

## INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
800-521-0600

**UMI<sup>®</sup>**



## **NOTE TO USERS**

**This reproduction is the best copy available.**

UMI<sup>®</sup>



**University of Alberta**

**FACIAL EXPRESSION ANALYSIS AND SYNTHESIS FOR MODEL BASED CODING**

by

**Lijun Yin**



A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

**Department of Computing Science**

**Edmonton, Alberta  
Fall 2000**



**National Library  
of Canada**

**Acquisitions and  
Bibliographic Services**

**395 Wellington Street  
Ottawa ON K1A 0N4  
Canada**

**Bibliothèque nationale  
du Canada**

**Acquisitions et  
services bibliographiques**

**395, rue Wellington  
Ottawa ON K1A 0N4  
Canada**

*Your file Votre référence*

*Our file Notre référence*

**The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.**

**The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.**

**L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.**

**L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.**

0-612-59702-4

**Canada**

**University of Alberta**

**Library Release Form**

**Name of Author:** Lijun Yin

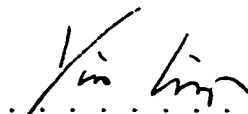
**Title of Thesis:** Facial Expression Analysis and Synthesis For Model Based Coding

**Degree:** Doctor of Philosophy

**Year this Degree Granted:** 2000

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.


  
.....  
Lijun Yin  
GSB 652  
University of Alberta  
Edmonton, AB  
Canada, T6G 2H1

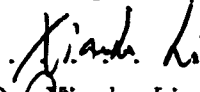
**Date:** *July 15, 2000*  
.....

University of Alberta


Faculty of Graduate Studies and Research


The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Facial Expression Analysis and Synthesis For Model Based Coding** submitted by Lijun Yin in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.


  
.....  
Dr. Anup Basu

  
.....  
Dr. Xiaobo Li

  
.....  
Dr. Paul Sorenson

  
.....  
Dr. Mrinal Mandal

  
.....  
Dr. Ehab S. Elmallah

  
.....  
Dr. Jonathan Wu

Date: *July 7, 2000*



# Abstract

The growing interest in video communication in areas such as education, entertainment, and business (videoconferencing) makes video compression an inexhaustible research topic. The greater the prior knowledge of objects there is in the image explored, the less the amount of information to be transmitted. Model-based coding is just one instance of this principle, which has attracted many researchers both in computer vision and in computer graphics, to attempt to build a framework, especially for the human face. After a decade of efforts, it is still a challenging research topic. The primary difficulties are: (1) how to model a face *automatically*, (2) how to model and track the expressions *accurately*, and (3) how to make the synthesized face look *realistic*.

In this thesis, four new approaches to deal with the above three issues are proposed. They are: (1) Automatic face modeling based on two views of a face: unlike the traditional methods of using the laser scanner and the techniques of shape from shading or shape from contour, a new approach to automatically generate a 3D facial model is presented, in which an individual 3D facial model is constructed by fitting a generic head model to front and side views of a person's head. (2) Facial feature shape detection: a color-based deformable template feature detection with active tracking is proposed. The method is the first attempt to incorporate the facial feature detection with tracking by an active camera. (3) Physics-based coarse-to-fine model adaptation: an energy minimization adaptation method is proposed for tracking the facial expression accurately. There are two steps consisting of physics-based dynamic mesh matching and energy-oriented mesh fitting. This method overcomes the convergence problem of the numerical solution for the elastic motion. (4) Active texture detection, compression, and synthesis for producing a realistic face: a partial active texture up-

date scheme is proposed, which deals with not only the facial organ features, but also the facial wrinkles. It efficiently reduces the computation load and the bit cost in transmission. The life-like face is synthesized using a new temporal-spatial blending technique. The feasibility and advantage of the proposed work are demonstrated by video sequences.

The human face is interesting and challenging because of its familiarity and diversity. People can recognize a face from vast universe of similar faces and are able to detect very subtle changes in facial expression. These skills are learned early in life, and they rapidly develop into major channel of communication.

- Frederic I. Parke and Keith Waters

To my beloved wife Dr. Xiao Fang

# Acknowledgements

I would like to express my appreciation to those who have helped me in the duration of my pursuing the Ph.D degree. I am especially thankful to Dr. Anup Basu, my supervisor, for his support, advice, and encouragement through this research. I would also like to thank my supervisory committee members, Dr. Xiaobo Li and Dr. Ben Watson. Their constructive discussion, suggestions, comments, and friendship, were valuable for my completing this work. My appreciation also goes to my defence committee, Dr. Paul Sorenson, Dr. Mrinal Mandal, Dr. Ehab S. Elmallah and Dr. Jonathan Wu for their very valuable comments and critique to my thesis. I would like to thank Dr. Joe Culberson for chairing my defence committee. I am also very grateful to Anne Nield, for the time she spent helping me proofreading this thesis, and for the encouragement that she gave me.

I would like to thank Dr. Hong Zhang, Dr. Lettice Tse, Dr. Terry Caelli, Jonathan Baldwin and Stefan Bernoegger for their very good suggestion and discussion on my research work, to Steve Sutphen and Rob Lake for their laboratory support. Thanks also go to Sunrose Ko, Edith Drummond, and Carol Smith for their concern and friendship, to Dr. Mario Costa Sousa, Dana Cobzas, Xun Tan and the colleagues and students in the Department of Computing Science for voluntarily providing their wonderful expressions in front of my video camera so that the original video sequences were made.

I am indebted to my family, my dream would have not come true without their inspiration, moral support and their sacrifice. Thanks go to my father, Zhengzhong Yin; my mother, Xiuli Yin; my mother-in-law, Prof. Cun Mei; my father-in-law, Dr. Dinghua Fang; and my grandmother, Ms. Meifang Bao who brought me up and gave me the first impression of what is the truth, the kindness, the beauty and the love.

And most of all, my warmest thanks to my cherished and beautiful wife, Dr. Xiao Fang, who dedicated herself for my struggling for my dream. Her love, wisdom, understanding, sacrifice, and patience, kept me going through all of it.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Principle of analysis and synthesis coding . . . . .	2
1.2	Synthetic/Natural Hybrid Coding (SNHC) within MPEG-4 . . . . .	3
1.3	Thesis overview . . . . .	7
1.3.1	Problems addressed . . . . .	9
1.3.2	Contributions of the thesis . . . . .	11
<b>2</b>	<b>Face modeling</b>	<b>14</b>
2.1	Overview . . . . .	14
2.1.1	Facial model property . . . . .	14
2.1.2	Facial model representation . . . . .	15
2.1.3	Facial model individualization . . . . .	16
2.2	Individual Face Modeling System . . . . .	18
2.2.1	Analysis of feature points . . . . .	20
2.2.2	Generic Model Modification . . . . .	25
2.3	Experiments . . . . .	27
2.4	Discussion . . . . .	27
<b>3</b>	<b>Facial motion estimation with active tracking</b>	<b>30</b>
3.1	Facial motion tracking — A brief review . . . . .	30
3.1.1	Facial feature detection . . . . .	31
3.1.2	Facial motion detection . . . . .	32

3.2	Head tracking . . . . .	35
3.2.1	Background compensation . . . . .	36
3.2.2	Moving head detection . . . . .	40
3.3	Facial organ features localization . . . . .	43
3.3.1	Location of feature centers using global region growing . . . . .	44
3.3.2	Location of feature regions using local region growing . . . . .	48
3.4	Shape estimation . . . . .	51
3.4.1	Detection and tracking of eye shape - brief overview [6] . . . . .	52
3.4.2	Detection and tracking of eyebrow . . . . .	55
3.4.3	Detection and tracking of mouth shape . . . . .	57
3.4.4	Detection and tracking of nose shape . . . . .	66
3.4.5	Detection and tracking of chin contours . . . . .	71
3.5	Experimental results . . . . .	73
3.6	Discussion . . . . .	77
<b>4</b>	<b>Model adaptation</b>	<b>80</b>
4.1	Overview . . . . .	80
4.2	Global adaptation of the individual facial model . . . . .	82
4.3	Model adaptation and animation with extended adaptive mesh . . . . .	84
4.3.1	Principle of Energy Minimization in Energy Oriented Mesh . . . . .	86
4.3.2	Detailed Algorithm . . . . .	87
4.4	Experimental Results . . . . .	92
4.4.1	Experiment I . . . . .	92
4.4.2	Experiment II . . . . .	95
4.4.3	Experiment III . . . . .	97
4.5	Discussion . . . . .	99
<b>5</b>	<b>Active texture detection, compression and synthesis</b>	<b>101</b>
5.1	Overview . . . . .	102

5.2	Fiducial points estimation . . . . .	105
5.2.1	Fiducial points . . . . .	106
5.3	Active texture detection . . . . .	107
5.4	Active texture compression . . . . .	111
5.5	Facial texture synthesis and expression generation . . . . .	113
5.6	Experimental results . . . . .	116
5.7	Discussion . . . . .	122
<b>6</b>	<b>Conclusions and future work</b>	<b>124</b>
6.1	Limitations of this thesis . . . . .	125
6.2	Directions for Future Work . . . . .	127
	<b>Bibliography</b>	<b>129</b>
<b>A</b>	<b>Proof of pin-hole camera model of Section 3.2.1</b>	<b>135</b>
<b>B</b>	<b>Proof of pan-tilt camera model of Section 3.2.1</b>	<b>137</b>
<b>C</b>	<b>Sample frames of eight video sequences of Table 3.1</b>	<b>140</b>



# List of Figures

1.1	An example of a MPEG-4 (SNHC) implementation . . . . .	4
1.2	Flow chart of the system composition . . . . .	8
2.1	Face modeling diagram . . . . .	19
2.2	Feature points definition . . . . .	19
2.3	A Person's Face in Two Views . . . . .	20
2.4	Processed two views . . . . .	20
2.5	Curve tracing rule: Two priority modes: PMc (clockwise) and PMa (anticlockwise). . . . .	22
2.6	Example of LMCT tracing algorithm . . . . .	24
2.7	Flow chart of LMCT tracing algorithm . . . . .	24
2.8	Fiducial Points Detection . . . . .	25
2.9	Model adjustment by vertices interpolation . . . . .	27
2.10	Two views' texture combination. $I_{90}^p$ (or $I_0^p$ ) is the texture warped from 90° view (or 0° view) to $t^\circ$ view. . . . .	28
2.11	Matching the generic model to a face in two views . . . . .	28
2.12	Individualized 3D Facial Model . . . . .	29
3.1	Review of existing methods for facial motion estimation . . . . .	30
3.2	Camera coordinate system and image coordinate system in pin-hole camera model . . . . .	37
3.3	3D point projected on two image planes with the same lens center . . . . .	39
3.4	Diagram of head detection . . . . .	40

3.5	Illustration of skin region and feature region . . . . .	43
3.6	Diagram of the feature region location using global region growing . .	45
3.7	Diagram of the feature region classification and feature center estimation	46
3.8	Classified features (The regions in the same class are labeled by the same grey scale label; the bottom terminal of each line is the center of each feature). . . . .	49
3.9	Extracted feature blobs. . . . .	50
3.10	The restricted window area based on the extracted feature blobs. . .	51
3.11	Deformable Template (model, initialization). . . . .	53
3.12	Image fields used for computing the potential energy (image, satura- tion, edge). . . . .	53
3.13	Sequence (sample frame 1, 24, 68, 169) showing eye movement. Original images (top); extracted eye template (bottom). . . . .	55
3.14	Example of the horizontal integral projection for eyebrow . . . . .	56
3.15	Example of the vertical integral projection for eyebrow . . . . .	57
3.16	Example of the eyebrow shape extraction using the consecutive integral projection . . . . .	57
3.17	Deformable template for mouth-open . . . . .	58
3.18	Deformable template for mouth-closed . . . . .	58
3.19	Vertical projection for determining the width of a mouth . . . . .	60
3.20	Horizontal projection for determining the corners of a mouth . . . . .	60
3.21	Example of the mouth corner estimation . . . . .	61
3.22	Horizontal projection for determining the mouth-closed and mouth-open	61
3.23	Example of the integral projection of the hue component for a mouth	62
3.24	Example of the horizontal projection of Hue for mouth-closed/open .	62
3.25	Sample frames of the deformable template matching on open/closed mouths . . . . .	67
3.26	Template of nostrils . . . . .	68

3.27	Template of nose side . . . . .	70
3.28	Sample frames of the deformable template matching on nostril and nose sides . . . . .	71
3.29	Deformable template of a chin . . . . .	72
3.30	Sample frames of the deformable template matching on chin contours	72
3.31	<i>1st row</i> : Original active video sequence (frame 10, 24, 41); <i>2nd row</i> : Compensated frame differences; <i>3rd row</i> : Noise remove using morphological filtering ( <i>kernel</i> = $9 \times 9$ ); <i>4th row</i> : Consecutive frames fusion; <i>5th row</i> : Motion areas detected by smoothed fusion frames ( <i>closing</i> ); <i>6th row</i> : Contours of motion areas; <i>7th row</i> : Detected silhouette of the motion head. . . . .	75
3.32	Top: coarse regions detected by color-based two-step region growing; Bottom: detected facial features (iris, eye, eyebrow, nostril, nose sides, mouth, and chin) on frame 10, 24, 41 of video sequence “Guan” . . .	76
4.1	Landmark points used in the estimation of the scale, orientation and position of a motion face for the global adaptation of a model. . . . .	83
4.2	Node displacement within a triangle area in each step: $n_1$ : node start position; $n_2$ : node end position; $n_3$ : node end position in next step. .	90
4.3	Schematic diagram of the extended adaptive mesh method. . . . .	92
4.4	Extracted eye templates in an actual sequence of eye images (frame 1, 24, 68, 169). . . . .	93
4.5	Eye movement. [top]: coarse adaptation using DM method ( $m_i = 1.2, \gamma_i = 1.2, k_1 = 1.0, k_2 = 9.0$ , threshold of $v_i$ and $a_i$ are 60.0 at the time of stopping adaptation). [bottom]: fine adaptation using EOM method (the largest nodal force is 0.05 when adaptation is stopped). .	93
4.6	Overlapped eye mesh: coarse adaptation (top); fine adaptation (bottom). .	94

4.7	Synthesized images using the first frame texture: texture-mapped results of coarse adaptation mesh (top) and the fine adaptation mesh (bottom). . . . .	94
4.8	Defined eye feature vertices. . . . .	94
4.9	Matched and animated 3D eye model . . . . .	94
4.10	Original facial image (man from SGI), the individualized model and the adaptation result. . . . .	96
4.11	Synthetic facial expressions (eye regions), frames 1, 10, 24, 36, 45, 70.	96
4.12	Synthetic facial expressions (eyes, lips, etc.), frames 80, 85, 90, 120, 130.	96
4.13	Original video input (Mario: frames 15, 18, 43, 63) . . . . .	97
4.14	Face model adaptation (frames 15, 18, 43, 63) . . . . .	97
4.15	Model adapting on a video sequence with a pan movement of an active platform (Dima: frame 3, 21, 33, size of 480*480 pixels per frame). top row: original frames. bottom row: results of model adaptation. . . .	98
4.16	Model adapting on a video sequence with a tilt movement of the active platform (Stan: frame 17, 28, 39, size of 640*480 pixels per frame). top row: original frames. bottom row: results of model adaptation. .	98
5.1	Flow chart of the active texture (AT) detection and synthesis . . . . .	106
5.2	(from left to right) (a) Feature points defined on a human face; (b) An example of feature points extracted on a person's face; (c) textures of interest; (d) Normalization of textures of interest (from left to right, UP: forehead area, glabella area, and left-right crowsfeet area; DOWN: left-right nasolabial area, mouth, and left-right eye areas. . . . .	107
5.3	Adapted models: frames 1 and 21. . . . .	107
5.4	Texture transformation within a triangle . . . . .	108
5.5	Example of whole face texture transform: from 15 <sup>th</sup> model (texture) to 1 <sup>st</sup> frame model (texture). . . . .	109

5.6	Diagram of active texture (AT) detection . . . . .	110
5.7	The weight value selection for texture blending . . . . .	114
5.8	Close view of texture synthesis. (top) <i>Nasolabial</i> : (original frame25; no Blend(B)&Filter(F); with B&F). (bottom) <i>Forehead</i> : (original frame63; no B&F; with B&F). . . . .	115
5.9	Expression synthesis using partial texture update and blending technique. Left two (frame 15 and frame 19): synthesized 2D facial texture using the 1st frame texture and the active textures of frame 15 and frame 19, respectively; Right two: texture-mapped 3D model using the left synthesized textures on frame 15 and frame 19. . . . .	115
5.10	Facial texture synthesis: Row1: fitted model; Row2: original texture; Row3: normalized face texture; Row4: normalized textures of interest; Row5: updated textures of interest; Row6: synthesized face; (from left to right: frame 20, 32, 42, 62 of video Mario). . . . .	117
5.11	Chart of temporal correlation of wrinkle textures between the first frame (left-most face of the top row) and the subsequent frames. The input video shows a person's face demonstrating different expressions from left to right, <i>i.e.</i> , initial natural expression, eyes closing, smiling, laughing, worried and sad, surprised . . . . .	119
5.12	Chart of temporal correlation of mouth and eyes textures between the first frame and the subsequent frames. The input video shows a person's face demonstrating different expressions from left to right, <i>i.e.</i> , initial natural expression, eyes closing, smiling, laughing, worried and sad, surprised . . . . .	120

5.13 (from left to right) (a) Original 1st frame; (b) Original 15th frame; (c) Synthesized face (15th frame) without wrinkle texture update (only 1st frame texture used); (d) Synthesized face with wrinkle texture update but no blending of the edges; (e) Synthesized face (frame 15) with texture update and texture blending. . . . .	122
C.1 “Mario” (550*700 pixels). left: original frame; right: feature shape detected . . . . .	140
C.2 “Guan” (480*640 pixels). left: original frame; right: feature shape detected . . . . .	140
C.3 “Dima” (480*480 pixels). left: original frame; right: feature shape detected . . . . .	141
C.4 “Stan” (480*640 pixels). left: original frame; right: feature shape detected . . . . .	141
C.5 “Dana” (480*640 pixels). left: original frame; right: feature shape detected . . . . .	141
C.6 “Xun” (480*640 pixels). left: original frame; right: feature shape detected . . . . .	141
C.7 “Claire” (288*360 pixels). left: original frame; right: feature shape detected . . . . .	142
C.8 “Alau” (512*512 pixels). left: original frame; right: feature shape detected . . . . .	142

# List of Tables

3.1	Performance evaluation of the detection of facial features in eight video sequences. (“l/r” stands for “left/right”). Sample frames of these video sequences are shown in Appendix C. . . . .	77
4.1	Performance evaluation of adaptation errors . . . . .	95
4.2	Computation time in DM and EOM. (note that above numbers represent the average time per frame in the individual sequence) . . . . .	99
5.1	Comparison of the partial texture update and whole texture update .	121
5.2	Comparison of with-or-without texture update and blending . . . . .	122

# Chapter 1

## Introduction

The concept of video communications has been established for many years. Recent technological developments have made it possible to provide the new video communication services to the consumer at home, in applications such as education, shopping, and entertainment. There is very little doubt that video communication will be a major communication tool in the 21st century.

With the expected growing demand for video communication services, data communication techniques are becoming increasingly important. Because in a video communication service, the transmission of image sequences contains the most information, compressing this type of data receives the most attention, and therefore is also the background of this thesis.

Traditionally, the most commonly used coding techniques generally remove the temporal and spatial redundancy. Motion in the image sequence is estimated by the block-based motion estimation or optical-flow-based motion tracking. Images, however, are two-dimensional projections of the 3-D world. Changes in an image sequence are caused by changes within this 3-D world – such as object movements or camera movements. Thus, the changes in the image sequence can then best be predicted by describing the changes in the 3-D world. This philosophy has led to the introduction



of a new image sequence coding technique, called 3-D model-based coding. One development of this idea is currently known as analysis-synthesis coding. The interest in this coding scheme has been strengthened by ISO/IEC-JTC1/SC29<sup>1</sup>, known as MPEG4-SNHC (synthetic/nature hybrid coding) [61].

## 1.1 Principle of analysis and synthesis coding

Within the analysis-synthesis coding principle, an image is viewed as a 2-D projection of the 3-D world, instead of as a statistical signal. Using this point of view, the redundancy in the image sequence can be significantly reduced: If we know that there is an object in the scene, and if the shape and color (texture) of such an object have been determined, it is sufficient to describe these properties *once*. The repeated coding of shape and color details would be *redundant*, at least as long as they remain unchanged.

This is where the principle of analysis-synthesis coding comes in: On the transmitter side, a 3-D model (shape and color) is constructed and adapted to the object in the corresponding input image sequence, and transmitted once. During the scene, only the dynamic model parameters such as 3-D motion parameters or illumination parameters are transmitted, and if necessary, the parameters describing object deformation are also transmitted. On the basis of this information and the knowledge about the model, the scene can be completely reconstructed on the receiver side. This sounds logical and easy to do. However, the first difficult task to be encountered is the construction of a 3-D model from 2-D image data. To date, modeling an arbitrary shape of an object is too complicated to be used, and hence prior knowledge must be involved. Having more *knowledge* about the scene, in most cases, increases the accuracy of the object's model, and consequently the input images can be predicted

---

<sup>1</sup>ISO/IEC-JTC1/SC29 stands for International Organization for Standardization/International Electrotechnical Commission/Joint Technical Committee 1/Scientific Committee 29

more accurately. For some applications, the type of images is indeed restricted. In the videophone situation, for example, the face of the speaker will always be visible, thus the major object to be modeled is the face.

The structure of an analysis-synthesis transmission system is depicted in Figure 1.1. On the transmitter side, image analysis methods are applied to the incoming image sequence in order to extract specific data – such as object description parameters, motion parameters. The analyzed parameters are then transmitted to the receiver side, where they are used to control the reverse process, *i.e.*, the reconstruction of realistic images.

Relying on more *prior* knowledge about a scene, however, will restrict the type of images to which this coding scheme can be applied. To be able to code images from arbitrary sources, this restriction must be removed. A practical solution is to use a Synthetic/Natural Hybrid Coding (SNHC) formed within MPEG-4, which combines the advantages of both 3-D model based coding and hybrid waveform coding. Recently, research on MPEG4-SNHC [13, 1, 5] has been intensified into the aspects of scalable texture coding, animation, and model based coding on body and face models, image analysis and synthesis, object modeling, and 3D model compression.

## **1.2 Synthetic/Natural Hybrid Coding (SNHC) within MPEG-4**

The recent MPEG-4 standard [65] opens new frontiers in the way users will play with, create, re-use, access, and consume audiovisual content. The MPEG-4 object-based representation approach, where a scene is modeled as a composition of objects – both natural and synthetic – with which the user may interact, is at the heart of the

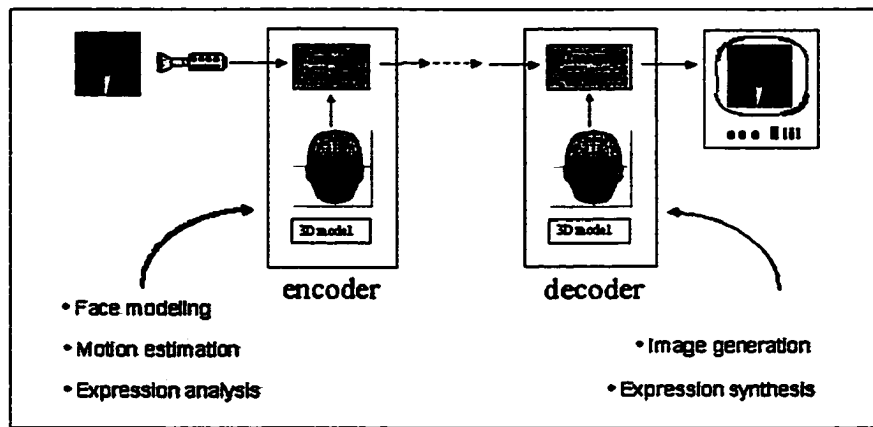


Figure 1.1: An example of a MPEG-4 (SNHC) implementation

MPEG-4 technology [34, 88].

MPEG-4 follows an object-based representation approach where an audiovisual scene is coded as a composition of objects, natural as well as synthetic, providing the first powerful hybrid playground. The objective of MPEG-4 is to provide an audiovisual representation standard supporting new ways of communication, access, and interaction with digital audiovisual data, and offering a common technical solution to various service paradigms – telecommunications, broadcast, and interactive – between which the borders are disappearing. MPEG-4 will supply an answer to the emerging needs of application fields such as video on the Internet, multimedia broadcasting, content-based audiovisual database access, games, audiovisual home editing, advanced audiovisual communications (notably over mobile networks), tele-shopping, and remote monitoring and control.

MPEG-4 is an object-based multimedia compression standard, which allows for encoding of different audio-visual objects (AVO) in a scene, independently. The visual objects may have natural or synthetic content, including arbitrary shape video objects; special synthetic objects such as human face and body; and generic 2-D/3-D objects composed of primitives such as rectangles, spheres, or indexed face sets, which

define an object surface by means of vertices and surface patches. The synthetic visual objects are animated by transforms and special purpose animation techniques, such as face/body animation and 2D-mesh animation.

MPEG-4 foresees that “talking head model” will serve an important role in future customer service applications. For example, a customized agent model can be defined for games or web-based customer service applications. To this effect, MPEG-4 enables integration of face animation with multimedia communications and presentations, and allows face animation over low bit-rate communication channels, for point to point as well as multi-point connections with low delay.

Specifying and animating 3D face models, compressing facial animation parameters, and integrating face animation with TTS (text-to-speech interface) are the major contents in SNHC for face animation of MPEG-4. MPEG-4 specifies a face model in its neutral state, a number of feature points on this neutral face as reference points, and a set of face animation parameters (FAPs) – each corresponding to a particular facial action deforming a face model in its neutral state. Deforming a neutral face model according to some specified FAP values at each time instant generates a facial animation sequence. The FAP value for a particular FAP indicates the magnitude of the corresponding action, *e.g.*, a big versus a small smile or deformation of a mouth corner. For an MPEG-4 terminal to interpret the FAP values using its face model, it has to have predefined model-specific animation rules to produce the facial action corresponding to each FAP. The terminal can either use its own animation rules or download a face model and the associated face animation tables (FAT) to produce a customized animation behavior. Since the FAPs are required to animate faces of different sizes and proportions, their values are defined in face animation parameter units (FAPUs). The FAPUs are computed from spatial distances between major facial features on the model in its neutral state.

SNHC-MPEG4 adopts a model-based approach that allows user-defined facial models to communicate with each other without requiring the standardization of a common facial model. The result of this approach is the definition of 68 face animation parameters as the basic data set that must be supported by all MPEG-4 face decoders. Among the 68 FAPs, two are high-level parameters (visual phoneme – viseme – and expression), and the others are low-level parameters that describe the movements of facial features defined over jaw, lips, eyes, mouth, nose, cheek, etc. Unlike the low-level parameters, the viseme and expression parameters describe facial movements at an abstract level, and each is a compound parameter consisting of multiple sub-parameters.

