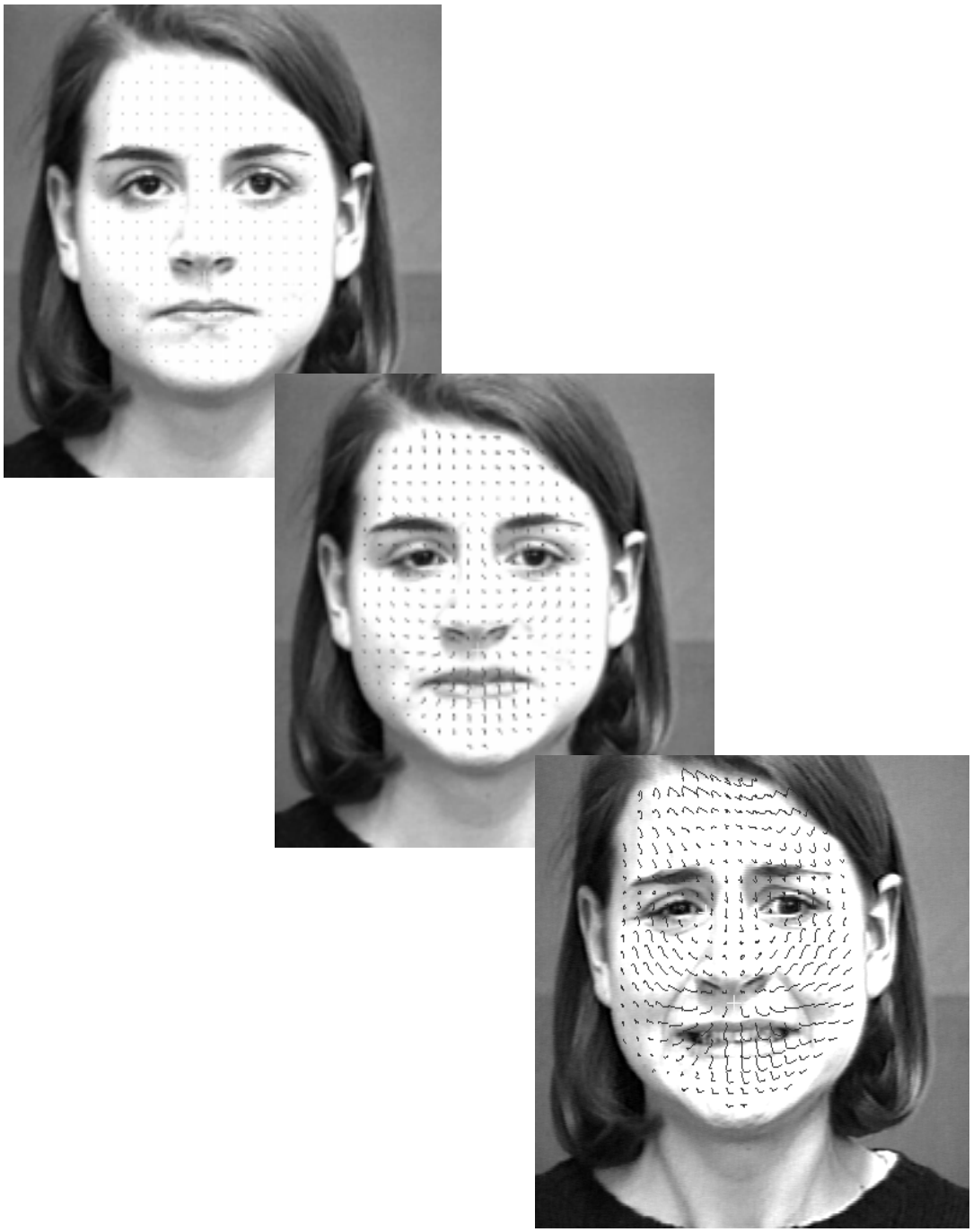


Figure 23 Automatic dense flow tracking for an image sequence. Out-of-plane motion (pitch) occurs at the bottom image. Dense flows are shown once for every 13 pixels.

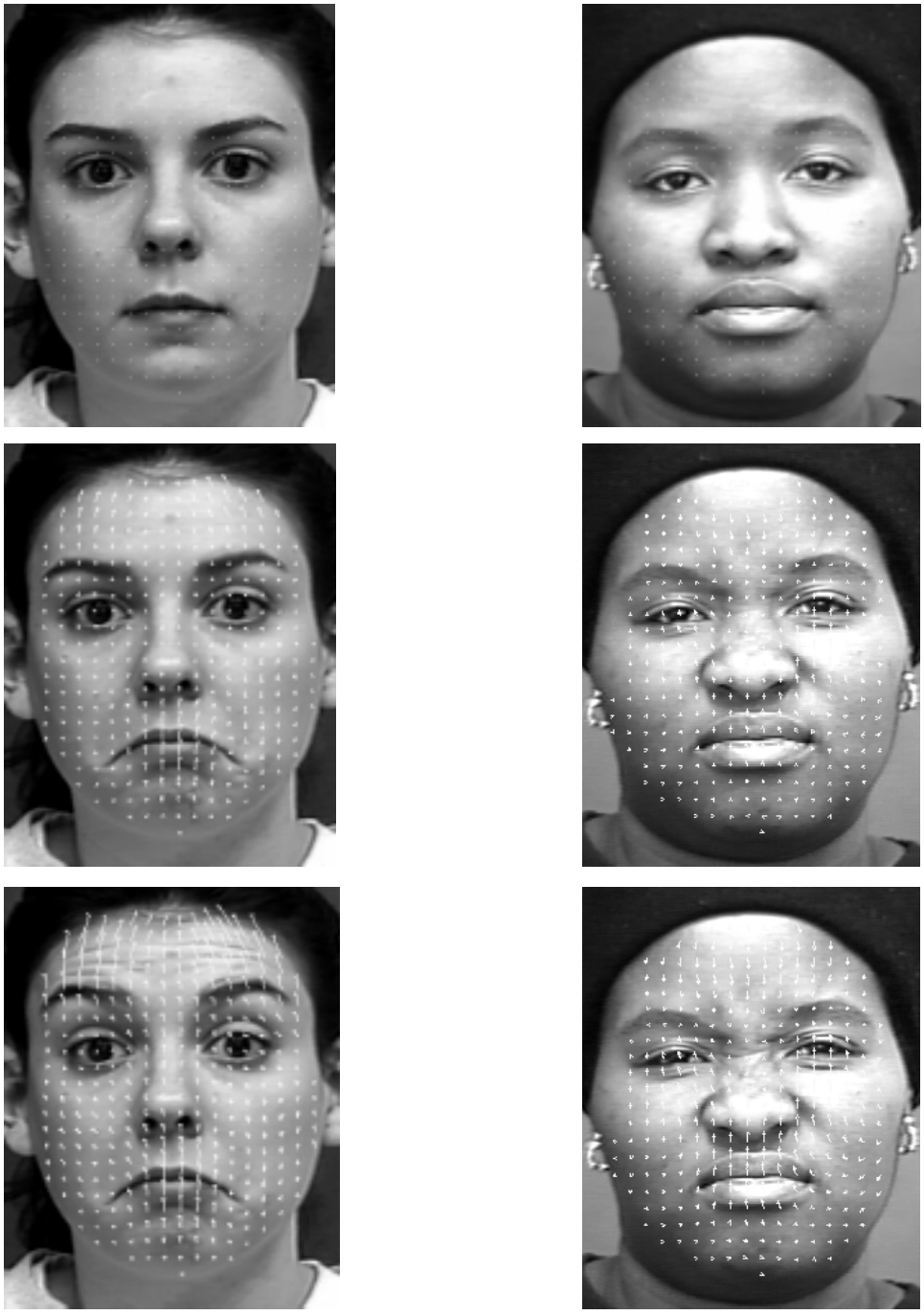
give a consistent motion information (Figure 24). It is also capable to track motions of certain discontinuities such as furrows with only two coarse-to-fine levels (-1 and 0) chosen in the current scheme, however, the resulting window functions are not localized enough to accurately capture sharp changes in the motion field (large motion more than 15 pixels) in a highly textured region such as the large movement of brows raised or of mouth opening together with the appearance of high gradient components, for example, teeth and tongue (Figure 25), while the multi-level feature point tracking can track 100- pixel movement with fast computation. Nevertheless, the overall performance of the wavelet-based dense flow is very good. The main issue is how to significantly improve the computation speed to enable more than 2-level estimation.

4.3 Eigenflow Computation

The motion captured in consecutive frames of an image sequence is strongly correlated. The information gathered by 417 x 385-pixel dense flows of many frames each sequence need to be compressed to retain significant characteristics and inter-frame correlations for yielding an efficient representation of facial expressions. The principal component analysis (PCA) has excellent properties and can be used to achieve this purpose. Although PCA has been widely applied to image gray values. This is one of the pioneering researches that it is being applied to motion fields.

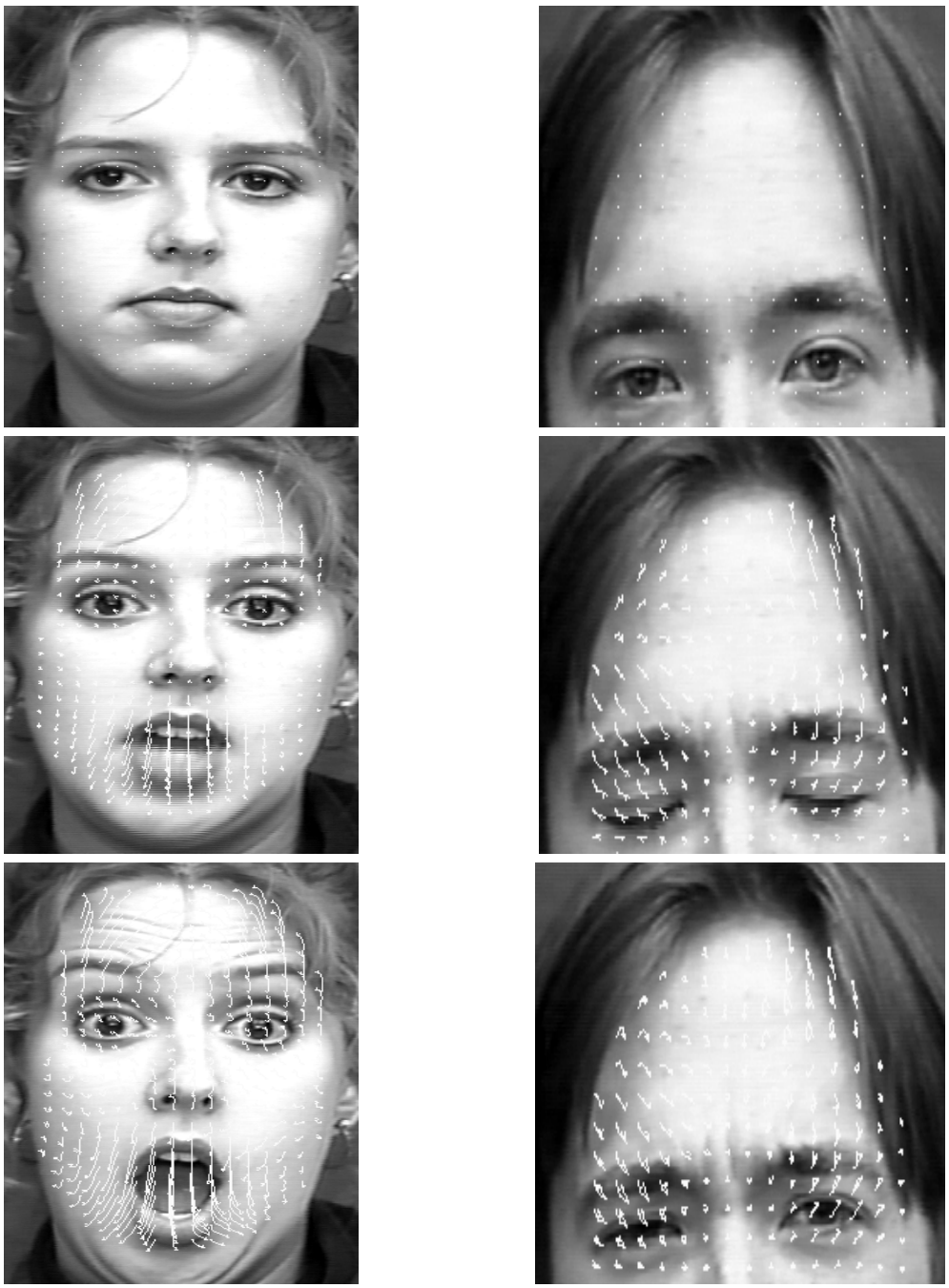
Before applying the PCA, it is necessary to ensure that the dense flows of individual frames have relative geometric correspondence. An affine transformation described before is used to automatically warp the dense flow of each frame to the two-dimensional face model based on three points: the medial canthus of both eyes and the uppermost point on the philtrum (Figure 26).

Based on FACS criteria, we can separate facial expressions into upper face motion (forehead, brows and eyes) and lower face motion (eyes, cheek, nose, mouth and chin) for facial expression analysis. It is assumed that AU expressions at upper and lower facial



(a) (b)

Figure 24 Good tracking performance of using the wavelet-based dense flow for (a) furrow discontinuities at the forehead and chin regions, and (b) textureless regions with reflections at the forehead and cheek.



(a) (b)

Figure 25 Tracking errors of the 2-level wavelet-based dense flow because of (a) large movements of brows or mouth, and (b) eye blinking also introduces motion error at brow regions.

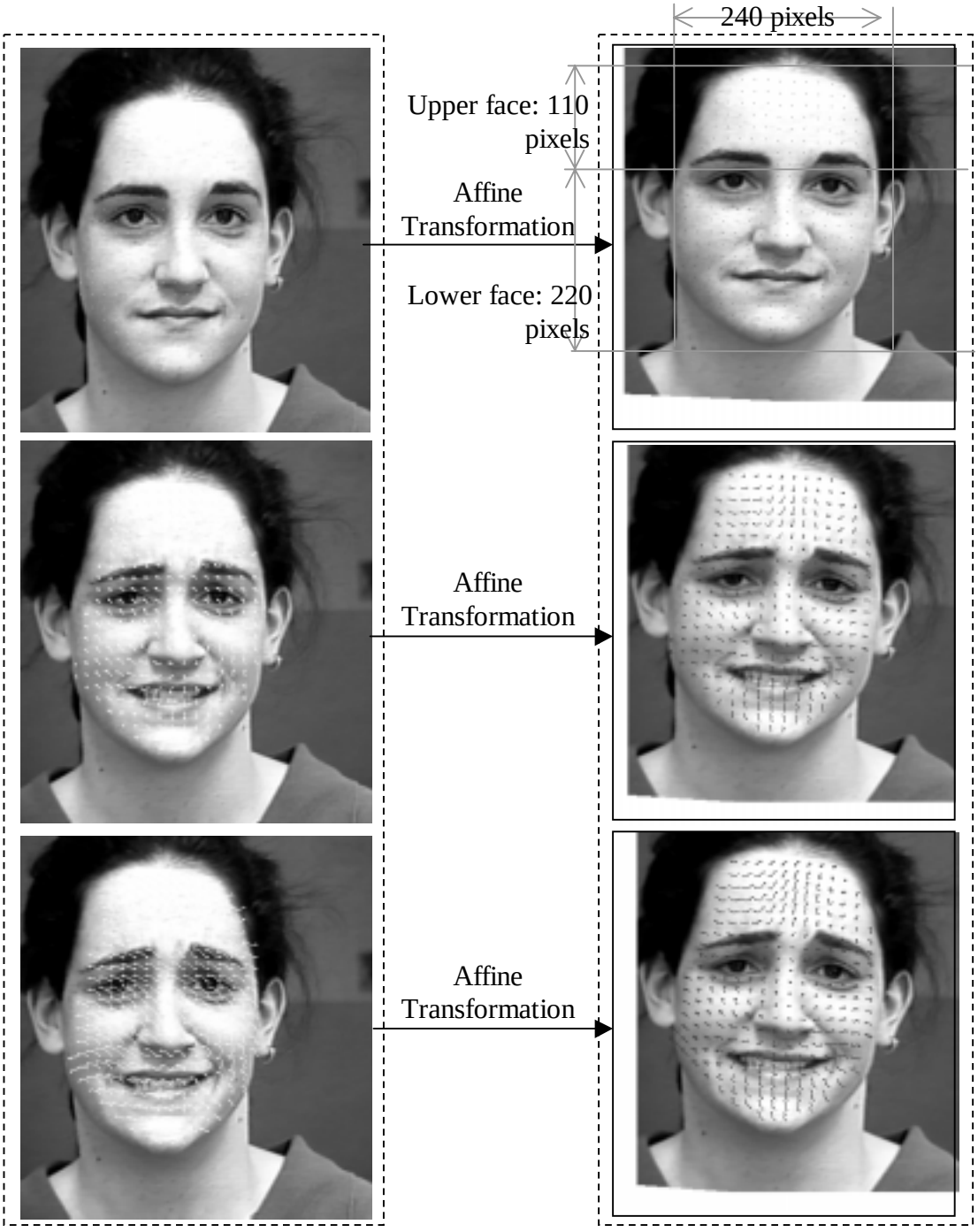


Figure 26 Dense flow normalization using affine transformation: (a) includes both the rigid head motion in upward and leftward direction and non-rigid facial expression, and (b) eliminates the rigid head motion by using the affine transformation.

regions are independent. The size of the face model is 417 x 385 (row x column) pixels. The upper face region is 110 x 240 pixels and the lower face region is 220 x 240 pixels (Figure 26). It should be noted that each flow unit contains both horizontal flow and vertical flow components. The PCA is applied to each dense flow region, the horizontal dense-flow region and vertical dense-flow region, separately.

For PCA computation, initially let the normalized estimated dense flow, either horizontal or vertical component, in a region (either upper or lower face) of frame *i* be represented lexicographically by a normalized dense-flow vector $f_{i,}$

$$f_i = [p_{i,1}, p_{i,2}, ..., p_{i,n}, ..., p_{i,N}]^T, \qquad p_{i.n} = u_{i,n} \ \ or \ \ v_{i,n} \tag{4-28}$$

where

$$1 \le i \le M \qquad and \qquad 1 \le n \le N$$

Here, $p_{i,n}$ is the normalized optical flow in either horizontal or vertical direction ($u_{i,n}$ or $v_{i,n}$) at pixel *n* of frame *i,* and *N* is the total number of pixels of frame *i* (or in upper or lower face region). There are *X* different facial expressions and a total of *M* training frames. The number of frames for each facial expression sequence varies from 9 to 47 frames. The variance matrix *F* of all normalized dense-flow training frames is given by

$$\begin{aligned} F &= [F_1, F_2, ..., F_i, ..., F_M] \\ &= [f_1 - c, f_2 - c, ..., f_i - c, ..., f_M - c] \qquad 1 \le i \le M \end{aligned} \tag{4-29}$$

where

$$c = \frac{\sum_{i=1}^{M} f_i}{M} \tag{4-30}$$

c is the mean flow (Figure 27) and the size of *F* is *N* x *M*. The *N* x *N* covariance matrix *C* of all normalized dense flows is given by

$$C = F\,F^T \tag{4-31}$$

which will have *N* different *N*-dimensional eigenvectors E_i (called eigenflows because of the flow-based eigenspace) corresponding to *N* eigenvalues λ_i of *C* ranked in the descending order, $\lambda_1 \ge \lambda_2 \ge ... \ge \lambda_N$,

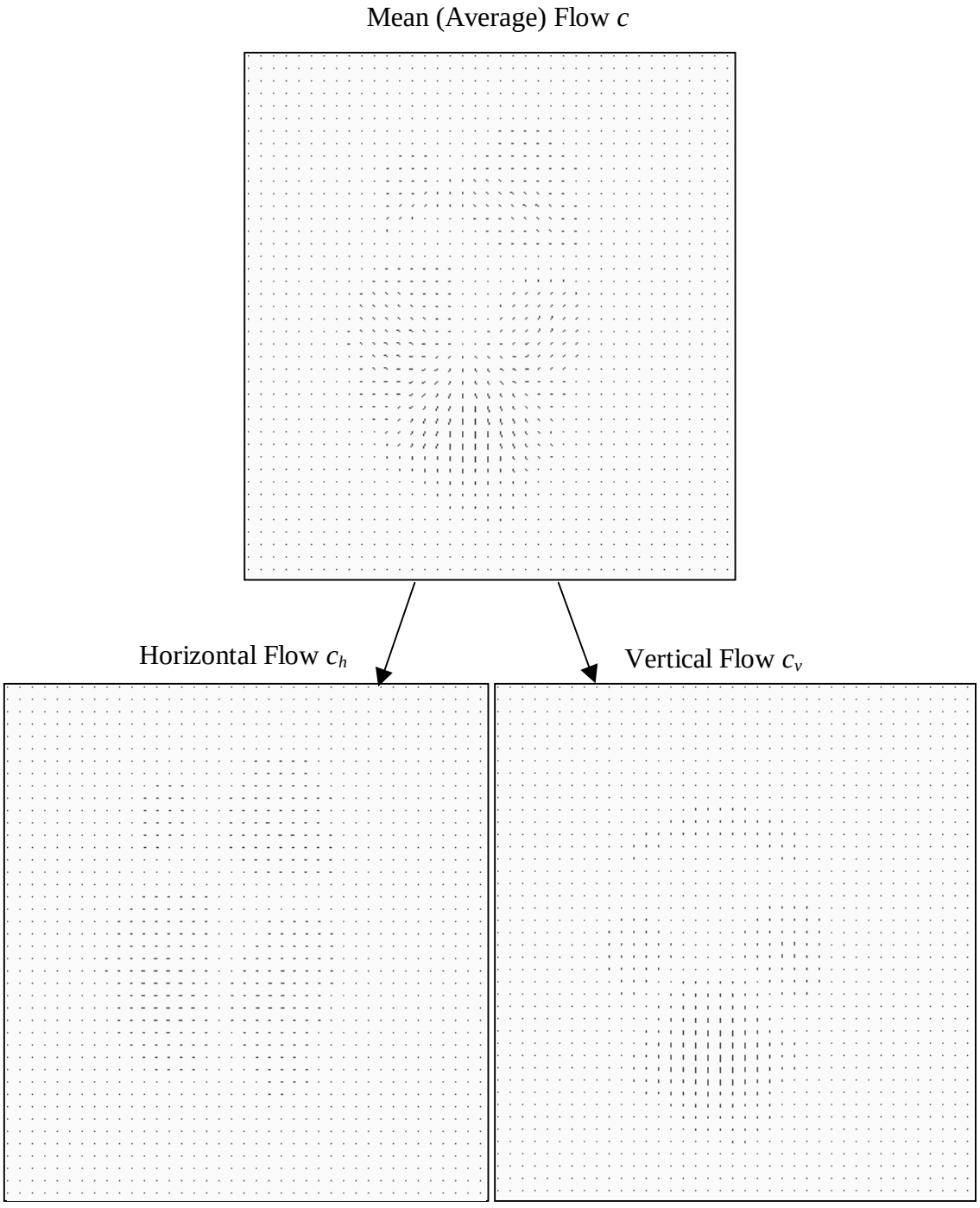


Figure 27 The mean (average) flow c is divided into horizontal flow c_h and vertical flow c_v for further processing by the principal component analysis (PCA).

$$C\,E_i = \lambda_i\,E_i, \qquad 1 \le i \le N \tag{4-32}$$

Generally, N is much larger than the total of training frames M, *i.e.* $N >> M$, where N is 110 x 240 pixels for the upper face region and 220 x 240 pixels for the lower face region, and M is 932 and 1212 training frames in the respective region. It will be impractical to compute eigenvalue λ_i and eigenvectors E_i directly from the N x N covariance matrix C. We will flow a more feasible computation approach described below.

When total number of dense-flow training frames M is far less than the number of dense flows N in the measured region, there are only M meaningful eigenvectors (eigenflows) associating with the non-zero eigenvalues of C. The remaining N - M eigenvectors correspond to zero eigenvalues. We can use a much more efficient method to obtain the M meaningful eigenvectors E_i by solving for the M-dimensional eigenvectors e_i and their corresponding eigenvalues $\tilde{\lambda}_i$ of an M x M covariance matrix $\tilde{C}$, where

$$\tilde{C} = F^T F \tag{4-33}$$

$$\tilde{C}e_i = \tilde{\lambda}_i e_i \qquad 1 \le i \le M \tag{4-34}$$

Multiplying both sides by F, gives

$$FF^T Fe_i = \tilde{\lambda}_i Fe_i \qquad 1 \le i \le M \tag{4-35}$$

then

$$CFe_i = \tilde{\lambda}_i Fe_i \qquad 1 \le i \le M \tag{4-36}$$

F x e_i are the first M eigenvectors of the covariance matrix C, leading to

$$\begin{aligned} E_i &= Fe_i \\ &= \sum_{j=1}^{M} e_{i,j} F_j \\ &= \sum_{j=1}^{M} e_{i,j} (f_j - c) \quad 1 \le i \le M \end{aligned} \tag{4-37}$$

This computation greatly reduces the order of time complexity from N-dimensional dense flows of each measured region to M dense-flow training frames where $M << N$.

We can reconstruct exactly the original dense flows of each measured region by a liner combination of all M eigenflows E_i. But, in general, using a small number M' ($M' < M$) of significant eigenflows that correspond to M' largest eigenvalues is adequate to reconstruct a good approximation of the original dense flow in each measured region without losing significant feature characteristics. The eigenflows (eigenvectors) with the largest eigenvalues represent the most significant characteristics in their corresponding dimensions of the eigenspace, where the variances of measured dense flows are bigger in terms of correlation. We rank the eigenflows according to the ranking of their corresponding eigenvalues, which contain the most useful information about the variational characteristics among the training dense flows. Furthermore, if the motion variation (deviation) among the training regions is large, we need a large number M' ($M' < M$) of eigenflows to accurately approximate the original dense flows. If the variation of motion among training regions is small, then a very small number M' ($M' << M$) of eigenflows is adequate to approximate the original dense flows. To determine the number M' of eigenflows needed to represent adequately the primary characteristics of the original dense flows in the region, we may use an information criterion such that

$$R = \frac{\sum_{i=1}^{M'} \tilde{\lambda}_i}{\sum_{i=1}^{M} \tilde{\lambda}_i} \geq T \quad where \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \ldots \geq \tilde{\lambda}_{M'} \geq \ldots \geq \tilde{\lambda}_M \tag{4-38}$$

and

$$R \ ln \ R \qquad \text{is the representative entropy} \tag{4-39}$$

where T is a threshold close to unity. The linear combination (the weighted sum) of the M' eigenflows is sufficient to reconstruct accurately the significant characteristics of each original dense flows in the region. The M' N-dimensional eigenflows E_i are computed by

$$E_i = \sum_{j=1}^{M} e_{i,j} F_j$$

$$= \sum_{j=1}^{M} e_{i,j} (f_j - c) \qquad 1 \leq i \leq M' \tag{4-40}$$

We project each variational dense-flow F_i (N dimensions) of facial expression sequence into the eigenflow space by taking its inner product with each eigenflow E_j (N dimensions) of the set $E = [E_1,E_2,...,E_{M'}]$ in the M'-dimensional subspace to produce the M'-dimensional weight vector W_i. Each element $w_{i,j}$ of the weight vector W_i is the projected component of F_i at the eigenflow dimension E_j in the eigenspace. Any N-dimensional normalized dense-flow f_i can be represented by its corresponding M'-dimensional weight vector W_i

$$W_i = [w_{i,1}, w_{i,2}, ..., w_{i,M'}]^T \qquad \textit{where } 1 \le i \le M \tag{4-41}$$

and

$$\begin{aligned} w_{i,j} &= E_j F_i^T \\ &= E_j (f_i - c)^T \qquad \textit{where } 1 \le i \le M \textit{ and } 1 \le j \le M' \end{aligned} \tag{4-42}$$

by projecting the N-dimensional variational region F_i to the M'-dimensional eigenspace ($M'<<N$).

4.4 Data Quantization and Conversion for the Recognition System

Figure 28 shows the flow image which will project to the flow-based eigenspace for PCA process. PCA enables us to convert dense flows of each image (or region) to a low-dimensional representative weight vector for input to the recognition system. For the 110 x 240-pixel upper face region, 10 most significant eigenflows are chosen and for the 220 x 240-pixel lower face region, 15 most significant eigenflows are chosen. The decision for the choice of M' is that R should be greater than or equal to T where T is set to be 0.9 (refer to equation (4-38)), which led us to select M'=10 for the upper facial expressions and M'=15 for the lower facial expressions as shown in Figure 29. The same of eigenflows should be used for both horizontal flow and vertical flow. These eigenflows are the eigenvectors corresponding to the 10 (and 15) largest eigenvalues of the 932 x 932- (and 1212 x 1212-) covariance matrix constructed by 932 (and 1212) dense-flow

training frames from 45 (and 60) training image sequences for the upper (and lower) face regions (Figure 29). The compression rate is 93:1 as shown in Figure 29.a (and 80:1 as shown in Figure 29.b). Because the variation (deviation) among the training data of the upper facial "expression units" is smaller than that of the lower facial "expression units," it is expected that the number of eigenflows used for representing the upper facial "expression units" (M' = 10) is fewer than that used for representing the lower facial "expression units" (M' = 15). As shown by the later experiments, this choice leads to good performance on overall recognition rate of 92% (Figure 29).

The dense flow at each frame region of an expression sequence is projected onto the flow-based eigenspace by taking its inner product with each element of the respective eigenflow set, producing a 10- (and 15-) dimensional weight vector for the upper (and lower) facial expressions as shown in Figure 30. The 10- (and 15-) dimensional horizontal-flow weight vector and 10- (and 15-) dimensional vertical-flow weigh vector are concatenated to form a 20- (and 30-) dimensional weight vector for each dense-flow region. After vector quantization, the concatenated weight-vector sequence is converted into a symbol sequence. Table 6 shows sample symbol sequences for nine facial expressions under consideration. Such symbol sequences are used as inputs to the HMMs of upper facial expressions and lower facial expressions, respectively, for automatic recognition.

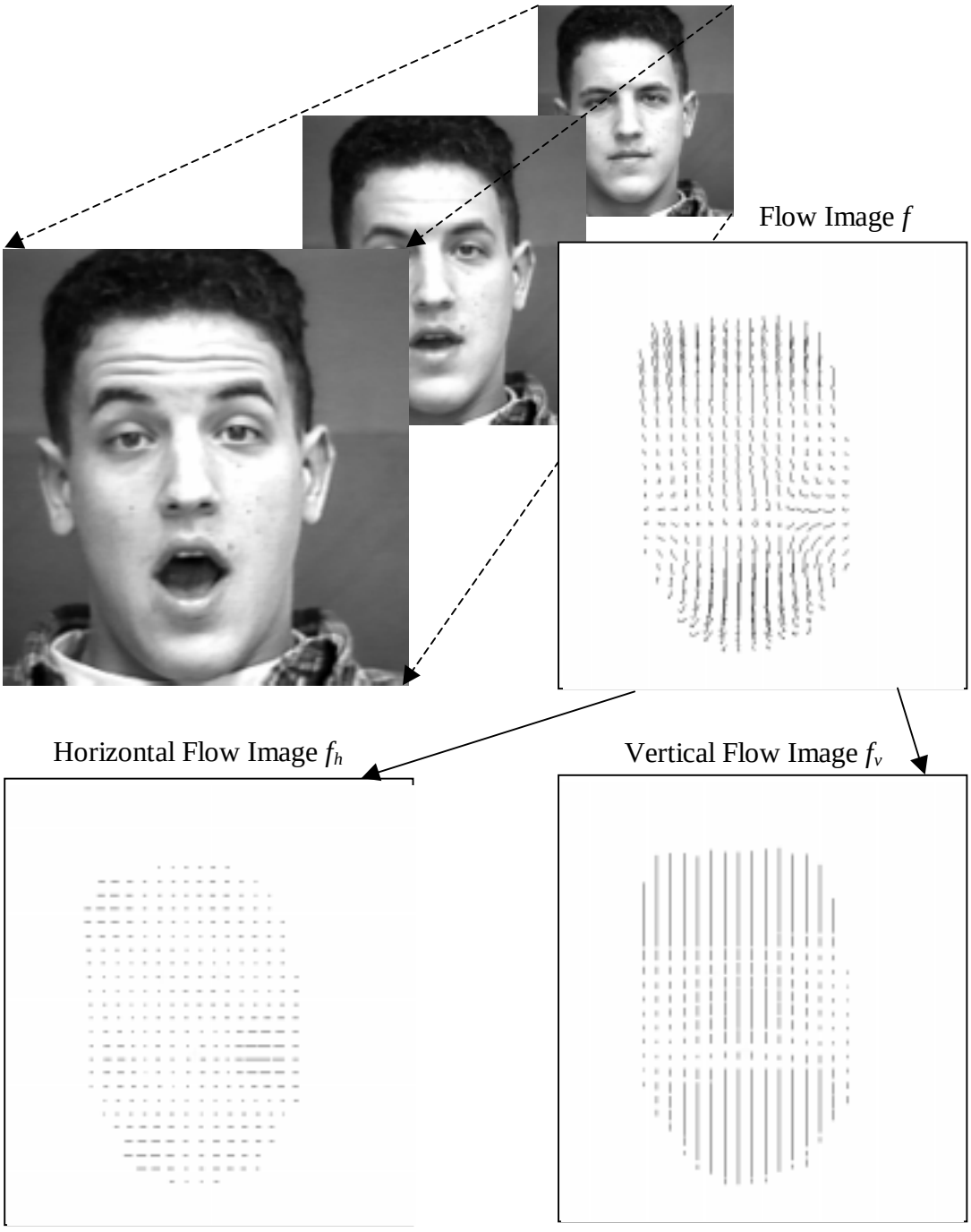
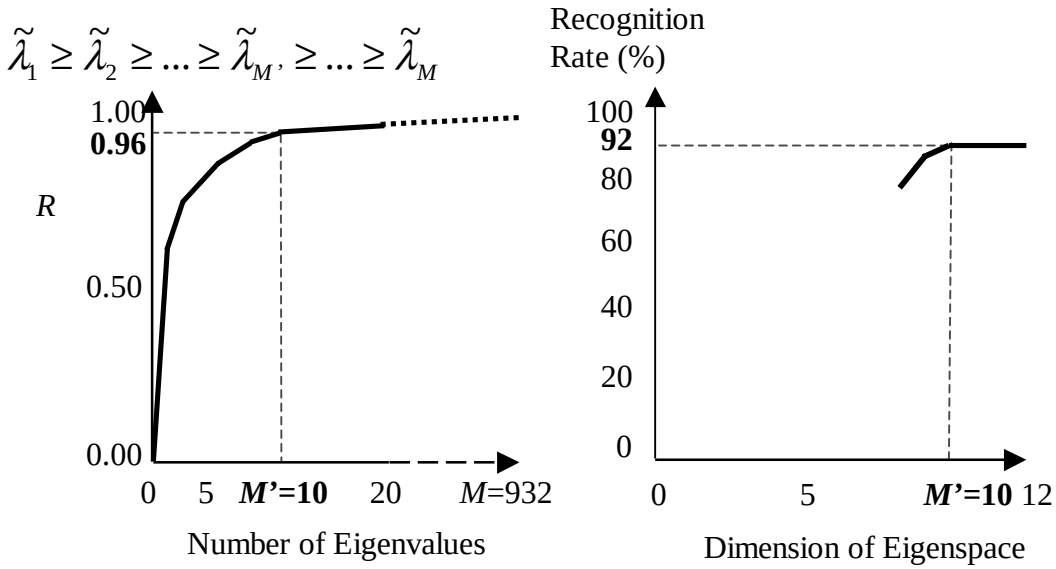
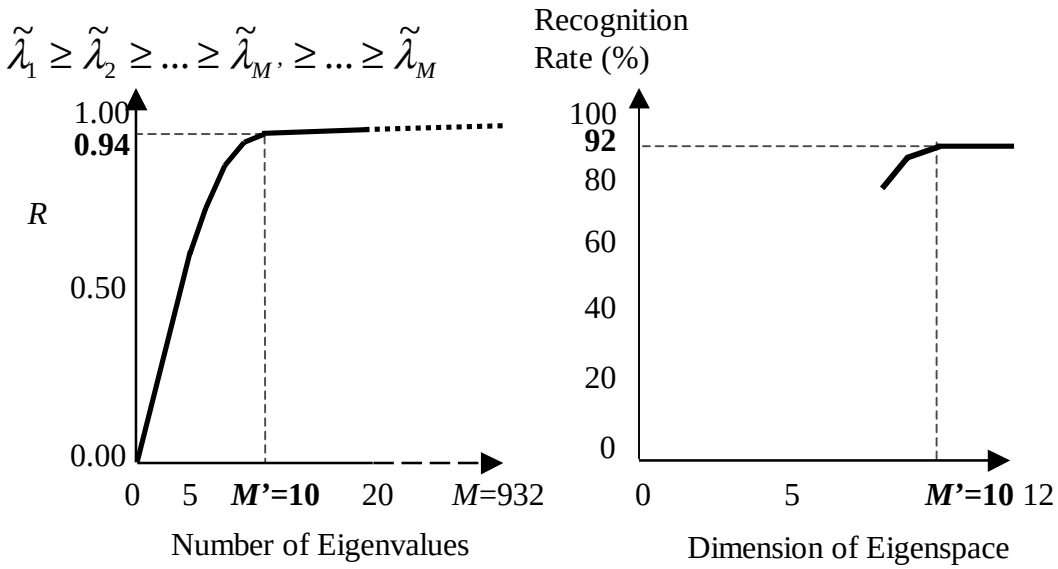


Figure 28 Each dense flow image is divided into horizontal and vertical flow images for the principal component analysis (PCA).