Synthetic/Natural Hybrid Coding (SNHC), formed within MPEG-4, is the integration of various multimedia data types into a single representation and presentation environment. It efficiently represents a number of data types, such as: (1) Video ranging from very low bit-rate to very high quality conditions, (2) Generic dynamic 3-D objects, as well as specific objects such as human faces and bodies, (3) Text and graphics. A major difference from previous audiovisual standards, at the basis of the new functionalities, is the object-based audiovisual representation model that underpins MPEG-4.

It is important to note that MPEG-4 only specifies the decoding of compliant bit streams in an MPEG-4 terminal. The encoders possess a large degree of freedom in how to generate MPEG-4 compliant bit streams. Because MPEG-4 does not specify any algorithms for facial feature detection, motion parameter estimation, adaptation of a model to the feature point locations, a real application system is far from the target at the current stage. This, therefore, motivates us to investigate the problems that standards cannot solve, and directs us toward the realization of a real 3D model

based coding system, which demonstrates highly realistic quality while keeping very low bit-rate transmission.

### 1.3 Thesis overview

The goal of this thesis is to address the issue of generating the realistic facial expressions from real video sequences for very low bit-rate model-based coding.

Model-based coding, within Synthetic/Natural Hybrid Coding, is a typical merger of the fields of Computer Vision, Computer Graphics, and Image Coding. Explicitly stated, it is the ability to understand the image contents (Computer Vision), which are represented in the most efficient way (Image Coding), and used to drive a graphical reconstruction of the input images (Computer Graphics). Although all of these aspects play an important part in the success of *realistic* facial expression generation, the ability to understand the image contents (and coupled with that the derivation of the description of the object's model from 2-D images) is a premise of its success, which is lacking in the MPEG4-SNHC standards.

The current development of very low bit-rate model-based coding is far from ideal, as reported by many research groups of MPEG-4, especially on the subjective quality: *less realistic*. The current methods of synthesis of face expressions are mainly based on the Face Action Coding System (FACS) [20, 21], which was the psychological discovery of face expression generation. It has been shown to have difficulty producing a large variety of complicated facial expressions [25], especially in the area of facial wrinkles. Therefore, in the development of model-based coding to date, the most challenging problems are still in that of automatic face modeling and realistic facial expression reproduction. In this thesis, we address the core problem of *how to produce an active virtual face to faithfully mimic real human actions without loss of fidelity in*

the condition of only two-dimensional visual image input. This challenging problem is divided into several sub-problems, which become the focus of this thesis: individual face modeling; motion parameter estimation with active tracking; face model adaptation; and active texture extraction, compression, update and synthesis. The whole system that we have developed is shown in Figure 1.2.

In the encoding side, there are three major modules for video analysis to produce

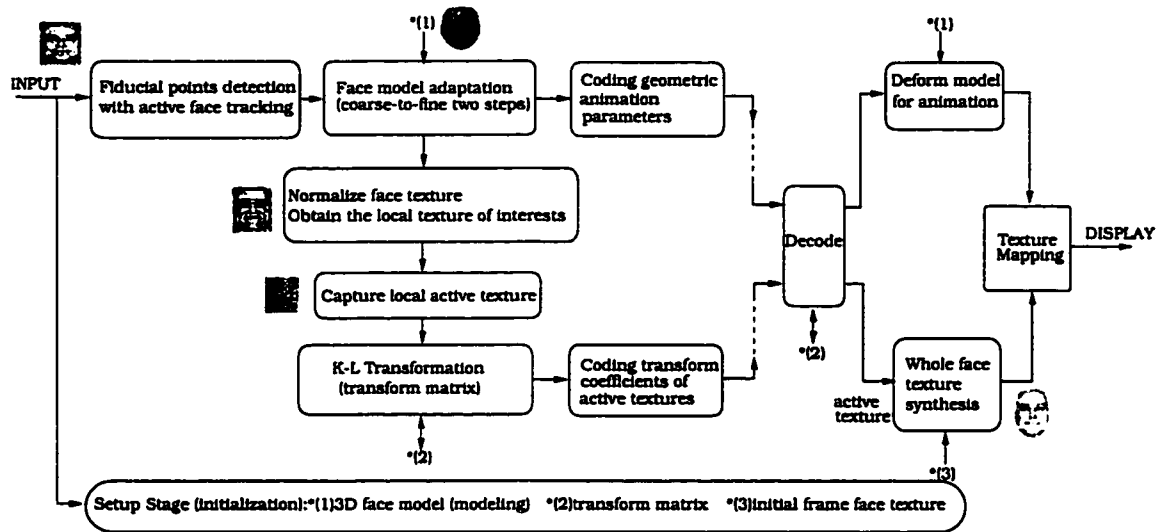


Figure 1.2: Flow chart of the system composition

the animation and texture parameters. These are: (1) a module of off-line setup for an individual head model generation, texture base (*i.e.*, transform matrix) generation for K-L transformation, and an initial frame texture setup in both sides, (2) a module of motion parameter estimation with active tracking for model adaptation, and (3) a module of the active texture detection and compression. In the setup stage, an individual 3D face model is constructed from 2D face images. The off-line setup starts before a real communication begins. A transform matrix is formed in this stage for active texture compression using K-L (Karhunen-Loeve) transformation, an initial face texture corresponding to the natural expression of the person is stored in both sides as the common knowledge. After the system setup, the real video sequence is input into the system and the geometric animation parameters are estimated by the

facial feature detection and facial model adaptation. The face model is fitted onto the face with an active tracking through the real video sequence. The extracted facial motion parameters can be encoded using the “PCA+DCT” method (principal component analysis and discrete cosine transform) recommended by the MPEG4-SNHC. The dynamic extraction of facial active textures including the expressive wrinkles are performed in the mean time, and the local active texture is then encoded with the principal component analysis method. These two types of video codes are the necessary information that needs be transmitted or stored eventually.

In the decoding side, the decoded motion parameters are used to deform the pre-stored individual facial model to achieve an animation. In the mean time, the active textures are rendered on the initial frame textures to synthesize the whole face texture, and finally mapped onto the animated facial model to generate a realistic appearance of the facial expressions.

### 1.3.1 Problems addressed

In this thesis, the first problem addressed is the creation of a 3-D facial model from 2-D images. An efficient approach to reconstruct a 3-D face model is presented. In order to accomplish this task successfully, a generic model is assumed available. We further study the *automatic* acquisition of the shape of an individual face in the scene, given a generic model and 2-D images from multiple views. We chose two-views face images as the input; 3-D information is derived from the two 2-D planes in orthogonal views. This simple and efficient method performs well in achieving the automatic face modeling. Notice that this 3D face modeling procedure works off-line, in other words, the individual face model must be built before the communication starts. This is the first constraint that we imposed in our system.



The second problem addressed is the facial feature extraction and motion parameter estimation with active tracking through the real video sequences. Combining the active tracking with local facial feature extraction results in a more robust estimation of the feature points. The feature detection takes place as the basis of a characteristic description of the region of interests in the 2-D images. A two-steps region growing and a color-based deformable template are the major approaches to achieve this task.

The third problem addressed is the adaptation of the individualized facial model onto the real motion of the face. To estimate the face animation parameters, the face model should be fitted to the face features accurately. A coarse-to-fine adaptation algorithm is developed, which is based on the extended physics-based dynamic mesh. The adaptation procedure makes the convergence of the mesh onto a face more stable and accurate. Notice that the face model adaptation includes the global head motion adaptation and local expression motion adaptation. To track the true 3D head motion, a complicated technique needs be explored to estimate the 3D head motion from 2D video sequence, which is so called *structure from motion*. To simplify our investigation, here we impose a certain constraint on the head motion, in which the head motion is assumed not large, a talking face appears with front view in the video sequences.

The fourth problem addressed is the active texture detection, compression, and synthesis for realistic expression generation with wrinkles. This is an important step toward the *realistic* synthesis of facial expressions. We introduce the concept of *active texture*, and show how to extract the active textures using the temporal correlation information. An efficient compression using principal component analysis is employed to produce very few bits for representing the active textures; Moreover, a texture-blending technique is proposed to synthesize the life-like face expressions with little fidelity loss. Notice that the most video sequences used here are in higher resolution

(larger than 480\*480) comparing to the CIF (352\*288) or QCIF (176\*144) format. Because the low resolution image cannot provide enough information for the accurate feature detection and model adaptation, it is necessary to use high resolution images for generating more realistic facial animation with wrinkles.

In Chapter 2, 3-D individual face modeling from two orthogonal views of the images is described. Chapter 3 discusses feature detection with active tracking in detail. The two-step coarse-to-fine model adaptation algorithm is proposed in Chapter 4, and the texture synthesis and compression methods are described in Chapter 5. Finally, concluding remarks and future work will be discussed in Chapter 6.

### 1.3.2 Contributions of the thesis

The major contributions of this thesis are as follows:

- A novel face modeling algorithm is developed which offers its simplicity and efficiency. We use only two views of a face to create the individual facial model *automatically*. The fiducial points defined on the profile of a face and the front view of a face can be extracted by a curve tracing algorithm, *i.e.*, *LMCT (Local Maximum Curvature Tracing) algorithm*, which is our newly proposed approach. This novel strategy provides a simple and fast way to generate an individual person's head model, other than the traditional way by using the complicated and expensive "Cyberware 3D scanner" technique. Although the created model by our approach is not as accurate as the one generated by a 3D scanner, the appearance is good enough for our purpose for very low bit rate video communication. (The detailed contribution of this part is presented in Section 2.2, Chapter 2).

- The issue of the integration of active tracking with the model-based coding is addressed in the first time after reviewing the existing work. Because the active camera motion is allowed in our work, the background constraint is delimited, the moving person can be tracked by the movable camera, which is considered towards a real application of MPEG-4 for low bit rate model based coding (see Section 3.2). We also proposed a new approach to localize the facial feature area, which is called *global region growing* and *local region growing* using skin color informations. The major contribution of this method is that it allows *iteratively* selects the growing seed based on the grown results, and makes the detection of regions of interest more robust and correct than existing methods (see Section 3.3). In order to extract the facial organ's shape, we make full use of the color information (such as saturation and hue) coupled with shape adaptive deformable template to extract the accurate shapes of eyes, iris, mouth, nose-side, nostril and chin. The deformable template matching is divided into several simple epochs, each epoch localize several template parameters, so that the shrink problem is greatly alleviated, the computation load is also greatly reduced comparing with the existing methods (the detailed contribution of this part is presented in Section 3.4, Chapter 3).
- We propose a new algorithm for energy-oriented face model adaptation through the video sequences, solve the convergence problems in physics-based dynamic mesh. The two-steps coarse-to-fine adaptation makes the facial model fit into the facial expression motion very accurately. Because this adaptation follows the content of facial features unlike the interpolation method recommended by MPEG-4 group, it produces more accurate results, which are critical for generating more realistic face animation in model based coding. (The detail contribution of this part is presented in Section 4.3, Chapter 4).

- A partial facial texture update strategy is presented in this thesis. We first proposed a concept called *active texture* (AT) to deal with the most significant texture on a person's face. A new algorithm based on temporal texture correlation and thresholding is developed for active texture detection (see Section 5.3). In addition, a method of temporal texture blending and spatial texture filtering is developed, which produces a smooth perceptual effect (see Section 5.5). To our knowledge, it is the first time that the expressive wrinkle generation has been specifically investigated. We generate the life-like facial expressions under very low bit rate (less than 40kbit/s) while keep the high fidelity. The advantage of the partial texture update scheme is demonstrated by comparing to the whole facial texture update scheme, which requires at least 200kbit/s bit cost. (The detail contribution of this part is presented in Section 5.6, Chapter 5).

# Chapter 2

## Face modeling

### 2.1 Overview

The goal of face animation is to control the modeled faces, over time, such that the rendered facial surfaces have the desired shapes, colors, and textures in each frame of the animated sequence. An important issue is how to geometrically represent faces in ways that allow both effective animation and efficient rendering. The face has a very complex, flexible, three-dimensional surface; it has color and texture variations and usually contains creases and wrinkles. The structural similarity, as well as the considerable variability of the face make face modeling one of the challenging tasks in real applications. The detailed anatomy of a face is a complex dynamic assembly of skin, bones, muscles, and tissue. To date, no facial animation model which represents and simulates this complete, detailed anatomy has been reported. Although for some applications – such as medical visualization and surgical planning – complete detailed models are the ultimate goal, for the application of model-based coding, a complete detailed model is not necessary. Fortunately, only the visible surface is of critical concern to us.

#### 2.1.1 Facial model property

Before discussing face modeling, there are several expected properties that must be taken into account for the facial model design:

- *general (or similar)*: The structure of the facial model should reflect the general appearance regardless of the person's gender, age, and race.
- *flexible*: The model is easy to modify by adding or deleting the control points for a scalable rendering.
- *manipulatable*: The model can be easily controlled in an interactive manner for HCI (human computer interface).
- *individualizable*: The model must be adaptable to the dynamic features of different persons.
- *simple*: A simple structure of the model will be preferred to the real time performance for some applications (*e.g.*,teleconferencing).

There are trade-offs between some of the properties. For example, the high-fidelity rendering of a model benefits from the high resolution of the model; it does so, however, at the expense of simplicity and manipulatability.

### 2.1.2 Facial model representation

Surface representation plays an important role in representing a geometric face model. There are two types of commonly used techniques to describe the 3D surface of a face – free-form parametric surface and polygon mesh surface. The free-form parametric surfaces are defined in terms of control parameters and basis functions, like B-splines, Bezier patches, and nonuniform rational B-spline surfaces (NURBS). The polygon mesh surfaces include regular polygon meshes (triangular or rectangular) and arbitrary polygon networks. The advantage of the parametric surface is based on the fact that it can construct a smoother surface with higher continuity, while the polygon surfaces only keep 0-order continuous ( $C^0$ ). However, the polygon surface has more flexibility in representing detailed and complicated surfaces than the parametric surface does. It is believed that polygon surfaces can approximate the object shape to a

reasonable precision as long as the real surface is sampled densely enough. Moreover, the polygon surface can be rendered efficiently, and be easy to manipulate and modify (adding or deleting control points), unlike the parametric surface. However, no matter what kind of surface is represented, it is still difficult to model detailed facial features, such as wrinkles, using these methods in simple forms.

### **2.1.3 Facial model individualization**

The goal in modeling an individual face is to synthesize the person's face image with high fidelity. Face modeling can be computed on-line (during the transmission of the image data) or off-line (before the transmission starts). Off-line computation can be done in a controlled environment, and allows the use of complex imaging systems such as a 3-D laser scanner [72, 70]. In an image coding system, it is desired that the on-line information can be readily extracted from the input image sequence in order to perform real time modeling. However, because of the difficulty of 3-D information extraction from 2-D images, on-line computation is restricted to 2-D space only. Few methods were reported which adapt the model to a person's face during transmission of images [58, 3]. Most face modeling methods work off-line. Three general approaches are used for determining surface shape for face model; they are (1) *three-dimensional surface measuring techniques*, (2) *interactive surface sculpting*, and (3) *creating new faces by modifying existing faces*.

#### **Three-dimensional surface measuring techniques**

Surface measurement can be accomplished using 3-D digitizers, photogrammetric techniques, or laser scanning techniques. 3-D digitizers require physically positioning a 3-D probe or locator at each surface point to be measured [53]. The process is very time-consuming if large number of points are to be measured. Photogrammetric techniques require taking multiple simultaneous photographs of the face, each from a different point of view. Once the camera projection matrix and the marked corre-

sponding points are known, the 3-D position of the visible points can be computed. The drawback of this method is that it is necessary to put certain landmarks on the face, and so the imaging condition is constrained [52]. Laser scanning systems (*e.g.*, Cyberware [53]) produce a very large regular mesh of data values in a cylindrical coordinate system. Post-processing of the scanned cylindrical data mesh is often required (*i.e.*, spatial filtering for removing noise, surface interpolation for filling in the missing data, and transformations from cylindrical to Cartesian coordinate systems). The main difficulty with this method is ‘missing data’ in certain obscured parts (*e.g.*, the underside of the chin, the nostril area, pupil of the eye) [70].

### **Interactive surface sculpting**

A model may be created using operations analogous to those used by sculptors. The typical example is the ‘3D paint’ technique [76]. The key idea is to interpret a painted 2D image as a 3D surface, where intensity at each pixel is used as the depth of the surface at that point. Interactive surface editors allow the face modeler to create and modify surfaces by interactively manipulating vertex positions or surface control points. Stereo display techniques can be a useful part of such systems.

### **Creating new faces by modifying existing faces**

This method refers to the transformation from a generic facial model into a specific individual face. The existing approaches [38, 48] use multiple views of the face with manually marked feature points to establish the correspondence, then apply the photogrammetric methods to derive the 3-D position of the feature points. Reinders [58] also uses the deformable template and elastic mesh deformation to adapt the face model in a single view, but only a partial 2-D view (no real 3-D view) is generated. To overcome the drawback of manually marking the landmarks, and the partial 2-D views generation, we have developed an automatic face modeling technique from two orthogonal views of a face [80], in which a curve tracing algorithm for profile feature



detection and the model modification method are developed. This technique will be described in the following section.

## 2.2 Individual Face Modeling System

In the past decade, videophones that code head and face movements as deformations to a head model have been reported by several authors [41]. The first important problem is modeling a specific person's face, *i.e.*, creating a 3D individual model. The precise 3D shape of a face can be obtained using a 3D shape acquisition system such as a range scanner [64], but the model obtained is not sufficient for generating human animations. One solution to modeling a specific person is to employ a strategy which adjusts a prototype model to the person's shape [35, 78, 63]. In this section, we describe a more reliable approach which automatically creates the 3D facial model of a specific person by fitting a generic head model to the front and side views of the person's head. In our system, a new algorithm LMCT (local maximum curvature tracing) is proposed to reliably identify features on the profile such as chin tip, mouth, nose tip, and nose bridge. By using the results, exact search areas for facial feature extraction can be limited. The extracted features are used for modifying the generic model to create an individualized face model.

The entire modeling system consists of two stages – the feature extraction stage and the generic model modification stage (as shown in Figure 2.1). In the first stage, the system will acquire the feature points in the two images of a face through the analysis of two views. Using these feature points, images in the system can adjust the generic model in the second stage to fit it to the two views' faces separately. The final individual 3D facial model can be obtained by the combination of the modified 2D models of the front view and the side view.

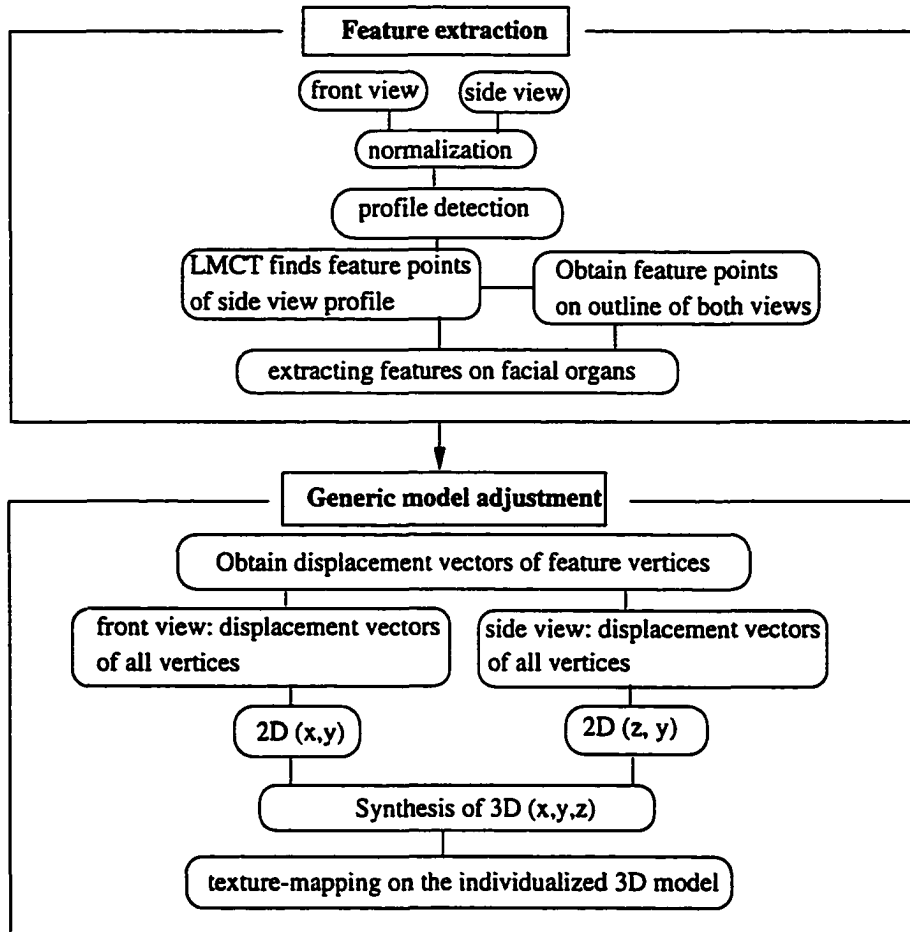


Figure 2.1: Face modeling diagram

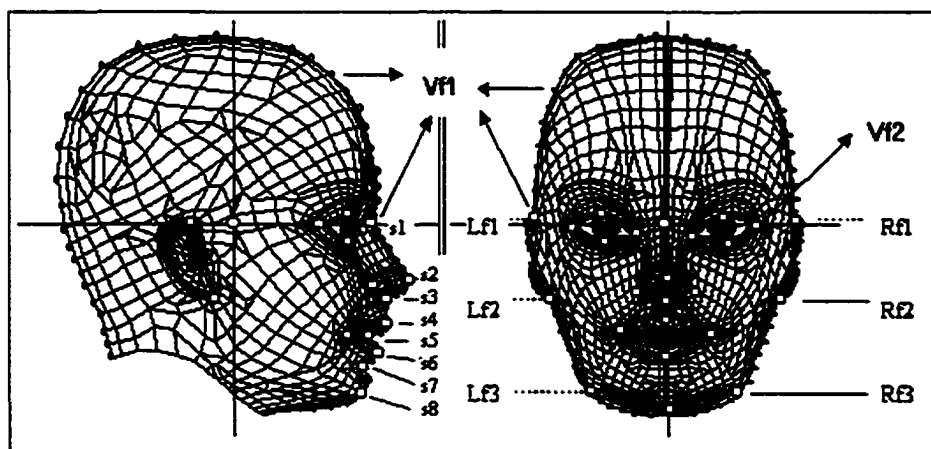


Figure 2.2: Feature points definition

## 2.2.1 Analysis of feature points

In a human face, there are some basic feature points to express the eyes, nose, mouth, face shape and so on. Hence, for the two views of a face, a set of feature vertices is defined on the generic 3D facial model. Two kinds of feature vertices are shown in the facial model (as shown in Figure 2.2):  $V_{f_1}$  (feature points on the outline of a face, shown as dots, in which  $s_1 \sim s_8$  are eight fiducial points on the profile of a face.  $Lf_1 \sim Lf_3$ ,  $Rf_1 \sim Rf_3$  are six fiducial points on the outline of the front view of a face; and  $V_{f_2}$  (fiducial points on facial organs of the interior face, shown as large dots). The center point (o) is an original reference point.

For both views of a face the fiducial points can be located by the following pro-

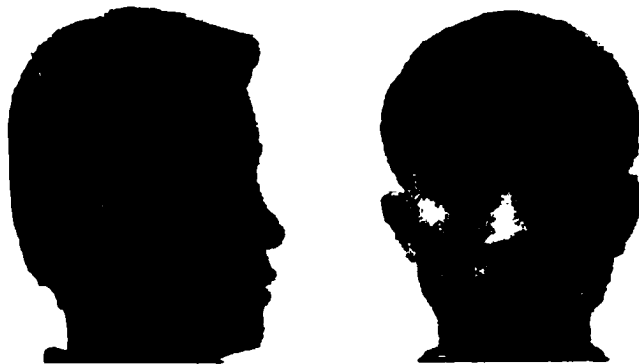


Figure 2.3: A Person's Face in Two Views

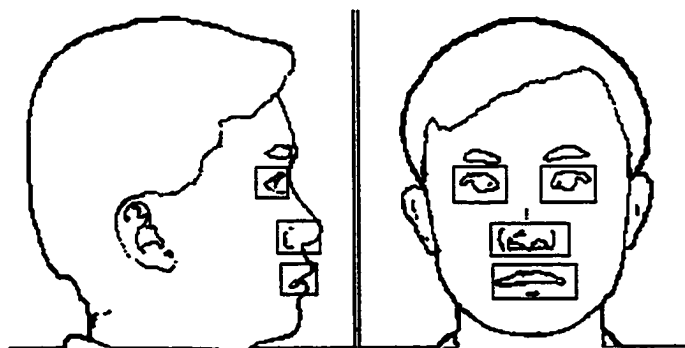


Figure 2.4: Processed two views

cedure: First, the head is separated from the background by using a thresholding

operation, and the profile of the head is extracted by an edge detection operator (Figure 2.3 and Figure 2.4). Second, the vertical positions of facial images, which are represented by eight fiducial points ( $s1 \sim s8$ ), such as center of eye line, nose tip, upper lip, lower lip, and chin tip, are determined by profile curve analysis (Figure 2.8); here a new algorithm for local maximum curvature tracing (LMCT) has been developed for estimating the fiducial points on the profile.

### (1) Profile analysis by a local maximum curvature tracing (LMCT)

The profile of a human face exhibits characteristic features. Eight salient feature points are categorized into two types: concave points (*i.e.*,  $s1, s3, s5, s7$ ) and convex points (*i.e.*,  $s2, s4, s6, s8$ ). The LMCT algorithm uses two kinds of tracing modes with dynamically alternating directions to:

- trace the curve of the facial outline;
- form a series of segment lines approximating the original digitized curve;
- measure the curvatures of the terminals of each segment;
- determine the feature points according to the local maximum curvature.

The algorithm is based on the computed values of the curvature. Suppose a curve is expressed as  $y = f(x)$ . The curvature  $C_n$  at a point  $P_n(x_n, y_n)$  on the curve is defined as:

$$C_n = \frac{y_n''}{[1 + (y_n')^2]^{\frac{3}{2}}} \quad (2.1)$$

where  $n$  is the index number of a point on the curve,  $n = 1, 2, \dots, N$ . The concave points and convex points are related to the points with a sharp change in their curvature values, where a local maximum absolute curvature value can be found. In discrete space, we approximate Equation 2.1 and use the difference of slopes between neighboring line segments to represent the average curvature values for the

corresponding segment terminals as defined by Equation 2.3. Before the LMCT algorithm is explained, a rule for curve tracing is defined. From a certain point on the digital curve, there are five potential paths to proceed (*i.e.*, five neighboring pixels  $p_1, p_2, \dots, p_5$ ), a decision rule must be made to choose a correct path. Two types of *proceeding modes* (PM) are regulated, *i.e.*, clockwise proceeding mode (PMc) and anticlockwise proceeding mode (PMA). PMc gives the five potential paths with a priority in a clockwise order; the priority of the next path to be selected, from high to low, is  $p_1, p_2, p_3, p_4, p_5$ . PMA gives the priority in a reverse order, as shown in Figure 2.5.