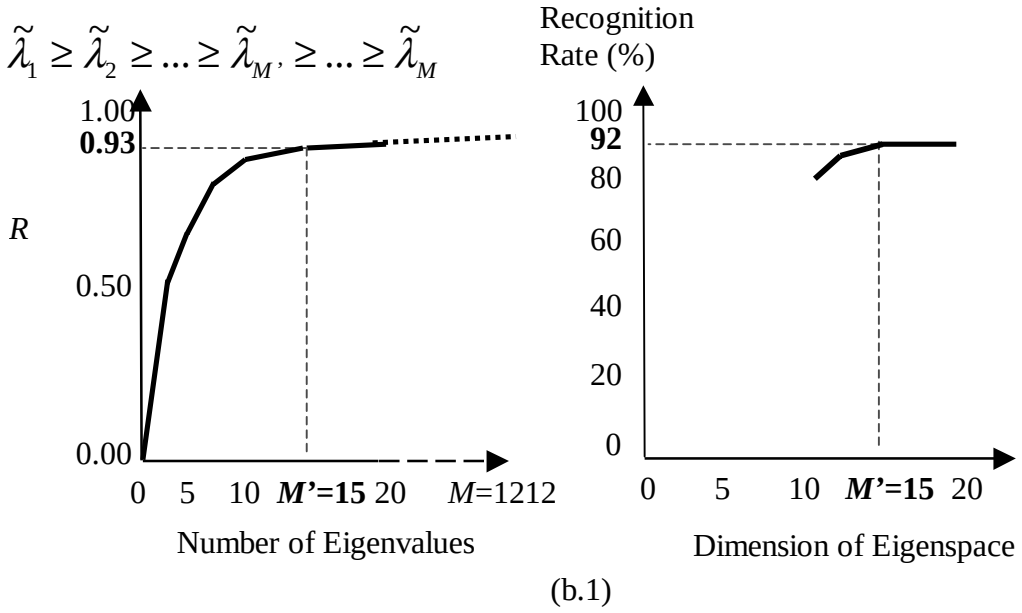


(a.1)

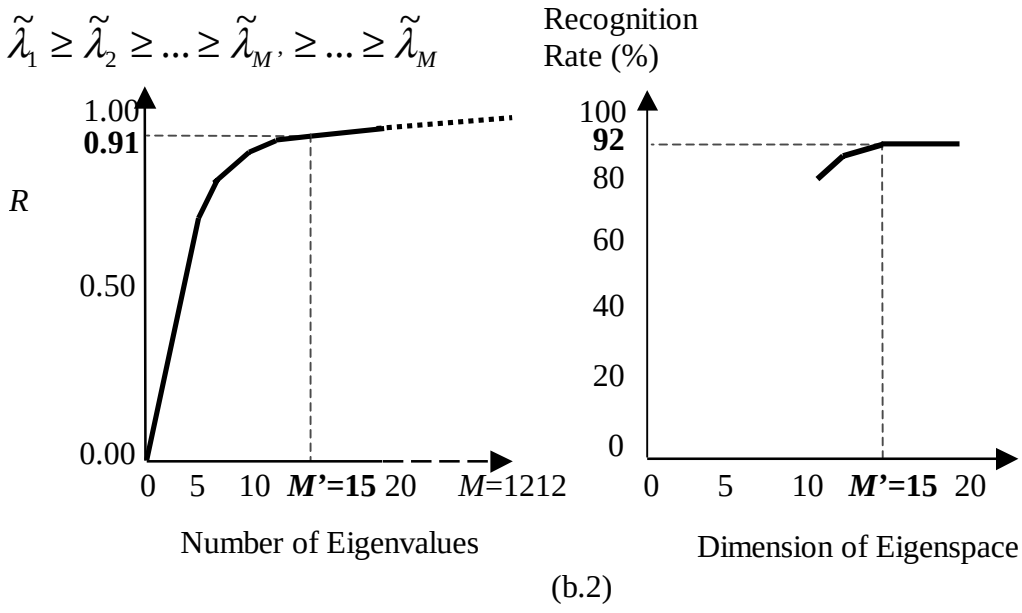


(a.2)

Figure 29.a Computation of eigenflow (eigenvector) number for the upper facial expressions: (a.1) is for the horizontal flow and (a.2) is for the vertical flow. The compression rate is 93:1 (932:10) and from which the recognition rate is 92% based on 45 training and 60 testing image sequences.

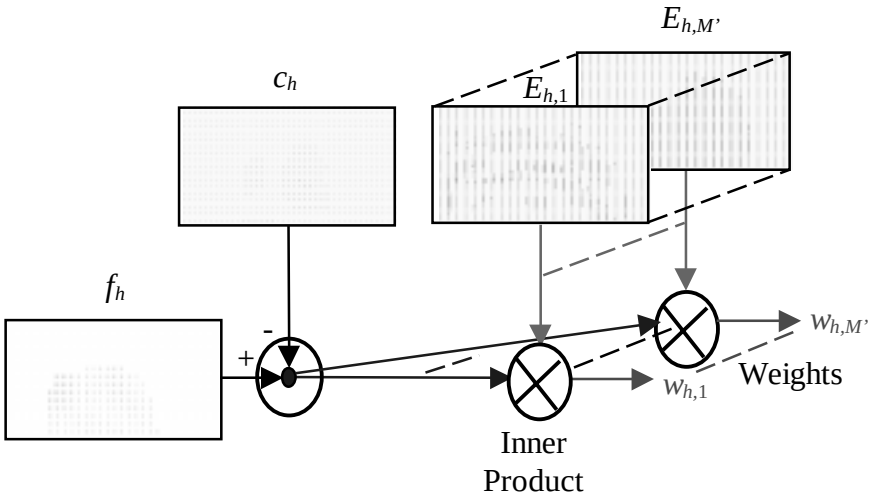


(b.1)

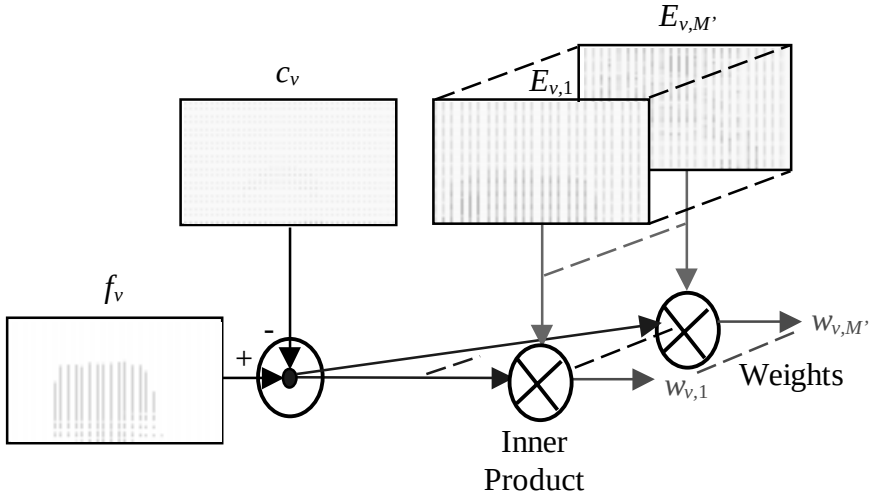


(b.2)

Figure 29.b Computation of eigenflow (eigenvector) number for the lower facial expressions: (b.1) is for the horizontal flow and (b.2) is for the vertical flow. The compression rate is 80:1 (1212:15) and from which the recognition rate is 92% based on 60 training and 90 testing image sequences.



(a)



(b)

Figure 30 Principal component analysis (PCA) for (a) horizontal flow weight vector and (b) vertical flow weight vector for the upper facial expressions (M' = 10).

Table 6 Sample symbol sequences for three upper facial expressions and six lower facial expressions under consideration.

AUs	**Dense Flows with PCA: Upper Facial Expressions (Symbol Sequence)**
4	3 3 3 3 *5* 5 5 5 5 *5* 5 5
1+4	3 3 3 3 *3* 7 7 7 7 *7* 7
1+2	3 3 3 6 *6* 2 2 2 2 *2* 2 2 2
AUs	**Dense Flow with PCA: Lower Facial Expressions (Symbol Sequence)**
12	6 6 6 6 *6* 6 9 9 1 *1* 1
6+12+25	6 6 9 9 *3* 3 3 3 3 *3* 3
20+25	6 6 6 11 *13* 13 13 13 13 *13*
9+17	6 6 6 7 *7* 7 7 2 2 *2* 2
15+17	6 6 6 6 *6* 14 14 14 14 *14* 14 14
17+23+24	6 6 6 12 *12* 4 4 4 4

4.5 Correlation and Distance in Eigenspace

The sum of squared differences (SSD) between two dense-flow frame regions f_i and f_j is given by

$$\begin{aligned} \left\| f_i - f_j \right\|^2 &= (f_i - f_j)^T (f_i - f_j) \\ &= (f_i^T f_i + f_j^T f_j) - 2 f_i^T f_j \end{aligned} \tag{4-43}$$

where $f_i^T f_j$ represents the correlation between f_i and f_j. The distance SSD becomes smaller when the correlation between the two flows is stronger. Each dense flow f_i in frame i can be represented by its representative weight vector W_i in the M'-dimensional eigenflow space. It can be reconstructed by a linear combination of M' eigenflows E_j.

$$f_i \approx \sum_{j=1}^{M'} w_{i,j} E_j + c \tag{4-44}$$

where $w_{i,j}$ is the jth element of the weight vector W_i flow f_i. Since

$$\begin{aligned} \left\| f_i - f_j \right\|^2 &\approx \left\| \left(\sum_{k=1}^{M'} w_{i,k} E_k + c \right) - \left(\sum_{k=1}^{M'} w_{j,k} E_k + c \right) \right\|^2 \\ &= \left\| \sum_{k=1}^{M'} (w_{i,k} - w_{j,k}) E_k \right\|^2 \\ &= \sum_{k=1}^{M'} \sum_{l=1}^{M'} E_k^T E_l * (w_{i,k} - w_{j,l})^2 \\ &= \begin{cases} \sum_{k=1}^{M'} (w_{i,k} - w_{j,k})^2 & \text{if} \quad k = l \\ 0 & \text{otherwise} \ (\text{since } E_k \perp E_l) \end{cases} \end{aligned} \tag{4-45}$$

This can also be expressed as

$$\left\| f_i - f_j \right\|^2 \approx \left\| W_i - W_j \right\|^2 \tag{4-46}$$

Thus, we can estimate the expression similarity between two dense flows f_i and f_j by measuring the distance between their representative weight vectors W_i and W_j in the M'-

dimensional eigenspace. Smaller distance indicates greater correlation or similarity between two dense flows. This can be used in expression intensity estimation as described below.

4.6 Expression Intensity Estimation

The expression intensity of any frame in a sequence may be estimated by using the correlation property of the PCA. The expression intensity of individual frames in each training image sequence is determined a priori by experts, beginning from the neutral expression (expression intensity: 0.0) to the peak expression (expression intensity: 1.0). The length of each image sequence varies from 9 to 47 frames. Each frame in the training sequence has its representative weight vector. So, the relationship between weight vector and expression intensity can be established.

After a test facial expression sequence is recognized, the expression intensity of any frame f_i in the sequence can be estimated as follows. Consider the distances between its representative weight vector W_i and all weight vectors W_k^0 in a training sequence whose expression intensity values are known, the minimum distance between W_i and W_k^0 indicates the maximum correlation or expression similarity between the two. Then, the expression intensity of frame i in the testing sequence will be estimated to have the same value as that of W_k^0 training data (Figure 31).

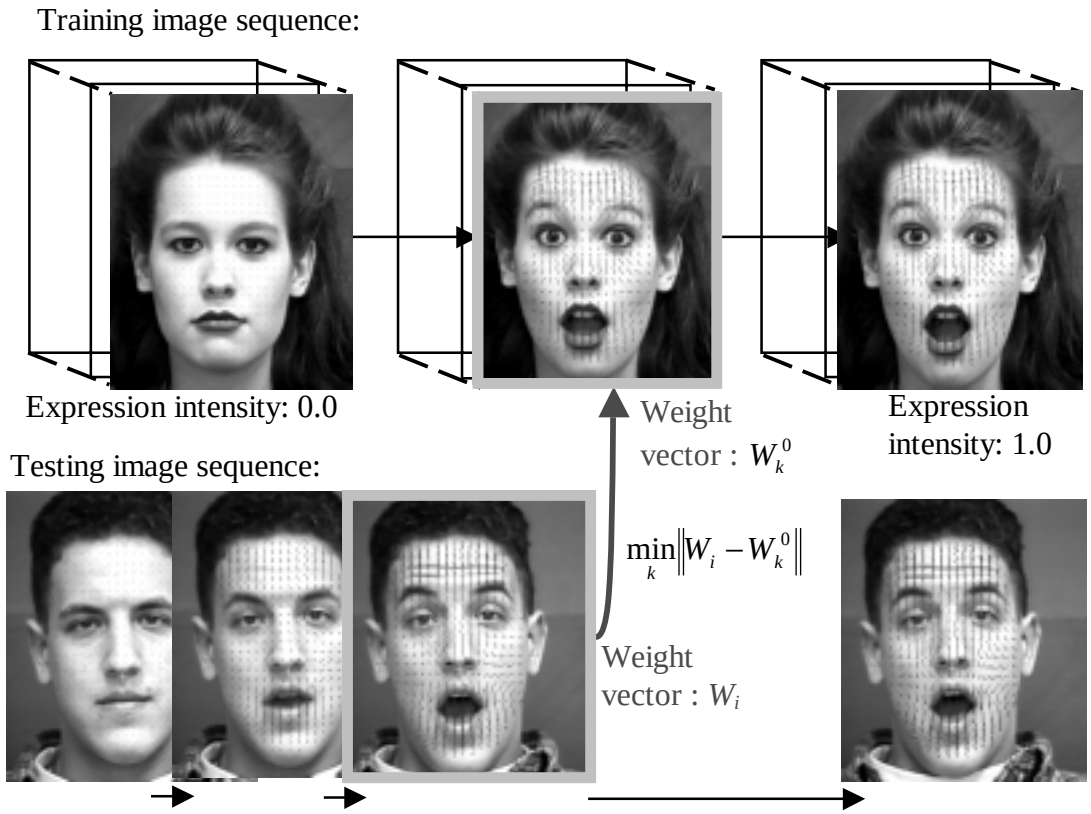


Figure 31 Expression intensity matching by seeking the minimum distance between the weight vector of the testing frame and the weight vectors of all frames in a training sequence, whose expression intensity values are known. Each weight vector of the training image corresponds to a given expression intensity value.

5.0 HIGH GRADIENT COMPONENT ANALYSIS

Facial motion produces transient darkened skin-color lines or edges perpendicular to the motion direction of the activated muscle. Those darkened lines or edges are called furrows or wrinkles. The facial action produces the motion position, shape, length and gray-value changes of these furrows in the face image and which are strongly associated with different meanings of the facial expression. Extracting (segmenting) and realizing the motion of those high gradient components (*i.e.*, furrows) may provide a very important source for recognizing facial expressions.

5.1 High Gradient Component Detection in the Spatial Domain

The shapes of furrows in the face image contain horizontal, vertical and/or diagonal directions of lines or arched curves (Figure 32). To extract the high gradient component at pixel (x,y) from the face image (in the spatial domain) I at time t, we use horizontal, vertical and diagonal line or edge detectors, L_x, L_y, L_{xy}, respectively.

$$\frac{\partial I(x,y,t)}{\partial x} = L_x \otimes I(x,y,t) = D_x(x,y,t) \tag{5-1}$$

$$\frac{\partial I(x,y,t)}{\partial y} = L_y \otimes I(x,y,t) = D_y(x,y,t) \tag{5-2}$$

$$\frac{\partial I(x,y,t)}{\partial (xy)} = L_{xy} \otimes I(x,y,t) = D_{xy}(x,y,t) \tag{5-3}$$

and

$$D(x,y,t) = \left\{ D_x(x,y,t), D_y(x,y,t), D_{xy}(x,y,t) \right\} \tag{5-4}$$

where $I(x,y,t)$ is the image gray value at position (x,y) and frame t, and $\otimes$ denotes convolution. $D_x(x,y,t)$, $D_y(x,y,t)$ and $D_{xy}(x,y,t)$ are gradient intensities of the high gradient

components at pixel (x,y) in the horizontal, vertical and diagonal directions, respectively. $D(x,y,t)$ is used as the general term of the gradient intensities including $D_x(x,y,t)$, $D_y(x,y,t)$ and $D_{xy}(x,y,t)$.

Before normalization of each 417 x 385-pixel image using affine transformation by three feature points (the medial canthus of both eyes and the uppermost point on the philtrum), a 5 x 5 Gaussian filter is used to smooth the image. For the upper face expression (Figure 32.a), a 3 x 5 (row x column) horizontal- and a 5 x 3 vertical-line detectors are used to detect horizontal lines (*i.e.*, high gradient components in the vertical direction) and vertical lines in the forehead region, around the eye region, and between brows or eyes for AU4, AU1+4 and AU1+2 expressions. Two 5 x 5 diagonal-line detectors are used to detect 45-degree and 135-degree diagonal furrows at the forehead region during AU1+4 expression. For the lower face expressions, two 5 x 5 diagonal-line detectors are used to detect 45-degree and 135-degree diagonal lines along the nasolabial furrow (Figure 32.b). Two 3 x 3 edge detectors are used to detect high gradient components around the lips and on the chin region by thresholding their magnitudes (Figure 32.b and 32.c). The components filtered out (or thresholded out) by the line or edge detectors are set to a value of zero.

5.2 High Gradient Component Detection in the Spatio-Temporal Domain

To verify the detected high gradient component $D(x,y,t)$ which is produced by transient skin or feature deformations and not a permanent characteristic of the individual's face (Figure 33), it is necessary to consider its the temporal gradient.

$$\frac{\partial D(x,y,t)}{\partial t} = D(x,y,t) - D(x,y,t-1) = \Gamma_t(x,y) \qquad (5\text{-}5)$$

where $D(x,y,t)$ and $D(x,y,t\text{-}1)$ are the gradient intensities of the detected high gradient components at pixel (x,y) and frames t and t-1 in the spatial domain, respectively. $\Gamma_t(x,y)$ is the temporal gradient intensity between gradient intensities at frames t and t-1 at pixel

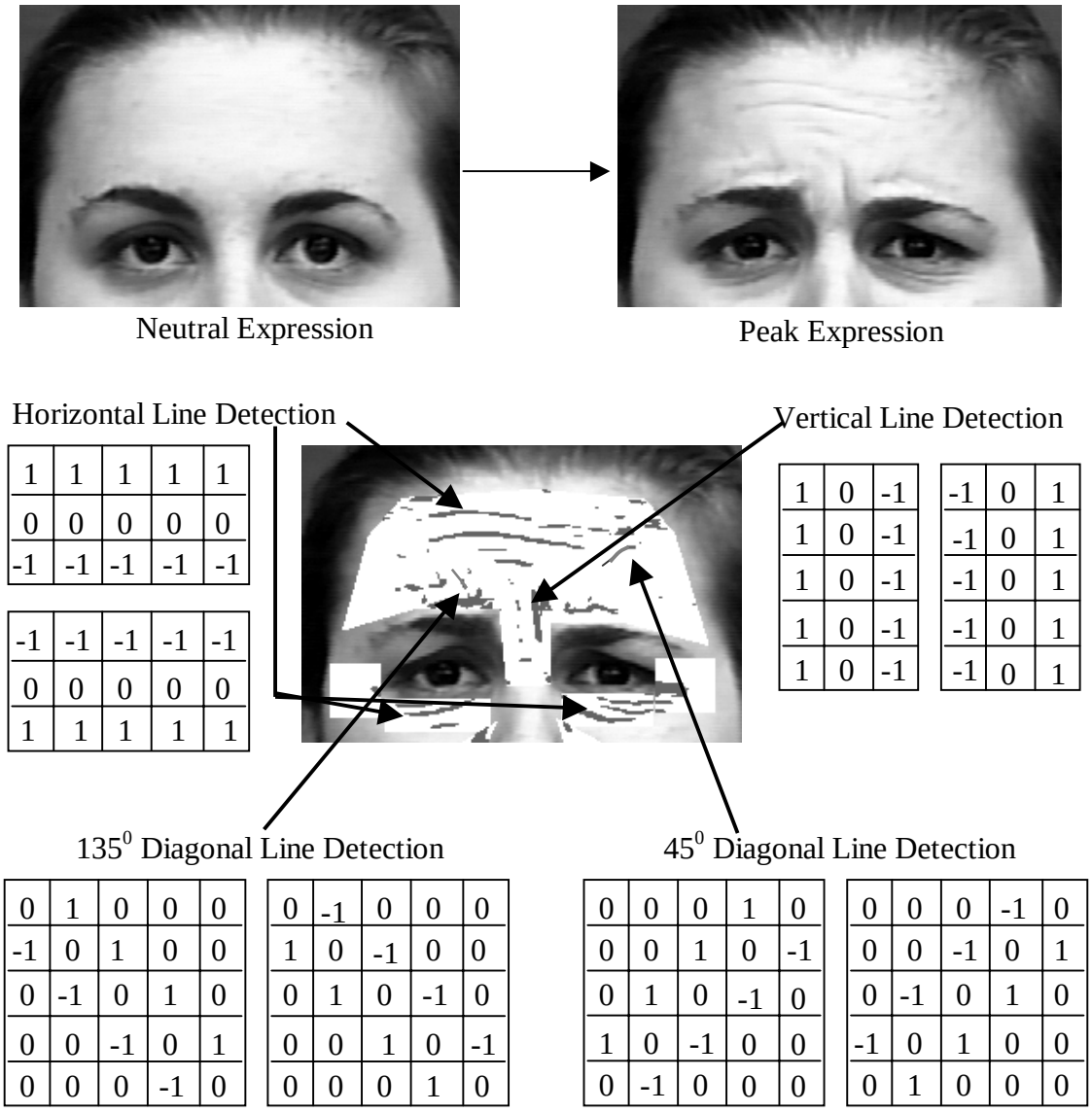


Figure 32.a High gradient component (furrow) detection for the forehead and eye regions.

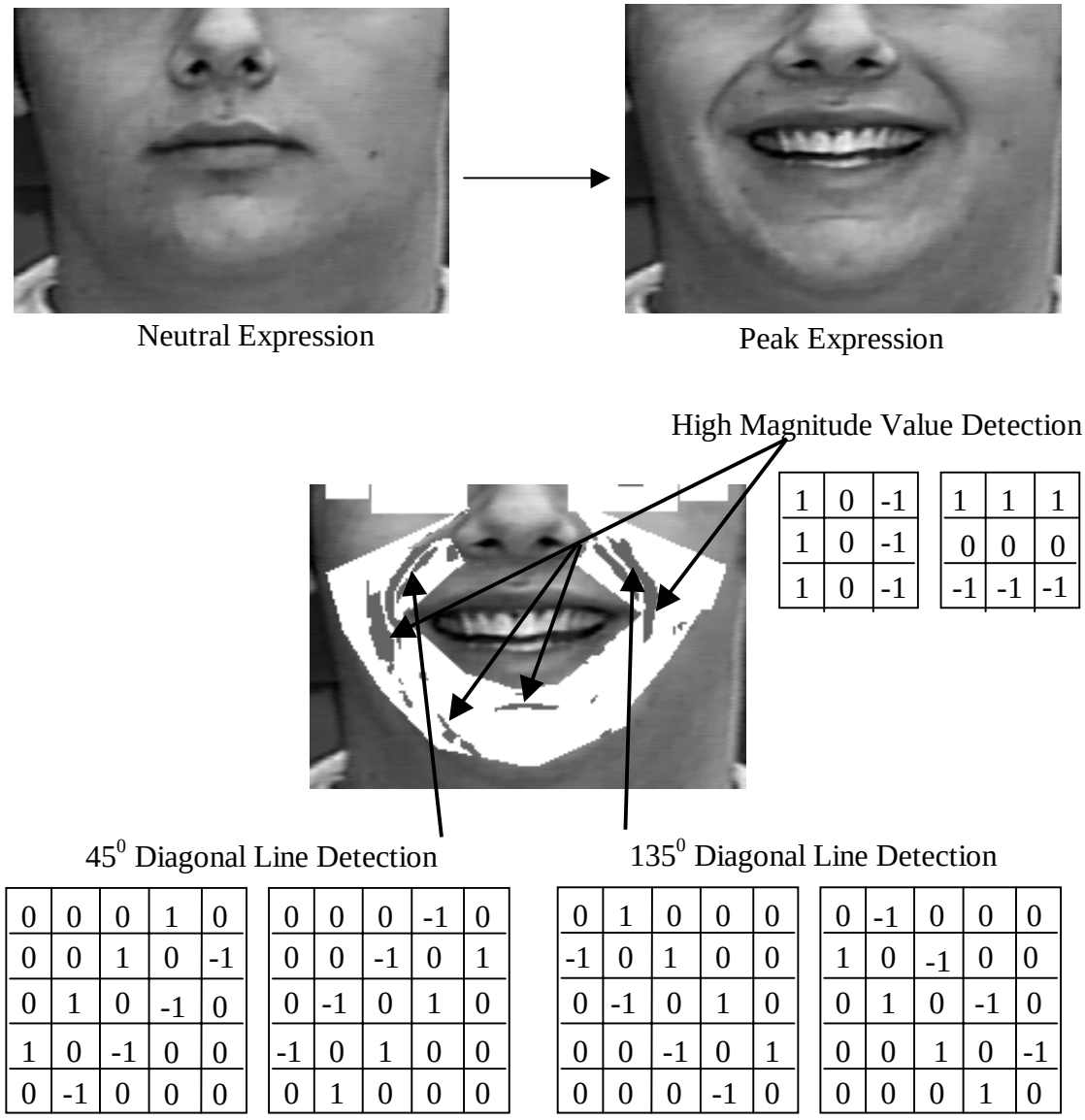


Figure 32.b High gradient component (furrow) detection for the mouth, cheek, and chin regions.

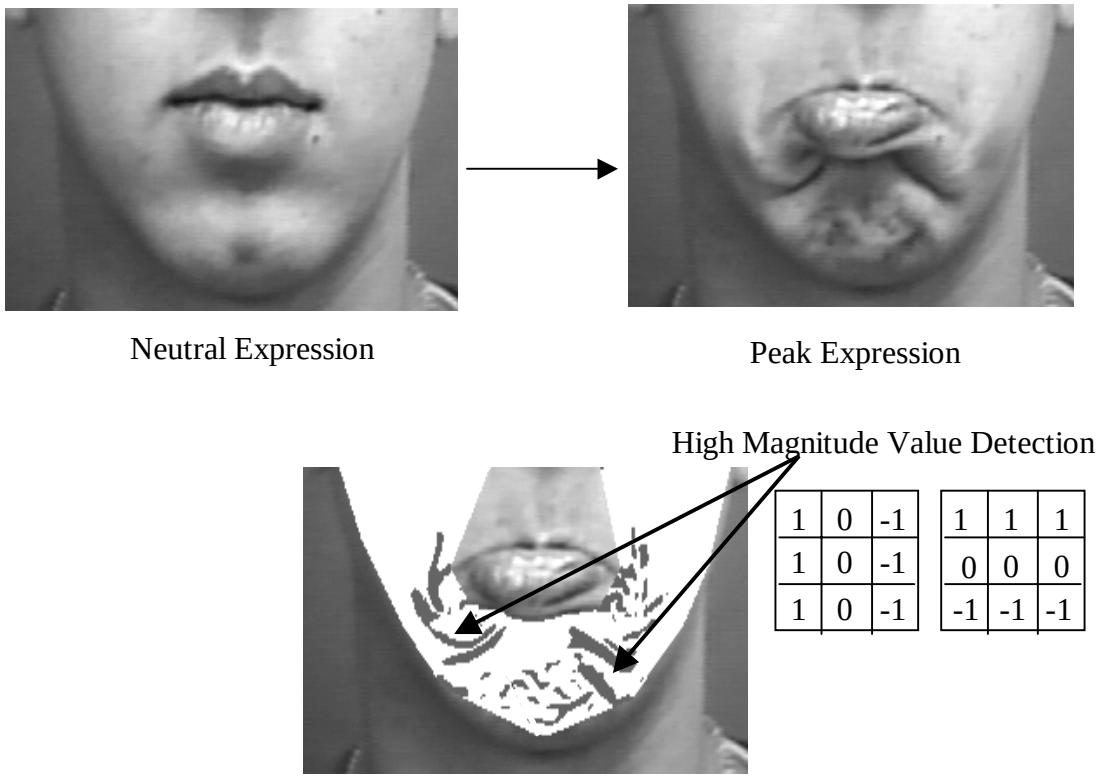


Figure 32.c High gradient component (furrow) detection for the chin region.

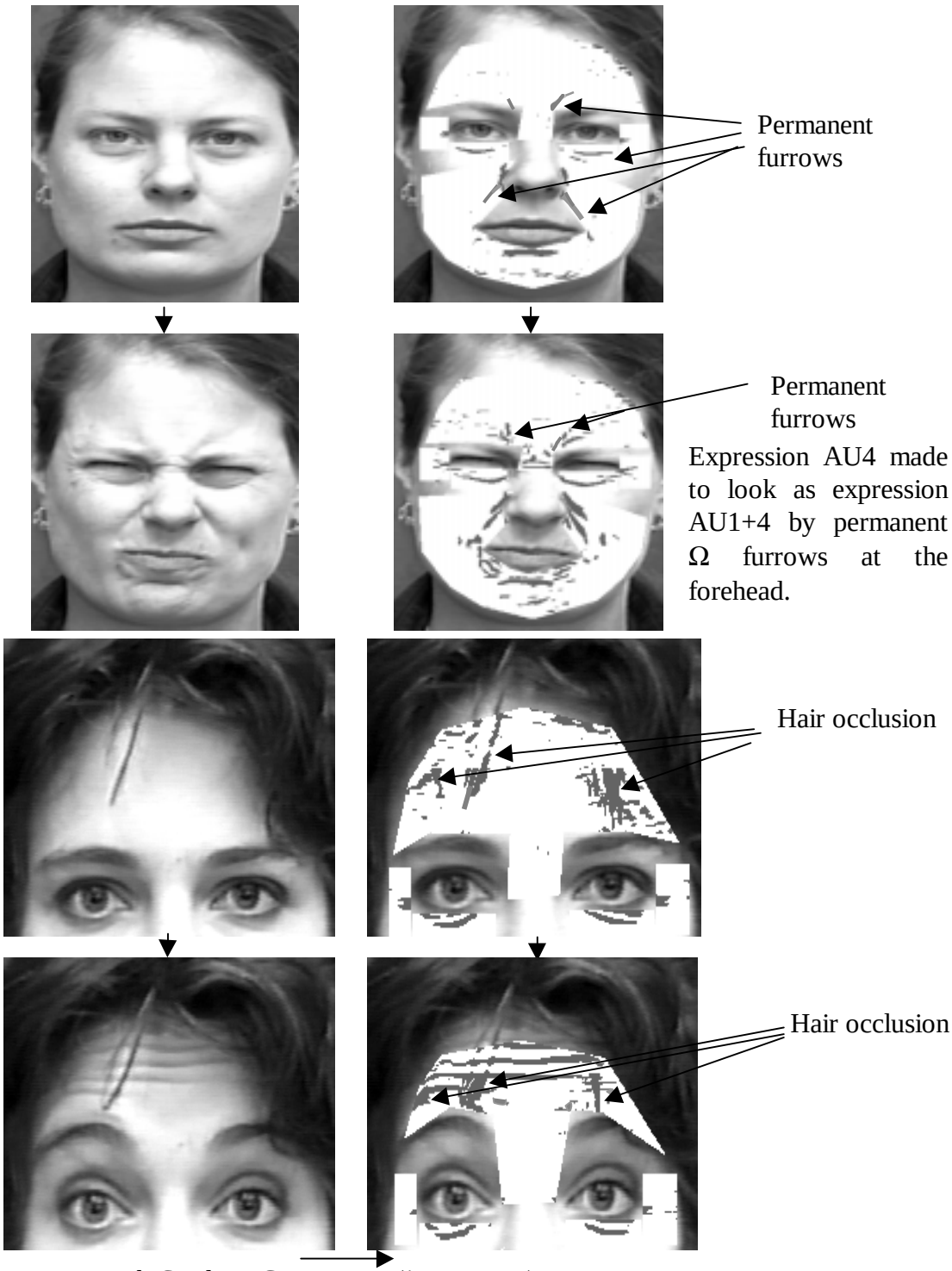


Figure 33 Permanent furrows or hair occlusion.

(x,y) considering the spatio-temporal domain. Equation (5-5) is the method used for tracking high gradient components such as the motion lines or edges in the spatial and temporal domains.

In our current study, the gradient intensity of each detected high gradient component $D(x,y,t)$ at the current frame t and pixel (x,y) is compared with corresponding points within a 3 x 3 region of the first frame for each sequence.

$$\frac{\partial D^0(x,y,t)}{\partial t} = D(x,y,t) - D(x-\Delta x, y-\Delta y, 0) = \Gamma_t^0(x,y) \qquad (5\text{-}6)$$

where

$$-1 \le \Delta x \le 1 \quad \text{and} \quad -1 \le \Delta y \le 1$$

A 3 x 3 region is used in order to avoid the error of geometrical correspondence since affine transformation works well for close (but not exact) geometrical correspondence. If the absolute value of the difference in gradient intensity between these points is higher than the threshold value, it is considered a valid high gradient component produced by facial expression. All other high gradient components are ignored. In the former case, the high gradient component (pixel) is assigned a value of 1. In the latter case, the pixels are assigned a value of 0. An example of the procedure for extracting high gradient components in the spatio-temporal domain for the upper facial expression is shown in Figure 34. A gray value of 0 corresponds to black and 255 to white. Using this procedure, we also can remove the hair blocking the forehead region (Figure 35).

5.3 Morphology and Connected Component Labeling

In order to use line and edge detectors, the threshold is employed to segment the higher gradient components for the foreground furrows and the lower gradient components for the background. If the given threshold is too high, then it will filter out more significant components. If the threshold is given too low, then it will include more high gradient components with unnecessary noise (Figure 36). For line or edge detection,

it is very difficult to give a constant or even dynamic threshold to satisfy all conditions, especially in dealing with images containing the conditions of rigid and non-rigid motion such as images of facial expression whose gray values vary in lighting, ages and individuals. Younger subjects, especially infants, show smoother furrowing than older ones, and initial expressions show weaker furrowing than that of peak expressions for each sequence (Figure 37). To overcome this difficulty, further low level image processing is needed.

Because we do not want to lose any useful information of high gradient components, we give a low threshold for each furrow detection processing sequence (Figure 36). An erosion morphological transformation is used to eliminate the piece regions or the very short lines, thin the lines, or smooth the line boundary (Figure 38.a). The eliminated (either one or several) pixels are assumed as noise and introduced might be because of the low threshold. A dilation morphological transformation is then used to connect two end-to-end close but separated lines (Figure 38.a). Finally, we implement a connected components labeling (CC labeling) algorithm based on Haralick and Shapiro's [43] to label each cluster of connected high gradient components (Figure 38.b). This algorithm is based on the 8-connected component to link components of its 8 neighbors at the binary image (1 is the high gradient component, 0 is the background) and includes two processes: top-down and bottom-up processes with 4 steps (APPENDIX). If the number of detected high gradient components for each connected component cluster is less than 6 pixels, or the horizontal to vertical ratio for the horizontal line detection or the vertical to horizontal ratio for the vertical line detection is less than 5, then this cluster is assumed to be noise, not furrow, and will be removed (Figure 38.b).

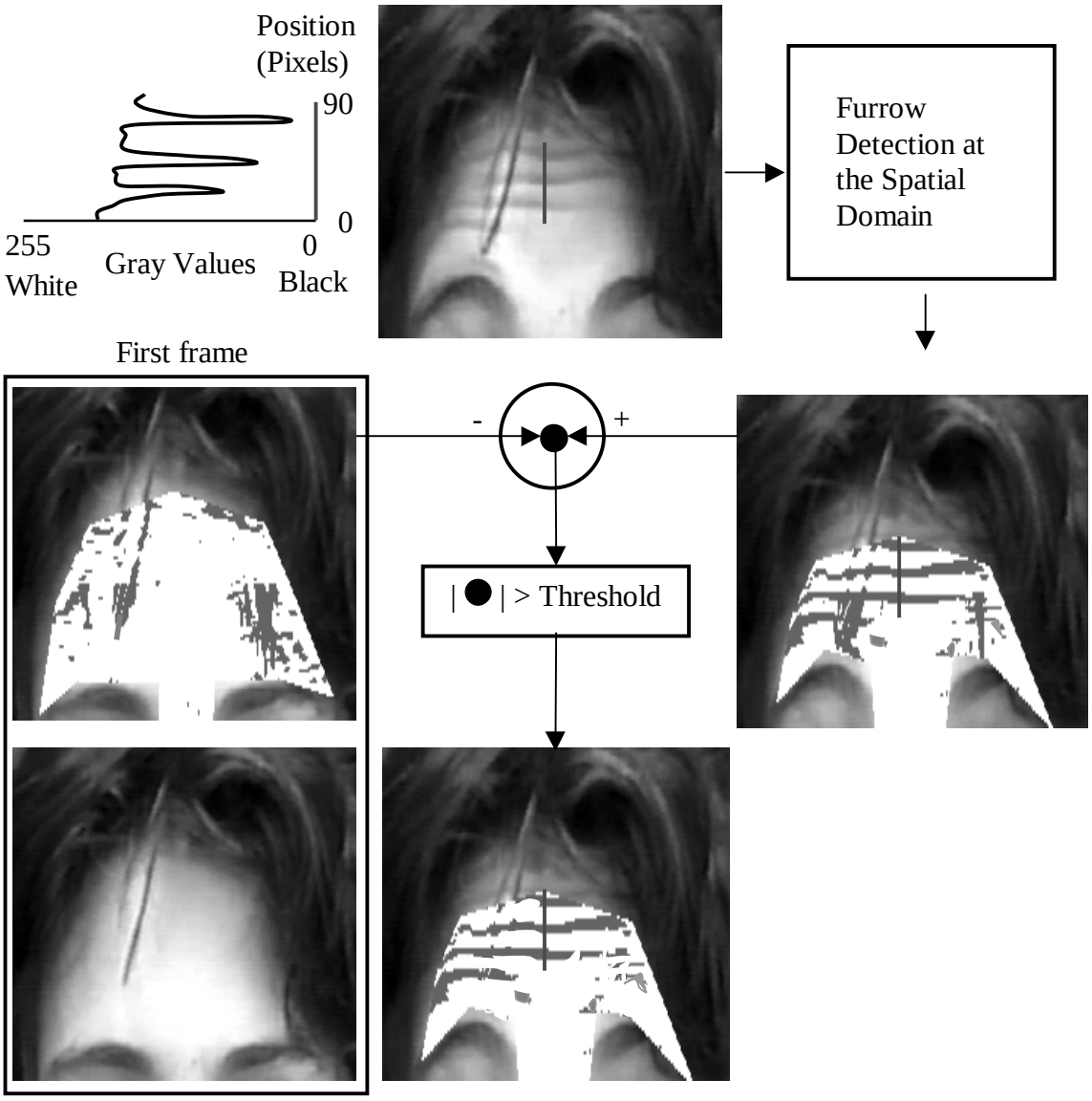
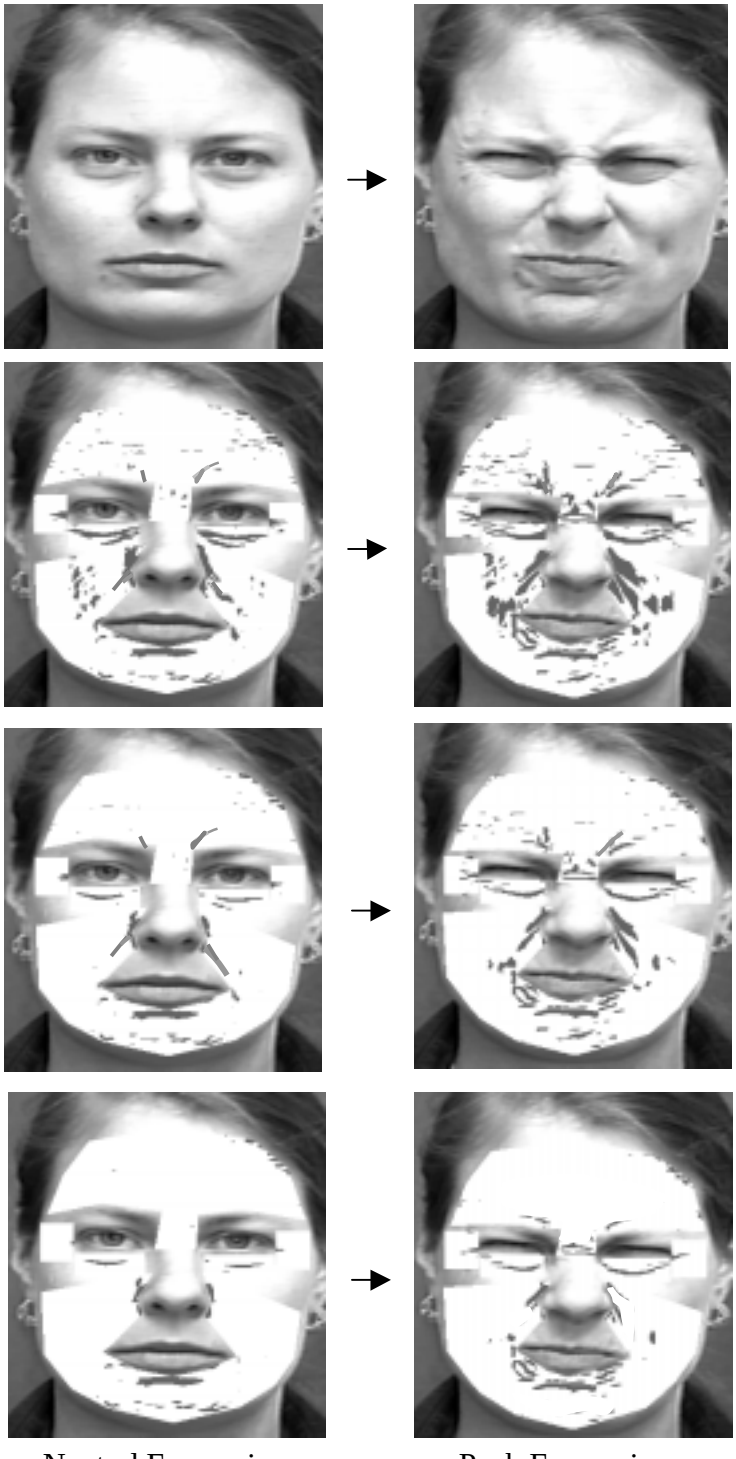


Figure 34 The procedure of the high gradient component analysis in the spatio-temporal domain, which can reduce the effect of the permanent high gradient components (furrows) and hair occlusion for the upper facial expression.

Figure 35 (a) Original gray value images. (b) High gradient component (furrow) detection in the spatial domain. (c) High gradient component analysis in the spatio-temporal domain.



(a) Gray value image

(b) High gradient component (furrow) detection with a very low threshold, which will include too many furrows or noise information.

(c) High gradient component (furrow) detection with a low threshold, which was used in our study.

(d) High gradient component (furrow) detection with a high threshold, which will lose some furrow information.

Neutral Expression Peak Expression

Figure 36.a High gradient component detection with different constant threshold values.

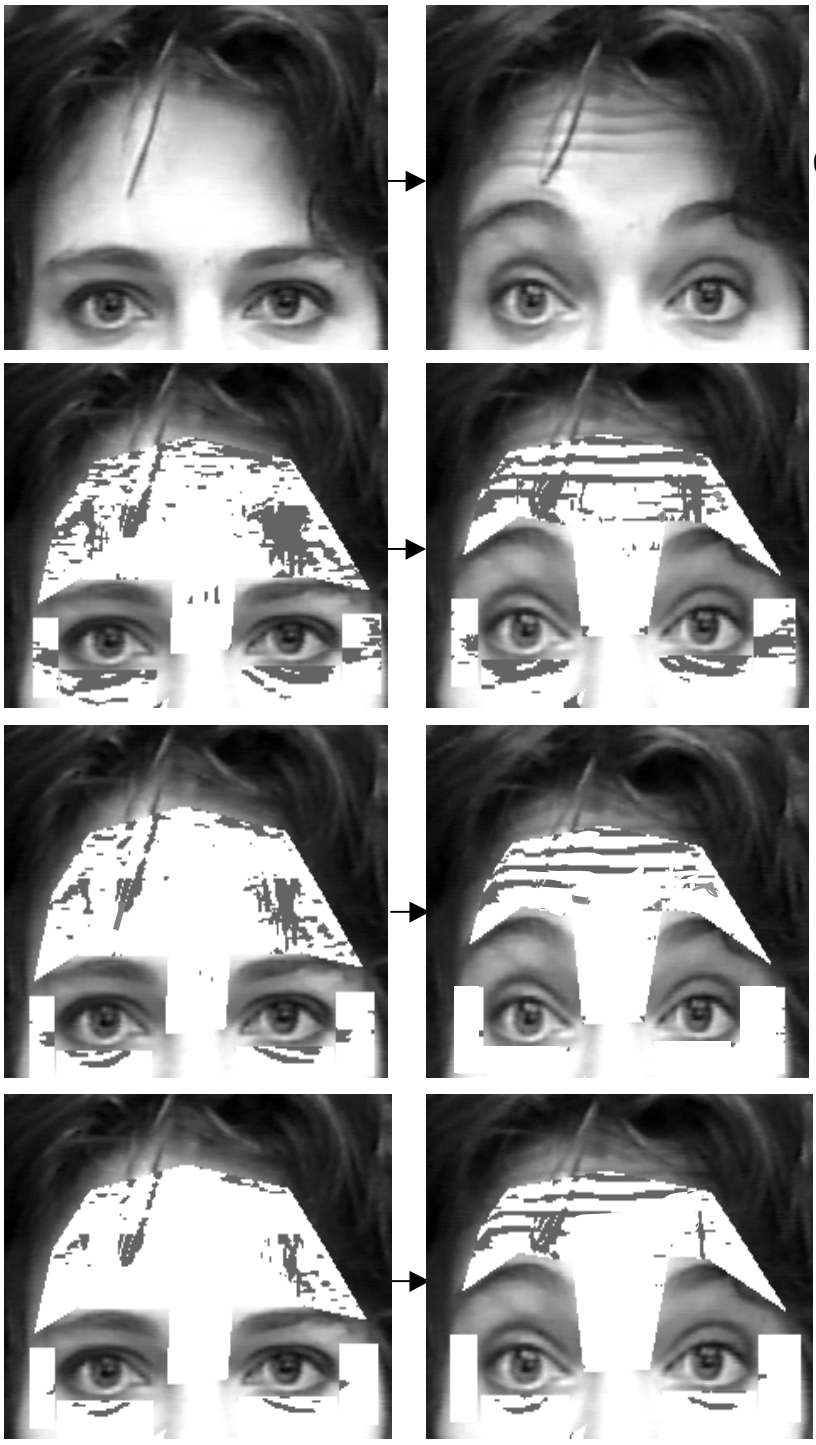


Figure 36.b High gradient component detection with different constant threshold values.

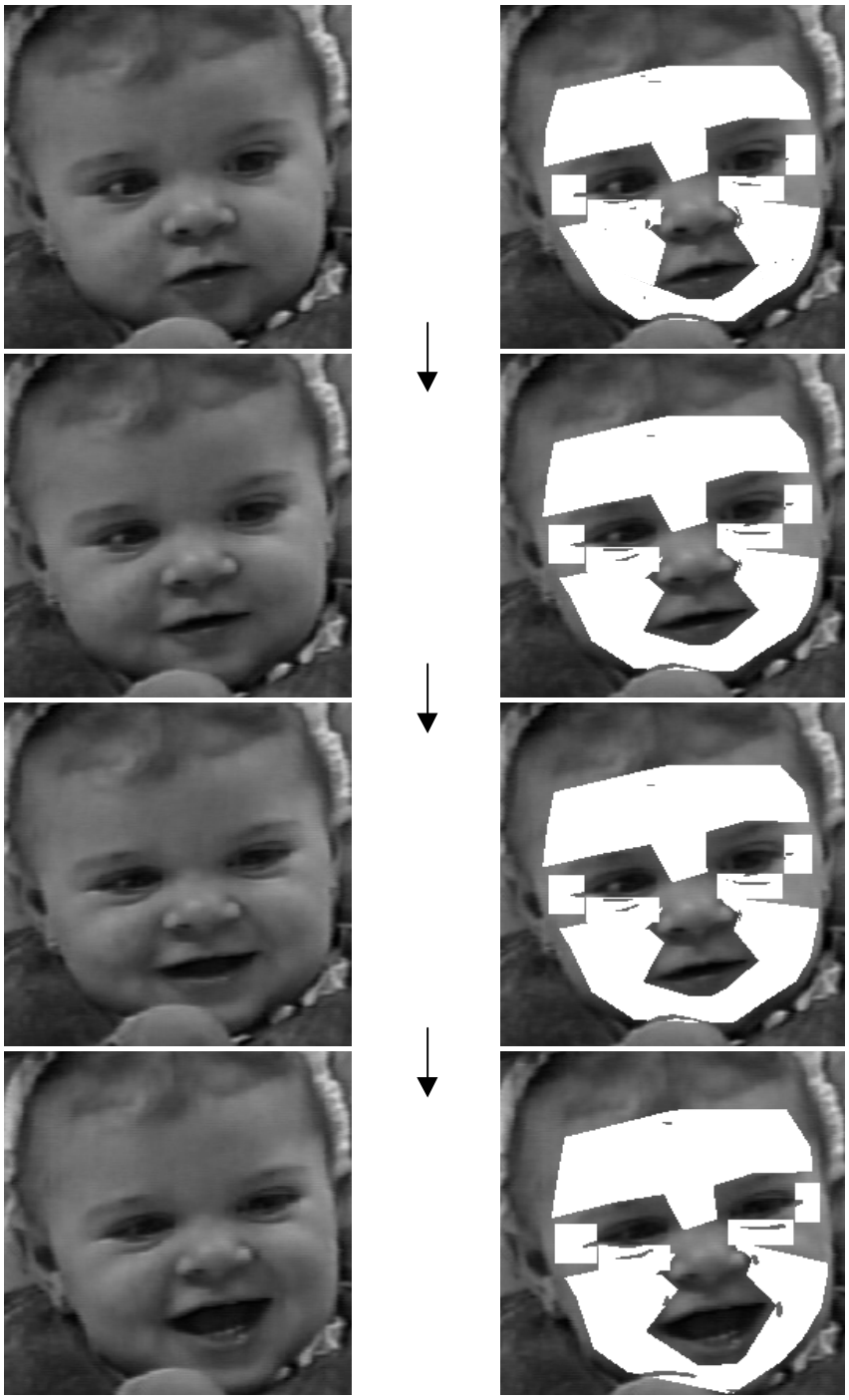


Figure 37.a Younger subjects, especially infants (Figure 37.a), show smoother furrowing than older ones (Figure 37.b), and initial expressions show weaker furrowing than that of peak expressions for each sequence.

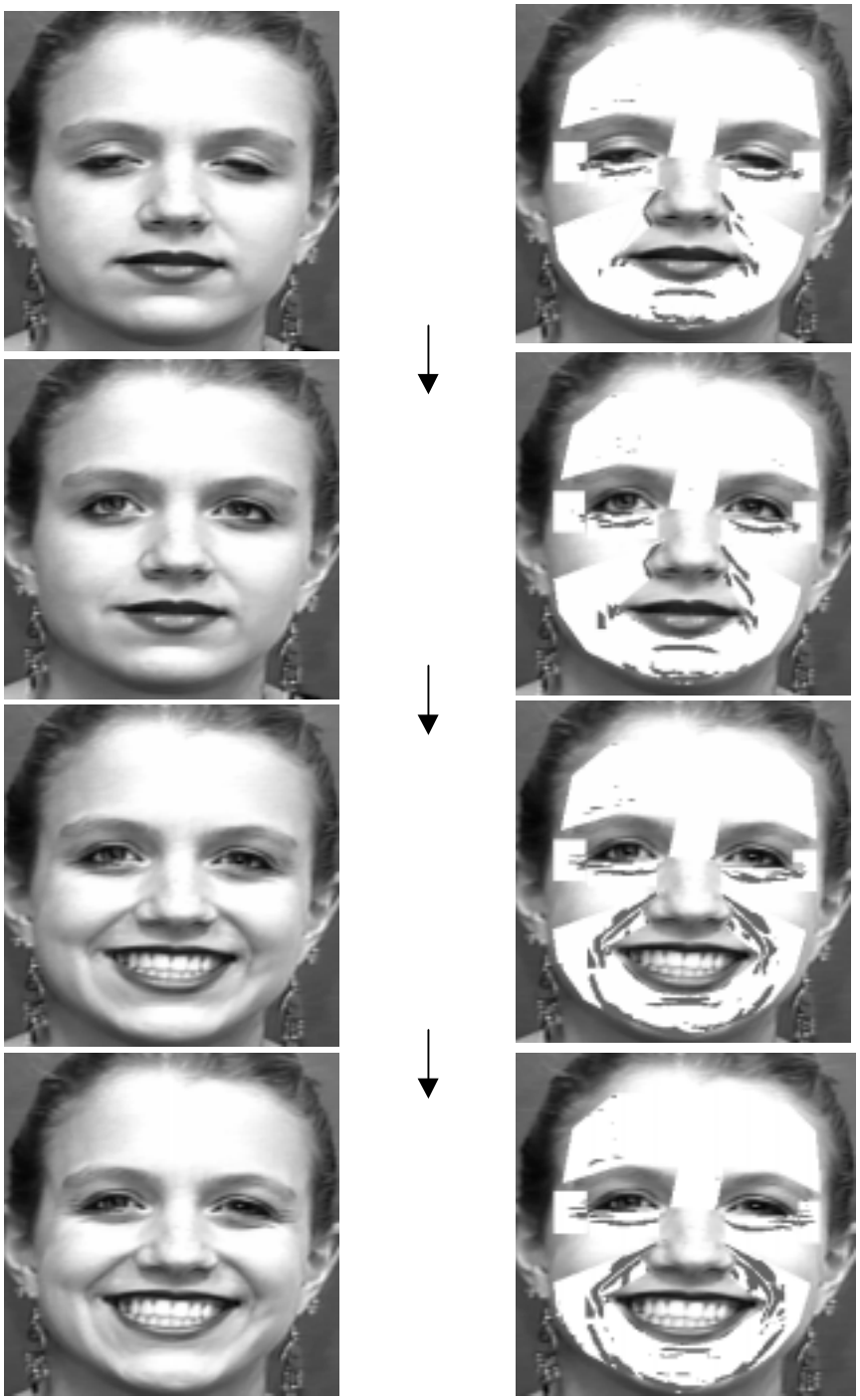
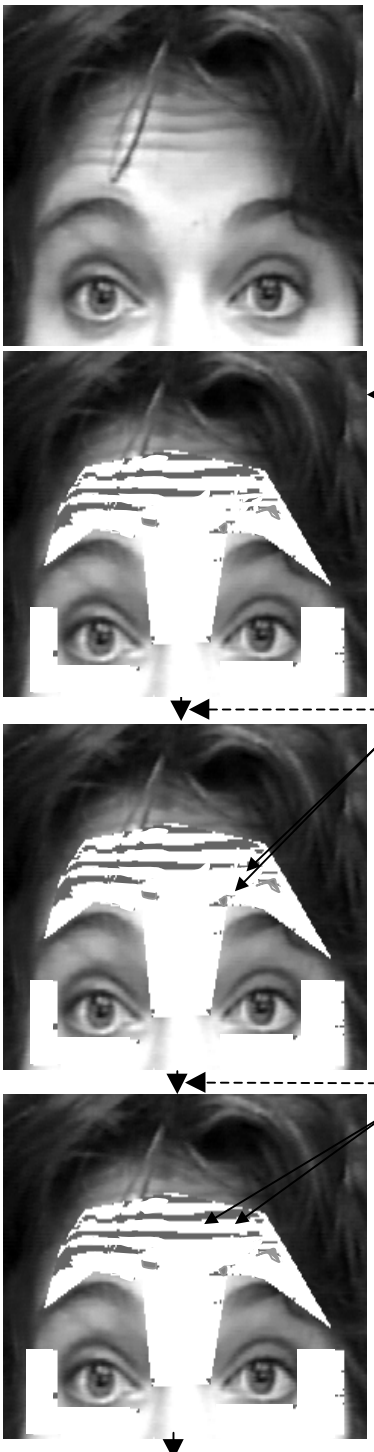


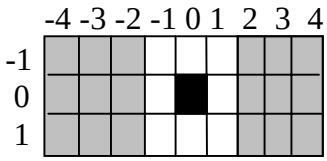
Figure 37.b Younger subjects, especially infants (Figure 37.a), show smoother furrowing than older ones (Figure 37.b), and initial expressions show weaker furrowing than that of peak expressions for each sequence.



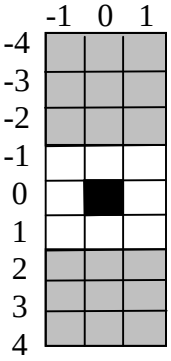
(a) Continue processing following Figure 34 & 35 (high gradient component analysis).

Template for erosion or dilation processing

Horizontal line template



Vertical line template



High gradient component

(b) Erosion processing – eliminating the small piece causing noise or the very short lines, thinning the lines, or smoothing the line boundary:
Consider the 8-connected component of the grey color region at each template, if there is no three-pixel connection of the horizontal (or vertical) direction at both regions after detection by the horizontal (or vertical) line template, then the center black pixel will be removed.

(c) Dilation processing – linking the broken lines:
Consider the 8-connected component of the grey color region at each template, if there is three-pixel connection of the horizontal (or vertical) direction at both regions after detection by the horizontal (or vertical) line template, then the center black pixel will be connected with both grey regions.

Figure 38.b: Connected Component Labeling.

Figure 38.a Delete redundant high gradient components using morphological transformation including erosion and dilation processings.

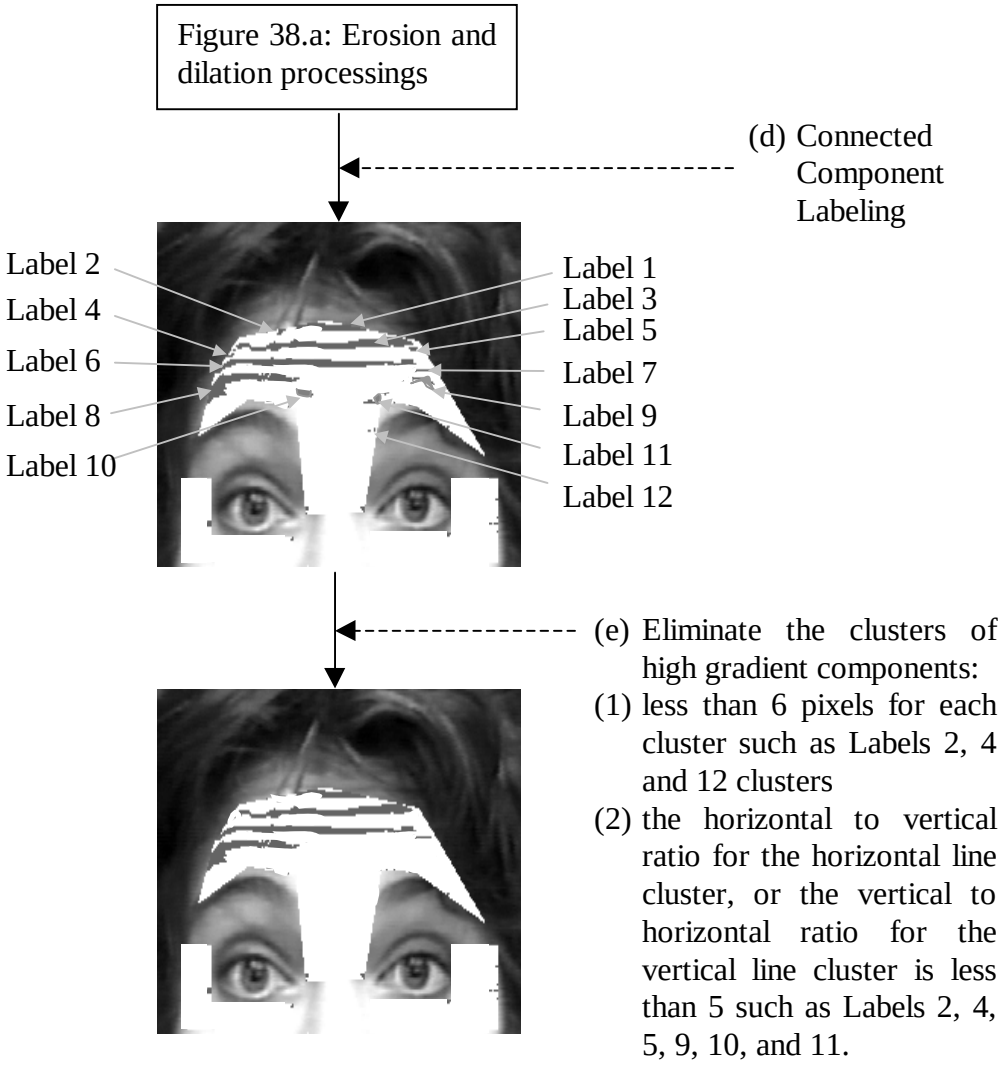


Figure 38.b Delete the redundant high gradient components using the connected component labeling algorithm.

5.4 Analysis of High Gradient Component Detection Problems

We have tried to use 3 x 3 Canny [19], Sobel [43], Prewitt [43] and wavelet-based [52] edge detectors to detect the high gradient components on the face image. We can not find any difference in the results.

Some high gradient components, such as the horizontal lines along the furrows at the forehead, are wide. If we use high gradient component detectors with small sizes, such as 3 x 3, to extract this line from vertical directions, which are perpendicular to this line or furrow direction, then this wide line will be detected into two lines (Figure 39). One way to solve this problem is to use line detectors with large sizes such as 3 x 5 or 5 x 7 in order to match the width and the length of the detected lines or furrows (Figure 39). In our approach, the 3 x 5, 5 x 3, and 5 x 5 horizontal, vertical, and diagonal line detectors, respectively, are adequate for most of the facial expression images. The larger size of detector requires more computation time.

The high gradient components (such as furrows) move in consecutive frames, so we need to consider the motion furrows in the spatio-temporal domain to ensure the detected high gradient components are produced by transient skin or feature deformations, and are not a permanent characteristic of the individual's face. Equation (5-5) is a way for tracking the motion high gradient component in the spatio-temporal domain. It is not accurate because of introducing the zeroing result: it is based on the tracking of individual pixels, which sometimes appear or disappear because of lighting or deformation of skin movement, or do not have any movement between consecutive frames because of a high sampling rate for an image sequence. To overcome above weakness, equation (5-6) can give a more reliable result by assuming the first (neutral expression) frame to be the background, then the foreground components. The motion of high gradient components can be extracted easily by subtraction instead of tracking processing from the remaining frames of each sequence.

According to equations (5-5) and (5-6), an interesting approach will be demonstrated by directly subtracting the gray values instead of gradient components: considering the

(a) Gray value image

(b) Using 3 x 3 horizontal line detectors

1	1	1
0	0	0
-1	-1	-1

-1	-1	-1
0	0	0
1	1	1

(c) Using 3 x 5 horizontal line detectors

1	1	1	1	1
0	0	0	0	0
-1	-1	-1	-1	-1

-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1

Figure 39 Horizontal line (furrow) detection using different sizes of detectors. (b) If the size of the detector is too small compared with the width and length of the line (furrow), then each line will be extracted to two lines. (c) It is necessary to adjust the size of the line detector to match the width and length of the line in order to obtain the correct result.

motion effect in the temporal domain and ignoring the gradient effect in the spatial domain.

$$\frac{\partial I(x,y,t)}{\partial t} = I(x,y,t) - I(x,y,t-1) = \tilde{\Gamma}_t(x,y) \tag{5-7}$$

or

$$\frac{\partial I^0(x,y,t)}{\partial t} = I(x,y,t) - I(x-\Delta x, y-\Delta y, 0) = \tilde{\Gamma}_t^0(x,y) \tag{5-8}$$

where

$$-1 \le \Delta x \le 1 \quad \text{and} \quad -1 \le \Delta y \le 1$$

Next, a threshold is given to remove unnecessary gray-value components at which the absolute gray-value differences between the two images are below the threshold. This threshold process can compensate for the lacking of without considering the gradient factor in the spatial domain. According to our experiments, it works well to extract the teeth when mouth is opening. Since the gray values of the target (teeth) are obviously different from the background (skin, lips, and tongue), it is easy to segment the foreground from background using a simple threshold process (Figure 40). It is very difficult to extract the furrows exactly by using a simple thresholding process of different gray values between two face images, because the gray values of the entire face image are affected by lighting and are different across the ages and individuals of subjects. Since we define and can observe furrows, which are constituted from the motion high gradient components on face image sequence, it is necessary to extract the motion furrow by considering the spatial and temporal gradient components at the mean time.

5.5 Data Quantization and Conversion for the Recognition System

After the motion and high gradient components (furrows) are extracted in the spatio-temporal domain, the high gradient pixels are assigned a value 1 and other background components are assigned a value 0 for each facial expression sequence, we want to summarize the high gradient components of many pixels into a low-dimensional vector for each face image as an input to the recognition system. The forehead (upper face) and lower face regions of each normalized face image are divided into 16 and 16 blocks (Figure 41). The mean number of high gradient components in each block is calculated by dividing the number of pixels having a value of 1 by the total number of pixels in the block. The positional variance of high gradient components in each block is calculated as the sum of variances in the row and column directions. The mean number and positional variance per block discussed here are simply abbreviated as mean-variance for brevity. These give 32 parameters for 16 blocks in each of the upper and lower face regions. For

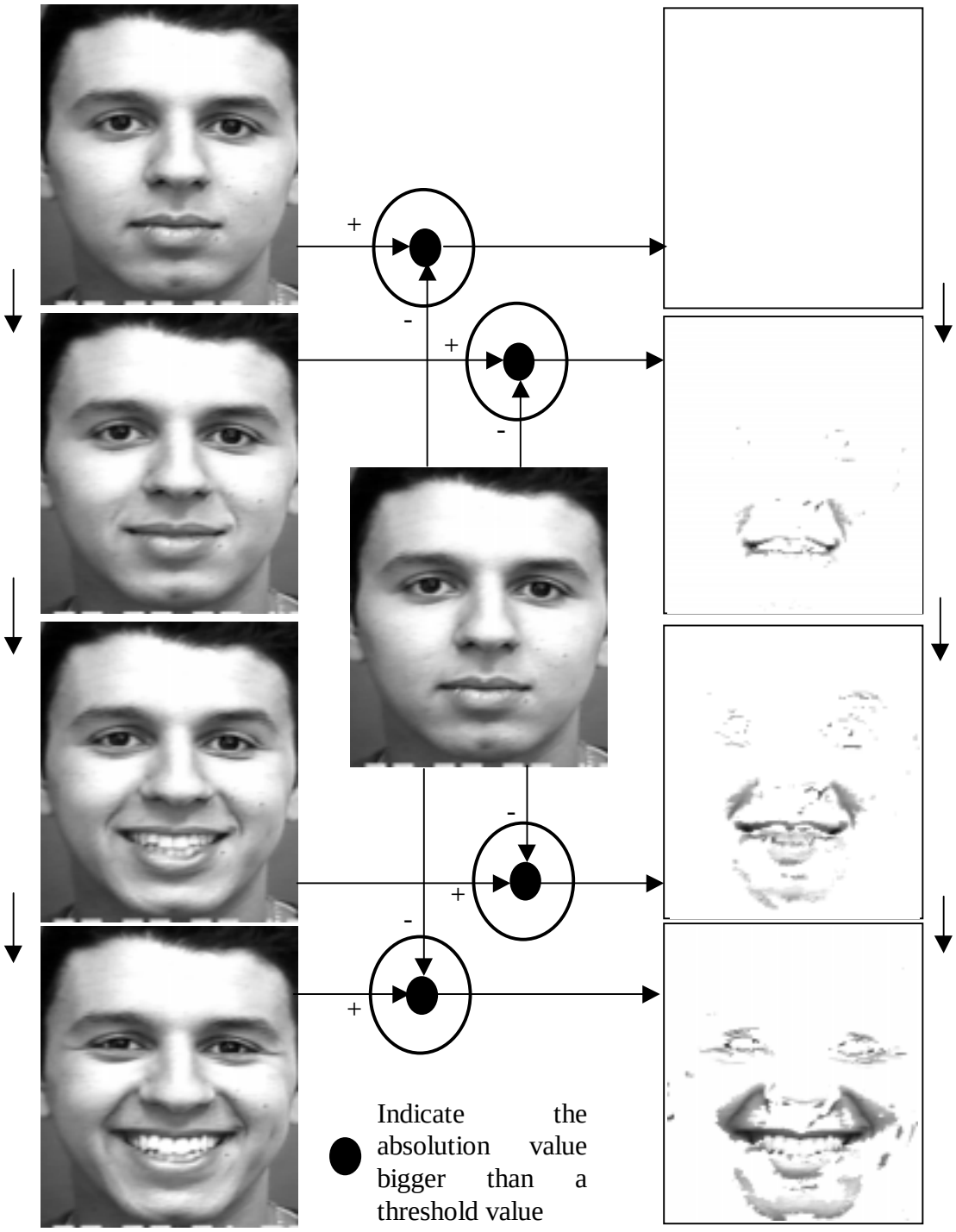
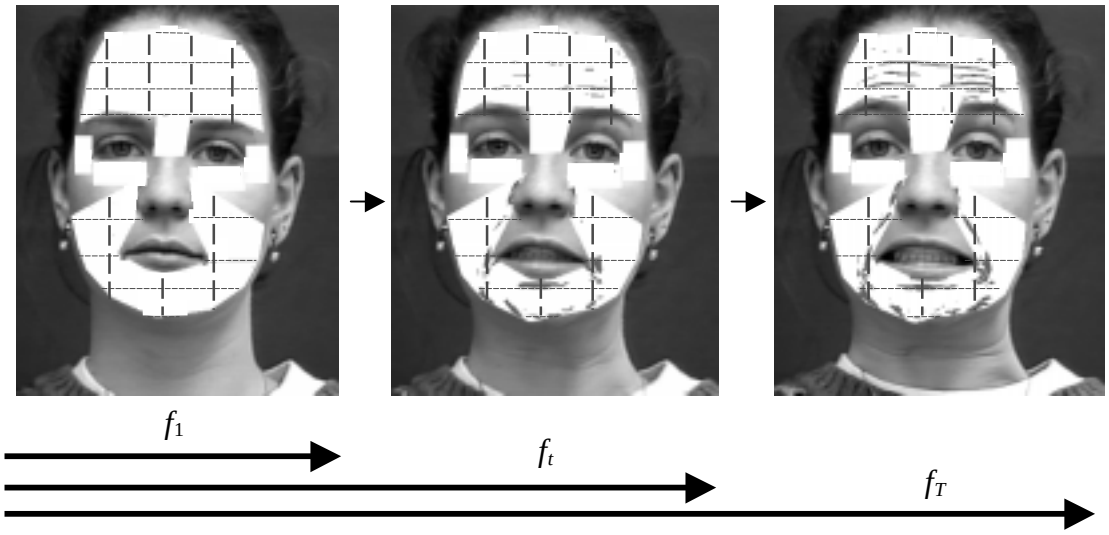


Figure 40 Teeth can be extracted directly from the subtraction of the gray value image at the current frame to that at first frame for each image sequence whose absolute value is larger than a constant threshold.



Mean-Variance vector $f_t = (m_{t,1}, m_{t,2}, \ldots, m_{t,j}, \ldots, m_{t,i}, \sigma_{t,1}, \sigma_{t,2}, \ldots, \sigma_{t,j}, \ldots, \sigma_{t,i})$

where $m_{t,j}$ is the mean number of high gradient components at the block j and frame t.

$\sigma_{t,j}$ is the positional variance of high gradient components at the block j and frame t.

and i = 16 (blocks) for the upper face region.

i = 16 (blocks) for the lower face region.

Mean-Variance vector sequence $f = (f_1, f_2, \ldots f_t, \ldots, f_T)$

where T is the length of this image sequence.

Figure 41 Mean-Variance vector of the high gradient component analysis in the spatio-temporal domain for input to the Hidden Markov Model.

Table 7 Sample symbol sequences for three upper facial expressions and six lower facial expressions under consideration.

AUs	**Motion Furrow Detection: Upper Facial Expressions (Symbol Sequence)**
0	0 0 0 0 *0* 0 0 0 0 *0* 0 0 0 0 *0* 0
4	0 0 0 0 *4* 4 10 10 10 *10*
1+4	0 0 0 12 *12* 12 12 12 6 *6* 6
1+2	0 0 0 0 *0* 12 9 9 3 *11* 11
AUs	**Motion Furrow Detection: Lower Facial Expressions (Symbol Sequence)**
12 (6+12+25)	10 10 10 10 *10* 10 1 1 1 *5* 5 5 5 5 *5* 5 5 5
9+17 (17+23+24)	10 10 10 10 *10* 10 12 12 12 *12* 12 4 4 4 *4* 4 4 4 4 *4* 4

upper and lower facial expression recognitions, these mean and variance values are concatenated to form a 32-dimensional mean-variance vector for each region in a frame. Table 7 shows sample symbol sequences for nine facial expressions under consideration. Such symbol sequences are used as inputs to the HMMs of upper facial expressions and lower facial expressions, respectively, for automatic recognition.

5.6 Expression Intensity Estimation

The sum of squared difference (SSD) criterion is employed to find a close estimation of furrow expression intensity. The furrow detection on each 417 x 385 image gives a 32-dimensional mean-variance vector for the upper facial expressions and similarly another 32-dimensional mean-variance vector for the lower facial expressions.

In the training data, the furrow expression intensity of individual frames of each facial expression sequence with length varying from 9 to 47 frames has been quantified by experts: from neutral expression (expression intensity: 0.0) to peak expression (expression intensity: 1.0). The corresponding mean-variance vector of each training frame has also been extracted. The Euclidean distance between the mean-variance vector of the testing frame to the mean-variance vector of the individual frame in the chosen training sequence is a measure of how close are their expression intensity values. After recognizing an input furrow expression sequence, the furrow expression intensity of an individual frame in the sequence is estimated by finding the best match of the furrow expression intensity from a training frame based on the shortest distance in the mean-variance space (Figure 42).

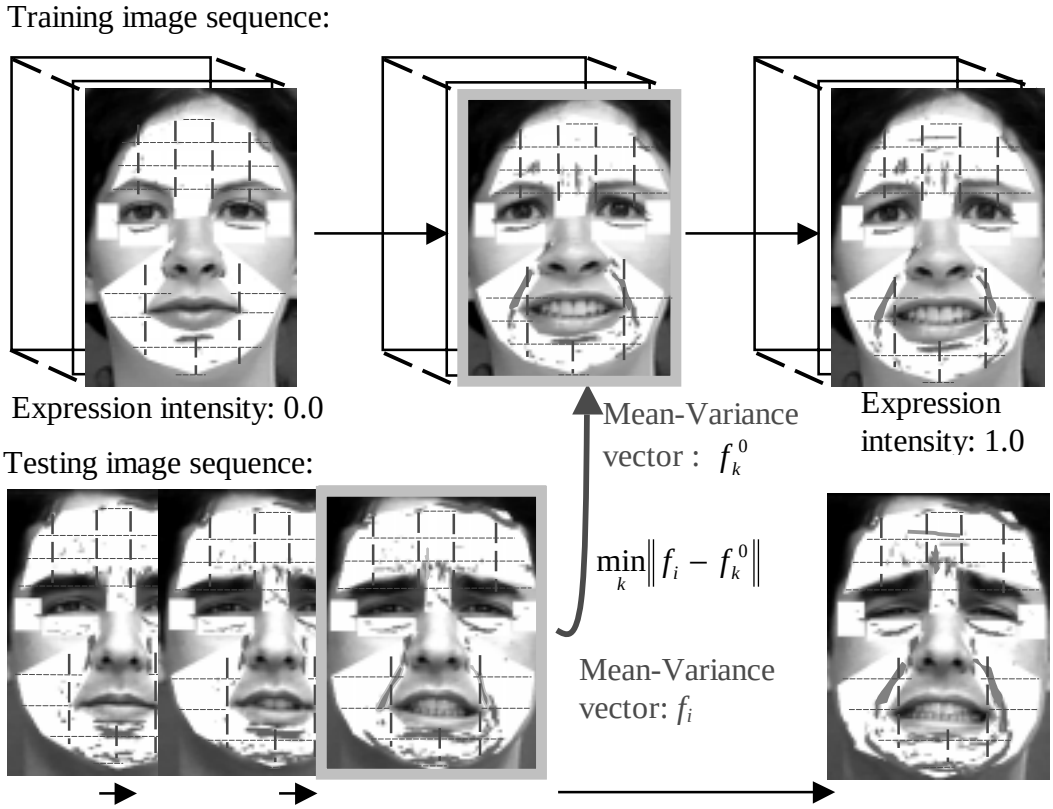


Figure 42 Furrow expression intensity matching by measuring the minimum value (distance) of the sum of squared differences (SSD) between the mean-variance vector of the known training image and that of the testing image. Each mean-variance vector of the training image corresponds to a given expression intensity value.

6.0 FACIAL EXPRESSION RECOGNITION USING HIDDEN MARKOV MODELS

If we try to build a signal model that can be used to explain and characterize the occurrence of the observable symbol sequences, then we can use this model to identify or recognize other sequences of observable symbols. A Hidden Markov Model (HMM) can be employed to represent the statistical behavior of an observable symbol sequence in terms of a network of states. For each observable symbol, the process being modeled occupies one of the states of the HMM. With each observable symbol, the HMM either stays in the same state or moves to another state based on a set of state transition probability associated with the state. The variety of the observable symbols for which the HMM uses a particular state is described in terms of the distribution of probability that each observable symbol will occur from that state. Thus, an HMM is a doubly (observable and hidden) stochastic model where the observable symbol probability distribution for each state captures the intra-state variability of the observable symbols, and the state transition probability describe the underling dynamic structure of the observable symbols.

We use HMMs to recognize subtly different facial expressions because of their simplicity and reliability. The HMM uses only three parameters: the initial state probability vector, the state-transition probability matrix, and the observable symbol probability matrix. The convergence of recognition computation may run in real time. Analysis of dynamic images naturally will yield more accurate recognition than that of a single static image, in our study, facial expressions are recognized in the context of entire image sequences of arbitrary lengths. Use of an HMM for facial expression recognition is advantageous because it is analogous to human performance which is a doubly stochastic process, involving a hidden immeasurable human mental state and measurable, observable human action. An HMM can produce satisfactory performance in the spatio-temporal domain and deal with the time warping problem. In addition, an HMM may allow for

multiple input sequences. This will result in a reliable recognition system as it will include a variety of extracted information from the facial expressions. It may also be used in combination for both expression recognition and speech recognition.

6.1 Preprocessing of Hidden Markov Models: Vector Quantization

In order to model various "expression units" of individual AUs or AU combinations for recognizing subtly different facial expressions, we train discrete HMMs (simply called HMMs in our study) to model facial expressions. We must first preprocess those training multi-dimensional vector sequences to convert them to those one-dimensional (discrete) symbol sequences. The specific preprocessing algorithm we chose is the vector quantization (VQ) [65]. VQ techniques have been used widely and successfully to solve quantization and data compression problems. In an HMM-based approach, we need to quantize each multi-dimensional feature or motion vector sequence into a finite symbol sequence before training HMMs.

The purpose of designing an M-level vector quantizer (called a codebook with size M) is to partition all k-dimensional training feature vectors into M clusters and associate each cluster C^i, whose centroid is the k-dimensional vector c^i, with a quantized value named codeword (symbol) o^i. While VQ will reduce data redundancy and get rid of small noise, it will inevitably cause a quantization error between each training feature vector x and c^i. As the size of the codebook increases, the quantization error decreases, and required storage for the codebook entries increases. It is very difficult to find a trade-off among these three factors.

In order to have a good recognition performance in using HMMs, it is critical to design a codebook for vector quantizing each k-dimensional training feature vector x into a symbol o^i with minimum quantization error. Therefore, two primary issues are considerable for the design of the codebook: (1) codebook creation (the size of codebook), and (2) distortion measurement. Defining the size of codebook is still an open

problem when we use the VQ technique. According to our experimental result, the recognition system has high performance when the size *M* of the codebook, which should be power of 2, is at least 1/50 less than the number of all *k*-dimensional training feature vectors.