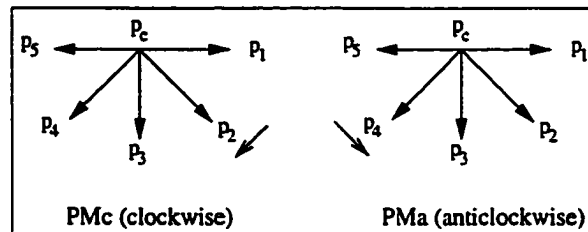


Figure 2.5: Curve tracing rule: Two priority modes: PMc (clockwise) and PMA (anticlockwise).

## (2) LMCT algorithm

To better describe the algorithm, an example is shown in Figure 2.6. The algorithm is composed of two phases: curve tracing and feature points localization. Figure 2.7 depicts the flow chart of the algorithm.

1. To make the curvature calculation less sensitive to noise, the digitized curve is traced in the discrete space to form a “smoother” digital curve. Curve tracing follows two proceeding modes as defined in Figure 2.5 – PMc and PMA, which guide the tracing in a correct path. Two modes are switched dynamically. If the current state is in PMc and the next selected path is  $p_4$  or  $p_5$ , the PMc (clockwise) is switched to the PMA (anticlockwise); If the current state is in PMA and the next selected path is  $p_1$  or  $p_2$ , the PMA (anticlockwise) is

switched back to the PMc (clockwise). In other cases, the proceeding mode remains unchanged.

2. In the course of curve tracing, the pixel in a weak path (*i.e.*,  $p_2$  or  $p_4$ ) is selected to be a new node to form a “smoothed” curve. A set of selected nodes ( $p_0(x_0, y_0), \dots, p_n(x_n, y_n)$ ) form a set of segment lines. The slope of a line  $p_n p_{n+1}$  is calculated by:

$$S_n = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}, n = 0, 1, \dots, N \quad (2.2)$$

3. After obtaining the set  $S_n$ , the curvature  $C_n$  for the node  $p_n$  can be calculated by:

$$C_n = \frac{S_{n-2} + S_{n-1}}{2} - \frac{S_n + S_{n+1}}{2}, n = 2, 3, \dots, N - 1 \quad (2.3)$$

Note that:

- (a) in the convex portion of the curve, there is a local-maximum absolute curvature ( $C_n$ ), while the sign changing from  $S_{n-1}$  and  $S_n$  is “+ to -”;
- (b) in the concave portion of the curve, there is a local-maximum absolute curvature ( $C_n$ ), but the sign changing from  $S_{n-1}$  and  $S_n$  is “- to +”.

### (3) Determination of feature points

Applying the LMCT algorithm, the feature points  $s_2, s_4, s_6, s_8$  of the side view face can be identified by the above 3(a). In the same way, the remaining points  $s_1, s_3, s_5, s_7$  can be identified by 3(b). Similarly, the feature points of  $Lf_1 \sim Lf_3$  and  $Rf_1 \sim Rf_3$  along the outline of the front view of a face can be extracted. The remaining feature points that lie along the outline of a face for both views ( $V_{f1}$ ) can be located by the rule of direct proportional assignment, in accordance with the distribution of  $V_{f1}$  as defined on the generic model. Given the vertical positions of the eye, nose, mouth, and chin ( $s_1 \sim s_8$ ) as determined by profile analysis, we can limit

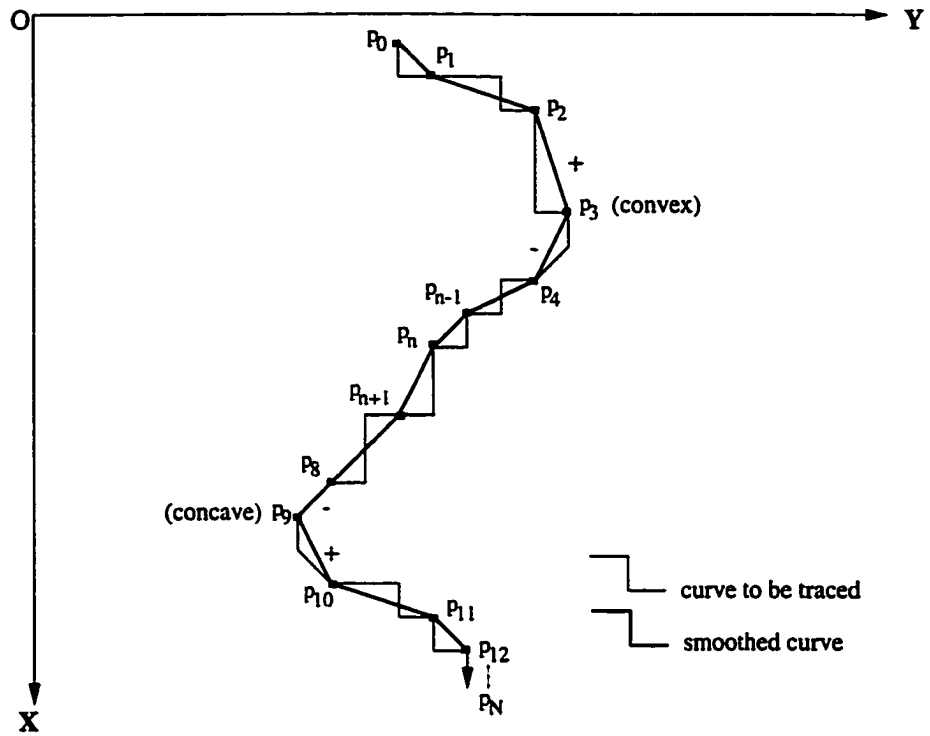


Figure 2.6: Example of LMCT tracing algorithm

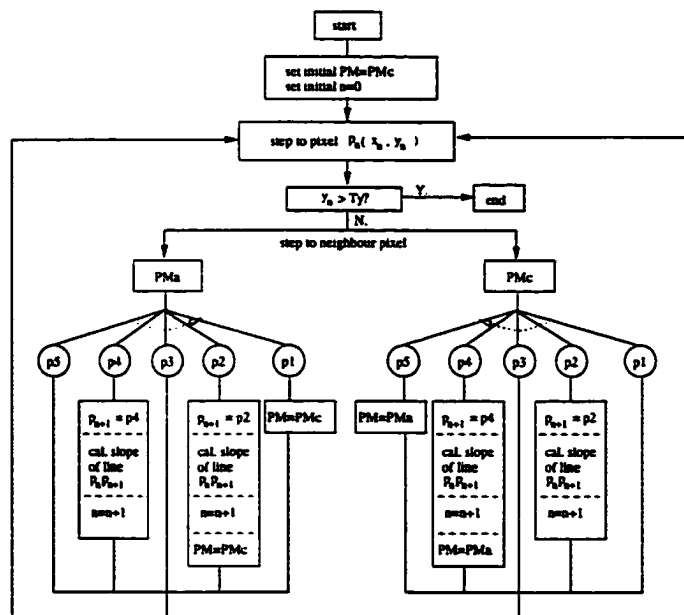


Figure 2.7: Flow chart of LMCT tracing algorithm

the search area for interior feature points. The search area for each organ is restricted to a rectangular window (Figure 2.4), and the size and position of the windows are determined by the vertical position of  $s1 \sim s8$ ,  $Lf1 \sim Lf3$ , and the typical size of the facial features. The interior feature points ( $V_{f2}$ ) can be identified by finding the strong edges and their relative position in each search area. The search area is filtered by a Sobel operator. The strong edge segments within these expectation windows are detected by thresholding the pixel values with a predefined threshold. Finally, the feature points can be estimated by their relative positions along the edge segments – *e.g.*, upmost and downmost, leftmost and rightmost (Figure 2.4). For the front view of the face, the nose tip is determined by the intersection of the vertical fiducial line of the face, the nose tip is determined by the intersection of the vertical fiducial line of the front view and the horizontal position of the nose tip of the side view.

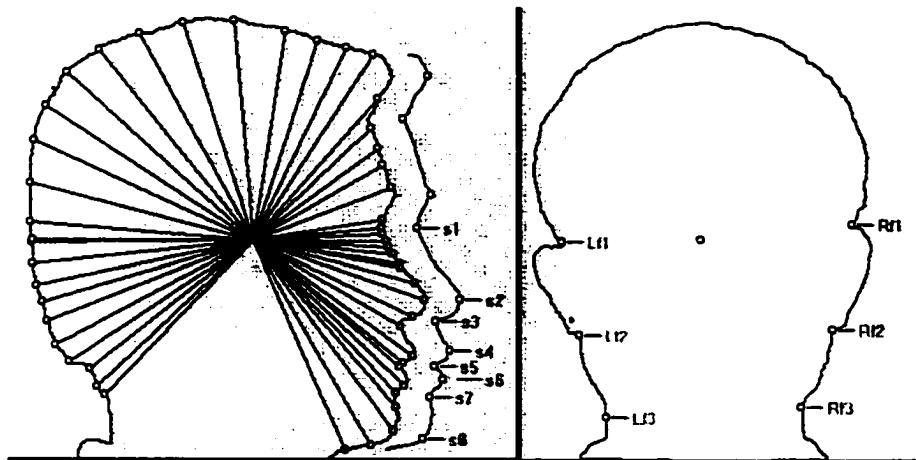


Figure 2.8: Fiducial Points Detection

### 2.2.2 Generic Model Modification

The generic model is modified separately for the front view and side view in 2D space. A 3D individual model is obtained by combining two 2D models modified for each view. The displacement vector ( $DV$ ) is defined as a vector from the position ( $V_f$ ) in the generic model to the corresponding position ( $P_f$ ) in the actual image. The  $DV$



of feature point  $n$  can be specified directly by

$$DV_f^n = P_f^n - V_f^n \quad (2.4)$$

The displacement vectors of non-feature points can be derived by the interpolation of the  $DV$ s of feature points and the derived  $DV$ s of non-feature points in the previous steps. As shown in Figure 2.9, the  $DV$  of non-feature point  $m$  (denoted as  $DV^m$ ) can be derived using the follow equation:

$$DV^m = \sum_{n=1}^N (w(d_{m,n}) \times DV_k^n) \quad (2.5)$$

where  $d_{m,n}$  is the distance between point  $m$  and point  $n$ ,  $d_{m,n}, n = 1, 2, \dots, N$ , (e.g.  $N = 4$ ) have been arranged in increasing order.  $w(d_{m,n})$  is a weight function which will have a large output of weight value for a small input of  $d_{m,n}$ ,

$$w(d_{m,n}) = \frac{d_{m,n'}}{\sum_{n=1}^N d_{m,n}} \quad (2.6)$$

where

$$n' = (N + 1) - n \quad (2.7)$$

The above procedure is applied to the side view and front view of the face separately. Two sets of 2D coordinates of all the vertices are obtained, which are  $(x_f, y_f)$  in front view plane and  $(z_s, y_s)$  in side view plane (Figure 2.11). Therefore, the 3D coordinates of all the vertices can be estimated as follows,

$$(x, y, z) = (x_f, (y_f + y_s)/2, z_s) \quad (2.8)$$

The resulting individual 3D facial model is shown in Figure 2.12. Texture-mapping with blending both views of image ( $I_0$  and  $I_{90}$ ) is shown in Figure 2.10, where

$$I_t = w_1 \times I_{90}^p + w_2 \times I_0^p \quad (2.9)$$

$$w_1 = \sin^2 t \quad (2.10)$$

$$w_2 = \cos^2 t \quad (2.11)$$

$$w_1 + w_2 = 1 \quad (2.12)$$

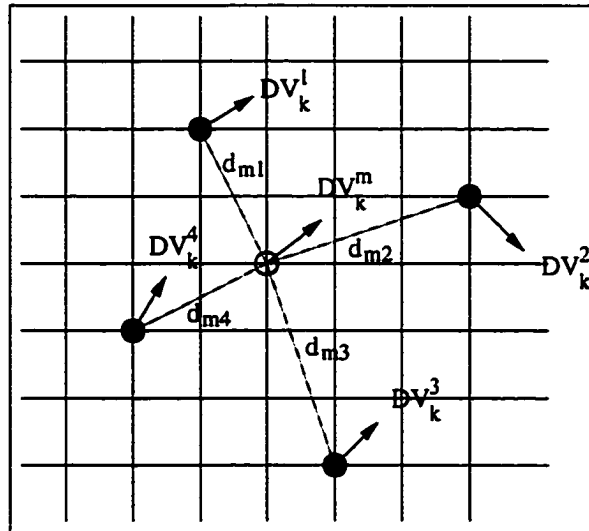


Figure 2.9: Model adjustment by vertices interpolation

## 2.3 Experiments

The experiments are shown for a specific person's facial pictures, taken by a CCD camera. The generic facial model has 2953 vertices and 3118 patches. Figure 2.4 and Figure 2.8 show the results of the analysis of the feature points on both views of the face. Figure 2.11 shows the results of the matching from the generic model to the individual face in both views, while Figure 2.12 shows the generated 3D individual face in different views. The appearance of the 3D individual face generated by the proposed procedure looks natural, and has a good visual effect on the human eye.

## 2.4 Discussion

The face modeling approach proposed here is a very simple and effective scheme requiring only moderate computing and imaging resources. In both the facial fea-

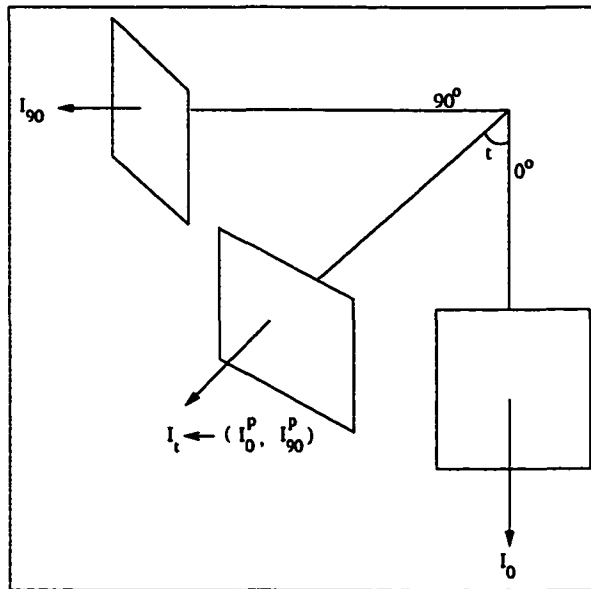


Figure 2.10: Two views' texture combination.  $I_{90}^P$  (or  $I_0^P$ ) is the texture warped from  $90^\circ$  view (or  $0^\circ$  view) to  $t^\circ$  view.

tures identification and model instantiation process, more sophisticated techniques are needed for further improving the results. The following chapters will address the issue for such techniques. In the current implementation, we have assumed that the human face is right-left symmetrical. To generate more accurate individualized head models, other facial views may be added to provide further information on the other side of the face. This scheme is general enough for it to be applied similarly when more facial views are given as input.

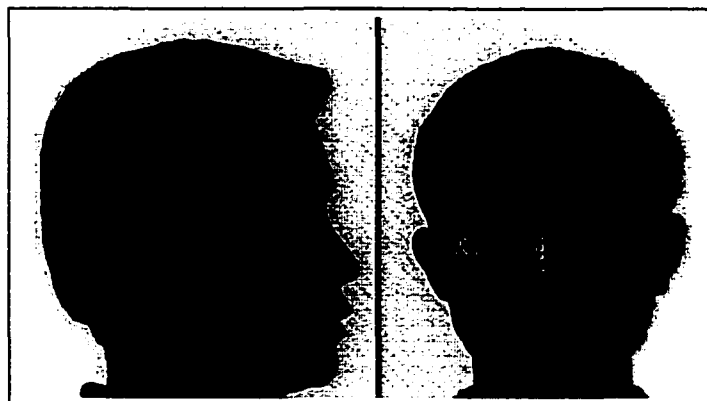
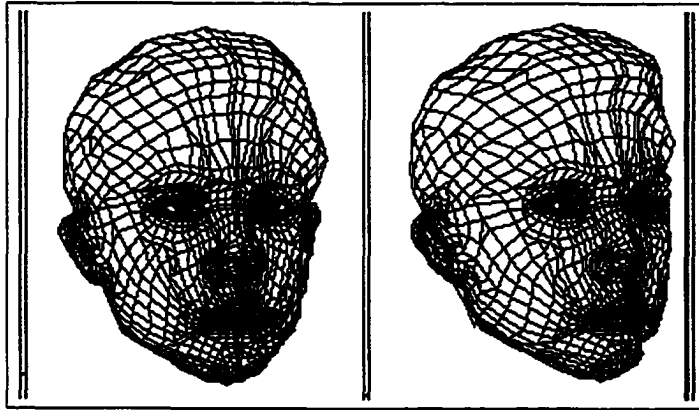
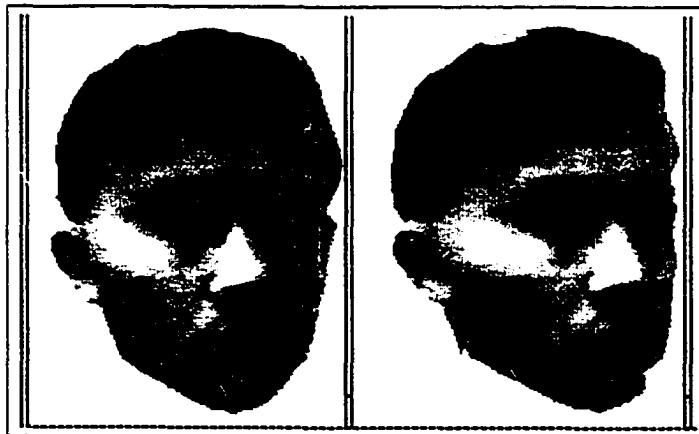


Figure 2.11: Matching the generic model to a face in two views



(a) Generated individual facial model



(b) Texture-mapped individual face in different views

Figure 2.12: Individualized 3D Facial Model

# Chapter 3

## Facial motion estimation with active tracking

### 3.1 Facial motion tracking – A brief review

The estimation of facial motion is classified into two fundamental tracking approaches – feature-based tracking and motion-based tracking. In the feature-based approach, the motion of a face can be derived from the motion of the feature points on the face. This is a feature detection method. The feature is recognized in successive images and its position is extracted. In the motion-based approach, motion detection is the major technique used. The advantage of this method is in that it is able to track any moving object regardless of size or shape. Figure 3.1 depicts the existing methods for facial motion estimation, which will be reviewed in the following sections.

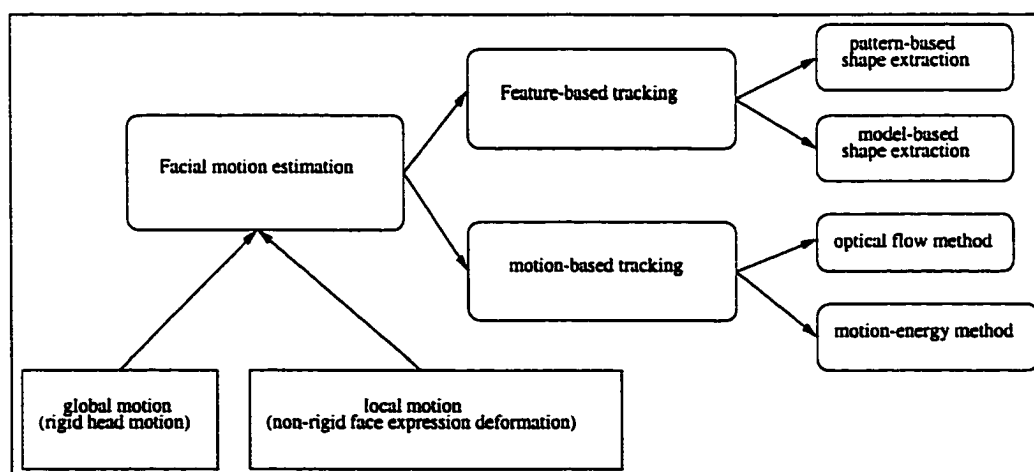


Figure 3.1: Review of existing methods for facial motion estimation

### 3.1.1 Facial feature detection

The feature-based approach locates characteristic points on the face, such as the center points of the eyes. A face can be tracked and recognized on the basis of the positions of these points. Generally, these methods start by detecting one or more characteristic points. Once these points are detected, the acquired information is used to find other characteristic points. The methods differ as to which characteristic points on the face are used, how these points are detected in the image, and how the search strategies for the other new points are defined [12, 8].

Sometimes the feature points may not be sufficient to depict the characteristics of the face, feature shape may need to be extracted for a better representation. Two commonly used approaches are pattern-based shape extraction and model-based shape extraction. In the pattern-based method, the global properties of the object region are emphasized by judging the 2D pattern of intensity variation of that object. To detect the object in an image, a stored version of this pattern is compared with the patterns in the image. For example, the pattern matching method requires that a pattern be generated, usually, on the basis of averaging these patterns over a number of different people. The pattern is matched against the image primitives, thus yielding the position of the object regions. Pattern matching is not shape invariant. Instead of creating the reference pattern by simply averaging the facial feature patterns, it is also possible to create this pattern by using the principle components of the training patterns. When considering faces, these principle components are called eigenfaces. Each individual face region can be represented by the linear combination of the eigenfaces. To detect a face in an image, the patterns in the image are projected onto the face space, which gives a measure for the presence of a face in the present image. In the model based approach, object regions are represented by an explicit geometric description which is related to the image content. To emphasize

the global characteristics of the object region, the measure of fit for all of the parts are evaluated simultaneously, which yields a global objective function. For example active contour (“snake”) and deformable template [74, 86].

While pattern-based methods seem to produce the best results within the problem of face recognition, however, they seem to have serious problems with the invariance requirements. Generally, the methods are not scale and orientation invariant. To be useful within a model-based coding scheme, an accurate description of the shape of facial feature regions is required. The model-based approaches, and in particular the deformable templates, are the most promising approach to extract the shape accurately. Unfortunately, these methods require that the initial parameters describing the model are close to the parameters describing the object region. When estimates for the locations of the facial feature regions are available, these initial model parameters can be predicted for a successful implementation.

### **3.1.2 Facial motion detection**

As mentioned above, feature-based tracking and motion-based tracking [46. 4. 16] are two major approaches to tracking a moving object. Feature-based tracking is really based on the object recognition technique; the performance of the tracking system is limited by the efficiency of the recognition method [12. 4]. Mismatching of the 2-D correspondences of the detected features is prone to cause errors of motion parameters. Motion-based tracking relies on the motion detection technique; it exploits an affine motion model, which incorporates the global head motion, local facial action and the spatiotemporal gradient constraints to estimate the motion parameters directly without establishing 2-D correspondences in advance. It can be divided into the optical-flow (spatiotemporal) method and the motion-energy method. In the optical-flow method, determining a complete optical flow field is ill-posed, and the difficulties increase when using the active camera for searching and matching feature points in

successive images, since the scene viewed is dynamically changeable. The complexity of this problem makes it difficult for real application. The motion-energy tracking method can segment an image into regions of motion and inactivity by calculating the temporal derivative of an image sequence and thresholding at a suitable level to filter out noise. This method is relatively simple and efficient.

In model-based coding for a human face, motion tracking is divided into head motion tracking (global motion) and facial expression tracking (local motion).

### **Global motion (head tracking)**

The motion of a face can be derived from the motion of the associated feature points. Clearly, the position of these feature points may only be changed by the motion of the head, and not by the facial expressions. Thus, points like the corners of the eyes or the position of the nostrils can be used, while corners of the mouth cannot. To detect these feature points and establish their correspondences in the neighboring frames, Aizawa et al. [3, 50] used white marks on the person's face. Welsh [75] established the correspondences for the moving feature points by template matching. The established correspondences were used to discover the best fitting value of the global motion parameters. Essa [23] used the concept of model flow, which is the model motion trajectories between two frames, and actual flow as computed by a general optical flow algorithm, to track the model 3-D position by minimizing the mean square error of these two types of flows. This method made the head tracking with large motion more accurate and robust; however, the computation is very time-consuming. Li and Choi [40, 14, 42] first estimated the global motion by ignoring the local facial expression, a spatiotemporal gradient motion equation was established between the two frames, and the solution was given by solving the linear equations.



## Local motion (expression action)

After estimating the global motion of the head, the local motion is estimated by removing the effect of the estimated global motion with a motion compensation method. Li [40] and Choi [14] used the FACS (facial action coding system) to describe the facial expression. They assumed that the facial expression could be represented by a linear combination of basic action units (AUs). The intensities of the basic AUs can also be estimated by solving the spatiotemporal gradient motion equations [39].

Terzopoulos and Waters [70] used eleven feature points in the areas of eyebrows, cheeks and mouth to estimate the contraction of six muscles. They assumed that the movement of a feature point caused by changing expression could only be originated by one of the muscles (independence between the muscles). When allowing more than one muscles to be active, this assumption runs into serious problems. This method traces the facial features, estimates the corresponding parameters of muscle movement, and reproduces facial expression by using a three layered facial structure. Although the active contour model ('snakes') are used, a significant limitation of this system is that it requires facial features be *highlighted* with make-up, especially on the cheeks.

Instead of using a single feature point, one can use windows in the image (referred as to '*muscle windows*'), each of which is associated with one primary direction of muscle contraction to correctly extract the muscle movement. The motion within these windows can either be estimated by using optical flow [43, 24], or by tracking a set of characteristic points within these windows [79]. These motions can then be used to compute the contraction of the muscles in the face.

Model-based shape extraction is a promising approach for the extraction of facial

feature components (*e.g.*, eye, mouth), such as the deformable template [86]. It uses image features to fit a deformable template to a face component, and the parameters of this template are then used for shape analysis. However, this method is still sensitive to the location of the initial template position. We developed an algorithm by using the two-step region growing to detect the coarse regions of interest, using the Hough Transform to detect the iris and a deformable template with color information to improve the accuracy of feature tracking, such as eyes [6], eyebrow, mouth, nostrils, nose sides and chin contours.

## 3.2 Head tracking

The MPEG-4 standard provides a fundamental framework for multimedia applications at low bit-rates [47, 73]. However, MPEG-4 does not specify the techniques to be used for feature detection and tracking in order to realize an actual implementation. This allows researchers to investigate alternative techniques for different applications. Up until now, most work in MPEG-4 (or related areas) has been limited to the still camera scenario [14, 23, 87, 4, 45, 26, 44, 36, 39] [58] and [73]. In this chapter, we address the problem considering a talking face in front of an active camera. The detection and tracking of the active talking face is a fundamental step towards realizing an application of MPEG-4 in real situations.