For the distortion measurement, there are two main considerations for optimizing the VQ:

1. The quantizer must satisfy the nearest neighbor rule.

$$x \in C^i \qquad if \quad \left\| x - c^i \right\| < \left\| x - c^j \right\| \tag{6-1}$$

$$where \quad \left\| x - c^i \right\| = \sum_{h=1}^{k} (x_h - c_h^i)^2 \qquad and \qquad i \neq j, \quad i, j = 0,1,...,M\text{-}1 \tag{6-2}$$

and

$$q(x) = o^i \qquad\qquad where \quad 0 \le o^i \le M - 1 \tag{6-3}$$

This means that the *k*-dimensional feature vector $x = [x_1,x_2,...,x_k]$ is classified to cluster C^i, whose centroid is the *k*-dimensional vector c^i, and encoded to be the codeword o^i because the distance between *x* and c^i is shorter than *x* and c^j. *q(.)* is the quantization operator.

2. Each cluster center c^i must minimize not only the distortion D^i in cluster C^i but also total quantization errors *D*.

$$D = \sum_{i=0}^{M-1} D^i \tag{6-4}$$

$$where \quad D^i = \sum_{n=1}^{N} \left\| x_n^i - c^i \right\| = \sum_{n=1}^{N} \sum_{h=1}^{k} (x_{n,h}^i - c_h^i)^2 \tag{6-5}$$

N k-dimensional feature vectors x_n^i are located at cluster C^i. Because the total distortion *D* is a linear combination of D^i which is the distortion in cluster C^i, the *k*-dimensional cluster center c^i can be independently computed after classification of *x*.

Using the overall distortion measurement, it is hard to guarantee global minimization. A similar technique used for cluster analysis with squared error cost functions is called the K-means algorithm. The K-means algorithm is an iterative algorithm which can guarantee a local minimum, and works well in practice.

The K-means Algorithm:

Step 1: **Initialization** - Define the codebook size to be M and choose M initial (1st iteration) k-dimensional cluster centers $c^0(1)$, $c^1(1)$,..., $c^{M-1}(1)$ corresponding to each cluster C^i where $0 \le i \le M$-1.

Step 2: **Classification** - At the lth iteration, according to the nearest neighbor rule, classify each k-dimensional sample x of training feature vectors into one of the clusters C^i.

$$x \in C^i(l) \quad \text{if} \quad \left\|x - c^i(l)\right\| < \left\|x - c^j(l)\right\| \quad \text{where} \quad i \neq j,\ i, j = 0,1,...,M\text{-}1 \tag{6-6}$$

Step 3: **Codebook Updating** - Update the codeword (symbol) o^i of each cluster C^i by computing new cluster centers $c^i(l+1)$ where $i = 0,1,...,M$-1 at the l+1th iteration.

$$c^i(l+1) = \frac{1}{N}\sum_{n=1}^{N} x_n^i \qquad \text{where } x^i \in C^i(l+1) \tag{6-7}$$

N is the number of feature vectors in cluster $C^i(l+1)$ at the l+1th iteration, and

$$q(x) = o^i \qquad \text{where} \quad 0 \le o^i \le M-1 \tag{6-8}$$

where $q(.)$ is the quantization operator.

Step 4: **Termination** - If the decrease in the overall distortion at the current iteration l+1 compared with that of the previous iteration l is below a selected threshold, then stop; otherwise goes back to Step 2.

$$\begin{cases} \text{if } |D(l+1) - D(l)| < \text{threshold, then Stop} \\ \text{if } |D(l+1) - D(l)| \ge \text{threshold, then Goes to Step } 2 \end{cases} \tag{6-9}$$

Note that the K-means algorithm can only converge to a local optimum. The behavior of the K-means algorithm is affected by the number of clusters specified and the choice of initial cluster centers. Instead of using K-means algorithm, our VQ approach is based on Linde, Buzo and Gray's algorithm [65] for vector quantizer design, which is an extended algorithm of K-means, but unlike K-means which initializes each cluster center in the beginning. This VQ algorithm uses iterative method, splits the training vectors from assuming whole data to be one cluster to 2,4,8,...,M (M's size is power of 2) clusters, and

determines the centroid for each cluster. The centroid of each cluster is refined iteratively by K-means clustering.

The Vector Quantization Algorithm:

Step 1: **Initialization** - Assume all N k-dimensional training vectors to be one cluster C^0, *i.e.*, codebook size $M = 1$ and codeword $o^0 = 0$, and find its k-dimensional cluster centroid $c^0(1)$ where 1 is the initial iteration.

$$c^0(1) = \frac{1}{N}\sum_{n=1}^{N} x_n^0 \tag{6-10}$$

where x is one sample of all N k-dimensional feature vectors at cluster C^0.

Step 2: **Splitting** - Double the size M of the codebook by splitting each cluster into two. The current codebook size M is split into $2M$. Set $M = 2M$ by

$$\begin{cases} c_+^i(1) = c^i(1) + \varepsilon \\ c_-^i(1) = c^i(1) - \varepsilon \end{cases} \quad where \;\; 0 \le i \le M-1 \tag{6-11}$$

c^i is the centroid of the ith cluster C^i, M is the size of current codebook, ε is a k-dimensional splitting parameter vector and is value 0.0001 for each dimension in our study. 1 is the initial iteration.

Step 3: **Classification** - At the lth iteration, according to the nearest neighbor rule, classify each k-dimensional sample x of training feature vectors into one of the clusters C^i.

$$x \in C^i(l) \quad if \; \left\|x - c^i(l)\right\| < \left\|x - c^j(l)\right\| \quad where \;\; i \ne j,\; i, j = 0,1,...,M\text{-}1 \tag{6-12}$$

Step 4: **Codebook Updating** - Update the codeword (symbol) o^i of each cluster C^i by computing new cluster centers $c^i(l+1)$ where $i = 0,1,...,M$-1 at the l+1th iteration.

$$c^i(l+1) = \frac{1}{N}\sum_{n=1}^{N} x_n^i \qquad where \;\; x^i \in C^i(l+1) \tag{6-13}$$

N is the number of feature vectors in cluster $C^i(l+1)$ at the l+1th iteration. And

$$q(x) = o^i \qquad where \;\; 0 \le o^i \le M-1 \tag{6-14}$$

where $q(.)$ is the quantization operator.

Step 5: **Termination 1** - If the difference between the current overall distortion $D(l+1)$ and that of the previous iteration $D(l)$ is below a selected threshold, proceed to Step 6; otherwise goes back to Step 3.

$$\begin{cases} if\ |D(l+1)-D(l)| < threshold,\ then\ Goes\ to\ Step\ 6 \\ if\ |D(l+1)-D(l)| \geq threshold,\ then\ Goes\ to\ Step\ 3 \end{cases} \quad (6\text{-}15)$$

(where *threshold* is 0.0001 in our study.)

Step 6: **Termination 2** -

Is the codebook size M equal to the VQ codebook size required ?

$$\begin{cases} if\ Yes,\ then\ Stop \\ if\ No,\ then\ Goes\ to\ Step\ 2 \end{cases} \quad (6\text{-}16)$$

Once the final codebook is obtained according to all training vectors by using this VQ algorithm, it is used to vector quantize each training and testing feature (or motion) vector into a symbol value (codeword) for the preprocessing of the HMM recognition process (Figure 43).

6.2 Beginning from Markov Models

The first order Markov chain is a stochastic process which follows the rule

$$P(q_{t+1}=j \mid q_0=k, q_1=l, \ldots, q_t=i) = P(q_{t+1}=j \mid q_t=i) \quad (6\text{-}17)$$

where q_t represents the state q at time t, and i, j, k, l represent the possible states of q at different instant of time. The first order Markov chain states that the probabilistic dependence is truncated at the preceding state. We consider only those processes in which the right-hand side of above equation is independent of time. We can then see that a time independent Markov chain is characterized by its state-transition probability a_{ij},

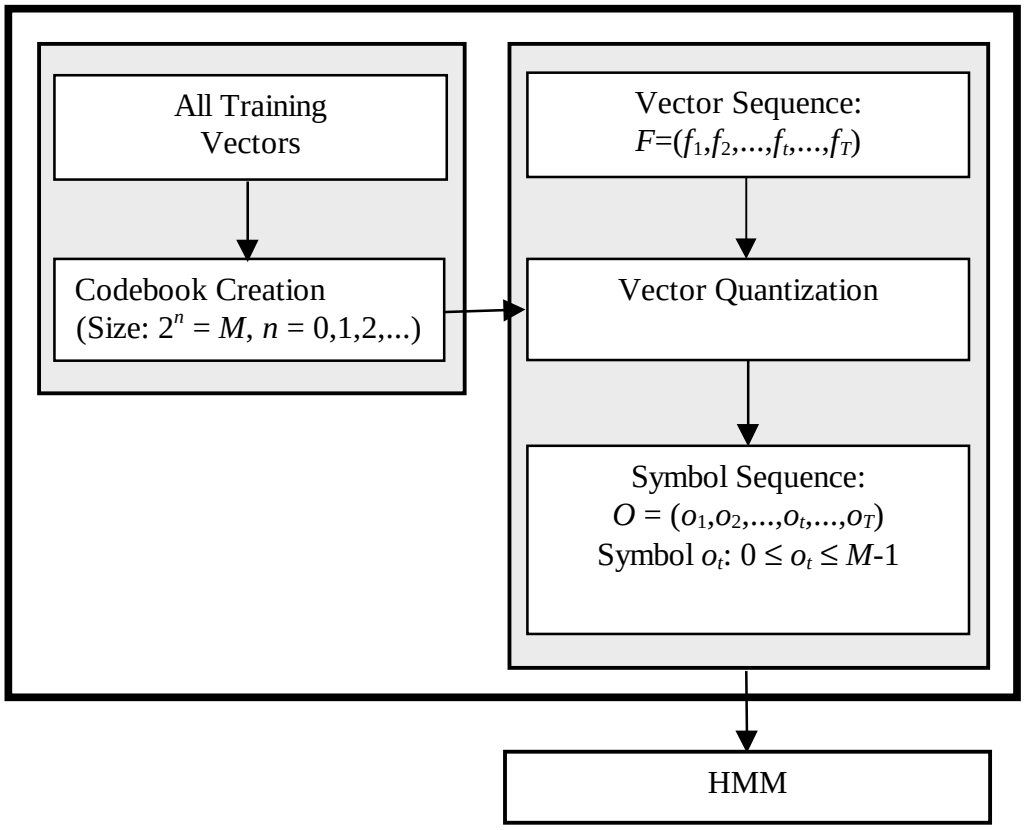


Figure 43 Vector quantization for encoding any vector sequence to a symbol sequence based on the codebook.

which is the probability of moving from one state i to another state j.

$$a_{ij} = P(q_t=j \mid q_{t-1}=i), \quad 1 \le i,j \le N \tag{6-18}$$

where N is the total number of states. The a_{ij} obeys standard stochastic constraints.

$$a_{ij} \ge 0 \qquad 1 \le i,j \le N \tag{6-19}$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad 1 \le i \le N \tag{6-20}$$

The Markov model could be called an observable Markov model because the output of the stochastic process is the state sequence where each state corresponds to each instant of time with a deterministically observable event (symbol).

6.3 Extension of Markov Models: Hidden Markov Models

In the Markov model, the state sequence is observable. The output observable event in any given state is deterministic, not random. This will be too constraining when we use it to model the stochastic nature of the human performance, which is related to doubly stochastic processes, namely human mental states (hidden) and human actions (observable). It is necessary that the observable event is a probabilistic function of the state. That is why an HMM is employed. HMM is a representation of a Markov process and is a doubly embedded stochastic process with an underlying stochastic process that cannot be directly observed, but can only be observed through another set of stochastic processes that produce the sequence of observable symbols.

Before the description of HMMs, we define the elements of an HMM by specifying the following parameters:

***N*:** The number of states in the model. The state of the model at time t is given by q_t,

$$1 \le q_t \le N \qquad and \qquad 1 \le t \le T \tag{6-21}$$

where T is the length (number of frames) of the output observable symbol sequence.

***M*:** The size of the codebook or the number of distinct observable symbols per state. Assume o_t is one of all possible observable symbols for each state at time t, then

$$0 \le o_t \le M\text{-}1 \tag{6-22}$$

π_N**:** An N-element vector indicates the initial state probability.

$$\pi = \{\pi_i\}, \qquad where \quad \pi_i = P(q_1=i),\ 1 \le i \le N \tag{6-23}$$

$\boldsymbol{A_{NxN}}$**:** An N x N matrix specifies the state-transition probability that the state will transit from state i to state j.

$$A = \{a_{ij}\} \qquad where \quad a_{ij} = P(q_t=j \mid q_{t-1}=i), \quad 1 \le i,j \le N \tag{6-24}$$

and

$$a_{ij} \ge 0, \qquad \sum_{j=1}^{N} a_{ij} = 1 \qquad 1 \le i \le N \tag{6-25}$$

B_{MxN}: An *M* x *N* matrix represents the probability that the system will generate the observable symbol o_t at state *j* and at time *t*.

$$B = \{b_j(o_t)\} \quad where \quad b_j(o_t) = P(O_t=o_t \mid q_t=j), \quad 1 \le j \le N, \; 0 \le o_t \le M\text{-}1, \tag{6-26}$$

and

$$b_j(o_t) \ge 0, \quad 1 \le j \le N, \quad and \quad \sum_{o_t=0}^{M-1} b_j(o_t) = 1, \quad 1 \le j \le N \tag{6-27}$$

The complete parameter set λ of the discrete HMM is represented by one vector π and two matrices *A* and *B*

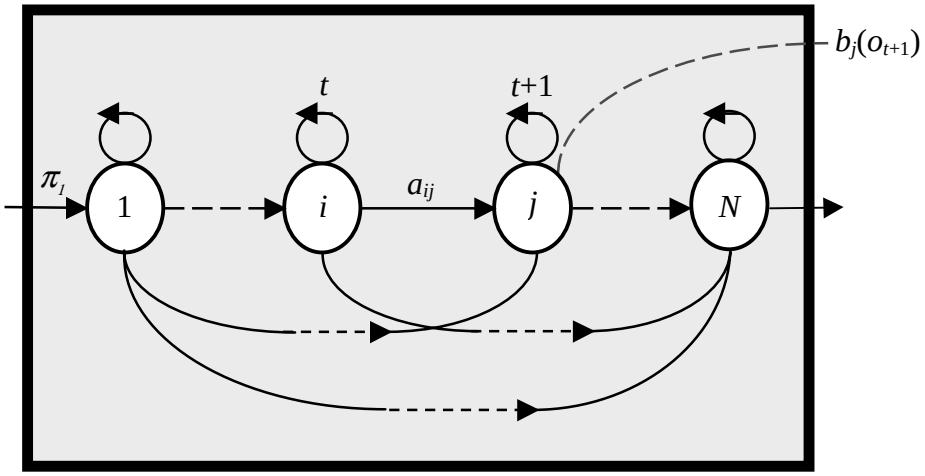
$$\lambda = (\pi, A, B) \tag{6-28}$$

In order to accurately describe a real-world process such as facial expression with an HMM, we need to appropriately select the HMM parameters. The parameter selection process is called the HMM "training."

This parameter set λ can be used to evaluate the probability $P(O \mid \lambda)$, that is to measure the maximum likelihood performance of an output observable symbol sequence *O*.

$$O = (o_1, o_2, ..., o_T) \tag{6-29}$$

where *T* is the number of frames for each image sequence. For evaluating each $P(O \mid \lambda)$, we need to select the number of states *N*, select the size of the codebook or the observable symbols *M*, and compute the results of probability density vector π and matrices *A* and *B* by training each HMM from a set of corresponding training data after VQ (Figure 44).



Symbol sequence:	$O = (o_1,o_2,...,o_t,...,o_T)$
State:	$q = (q_1=1,\ldots,q_t=i,q_{t+1}=j,\ldots,q_T=N)$
Codebook size:	M
HMM parameter set:	$\lambda = (\pi,A,B)$
Initial state distribution:	$\pi_1 = 1.0, \quad \pi_k = 0.0$ if $2 \le k \le N$
State-transition probability:	$A_{NxN} = \{a_{ij}\}$ from state i to j
Observable symbol probability:	$B_{MxN} = \{b_j(o_{t+1})\}$ at state j and time t+1
Output probability:	$P(O\|\lambda)$

Figure 44 The construction (topology) of the Hidden Markov Model.

6.4 Three Basic Problems of Hidden Markov Models

There are three basic problems in HMM design:

1. Problem of Probability Evaluation: How do we efficiently evaluate $P(O \mid \lambda)$, the probability (or likelihood) of an output observable symbol sequence $O = \{o_1,o_2,...,o_T\}$ given an HMM parameter set $\lambda = (\pi,A,B)$?

2. Problem of Optimal State Sequence: How do we determine an optimal state sequence $q = \{q_1,q_2,...,q_T\}$, which is associated with the given output observable symbol sequence $O = \{o_1,o_2,...,o_T\}$, by given an HMM parameter set $\lambda = (\pi,A,B)$?

3. Problem of Parameter Estimation: How do we regulate an HMM parameter set $\lambda = (\pi,A,B)$ in order to maximize the output probability $P(O \mid \lambda)$ of generating the output observable symbol sequence $O = \{o_1,o_2,...,o_T\}$?

Analyzing and solving the above three basic problems can help us to design and understand the HMM for training and recognition processes.

6.4.1 Probability Evaluation Using the Forward-Backward Procedure

In order to use an HMM for facial expression recognition, we need to compute the output probability $P(O \mid \lambda)$ with which the HMM will generate an output observable symbol sequence $O = \{o_1,o_2,...,o_T\}$ given the parameter set $\lambda = (A,B,\pi)$. The most straightforward way to compute this is by enumerating every possible state sequence of length T, so there will be N^T possible combinations of state sequence where N is the total number of states. Suppose there is one state sequence

$$q = \{q_1, q_2,..., q_T\} \tag{6-30}$$

Assume statistical independence of observable symbol o, and given the above state sequence q, the probability of the output observable symbol sequence will be

$$P(O \mid q,\lambda) = \prod_{t=1}^{T} P(o_t \mid q_t,\lambda) = b_{q_1}(o_1)\, b_{q_2}(o_2) \ldots b_{q_T}(o_T) \quad (6\text{-}31)$$

Also, we can get the probability of such a state sequence q by given an HMM parameter set λ.

$$P(q \mid \lambda) = \pi_{q_1}\, a_{q_1 q_2}\, a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \quad (6\text{-}32)$$

The joint probability of O and q (or the probability that O and q occur at the same time) is

$$P(O,q \mid \lambda) = P(O \mid q,\lambda)\, P(q \mid \lambda) \quad (6\text{-}33)$$

The probability of $P(O \mid \lambda)$ is the summation of this joint probability over all N^T possible state sequences q.

$$\begin{aligned} P(O \mid \lambda) &= \sum_{i=1}^{N} P(o_1 o_2 \ldots o_T,\ q_T = i \mid \lambda) \\ &= \sum_{i=1}^{N} P(O,\ q_T = i \mid \lambda) \\ &= \sum_{i=1}^{N} P(O \mid q_T = i,\lambda)\, P(q_T = i \mid \lambda) \\ &= \sum_{q_1 q_2 \ldots q_T} \pi_{q_1}\, b_{q_1}(o_1)\, a_{q_1 q_2}\, b_{q_2}(o_2) \ldots a_{q_{T-1} q_T}\, b_{q_T}(o_T) \end{aligned} \quad (6\text{-}34)$$

For the time complexity of the above computation, we can base the interpretation on Figure 45. Each state q_{t+1} at time $t+1$ has N possible paths with order O(1) calculations to be reached from the previous N state q_t at time t. That is, each state q_2 at time $t = 2$ can be reached from N possible state q_1 at time $t = 1$, and N possible state q_2 at $t = 2$. Each state q_3 at time $t = 3$ can have N^2 possible paths to be reached from N possible states q_1 at t = 1. Overall there are N^{T-1} possible paths with the order O(T) calculations to reach each final state q_T at time T for each state sequence. (According to the above equation, 2T-1 multiplications are required for each state sequence.) The time complexity is the order O(N^{T-1} T) for each state sequence. The totally N final state q_T at time T can be reached. The overall time complexity of computing the probability of $P(O \mid \lambda)$ is the order O(N^{T-1} T) x O(N) = O(N^T T). The value is very difficult to calculate. Even if we have a small

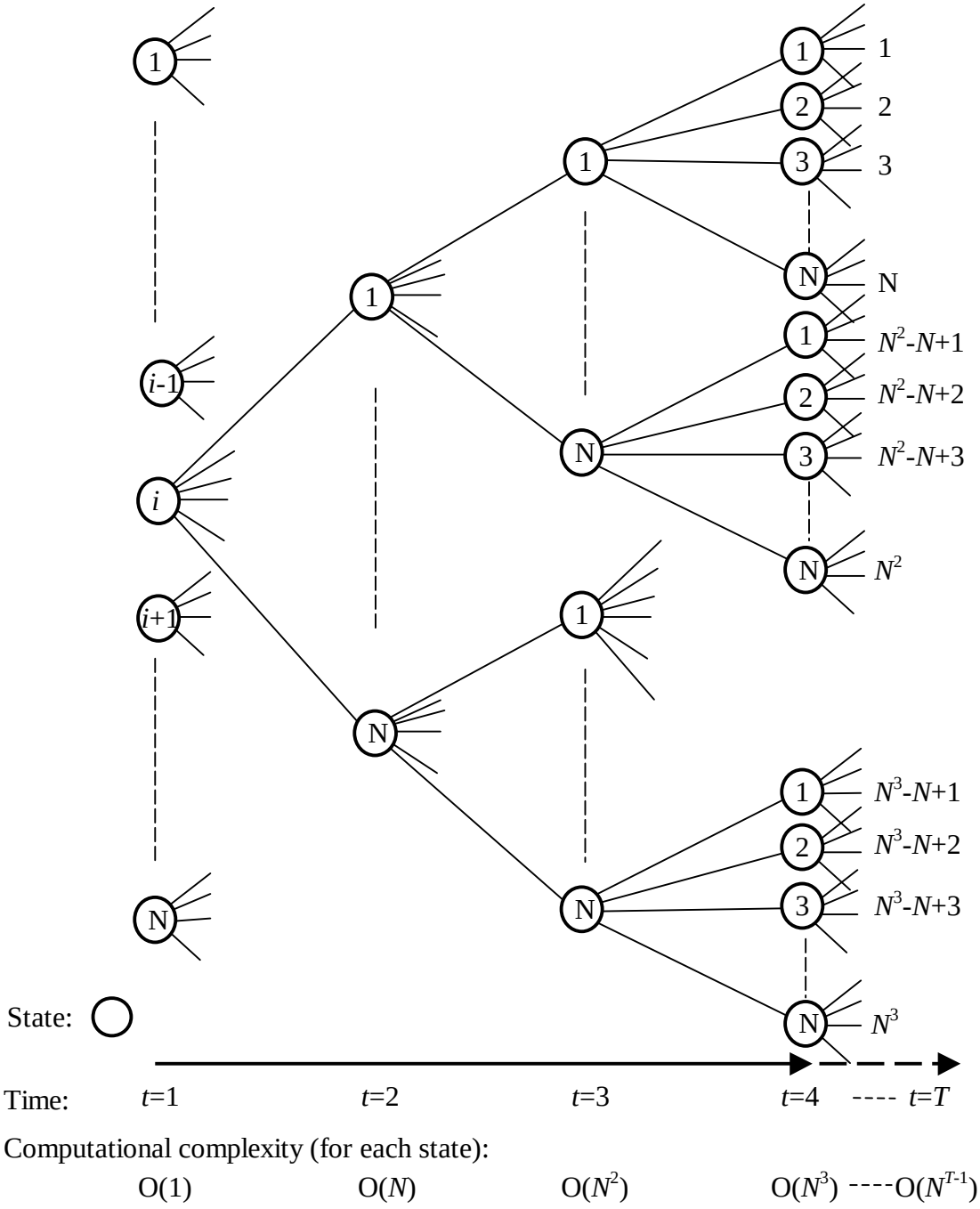


Figure 45 The tree structure of the computational complexity for direct evaluation of the output probability $P(O|\lambda)$ [105].

number of states N and T frames of state sequence, *e.g.*, $N = 4$ and $T = 20$, it still requires on the order of $4^{20} \times 20 \approx 2.2 \times 10^{13}$ calculations. Fortunately, we can use a more efficient procedure called the Forward-Backward procedure [79] to overcome this limitation.

Figure 46 can help us to describe the Forward procedure easily and clearly. We define the forward variable

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t, q_t = i \mid \lambda) \tag{6-35}$$

as the probability of the partial observable symbol sequence $o_1 o_2 \ldots o_t$ at state i and at time t by given the HMM parameter set λ. We can solve for $\alpha_t(i)$ inductively as follows:

The Forward Procedure:

1. Initialization: The initial forward variable is the joint probability of state i, time $t = 1$ and initial observable symbol o_1 by given the HMM parameter set λ.

$$\alpha_1(i) = P(o_1, q_1 = i \mid \lambda) = \pi_i b_i(o_1) \qquad \textit{where} \quad 1 \le i \le N \tag{6-36}$$

2. Induction (or Recursion): State j can be reached at time $t+1$ from the N possible states i, $1 \le i \le N$, at time t with the state-transition probability a_{ij}.

$$\begin{aligned} \alpha_{t+1}(j) &= P(o_1 o_2 \ldots o_{t+1}, q_{t+1} = j \mid \lambda) \\ &= \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}), \qquad \textit{where} \quad 1 \le t \le T-1,\ 1 \le i, j \le N \end{aligned} \tag{6-37}$$

3. Termination: The sum of all N final forward variables $\alpha_T(i)$, $1 \le i \le N$.

$$\begin{aligned} P(O \mid \lambda) &= \sum_{i=1}^{N} P(o_1 o_2 \ldots o_T,\ q_T = i \mid \lambda) \\ &= \sum_{i=1}^{N} P(O,\ q_T = i \mid \lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned} \tag{6-38}$$

The second step of the Forward procedure reduces the computational complexity since the calculation of the forward variable $\alpha_{t+1}(j)$ at time $t+1$, state j and observable symbol

o_{t+1} depends only on the previous (at time t) N forward variables $\alpha_t(i)$, $1 \le i \le N$ (Figure 46, 47). This computation is performed for all states j, $1 \le j \le N$, and then iterated from the initial frame at $t = 1$ to $t = T$-1 for all possible state sequences. In other words, because there are only N states at each instant time, all the possible state sequences will remerge into these N states, no matter how long the observable symbol sequence will be (Figure 47). The time complexity is the order O(N T) for each observable symbol sequence. This computation obviously reduces the computational complexity of each state sequence from the order O(N^{T-1}) to O(N). There are N states for each instant time or at the end time $t = T$. The overall time complexity of computing the probability of $P(O \mid \lambda)$ is O(N T) x O(N) = O(N^2 T) whose origin is O(N^T T). Compared to the original example $N = 4$ and $T = 20$, it requires only the order of 4^2 x 20 = 3.2 x 10^2 which is much less than 4^{20} x 20 ≈ 2.2 x 10^{13} calculations.

Figure 46 describes the Backward procedure. We define a backward variable

$$\beta_t(i) = P(o_{t+1}o_{t+2}...o_T \mid q_t=i, \lambda) \qquad (6\text{-}39)$$

which means the probability of the partial observable symbol sequence from t+1 to the end time T by given state i at time t and the HMM parameter set λ. We can compute the $\beta_t(i)$ using the following steps:

The Backward Procedure:

1. **Initialization:** Arbitrarily defines the backward variable at the end time T and state i as

$$\beta_T(i) = 1 \qquad \textit{where} \quad 1 \le i \le N \qquad (6\text{-}40)$$

2. **Induction (or Recursion):** State i can reach N possible states j, $1 \le j \le N$, at time t+1 as well as the observable symbol o_{t+1} by state-transition probability a_{ij} and observation probability $b_j(o_{t+1})$.

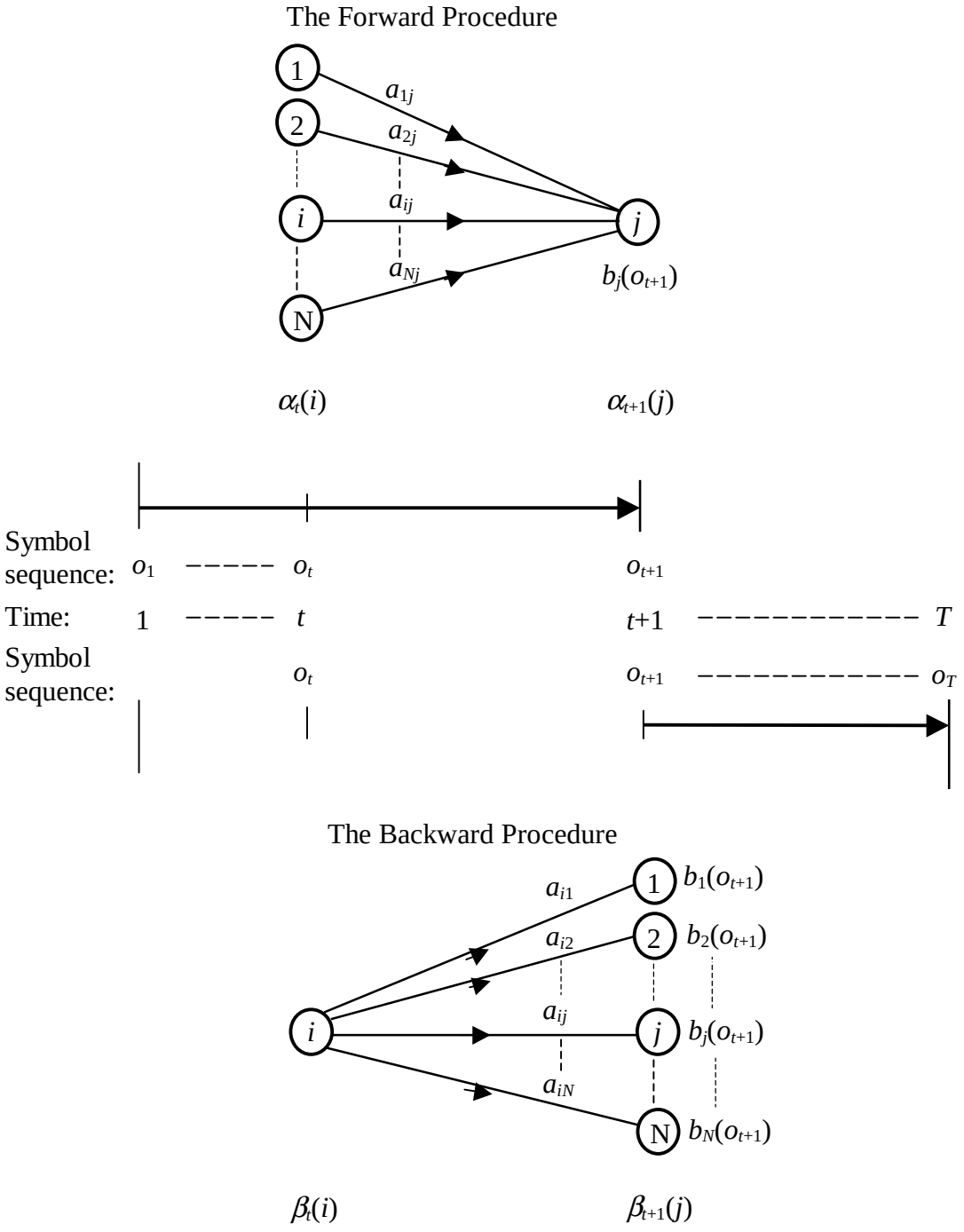
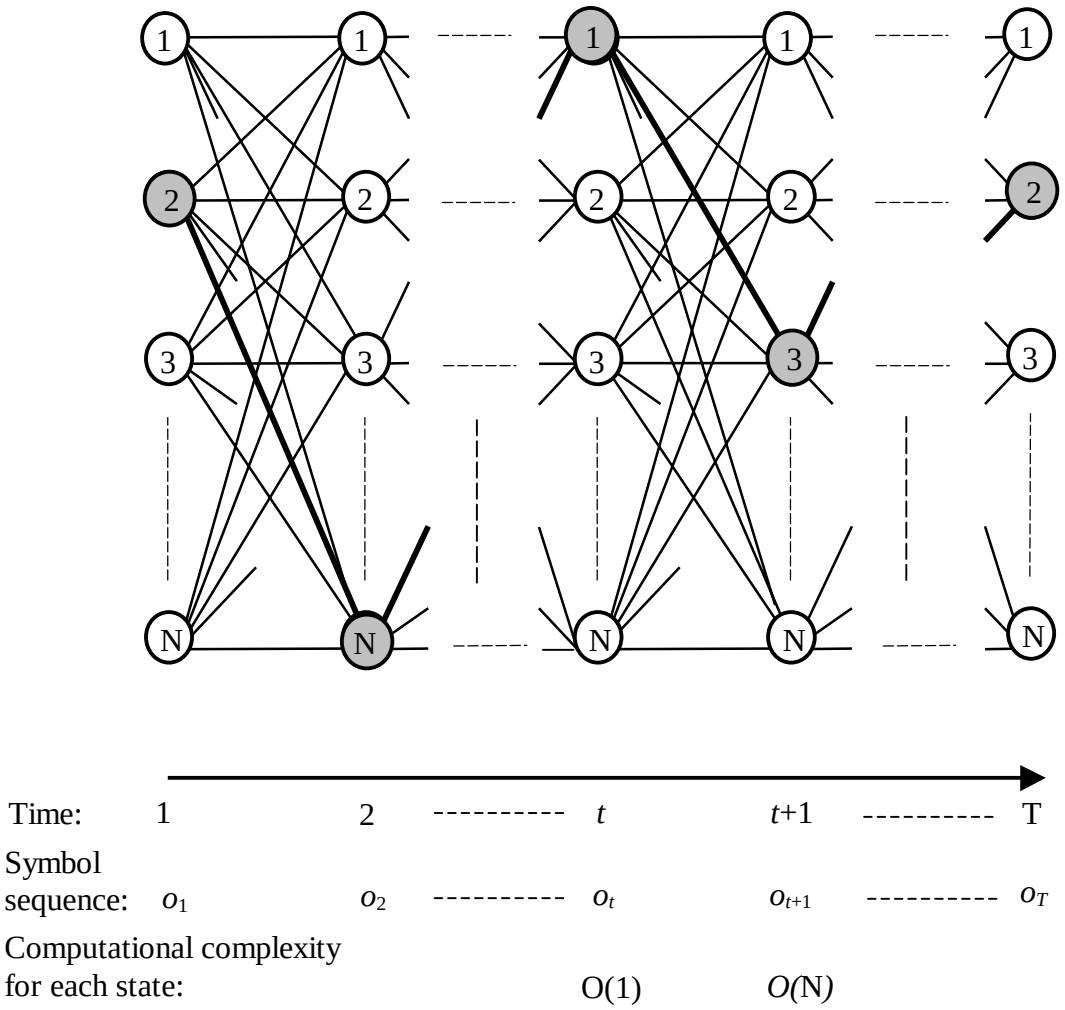


Figure 46 The Forward and Backward Procedures.



Computational complexity for total N states: $O(N) * O(N) = O(N^2)$

The single shortest (best) path (state sequence):

Figure 47 The tree structure of the computational complexity for the forward and backward procedures [79].

$$\beta_t(i) = P(o_{t+1}o_{t+2}...o_T \mid q_t=i,\lambda)$$

$$= \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\beta_{t+1}(j) \quad \textit{where} \quad t = T\text{-}1, T\text{-}2,...,1, \textit{ and } 1 \le i \le N \qquad (6\text{-}41)$$

The backward procedure's computational complexity is the same as the Forward procedure, $O(N^2 T)$, using the similar but opposite approach direction of Figure 47.

6.4.2 Optimal State Sequence Using the Dynamic Programming Approach

We use a dynamic programming method called the Viterbi algorithm [37,79,98] to find the single best state sequence $q = (q_1q_2...q_T)$ (or the most likely path) given the observable symbol sequence $O = (o_1o_2...o_T)$ and the HMM parameter set λ in order to maximize $P(q \mid O,\lambda)$. Since

$$P(q \mid O,\lambda) = \frac{P(q,O \mid \lambda)}{P(O \mid \lambda)} \qquad (6\text{-}42)$$

Maximizing $P(q \mid O,\lambda)$ is equivalent to maximizing $P(q,O \mid \lambda)$ using the Viterbi algorithm. The basic idea of the Viterbi algorithm (a dynamic programming method) is similar to the Forward procedure (Figure 46) whose calculation at each time is considered only between two consecutive times t and $t+1$, and starts at the initial time $t = 1$ and proceeds forward to the end time $t = T$. The major difference is during this calculation between two instant times. The control, which produces the maximum value corresponding to the single shortest or best path (state sequence), is "saved" instead of the summation of overall calculations (Figure 47). At the end of the state sequence for the calculation, the "remembered" best controls can be used to recover the state space trajectory based on path backtracking.

We define the maximum probability along a single best path at time t, which accounts for the first t observable symbols and ends in state i given the HMM parameter set λ, as

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t \mid \lambda) \tag{6-43}$$

We also define the "remembered" array $\psi_t(j)$ for each state j at time t, which keeps track of the argument that maximizes the value $\delta_{t-1}(i)$ x a_{ij}, in order to retrieve the optimal state sequence during the path backtracking. $\bar{q}_t$ is the single most likely state at time t.

The Viterbi Algorithm:

1. Initialization: The initial probability $\delta_1(i)$ is at state i, time $t = 1$ and initial observable symbol o_1 by given the HMM parameter set λ.

$$\delta_1(i) = P(o_1, q_1 = i \mid \lambda) = \pi_i b_i(o_1) \qquad \textit{where} \quad 1 \le i \le N \tag{6-44}$$

$$\psi_1(i) = 0 \qquad \textit{where} \quad 1 \le i \le N \tag{6-45}$$

$$\bar{q}_1 = \arg \max_{1 \le i \le N} \left(\delta_1(i)\right) \tag{6-46}$$

2. Recursion: The single best path (state sequence) among N possible paths from N possible states i at time t to state j at time t+1 with the state-transition probability a_{ij} is

$$\delta_{t+1}(j+1) = \max_{1 \le i \le N} P(o_1 o_2 \ldots o_{t+1}, q_{t+1} = j \mid \lambda)$$

$$= \left[\max_{1 \le i \le N} \left(\delta_t(i) a_{ij}\right)\right] b_j(o_{t+1}) \quad \textit{where} \quad 1 \le t \le T\text{-}1,\ 1 \le j \le N \tag{6-47}$$

$$\psi_{t+1}(j) = \arg \left[\max_{1 \le i \le N} \left(\delta_t(i) a_{ij}\right)\right] \qquad \textit{where} \quad 1 \le t \le T\text{-}1,\ 1 \le j \le N \tag{6-48}$$

$$\bar{q}_{t+1} = \arg \max_{1 \le i \le N} \left(\delta_{t+1}(i)\right) \tag{6-49}$$

3. Termination: The single best path reaches the end time T for each state sequence.

$$\bar{P} = \max_{1 \le i \le N} P(o_1 o_2 \ldots o_T,\ q_T = i \mid \lambda)$$

$$= \max_{1 \le i \le N} P(O,\ q_T = i \mid \lambda)$$

$$= \max_{1 \le i \le N} \left(\delta_T(i)\right) \tag{6-50}$$

$$\bar{q}_T = \arg \max_{1 \le i \le N} \left(\delta_T(i)\right) \tag{6-51}$$

4. Path (State Sequence) Backtracking: Backtracking is retrieving the path which have been saved as the most likely states.

$$\bar{q}_t = \psi_{t+1}(\bar{q}_{t+1}) \qquad \textit{where} \quad t = T\text{-}1, T\text{-}2, ..., 1 \tag{6-52}$$

For computation simplicity, the Viterbi algorithm can be implemented by additional preprocessing which takes the logarithms of the HMM parameters in order to convert multiplication to addition.

0. Preprocessing:

$$\pi_i^{\#} = \log(\pi_i) \qquad \textit{where} \quad 1 \le i \le N \tag{6-53}$$

$$a_{ij}^{\#} = \log(a_{ij}) \qquad \textit{where} \quad 1 \le i \le N \textit{ and } 1 \le t \le T \tag{6-54}$$

$$b_i^{\#}(o_t) = \log(b_i(o_t)) \qquad \textit{where} \quad 1 \le i \le N \textit{ and } 1 \le t \le T \tag{6-55}$$

1. Initialization:

$$\delta_1^{\#}(i) = \log(\delta_1(i)) = \pi_i^{\#} + b_i^{\#}(o_1) \qquad \textit{where} \quad 1 \le i \le N \tag{6-56}$$

$$\psi_1^{\#}(i) = 0 \qquad \textit{where} \quad 1 \le i \le N \tag{6-57}$$

$$\bar{q}_1^{\#} = \arg \max_{1 \le i \le N} \left(\delta_1^{\#}(i)\right) \tag{6-58}$$

2. Recursion:

$$\delta_{t+1}^{\#}(j) = \log(\delta_{t+1}(j))$$

$$= \left[\max_{1 \le i \le N} \left(\delta_t^{\#}(i) + a_{ij}^{\#}\right)\right] + b_j^{\#}(o_{t+1}) \qquad \textit{where} \quad 1 \le t \le T\text{-}1,\ 1 \le j \le N \tag{6-59}$$

$$\psi_{t+1}^{\#}(j) = \arg \left[\max_{1 \le i \le N} \left(\delta_t^{\#}(i) + a_{ij}^{\#}\right)\right] \qquad \textit{where} \quad 1 \le t \le T\text{-}1,\ 1 \le j \le N \tag{6-60}$$

$$\bar{q}_{t+1}^{\#} = \arg \max_{1 \le i \le N} \left(\delta_{t+1}^{\#}(i)\right) \tag{6-61}$$

3. Termination:

$$\bar{P}^{\#} = \max_{1 \le i \le N} \left(\delta_T^{\#}(i)\right) \tag{6-62}$$

$$\bar{q}_T^{\#} = \arg \max_{1 \le i \le N} \left(\delta_T^{\#}(i) \right) \tag{6-63}$$

4. Path (State Sequence) Backtracking:

$$\bar{q}_t^{\#} = \psi_{t+1}^{\#}(\bar{q}_{t+1}^{\#}) \qquad \textit{where} \quad t = T\text{-}1, T\text{-}2, ..., 1 \tag{6-64}$$

6.4.3 Parameter Estimation Using the Baum-Welch Method

We can use a set of training observable symbol sequences to adjust the model parameters in order to build a signal model that can be used to identify or recognize other sequences of observable symbols. There is, however, no efficient way to optimize the model parameter set that globally maximizes the probability of the symbol sequence. Therefore, the Baum-Welch method [6] is used for choosing the maximum likelihood model parameter set $\lambda = (\pi, A, B)$ such that its likelihood function $P(O \mid \lambda)$ is locally maximized using an iterative procedure.

To easily describe the procedure for reestimation (iterative computation) of the HMM parameter set $\lambda = (\pi, A, B)$, we define a posterior probability variable $\gamma_t(i)$, shown in Figure 48, as the probability of being in state i at time t by given the HMM parameter set λ and the entire observable symbol sequence O.

$$\gamma_t(i) = P(q_t = i \mid O, \lambda) = \frac{P(O, q_t = i \mid \lambda)}{P(O \mid \lambda)} = \frac{P(O, q_t = i \mid \lambda)}{\sum_{i=1}^{N} P(O, q_t = i \mid \lambda)}$$

$$= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{6-65}$$

where

$$P(O, q_t = i \mid \lambda) = \alpha_t(i)\, \beta_t(i) \tag{6-66}$$

$$\alpha_t(i) = P(o_1 o_2 ... o_t,\ q_t = i \mid \lambda) \tag{6-67}$$

$$\beta_t(i) = P(o_{t+1} o_{t+2} ... o_T \mid q_t = i, \lambda) \tag{6-68}$$

We define the other probability variable $\xi_t(i,j)$ (illustrated in Figure 49), which represents the probability of being in state i at time t, and state j at time t+1 given the observable symbol sequence O and the HMM parameter set λ.

$$\begin{aligned}\xi_t(i,j) &= P(q_t = i, q_{t+1} = j \mid O, \lambda) \\ &= \frac{P(q_t = i, q_{t+1} = j, O \mid \lambda)}{P(O \mid \lambda)} \\ &= \frac{P(q_t = i, q_{t+1} = j, O \mid \lambda)}{\sum_{i=1}^{N} P(O, q_t = i \mid \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}\end{aligned} \tag{6-69}$$

Then the relationship between $\gamma_t(i)$ and $\xi_t(i,j)$ is

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{6-70}$$

If we sum $\gamma_t(i)$ and $\xi_t(i,j)$ from the initial time t = 1 to the time $t = T$-1, we can find

$$\sum_{t=1}^{T-1} \gamma_t(i) \text{ = expected number of transitions or times, } i.e.\text{, frequency, from state } i \text{ given observable symbol sequence } O \tag{6-71}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) \text{ = expected number of transitions from state } i \text{ to state } j \text{ given } O \tag{6-72}$$

A set of reasonable reestimation formulas for HMM parameters π, A, and B is given

$$\begin{aligned}\pi_i &= \text{expected number of transitions in state } i \text{ at time } t\ (=1) \\ &= \gamma_1(i) = \frac{P(O, q_1 = i \mid \lambda)}{P(O \mid \lambda)} \\ &= \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N} \alpha_T(i)}\end{aligned} \tag{6-73}$$

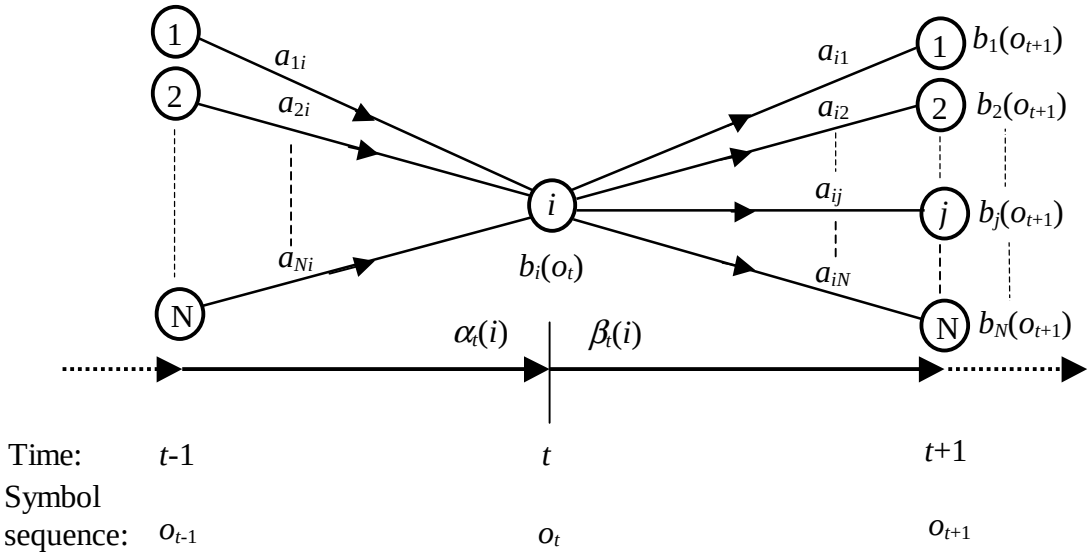


Figure 48 A posterior probability variable $\gamma_t(i)$ which is the probability of being in state i at time t by given the HMM parameter set λ and the entire observable symbol sequence O.

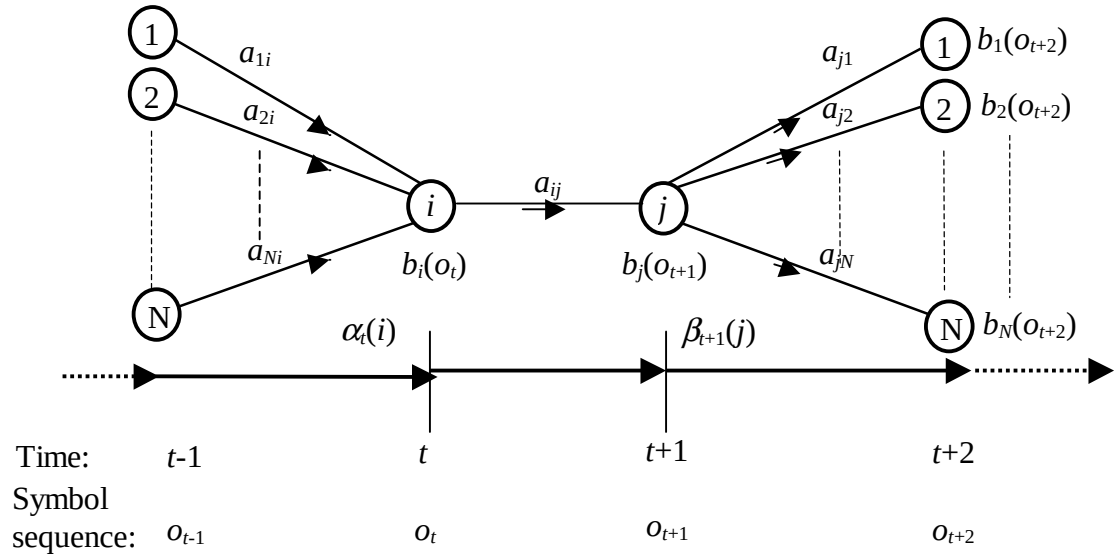


Figure 49 The probability variable $\xi_t(i,j)$ which represents the probability of being in state i at time t, and state j at time t+1 given the observable symbol sequence O and the HMM parameter set λ.

$$a_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{exptected number of transitions in state } i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} P(q_t = i, q_{t+1} = j, O \mid \lambda)}{\sum_{t=1}^{T-1} P(q_t = i, O \mid \lambda)}$$

$$= \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \qquad (6\text{-}74)$$

$$b_j(o_t) = \frac{\text{expected number of transitions in state } j \text{ and observable symbol } o_t \text{ at time } t}{\text{expected number of transitions in state } j}$$

$$= \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \overset{s.t.\ O_t = o_t}{=} \frac{\sum_{t=1}^{T} P(q_t = j, O \mid \lambda) \delta(O_t, q)}{\sum_{t=1}^{T} P(q_t = j, O \mid \lambda)}$$

$$= \frac{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j) \delta(O_t, o_t)}{\sum_{t=1}^{T} \alpha_t(j) \beta_t(j)} \qquad \text{where } \delta(O_t, o_t) = \begin{cases} 1 & \text{if } O_t = o_t \\ 0 & \text{otherwise} \end{cases} \qquad (6\text{-}75)$$

where

$$P(q_t = i, O \mid \lambda) = \alpha_t(i)\, \beta_t(i) \qquad (6\text{-}76)$$

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i) = \sum_{i=1}^{N} \alpha_T(i) \qquad (6\text{-}77)$$

$$P(q_t = i, q_{t+1} = j, O \mid \lambda) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \qquad (6\text{-}78)$$

Note that these updated parameters should satisfy stochastic constraints for computation normalization.

$$\sum_{i=1}^{N} \pi_i = 1 \qquad (6\text{-}79)$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad where \quad 1 \le i \le N \tag{6-80}$$

$$\sum_{o_t=0}^{M-1} b_j(o_t) = 1 \quad where \quad 1 \le j \le N \qquad and \qquad 1 \le t \le T \tag{6-81}$$

6.5 Computation Considerations

To be able to enhance the effectiveness of HMM performance in the practical applications, such as facial expression recognition, it is necessary to have accurate computation and guarantee the local maximum of the likelihood function using an iterative procedure (reestimation procedure), *i.e.* the Forward-Backward procedure, convergence.

6.5.1 Choice of Hidden Markov Model

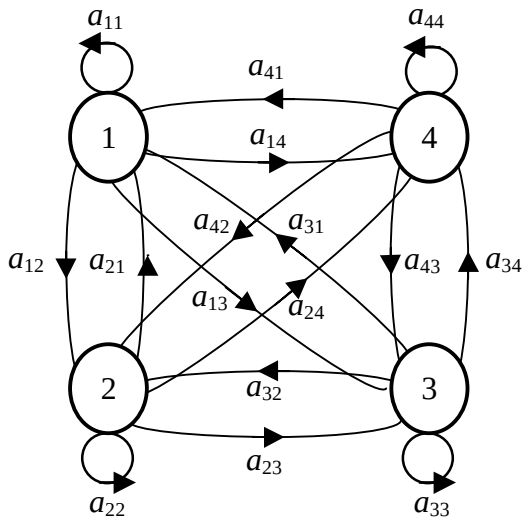
There are several types of HMMs [79] such as the ergodic model (Figure 50.a) in which every state of the model can be reached in a single step from any state of the model. Its state-transition probability matrix A is a full matrix.

$$A = \{a_{ij}\} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \tag{6-82}$$

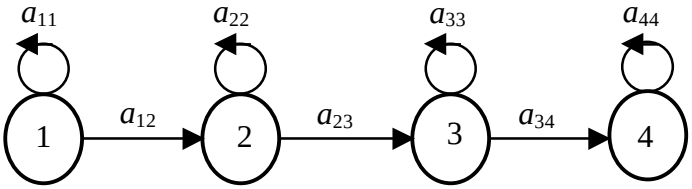
where

$$a_{ij} \ge 0, \qquad \sum_{j=1}^{N} a_{ij} = 1 \qquad and \qquad 1 \le i \le N \tag{6-83}$$

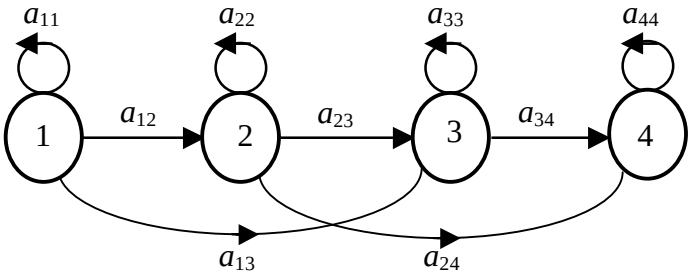
The left-right model (the Bakis model) which is used for facial expression recognition in various lengths of image sequences because they perform well in the spatio-temporal domain and are able to deal with the time warping problem. The left-right type of HMM



(a)



(b)



(c)

Figuer 50 (a) 4 state ergodic HMM (b) 1st-order 4-state left-right (Bakis) HMM (c) 2nd-order 4-state left-right HMM.

has the desirable properties that the initial state probability has the characteristic

$$\pi_i = \begin{cases} 0 & \text{if } i \neq 1 \\ 1 & \text{if } i = 1 \end{cases} \qquad \text{where } 1 \leq i \leq N \tag{6-84}$$

The state sequence must begin at the first state 1 with left to right order and end at the final state *N*. As the time increases, the observable symbols in each sequence either stay at the same state or increase in a successive manner. The state-transition coefficients of the left-right model have the property

$$a_{ij} = 0 \qquad \text{if } j < i \tag{6-85}$$

No transitions can occur from the current state to a state with a lower index. An additional constraint for the state-transition coefficients of the left-right HMM is

$$a_{ij} = 0 \qquad \text{if } j > i + \Delta i \tag{6-86}$$

for some value Δi which means the order of the left-right HMM. No jumps of more than Δi number of states are allowed. For example, $\Delta i = 1$ and $N = 4$ means the 1st-order 4-state left-right HMM (Figure 50.b) whose state-transition probability matrix *A* is

$$A=\{a_{ij}\}=\begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{6-87}$$

where

$$a_{ij} \geq 0, \qquad \sum_{j=1}^{4} a_{ij} = 1 \qquad \textit{and} \qquad 1 \leq i \leq 4 \tag{6-88}$$

$\Delta i = 2$ and $N = 4$ indicates the 2nd-order 4-state left-right HMM (Figure 50.c) whose state-transition probability matrix *A* is

$$A=\{a_{ij}\}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{6-89}$$

where

$$a_{ij} \geq 0, \qquad \sum_{j=1}^{4} a_{ij} = 1 \qquad and \qquad 1 \leq i \leq 4 \tag{6-90}$$

6.5.2 Initialization of Hidden Markov Model Parameter Estimation

For the training process, accurate initial estimations of the HMM parameters π, A and B will help the local maximum approach using an iteration procedure as close as possible to the global maximum of the likelihood function. If the elements of the parameters are set to zero initially, they will remain at zero during the entire process. In practice, it is not important to reach the global maximum of the likelihood function. Instead, finding a set of HMM parameters which promote a highly accurate recognition result is more important. Our experiment shows that very small random initial values of the HMM parameters (smaller than 10^{-6}) are adequate and useful and recognition accuracy is high. Remember that for the left-right HMM, the initial state probability for the first state is 1 ($\pi_1 = 1$) and other states are 0.

6.5.3 Computation of Scaling

The forward and backward variables $\alpha_t(i)$ and $\beta_t(i)$ are computed recursively, so they are composed of a large number of accumulated $\{a_{ij}\}$ and $\{b_j(o_t)\}$ multiplications. Since each a_{ij} and $b_j(o_t)$ is less (or extensively less) than 1, each term of $\alpha_t(i)$ or $\beta_t(i)$ will start to head exponentially to zero when the length (or number of frames) T in each image sequence increases. The dynamic range of the $\alpha_t(i)$ or $\beta_t(i)$ computation will then be beyond the precision range of any computer capability. To keep $\alpha_t(i)$ and $\beta_t(i)$ within the dynamic range of the computer, the most straightforward method is to multiply $\alpha_t(i)$ and $\beta_t(i)$ by scaling coefficients which can be canceled out completely at the end of the computation. The scaling coefficient for $\alpha_t(i)$ is defined as c_t [79]

$$\sum_{i=1}^{N} c_t \alpha_t(i) = c_t \sum_{i=1}^{N} \alpha_t(i) = 1 \quad where \quad 1 \le t \le T \tag{6-91}$$

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{6-92}$$

The scaling coefficient c_t is the inverse of the sum overall states N of $\alpha_t(i)$ at time t, is dependent only on time t and independent of state i, and effectively rebuilds the magnitude of the $\alpha_t(i)$ to 1. To keep the $\beta_t(i)$ computation within reasonable bounds as $\alpha_t(i)$, we can apply the same scaling coefficient c_t to $\beta_t(i)$ because the magnitudes of the $\alpha_t(i)$ and $\beta_t(i)$ are comparable. For efficient computation within rational bounds, the $\alpha_t(i)$ and $\beta_t(i)$ should be replaced by

$$\tilde{\alpha}_t(i) = c_t \alpha_t(i) = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)} \tag{6-93}$$

$$\tilde{\beta}_{t+1}(j) = c_{t+1} \beta_{t+1}(j) = \frac{\beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_{t+1}(i)} \tag{6-94}$$

where $\tilde{\alpha}_t(i)$ and $\tilde{\beta}_{t+1}(j)$ are the scaling results of $\alpha_t(i)$ and $\beta_{t+1}(j)$, respectively.

In addition, when $\tilde{\alpha}_t(i)$ and $\tilde{\beta}_{t+1}(j)$ are applied at the scaling intermediate probability $\tilde{\gamma}_t(i)$ and $\tilde{\xi}_t(i,j)$.

$$\tilde{\gamma}_t(i) = \frac{\tilde{\alpha}_t(i)\tilde{\beta}_t(i)}{\sum_{i=1}^{N} \tilde{\alpha}_t(i)\tilde{\beta}_t(i)} = \frac{(c_t\alpha_t(i))(c_t\beta_t(i))}{\sum_{i=1}^{N} (c_t\alpha_t(i))(c_t\beta_t(i))} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} = \gamma_t(i) \tag{6-95}$$

$$\tilde{\xi}_t(i,j) = \frac{\tilde{\alpha}_t(i) a_{ij} b_j(o_{t+1}) \tilde{\beta}_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \tilde{\alpha}_t(i) a_{ij} b_j(o_{t+1}) \tilde{\beta}_{t+1}(j)} = \frac{(c_t\alpha_t(i)) a_{ij} b_j(o_{t+1}) (c_t\beta_{t+1}(j))}{\sum_{i=1}^{N}\sum_{j=1}^{N} (c_t\alpha_t(i)) a_{ij} b_j(o_{t+1}) (c_t\beta_{t+1}(j))}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} = \xi_t(i,j) \tag{6-96}$$

The numerator and denominator terms are both deleted because the scaling coefficient c_t is independent of states *i* and *j*. The scaling intermediate probability $\tilde{\gamma}_t(i)$ and $\tilde{\xi}_t(i,j)$ has the same values as the intermediate probability without scaling $\gamma_t(i)$ and $\xi_t(i,j)$. Furthermore, the HMM parameters π, *A*, and *B* will also keep the same probability values when both $\alpha_t(i)$ and $\beta_{t+1}(j)$ are scaled, because those parameters are constructed from either or both intermediate probability $\gamma_t(i)$ and $\xi_t(i,j)$.

$$\tilde{\pi}_i = \pi_i, \qquad \tilde{a}_{ij} = a_{ij}, \qquad \text{and} \qquad \tilde{b}_j(o_t) = b_j(o_t) \tag{6-97}$$

The only real affected event of the HMM procedure by scaling coefficient is computing the maximum likelihood function $P(O \mid \lambda)$. Since [79]

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = 1 \tag{6-98}$$

where scaling coefficient c_t is independent of state *i*. So

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) = \frac{1}{\prod_{t=1}^{T} c_t} = \frac{1}{\prod_{t=1}^{T} \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}} \tag{6-99}$$

The computation for the denominator term of the maximum likelihood function $P(O \mid \lambda)$ will be extremely small, which is out of the dynamic range of the computer's ability. It can be solved by taking logarithms.

$$\log P(O \mid \lambda) = -\sum_{t=1}^{T} \log c_t = \sum_{t=1}^{T} \log \sum_{i=1}^{N} \alpha_t(i) \tag{6-100}$$

This can be used for evaluating the most likely performance of any input data corresponding to a set of HMMs. The Viterbi algorithm for finding the maximum likelihood state sequence also uses the logarithms, which will be within the dynamic bounds of the computer, so no scaling process is needed.

6.5.4 Computation of Smoothing for Insufficient Training Data

The amount of data (observable symbol sequences) used to train an HMM is always limited because of considerations of the computational cost of HMM training or the availability of training data. Thus, there is always an inadequate number of occurrences of low-probability state transitions between states and observable symbols within states to give reasonable estimates of the model parameters. If the training number of the observable symbol sequences are so small that they do not have any occurrences simultaneously to satisfy the conditional probability of HMM parameters, then $\tilde{\pi}_i = 0$, $\tilde{a}_{ij} = 0$, and $\tilde{b}_j(o_t) = 0$ will occur and will stay at 0 after each reestimation. When this resultant model is employed to evaluate any other observable symbol sequence which possibly contains the state transitions or the observable symbol that do not occur in the training sequences, this model will produce a zero probability result for this evaluated observable symbol sequence. Such a singular outcome is certainly a consequence of the unreliable estimation that $\tilde{\pi}_i = 0$, $\tilde{a}_{ij} = 0$, and $\tilde{b}_j(o_t) = 0$ due to the insufficiency of the training data to cover all possible varieties.

There are many possible solutions for handling the effects of insufficient training data [59,79], such as production of the codebook size (reduction of the number of observable symbols at each state) or the number of states. The simplest and most practical way for combating the insufficient training data problem is to add the numeric floor ε for smoothing the probability distributions of HMM parameters in order to ensure that no model parameter estimation falls below a specified threshold ε for each iterative estimation.

$$\tilde{\pi}_i = \begin{cases} \tilde{\pi}_i & if \quad \tilde{\pi}_i \geq \varepsilon_{\tilde{\pi}} \\ \varepsilon_{\tilde{\pi}} & if \quad \tilde{\pi}_i < \varepsilon_{\tilde{\pi}} \end{cases} \quad where \quad \varepsilon_{\tilde{\pi}} > 0.0 \tag{6-101}$$

$$\tilde{a}_{ij} = \begin{cases} \tilde{a}_{ij} & if \quad \tilde{a}_{ij} \geq \varepsilon_{\tilde{a}} \\ \varepsilon_{\tilde{a}} & if \quad \tilde{a}_{ij} < \varepsilon_{\tilde{a}} \end{cases} \quad where \quad \varepsilon_{\tilde{a}} > 0.0 \tag{6-102}$$

$$\tilde{b}_i(o_t) = \begin{cases} \tilde{b}_i(o_t) & \text{if } \tilde{b}_i(o_t) \geq \varepsilon_{\tilde{b}} \\ \varepsilon_{\tilde{b}} & \text{if } \tilde{b}_i(o_t) < \varepsilon_{\tilde{b}} \end{cases} \qquad \text{where } \varepsilon_{\tilde{b}} > 0.0 \tag{6-103}$$

In our study, $\varepsilon_{\tilde{\pi}} = \varepsilon_{\tilde{a}} = \varepsilon_{\tilde{b}} = 0.0001$.

6.5.5 Computation of Normalization

Each probability distribution of HMM parameters, which consist of conditional probability, should satisfy the stochastic constraints at each iteration estimation.

$$\sum_{i=1}^{N} \tilde{\pi}_i = 1 \tag{6-104}$$

$$\sum_{j=1}^{N} \tilde{a}_{ij} = 1 \qquad \text{where } 1 \leq i \leq N \tag{6-105}$$

$$\sum_{o_t=0}^{M-1} \tilde{b}_j(o_t) = 1 \quad \text{where } 1 \leq j \leq N \tag{6-106}$$

Since the probability of each parameter is reestimated at each iteration, the conflict with the stochastic constraints always occurs. The sum of the above equation is not equal to 1, particularly if the probability distribution of each parameter are smoothed by a numeric floor at each iterative estimation. Therefore, it is necessary to normalize the probability distributions of the HMM parameters so that the densities obey the required stochastic constraints after each iteration of parameter reestimation and smoothing.

$$\tilde{\pi}_i = \frac{\tilde{\pi}_i}{\sum_{k=1}^{N} \tilde{\pi}_k} \qquad \text{where } 1 \leq i \leq N \tag{6-107}$$

$$\tilde{a}_{ij} = \frac{\tilde{a}_{ij}}{\sum_{k=1}^{N} \tilde{a}_{ik}} \qquad \text{where } 1 \leq i,j \leq N \tag{6-108}$$

$$\tilde{b}_j(o_t) = \frac{\tilde{b}_j(o_t)}{\sum_{k_t=0}^{M-1} \tilde{b}_j(k_t)} \qquad where \;\; 1 \le j \le N \;\; and \;\; 0 \le o_t \le M\text{-}1 \tag{6-109}$$

6.5.6 Computation of Convergence

Since the Forward-Backward procedure is based on local maxima estimation by iterative computation to achieve global maximum, it is important to guarantee that the reestimated parameter set $\tilde{\lambda} = (\tilde{\pi}, \tilde{A}, \tilde{B})$ is convergent. It is necessary to prove that model $\tilde{\lambda}^{i+1} = (\tilde{\pi}^{i+1}, \tilde{A}^{i+1}, \tilde{B}^{i+1})$ following the *i*+1th iterative reestimation is either equal or more likely than model $\tilde{\lambda}^{i} = (\tilde{\pi}^{i}, \tilde{A}^{i}, \tilde{B}^{i})$ at current *i*th reestimation in the sense that $P(O \mid \tilde{\lambda}^{i+1}) \ge P(O \mid \tilde{\lambda}^{i})$. In other words, if the model $\tilde{\lambda}^{i}$ is replaced by $\tilde{\lambda}^{i+1}$ and this reestimation is repeated, then the probability of symbol sequence *O* being observed from the given update model $\tilde{\lambda}^{i+1}$ is improved until some limiting point is reached. The final result of this reestimation procedure is a maximum likelihood estimation of the HMM.

Because

$$P(O \mid \tilde{\lambda}) = \sum_{q_1 \ldots q_T} P(O, q \mid \tilde{\lambda}) \tag{6-110}$$

Then an auxiliary function [6,79] is defined as

$$Q(\tilde{\lambda}^{i}, \tilde{\lambda}^{i}) = \sum_{q_1 \ldots q_T} P(O, q \mid \tilde{\lambda}^{i}) \;\log\; P(O, q \mid \tilde{\lambda}^{i}) \tag{6-111}$$

$$Q(\tilde{\lambda}^{i}, \tilde{\lambda}^{i+1}) = \sum_{q_1 \ldots q_T} P(O, q \mid \tilde{\lambda}^{i}) \;\log\; P(O, q \mid \tilde{\lambda}^{i+1}) \tag{6-112}$$

over $\tilde{\lambda}^{i+1}$. Since

$$Q(\tilde{\lambda}^{i}, \tilde{\lambda}^{i+1}) \ge Q(\tilde{\lambda}^{i}, \tilde{\lambda}^{i}) \Rightarrow P(O \mid \tilde{\lambda}^{i+1}) \ge P(O \mid \tilde{\lambda}^{i}) \tag{6-113}$$

we can maximize the auxiliary function $Q(\tilde{\lambda}^i, \tilde{\lambda}^{i+1})$ over $\tilde{\lambda}^{i+1}$ to the better $\tilde{\lambda}^i$ in order to optimize the likelihood function $P(O \mid \tilde{\lambda})$. By iterating the procedure, the likelihood function eventually converges to a critical point.

6.5.7 Computation of Confidence

Unlike the artificial neural networks, even though there is no learnable mapping between input and output from the training process, HMM still can generate a satisfactory input-output confidence (mapping) for the recognition process (because of the computation consideration in section 6.5.4). The output of the HMM, $P(O \mid \tilde{\lambda})$, is a probability instead of one taking all. If the output probability is close to 1, it indicates that the input symbol sequence has high confidence (similarity) with the training model. If the output probability is close to 0, it implies the input symbol sequence has low confidence with the training model. The highest output probability among all training models is always chosen to be the recognition result and the recognition confidence is evaluated as well.

7.0 DETERMINATION OF HIDDEN MARKOV MODEL TOPOLOGY

An HMM topology is defined as the statistical behavior of an observable symbol sequence in terms of a network of states, which represents the overall process behavior with regard to movement between states of the process, and describes the inherent variations in the behavior of the observable symbols within a state. An HMM topology, then, consists of the number of states with varied connections between states which depend on the occurrence of the observable symbol sequences being modeled. Each state represents a similarly-behaving portion of an observable symbol sequence process, such as phonemes to speech [59,76,79] and facial features to face identification [84]. The variety of the observable symbols for which the HMM uses a particular state is described in terms of the distribution of probability that each observable symbol will occur from that state. At each instant of time, the observable symbol in each sequence either stays at the same state (called self-transition) or moves to another state based on a set of state-transition probability associated between the states, which models the duration of each similarly-behaving portion of the process. So, to determine the HMM topology is to choose the types of HMM, such as ergodic, left-right or some others, and to decide the number of states and the connections between states. To accomplish this, it is necessary to understand the physical meaning of states represented by the corresponding observable symbols.

7.1 The Method

If different HMM topologies are equivalent in their abilities to generate the same recognition performance, then the simplest topology of these models is the best with the fewest elements of model parameters. If the number of elements of model parameters for

an HMM is unnecessarily large, then its topology will not efficiently represent the performance of the training data, and the model will incur unnecessary computational cost.

There exists no simple and theoretical method for determining the optimum topology for an HMM. Currently, most HMM topologies are determined by experiments using iterative trial-and-error processes, without discussing the physical meaning to support their determinations of HMM topologies. To optimize the HMM recognition performance, we develop a method to determine the HMM topology for our facial expression recognition.

Facial expressions are recognized in the context of the entire image sequence of an arbitrary length, so the left-right (or Bakis) model is employed to model the image sequences whose properties change over time in a successive manner.

7.1.1 Step 1: The 1st-Order Markov Model

Since HMM is derived from a (observable) Markov model, the method for determining the HMM topology is to start from the 1st-order left-right Markov model. The rule is:

Rule 1.1: The same and successive observable symbols of portion of each sequence will occupy at one and only one state of the 1st-order left-right Markov model and states are connected in order as the symbols in each sequence.

Example:

Based on a codebook of size M = 16, there is a training observable symbol sequence with 3 different symbols o_t , $0 \le o_t \le M\text{-}1$, in order $o_{t=1\sim6}$ = 14, $o_{t=7\sim8}$ = 6, and $o_{t=9\sim11}$ = 9, where t is the instant time (or frame number), and 11 frames ($1 \le t \le 11 = T$) in Figure 51. This symbol sequence can be represented by a 1st-order 3-state left-right Markov model where each state is occupied by an individual symbol.

Subject Number: 111

Expressions: AU12

Codebook Size: $M = 16$

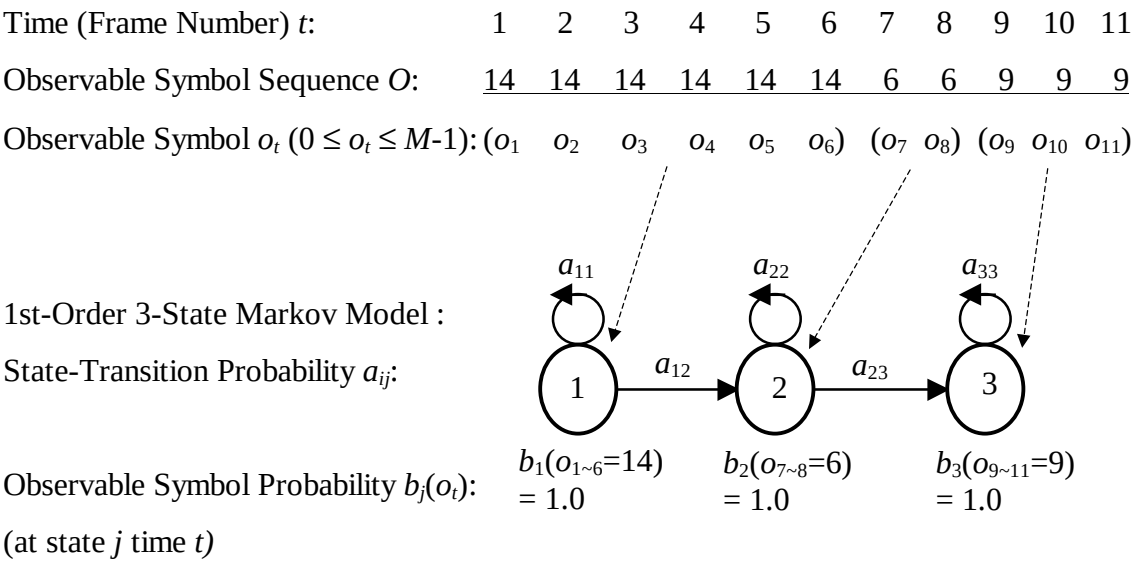


Figure 51 A 1st-order 3-state Markov Model used to represent the observable symbol sequence.

7.1.2 Step 2: The 1st-Order Hidden Markov Model

The Markov model can be seen as a special case of an HMM having redundant states. That is, for each state j at time t, the observable symbol probability $b_j(o_t)$ has one 1.0, others are 0.0. In HMM, each state tolerates probability distributions of different observable symbols occurring at the same time. This HMM property allows us to combine different 1st-order left-right Markov models into one 1st-order left-right HMM by eliminating the redundant states without losing the individual performance. The rules are as following:

Rule 2.1: If the observable symbols of each training sequence are shown only at the initial state or at the last state of the 1st-order left-right Markov models, then these observable symbols should be at the first state or at the last state of the 1st-order left-right HMM.

Rule 2.2: Each symbol having higher occupied probability among total observable symbols from all 1st-order left-right Markov models will dominate one state of the 1st-order left-right HMM. Each symbol having the lower occupied probability among all observable symbols can be combined (absorbed) by its neighbor states having the higher occupied probability among all observable symbols at the 1st-order left-right HMM.

Rule 2.3: Symbols having the lower occupied probability among all observable symbols taken from consecutive states at the 1st-order left-right Markov model can combine to take one state at the 1st-order left-right HMM. If the occupied probability of the sum of these symbols at this state at the 1st-order left-right HMM is still lower, then as Rule 2.2, this state can be combined by its neighbor states having the higher occupied probability among all observable symbols at the 1st-order left-right HMM.

Rule 2.4: According to all 1st-order left-right Markov models, we can count the number of connections (or state transitions) between two states as the connective intensity between these two states. More numbers of connections have stronger connective intensity between both states. Fewer numbers of connections have weaker connective intensity between both states. During the combination process, the connection between both states having stronger connective intensity should keep the same status as that at the 1st-order left-right Markov model. If the connection between both states having weaker connective intensity will be easily broken, then either both states will combine to become one state, or each state is combined by its neighbor state having the

higher occupied probability among all observable symbols at the 1st-order left-right HMM.

Rule 2.5: The number of states at the 1st-order left-right HMM should be no more than the maximum number of states among all 1st-order left-right Markov models.

Rules 2.2 and 2.3 are created according to the following HMM property. For each state, if its self-transition probability is much lower than other state-transition probability, then this state can be combined by its neighbor states without losing its original performance. This is because without the self-transition, a state is merely an intermediate state between other states. It cannot represent a meaningful portion of the process. The meaningful portion of the process is represented by the probability of observable symbols at the HMM.

The following examples demonstrate execution of the rules to combine different 1st-order left-right Markov models into one 1st-order left-right HMM for each "expression unit." According to the number of total 1212 training images for recognition of the lower facial expression using the dense flow tracking with principal component analysis (PCA) method, the codebook size is $M = 16$ ($2^4 < 1212/50 < 2^5$). The observable symbol o_t at time t will be $0 \leq o_t \leq 15$.

Example 1:

In Figure 52 for "expression unit" AU12, there are twelve 1st-order left-right Markov models with three different symbols having high occupied probability (14: 0.377, 6: 0.293, and 9: 0.330). Using the rules:

Rule 2.1: Observable symbol 14 should be at the first state and observable symbol 9 should be at the last state of the 1st-order left-right HMM, since symbols 14 and 9 are at the first state and the last state of each Markov model, respectively, with high occupied probability.

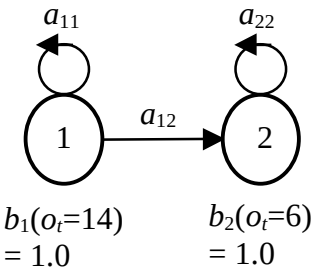
Rule 2.4: Symbol 6 has very strong connective intensity with symbol 14. It also has very

The overall probability for each observable symbol from all 12 different symbol sequences of the facial expression AU12: (O = {o_t=6, o_t=9, o_t=14})

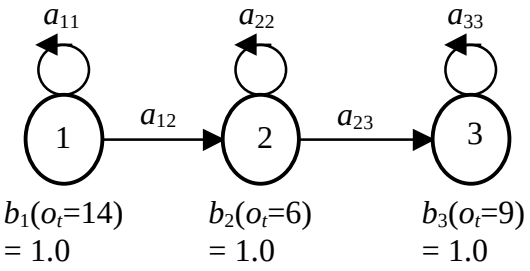
Major Symbols:
$P(o_t = 6|O) = 0.293$
$P(o_t = 9|O) = 0.330$
$P(o_t = 14|O) = 0.377$

Step 1: The 1st-Order Markov Model

There are 5
1st-order 2-state Markov Models:



There are 7
1st-order 3-state Markov Models:



Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 3-state Hidden Markov Model.