Various approaches to the segmentation and feature detection of face images have been attempted, mainly in the field of face recognition. From the large amount of previous work done in this area [12], it has become obvious that face detection and recognition is still a very difficult subject. Segmenting and tracking a talking face from a background is a prerequisite when applying model-based coding schemes (MPEG4-SNHC [73]) to compress face-to-face video communication data [45, 26, 44, 36, 87]. For this purpose, the region of interest (*i.e.*, face region) must be detected and tracked temporally. Detecting the moving part in an image reveals the silhouette region of

the speaker, which can be relatively easily detected on the basis of frame differences if the speaker is the only moving object within the scene, in the case of a still camera. However, in the situation of a movable camera, the difficulties increase since the background viewed is dynamically changeable.

As discussed in Section 3.1.2, the spatio-temporal (ill-posed optical-flow) method provides difficulties in real application. The motion-energy tracking method is relatively simple and efficient, however, it is not suitable for application in an active camera system without modification. Since the active camera system can induce apparent motion on the scenes they view, compensation for camera motion must be made before the motion-energy detection technique can be used.

In this chapter, we present a system to track a talking face with an active camera. After the background compensation in successive frames [46], the motion-energy tracking approach is used, coupled with a morphological filter to reduce the noise. Evidence about moving objects in the scene gathered from a single frame difference may not suffice to portray a speaker entirely. It is necessary to recover the speaker's silhouette by observing a number of successive frame differences (called motion masks). We believe that the motion masks are temporally correlated if the motion of the speaker is assumed to be slow with respect to the frame rate. We propose a progressive silhouette generation method, to detect the motion of a talking head, in which the consecutive motion masks are accumulated to complete the silhouette estimation.

### **3.2.1 Background compensation**

Before applying the motion detection technique, we must compensate for the apparent motion of the background of a scene which is caused by camera motion. Our camera is mounted on a pan/tilt device and hence is constrained to rotate only. The objective in background compensation is to find a relationship between pixels representing the

same 3D point in images taken at different camera orientations. For camera rotation, the only components of the system that move are the camera coordinate system and the image plane. An example of this motion is shown in Figure 3.3. Let us briefly review two types of camera models, pin-hole camera model and pan/tilt model, which are based on the pioneering work by Basu's group [46].

### Camera model

Two kinds of mathematical models are defined for describing the relationships between a 3-D point ( $P$ ), its projection on the image plane ( $p$ ), and the motion parameters of the camera. The coordinate systems and the relationships between them are described as follows:

- **Pin-hole camera model**

As shown in Figure 3.2, the origin of a Cartesian coordinate system  $OXYZ$  is the viewpoint of the camera and its optical axis is aligned with the  $Z$  axis. The image is the projection of a 3-D scene onto a plane located at distance  $f$  from  $O$ , where  $f$  is the *focal length*. The image plane is perpendicular to the  $Z$ -axis and intersects it at a point  $(0,0,f)$ . The image coordinate system is represented by  $oxy$  in Figure 3.2. Using

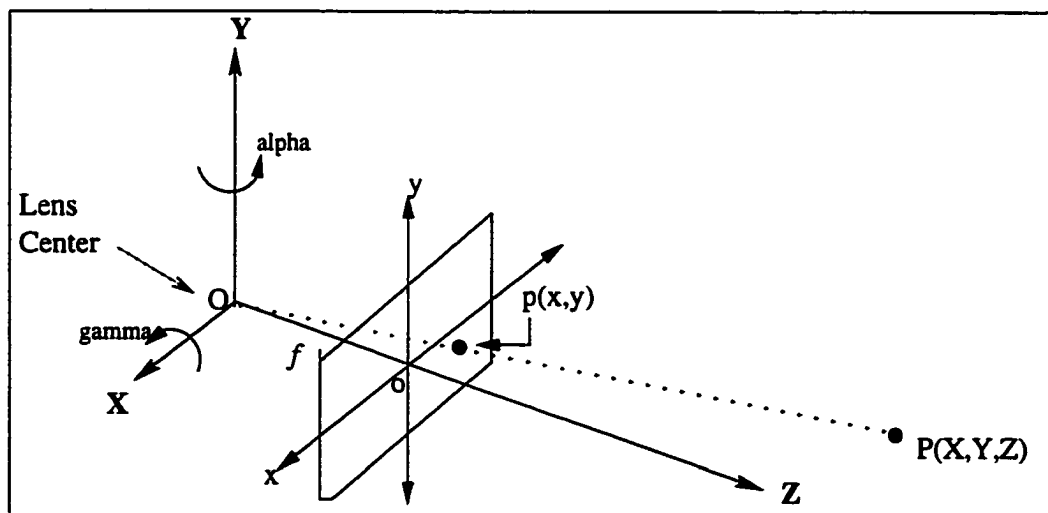


Figure 3.2: Camera coordinate system and image coordinate system in pin-hole camera model

this model and assuming perspective projection, the relationships between points in the image plane and points in 3-D with respect to the camera coordinate system are:

$$x = f \frac{X}{Z} \quad (3.1)$$

$$y = f \frac{Y}{Z} \quad (3.2)$$

where  $P(X,Y,Z)$  is a 3-D location of a point in the camera coordinate system, and  $p(x,y)$  is its projection on the image plane;  $f$  is the focal length.

Consider a camera rotation around  $O$  (lens center) and suppose the camera is rotated by rotation matrix  $R$ , which is an orthogonal matrix, *i.e.*,  $RR^T = I$ . Then, the point in the scene which was seen at  $(x,y)$  on the image plane now moves to another point  $(x',y')$ , the relationship between every pixel position in two images, taken from different positions of rotation about the lens center is as follows:

$$x' = f \frac{x + \alpha\gamma y + \alpha f}{-\alpha x + \gamma y + f} \quad (3.3)$$

$$y' = f \frac{y - \gamma f}{-\alpha x + \gamma y + f} \quad (3.4)$$

where  $f$  is the focal length of the camera;  $\alpha$  is a small angle of rotation about the pan axis ( $Y$ -axis);  $\gamma$  is a small angle of rotation about the tilt axis ( $X$ -axis). (see Appendix A for the proof).

#### • Pan/Tilt model

Above we described the relationship between every pixel position in two images, taken from different positions of rotation about the *lens center*. Consider the real situation, the active camera is mounted on a pan/tilt device that allows rotation about two axes (pan:  $Y$ -axis, tilt:  $X$ -axis). The center of rotation is the device center other than the lens center. The small displacements of the lens center from the center of rotation

will result in a little change of the relationship between every pixel position in two consecutive images, but approximately similar to the case in pin-hole camera model.

In the case of pan/tilt model, the origin of the camera coordinate system is located at the lens center, the origin of the device coordinate system is at the device center. For an initial inclination ( $\theta$ ) of the camera system and pan and tilt rotation of  $\alpha$  and  $\gamma$ , respectively, the relationship between every pixel position in two images is as follows:

$$x_{t-1} = f \frac{x_t + \alpha \sin \theta y_t + f \alpha \cos \theta}{-\alpha \cos \theta x_t + \gamma y_t + f} \quad (3.5)$$

$$y_{t-1} = f \frac{-\alpha \sin \theta x_t + y_t - f \gamma}{-\alpha \cos \theta x_t + \gamma y_t + f} \quad (3.6)$$

where  $f$  is the focal length. With knowledge of  $f$ , initial inclination  $\theta$ , and pan/tilt rotation  $\gamma$  and  $\alpha$ , for every pixel position  $(x_t, y_t)$  in the current image we can calculate the position  $(x_{t-1}, y_{t-1})$  of the corresponding pixel in the previous image. (See Appendix B for the proof).

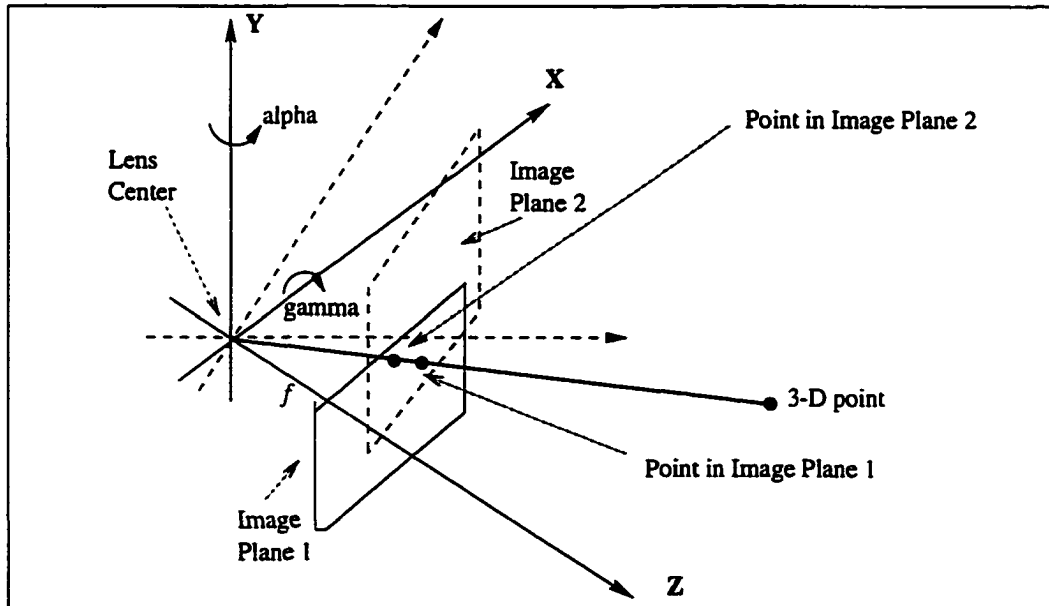


Figure 3.3: 3D point projected on two image planes with the same lens center

### 3.2.2 Moving head detection

The inaccuracies in the input to the compensation algorithm, and a small amount of camera translation, may induce noise and corrupt the compensation method. A single frame difference may not suffice to portray a speaker entirely. It is necessary to recover the speaker's silhouette by observing a number of successive frame differences. Figure 3.4 shows a block diagram describing the moving head detection algorithm, which consists of the following steps:

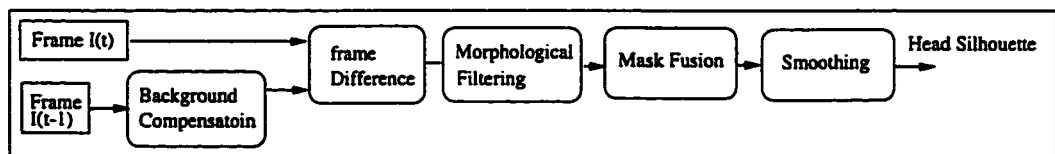


Figure 3.4: Diagram of head detection

- After the background is compensated in the previous frame, the difference of the current frame and the previous frame is calculated, and the absolute value is thresholded.
- If we could achieve exact background compensation, the method described above would be sufficient. In the presence of position inaccuracies, however, the results of these methods deteriorate. Since errors in angle and position readings are inevitably present, it is desirable to develop methods of motion detection that can robustly reject the false motion they cause. By using morphological erosion and dilation we eliminate narrow regions of detected motion, while preserving the original size and shape of the wide regions.

**Morphological filtering:** One application of morphological filters is in noise reduction or suppression<sup>1</sup>. The basic idea is to apply a mask  $M$  over an image  $I$ . The value of an element  $(m_{ij})$  in  $M$  is either 0 or 1.

<sup>1</sup>For a similar application of morphological filtering in motion tracking, see [46]

Two operations in morphological filtering are of interest to us. These are *erosion* and *dilation*. The erosion operation can be considered as a single pass of a thinning operator. The dilation operation, on the other hand, is the reverse of erosion. In the other words, it is an expansion of a set (*e.g.*, an object) into all the background pixel cells which border the set (the object). Thus, erosion shrinks an object whereas dilation enlarges it. Formally, given a mask  $M(n \times n)$  and a part of a binary image  $A$  of the same size as the mask, we can define the erosion operator as follows:

$$A \ominus M = \begin{cases} 1 & \text{if } ((\forall p_{ij} \in A) \wedge (p_{ij} = 1)) \\ 0 & \text{otherwise} \end{cases} \quad (3.7)$$

Dilation is the dual of erosion:

$$A \oplus M = \begin{cases} 1 & \text{if } ((\exists p_{ij} \in A) \wedge (p_{ij} = 1)) \\ 0 & \text{otherwise} \end{cases} \quad (3.8)$$

By applying erosion to the image, narrow regions can be eliminated while wider ones are thinned. In order to restore these wide regions back, dilation is applied using a mask of the same size. After a morphological filtering (*opening* operator = erosion + dilation) is applied, the background noise can be reduced. The kernel size of erosion and dilation operations depends on the noise characteristics caused by compensation error [46], so the filter must be at least as wide as the error. In our system, the kernel of filtering is set as  $9 \times 9$ .

- To generate the head silhouette, a frame motion fusion algorithm is developed in which the multiple frame differences are integrated to generate the continuous motion areas.

#### **Progressive motion fusion:**

The output image of the morphological filtering (*opening*) in Figure 3.4 is called motion mask (MM), denoted by  $m_t(x)$  ( $t$  stands for time and  $x$  for pixel position). The successive MMs are processed with a spatiotemporal filter. The



function of the spatiotemporal filter is to *fuse* a number of consecutive masks. The temporal fusion in the consecutive  $N$  MMs from time  $t_0$  to time  $T = t_0 + N$  is implemented as:

$$\begin{aligned}
 & \text{for } t = t_0, t_0 + 1, \dots, t_0 + N \\
 & \{ \\
 & \quad I_t(x) = ADD\{m_t(x), s_{t-1}(x)\} \\
 & \quad s_t(x) = Median\{I_t(x)\} \\
 & \}
 \end{aligned}$$

where  $m_t(x) \in \{0, 1\}$ , and  $s_{t_0}(x) = 0$ . To reduce noise effects, the combined images are further smoothed by a spatial median filter. The parameter  $N$  controls how long frame information is integrated. For example, keeping  $N$  large will tend to create connected and smooth foreground blobs at the expense of smearing the silhouette, especially if the object motion is excessive. (In our experiment,  $N$  is set to 8 as a tradeoff for the motion mask integration and the contour smearing). Finally, the temporally fused image  $s_T(x)$  is thresholded to reveal the motion regions of the head. The threshold value is set to one so that all the motion area reflected in the fused image is taken into account.

- After obtaining the fused image  $s_T(x)$ , a morphological filter (*closing* operator = dilation + erosion) is applied to make the image more smooth. Then, the edges of motion areas are detected simply by a gradient operator. Although a few connected contours are detected, the head silhouette is the largest connected contour, and it can be extracted by a region growing method.

### 3.3 Facial organ features localization

Analyzing the head image is required in order to make an estimation of the position and the size of the characteristic facial organs, such as eyes, eyebrows, mouth, and nose. Since facial skin, other than facial features and hair, exhibits a similar property in color and luminance, region growing is suitable for extracting the connected skin area, as illustrated in Figure 3.5. The whole head region is denoted as  $H$ . The skin region, the subset of  $H$ , is denoted as  $S$ . The feature region is the complementary set of  $S$ , as denoted by  $\bar{S}$ ,  $H = S \cup \bar{S}$ . Determining growing threshold is not a trivial

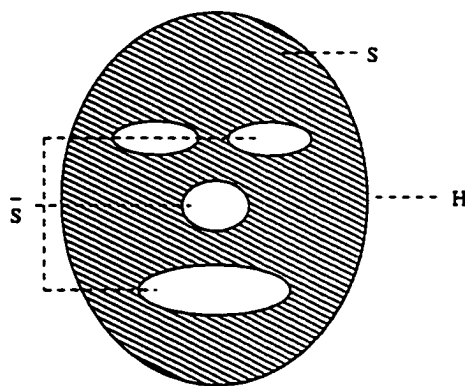


Figure 3.5: Illustration of skin region and feature region

problem because of the small variation of color and luminance in various regions of a face. A large growing threshold produces a large connected skin region, splits the feature region into many pieces, and loses the completeness and shape of the facial features. Moreover, it is difficult to identify and group too many pieces into meaningful facial features. A small growing threshold produces a small skin region and keeps the completeness of facial features. However, sometimes the skin region cannot be completely grown, and when stopped in the middle, its incompleteness also causes the failure of facial organ detection. To overcome the difficulties encountered in the region growing, and make the facial feature detection less sensitive to noise, a two-stage region growing algorithm is developed – global region growing and local region growing. In global region growing, a large growing threshold is selected in

order to explore more skin area. Only the most distinct features, such as the darkest parts of the eyes, the nostrils and in-between the lips, remain. These parts provide important information of the locations of the facial feature, although the information of the shape and the size is lost. The feature region information is estimated in the second stage region growing (*i.e.*, local region growing). In this stage, the region growing is performed in the feature areas individually. The initial seed pixel of the skin is selected in the local area, and a smaller growing threshold can be selected so that a small skin area surrounding the facial organ is detected, and produces as much feature information as possible.

### **3.3.1 Location of feature centers using global region growing**

The head silhouette is obtained in the previous stage, the search of facial features is then limited within this area. In general lighting conditions, the hair and the face exhibit distinct color appearances (except for bald people, in which case segmentation is not necessary). Also, the facial features of iris, eyebrow, in-between lips, and nostril have a distinguishable darkness when compared with the facial skin, which usually exhibits the approximate uniform in color. Hence, the face feature region can be extracted by segmenting the image using region growing based on the color components, *i.e.*, luminance ( $Y$ ) and chrominance ( $U, V$ ). First of all, the image is filtered with a low-pass filter for making the color appear more uniform. Second, the image is partitioned based on the three color properties and the chessboard distance in region growing, the distance  $D$  between seed pixel  $(y_s, u_s, v_s)$  and the growing pixel  $(y_i, u_i, v_i)$  is defined in Formula 3.9. Larger variations are allowed in the luminance component of a region than in the chrominance component, in order to ignore small changes of the luminance in the face region due to shading. Therefore weights  $(1/4, 1, 1)$  are used for YUV. Third, regions smaller than a certain number of pixels are not taken into

consideration, and are removed.

$$D = \max\left(\frac{1}{4}|y_i - y_s|, |u_i - u_s|, |v_i - v_s|\right) \quad (3.9)$$

The diagram of feature region location is plotted in Figure 3.6. The region growing

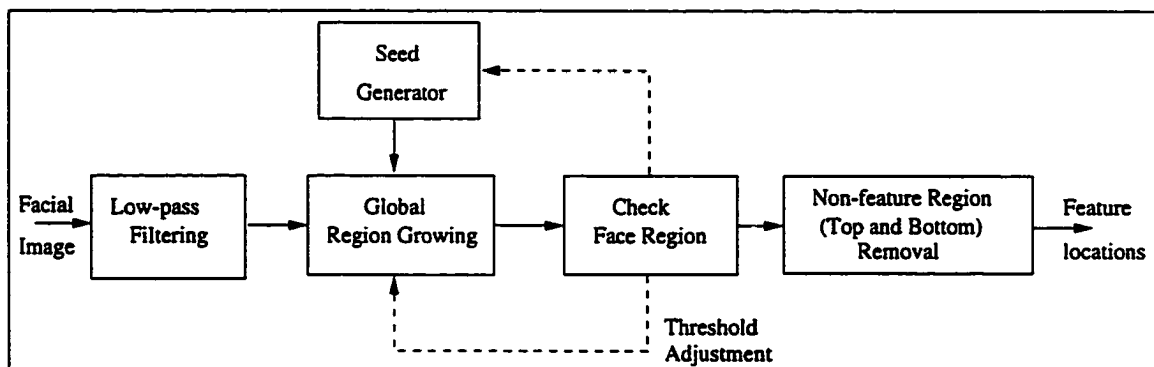


Figure 3.6: Diagram of the feature region location using global region growing

is performed by selecting the central point of the head area as a seed pixel. Assume the initial seed is represented as  $s_0$ , and the skin region grown from  $s_0$  is  $G_0$ . The size of grown region,  $G_0$ , must be checked to make sure that it will be a reasonable face region. The size of  $G_0$ , denoted as  $|G_0|$ , should be restricted to a bounded region, that is

$$T_l \leq |G_0| \leq T_h \quad (3.10)$$

where  $T_l$  and  $T_h$  are predefined constants. Sometimes it is possible that the central point of the facial image falls into a feature region (*e.g.*, nostril). In this case,  $s_0$  will be the feature region instead of the face region, and  $|G_0|$  is small. The size of the grown region depends on the predetermined growing threshold,  $D_T$ ; however, the size of grown region will not go beyond the head silhouette area  $T_h$ ;  $T_l$  is set as the half size of the head silhouette area.

If  $|G_0|$  is beyond the range  $[T_l, T_h]$ , either a new seed should be selected (a new seed  $s_1$  can be selected from the neighboring pixels of  $s_0$ ), or the threshold should be

adjusted, to generate a new region  $G_1$ . This process is iterated until a skin region  $G_i$  is found.  $G_i$  is a subset of the head silhouette area  $H$ , or equal to  $H$ , i.e.,  $G_i \subseteq H$ . Notice that the traditional region growing method only selects one growing seed, and uses the growing seed once; our region growing method differs from the traditional method [31, 10, 11, 60] in that: our region growing procedure is a *dynamic* procedure, which selects the growing seed at least once, the iteration process is performed until the grown region fulfills the pre-defined condition.

After the skin region growing, a number of regions (blobs) are obtained. In order to extract the facial organ blobs (e.g., eyes, mouth, nose, etc.), the top blobs and the bottom blobs (e.g., hair, collar, cloth, etc.) are removed. The feature blobs are then processed for further classification.

### Classification of the regions of eyes, nose and mouth

After obtaining the set of feature blobs, the next task is to classify them into four groups, i.e., left eye group, right eye group, nose group and mouth group. These four classes are distinguishable by the vertical distance and horizontal distance of the blobs. The k-means classifier is an ideal tool to classify them. The diagram of the feature blobs classification and feature center estimation are shown in Figure 3.7. Before describing the algorithm, several terms are explained below first:

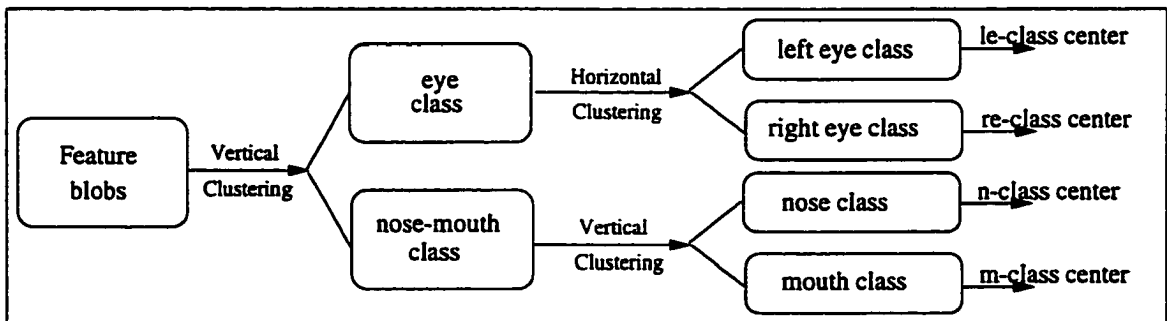


Figure 3.7: Diagram of the feature region classification and feature center estimation

- **Central Point** of a blob is the pixel with an average coordinate of the blob.

For example, the central point  $p$  of blob  $b$  with  $n$  pixels is:

$$x_p = \frac{1}{n} \sum_{i=1}^n x_i \quad (3.11)$$

$$y_p = \frac{1}{n} \sum_{i=1}^n y_i \quad (3.12)$$

- **Blob Distance** is the distance between the central points of two blobs. *Vertical Distance* is the distance of two blobs in vertical  $y$ -coordinate. *Horizontal Distance* is the distance of two blobs in horizontal  $x$ -coordinate.

The algorithm is described as follows:

1. Clustering eye class and nose-mouth class.

Choose the top blob and the bottom blob from the set of feature blobs as the initial eye class and the nose-mouth class respectively. Take the *vertical distance* as a measurement to cluster all the blobs into the two classes by k-means classifier.

2. Clustering left-eye class and right-eye class.

Choose the left-most blob and the right-most blob from the eye class as the initial left-eye class and the right-eye class, respectively. Take the *horizontal distance* as a measurement to cluster all blobs into the two classes by k-means classifier. To verify the correctness of the left-eye and right-eye classification, a confidence measurement is taken into consideration, based on the assumption that eye pair is vertically symmetrical. The vertical symmetry axis, denoted as the *eye vertical*, is the line passing through the center point between the eye regions, and perpendicular to the line passing through both eye region centers, which is denoted as the *eye horizontal*. Let us denote the binary image containing the left-eye and right-eye classes by  $I_e$ . Assume that the extracted

eye regions are labeled *one* and the background is labeled *zero*. For each pair of left and right regions  $(e_l, e_r)$ , a symmetry axis  $a_{lr}$  can be constructed. The confidence value of the symmetry property for this pair is calculated as an inner product between the image  $I_e$  and its mirror reflection with respect to the  $a_{lr}$  axis:

$$C_{symm} = \frac{1}{|U|} \sum_{x, x_R \in U} I_e(x) I_e(x_R) \quad (3.13)$$

where  $U = \{x = (i \ j \ 1)^T | 0 \leq i \leq w_I, 0 \leq j \leq h_I\}$  in which  $w_I$  and  $h_I$  are the width and height of the image  $I_e$ , and  $R$  denotes the reflection operator in the line  $a_{lr}$ . For the correct eye pair regions, this confidence value is expected to be high, ideally 1. The lower the symmetry value is, the less confidence one has.

### 3. Clustering nose class and mouth class.

Choose the top blob and the bottom blob from the nose-mouth class as the initial nose class and mouth class respectively. Take the *vertical distance* as a measurement to cluster all blobs into the two classes by k-means classifier.

Note that the eyebrow is also included in the eye class, their separation will be done later in the shape estimation section. After obtaining the four feature classes, the *class center* for each feature class can be obtained by taking the average of all blob centers of that class. Figure 3.8 shows some examples of the feature estimation.