The major observable symbols for each state:

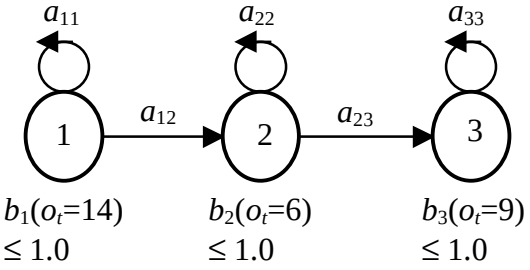


Figure 52 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU12.

strong connective intensity with symbol 9 but in the opposite direction.

Symbol 9 does not have any connective intensity with symbol 14.

We can use a 1st-order 3-state left-right HMM with symbols 14, 6 and 9 for states 1, 2, and 3, respectively, to represent all Markov models for "expression unit" AU12.

Example 2:

In Figure 53 for "expression unit" AU12, there are twelve 1st-order left-right Markov models with three different symbols having high occupied probability (2: 0.385, 9: 0.372, and 1: 0.243).

Rule 2.1: Symbols 2 and 1 should be at the first state and the last state of the 1st-order left-right HMM, since both have high occupied probability and are at the first state and last state for each Markov model.

Rule 2.2 & 2.4: Symbol 9 is either at the middle or last state at Markov model. Symbol 2 has stronger connective intensity with symbol 9 than that of symbol 1, so the connection between symbol 2 and symbol 1 will be easy to break.

Based on the above analysis, we insert symbol 9 between symbols 1 and 2 by breaking the connection between symbols 2 and 1. That is, we can model the training image sequences of "expression unit" AU12 by a 1st-order 3-state HMM whose major probability of observable symbols for state 1 is 2, for state 2 is 9, and for state 3 is 1.

Example 3:

In Figure 54 for "expression unit" AU15+17, there are thirteen 1st-order left-right Markov models with two major symbols having higher occupied probability (2: 0.428 and 6: 0.459) and with two minor symbols having lower occupied probability (1: 0.082 and 12: 0.031).

Rule 2.1: Symbol 2 should be at the initial state and symbols 1 and 6 should be at the last state of the 1st-order left-right HMM.

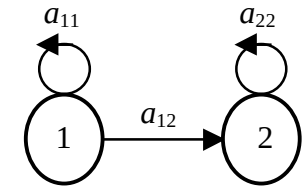
Rule 2.2: Since symbols 1 and 6 should be at the last state of the 1st-order left-right

The overall probability for each observable symbol from all 12 different symbol sequences of the facial expression AU12: (O = {o_t=1, o_t=2, o_t=9})

Major Symbols:
$P(o_t = 1|O) = 0.243$
$P(o_t = 2|O) = 0.385$
$P(o_t = 9|O) = 0.372$

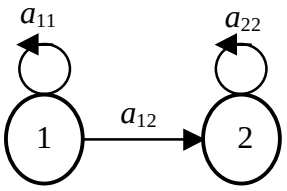
Step 1: The 1st-Order Markov Model -

There are 7
1st-order 2-state Markov Models:



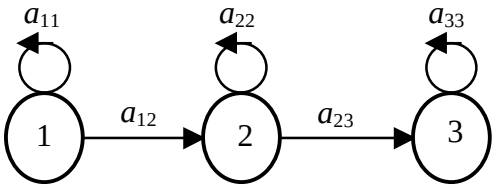
$b_1(o_t=2) = 1.0$ $b_2(o_t=9) = 1.0$

There are 4
1st-order 2-state Markov Models:



$b_1(o_t=2) = 1.0$ $b_2(o_t=1) = 1.0$

There is 1
1st-order 3-state Markov Model:

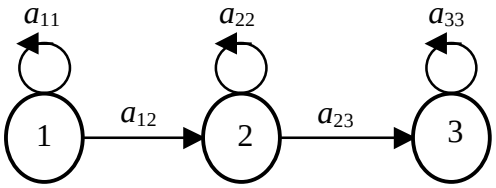


$b_1(o_t=2) = 1.0$ $b_2(o_t=9) = 1.0$ $b_3(o_t=1) = 1.0$

Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 3-state Hidden Markov Model.

The major observable symbols for each state:



$b_1(o_t=2) \leq 1.0$ $b_2(o_t=9) \leq 1.0$ $b_3(o_t=1) \leq 1.0$

Figure 53 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU12.

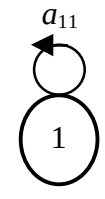
The overall probability for each observable symbol from all 13 different symbol sequences of the facial expression AU15+17: (O = {o_t=1, o_t=2, o_t=6, o_t=12})

Major Symbols:
$P(o_t= 2|O) = 0.428$
$P(o_t= 6|O) = 0.459$

Minor Symbols:
$P(o_t= 1|O) = 0.082$
$P(o_t= 12|O) = 0.031$

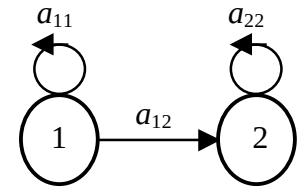
Step 1: The 1st-Order Markov Model

There are 2
1st-order 1-state Markov Models:



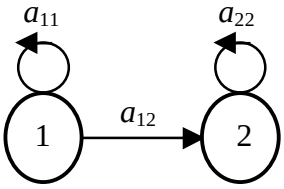
$b_1(o_t=2) = 1.0$

There are 2
1st-order 2-state Markov Models:



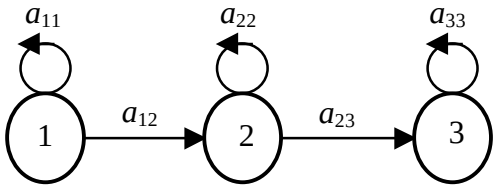
$b_1(o_t=2) = 1.0$ $b_2(o_t=1) = 1.0$

There are 7
1st-order 2-state Markov Models:



$b_1(o_t=2) = 1.0$ $b_2(o_t=6) = 1.0$

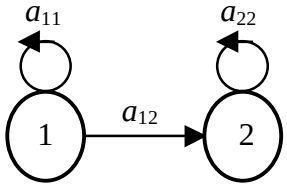
There are 2
1st-order 3-state Markov Models:



$b_1(o_t=2) = 1.0$ $b_2(o_t=12) = 1.0$ $b_3(o_t=6) = 1.0$

Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 2-state Hidden Markov Model.



The major observable symbols for each state:

$b_1(o_t=2) \leq 1.0$ $b_2(o_t=6) \leq 1.0$

Figure 54 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU15+17.

HMM and symbol 6 has a much higher occupied probability than that of symbol 1, symbol 1 can be combined by symbol 6 and be at the same last state of the HMM.

Rule 2.2 & 2.4: Symbols 2 and 6 have very strong connective intensity compared with that between symbols 2 and 12 which has lower occupied probability. So symbol 12 can be combined by its neighbor states taken by symbols 2 and 6 with higher occupied probability.

We use a 1st-order 2-state HMM with major symbols 2 and 6 for the first state and the second (last) state, respectively.

Example 4:

In Figure 55 for "expression unit" AU15+17, there are twelve 1st-order left-right Markov models with three major symbols having higher occupied probability (2: 0.539, 6: 0.269, and 14: 0.159) and with two minor symbols having lower occupied probability (1: 0.028 and 12: 0.005).

Rule 2.1: Symbol 2 should be only at the initial state, and symbols 1 and 14 should be only at the last state of the 1st-order left-right HMM.

Rule 2.2: Symbol 1 (lower occupied probability) will be combined by symbol 14 (higher occupied probability) and both are at the last state of the 1st-order left-right HMM.

Rule 2.4: Symbols 2 and 6 have very strong connective intensity compared with those of symbol 2 and 14, and 2 and 12 (lower occupied probability), so symbols 2 and 6 will strongly connect together by combined symbol 12.

According to the above analysis, three major occupied probability (2, 6, and 14) should have their own states at HMM, since symbols 2 and 14 should dominate the first state and last state, respectively. Symbol 6 has strong connective intensities between symbol 2 and 14, so symbol 6 will be at the middle state of the HMM. That is, a 1st-order 3-state left-right HMM will be applied with major symbols 2, 6, and 14 at states 1, 2, and 3,

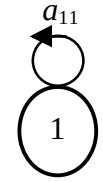
The overall probability for each observable symbol from all 12 different symbol sequences of the facial expression AU15+17: (O = {o_t=1, o_t=2, o_t=6, o_t=12, o_t=14})

Major Symbols:
$P(o_t= 2|O) = 0.539$
$P(o_t= 6|O) = 0.269$
$P(o_t= 14|O) = 0.159$

Minor Symbols:
$P(o_t= 1|O) = 0.028$
$P(o_t= 12|O) = 0.005$

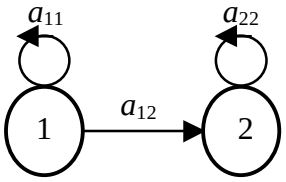
Step 1: The 1st-Order Markov Model

There are 3 1st-order 1-state Markov Models:



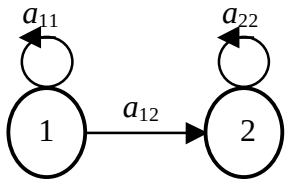
$b_1(o_t=2) = 1.0$

There is 1 1st-order 2-state Markov Model:



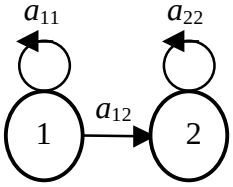
$b_1(o_t=2) = 1.0$ $b_2(o_t=1) = 1.0$

There are 3 1st-order 2-state Markov Models:



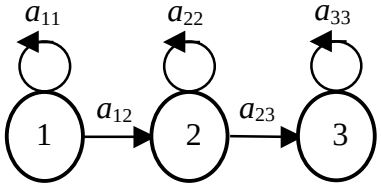
$b_1(o_t=2) = 1.0$ $b_2(o_t=6) = 1.0$

There is 1 1st-order 2-state Markov Model:



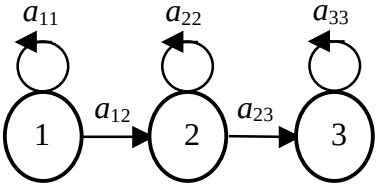
$b_1(o_t=2) = 1.0$ $b_2(o_t=14) = 1.0$

There are 3 1st-order 3-state Markov Models:



$b_1(o_t=2) = 1.0$ $b_2(o_t=6) = 1.0$ $b_3(o_t=14) = 1.0$

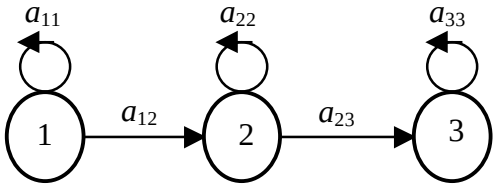
There is 1 1st-order 3-state Markov Model:



$b_1(o_t=2) = 1.0$ $b_2(o_t=12) = 1.0$ $b_3(o_t=6) = 1.0$

Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 3-state Hidden Markov Model.



The major observable symbols for each state:

$b_1(o_t=2) \leq 1.0$ $b_2(o_t=6) \leq 1.0$ $b_3(o_t=14) \leq 1.0$

Figure 55 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU15+17.

respectively.

Example 5:

In Figure 56 for "expression unit" AU17+23+24, there are thirteen 1st-order left-right Markov models with five major symbols having higher occupied probability (0: 0.111, 4: 0.191, 8: 0.155, 12: 0.133, and 14: 0.346) and with one minor symbol having lower occupied probability (1: 0.064).

Rule 2.1: Symbol 14 should be at the first state, and symbols 0 and 8 should be at the last state of the 1st-order left-right HMM.

Rule 2.2: Symbol 1 will be combined by its neighbor states since it has lower occupied probability.

Rule 2.4: Symbol 12 has very strong connective intensities between symbols 14 and 4. Symbol 4 has strong connective intensities between symbols 12 and 0.

Since symbols 0, 4, 8, 12, and 14 all have high occupied probability, symbol 14 should be at the first state, and symbol 12 will be at the second state of the 1st-order left-right HMM, which is right after symbol 14 but before symbol 4. Symbol 4 should domain one state which is between symbols 12 and 0. Symbols 0 and 8 should be at the last states.

Rule 2.5: The total state numbers of the HMM should be no more than the maximum state number of Markov models.

The final result is a 1st-order 4-state left-right HMM whose major probability of observable symbols for each state is symbol 14 for state 1, symbol 12 for state 2, symbol 4 for state 3, and symbol 0 and 8 for the last state.

Example 6:

In Figure 57 for "expression unit" AU6+12+25, there are eleven 1st-order left-right Markov models with three major symbols having higher occupied probability (3: 0.262, 7: 0.302, and 14: 0.274) and four minor symbols having lower occupied probability (6: 0.044, 9: 0.011, 11: 0.044, and 15: 0.063).

The overall probability for each observable symbol from all 13 different symbol sequences of the facial expression AU17+23+24: (O = {o_t=0, o_t=1, o_t=4, o_t=8, o_t=12, o_t=14})

Major Symbols:

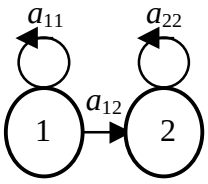
$P(o_t= 0|O) = 0.111$ $P(o_t= 4|O) = 0.191$
$P(o_t= 8|O) = 0.155$ $P(o_t= 12|O) = 0.131$

Minor Symbols:

$P(o_t= 1|O) = 0.064$

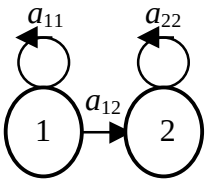
Step 1: The 1st-Order Markov Model

There is 1 1st-order 2-state Markov Model:



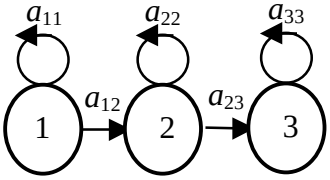
$b_1(o_t=14) = 1.0$ $b_2(o_t=12) = 1.0$

There is 1 1st-order 2-state Markov Model:



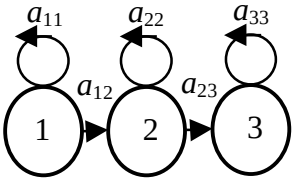
$b_1(o_t=14) = 1.0$ $b_2(o_t=1) = 1.0$

There are 3 1st-order 3-state Markov Models:



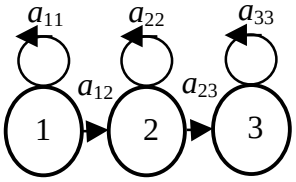
$b_1(o_t=14) = 1.0$ $b_2(o_t=12) = 1.0$ $b_3(o_t=4) = 1.0$

There are 3 1st-order 3-state Markov Models:



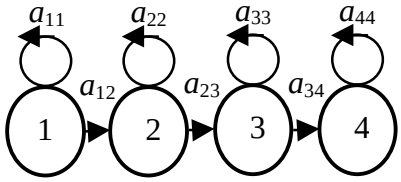
$b_1(o_t=14) = 1.0$ $b_3(o_t=8) = 1.0$
$b_2(o_t=12) = 1.0$

There is 1 1st-order 3-state Markov Model:



$b_1(o_t=14) = 1.0$ $b_3(o_t=8) = 1.0$
$b_2(o_t=1) = 1.0$

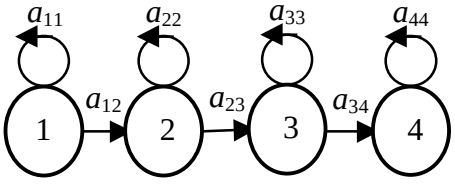
There are 4 1st-order 4-state Markov Models:



$b_1(o_t=14) = 1.0$ $b_3(o_t=4) = 1.0$
$b_2(o_t=12) = 1.0$ $b_4(o_t=0) = 1.0$

Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 4-state Hidden Markov Model.



The major observable symbols for each state:

$b_1(o_t=14) \leq 1.0$ $b_2(o_t=12) \leq 1.0$ $b_3(o_t=4) \leq 1.0$ $\{b_4(o_t=0) + b_4(o_t=8)\} \leq 1.0$

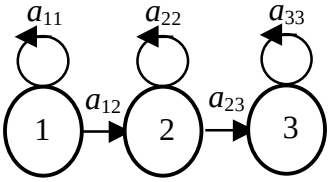
Figure 56 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU17+23+24.

The overall probability for each observable symbol from all 11 different symbol sequences of the facial expression AU6+12+25: (O = {o_t=3, o_t=6, o_t=7, o_t=9, o_t=11, o_t=14, o_t=15}) Major Symbols: Minor Symbols:

$P(o_t= 3|O) = 0.262$ $P(o_t= 6|O) = 0.044$ $P(o_t= 9|O) = 0.011$
$P(o_t= 7|O) = 0.302$ $P(o_t= 11|O) = 0.044$ $P(o_t= 15|O) = 0.063$
$P(o_t= 14|O) = 0.274$

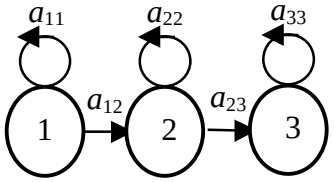
Step 1: The 1st-Order Markov Model

There is 1 1st-order 3-state Markov Model:



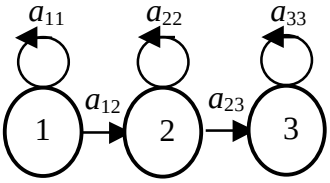
$b_1(o_t=14) = 1.0$ $b_2(o_t=11) = 1.0$ $b_3(o_t=15) = 1.0$

There are 3 1st-order 3-state Markov Models:



$b_1(o_t=14) = 1.0$ $b_2(o_t=11) = 1.0$ $b_3(o_t=3) = 1.0$

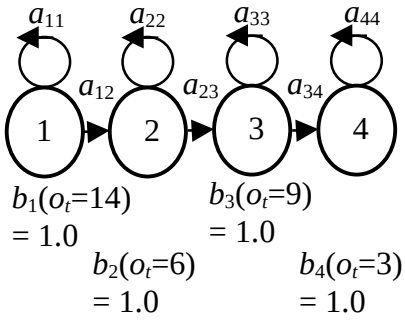
There is 1 1st-order 3-state Markov Model:



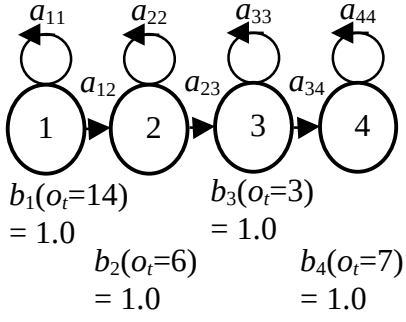
$b_1(o_t=14) = 1.0$ $b_2(o_t=6) = 1.0$ $b_3(o_t=3) = 1.0$

Figure 57 (Continued)

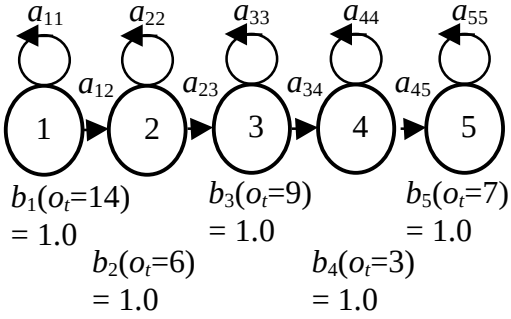
There is 1 1st-order 4-state Markov Model:



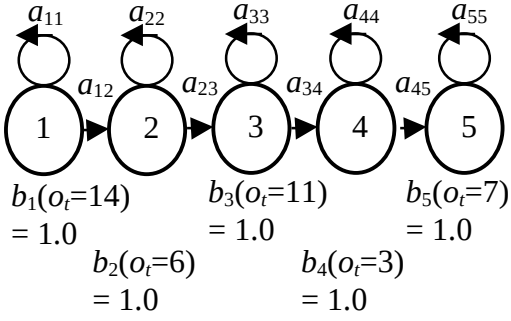
There is 1 1st-order 4-state Markov Model:



There are 2 1st-order 5-state Markov Models:

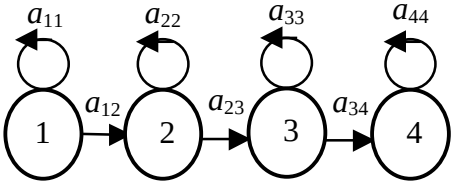


There is 1 1st-order 5-state Markov Model:



Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 4-state Hidden Markov Model.



The major observable symbols for each state:

$b_1(o_t=14) \le 1.0$

$\{b_2(o_t=6) + b_2(o_t=9) + b_2(o_t=11)\} \le 1.0$

$b_3(o_t=3) \le 1.0$

$b_4(o_t=7) \le 1.0$

Figure 57 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU6+12+25.

Rule 2.1: Symbol 14 should be at the first state, and symbols 7 and 15 should be at the last state of the 1st-order left-right HMM.

Rule 2.2: Symbol 15 (lower occupied probability) will be combined by symbol 7 (higher occupied probability) at the last state of the HMM.

Rule 2.3 & 2.4: For three symbols having lower probability (6, 9, and 11), symbol 9 is always after symbol 6. Symbol 6 is always at the second state after symbol 14 at the first state. Symbol 11 is either following symbol 6 or at the second state after symbol 14. Symbol 3 has no connective intensity with symbol 14 taken the first state. So all three symbols (6, 9, and 11) can combine together and take the second state of the HMM. We can then find that symbol 3 has strong connective intensities between the second state (taken by symbols 6, 9, and 11) and the last state (taken by symbols 7 and 15).

A 1st-order 4-state left-right HMM whose major probability of observable symbols for state 1 is 14, for state 2 is 6, 9, and 11, for state 3 is 3, and for state 4 is 7 will be employed.

Example 7:

In Figure 58 for "expression unit" AU6+12+25, there are twelve 1st-order left-right Markov models with four major symbols having higher occupied probability (2: 0.305, 3: 0.109, 5: 0.287 and 11: 0.236) and with three minor symbols having lower occupied probability (1: 0.003, 9: 0.030, and 10: 0.030).

Rule 2.1: Symbol 2 should be at the first state, and symbols 3 and 11 should combine and be at the last state of the 1st-order left-right HMM.

Rule 2.3 & 2.4: Symbols 9 and 10 are always at the second state which is after symbol 2 at the first state, so both symbols can combine together at the second state. Symbol 5 has very weak connective intensity with the first state, but it has very strong connective intensities between the second and the last state.

Rule 2.2: Symbol 1 can be combined by both symbols 9 and 10 at the second state and

The overall probability for each observable symbol from all 12 different symbol sequences of the facial expression AU6+12+25: (O = {o_t=1, o_t=2, o_t=3, o_t=5, o_t=9, o_t=10, o_t=11}) Major Symbols:

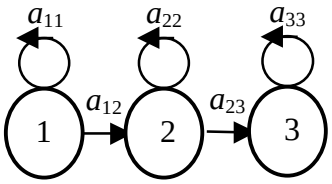
$P(o_t= 2|O) = 0.305$ $P(o_t= 3|O) = 0.109$
$P(o_t= 5|O) = 0.287$ $P(o_t= 11|O) = 0.236$

Minor Symbols:

$P(o_t= 1|O) = 0.003$
$P(o_t= 9|O) = 0.030$
$P(o_t= 10|O) = 0.030$

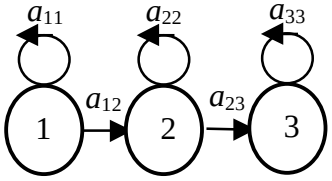
Step 1: The 1st-Order Markov Model

There are 2 1st-order 3-state Markov Models:



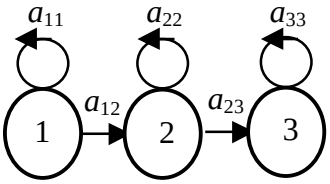
$b_1(o_t=2) = 1.0$ $b_2(o_t=5) = 1.0$ $b_3(o_t=11) = 1.0$

There are 3 1st-order 3-state Markov Models:



$b_1(o_t=2) = 1.0$ $b_2(o_t=10) = 1.0$ $b_3(o_t=5) = 1.0$

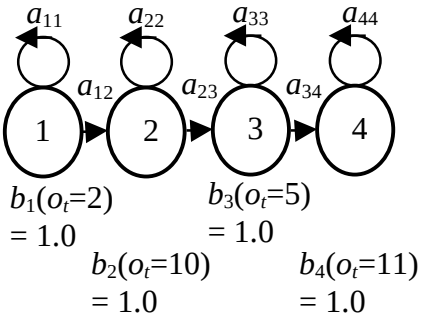
There are 2 1st-order 3-state Markov Models:



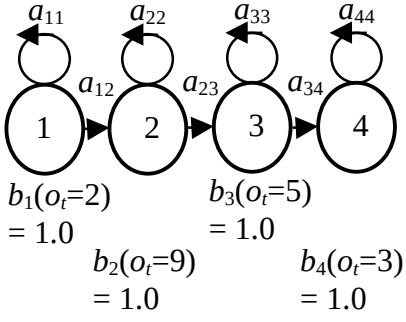
$b_1(o_t=2) = 1.0$ $b_2(o_t=9) = 1.0$ $b_3(o_t=5) = 1.0$

Figure 58 (Continued)

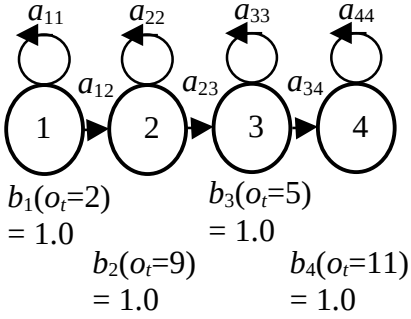
There is 1 1st-order 4-state Markov Model:



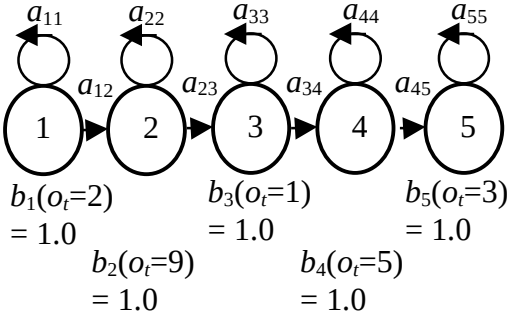
There is 1 1st-order 4-state Markov Model:



There are 2 1st-order 5-state Markov Models:

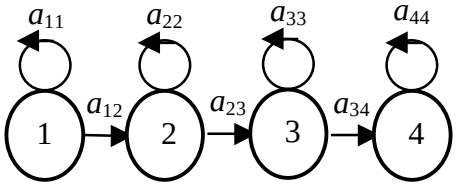


There is 1 1st-order 5-state Markov Model:



Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 4-state Hidden Markov Model.



The major observable symbols for each state:

$b_1(o_t=2) \leq 1.0$	$\{b_2(o_t=9) + b_2(o_t=10)\} \leq 1.0$	$b_3(o_t=5) \leq 1.0$	$\{b_4(o_t=3) + b_4(o_t=11)\} \leq 1.0$

Figure 58 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU6+12+25.

symbol 5 having high occupied probability.

We can use a 1st-order 4-state left-right HMM whose major probability of observable symbols is 2 for state 1, 9 and 10 for state 2, 5 for state 3, and 3 and 11 for state 4 to represent the performance of original all Markov models.

Example 8:

In Figure 59 for "expression unit" AU20+25, there are eighteen 1st-order left-right Markov models with three major symbols having higher occupied probability (2: 0.404, 10: 0.212, and 11: 0.128) and with seven minor symbols having lower occupied probability (3: 0.080, 5: 0.060, 6: 0.003, 7: 0.014, 12: 0.003, 13: 0.074, and 14: 0.021).

Rule 2.1 & 2.3: Symbol 2 should be at the first state. Symbols 5, 7, 13, and 14 will be combined by symbol 11 (higher occupied probability) and be at the last state of the 1st-order left-right HMM.

Rule 2.2 & 2.4: Since symbol 10 (high occupied probability) has strong connective intensity with the first state taken by symbol 2, and also has strong connective intensity with the following state, it is not likely to be at the last state of HMM.

Rule 2.2: The state having symbols 3, 6 or 12 will be combined by its neighbor states with higher occupied probability of symbols.

Rule 2.4: Symbols 5, 7, 11, 13, and 14 at the last state will have strong connective intensity with symbol 10 at the second state of HMM.

So a 1st-order 3-state left-right HMM whose major probability of observable symbols is 2 for state 1, 10 for state 2, and 5, 7, 11, 13 and 14 for state 3 will be employed.

The overall probability for each observable symbol from all 18 different symbol sequences of the facial expression AU20+25: (O = {o_t=2, o_t=3, o_t=5, o_t=6, o_t=7, o_t=10, o_t=11, o_t=12, o_t=13, o_t=14})

Major Symbols:

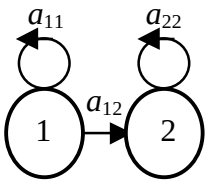
$P(o_t= 2|O) = 0.404$
$P(o_t= 10|O) = 0.212$
$P(o_t= 11|O) = 0.128$

Minor Symbols:

$P(o_t= 3|O) = 0.080$ $P(o_t= 5|O) = 0.060$
$P(o_t= 6|O) = 0.003$ $P(o_t= 7|O) = 0.014$
$P(o_t= 12|O) = 0.003$ $P(o_t= 13|O) = 0.074$
$P(o_t= 14|O) = 0.021$

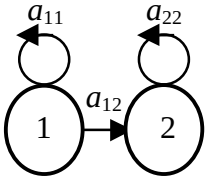
Step 1: The 1st-Order Markov Model

There is 1 1st-order 3-state Markov Model:



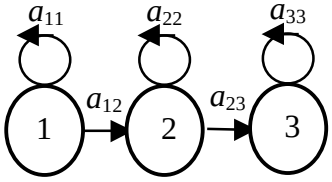
$b_1(o_t=2) = 1.0$ $b_2(o_t=5) = 1.0$

There are 4 1st-order 3-state Markov Models:



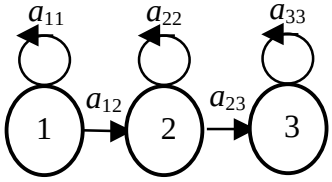
$b_1(o_t=2) = 1.0$ $b_2(o_t=10) = 1.0$

There is 1 1st-order 3-state Markov Model:



$b_1(o_t=2) = 1.0$ $b_2(o_t=12) = 1.0$ $b_3(o_t=5) = 1.0$

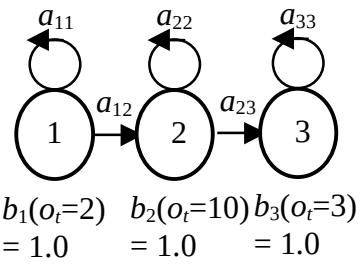
There are 3 1st-order 3-state Markov Models:



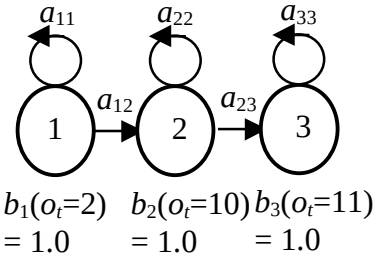
$b_1(o_t=2) = 1.0$ $b_2(o_t=10) = 1.0$ $b_3(o_t=13) = 1.0$

Figure 59 (Continued)

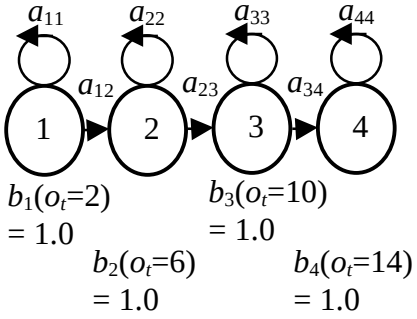
There are 3 1st-order 3-state Markov Models:



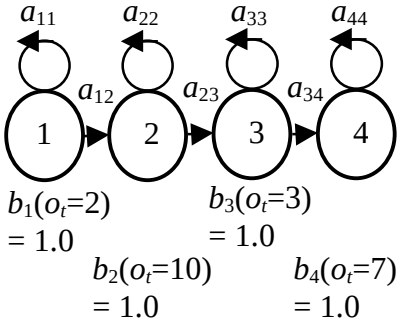
There are 4 1st-order 3-state Markov Models:



There is 1 1st-order 4-state Markov Model:



There is 1 1st-order 4-state Markov Model:



Step 2: The 1st-Order Hidden Markov Model -

All 1st-order Markov Models at Step 1 can be represented by this 1st-order 3-state Hidden Markov Model.

The major observable symbols for each state:

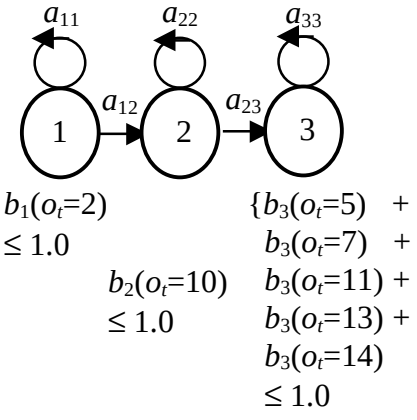


Figure 59 A 1st-order Hidden Markov Model can be used to represent the combination of all 1st-order Markov Models for facial expression AU20+25.

7.1.3 Step 3: The Multi-Order Hidden Markov Model

The property of an HMM is such that each state can be reached by any state based on the connections of state transitions. We can therefore combine different state numbers of 1st-order left-right HMMs into one multi-order multi-state left-right HMM, without loss of the individual performance.

Rule 3.1: The initial multi-order *N*-state left-right HMM is the same as the 1st-order *N*-state left-right HMM which has the maximum number state *N* of all 1st-order *n*-state left-right HMMs formed at Step 2.

$$N = \max\{n\} \tag{7-1}$$

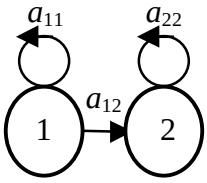
Rule 3.2: If there exist 1st-order *n*-state left-right HMMs at Step 2, then the states at the multi-order *N*-state left-right HMM should be connected between states *i* and *j* (*i.e.*, there exists state-transition probability a_{ij}) as follow:

$$a_{ij} \geq 0 \quad \text{if} \quad j = i + (N - n + 1) \qquad \text{where} \quad 1 \leq i, j \leq N \tag{7-2}$$

Example:

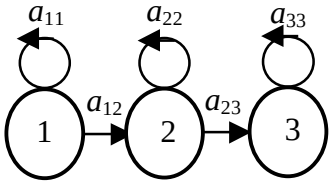
According to the above example analysis at Step 2, we use 1st-order 2-state (n = 2), 3-state (n = 3), and 4-state (n = 4) left-right HMMs to model various "expression units" of individual AUs or AU combinations for lower facial expressions. Using the above rules, we can use a 3rd-order 4-state (N = max $\{n\}$ = 4) left-right HMM to include all possibilities of state-transition probability (a_{ij}) and observable symbol probability ($b_j(o_t)$) among all 1st-order left-right HMMs at Step 2 (Figure 60).

1st-order *n*-state Hidden Markov Model (*n* = 2, 3, 4):



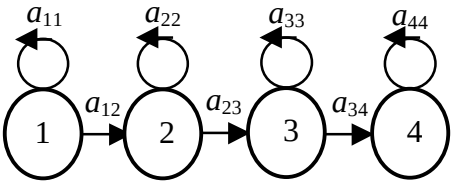
Number of states:

n = 2



Number of states:

n = 3



Number of states:

n = 4

Rule 3.1: The number of states *N* for the multi-order *N*-state Hidden Markov Model -

N = max {*n*} = max {2,3,4}= 4

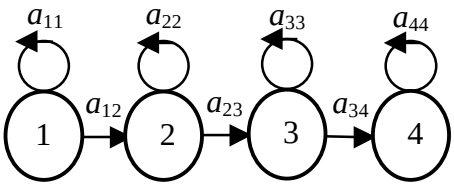
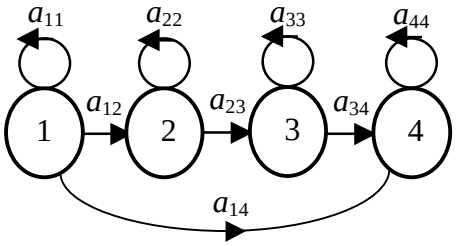


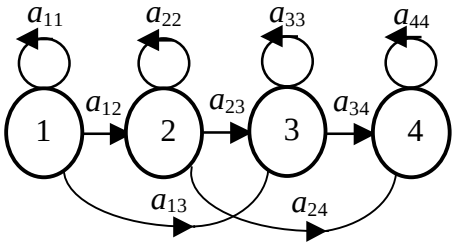
Figure 60 (Continued)

Rule 3.2: The combination of the 1st-order 2-, 3- and 4-state Hidden Markov Models -

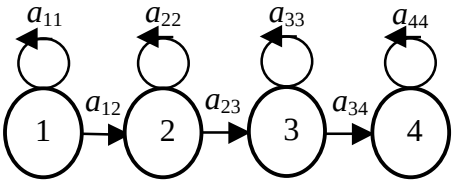
1. The 1st-order 2-, 3- and 4-state Hidden Markov Models can be represented by different multi-order 4-state Hidden Markov Models:



Compatible to the 1st-order 2-state Hidden Markov Model



Compatible to the 1st-order 3-state Hidden Markov Model



The 1st-order 4-state Hidden Markov Model

2. All above multi-order 4-state Hidden Markov Models can combine to be a 3rd-order 4-state Hidden Markov Model.

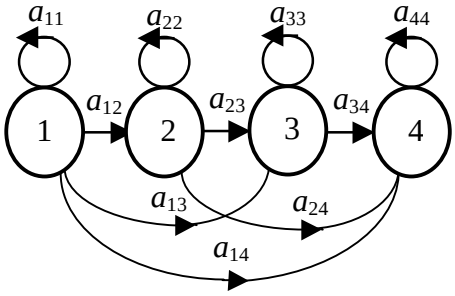


Figure 60 The 1st-order 2-, 3- and 4-state Hidden Markov Models can combine to be a 3rd-order 4-state Hidden Markov Model.

7.2 Physical Meaning of Hidden Markov Model Topology

One advantage of starting from the Markov model in determining the HMM topology is avoidance of the redundant states and connections (state transitions) at the HMM. The "redundant state" means multiple states connected in series which are represented by the same observable symbol having the major observable symbol probability $b_j(o_t)$ (such as $b_j(o_t) > 0.90$) at each state. In addition, this kind of redundant state is always an intermediate state and always occurs simultaneously with a very low self-transition probability a_{ii} (such as $a_{ii} < 0.01$). The "redundant state transition" means the state-transition probability is close or equal to zero. Both redundant situations will increase the elements of HMM parameters and increase the computation time. Because of this, HMM parameters may difficult to adapt in order to perform well the statistical behavior of the training data. Our method begins with assigning each individual symbol of each observable symbol sequence to one state of the 1st-order left-right Markov model. Then, based on the HMM properties, the state numbers of the 1st-order left-right Markov model are reduced by the combination of redundant states. State connections of the 1st-order left-right HMM are increased when required to form the optimal multi-order left-right HMM.

If the HMM has an insufficient number of states or state transitions, then the physical meaning corresponding to each state will be difficult to represent, and the temporal consideration for state transitions will be lost. Furthermore, the ability of the model to represent the training sequences will degrade, which will lower performance of the recognition process. Using our method, the state combinations still maintain the property of each major symbol (high occupied probability) dominating one state of the multi-order left-right HMM (but each state of this HMM can contain one or several major symbols). Further, the connections between states (*i.e.*, state transitions) at the multi-order left-right HMM contain all transition possibilities over all training symbol sequences. Therefore, our method can ensure that the multi-order left-right HMM has exactly as many states and state transitions as the training symbol sequences required to be modeled.

Table 8 Physical meaning of the Hidden Markov Model topology.

Extracted Methods	Input Vector Sequence for the HMM	Physical Meaning of each HMM State
Feature Point Tracking	Displacement Vector Sequence	The displacements of facial feature points
Dense Flow Tracking with Principal Component Analysis	Weight Vector Sequence	The displacements of the entire upper or lower facial region
High Gradient Component Analysis	Mean-Variance Vector Sequence	The distribution of furrow variation

According to our method for determining the HMM topology, each state at the multi-order left-right HMM consists of one or several major symbols having high occupied probability, and many other minor symbols having low occupied probability to represent a similarly-behaving portion of training symbol sequences. Using these symbols, it can obviously realize the physical meaning of each state corresponding to a portion of these training observable symbol sequences, such as those in Table 8.

The number of states for each HMM is proportional to some measure of variation in motion as well as variation (deviation) among the training motion data of different classes. In our study, the motion deviation among these three "expression units" of the upper facial expressions is smaller than that among these six "expression units" of the lower facial expressions. Based on this method for determination of HMM topology, a 2nd-order 3-state left-right HMM is created to model the three upper facial expressions, and a 3rd-order 4-state left-right HMM for the six lower facial expressions.

8.0 EXPERIMENTAL RESULTS

The computer-vision based facial expression recognition system described in the previous chapters has been trained and experimented with a large set of image sequences containing nine frequently occurred facial expressions of many subjects with various expression intensities. The experimental results are very exciting and have shown a great promise of our automatic recognition system.

8.1 Data Acquisition, Experimental Setup, and Digitizing

The database consists of 90 adult volunteers and 4 infants. The subjects included both male (35%) and female (65%). They ranged in both age (from 1 to 35 years of age) and ethnicity (81% Caucasian, 14% African-American, 4% Asian or Indian, and 1% Hispanic). The data acquisition was done in 8 sessions over a 2-month period [1]. More than 400 image sequences and 8000 images were made available to this research.

Adult subjects were seated 2 meters directly in front of a standard VHS video camera, with a video rate of 30 frames per second, which was manually adjusted to capture a full-face frontal view. None of the subjects wore eyeglasses. Some of subjects had hair covering their foreheads, and several subjects wore caps, or had makeup on their brows, eyelids or lips. Overhead fluorescent and incandescent lights as well as two halogen lights attached to portable umbrellas were positioned to the front at 30 degrees left and right, and were adjusted to provide maximum illumination with a minimum of facial shadows (Figure 61). Although reflection and lighting may have varied across individuals because of different facial skin colors and different times, constant illumination was used for each subject. These constraints - constant illumination using

[1] This was done by Miss Adena J. Zlochower and Dr. Jeffrey F. Cohn, Department of Psychology, University of Pittsburgh.

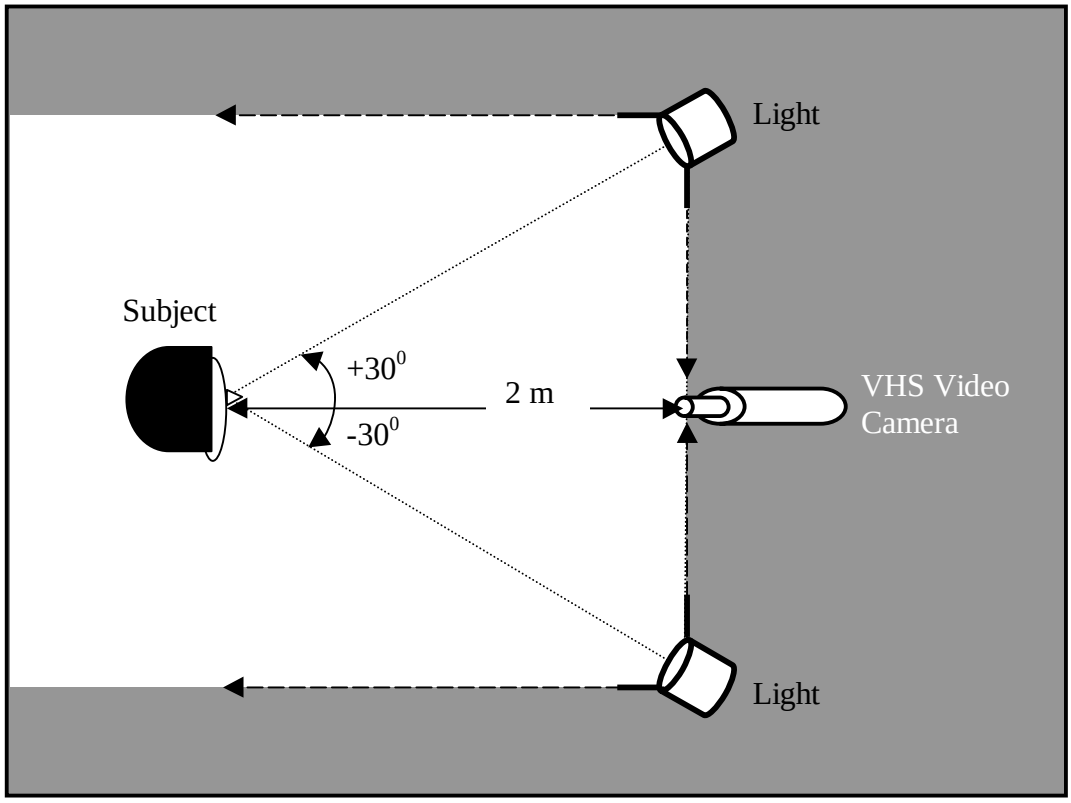


Figure 61 Experimental setup.

fixed light sources, and no eyeglasses - were imposed to minimize optical flow degradation.

None of the subjects were previously trained in displaying specific facial expressions. Prior to video recording, subjects practiced the expressions with FACS experts. During recording, subjects were free to look at the experts and copy their expressions. Subjects were asked to perform six basis expressions (joy, fear, anger, disgust, sadness, and surprise), and a series of "expression units" corresponding to individual AUs (*e.g.*, AU12) and AU combinations (*e.g.*, AU12+25) (see Table 3 for a complete list, and Figure 4 for "expression units"). Each expression was repeated 3 times, and the best expression was chosen. Each posed expression began from neutral, reached peak, and ended at neutral

expressions again. There is at least a half second duration (15 frames) of neutral expression between posed expressions. Even though these untraining subjects have seen the expressions demonstrated by experts, subjects still showed a range of posing ability. Not all of the expressions conformed to the "expression units," such as the combination of "expression units": AU1+2+4, AU12+20+25 and AU12+15. The spontaneous expressions showed more variability. In addition, facial expressions (non-rigid motion) with some out-of-plane head motion (rigid motion) such as yawing or pitch less than $\pm 10^0$ occurred concurrently, even though all subjects were viewed frontally.

Each frame of video sequence was automatically digitized into 490 x 640-pixel image on a Sun Sparc 20 workstation using the K2T digitizer. For feature point tracking and high gradient component analysis, the size of each frame was kept the same as the original 490 x 640-pixel image. To save the computing time when using dense flow tracking, the image size of each frame was automatically cropped to 417 x 385 pixels, which exactly covered the entire face and cut out the unnecessary background.

8.2 Segmentation and Coding by Human Observers (Ground Truth)

Before digitizing, the image sequences were segmented and coded by two certified FACS coders. Training a FACS coder is time consuming and takes approximately 100 hours to achieve acceptable levels of reliability, and coding criteria are subject to drift over the course of prolonged studies. It can take up to 10 hours of coding time per minute (30 frames/second) of taped facial behavior depending on the comprehensiveness of the system and the density of behavior changes.

Certified FACS coders segmented video tape from the beginning of the beginning duration, to the apex duration, and finally to the end of the ending duration to capture an expression sequence [(103)]. For each expression sequence, the beginning duration is defined as the last 2 ~ 4 frames of the neutral expression, which are prior to the facial movement, to the beginning of the apex duration. The ending duration is defined as from the end of

the apex duration to the first 2 ~ 4 frames of the neutral expression, which are after facial movement. The apex duration is defined as the maximum movement of facial motion, which is between the end of the beginning duration and the beginning of the ending duration. According to our experiments, generally there are at least 3 frame without any obvious movement at the apex duration (Figure 62). The velocity of facial motion for the beginning duration or the ending duration increases then decreases, like the oscillation of a spring between compression and release (Figure 62). Different expressions (from the beginning of the beginning duration to the end of the ending duration) correspond to different durations from 1/2 second (15 frames) to 3 1/3 seconds (100 frames).

All expression sequences were coded by two FACS experts at different dates. The overall agreement between the two FACS experts is 97%. Disagreement occurred due to fatigue during observation, which produced misclassification of subtly asymmetric expressions, eye blinking, or out-of-plane head motion. The agreement at eyebrow (upper face) expressions for AU1+4 was only 78%, because it is easy to confuse AU1+4 with either AU1 or AU4. The confusing between AU1+4 and AU1 occurs when very weak inner brows close together during inner brow raised, or the Ω shape of the furrow appears at the forehead during inner brow raised but without closing inner brows together (Figure 63.a). The confusion between AU1+4 and AU4 occurs when inner brows close together with eye blinking, head rotation pitching in the vertical direction, the Ω shape of furrow at the forehead, or asymmetric brow motion (Figure 63.b) In addition, the confusion among AU12+25, AU20+25 and AU12+20+25 also occurs sometimes, because these “expression units” have common muscle movement in the lip region (Figure 63.c). The final FACS AU coding for each expression sequence, which includes the subtle motion, instant motion (expression appears temporarily for only few frames but can not be seen at the peak expressions) and asymmetric motion, is set by the agreement of both FACS experts to be the ground true of our training and recognition processes.

In our experimental study, we used image sequences which start from the beginning of the beginning duration and end during the apex duration. These digitized image sequences

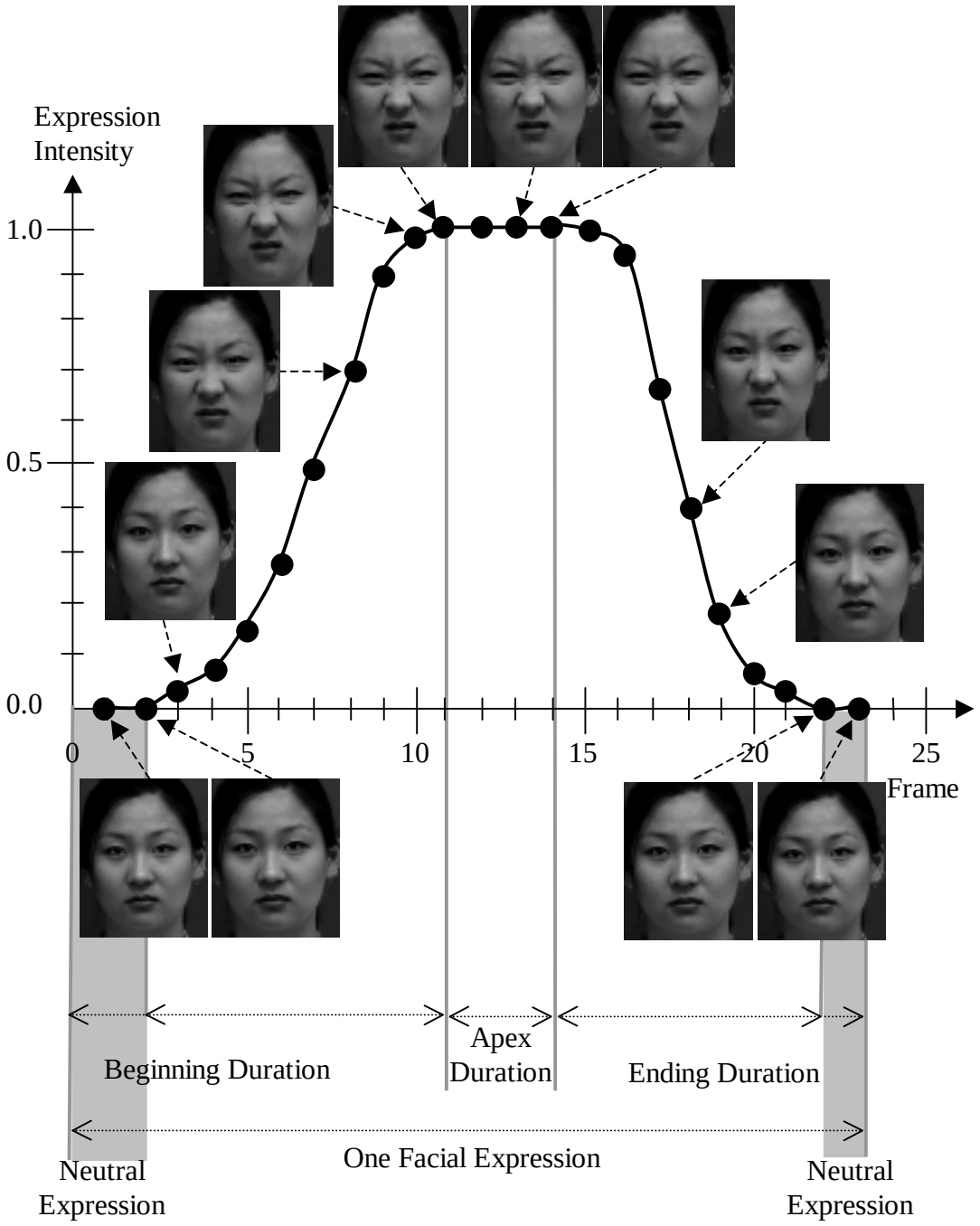
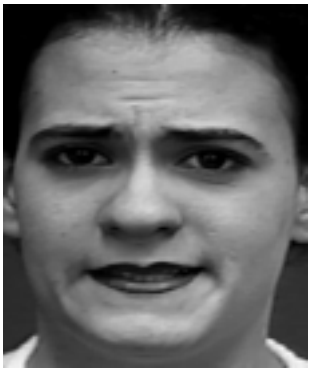


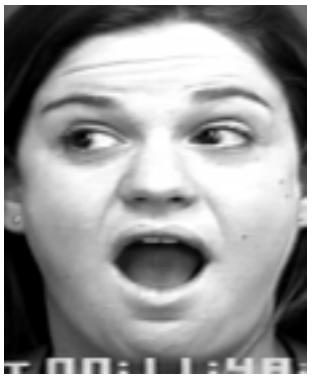
Figure 62 Each facial expression begins from the beginning duration, continues through the apex duration, and ends at the ending duration. In our current work, we segmented each facial expression to include only the beginning and apex durations.

Standard AU1+4



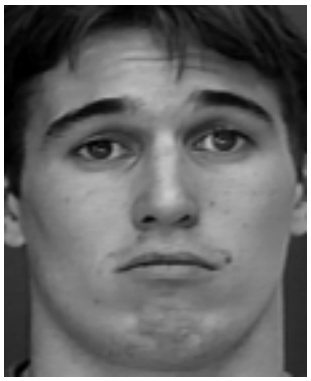
AU1+4+20+25 AU1+4+20+25

Medial portion of the eyebrows is raised (AU1) and pulled together (AU4).



AU1+15+17 AU1+2+12+27

Both left (AU1) and right images (AU1+2) have the same permanent Ω shapes of furrows at her forehead (usually in AU1+4).

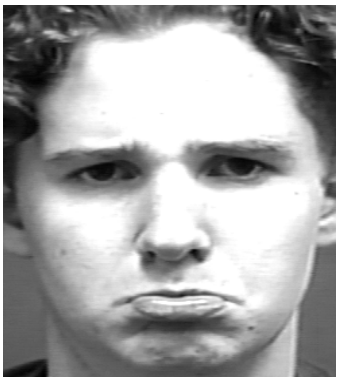


AU1+17 AU1+15+17

Ω shape of furrows (usually occurs in AU1+4) appears when the inner brows are raised (AU1).

Figure 63.a Standard AU1+4 expressions and manual misclassification of three AU1 expressions and one AU1+2 expression to AU1+4 expressions.

(1) AU4+15+17



(2) AU4+15+17

(3) AU4+20+25

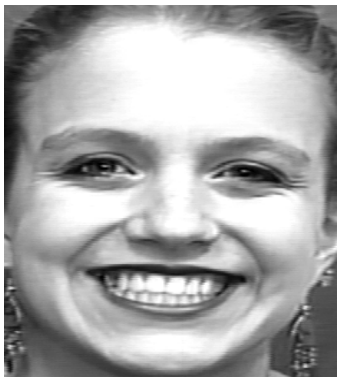
Figure 63.b Manual misclassification of three AU4 expressions to AU1+4 expressions. These mistakes are because of (1) Ω shape of furrows at the forehead, (2) confusing expression, and (3) asymmetric brow motion. The standard AU4 expression is shown in Figure 63.c (3).

(1) AU1+4+20+25

(2) AU1+4+12+20+25
Confusion between
AU4 and AU1+4 occurs

(3) AU4+12+20+25

(4) AU6+12+25



(5) AU6+12+25

Figure 63.c Confusions among AU12+25, AU20+25 (also in Figure 63.a and 63.b) and AU12+20+25.

are in arbitrary length varying from 9 to 47 frames. The average number of images per expression sequence is about 20.

8.3 Automatic Expression Recognition

Our goal is to discriminate subtle differences in facial expressions for the upper face region: AU4, AU1+4, and AU1+2, and for the lower face region: AU12, AU6+12+25, AU20+25, AU9+17, AU17+23+24, and AU15+17. The experimental image sequences were processed for extraction of expression information and coding. About one half of them were used in training, and the other half in testing. From these two sets, subsets were processed by three methods (facial feature point tracking, dense flow tracking with PCA, and high gradient component analysis). The extracted expression information is normalized using affine transformation, converted to displacement vector sequences, weight vector sequences, and mean-variance vector sequences, and then vector quantized into symbol sequences for use in training and recognition processes (Figure 64).

8.3.1 Training Process

In reality, the same facial expression may appear different among individuals because of different motion intensities. To design a robust recognition system, the training data were selected to cover all possible facial actions and expression intensities for each facial expression (Figure 65). Motions in upper facial expressions and in lower facial expressions were separately extracted. For upper facial expressions, the training data consist of 100 image sequences for high gradient component analysis, 60 image sequences for facial feature point tracking, and of which a subset of 45 sequences for dense flow tracking. For lower facial expressions, the training data consist of 120, 120, and 60 image sequences, respectively (Table 9). We used a smaller subset of data for dense flow tracking because of its requirement of excessive processing time.

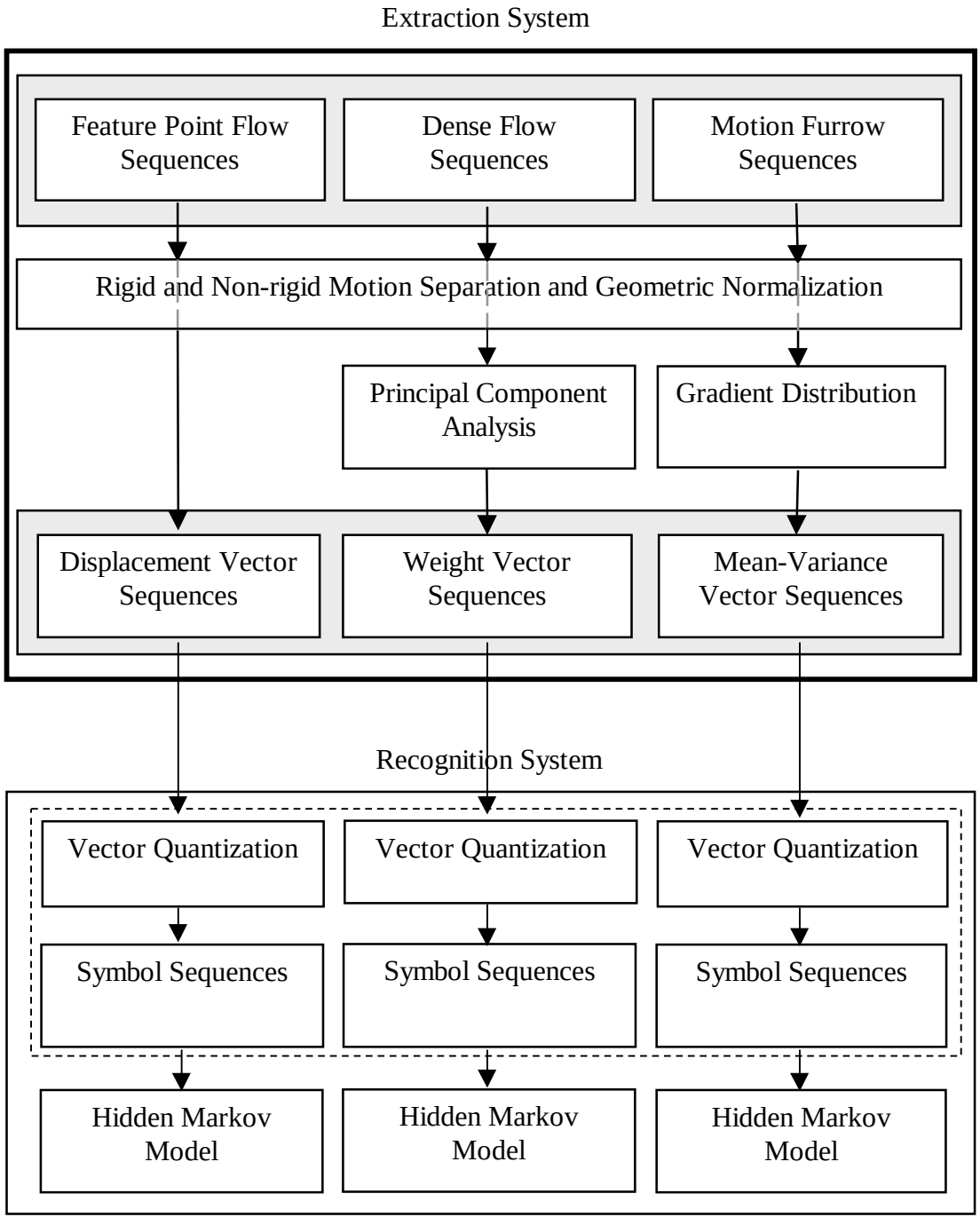


Figure 64 Three sets of extracted information as inputs to the recognition system using Hidden Markov Models.

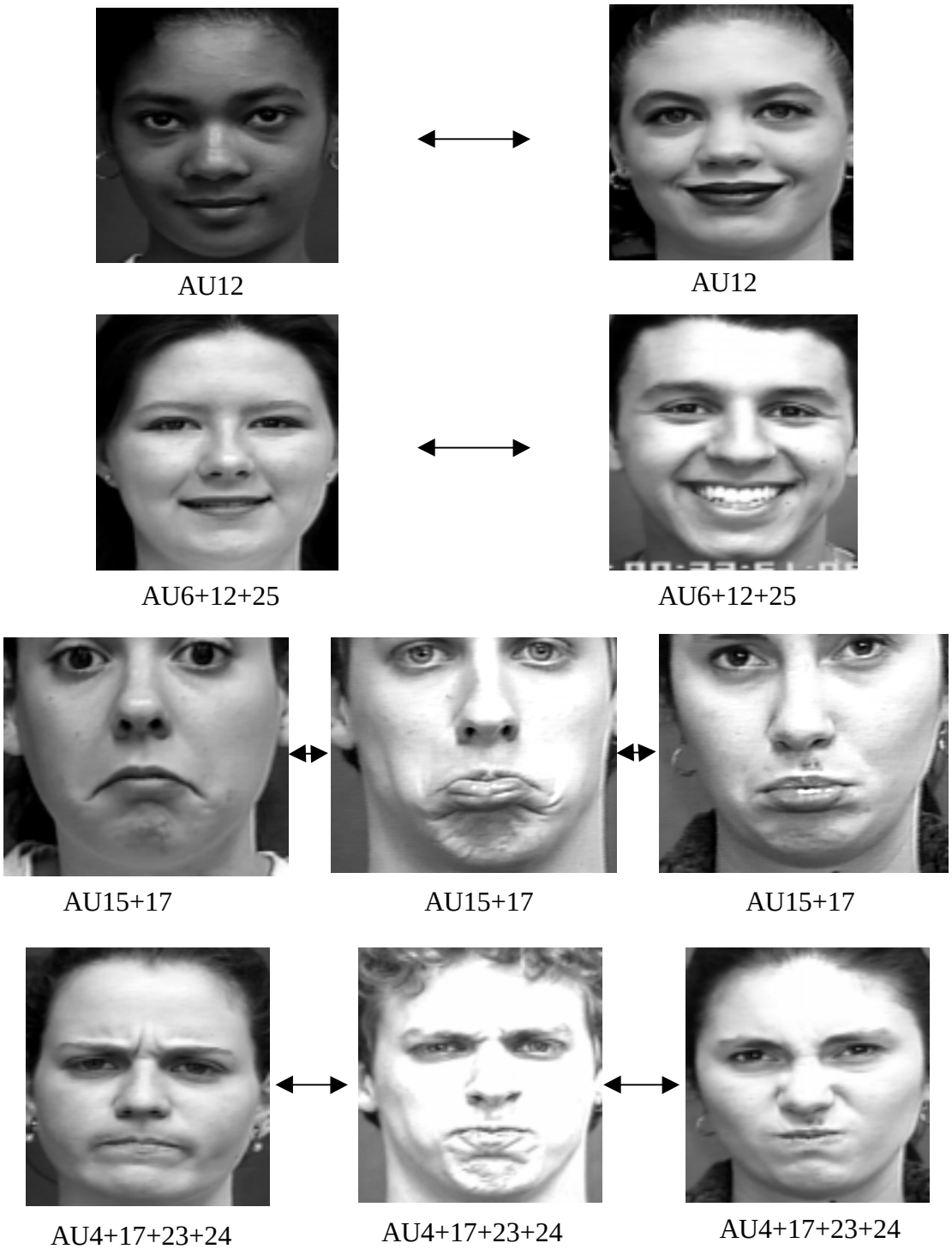
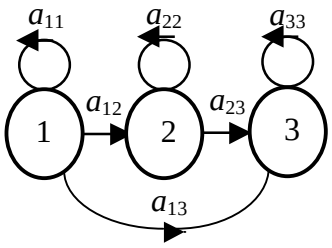


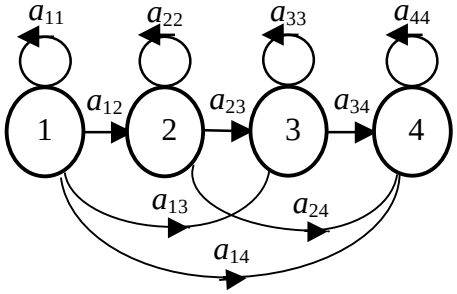
Figure 65 The images at the same row have the same facial expressions, but different facial actions or expression intensities.

Table 9 Different Hidden Markov Models for 3 upper facial expressions and 6 lower facial expressions.

Methods for the Extraction of Expression Information	Feature Point Tracking	Dense Flow Tracking with Principal Component Analysis	High Gradient Component Analysis in the Spatio-Temporal Domain
Codebook Size (*M*) for the Upper Facial Expressions	*M* = 16 (60 training symbol sequences)	*M* = 16 (45 training symbol sequences)	*M* = 32 (100 training symbol sequences)
Hidden Markov Model (HMM)	2nd-order 3-state left-right HMM	2nd-order 3-state left-right HMM	2nd-order 3-state left-right HMM
Codebook Size (*M*) for the Lower Facial Expressions	*M* = 32 (120 training symbol sequences)	*M* = 16 (60 training symbol sequences)	*M* = 32 (120 training symbol sequences)
Hidden Markov Model (HMM)	3rd-order 4-state left-right HMM	3rd-order 4-state left-right HMM	3rd-order 4-state left-right HMM



The 2nd-order 3-state left-right Hidden Markov Model for AU4, AU1+4, and AU1+2.



The 3rd-order 4-state left-right Hidden Markov Model for AU12, AU6+12+25, AU20+25, AU9+17, AU17+23+24, and AU15+17.

Before using HMMs for training or recognition process, any motion vector sequence is preprocessed by vector quantization to an observable symbol sequence O. The codebooks are created based on their corresponding training data. The codebook size M, which is power of 2, is chosen to be less than or equal to the total number of training frames divided by 50. So, the codebook size for the training data having 45 and 60 image sequences (around 45 x 20 = 900 and 60 x 20 = 1200 frames) is $M = 2^4$ ($2^4 \leq 900/50 = 18$, and $1200/50 = 24 < 2^5$) (Table 9). The codebook size for the training data having 100 and 120 image sequences (around 100 x 20 = 2000, and 120 x 20 = 2400 frames) is $M = 2^5$ ($2^5 \leq 2000/50 = 40$, and $2400/50 = 48 < 2^6$) (Table 9). The 12- and 20-dimensional displacement vectors from feature point tracking (for upper and lower facial expressions, respectively) (Figure 19), the 20- and 30-dimensional weight vectors from the dense flow tracking with principal component analysis (PCA) (Figures 29 and 30), and the 32- and 32-dimensional mean-variance vectors from the high gradient component analysis (Figure 41) are each vector quantized to one codeword (or observable symbol) o_t, $0 \leq o_t \leq M$-1, according to its respective codebook, where the subscript t denotes the frame t.

Based on these training symbol sequences, we determine the HMM topology using the method that we have developed. Thus, a 2nd-order 3-state left-right HMM and a 3rd-order 4-state left-right HMM are used for modeling the three upper facial expressions and six lower facial expressions, respectively (Table 9).

There are three sets of information extracted from the upper and lower facial expressions data by three methods. For each set of data, two sets of HMM parameters $\lambda = (\pi,A,B)$ are trained. Parameter sets λ_{AU4}, λ_{AU1+4}, and λ_{AU1+2} characterize the most likely occurrences of the three upper facial "expression units" (individual AUs or AU combinations), and sets λ_{AU12}, $\lambda_{AU6+12+25}$, $\lambda_{AU20+25}$, λ_{AU9+17}, $\lambda_{AU17+23+24}$, and $\lambda_{AU15+17}$ characterize the six lower facial "expression units." These trained HMM parameter sets serve to evaluate any observable symbol sequence of facial expression to give the most likely classification (Figure 66).

To initialize the training of each model parameter set, the initial state probability at the

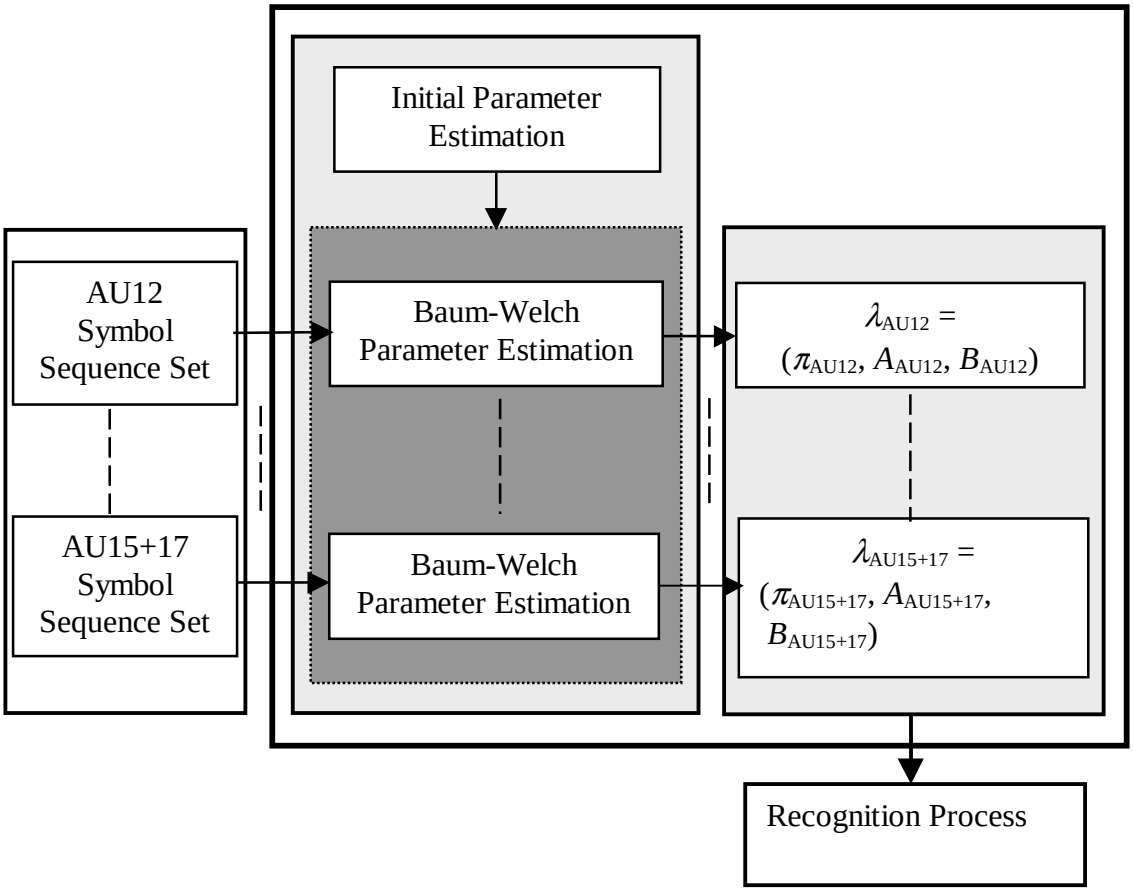


Figure 66 The training process for the Hidden Markov Model (an example for the lower facial expressions: AU12, AU6+12+25, AU20+25, AU9+17, AU17+23+24 and AU15+17).

first state π_1 is set to 1, and the rest states are set to 0. Each element of the state-transition probability matrix A_{NxN} and the output observable symbol probability matrix B_{MxN} is initialized to a very small value (say, 10^{-6}) of a uniformly distributed random variable. The Baum-Welch method is then applied to estimate the parameters $\lambda = (\pi, A, B)$ in iterations based on the Forward procedure (variable α) and the Backward procedure (variable β). After each iteration, the estimated probability for each element of these parameters is smoothed by setting a numeric floor 0.0001 to avoid zeroing the parameter element and producing an unreliable result. They are then renormalized to meet the

Table 10 The trained parameter set $\lambda = (\pi, A, B)$ of the 3rd-order 4-state Hidden Markov Model, whose topology is determined in Figures 58 and 60, for the lower facial expression AU6+12+25 using dense flow tracking method (codebook size M=16).

π				
	State 1	State 2	State 3	State 4
	1.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000

A				
State	State 1	State 2	State 3	State 4
1	0.852713734696394	0.147286265303606	0.000000000000000	0.000000000000000
2	0.000000000000000	0.530337222246666	0.106073169816378	0.363589607936957
3	0.000000000000000	0.000000000000000	0.861441737692501	0.138558262307499
4	0.000000000000000	0.000000000000000	0.000000000000000	1.000000000000000

B				
Symbol	State 1	State 2	State 3	State 4
0	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
1	0.000000000000000	0.000006813844404	0.028598228023638	0.000000000000000
2	0.999999999231619	0.133964815503377	0.000000000000000	0.000000000000000
3	0.000000000000000	0.000000000000000	0.113344687439651	0.534742185973886
4	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
5	0.000000000000000	0.017482535428337	0.858057084052733	0.000000182200935
6	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
7	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
8	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
9	0.000000000587439	0.424273591136245	0.000000000483978	0.000000000000000
10	0.000000000180942	0.424272244087637	0.000000000000000	0.000000000000000
11	0.000000000000000	0.000000000000000	0.000000000000000	0.465257631825179
12	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
13	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
14	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000
15	0.000000000000000	0.000000000000000	0.000000000000000	0.000000000000000

required statistical constraint, and go on for further iteration. The trained parameter values for $\lambda_{AU6+12+25}$, for example, are shown in Table 10.

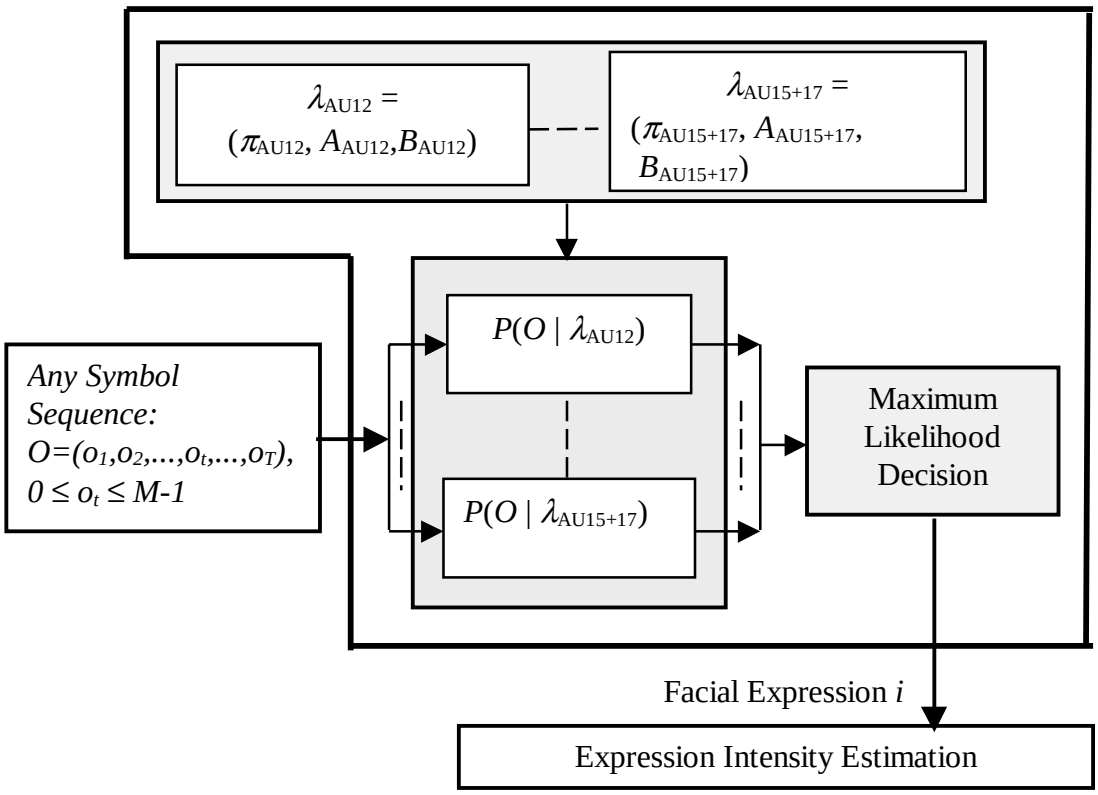


Figure 67 The recognition process for the Hidden Markov Model (an example for the lower facial expressions: AU12, AU6+12+25, AU20+25, AU9+17, AU17+23+24 and AU15+17).

8.3.2 Recognition Results

From the testing data, any observable symbol sequence O is evaluated or recognized by selecting the maximum output probability $P(O \mid \lambda_i)$ from the HMM parameter set λ_i, where λ_i is one of the HMM parameter set such as λ_{AU12}, $\lambda_{AU6+12+25}$, $\lambda_{AU20+25}$, λ_{AU9+17}, $\lambda_{AU17+23+24}$ and $\lambda_{AU15+17}$ for the lower facial expressions (Figure 67). If the output probability $P(O \mid \lambda_i)$ (usually it is close to 1) is greater than other output probability $P(O \mid \lambda_k)$, then the symbol sequence O is recognized as the facial expression represented by the model λ_i.