### 3.3.2 Location of feature regions using local region growing

The local area of each feature (organ) is determined by the estimated location of the feature center and the predefined surrounding size. To grow a skin region in the local area, a seed pixel is selected from the periphery skin area of the feature organ. A smaller growing threshold  $D_{T_S}$  is predefined. The feature blobs with large area are created, based only on the luminance of the image rather than three color components. The feature region is a complementary set of the grown region within a

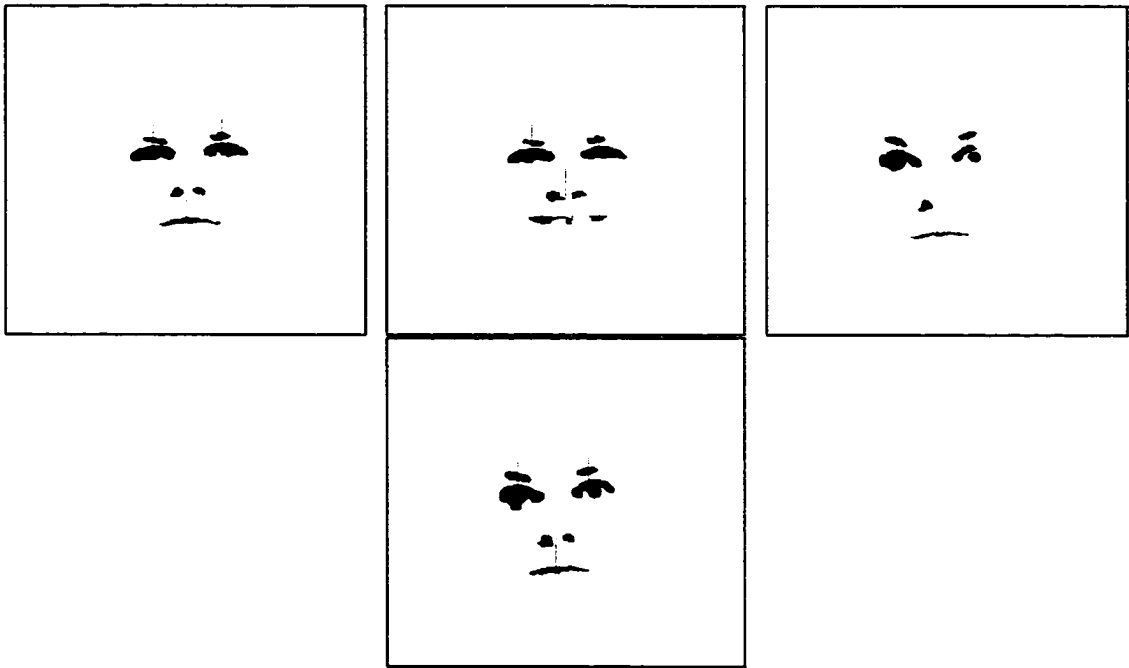


Figure 3.8: Classified features (The regions in the same class are labeled by the same grey scale label; the bottom terminal of each line is the center of each feature).

surrounding window. Finally the region size of each organ (called the *organ window*) can be derived from the extracted feature blobs which are represented by a binary image  $\mathbf{b}$ . The feature area is represented by a high value while the background area is represented by value zero, as shown in Figure 3.9.

In order to determine the size of the detected feature organs, as an example, assume one of the central points of the initial feature windows is at  $(x, y)$ , then the intensity accumulation of the four boundaries of the window that contains  $(x, y)$  can be calculated as

$$M_0 = \sum_{i=x-\frac{w}{2}}^{x+\frac{w}{2}} \mathbf{b}(i, y - \frac{H}{2}), \quad (3.14)$$

$$M_1 = \sum_{i=x-\frac{w}{2}}^{x+\frac{w}{2}} \mathbf{b}(i, y + \frac{H}{2}), \quad (3.15)$$



$$M_2 = \sum_{j=y-\frac{H}{2}}^{y+\frac{H}{2}} \mathbf{b}(x - \frac{W}{2}, j), \quad (3.16)$$

$$M_3 = \sum_{j=y-\frac{H}{2}}^{y+\frac{H}{2}} \mathbf{b}(x + \frac{W}{2}, j) \quad (3.17)$$

where  $W$  and  $H$  are the width and height of the rectangle window, respectively. The rectangle window is shrunk until the condition 3.18 is satisfied:

$$M_0 \cdot M_1 \cdot M_2 \cdot M_3 \neq 0 \quad (3.18)$$

The other feature windows are also shrunk by following the same rule. Note that the nose window and the eye window may be partially overlapped. In that case, the nose window can be adjusted by shrinking in the vertical direction. This adjustment prevents the class boundary from being overlapped each other. Eventually, feature areas with clear boundaries can be obtained, as shown in Figure 3.10.

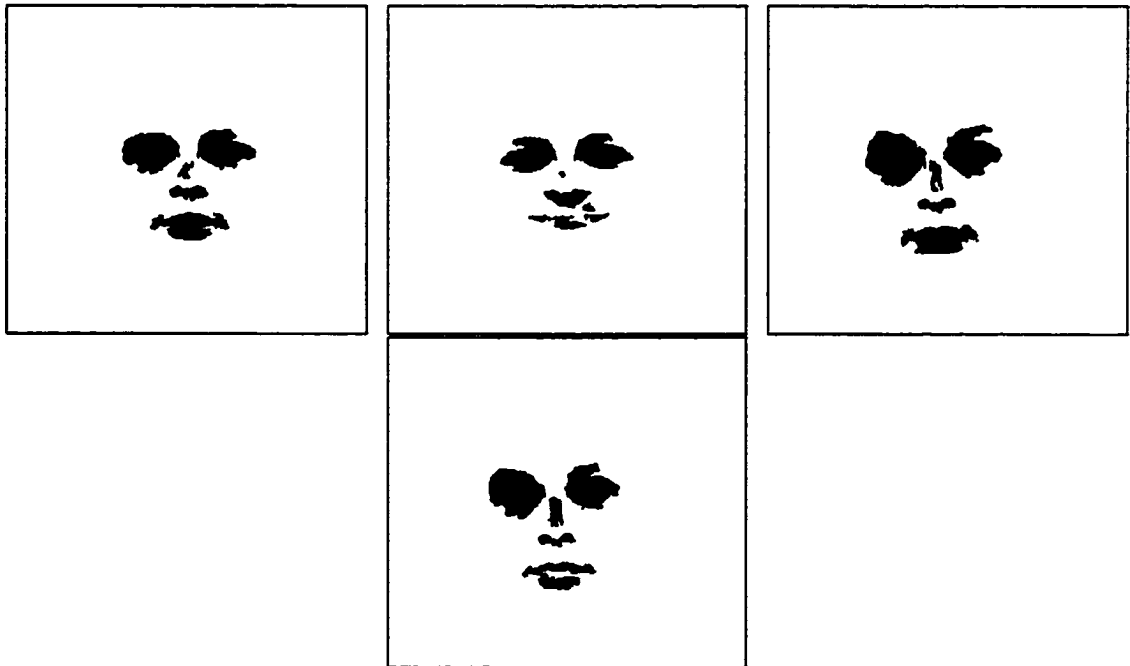


Figure 3.9: Extracted feature blobs.

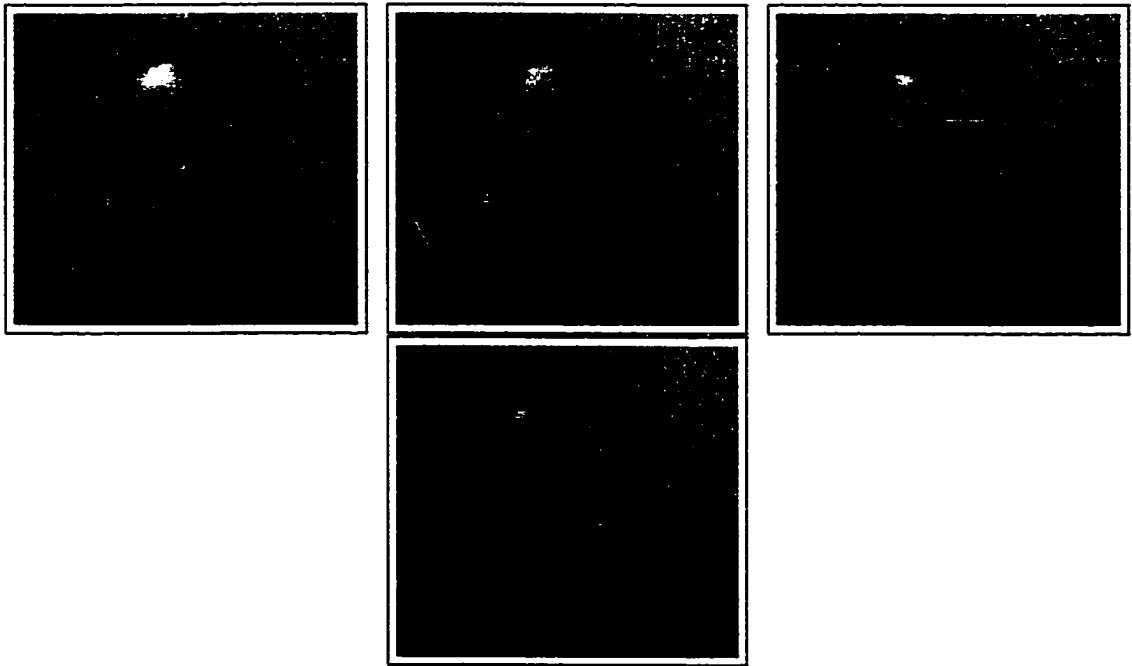


Figure 3.10: The restricted window area based on the extracted feature blobs.

### 3.4 Shape estimation

The above work shows how facial features are restricted in the corresponding windows. The next stage is to extract the real shape for the later model adaptation. The shape of facial features mainly refers to the contour of eye, eyebrow, mouth, nose (nostril and nose side), and chin.

As reviewed in section 3.1, a variety of approaches has been proposed for the shape detection of facial features. These include deformable template matching [15, 19, 86], Hough transforms, and color image processing [9, 77]. Matching deformable templates requires a fairly accurate initial localization of the template because the energy minimization process only finds a local minimum. Other problems are the complexity of the computation, which is caused by using many energy terms and weighting factors during the different epochs of matching. Because of the definition of the energy terms in [86] the template is also inclined to shrink. In this chapter we overcome some of these difficulties by improving the initial localization process. We show that simple

processing on color images, coupled with deformable template matching, can produce the accurate results.

The approaches to detecting eyes and mouth are similar; both use deformable template matching and exploit color information. In addition, eye detection uses Hough transform to search the iris position and size in order to determine the initial location of the eye template. The work of eye detection using saturation information was first presented by S. Bernoegger, L. Yin and A. Basu [6] for the collaborative project of MPEG-4 video coding.

### **3.4.1 Detection and tracking of eye shape - brief overview [6]**

The approach to detecting the eyes is similar to [15] in that it uses Hough transform and deformable template matching, however, it also exploits color information to extract the eyes accurately. The algorithm can be outlined as follows:

- Determine two coarse regions of interest for the eyes.
- Within two coarse regions of eyes, search the iris of the eyes using a gradient based Hough transform.
- Using color information (saturation) get an initial approximation for the eye lids.
- Localize the eye lids using deformable templates.

Since the coarse regions of two eyes have been robustly detected in the previous section, the iris searching range is greatly reduced. The iris is the most significant feature of the eye and has a simple circular shape. It is detected first by using a gradient-based Hough transform for circles [17, 33].

After extracting the circles the deformable templates for the eye lids have to be initialized. The model, along with all the parameters used is shown in Figure 3.11, with the parameters being set as follows:

$$h_1 = 1.5r_{iris} \quad h_2 = 0.5r_{iris} \quad w = 2.2r_{iris} \quad (3.19)$$

where  $r_{iris}$  is the radius of the extracted circle. The orientation  $\alpha$  is determined by the center points of the two circles. The initialized deformable template is also shown in Figure 3.11. In order to avoid the shrink problem, the location of the

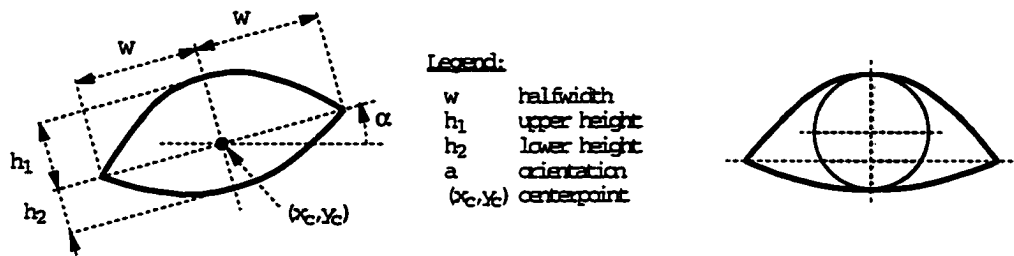


Figure 3.11: Deformable Template (model, initialization).

template and the size of the template are determined separately. Two different types of image information are used to create potential fields, one in each epoch. The image information extracted from a typical eye is shown in Figure 3.12. First, the color



Figure 3.12: Image fields used for computing the potential energy (image, saturation, edge).

information (saturation) is exploited to locate the eye lids by approximating the position of the deformable template relative to the iris. This is done by minimizing

the following energy ( $E_{sat}$ ) which is similar to the valley energy in [86]:

$$E_{sat} = -\frac{1}{|A_w|} \int_{A_w} \Phi_{sat}(\vec{x}) dA \quad (3.20)$$

$A_w$  is the area inside the parabolas but not inside the circle of the iris, and  $\Phi_{sat}(\vec{x})$  is the inverted saturation value of the color image. Since only the location (not the size) is changed this method does not have the shrinking effect.

Next, the deformable template is matched accurately to the eye lids by minimizing the following energy ( $E_{edge}$ ):

$$E_{edge} = -\frac{1}{|B_w|} \int_{B_w} \Phi_{edge}(\vec{x}) ds \quad (3.21)$$

$B_w$  is the boundary of the parabolas and  $\Phi_{edge}(\vec{x})$  is the edge magnitude. During this minimization every parameter of the deformable template (location, orientation, height, width) can be changed <sup>2</sup>.

The tracking of eye features is similar to detection of eye features with the following differences:

- The region of interest, as well as the possible size and therefore the Hough space for the extraction of the iris, can be restricted by using the position and the size of the extracted eye in the previous frame.
- Instead of using the deformable template shown in Figure 3.11, the matched template of the previous frame is used for the initialization of the new template.

Some sample frames with the detected iris and lids shape are shown in Figure 3.13

---

<sup>2</sup>The procedure of numerical solutions for finding the parameters of the deformable template using the energy minimizations can be found in [57] or in the site <http://www.nr.com/>

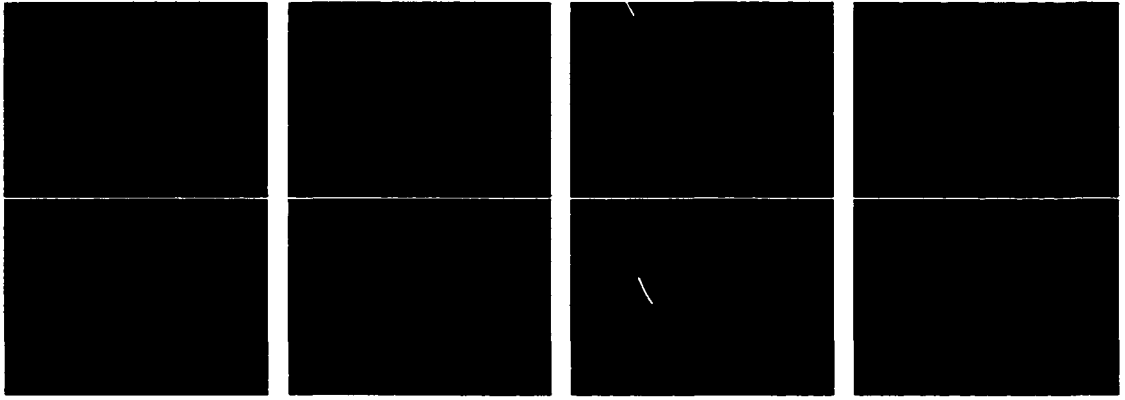


Figure 3.13: Sequence (sample frame 1, 24, 68, 169) showing eye movement. Original images (top); extracted eye template (bottom).

### 3.4.2 Detection and tracking of eyebrow

Notice that in the feature location stage described in the previous section, the eyebrows and eyes sometimes are difficult to separate because of the connection of the regions. Thus in that stage, these two features are not distinguished; instead the eye regions detected contain the eyebrow region. In this section we address eyebrow extraction from the eye region. Because the iris and eye shape have been extracted by the deformable template technique, the eye area can be excluded from the set of eye class, the remaining set is the eyebrow area. The eyebrow has a distinguishable dark luminance from the surrounding skin regardless of race. The integral projection from horizontal and vertical directions can be used respectively for a good shape estimation of the eyebrow.

The integral projection is a very useful technique for the extraction of regional salient features. This technique was successfully used by Kanade [32] and Poggio [8] in their pioneering work on recognition of human faces. Projections can be extremely effective in determining the position of features, provided the window on which they act is suitably located to avoid misleading interferences. Let  $I(x, y)$  be an image patch which contains the object of interest. The vertical integral projection of  $I(x, y)$  in the

$[x_1, x_2] \times [y_1, y_2]$  rectangle is defined as

$$V(x) = \sum_{y=y_1}^{y_2} I(x, y) \quad (3.22)$$

The horizontal integral projection is similarly defined as

$$H(y) = \sum_{x=x_1}^{x_2} I(x, y) \quad (3.23)$$

Since we have detected the eye shape, the approximate eyebrow region can be determined using anthropometric measure (geometric relation of the eye and eyebrow). In the delimited window of the eyebrow, the eyebrow width is computed by using the vertical integral projection of the intensity and finding the two most significant opposite gradients in the projection, the left-most and right-most of the eyebrow can be detected.

In order to estimate the shape of the eyebrow, the horizontal integral projection is applied to each column from the left-most eyebrow to the right-most eyebrow. The projection in each column produces two most significant opposite gradients, whose positions are detected for estimating the height of the eyebrow at that column. The estimated column height and the position form the entire shape of the eyebrow. The example of the eyebrow detection is shown in Figure 3.14, 3.15 and 3.16.

In order to track the eyebrow effectively, the estimated shape of the previous



Figure 3.14: Example of the horizontal integral projection for eyebrow

frame can be used for setting the initial delimited window in the current frame. The

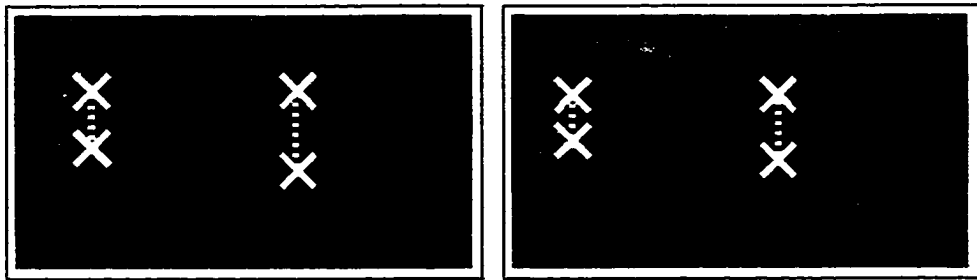


Figure 3.15: Example of the vertical integral projection for eyebrow

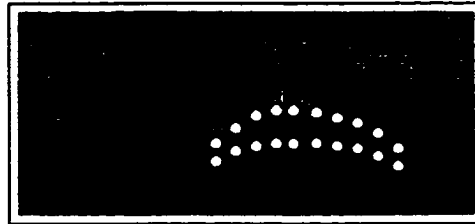


Figure 3.16: Example of the eyebrow shape extraction using the consecutive integral projection

eyebrow width estimated in the previous frame can be also used as the width verification for the current frame.

### 3.4.3 Detection and tracking of mouth shape

The algorithm for mouth detection can be outlined as follows:

- Determine a coarse region of interest for the mouth, and determine the mouth status: *closed* or *open*.
- Using color information (*hue* component) get an initial approximation for the lip. This is done by minimizing the energy cost function.
- Localize the mouth contour by minimizing the edge cost function.

#### (1) Deformable template for closed mouth and open mouth

The features of the mouth are described by an open-mouth template and a closed-mouth template, separately, which are composed of four or three parabolas  $P_i$ , respectively, as shown in Figure 3.17 and Figure 3.18. The mouth features are represented



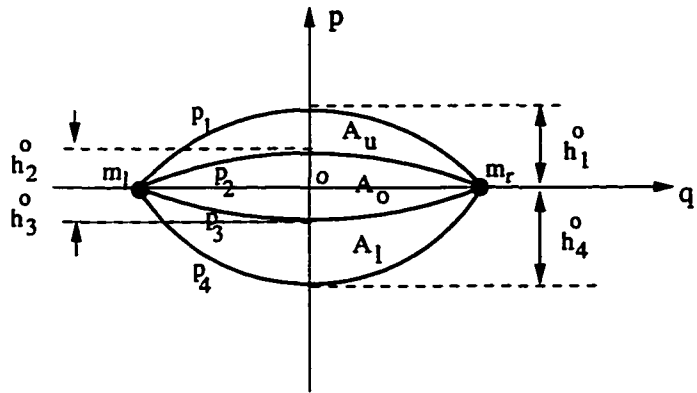


Figure 3.17: Deformable template for mouth-open

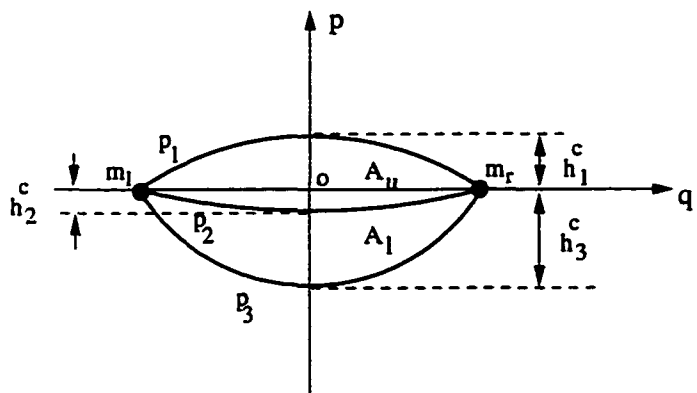


Figure 3.18: Deformable template for mouth-closed

by the mouth corner points and the thickness of the lips. The open-mouth template consists of four parabolic curves which are defined as follows:

$$p_1 = h_1^o \left(1 - \left(\frac{q}{L_M}\right)^2\right) \quad (3.24)$$

$$p_2 = h_2^o \left(1 - \left(\frac{q}{L_M}\right)^2\right) \quad (3.25)$$

$$p_3 = h_3^o \left(1 - \left(\frac{q}{L_M}\right)^2\right) \quad (3.26)$$

$$p_4 = h_4^o \left(1 - \left(\frac{q}{L_M}\right)^2\right) \quad (3.27)$$

where the parameters  $h_i^o$ ,  $i=1,2,..4$ , described opening height of the upper lip and lower lip. The width of the mouth  $L_M$  is calculated as the distance between the two corners of the mouth. The thickness of the lips are calculated as  $|h_1^o - h_2^o|$  and  $|h_4^o - h_3^o|$ . To represent an open-mouth features, six parameters are needed:  $h_i^o$ ,  $i=1,2,..4$  and the two corners of the mouth  $m_l$  and  $m_r$ .

The closed-mouth template consists of three parabolic curves which are defined as

$$p_1 = h_1^c \left(1 - \left(\frac{q}{L_M}\right)^2\right) \quad (3.28)$$

$$p_2 = h_2^c \left(1 - \left(\frac{q}{L_M}\right)^2\right) \quad (3.29)$$

$$p_3 = h_3^c \left(1 - \left(\frac{q}{L_M}\right)^2\right) \quad (3.30)$$

To represent a closed-mouth features, five parameters are needed:  $h_i^c$  ( $i=1,2,3$ ) and the two corners of the mouth  $m_l$  and  $m_r$ .

## (2) Determination of mouth corners

In the previous section, the region center of the mouth has been estimated. Using the vertical integral projection of the hue component of the mouth region, two vertical

lines at the left-most and the right-most of the mouth can be determined by finding the two most significant opposite gradients in the projection (Figure 3.19).

Suppose that the orientation of the mouth (*i.e.*, a line in-between lips) follows the orientation of the eye (*i.e.*, a line connecting the centers of the left eye and the right eye). At both sides, the horizontal integral projection, along the orientation of the mouth, is performed to find the local peak for determining the location of the mouth corner (Figure 3.20). The example of the estimation of mouth corners is shown in Figure 3.21.

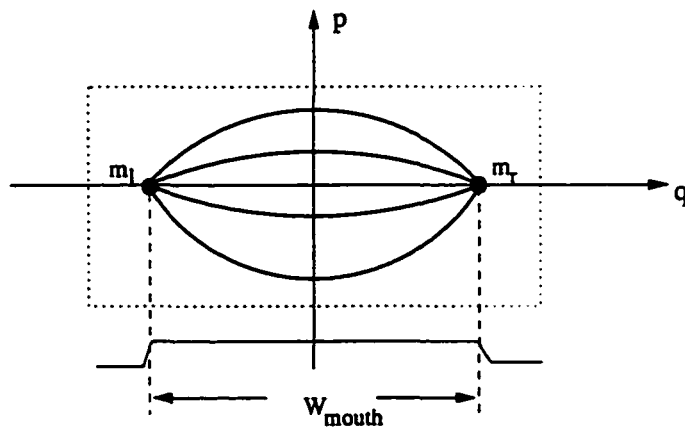


Figure 3.19: Vertical projection for determining the width of a mouth

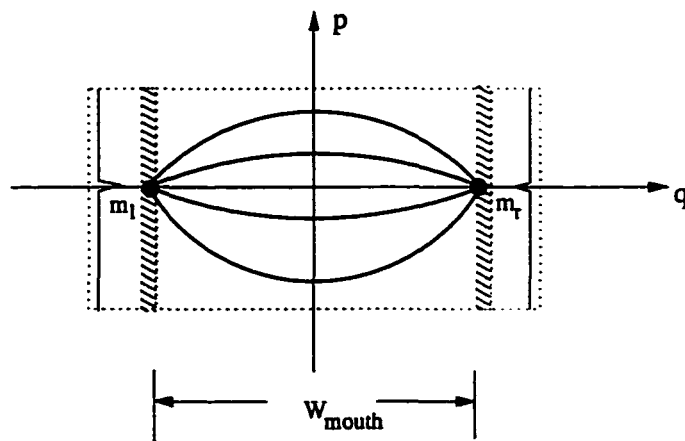


Figure 3.20: Horizontal projection for determining the corners of a mouth

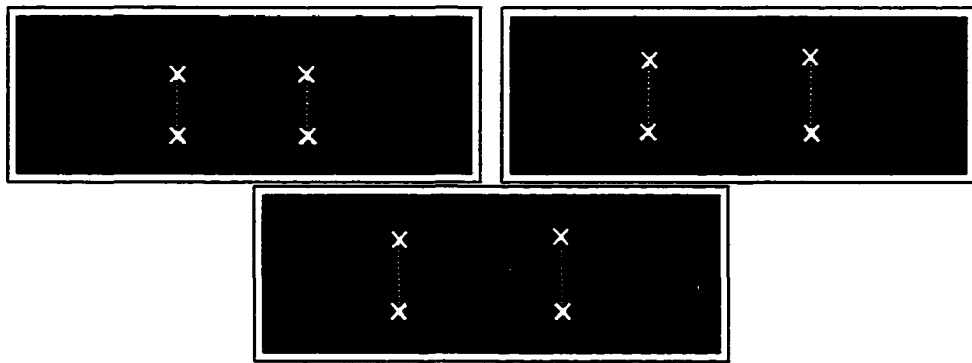


Figure 3.21: Example of the mouth corner estimation

### (3) Determination of closed mouth and open mouth

Because there are two deformable templates, an open mouth template and a closed mouth template, the situation of whether the mouth is open or closed has to be determined first. When considering the situation in color space, the lips present a distinctive color property from the other areas of the mouth. In a vertical strip crossing the center of the mouth, the horizontal integral projection of the *Hue* component appears two peaks in the projection map. The width of the valley of the horizontal projection is a rule for discriminating whether the mouth is open or closed. If the width of the valley is beyond two pixels, the mouth is assumed to be open, otherwise, the mouth is assumed to be closed (see Figure 3.22 ~ 3.24). The width of the valley

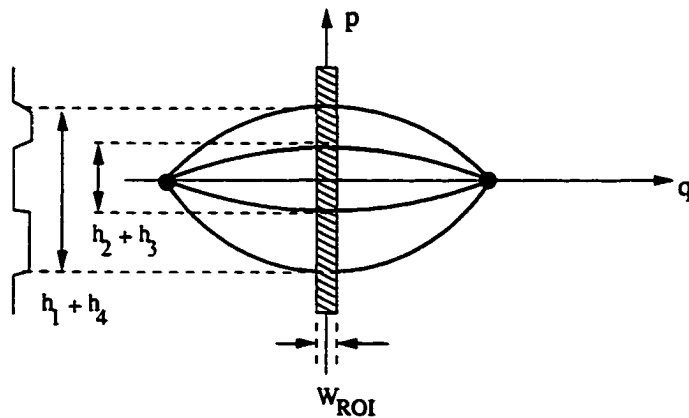


Figure 3.22: Horizontal projection for determining the mouth-closed and mouth-open is estimated to be an initial parameter for the open mouth template (*i.e.*,  $h_2^o + h_3^o$ ).

The distance between the two most significant opposite gradients of the outside valley is estimated as the initial parameter of the lip distance (i.e.,  $h_1^o + h_4^o$ ).

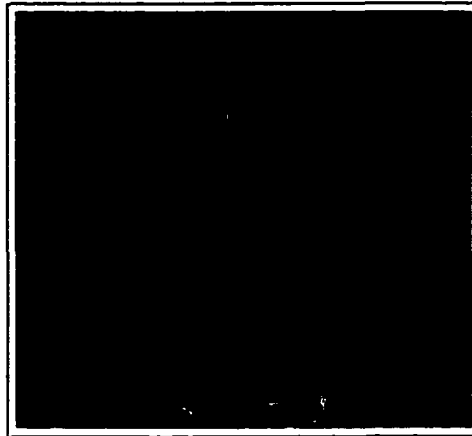


Figure 3.23: Example of the integral projection of the hue component for a mouth

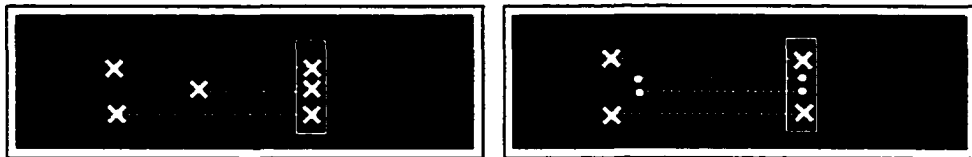


Figure 3.24: Example of the horizontal projection of Hue for mouth-closed/open

#### (4) Deformable template matching

After it has been determined whether the mouth is open or closed, deformable template matching can be performed to select the initial parameters from the valley position. For instance, the initial parameter for the height of the outside upper lip  $h_1^o$  is calculated as the distance from the upper edge of the outside valley to the mouth center, while the one for the inside upper lip  $h_2^o$  is the distance from the upper edge of the inside valley to the mouth center. The similar situation is done for the estimation of the initial parameters  $h_3^o$  and  $h_4^o$ .

In the case of mouth closed, the initial parameter of  $h_2^o$  is estimated as the distance from the lower edge of the valley to the mouth center.

For the template matching in two different situations (*i.e.*, mouth open and mouth closed), different cost functions are proposed to deal with that individually.

• ***mouth-open cost function***

The first cost function for an open mouth  $E_1^o$  is based on the color information of the lips, which is defined as follows:

$$E_1^o = k_1 E_{hue}^o + k_2 E_{mean}^o + k_3 E_{var}^o \quad (3.31)$$

with  $E_{hue}^o$  being

$$E_{hue}^o = \frac{1}{|A_u|} \int_{A_u} \Phi_{hue}(\vec{x}) dA + \frac{1}{|A_l|} \int_{A_l} \Phi_{hue}(\vec{x}) dA \quad (3.32)$$

$A_u$  and  $A_l$  are the areas inside the parabolas of the upper lip and the lower lip respectively;  $\Phi_{hue}(\vec{x})$  is the value of the hue component of the mouth image. Since the lips appear more “red” ingredient in the common lighting condition, the hue components on the lip regions are usually in small values.

$E_{mean}^o$  being

$$E_{mean}^o = -|m_{A_u} - m_{A_o}| - |m_{A_l} - m_{A_o}| + \sigma_{A_u} + \sigma_{A_o} + \sigma_{A_l} \quad (3.33)$$

and  $E_{var}^o$  being

$$E_{var}^o = |\sigma_{A_u} - \sigma_{A_o}| + |\sigma_{A_o} - \sigma_{A_l}| + |\sigma_{A_l} - \sigma_{A_u}| \quad (3.34)$$

where  $m_{A_u}, m_{A_l}, m_{A_o}$  and  $\sigma_{A_u}, \sigma_{A_o}, \sigma_{A_l}$  are the means and the variances of the hue component of the image in the regions of  $A_u$  (upper lip),  $A_l$  (lower lip), and  $A_o$  (in-between the lips) respectively (Figure 3.17). Since the teeth and the rest between the lips have very different values in the  $Y$  component but not in the hue component, the hue component is used here for evaluation rather than the  $Y$  component. Coefficients  $k_i, (i = 1, 2, 3)$  are the weighting factors which were set to 1 in the experiments, so

that the three energy terms are treated equally on their contribution to the calculation of the cost function. The second term  $E_{mean}^o$  considers that the regions  $A_u$ ,  $A_l$  and  $A_o$  have different hue values but inside each region there are almost no differences of the hue value. The third term  $E_{var}^o$  considers that the variance of the camera noise is the same in all regions. The deformable template matching in this step is done by minimizing the energy  $E_1^o$ , the combination of  $(h_1^o, h_2^o, h_3^o, h_4^o)$  with the minimum value of  $E_1^o$  is selected as the approximate lip outline parameters of the open mouth.

In order to match the deformable template accurately to the mouth shape, the four curves are further tuned by minimizing the following edge energy:

$$E_2^o = - \sum_{i=1}^4 \frac{1}{|B_{p_i}|} \int_{B_{p_i}} \Phi_{edge}(\vec{x}) ds \quad (3.35)$$

$B_{p_i}$  ( $i=1,2,3,4$ ) are the boundaries of the parabolas and  $\Phi_{edge}(\vec{x})$  is the edge magnitude.

- ***mouth-closed cost function***

The cost function for a closed mouth is defined as:

$$E_1^c = k_1 E_{hue}^c + k_2 E_{mean}^c + k_3 E_{var}^c \quad (3.36)$$

with  $E_{hue}^c$  being

$$E_{hue}^c = \frac{1}{|A_u|} \int_{A_u} \Phi_{hue}(\vec{x}) dA + \frac{1}{|A_l|} \int_{A_l} \Phi_{hue}(\vec{x}) dA \quad (3.37)$$

$A_u$  and  $A_l$  are the areas inside the parabolas of upper lip and lower lip respectively;  $\Phi_{hue}(\vec{x})$  is the value of hue component of the mouth image.

$E_{mean}^c$  being

$$E_{mean}^c = |m_{A_u} - m_{A_l}| + \sigma_{A_u} + \sigma_{A_l} \quad (3.38)$$

and  $E_{var}^c$  being

$$E_{var}^c = |\sigma_{A_l} - \sigma_{A_u}| \quad (3.39)$$

Coefficients  $k_i$  ( $i=1,2,3$ ) are the weighting factors which are set to 1 in the experiments.

After minimizing the region energy  $E_1^c$  for the closed mouth, the parameters of the template are further tuned by minimizing the edge energy, which is similar to the case of the open mouth, *i.e.*,

$$E_2^c = - \sum_{i=1}^3 \frac{1}{|B_{p_i}|} \int_{B_{p_i}} \Phi_{edge}(\vec{x}) ds \quad (3.40)$$

$B_{p_i}$  ( $i=1,2,3$ ) are the boundaries of the parabolas and  $\Phi_{edge}(\vec{x})$  is the edge magnitude.

• ***Confidence measurement for the estimated mouth parameters***

After the lip outline parameters have been estimated, they have to be verified, different criteria are explored for an open or a closed mouth.

**(a) The confidence measurement for open mouth**

For an open mouth, the estimated lip parameters must satisfy the following criteria, *criterion 1*:

$$|h_1^o - h_2^o| \leq |h_3^o - h_4^o| \quad (3.41)$$

*criterion 2*:

$$m_{A_o} < m_{A_u}, m_{A_o} < m_{A_l} \quad (3.42)$$

where  $m_{A_o}$ ,  $m_{A_u}$  and  $m_{A_l}$  are the mean values of the inverse-Hue component in the regions  $A_o$ ,  $A_u$  and  $A_l$  respectively.

Criterion 1 denotes that the upper lip can not be thicker than the lower lip, while



criterion 2 stands for the fact that the “red” component within lip is more “intense” than within the region between the lips.

**(b) The confidence measurement for closed mouth**

For a closed mouth, the estimated lip parameters must satisfy the following criteria, *criterion 1*:

$$|h_1^c - h_2^c| \leq |h_3^c - h_2^c| \tag{3.43}$$

*criterion 2*:

$$|m_{A_u} - m_{A_l}| \leq T \tag{3.44}$$

$T$  is set to 0.5 in the experiment. Criterion 1 denotes that the lower lip is usually thicker than the upper lip, while criterion 2 expresses that the component (*red*) in upper lip and lower lip is similar enough when the mouth is closed. Note that criterion 2 might not be suitable to the case of open mouth because of the reflection variation of the both lips when the mouth is open. If the estimation values do not pass the verification, they are rejected; instead, the matched template in the previous frame is used in the current frame.

To track the mouth motion effectively, the height parameters of the mouth in the previous frame can be used as the initial parameters for the current frame. Some sample results are shown in Figure 3.25.

**3.4.4 Detection and tracking of nose shape**

The important nose features lie in the shapes of nostril and nose side. The accurate shape in the nostril and the nose side can be an important sign composing a facial expression, especially when a person is smiling or laughing, the nostril shape and the nose side are obviously changed.

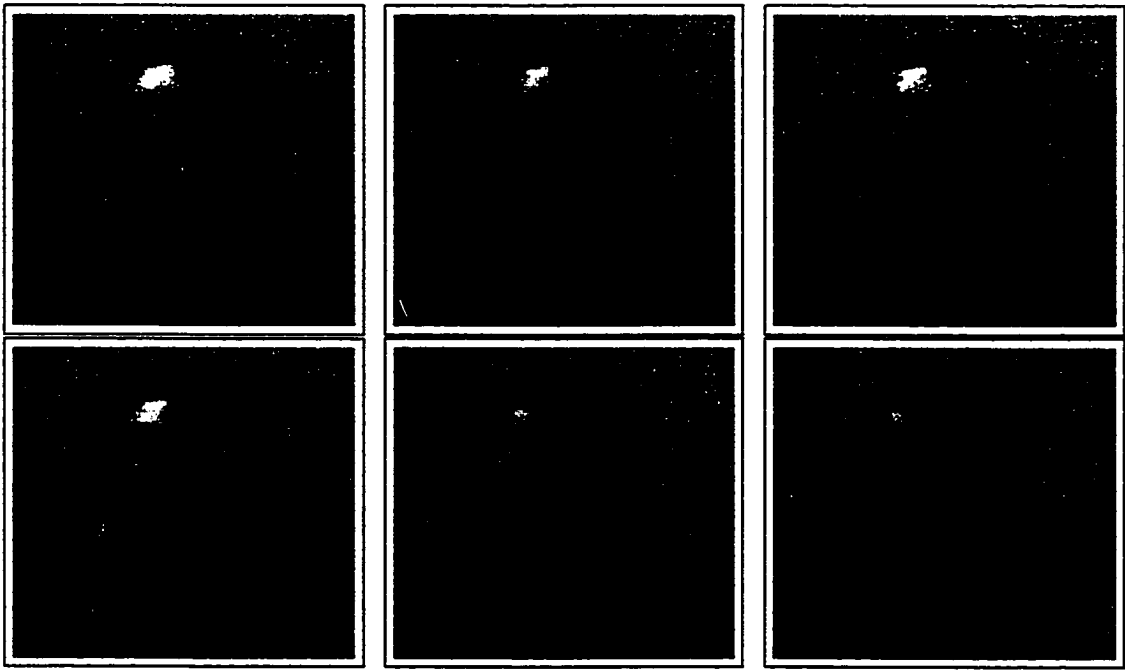


Figure 3.25: Sample frames of the deformable template matching on open/closed mouths

### Nostril estimation

The nostril has a distinctive darkness from the facial skin. As mentioned in the previous section, color-based region growing can roughly detect the position and the approximate shape of the nostrils. In order to detect the nostril shape correctly, a geometric template is further applied on the nostril region, which is a twisted pair curve with a leaf-like shape, as shown in Figure 3.26. The nostril template is defined as a part of a twisted pair curve, which is represented in a polar coordinate system,

$$\rho^2 = a^2 \cos(s\theta) \quad (3.45)$$

The nostril in the right side and the left side of a nose can be represented by the up-right curve and the up-left curve, respectively. Parameter  $a$  is the width of a nostril. Parameter  $s$  controls the shape of the curve, which is a float value in the range of  $[1,10]$ . The smaller the  $s$  value has, the thicker the leaf-shape appears. Besides the parameters  $a$  and  $s$ , the orientation of the  $x$  axis and the position of the origin  $o$  can

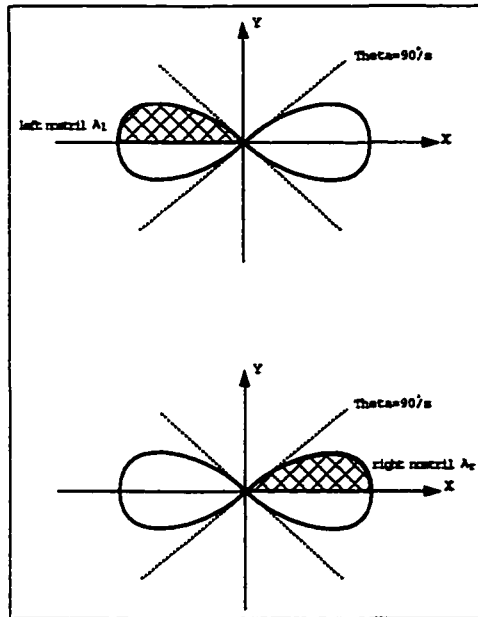


Figure 3.26: Template of nostrils

be adjustable as well. The relationship between  $(x, y)$  in Cartesian coordinate system and  $(\rho, \theta)$  in polar coordinate system is simply defined as,

$$x = \rho \cos(\theta) \quad (3.46)$$

$$y = \rho \sin(\theta) \quad (3.47)$$

From the detection of the nose regions in the previous section using color-based region growing, the initial width of the nostril can be determined by calculating the distance between the left-most pixel and the right-most pixel in the nostril region. The initial orientation of the nostril can be determined by the eigenvector with the largest eigenvalue of the nostril pixel region. The initial shape control parameter  $s$  is set as the value 2. Note that the two nostrils are assumed to be symmetric with respect to the center line of the face, the missed nostril in the previous stage can be compensated using the flipping technique.

After the initial parameters are estimated, the deformable template matching can

be performed, the cost function for energy minimization is defined based on the assumption that (1) the nostrils have distinctive darkness comparing the skin, and (2) the luminance gradient has a high value at the border of a nostril. The cost function for the right nostril is defined as,

$$E^{nl} = k_1 E_{lumin}^{nl} + k_2 E_{grad}^{nl} \quad (3.48)$$

with  $E_{lumin}^{nl}$  being

$$E_{lumin}^{nl} = \frac{1}{|A_r|} \int_{A_r} \Phi_{lumin}(\vec{x}) dA \quad (3.49)$$

with  $E_{grad}^{nl}$  being

$$E_{grad}^{nl} = -\frac{1}{|B_r|} \int_{B_r} \Phi_{grad}(\vec{x}) ds \quad (3.50)$$

$A_r$  is the area inside the up-half of the right leaf-shape curve.  $\Phi_{lumin}(\vec{x})$  is the value of luminance component of the nostril region.  $\Phi_{grad}(\vec{x})$  is the edge magnitude.

The cost function for the left nostril has the same definition as the one for the right nostril, except the left leaf-shape curve is used instead.

During this minimization every parameter of the deformable template (location, orientation, width, shape) can be changed.

### Nose side estimation

The nose side has an apparent shape, like a vertical parabola. (Assume that the nose orientation has been normalized into the straight direction which is parallel to the face center line). A pair of templates of the nose side (both the right side and the left side) are defined in Figure 3.27,

$$x_l = (h_{ns}(1 - (\frac{y}{|n_{s2} - n_{s1}|})^2) \quad (3.51)$$

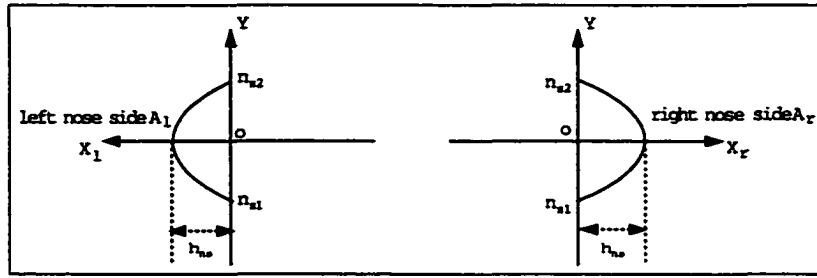


Figure 3.27: Template of nose side

$$x_r = (h_{ns}(1 - (\frac{y}{|n_{s2} - n_{s1}|})^2)) \quad (3.52)$$

The initial corner of the nose side  $n_{s1}$  has been fixed since the nostril has been estimated. The parameter  $n_{s2}$  and the width  $h_{ns}$  need to be estimated by minimizing the energy, as defined below – the cost function for the right side of a nose:

$$E^{ns} = k_1 E_{lumin}^{ns} + k_2 E_{grad}^{ns} \quad (3.53)$$

with  $E_{lumin}^{ns}$  being

$$E_{lumin}^{ns} = -\frac{1}{|A_r|} \int_{A_r} \Phi_{lumin}(\vec{x}) dA \quad (3.54)$$

with  $E_{grad}^{ns}$  being

$$E_{grad}^{ns} = -\frac{1}{|B_r|} \int_{B_r} \Phi_{grad}(\vec{x}) ds \quad (3.55)$$

$A_r$  is the area inside the parabola, which has bright luminance value  $\Phi_{lumin}(\vec{x})$  in the nose wing.  $\Phi_{edge}(\vec{x})$  is the edge magnitude.

The cost function for the left side of a nose has the same definition as the one for the right side. In the course of nose tracking, the obtained parameters of the previous frame can be also used as the initial parameters of the nose template in the current frame. Some sample frames with the detected nostril and nose sides are shown in Figure 3.28.



Figure 3.28: Sample frames of the deformable template matching on nostril and nose sides

### 3.4.5 Detection and tracking of chin contours

For an automatic analysis of the person's facial expressions, a more accurate adaptation of the face model to the chin contours of the individual person is necessary. Therefore, the chin contours in the image sequence have to be estimated.

The geometric model of the chin region simply consists of two parabolas (see Figure 3.29). The energy function is relatively simple, as shown,

$$E^{chin} = k_1 E_{symm}^{chin} + k_2 E_{grad}^{chin} \quad (3.56)$$

First term is based on the assumption that the chin has a symmetric structure with respect to face vertical center,

$$E_{symm}^{chin} = [L_c - R_c]^2 \quad (3.57)$$

The second term  $E_{grad}^{chin}$  is the gradient energy in the chin boundary,

$$E_{grad}^{chin} = -\frac{1}{|B_r|} \int_{B_r} \Phi_{grad}(\vec{x}) ds - \frac{1}{|B_l|} \int_{B_l} \Phi_{grad}(\vec{x}) ds \quad (3.58)$$

The initial endpoints  $b_l$  and  $b_r$  can be determined by the intersection of the facial silhouette and the orientation of mouth horizontal, both of which have been estimated in the previous section. The chin bottom point  $a$  must occur at a certain position between  $a_1$  and  $a_2$  with respect to the origin mouth center, based on the anatomy of

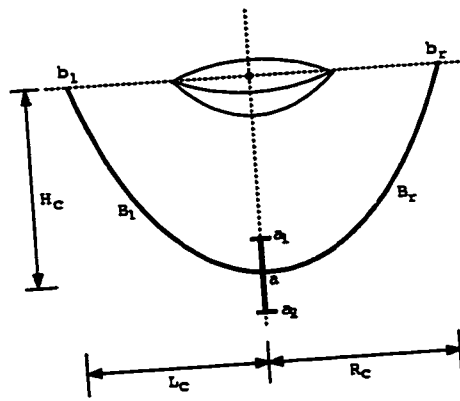


Figure 3.29: Deformable template of a chin

an average face empirically, *i.e.*,

$$a_1 = |center_{nose} - center_{mouth}|/2 + center_{mouth} \quad (3.59)$$

$$a_2 = |center_{nose} - center_{mouth}| \times 2 + center_{mouth} \quad (3.60)$$

During the energy minimization, only the positions  $b_l$ ,  $b_r$  and  $a$  are adjustable. Some examples to detect the chin contours are shown in Figure 3.30.

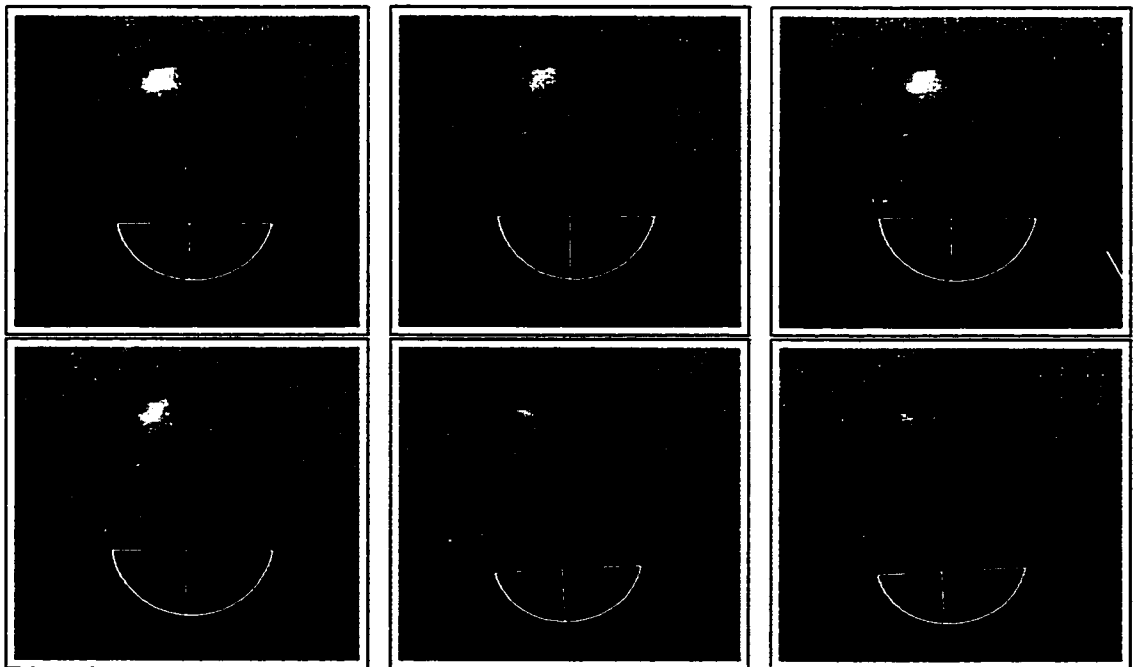


Figure 3.30: Sample frames of the deformable template matching on chin contours

## 3.5 Experimental results

As shown before, the deformable template incorporating with color information and integral projection detected the facial feature correctly. Facial features are estimated with high accuracy subjectively. To evaluate the algorithms developed, more experiments with various color images of different facial video sequences were made.

Similar to most existing work on face feature detection [58, 7], some moderate assumptions are imposed in our work, such as:

- The input images are a sequence of images showing the head and shoulders of a person talking in front of a camera. The image data contains color information.
- The motion of the speaker is moderate with respect to the frame rate (and image size).
- Certain facial features are always visible. Thus, head rotations that impede the visibility of eyes and mouth are excluded. Similarly, the face may not be occluded by other objects such as gesticulating hands.
- The head inclination (rotation around the three main axes  $x, y, z$ ) is less than 45 degrees.
- The face has no facial hair (like a beard or a moustache), glasses, etc.

We use a camera mounted on an active platform (pan/tilt) to take an active video sequence, which shows a talking person with an unconstrained background. The camera rotation is less than  $\pm 5^\circ$ . Figure 3.31 and Figure 3.32 show the results of the active head detection and the facial feature detection in the sequence “Guan”.

The active tracking with color-based deformable template method produced accurate results in shoulder-head silhouette detection and the facial feature shapes estimation.



The two-step region growing limits the feature search area correctly. In the eye tracking, the Hough transform coupled with the saturation-based deformable template is the first important step to extract the eye shape. The hue information is explored for lip shape extraction. The closed-mouth and open-mouth detection gives us sufficient accuracy to use the corresponding template to match into the mouth.

The extraction of the chin, nostril and nose side has been attempted as well in this section. Note that our deformable template matching algorithm only uses several single steps to fit the template to the facial features, it avoids the computation complexity for the large amount of search of the parameters, and the template shrink problem.

The feature tracking algorithm is also tested on other video sequences (such as *Dima*, *Stan*, *Dana*, *Xun*) with pan and tilt movement of the active platform, and with still camera platform (such as *Claire*, *Alau*). The performance for detecting the feature objects is shown in Table 3.1. The sample frames of the eight video sequences are shown in Appendix C. The features of objects are said to be detected correctly if the feature points of the corresponding region falls close enough to the manually determined points in each frame (*i.e.*, the distance between the detected point and the manually picked point should be less than 4 pixels). Table 3.1 indicates for each sequence: the number of frames tested, the absolute mean error in the feature regions in the regions of eye, eyebrow, nose-side, nostril, mouth and chin. The points picked for the performance evaluation are distributed on each feature shape, for example,

- eye: two corners of the eye, a point in the up-most of the eye-lid, and a point in the lower-most of the eye-lid are selected;
- mouth: two corners of the mouth, four points along the central vertical line across the two lips are selected;



Figure 3.31: *1st row*: Original active video sequence (frame 10, 24, 41); *2nd row*: Compensated frame differences; *3rd row*: Noise remove using morphological filtering ( $kernel = 9 \times 9$ ); *4th row*: Consecutive frames fusion; *5th row*: Motion areas detected by smoothed fusion frames (*closing*); *6th row*: Contours of motion areas; *7th row*: Detected silhouette of the motion head.