The recognition result using the HMM classifiers was evaluated by comparison with the coding of human observers coding taken as the ground truth. Two FACS experts

agreed on 97% of the collected facial expressions. The agreement for the AU1+4 was only 78%, since AU1+4 was easily confused with AU1 or AU4. The other point of disagreement was AU12+25, AU20+25 and AU12+20+25.

Because the computation time of different extraction methods is very different (the dense flow tracking is very time consuming when compared with the other two methods), and these methods were developed at different time (about 6 months apart), the number of image sequences used in experiments are not the same as indicated in Table 11. Those used in the dense flow study is a subset of the image sequences used in the feature point tracking study, which is a subset of those used in the high gradient components study. This situation is true for both training and testing. The test results of each study are given in Table 12, 13 and 14, respectively. The average recognition rate of the three upper facial expressions is 85% by feature point tracking, 92% by dense flow tracking with PCA, and 85% by high gradient component analysis. These results are based on 60, 45, and 100 training image sequences and 75, 60, and 160 testing image sequences. The average recognition rate of the six lower face expressions is 88% by feature point tracking, 92% by dense flow tracking with PCA, and 81% by high gradient component detection, based on 120, 60, and 120 training image sequences and 150, 90, and 150 testing image sequences (Table 11).

Comparing the recognition results of three different extraction methods, it is obvious that the dense flow tracking with PCA has the best performance for all facial expressions tested except AU17+23+24, for which the feature point tracking method is better, 92% against 87% (Table 12 and 13). The high gradient component analysis has the worst performance (Table 14). This is because dense flow tracking includes the entire motion information of a facial expression, such as allowing tracking a textureless region, which provides more complete information for the recognition process; but it requires substantially more computation time. It is subject to error due to occlusion (hair covering the forehead) or large discontinuities appearance of tongue or teeth when the mouth opens). The latter occurred in the case of 2-level dense flow estimation used for saving

Table 11 The number of the training and testing image sequences (the average number of frames per image sequence is 20) and their corresponding recognition rates.

	Training Image Sequences		
Three Methods	Feature Point Tracking	Dense Flow Tracking with PCA	High Gradient Component Analysis
No. of Sequences for the Upper Facial Expressions	60	45	100
No. of Sequences for the Lower Facial Expressions	120	60	120
	Testing Image Sequences		
Three Methods	Feature Point Tracking	Dense Flow Tracking with PCA	High Gradient Component Analysis
No. of Sequences for the Upper Facial Expressions	75	60	160
Recognition Results	85%	92%	85%
No. of Sequences for the Lower Facial Expressions	150	90	150
Recognition Results	88%	92%	81%

Table 12 Recognition results of the feature point tracking method. (The number given in each block is the number of testing image sequences.)

The average recognition rate for three upper facial expressions is 85% based on 75 testing image sequences.

HMM / Human	AU4	AU1+4	AU1+2	Recognition Rate
AU4	**22**	3	0	88%
AU1+4	4	**19**	2	76%
AU1+2	0	2	**23**	92%

The average recognition rate for six lower facial expressions is 88% based on 150 testing image sequences.

HMM / Human	AU12	AU6+12+25	AU20+25	AU9+17	AU17+23+24	AU15+17	Recognition Rate
AU12	**25**	0	0	0	0	0	100%
AU6+12+25	0	**21**	4	0	0	0	84%
AU20+25	0	5	**20**	0	0	0	80%
AU9+17	0	0	0	**22**	3	0	88%
AU17+23+24	0	0	0	0	**23**	2	92%
AU15+17	0	0	0	1	3	**21**	84%

Table 13 Recognition results of the dense flow tracking method. (The number given in each block is the number of testing image sequences.)

The average recognition rate for three upper facial expressions is 92% based on 60 testing image sequences.

HMM / Human	AU4	AU1+4	AU1+2	Recognition Rate
AU4	**21**	2	0	93%
AU1+4	2	**12**	1	80%
AU1+2	0	0	**22**	100%

The average recognition rate for six lower facial expressions is 92% based on 90 testing image sequences.

HMM / Human	AU12	AU6+12+25	AU20+25	AU9+17	AU17+23+24	AU15+17	Recognition Rate
AU12	**15**	0	0	0	0	0	100%
AU6+12+25	0	**13**	2	0	0	0	87%
AU20+25	0	2	**13**	0	0	0	87%
AU9+17	0	0	0	**15**	0	0	100%
AU17+23+24	0	0	0	0	**13**	2	87%
AU15+17	0	0	0	0	1	**14**	93%

Table 14 Recognition results of the motion furrow detection method. (The number given in each block is the number of testing image sequences.)

The average recognition rate for three upper facial expressions is 85% based on 160 testing image sequences.

HMM / Human	AU0	AU4	AU1+4	AU1+2	Recognition Rate
AU0	**26**	4	0	0	87%
AU4	5	**43**	2	0	86%
AU1+4	0	1	**24**	5	80%
AU1+2	0	0	7	**43**	86%

The average recognition rate for six lower facial expressions is 81% based on 150 testing image sequences.

HMM / Human	AU12 or AU6+12+25	AU9+17 or AU17+23+24	Recognition Rate
AU12 or AU6+12+25	**86**	14	86%
AU9+17 or AU17+23+24	12	**38**	76%

processing time.

High gradient component detection is sensitive to changes in transient facial features (*e.g.*, furrows), but is subject to error due to individual differences in subjects. Younger subjects, especially infants, show less furrowing than older ones, which reduces the information value of the high gradient components. Older subjects, in general, have permanent shapes of furrows on their faces. No matter how different their expressions or expression intensities are, the similar shape of furrows still can be seen such as in Figure 63.a (images at the second row). Occasionally, different FACS AUs may have the similar shape of furrows such as between AU12 and AU6+12+25, between AU6+12+25 and AU20+25, or between AU9+17 and AU17+23+24, since there are common facial muscle actions for both facial motions (Figure 68). Furthermore, the crow-feet wrinkles may or may not appear during facial expression AU6+12+25 (Figure 1). With these reasons, explaining the FACS AUs based only on the shape of furrows is not adequate. Furthermore, we used a constant threshold for motion line or edge detection. Since the gray values on each facial image are sensitive to facial motion and lighting, which depend on individual subjects, a dynamic thresholding would be needed.

The optical flow using the pyramid approach is a simple, fast, and accurate method of tracking facial feature points. It tracks large displacement well and is also sensitive to subtle feature motion.

In general, the pattern of errors in all three methods are similar, *i.e.*, errors were resulted from classifying an expression to an expression type which is most similar to the target (*e.g.*, AU4 was confused with AU1+4 but not AU1+2). It appears that the automatic feature point tracking method has given very good performance and its processing was very efficient. Potentially it can be developed into a real time recognition system. Summary of all three different extraction and recognition methods for facial expressions is in Table 15.

AU12

AU6+12+25

AU6+12+25

AU(1+4)+20+25

AU(4+)9+17

AU(4+)17+23+24

Figure 68 Different FACS AUs have the similar shape of furrows such as between AU12 and AU6+12+25, between AU6+12+25 and AU20+25, and between AU9+17 and AU17+23+24, since there are common facial muscle actions for both facial motions.

Table 15 Summary of three different extraction and recognition methods for facial expressions.

Extraction System			
Three Methods	**Feature Point Tracking (5-Level Pyramid)**	**Dense Flow Tracking with PCA (2-Level Pyramid)**	**High Gradient Component Analysis in the Spatio-Temporal Domain**
Computing Time	Fast (1%) (70 (13x13-pixel) windows: 20 seconds/frame) (SUN Sparc 5)	Very Slow (98%) (417 x 385 pixels: 20 minutes/frame) (SGI-Irix: 6 times faster than Sparc 5)	Fast (1%) (417 x 385 pixels: 5 second/frame) (SUN Sparc 5)
Hair at Forehead	Occlusion	Occlusion	No occlusion
Lighting	Sensitive	Sensitive	Sensitive
Subtle Motion (< 2 pixel)	Sensitive (Subpixel accuracy)	Insensitive	Sensitive
Large Motion (> 15 pixels)	100 pixels (Subpixel accuracy)	Missed tracking	Sensitive
Advantage	Simple and accurate May run in real time	Includes the entire face motion region	May run in real time
Disadvantage	Limited to pre-selected features	Time consuming	Detection error from individual differences in subjects (younger > older)
Recognition System (Recognition Rate)			
Different Inputs to HMMs	**Displacement Vector Sequence**	**Weight Vector Sequence**	**Mean-Variance Vector Sequence**
Upper Facial Expressions	85%	92%	85%
Lower Facial Expressions	88%	92%	81%

9.0 CONCLUSIONS

This research addressed the problem of automatic facial expression recognition based on FACS AUs. We have developed a computer vision system that automatically recognizes facial expressions with subtle differences and also estimates expression intensity. Three methods are used to extract facial motion information and estimate motion intensity: feature point tracking using the coarse-to-fine pyramid method, dense flow tracking together with the principal component analysis, and high gradient component analysis in the spatio-temporal domain, and is then used to discriminate subtle differences in facial expressions observed from image sequences of varying lengths. To determine the optimum HMM topology, a method has been developed and successfully applied to the facial expression recognition. Facial expressions of different types, intensities, and durations from a large number of untrained subjects have been tested. The results show that our system has high accuracy in facial expression recognition.

9.1 Contributions

The major contributions of this dissertation are summarized below.

This is the first automatic facial expression recognition system that has the capability of recognizing nine facial expressions based on FACS AUs: 3 subtly different expressions occurring in the upper face region, and 6 occurring in the lower face region. This system has successful trained with a significant number of image sequences containing various expressions from different individuals. It has been tested yielding good recognition performance.

Three methods have been developed for automatic extraction of information in facial expressions from an image sequence. The first method is facial feature point tracking by

applying the coarse-to-fine pyramid approach to track facial feature motion starting from selected feature points pertinent to facial expressions. It has subpixel accuracy for both subtle feature motion and large facial motion. A facial expression is then represented by a displacement vector sequence formed by concatenation of displacement vectors of facial feature points. The process is automatic except the interactive initialization of the feature points selection. It may run in real time. The second method is the wavelet-based multi-resolution level dense flow tracking over an entire region. The principal component analysis is then applied to the computed motion field to compress the information in terms of weighted eigenflows. A facial expression is then abstracted as an appropriate weight vector sequence. The third method is the high gradient component analysis in the spatio-temporal domain to automatically extract motion lines and edges corresponding to furrows appearing in face regions. This information is abstracted as a vector sequence where each vector consists of means and variances of high gradient components in 16 blocks for each frame region.

It is our pioneering effort to apply HMM for automatic recognition of facial expressions in image sequences of arbitrary lengths. For each type of the facial motion extraction and with the encoded symbol sequences obtained therefrom as inputs, two HMMs have been constructed: one models 3 facial expressions in the upper face region, and the other models 6 facial expressions in the lower face region. These models are of low order with simple feedforward connections and are used in the maximum likelihood decision. They have been successfully trained and tested yielding high recognition rate.

A method has been developed to determinate an “optimal” HMM topology for modeling facial expressions, “optimal” in the sense of plausible fewest elements of model parameters. Each state of the HMM corresponds to one level of motion of facial expressions. The number of states required for each HMM is proportional to some measure of variation in motion among the training data of different classes. Applying these method gives a 2nd-order 3-state left-right HMM for modeling the upper facial

expressions and a 3rd-order 4-state left-right HMM for modeling the lower facial expressions, respectively.

Differences in facial expression intensity can be used to discriminate between deliberate and spontaneous expressions, measure the degree of emotion, and analyze and synthesize facial expressions for teleconferencing or MPEG-4 applications. Expression intensity estimation may also provide an index for expression segmentation. We have given a method for estimating expression intensity based on each type of the extracted expression information and either a nonlinear mapping or a minimum distance criterion.

9.2 Suggestions for Future Work

Potential applications of our automatic facial expression recognition system include: assessment of nonverbal behavior in clinical and research settings such as psychological research of facial behavior coding, the communication between parents and preverbal infants, biomedical applications such as pre/post surgical path planning or clinical improvement prediction, law enforcement for lie detection, tiresome detecting, detection of tired drivers to help avoid car accidents, lip-reading to compliment speech recognition (audio-vision analysis), teleconferencing and MPEG-4 by analyzing and synthesizing facial signals, animation of the face, and the human-computer interface/interaction to make a computer able to "see."

Based on the work in this dissertation, the following problems are suggested for further research:

1. Recognition of more facial expressions by adding more "expression units" of individual AUs and AU combinations into the automatic recognition system.

2. Separation of non-rigid facial motion from rigid head motion when the latter involves a motion greater than $\pm 30^0$.

3. Detail comparison of three methods of extraction of facial expression information by using a larger but the same set of training and testing data.

4. Studies on Hidden Markov Model integration either from the view point of a multi-dimensional HMM with multiple inputs or from the view point of decision integration of multiple HMMs.

5. Investigations on computational issues with regard to (a) both 2-level and more than 2-level wavelet-based dense flow estimation, and (b) eigenflow computation.

6. Standardization of expression intensities.

7. Automatic segmentation of facial expression subsequences from a video sequence based on, for example, expression intensity estimation, for use in a real-time system.

8. Implementation of a real-time system. At the present time, the facial feature point tracking is the most likely method to approach running in real time; automatic segmentation of facial expression subsequences will be one of the key problems to be solved.

APPENDIX

APPENDIX

The Connected Component Labeling (CC labeling) Algorithm [43]:

I. The Top-Down Process:

Step 1: Three tables are used for the global minimum of each row in a binary image (1: foreground, and 0: background). All three tables are initialized at the beginning when processing each row.

a. Label table: This table records every label number which occurred in this row.

b. Equal table: This table has two columns indicating the different labels at the left (first) and right (second) columns belonging to the same classification. The label number at the first column is equal to or larger than that at the second column.

c. Link table: Each link table records the label numbers within the same classification, which can be linked together and arranged by increasing order.

Step 2: A 2 x 3 (row x column = r x c) CC labeling operator

(r-1,c-1)	(r-1,c)	(r-1,c+1)
(r,c-1)	(r,c)	

has been used for each row beginning from the left most and top most (0,0) position of the image. (r,c) is the center position of the operator at row 'r' and column 'c', and I(r,c) is its corresponding binary value. Label(r,c) means a label number is assigned to the pixel at position (r,c).

If I(r,c) > 0, then process as following; otherwise go to next pixel.

a. If I(r-1,c) > 0, then Label(r,c) = Label(r-1,c). Else

b. If I(r-1,c+1) > 0, then Label(r,c) = Label(r-1,c+1).

1. If I(r-1,c-1) > 0 and Label(r-1,c+1) ≠ Label(r-1,c-1), then record both label numbers to Equal table and put both number to the Link table. Else

2. If I(r,c-1) > 0 and Label(r-1,c+1) ≠ Label(r,c-1), then record both label numbers to Equal table and put both number to the Link table. Else

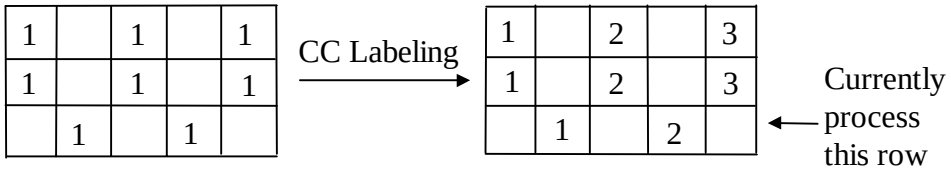
c. If I(r-1,c-1) > 0, then Label(r,c) = Label(r-1,c-1). Else

d. If I(r,c-1) > 0, then Label(r,c) = Label(r,c-1).

Step 3: Check each Link table. If nothing exists in this table, then no label number will be changed. Otherwise, use the label number in the current Link table to do the following process in the top-down order.

Scan the second column in the whole Equal table from top to bottom. If there is the same label number as the Link table, then check if the label number at the first column is larger than the label number at the current Link table. If it is, then the label number at the first column of Equal table is replaced by the label number at the current Link table.

Example:



After this step, these three tables will be like these examples:

Label Table

Index	Label
Label[0]	1
Label[1]	2
Label[2]	3

Equal Table

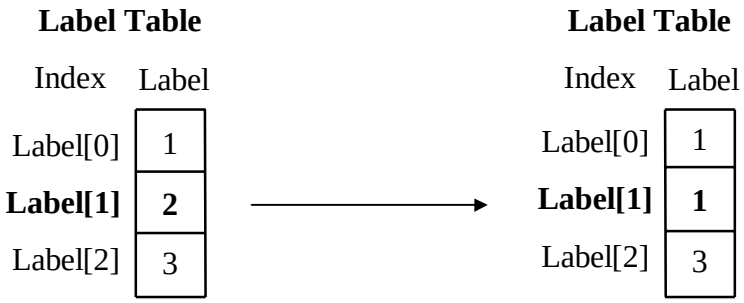
Index	Label 1st	Label 2nd
Equal[0]	2	1
Equal[1]	3	2

Link Table

Index	Link Label
Link[0]	1
Link[1]	2
Link[2]	3

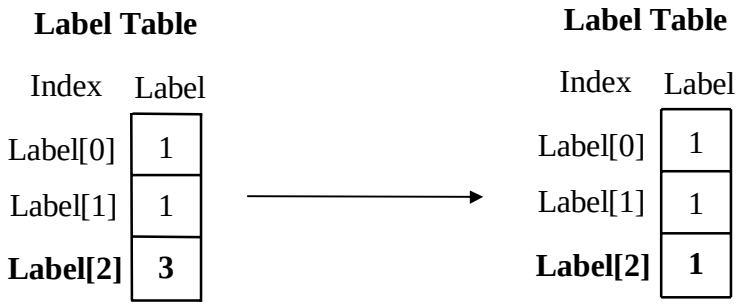
1.1 Pop out label number Link[0] = 1 from the Link table, and find out if label number '2' is equal to '1' from the Equal table: Equal[0](Label[1],Label[0]) where Label[1] = 2 and Label[0] =1.

1.2 Because Equal[0](Label[1],Label[0]), and (Label[1] = 2) > (Label[0] = 1) at Equal table, Label [1] = 1. That is, Label[1] = 2 = 1 = Label[0].



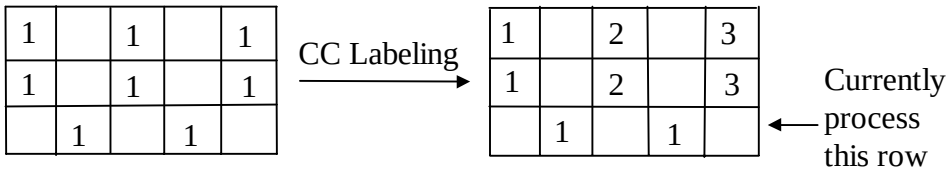
2.1 Pop out label number Link[2] = 3 from the Link table, and find out if label number '3' is equal to '2' from the Equal table: Equal[1](Label[2],Label[1]) where Label[2] = 3 and Label[1] = 2 = 1 = Label[0].

2.2 Because Equal[1](Label[2],Label[1]), and (Label[2] = 3) > (Label[1] = 2) at Equal table, Label [2] = 1. That is, Label[2] = 3 = Label[1] = 2 = Label[0] = 1.



3.1 Pop out label number Link[3] = 3 from the Link table, but there is no label number '3' at the second column of the Equal table: Then stop.

Step 4: The same row of the image is examined again (the second pass). If the label number is different from the Label table, then change it.



II. The Bottom-Up Process:

The process is similar to the Top-Down process by two passes for each row, beginning from the bottom to the top rows of the image, and left to right for each row. The 2 x 3 operator is:

(r,c-1)	(r,c)	
(r+1,c-1)	(r+1,c)	(r+1,c+1)

Since the Link table is already set, we only need the Label table and Equal table to set the same connected components to have the same label number.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Aizawa, K., Harashima, H., and Saito, T., "Model-Based Analysis Synthesis Image Coding System for a Person's Face," *Signal Processing: Image Communication 1*, pp. 139-152, 1989.

2. Anandan, P., "A Computational Framework and an Algorithm for the Measurement of Visual Motion," *International Journal of Computing Vision*, Vol. 2, pp. 283-310, 1989.

3. Barron. J.L., Fleet, D.J., and Beauchemin, S.S., "Systems and Experiment Performance of Optical Flow Techniques," *International Journal of Computer Vision*, Vol. 12, No. 1, pp. 43-77, 1994.

4. Bartlett, M.S., Viola, P.A., Sejnowski, T.J., Golomb, B.A., Larsen, J., Hager, J.C., and Ekman, P., "Classifying Facial Action," *Advances in Neural Information Processing Systems 8*, pp. 823-829, MIT Press, Cambridge, MA, 1996.

5. Bassili, J.N., "Emotion Recognition: The Role of Facial Movement and the Relative Importance of Upper and Lower Areas of the Face," *Journal of Personality and Social Psychology*, Vol. 37, pp. 2049-2059, 1979.

6. Baum, L.E., Petrie, T., Soules, G., and Weiss, N., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Mathematical Statistics*, Vol. 41, No. 1, pp. 164-171, 1970.

7. Belhumeur, P.N., Hespanha, J.P., and Kriegman, D.J., "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 711-720, 1997.

8. Bergen, J.R., Anandan, P., Hanna, K.J., and Hingorani, R., "Hierarchical Model-Based Motion Estimation," *Proc. of Second European Conference on Computer Vision*, pp. 237-252, Springer-Verlag, May 1992.

9. Beymer, D., "Face Recognition Under Varying Pose," *MIT Artificial Intelligence Laboratory*, A.I. Memo No. 1461, December 1993.

10. Beymer, D., "Vectorizing Face Images by Interleaving Shape and Texture Computations," *MIT Artificial Intelligence Laboratory*, A.I. Memo No. 1537, September 1995.

11. Black, M.J., and Anandan, P., “A Framework for the Robust Estimation of Optical Flow,” *In Proc. International Conference on Computer Vision*, pp. 231-236, Berlin, Germany, May 1993.

12. Black, M.J., and Yacoob, Y., "Tracking and Recognizing Facial Expressions in Image Sequences, Using Local Parameterized Models of Image Motion," *University of Maryland*, Technical Report CS-TR-3401, January 1995.

13. Black, M.J., Yacoob, Y., Jepson, A.D., and Fleet, D.J., "Learning Parameterized Models of Image Motion," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 561-567, Puerto Rico, June 1997.

14. Blake, A., Isard, M., and Reynard, D., “Learning to Track the Visual Motion of Contours,” *In J. Artifical Intelligence*, 1995.

15. Bregler, C., and Konig, Y., “Eigenlips for Robust Speech Recognition,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 669-672, Adelaide, 1994.

16. Bruce, V., *Recognizing Faces*, Lawrence Erlbaum Associates, London, 1988.

17. Brunelli, R., and Poggio, T., “Face Recognition: Features versus Templates,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 10, pp. 1042-1052, October 1993.

18. Cai, W., and Wang, J., “Adaptive Multiresolution Collocation Methods for Initial Boundary Value Problems of Nonlinear PDEs,” *Society for Industrial and Applied Mathematics*, Numer. Anal. , Vol. 33, No. 3, pp. 937-970, June 1996.

19. Canny, J., “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679-698, 1986.

20. Chellappa, R., Sirohey, S., Wilson, C.L., and Barnes, C.S., "Human and Machine Recognition of Faces: A Survey," *University of Maryland*, Technical Report CS-TR-3339, August 1994.

21. Choi, C.S., Harashima, H., and Takebe, T., "Analysis and Synthesis of Facial Expressions in Knowledge-Based Coding of Facial Image Sequences," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2737-2740, Toronto, Canada, May 1991.

22. Chui, C.K., *An Introduction to Wavelets*, Academic Press, 1992.

23. Cohn J.F., and Elmore, M., "Effect of Contingent Changes in Mothers' Affective Expression on the Organization of Behavior in 3-Month-Old Infants," *Infant Behavior and Development*, Vol. 11, pp. 493-505, 1988.

24. Cohn, J.F., Zlochower, A.J., Lien, J.J., Wu, Y.T., and Kanade, T., "Facial Expression Analysis: Preliminary Results of a New Image-Processing Based Method," *Proceedings of the 9th Conference of the International Society for Research on Emotions*, pp. 329-333, Toronto, Canada, August 1996.

25. Cohn, J.F., Zlochower, A.J., Lien, J.J., Wu, Y.T., and Kanade, T., "Facial Expression Can Be Measured by Image Processing of Video Sequences," *Biennial Meeting of the Society for Research in Child Development*, pp. 98, Washington, D.C., April 1997.

26. Cohn, J.F., Zlochower, A.J., Lien, J.J., Wu, Y.T., and Kanade, T., "Automated Face Coding: A Computer-Vision Based Method of Facial Expression Analysis," *7th European Conference on Facial Expression, Measurement, and Meaning*, Salzburg, Austria, July 1997.

27. Cohn, J.F., Zlochower, A.J., Lien, J.J., and Kanade, T., "Feature-Point Tracking by Optical Flow Discriminates Subtle Differences in Facial Expression," *Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 396-401, Nara, Japan, April 1998.

28. Cohn, J.F., Zlochower, A.J., Lien, J.J., and Kanade, T., "Automated Face Coding: A Computer-Vision Based Method of Facial Expression Analysis," *Journal of Psychophysiology*, In Press.

29. Cootes, T.F., Taylor, C.J., Cooper, D.H., and Graham, J., "Active Shape Models - Their Training and Application," *Computer Vision and Image Understanding*, Vol. 61, No. 1, pp. 38-59, January 1995.

30. Cottrell, G.W., and Metcalfe, J., "EMPATH: Face, Gender and Emotion Recognition Using Holons," *Advances in Neural Information Processing Systems 3*, pp. 564-571, San Mateo, CA, 1991.

31. Craw, I., and Cameron, P., "Face Recognition by Computer," *Proceedings of the British Machine Vision Conference*, pp. 487-507, 1992.

32. Darwin, C., *The Expression of Emotions in Man and Animals*, John Murray, 1872, reprinted by University of Chicago press, 1965.

33. Ding, J., Shimamura, M., Kobayashi, H., and Nakamura, T., "Neural Network Structures for Expression Recognition," *Proceedings of International Joint Conference on Neural Network*, pp. 1420-1423, 1993.

34. Ekman, P., and Friesen, W.V., *The Facial Action Coding System*, Consulting Psychologists Press Inc., San Francisco, CA, 1978.

35. Ekman, P., "Facial Expression and Emotion," *American Psychologist*, Vol. 48, pp. 384-392, 1993.

36. Essa, I.A., "Analysis, Interpretation and Synthesis of Facial Expressions," *MIT Media Laboratory*, Perceptual Computing Technical Report 303, Ph.D. dissertation, February 1995.

37. Forney, G.D., "The Vitervi Algorithm," *Proceedings of the IEEE*, Vol. 61, No. 3, pp. 268-278, March 1973.

38. Fridlund, A.J., *Human Facial Expression: An Evolutionary View*, Academic Press, San Diego, CA, 1994.

39. Golomb, B., Lawrence, D., and Sejnowski, T., "SEXnet: A Neural Network Identifies Sex From Human Faces," *Advances in Neural Information Processing System 3*, pp. 572-577, San Mateo, CA, 1991.

40. Gray, R.M., "Vector Quantization," *IEEE ASSP Magazine*, pp. 4-29, April 1984.

41. Hager, J., and Ekman, P., "The Essential Behavioral Science of the Face and Gesture that Computer Scientists Need to Know," *Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition*, pp. 7-11, Zurich, Switzerland, June 1995.

42. Hancock, P.J.B., Burton, A.M., and Bruce, V., "Face Processing: Human Perception and Principal Components Analysis," *Memory and Cognition 24*, pp. 26-40, 1996.

43. Haralick, R.M., and Shapiro, L.G., *Computer and Robot Vision Volume I*, Addison-Wesley, 1992.

44. Hashiyama, T., and Furuhashi T., Uchikawa, Y., and Kato, H., "A Face Graph Method Using a Fuzzy Neural Network for Expressing Conditions of Complex Systems."

45. Heeger, D.J., "Optical Flow Using Spatiotemporal Filters," *International Journal of Computer Vision*, Vol. 1, pp. 279-302, 1988.

46. Himer, W., Schneider, F., Kost, G., and Heimann, H., "Computer-Based Analysis of Facial Action: A New Approach," *Journal of Psychophysiology*, Vol. 5, No. 2, pp. 189-195, 1991.

47. Horn, B.K.P., and Schunk, B.G., "Determining Optical Flow," *Artificial Intelligence*, Vol. 17, pp. 185-203, 1981.

48. Kaiser, S., and Wherle, T., "Automated Coding of Facial Behavior in Human-Computer Interactions with FACS," *Journal of Nonverbal Behavior*, Vol. 16, No. 2, pp. 65-140, 1992.

49. Kanade, T., *Computer Recognition of Human Faces*, Basel & Stuttgart, Birkhauser Verlag, 1977.

50. Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: Active Contour Models," *International Journal of Computer Vision*, pp. 321-331, 1988.

51. Kaucic, R., Dalton, B., and Blake, A., "Real-Time Lip Tracking for Audio-Visual Speech Recognition Applications," In Proc. European Conference Computer Vision, pp. 376-387, Cambridge, UK, 1996.

52. Kim, H.J., and Li, C.C., "A Non-Orthogonal Wavelet Edge Detector with Four Filter-Coefficients," *Mathematical Imaging: Wavelet Applications in Signal and Image Processing*, SPIE, Vol. 2034, San Diego, CA, 1993.

53. Kimura, S., and Yachida, M., "Facial Expression Recognition and its Degree Estimation," *IEEE Computer Vision and Pattern Recognition*, pp. 295-300, 1997.

54. Kirby, M., and Sirovich, L., "Application of the Karhuneh-Loeve Procedure for the Characterization of Human Faces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 1, pp. 103-108, January 1990.

55. Kobayashi, H., and Hara, F., "The Recognition of Basic Facial Expressions by Neural Network," *Proceedings of International Joint Conference on Neural Network*, pp. 460-466, 1991.

56. Kobayashi, H., and Hara, F., "Recognition of Six Basic Facial Expressions and their Strength by Neural Network," *IEEE International Workshop on Robot and Human Communication*, pp. 381-386, September 1992.

57. Kobayashi, H., and Hara, F., "Recognition of Mixed Facial Expressions by Neural Network," *IEEE International Workshop on Robot and Human Communication*, pp. 387-391, September 1992.

58. Lanitis, A., Taylor, C.J., and Cootes, T.F., "Automatic Interpretation and Coding of Face Images Using Flexible Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 743-756, 1997.

59. Lee, K-F, "Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System," *Carnegie Mellon University*, Computer Science Department, Ph.D. dissertation, April 1988.

60. Li, H., Roivainen, P., and Forchheimer, R., "3-D Motion Estimation in Model-Based Facial Image Coding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 6, pp. 545-555, June 1993.

61. Li, Y., and Kobatake, H., "Extraction of Facial Sketch Images and Expression Transformation Based on FACS," *IEEE*, pp. 520-523.

62. Lien, J.J., Kanade, T., Zlochower, A.J., Cohn, J.F., and Li, C.C., "Automatically Recognizing Facial Expressions in the Spatio-Temporal Domain," *Workshop on Perceptual User Interfaces*, pp. 94-97, Banff, Alberta, Canada, October 19-21, 1997. Also available at http://www.cs.cmu.edu/~jjlien

63. Lien, J.J., Kanade, T., Cohn, J.F., and Li, C.C., "Automated Facial Expression Recognition Based on FACS Action Units," *Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 390-395, Nara, Japan, April 14-16, 1998. Also available at http://www.cs.cmu.edu/~jjlien

64. Lien, J.J., Kanade, T., Cohn, J.F., and Li, C.C., "Subtly Different Facial Expression Recognition and Expression Intensity Estimation," *IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara*, CA, June 23-25, 1998. Also available at http://www.cs.cmu.edu/~jjlien

65. Linde, Y., Buzo, A., and Gray, R., "An Algorithm for Vector Quantizer Design," *IEEE Transaction on Communications*, Vol. COM-28, NO. 1, pp. 84-95, January 1980.

66. Lucas, B.D., "Generalized Image Matching by the Method of Differences," *Carnegie Mellon University*, Technical Report CMU-CS-85-160, Ph.D. dissertation, July 1984.

67. Mase, K., and Pentland, A., "Automatic Lip-reading by Optical-Flow Analysis," *Systems and Computers in Japan*, Vol. 22, No. 6, pp. 67-76, 1991.

68. Mase, K., "Recognition of Facial Expression from Optical Flow," *Institute of Electronics, Information and Communication Engineers Transactions*, Vol. E74, pp. 3474-3483, 1991.

69. Matsuno, K., Lee, C-W, Kimura, S., and Tsuji, S., "Automatic Recognition of Human Facial Expressions," *International Conference on Computer Vision*, pp. 352-359, 1995.

70. McNeil, D., "So You Think Gestures Are Nonverbal ?" *Psychological Review*, 92, pp. 350-371, 1985.

71. Morimoto, C., Yacoob, Y., and Davis, L., "Recognition of Head Gestures Using Hidden Markov Models," *International Conference on Pattern Recognition*, pp. 461-465, Austria, 1996.

72. Moses, Y., Reynard, D., and Blake, A., "Determining Facial Expressions in Real Time," *International Workshop on Automatic Face- and Gesture-Recognition*, pp. 332-337, Zurich, Switzerland, June 1995.

73. Murase, H., and Nayar, S.K., "Visual Learning and Recognition of 3-D Objects from Appearance," *International Journal of Computer Vision*, Vol. 14, No. 1, pp. 5-24, 1995.

74. O'Toole, A.J., and Edelman, S., "Structural Aspects of Face Recognition and the other Race Effect," *Memory and Cognition*, 22, pp. 208-224, 1994.

75. Padgett, C., and Cottrell, G., "Representing Face Images for Emotion Classification," *Advances in Neural Information Processing Systems 9*, pp. 894-900, Cambridge, MA, 1997.

76. Paul, D.B., "Speech Recognition Using Hidden Markov Models," *The Lincoln Laboratory Journal*, Vol. 3, No. 1, pp. 41-62, 1990.

77. Poelman, C.J., "The Paraperspective and Projective Factorization Methods for Recovering Shape and Motion," *Carnegie Mellon University*, Technical Report CMU-CS-95-173, Ph.D. dissertation, July 1995.

78. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., *Numerical Recipes in C, The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, 1992.

79. Rabiner, L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of The IEEE*, Vol. 77, No. 2, pp. 257-285, February 1989.

80. Ralescu, A., and Hartani, R., "Some Issues in Fuzzy and Linguistic Modeling," *IEEE Proc. of International Conference on Fuzzy Systems*, 1995.

81. Rinn, W.E., "The Neuropsychology of Facial Expression: A Review of the Neurological and Psychological Mechanisms for Producing Facial Expressions," *Psychological Bulletin*, 95, pp. 52-77, 1984.

82. Rosenblum, M., Yacoob, Y., and Davis, L.S., "Human Emotion Recognition from Motion Using a Radial Basis Function Network Architecture," *University of Maryland*, Technical Report CS-TR-3304, June 1994.

83. Samal, A., and Iyengar, P.A., "Automatic Recognition and Analysis of Human Faces and Facial Expression: A Survey," *Pattern Recognition*, Vol. 25, No. 1, pp. 65-77, 1992.

84. Samaria, F., and Young, S., "HMM-Based Architecture for Face Identification," *Image and Vision Computing*, Vol. 12, No. 8, pp. 537-543, October 1994.

85. Sato, T., Yamaguchi, T., "Generation of Facial Expression Using Chaotic Retrieval," *IEEE Symposium on Emerging Technologies & Factory Automation*, 1994.

86. Scherer, K.R., and Ekman, P., (Eds.), *Approaches to Emotion*, Lawrence Erlbaum Associates, 1984.

87. Starner. T., and Pentland, A., "Visual Recognition of American Sign Language Using Hidden Markov Models", *Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition*, pp. 189-194, Zurich, Switzerland, June 1995.

88. Stork, D.G., and Hennecke, M.E., "Speechreading: An Overview of Image Processing, Feature Extraction, Sensory Integration and Pattern Recognition Techniques," *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, Killington, Vermont, pp. xvi-xxvi, October 1996.

89. Suwa, M., Sugie, N., and Fujimura, K., "A Preliminary Note on Pattern Recognition of Human Emotional Expression," pp. 408-410.

90. Szeliski, R. and Coughlan, J. “Spline-Based Image Registration,” *International Journal of Computer Vision*, Vol. 22, No. 3, pp. 199-218, 1997.

91. Terzopoulos, D., and Waters, K., “Analysis and Synthesis of Facial Image Sequences Using Physical and Anatomical Models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 6, pp. 569-579, June 1993.

92. Tomasi, C., "Shape and Motion from Image Streams: A Factorization Method," *Carnegie Mellon University*, Technical Report CMU-CS-91-172, Ph.D. dissertation, September 1991.

93. Turk, M., and Pentland, A., "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, Vol. 3, No. 1, pp. 71-86, 1991.

94. Ushida, H., Takagi, T., and Yamaguchi, T., “Recognition of Facial Expressions Using Conceptual Fuzzy Sets,” *Proc. of the 2nd IEEE International Conference on Fuzzy Systems*, pp. 594-599, 1993.

95. Valentin, D., Abdi, H., O’Toole, A.J., and Cottrell, G.W., “Connectionist Models of Face Processing: A Survey,” *Pattern Recognition*, Vol. 27, No. 9, pp. 1209-1230, 1994.

96. Vanger, P., Honlinger, R., and Haken, H., “Applications of Synergetics in Decoding Facial Expressions of Emotion,” *Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition*, pp. 24-29, Zurich, Switzerland, June 1995.

97. Vetter, T., and Poggio, T. “Linear Object Classes and Image Synthesis from a Single Example Image,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 7, pp. 733-741, 1997.

98. Viterbi, A.J., “Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm,” *IEEE Transactions on Information Theory*, Vol. 13, pp. 260-269, April 1967.

99. Wallbott, H., “Effects of Distortion of Spatial and Temporal Resolution of Video Stimuli on Emotion Attributions,” *Journal of Nonverbal Behavior*, Vol. 16, No. 1, pp. 5-20, 1992.

100. Waters, K., “A Physical Model of Facial Tissue and Muscle Articulation Derived from Computer Tomography Data,” *SPIE Visualization in Biomedical Computing*, 1808, pp. 574-583, 1992.

101. Wu, Y.T., Kanade, T., Cohn, J.F., and Li, C.C., “Optical Flow Estimation Using Wavelet Motion Model,” *IEEE International Conference on Computer Vision*, pp. 992-998, Bombay, India, January 1998.

102. Xiong, Y., “High Precision Image Matching and Shape Recovery,” *Carnegie Mellon University*, Ph.D. dissertation, 1995.

103. Yacoob, Y., and Davis, L., "Recognizing Human Facial Expression,” *University of Maryland*, Technical Report CS-TR-3265, May 1994.

104. Yamato, J., Ohya, J., and ISHII, K., "Recognizing Human Action in Time-Sequential Images Using Hidden Markov Model," *IEEE International Conference on Computer Vision*, pp. 379-385, 1992.

105. Yang, J., "Hidden Markov Model for Human Performance Modeling," *University of Akron*, Ph.D. dissertation, August 1994.

106. Yuille, A.L., Cohen, D.S., and Hallinan, P.W., “Feature Extraction from Faces Using Deformable Templates,” *International Journal of Computer Vision*, Vol. 8, pp. 104-109, 1992.

107. Zlochower, A.J., Cohn, J.F., Lien, J.J., and Kanade, T., “Automated Face Coding: A Computer Vision Based Method of Facial Expression Analysis in Parent-Infant Interaction,” *International Conference on Infant Studies*, Atlanta, Georgia, April 1998.