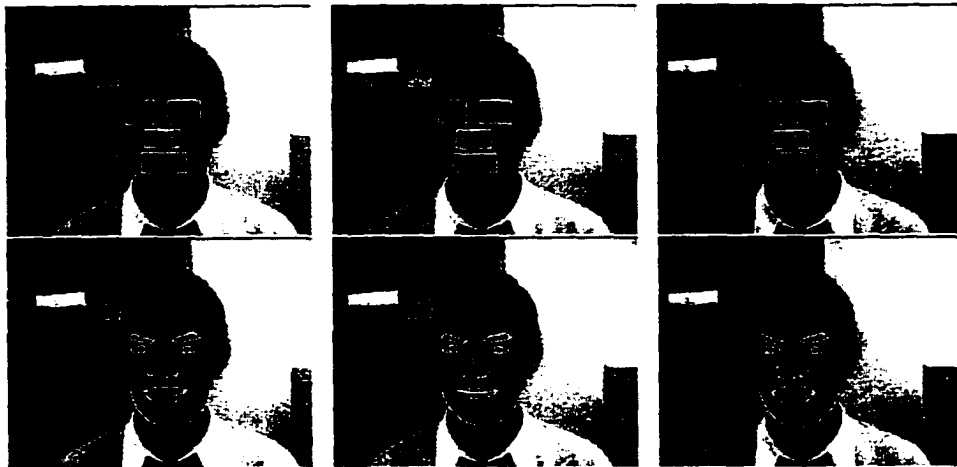


Figure 3.32: Top: coarse regions detected by color-based two-step region growing; Bottom: detected facial features (iris, eye, eyebrow, nostril, nose sides, mouth, and chin) on frame 10, 24, 41 of video sequence "Guan"

- eyebrow: two corners of the eyebrow, and the up and lower points across the central eyebrow;
- nostril: a center point, and the left-most and the right-most points;
- nose-side: two end-points of the nose wing and a middle point of the wing;
- chin: two end-points on the chin which are corresponding to the two corners of the mouth, and a pin-point of chin bottom.

The overall performance (measured over all tested frames) gives an indication of the capability of the system to detect most feature objects correctly (only 18 frames out of 270 frames have an indication that the position errors in mouth, nose-side, eyebrow and chin regions are beyond 4 pixels). The reasons for the failure are: (1) initial template localization is not accurate enough if the head rotation is large (such as "Xun" and "Dima"); (2) the hue and saturation signals are not strong enough in certain imaging condition (such as "Stan", "Dima" and "Dana"); (3) the size of feature area appears small if the image resolution is low, and cannot provide enough feature information for the template matching (such as "Claire"). Notice that the

Sequence title	Number of tested frames	Absolute mean deviation (in pixels)					
		eye(l/r)	eyebrow(l/r)	noside(l/r)	nostril(l/r)	mouth	chin
Mario	60	1.9/2.0	2.1/1.9	1.8/2.0	2.1/1.9	2.8	3.0
Guan	30	2.1/1.8	1.8/1.8	1.7/1.9	1.6/2.0	3.0	3.4
Dima	30	3.4/3.8	1.8/1.4	2.1/2.0	1.5/1.9	2.9	3.8
Stan	30	3.3/3.5	2.5/2.2	2.0/1.7	1.9/2.2	4.0	5.1
Dana	30	2.5/3.0	2.3/2.5	2.3/2.7	2.0/1.8	3.7	3.5
Xun	30	3.2/3.6	3.1/2.9	3.0/2.8	3.0/2.4	2.5	3.2
Claire	30	1.2/1.3	2.4/2.7	2.8/2.3	1.4/1.4	2.9	2.7
Alau	30	1.5/1.4	1.3/2.1	1.4/1.9	1.8/1.6	2.3	2.3
<b>Total</b>	<b>270</b>	<b>2.4</b>	<b>2.1</b>	<b>2.1</b>	<b>1.9</b>	<b>2.9</b>	<b>3.3</b>

Table 3.1: Performance evaluation of the detection of facial features in eight video sequences. (“l/r” stands for “left/right”). Sample frames of these video sequences are shown in Appendix C.

resolution of the video sequences we used in our experiment is higher than 480\*480 pixels, only one exception for video “Claire”, which has small size of 288\*360 pixels.

### 3.6 Discussion

In this chapter, we proposed a new algorithm for tracking a face observed with an active camera for an arbitrary background. We also presented a new way to limit the region and the initial parameter setting for feature detection, as a remedy for the shrinking effect of the deformable template. Experimental results show that this approach is feasible in practical applications.

The concept of deformable templates does not encompass a fixed pattern to find the shape of an object region (as in pattern-based methods [71]), and, consequently, the method is, to some extent, invariant under changing lighting conditions. However, when these changes drastically influence the object region’s image characteristics (*e.g.*, when there is an artificial edge in image due to shadowing effects), the shape extraction scheme may find an incorrect shape. Hence, more robust methods for face tracking and expression detection are needed in future work. Future research should

mainly be directed toward improving the robustness and accuracy of the shape extraction scheme. We have the following suggestions to accomplish this:

- Because the image characteristics of the object change when the imaging conditions change, it may be necessary to introduce different templates for these conditions. This would extend the range of conditions in which the shape extraction scheme can be applied successfully. The template which is used in a given situation should be predictable on the basis of the information extracted from the coarse region.
- In the current implementation, the shape of a template is sampled according to the positions of the sample points in the image (the pixels). More accurate parameters may be found when subpixel resolution is used. In this case, the boundaries of the template are no longer sampled according to the pixel raster in the image, but are sampled along these boundaries. The image potentials at these points are then measured by linear interpolation of the measured values at the surrounding pixel locations.
- In the current implementation, parabolic boundary curve templates are used for feature shape estimation, such as eye contours, lips shape, etc., which mostly use three parametric curves with three control points for each boundary curve (*i.e.*, two endpoints and one middle point). In order to increase the accuracy of the template matching, the old template model can be improved by increasing the order of each boundary curve, through adding more control points to each outline curve. Therefore, for instance, a cubic spline curve, as a new template model, can be formed using four control points for each outline curve by adding an extra control point to an old parabolic boundary curve model.

Indeed, this idea can be extended to a so called “fine-tuned template” method, which is saying that “the existing template model can always be tuned by adding

a number of control points to achieve a better model". However, a higher order model is more prone to image noise and it is also computationally expensive. Therefore, there is a trade-off between the order (accuracy) of the model, the amount of image noise and the speed of the algorithm.

# Chapter 4

## Model adaptation

An important component in model-based coding is to synthesize the input images at the receiver side from geometric descriptions of the objects in the scene, that is synthesizing facial movements and expressions at a remote side, using the motion parameters detected on an actual facial image and animation on a model of this face. The previous chapter described how the information about the shape and position of the objects of interest, such as eyes, mouth, nostrils, nose sides, chin, are extracted. In this chapter, these 2-D observations are related to a 3-D model in front of the camera. By fitting the shape of the individual 3-D model to the observations, a reasonable geometric description of the objects of interest in the scene is automatically acquired. Although the described method is applicable for arbitrary objects, in this chapter it is mainly applied to facial images, because of the supposed importance of the facial expression generation and the compression of face-to-face video communication data.

### 4.1 Overview

To represent a facial expression, several approaches have been proposed relying on facial motion analysis and facial expression synthesis [70, 14, 25, 27, 56], the quality of the synthesized expression is directly related to the situation of the facial model adaptation. The model adaptation problem is decomposed into a global part and a local part. The global adaptation should estimate the scale, position and orientation

of the observed face and fit the model accordingly. The adaptation can be performed by an affine transformation which affects all nodes of the model in the same way. After the face model has been globally adapted, there will still be a difference between the contours of this model and the ones observed in the images (due to individual physiognomy). The local adaptation scheme intends to resolve these differences by locally deforming the face model. The emphasis, in this chapter, lies on the local adaptation.

Aizawa [3] describes a local adaptation scheme in which only the position of the points on the lower half of the face and points lying on the contours of the facial features are adapted. The scheme is primarily based on the adaptation of some control points, which are locally matched to measured contours. Thus, when the initial difference between the model and the measured contours is large, mismatches may occur. Furthermore, the repositionings of the control points on the facial features are not propagated to neighboring points, which may result on topologic changes of the model when these repositionings are large with respect to the distances between these points (*e.g.*, triangles may get overlapped with each other).

Terzopoulos [70] describes a local adaptation scheme in which a dynamic mesh is presented for adaptive sampling of an image. The distribution of the mesh follows the features of the images; however, it suffers the problem of convergence of the dynamic mesh. Stability is a major concern of this approach.

In this chapter, we present an approach to adapting the individual model onto the facial image using the extracted features and the extended dynamic mesh (called extended adaptive mesh) to compute the deformation of the face in a 3D model. In our work, the face model is locally adapted by using information of all facial feature contours simultaneously. The matching is performed in coarse-to-fine two steps – coarse adaptation and fine adaptation for both feature vertices and non-feature vertices.



The work described here is supplemental to the MPEG4-SNHC for developing very low bit-rate coding systems [73, 61, 13, 88]. As mentioned before, MPEG4-SNHC does not specify the techniques to be used for feature detection and adaptation to realize an actual implementation; this allows researchers to investigate alternative techniques. The MPEG-4 animation guideline outlines animation of feature points and suggests interpolation for non-feature points. We demonstrate that an extended adaptive mesh produces much more accurate and realistic animation than existing methods [73, 54].

After creating an individualized 3D face model [80] and extracting the feature shapes of the face, a new strategy is applied to match the model onto the face image dynamically in successive frames. First, the model is adjusted by the estimation of the scale, orientation and position of the face, which is called global adaptation. Second, a local adaptation using an extended adaptive mesh technique is applied, which is called coarse-to-fine progressive adaptation based on the mesh energy minimization. The detailed algorithm is described in the following sections.

## 4.2 Global adaptation of the individual facial model

The global adaptation scheme estimates the scale, orientation and position of the observed face and adapts the model accordingly. These parameters can be derived from the 3-D positions of a set of landmark points defined on the face. The positions of these landmark points are presumed not to be affected by deformations of the face, *i.e.*, the facial expressions. Otherwise, it is difficult to decide whether their positions are changed by the active expression or by the global adaptation parameters. Six landmark points which fulfill this condition are: the four corners of the eyes and the two centers of the nostrils, as shown in Figure 4.1. Most existing works use either

distinctive markers placed on the face to detect the landmark points [2, 27, 56] or indicate them manually in the image (at least for the first frame of the image sequence) [70]. In the previous chapter, we showed how the position of these points can be determined *automatically*.

To estimate the scale, orientation and position of the observed face, first the posi-

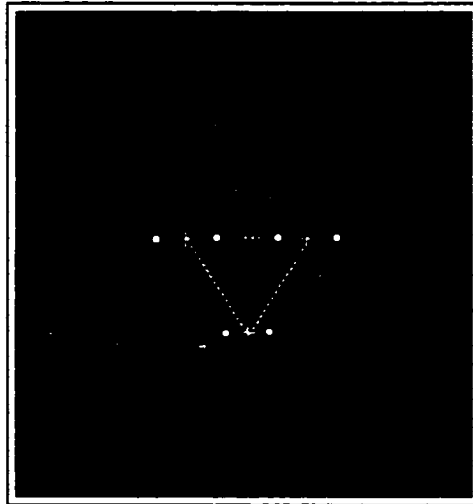


Figure 4.1: Landmark points used in the estimation of the scale, orientation and position of a motion face for the global adaptation of a model.

tion of the landmark points in the 3-D scene must be derived from their known 2-D positions in the images. By tracking the position of these landmark points through a motion sequence, the 3-D motion along with the 3-D locations of those points can be recovered. this is a typical problem related to a large body of work called *structure from motion* [29]. Because we assume a talking face has a small motion and rotation (in most situations, the rotation is around  $z$  axis), we choose for a simple version of global adaptation. The global adaptation parameters can be easily determined:

- The rotation around the  $z$  axis is determined by the angle between the line connecting the left and right eyes, the eye horizontal, and the  $x$  axis.
- The scale in the  $x$  and  $y$  directions is derived by comparing, respectively, the measured eye-to-eye distance and the measured distance between the center of

the eyes and the center of the nostrils with those distances in the individual facial model.

- The scaling in the  $z$  direction is assumed to be the mean of the scaling in the  $x$  and  $y$  directions.
- The  $x$  and  $y$  translations are derived from aligning the center of the eyes in the facial model with the center of the measured eye regions.

This method is sufficient to initialize the local adaptation scheme, because we have only used frontal view images of a speaker in the current experiments.

### 4.3 Model adaptation and animation with extended adaptive mesh

After positioning, scaling and rotating the model on the face image for a global adaptation, a local adaptation on each frame of the image sequence can then be performed for an accurate fitting. The local adaptation aims at deforming every node of the facial model onto a correct position of the face image.

To make the adaptation accurate, and the resulted facial animation realistic, we apply an extended adaptive mesh approach (*i.e.*, extended dynamic mesh (EDM)), instead of the interpolation method [6], to animate the face movement. Adaptive mesh (*i.e.*, dynamic mesh (DM)) is a well known approach for adaptive sampling of images [68, 69] and physically based modeling of non-rigid objects [70, 30]. The results shown by previous work [68, 69, 70, 30] demonstrate that this technique has become the basis for many powerful approaches in computer vision and computer graphics. The adaptive mesh can be assembled from nodal points connected by adjustable springs. The fundamental equation [68] is a second order differential equation, which can be

written as:

$$m_i \frac{d^2 \mathbf{x}_i}{dt^2} + \gamma_i \frac{d\mathbf{x}_i}{dt} + \mathbf{g}_i = \mathbf{f}_i; i = 1, \dots, N \quad (4.1)$$

where  $\mathbf{x}_i$  is the position of node  $i$ ,  $m_i$  is a point mass of node  $i$ ,  $\gamma_i$  is the damping coefficient dissipating kinetic energy in the mesh through friction,  $\mathbf{f}_i$  is the external force acting on node  $i$ ,  $\mathbf{g}_i$  is the internal force on node  $i$  due to the springs connected to neighboring nodes  $j$ .

To simulate the dynamics of the deformable mesh, the equations of motion are numerically integrated forward through time until the mesh is *nearly* stabilized. Although a number of numerical methods to solve this equation have been used (*e.g.*, Euler method, Runge-Kutta method [57, 70]), stability is still the main concern in achieving a satisfactory solution. For example, when a node moves across an image feature boundary associated with an abrupt change in its image intensity, the stiffness of those springs connecting with the node changes rapidly and results in a possible reversal of the nodal force, which may lead to perpetual oscillation of the node. In this type of situation a new equilibrium state cannot be reached. To make the mesh converge to a stable state,  $m_i$  and  $\gamma_i$  must be carefully chosen. The overdamped behavior (*i.e.*, large values of  $m_i$  and  $\gamma_i$ ) will contribute to enhancing the stability of the numerical simulation, however it is at the expense of the accuracy of the solution. To make the solution more stable and accurate, we extend the conventional dynamic mesh method [68] by introducing a so called “energy-oriented mesh” (EOM) to refine adaptive meshes. The major differences between conventional dynamic mesh (DM) and the EOM are: (1) EOM makes the mesh movement in the direction of mesh energy decrease instead of decreasing the node velocities and accelerations; (2) EOM checks the node energy in each motion step without considering the velocity, it is independent of the DM and can be the supplemental step to DM for stabilizing mesh movements. Therefore, our model adaptation procedure consists of two major steps:

(1) coarse adaptation, which applies the DM method to make the large movement converge quickly to the region of an object; (2) fine adaptation, which applies the EOM method to finely adjust the mesh obtained after the first step and make the adaptation more “tight” and “accurate”.

### 4.3.1 Principle of Energy Minimization in Energy Oriented Mesh

According to the principle of minimum potential energy, of all possible kinematically admissible displacement configurations that an elastic body can take up, the configuration which satisfies equilibrium makes the total potential energy assume a minimum value [18]. The potential energy stated in the principle of minimum potential energy includes the strain energy and the potential energy formed by external forces. In our model, there is no external force and all the strain energy is stored as the elastic energy in springs. To reach the equilibrium state, the elastic energy in the springs has to be minimized by displacing nodes. If we let node  $i$  move under a nodal force while all the neighboring nodes are fixed, the node will move in the direction of the nodal force because the gradient of total spring energy on node  $i$  ( $E_i$ ) is in the same direction as the nodal force ( $\mathbf{g}_i$ ). This implies that for meshes associated with the image observations, if we let nodes move by successive steps based on the principle of minimum potential energy, and reduce strain energy at each step, finally we should obtain a fine adaptation on this image.

When a spring mesh is not in an equilibrium state, those nodes with non-zero forces acting on them tend to move in the direction of the resultant nodal forces. The movements of nodes will reduce the energy caused by strain. When a final equilibrium state is reached, no further movements will occur. In order to prevent a node from being over-displaced at each step, energy change for each step must be checked to ensure that the step has reduced the energy in a non-increasing way along the direction of

movement.

### 4.3.2 Detailed Algorithm

In order to synthesize the subtle expressions, all the detailed movement of the adaptive model is highly desirable, especially on the expression-*sensitive* regions (for example, eye regions, and mouth regions). Our individualized model has high resolution vertices (*e.g.*, 190 vertices for the eye region, 310 vertices for the mouth region). The following is the main procedure for the facial model adaptation:

1. Based on the feature shape detected, the corresponding feature points on the template can be determined simply by computing the points on the boundary of the parabola, according to the distribution of the vertices on the model. The displacement from the feature vertices on the model to the feature points on the face image can be easily derived. The adaptation of these feature vertices is carried out directly.
2. To adapt the remaining vertices (*i.e.*, non-feature vertices) onto the face image, two steps are applied: (a) coarse adaptation (DM method), (b) fine adaptation (EOM method).

#### (a) Coarse adaptation:

Solve the dynamic motion equation (4.1) using conventional explicit Euler time-integration procedure [57]<sup>1</sup> until the motion parameters (velocity  $\mathbf{v}_i$  and acceleration  $\mathbf{a}_i$ ) are less than a certain threshold.

$$\mathbf{a}_i^t = \frac{B_i}{m_i}(\mathbf{f}_i^t - \gamma_i \mathbf{v}_i^t - \mathbf{g}_i) \quad (4.2)$$

$$\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \Delta t \mathbf{a}_i^t \quad (4.3)$$

$$\mathbf{x}_i^{t+\Delta t} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+\Delta t} \quad (4.4)$$

---

<sup>1</sup>The procedure of numerical solutions for solving the second order differential equation can be found in the site <http://www.nr.com/>

where  $B_i$  denotes an operator whose role is to enforce boundary conditions or constraints. Equations (4.2), (4.3), and (4.4) are evaluated for all nodes, *i.e.*,  $i = 1, \dots, N$ , and consecutive time steps, *i.e.*,  $t = 0, \Delta t, 2\Delta t, \dots$ , until  $\mathbf{v}_i$  and  $\mathbf{a}_i$  are less than a certain threshold.

In our implementation of Equation (4.1) no external force is involved. The boundary vertices are fixed. These include the *feature vertices* defined on the face model and the *vertices on the contour of the face*. Let node  $i$  be connected to a set of nodes (as denoted  $M_i$ ). The set  $M_i$  contains  $n_i$  nodes, *i.e.*, node  $i$  is attached to  $n_i$  springs. The total internal force acting on node  $i$  due to these springs' movement is:

$$\mathbf{g}_i = \sum_{j \in M_i} C_{ij} (\|\mathbf{r}_{ij}\| - l_{ij}) \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|} \quad (4.5)$$

where  $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ ,  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are the positions of nodes  $i$  and  $j$ ;  $l_{ij}$  is the natural length of the spring connected from nodes  $i$  to  $j$ ;  $\|\mathbf{r}_{ij}\|$  is its actual length; and  $C_{ij}$  is the stiffness of the spring  $ij$ .

Based on the nodal value (*i.e.*, intensity in the nodal position), the springs automatically adjust their stiffness so as to distribute meshes in accordance with the local complexity of the image. Before calculating the stiffness, we apply a Sobel operator to obtain a gradient image, then normalize the intensity values of the gradient image within the range of  $[0, 1]$ . Suppose the stiffness of a spring changes linearly, along with the nodal values on the normalized gradient image, the calculation is then as follows:

$$C_{ij} = -(k_2 I + k_1) \quad (4.6)$$

where  $k_1$  is the pre-defined minimum stiffness of springs in the mesh;  $k_1 + k_2$  is the maximum stiffness of springs; and  $I$  is derived from the nodal

values on the normalized gradient image. Unlike the stiffness calculation in [68], which takes the average of two nodal values on a spring, we apply a weighted sum of nodal values as shown in Equation (4.7). This implies that the node closer to the feature vertices will contribute more to the stiffness.

$$I = \frac{d_j}{d_i + d_j} S_i + \frac{d_i}{d_i + d_j} S_j \quad (4.7)$$

where  $S_i$  and  $S_j$  are the nodal values on the normalized gradient image at nodes  $i$  and  $j$  respectively.  $d_i$  (or  $d_j$ ) is the minimum distance from node  $i$  (or  $j$ ) to the nearest vertex in the set of extracted feature vertices.

To obtain the nodal values ( $S_i, S_j$ ), we use the conventional finite element concept to calculate the *sub-pixel* values in between the neighboring pixels. As shown in Figure 4.2, we split the pixel rectangle into two triangular elements so that sub-pixels within a certain triangle (a plane) are linearly distributed with the same property. The purpose of splitting into two triangular elements is to prevent the node movement over-displacing in the next fine adaptation process (*e.g.*, jumping across an edge boundary in a motion step). Let A denote a pixel at position  $(x_A, y_A)$  having value  $I_A$ . Four neighboring pixels A, B, C, D are split into two triangular elements  $\triangle ABC$  and  $\triangle BCD$ . The value of sub-pixel  $p$  within  $\triangle ABC$  can be obtained from Equation (4.8) (see [18]):

$$I_p = a_1 x_p + a_2 y_p + a_3 \quad (4.8)$$

where

$$a_1 = (x_C - x_B)I_A + (x_A - x_C)I_B + (x_B - x_A)I_C \quad (4.9)$$

$$a_2 = (y_B - y_C)I_A + (y_C - y_A)I_B + (y_A - y_B)I_C \quad (4.10)$$



$$a_3 = (x_{BYC} - x_{CYB})I_A + (x_{CYA} - x_{AYC})I_B + (x_{AYB} - x_{BYA})I_C \quad (4.11)$$

Similar equations can be used if the point  $p$  is within  $\triangle BCD$ .

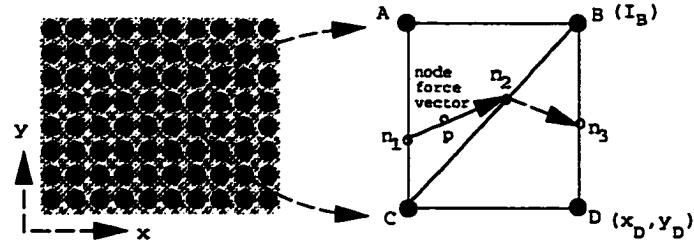


Figure 4.2: Node displacement within a triangle area in each step:  $n_1$ : node start position;  $n_2$ : node end position;  $n_3$ : node end position in next step.

(b) **Fine adaptation:**

After a mesh stabilizes, fine adjustments can be done by EOM method for making meshes converge to the image “tightly”. Assuming that the image intensity changes continuously over the spatial domain, the grey values in between pixels can be obtained from Equation (4.8). The criteria of the fine movement of nodes is that *only movements that decrease the node energy stored in the connected springs are allowed*. The nodes energy calculation and the nodes motion rules are described below:

- **obtain node force.** Of all the nodes on the mesh, except the boundary vertices, find a node with the largest value of the nodal force using Equation (4.5), *e.g.*,  $g_i$ . Within a sub-pixel domain (triangle area), search sub-pixels along the direction of the  $g_i$  vector in order to find one having minimum node energy.
- **obtain node energy.** The strain energy  $E_{ij}$  stored in a spring  $ij$  is calculated as follows:

$$E_{ij} = C_{ij}(\|\mathbf{r}_{ij}\| - l_{ij})^2 \quad (4.12)$$

*Node energy*  $E_i$  is defined as the summation of the energy stored in all the springs connected to node  $i$ , *i.e.*,

$$E_i = \sum_{j \in M_i} E_{ij} \quad (4.13)$$

- **Displacement of a node.** Displacement of a node at a step is along the direction of the resultant nodal force. Theoretically, a node should move to a new position within the triangle domain where the node energy is a local minimum. To simplify the computation, in the current implementation, we calculate the node energy in three positions – node start position (*e.g.*,  $n_1$ ), middle position (*e.g.*,  $p$ ), and node end position (*e.g.*,  $n_2$ ). The position with the minimum node energy is the new position that the node is allowed to move to. So a displacement at a step is only within a triangular area (including the boundary lines  $AC$  and  $BC$ , for example in Figure 4.2 from the node start position to the node end position). The maximum displacement of a node in one step will not exceed the distance between two adjacent pixels.

The rules for moving a node to a new position follow two conditions:

- Check the node energy at the start position and the end position. The energy must decrease; this ensures that the spring system has reached a state with less energy.
- To prevent the reversal of nodal force in the new position, the inner product of the force vector ( $F_L$ ) at the node position with the force vector ( $F_M$ ) at the new position must be greater than zero; this prevents the oscillating movement of nodes.

If the two conditions above are satisfied, the node is allowed to move to the new position. Otherwise, the node stays in its current place, and the procedure checks the node with second largest force, repeats

the above procedure, and continues until no node satisfies the above two conditions or the largest nodal force in the mesh is small enough (less than a certain threshold). Figure 4.3 shows a flowchart of the extended adaptive mesh algorithm in its current implementation.

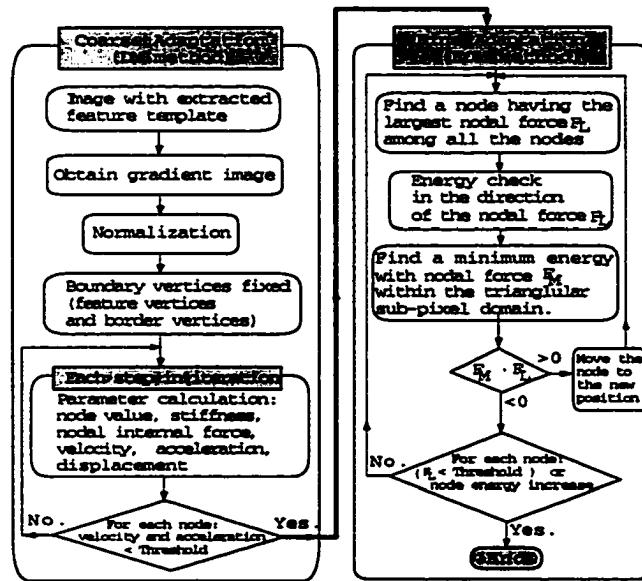


Figure 4.3: Schematic diagram of the extended adaptive mesh method.

## 4.4 Experimental Results

### 4.4.1 Experiment I

To evaluate the algorithms developed, and present the experimental result clearly, we take the eye images as an example to show how the coarse-to-fine adaptation works. Experiments with various color images of different eyes and eye sequences were made. The eye template matched to the iris and the eye lids in one image sequence are shown in Figure 4.4. The animated eye sequence (for Figure 4.4 images) after the first step (DM) and the second step (EOM) are shown in Figure 4.5, in which a plane mesh is used for testing our extended adaptive mesh algorithm. The overlapped results of coarse and fine adaptations are shown in Figure 4.6. Figure 4.7 shows the texture-mapped results using the first frame texture of the original sequence.



Figure 4.4: Extracted eye templates in an actual sequence of eye images (frame 1, 24, 68, 169).

The improvement of the adaptation accuracy from coarse adaptation (DM) to fine adaptation (EOM) can be clearly seen by comparing the top sequence with the bottom sequence in Figure 4.5 and Figure 4.6. Figure 4.7 (bottom) shows that the synthesized eye movements using the extended adaptive mesh are very close to representing the original sequence (Figure 4.4).

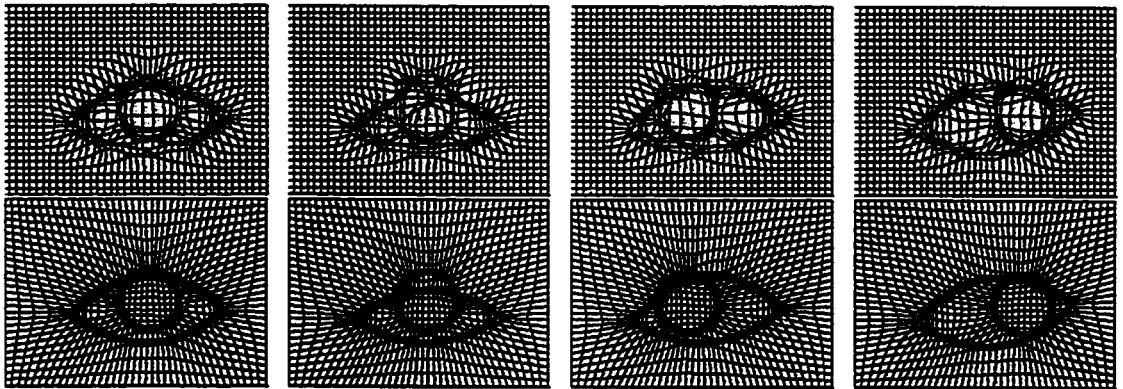


Figure 4.5: Eye movement. [top]: coarse adaptation using DM method ( $m_i = 1.2$ ,  $\gamma_i = 1.2$ ,  $k_1 = 1.0$ ,  $k_2 = 9.0$ , threshold of  $v_i$  and  $a_i$  are 60.0 at the time of stopping adaptation). [bottom]: fine adaptation using EOM method (the largest nodal force is 0.05 when adaptation is stopped).

To apply a real eye model on the eye image, 11 feature vertices are defined for each eye, as shown in Figure 4.8. Once the eye lid contour and the iris are detected in the image sequence, the corresponding feature points on the template can also be determined simply by computing the points on the boundary of the parabola, and by using the center of the circle. The real eye model adaptation on the eye sequence is shown in Figure 4.9.

Applying the same extended adaptive mesh method, the other regions on the same

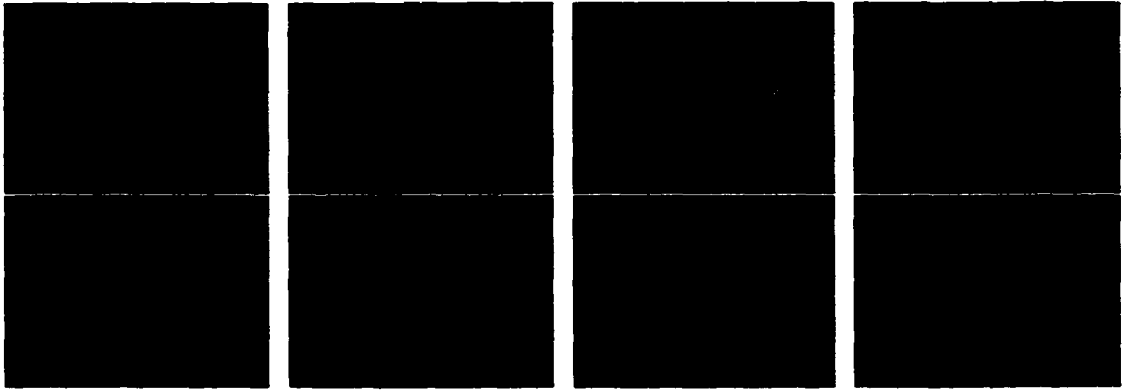


Figure 4.6: Overlapped eye mesh: coarse adaptation (top); fine adaptation (bottom).

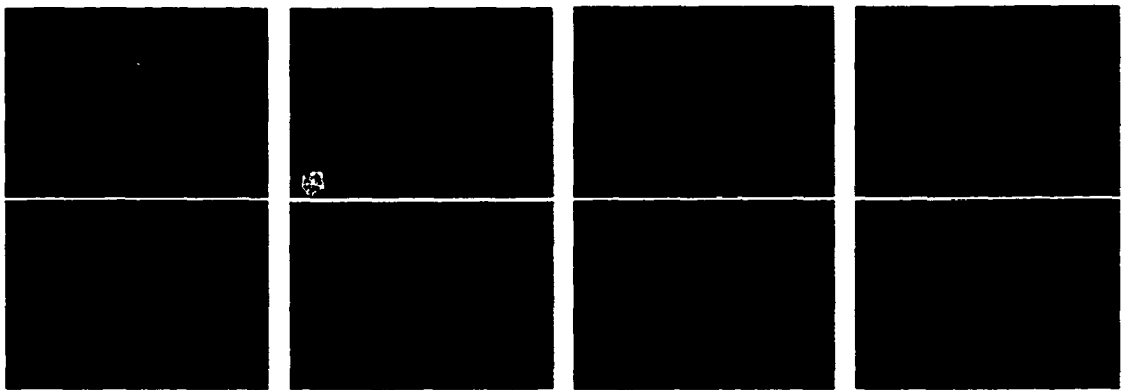


Figure 4.7: Synthesized images using the first frame texture: texture-mapped results of coarse adaptation mesh (top) and the fine adaptation mesh (bottom).

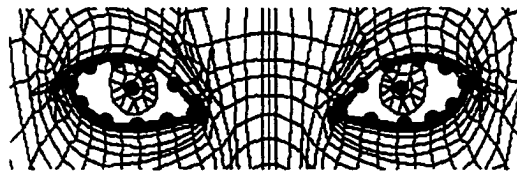


Figure 4.8: Defined eye feature vertices.



Figure 4.9: Matched and animated 3D eye model

<i>Synthetic face (SGI man)</i>	<i>mean errors (in pixels)</i>	
	coarse adaptation	fine adaptation
Number tested frames (140)		
eye (l)	3.15	0.66
eye (r)	2.98	0.61
eyebrow (l)	2.79	0.32
eyebrow (r)	3.04	0.33
nose	3.57	0.77
mouth	3.97	0.83
forehead	4.01	1.03
cheek	4.12	1.06
<i>whole face</i>	<i>3.45</i>	<i>0.70</i>

Table 4.1: Performance evaluation of adaptation errors

set of faces can also be adapted. The theory and implementation behind fitting the other regions of movements using the energy minimization of extended dynamic mesh are similar to eye tracking and adaptation.

#### 4.4.2 Experiment II

The accuracy of the adapted model is tested by an artificial facial sequence, whose adaptation results are known beforehand. To accomplish this, we first manually fit the model to the first frame of a face, and synthesize the subsequent frames by invoking Action Units – the shape of the facial features (such as eyes, mouth) can be controlled. Synthetic images of a person with changed facial expressions were created by mapping the texture onto the deformed models. Because the facial features are changed by invoking Action Units, the synthetic features corresponding to the fitted model are best matched, and the shape and positions of the vertices on the feature regions are also known.

Figure 4.10, 4.11, 4.12 are the artificial test sequence, which shows the deformation of the face expressions derived from the initial frame. The adaptation errors in positions between the original vertices and the adapted vertices are listed in Table 4.1. Observe that the main advantage of our method is the improvement in accuracy

and quality of the animated mesh from the coarse adaptation to the fine adaptation.

Note that even though the DM method is faster, the mesh is not as accurate as our EOM improvement. The EOM improvement is essential to create realistic animation.

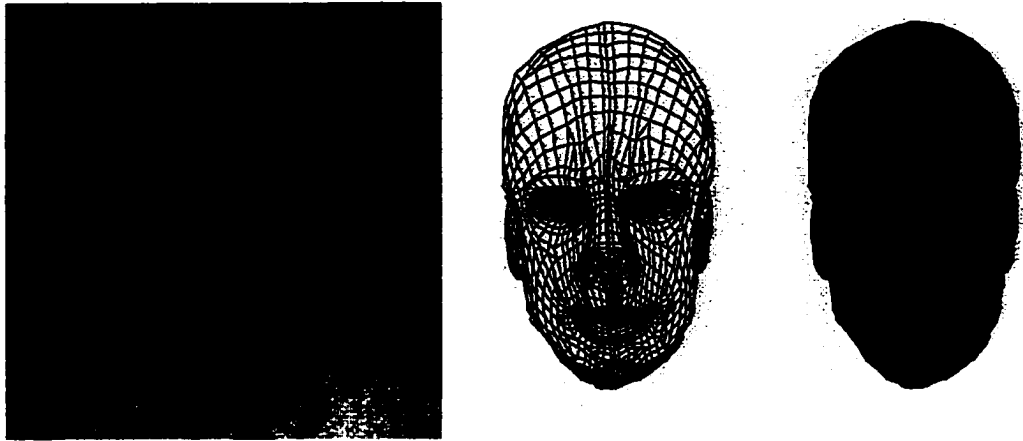


Figure 4.10: Original facial image (man from SGI), the individualized model and the adaptation result.

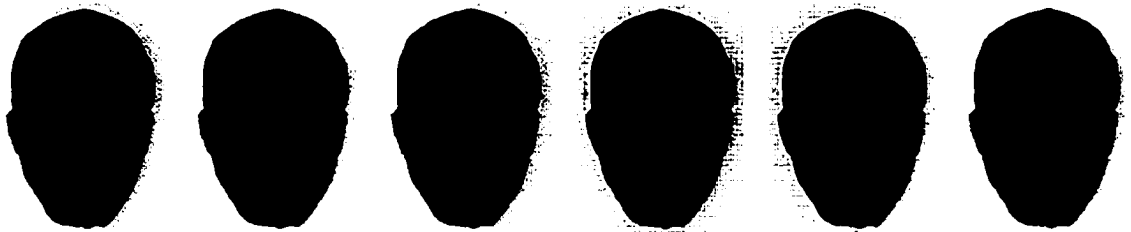


Figure 4.11: Synthetic facial expressions (eye regions), frames 1, 10, 24, 36, 45, 70.



Figure 4.12: Synthetic facial expressions (eyes, lips, etc.), frames 80, 85, 90, 120, 130.

### 4.4.3 Experiment III

A real video sequence is tested with different expressions. as shown in Figure 4.13 and Figure 4.14. From the experimental results, we can see that the algorithm proposed here behaves well for adapting various facial expressions.

Our model adaptation algorithm is also tested on other video sequences (named

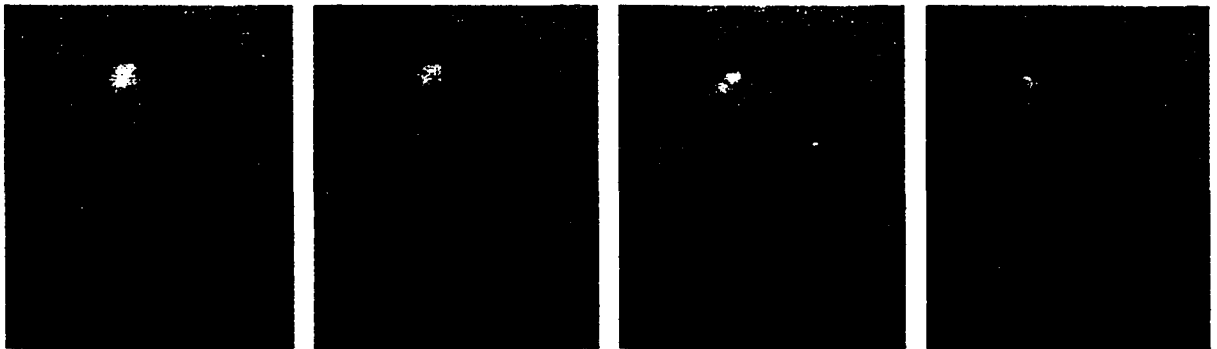


Figure 4.13: Original video input (Mario: frames 15, 18, 43, 63)

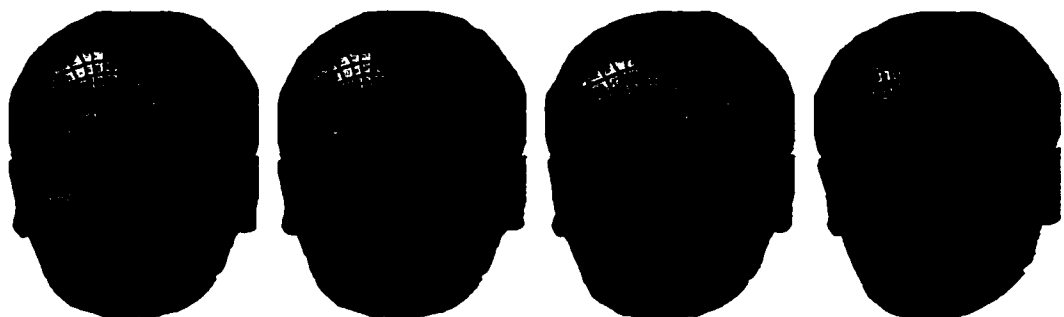


Figure 4.14: Face model adaptation (frames 15, 18, 43, 63)

Dima and Stan) with pan and tilt movement of the active platform. Figure 4.15 and Figure 4.16 show the final results after the background compensation and model adaptation. The results demonstrate the accurate adaptation of our system by perceptions.



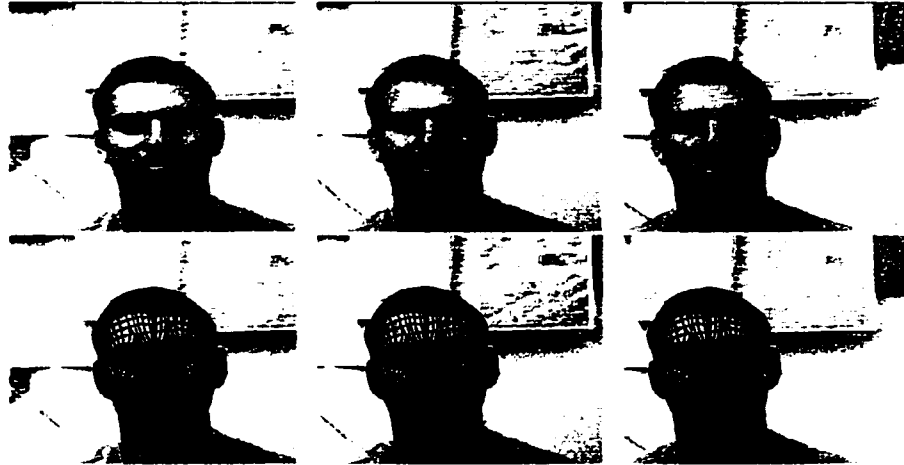


Figure 4.15: Model adapting on a video sequence with a pan movement of an active platform (Dima: frame 3, 21, 33, size of 480\*480 pixels per frame). top row: original frames. bottom row: results of model adaptation.

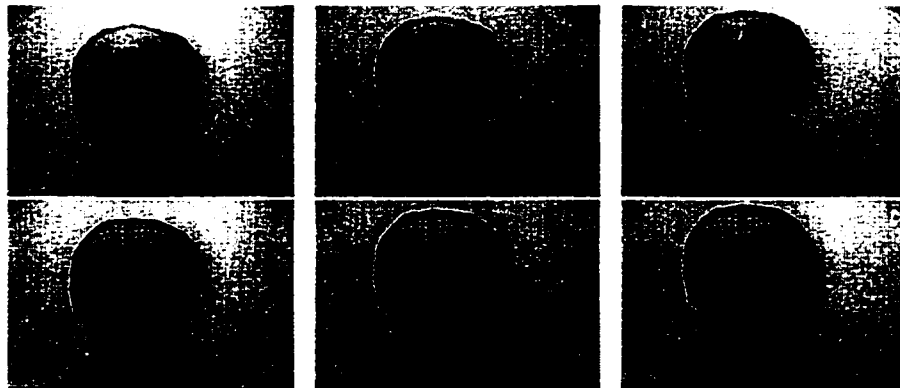


Figure 4.16: Model adapting on a video sequence with a tilt movement of the active platform (Stan: frame 17, 28, 39, size of 640\*480 pixels per frame). top row: original frames. bottom row: results of model adaptation.

Experimental Sequences	Computation Time	
	DM (sec/frame)	EOM (sec/frame)
<i>Tested Eyes</i>	14.5	25.7
<i>Synthetic Eyes</i>	12.8	23.5
<i>Synthetic Lips</i>	11.9	22.3
<i>Mario Face</i>	32.4	79.7

Table 4.2: Computation time in DM and EOM. (note that above numbers represent the average time per frame in the individual sequence)

## 4.5 Discussion

In this chapter, an energy oriented method is described as an extension to dynamic meshes consisting of a network of springs. The proposed method refines the model adjustment process so as to improve the accuracy of model adaptation and tracking. It has also overcome the convergence problem that is commonly encountered in numerical solutions to dynamic-motion equations. Experimental results show that realistic animations with subtle expressions can be achieved by our system. We expect that the algorithms proposed in this chapter will contribute towards future modification to MPEG-4(SNHC).

However, the method proposed in this chapter for model adaptation is very time-consuming. EOM method takes more computation time than DM method. Table 4.2 lists the computation time in various experiments when running on a SGI-O2 machine. Speeding up the process of EOM and DM is highly desired in our future work. In our current implementation, the maximum force that satisfies the displacement criteria is searched from all the nodes after each movement of a node, which takes significant computation time. One alternative is to move a node until the node cannot be moved any further, while keeping the neighboring nodes fixed. This change is expected to reduce the search time and improve the efficiency of the EOM adaptation process. Note that the accuracy of the coarse-to-fine adaptation is at the cost of a higher bit-rate, because more motion parameters of vertices need be transmitted as compared to

the interpolation method. We will investigate developing an efficient parameter compression method for non-feature vertices in the future. Another aspect that should be noted is that the global adaptation applied here has a strict assumption: the motion is almost in 2-D plane, no large motion (*e.g.*, large rotation) is considered. Considering the real situation, a true 3-D global motion parameters should be estimated by using more complex techniques, such as structure from motion.

# Chapter 5

## Active texture detection, compression and synthesis

In the previous chapters, we discussed the facial features extraction and the facial model adaptation for tracking the motion of the face. The objective of the motion tracking is to synthesize the facial expressions at the receiver side using these extracted animation parameters <sup>1</sup> and the original individual facial model. However, due to the limitations of current computer graphics technology, mimicing realistic facial textures, such as wrinkles, is very difficult. Facial texture updating and compression are crucial in order to achieve realistic facial animation for low bit-rate coding. In this chapter, we present an efficient way to update and encode facial textures in model based coding. Differing from traditional methods, which update the entire facial texture, we propose a partial texture updating method for realistic facial expression synthesis with facial wrinkles. First, fiducial points on a face, such as corners of facial organs, are estimated. After the model adaptation is performed on a sequence of face images, all the face textures are normalized to an initial frame model size. Next, textures of interest (TOI), in the potentially expressive wrinkles and mouth-eye texture areas, are captured by exploiting the geometric information based on the detected fiducial points. Among the TOI, the so-called active textures (AT) or expressive textures

---

<sup>1</sup>animation parameter (also called motion parameter or motion vector), is the trajectory of the model vertex from one frame to another frame. Since the model is adapted to the face in each frame, the motion vector of each vertex can be easily derived by its positions corresponding to each frame.

(ET), which are generated due to acting the different expressions, and which are used later for facial texture updating and synthesis, can be detected by exploring temporal correlation information. Finally, the K-L (Karhunen-Loeve) transformation is used for the efficient compression of all the active textures. The entire facial texture at the current frame is synthesized from the initial frame texture and the decoded active textures from the subsequent frames; then a temporal blending technique is used to make the results more smooth and realistic. Compared to the entire texture updating scheme, partially updating and compressing facial textures significantly reduces the computation intensity and bit-rate, while still producing an acceptable visual quality. In this chapter, experiments on the video sequences will demonstrate the advantage of the proposed algorithm in keeping transmission cost low while producing realistic expressions through partial texture update and blending.

## 5.1 Overview

Keeping high fidelity and reducing the number of bits transmitted are the major objectives in realizing low bit rate video coding. Up until now, most work in this field has concentrated on facial feature detection, 3D facial model adaptation, and motion estimation and tracking [67, 2, 14]; little investigation has been done into strategies for facial texture updating and encoding. Accuracy in 3D geometry and face texture contribute to the reconstruction of realistic facial expressions. The vivid facial textures, such as furrows, which are generated by different expressions, are very difficult to capture and model. To create these features using geometric data (geometric model) rather than texture data would require an extremely detailed 3D capture of the face geometry, and a resulting high polygon count in the 3D model. In addition, shading these details properly, if they were represented geometrically, would be very difficult since it would require computing shadows and possibly even diffuse inter-reflection effects in order to look realistic. The most widely used facial action coding system (FACS) [21], the representation of facial motion, has also been shown

to have difficulty producing a large variety of complicated facial expressions [25]. Due to these difficulties in current computer graphics technology, facial texture extraction and update are the best choice for our current application. If facial texture can be updated for every frame, the subtle change of facial textures (*e.g.*, facial wrinkles) generated by a variety of expressions can be easily captured and compensated for, and thus a realistic facial animation result can be achieved. However, the entire texture updating using traditional waveform encoding methods (such as JPEG and MPEG) [28, 41] is an expensive approach requiring high bandwidth (bit rates). Guenter *et al.* [27] produced a face by capturing six camera views of high resolution, registered video images of the face, and by tracking the geometric deformation using colored circular paper fiducials glued on a person's face. Although the system produced a synthesized face of impressive quality, physical dots must be glued on a real face, and a huge amount of storage space is required for texture-mapping. Around 200kbits/sec bit rate can be achieved while keeping the high quality image, which is, however, not in the range of very low bit rate transmission (*i.e.*, less than 64kbits/s) [34, 59, 65]. Tao and Huang [67] proposed a piecewise Bezier volume deformation model to track the face motion, and impressive results of the synthesized face expressions are achieved. However, only an initial frame texture is used; the realistic texture (*e.g.*, wrinkle) is still hard to synthesize whereas it is desired for a lifelike expression generation.

Strom *et al.* [62] coded the entire face texture by employing the classical eigenface approach, which requires a large number of faces as a training set to create a set of eigen textures beforehand (*i.e.*, working off-line). Even though the image is divided into small blocks, the complexity of computation is still a major concern in that scheme. In addition, the model adaptation and geometric normalization are performed manually. To achieve a SNR (signal-noise ratio) of 35dB, the bit rate must average over 200kbit/sec, while very low bit rate video coding requires less than 64kbits/sec. Recently, Nishino *et al.* [51] came up with a similar idea (called eigen texture method)

to represent and interpolate the object texture in the eigenspace domain, in which an image is divided into a number of small triangle patches. However, a sequence of triangle patches with 360-degree viewer direction is required for deriving the transform matrix using principal component analysis. The heavily computational expense and large number of eigen textures storage are still the main drawbacks. To alleviate the computational load, some new approaches which are less time-consuming have been investigated recently. A real-time rendering system using a multi-texture mapping method was reported by Kunita et al. [37]. This prototype system can synthesize an image at a high spatial sampling density; however, it requires 12 synchronized CCD cameras working together to capture textures simultaneously. Conceivably, more bits are needed to encode such a large number of textures.

In this chapter, we propose an efficient method to detect and update the facial textures in active wrinkle areas and mouth-eye areas, and eventually synthesize the facial expressions using these captured textures to achieve a very low bite rate coding (less than 40kbit/s). The fiducial point tracking and model adaptation are performed automatically, as described in the previous chapters.

Psychology research shows that a significant contribution to a realistic facial expression comes not only from facial organs (*i.e.*, eyes, mouth), but also from facial wrinkles generated from the expression (*i.e.*, expressive wrinkles) [22, 53]. The most significant wrinkles on a human face are categorized into four types: (1) forehead wrinkle; (2) glabella wrinkle (in between eyebrows) (3) crows-feet wrinkle (outer corners of eyes), and (4) nasolabial wrinkle (below the cheek, linking nose side and mouth corner), as shown in Figure 5.2(a). In this work, instead of updating the entire facial texture [62, 49, 27, 56], we update only partial textures, defined in the wrinkle areas and mouth-eye areas, which are called *textures of interest* (TOI). Among the TOI, only the significant textures which are generated along with the different expressions

are used for image synthesis; they are called *active textures* (AT). To synthesize a realistic face expression, a wireframe model is first adapted onto the face images to track the movement of expressions. The TOI is estimated based on the fiducial points detected, such as the corners of facial organs, and the geometric information. The subsequent facial textures are normalized to an initial frame model, which is taken as a standard size. Before extracting the active textures on the normalized face image, the TOI with arbitrary shapes is transformed (normalized) to a rectangle shape. Among the normalized TOI, the *active textures* can be extracted by computing the temporal texture correlation between the current frame and the initial frame. The active textures are then coded and transmitted to the other side in order to synthesize the entire facial texture. The principal component analysis method is applied to compress the active textures. To synthesize the facial texture on the receiving side, the first frame texture is mapped onto the subsequent deformed frame model, then the active textures which have been decoded are added onto the corresponding position of the face. A temporal blending technique is used to make the synthesized texture appear more smooth. The system is outlined in Figure 5.1. The animation parameters can be compressed by the “PCA+DCT” compression algorithm presented by Tao and Huang [66] in MPEG-4 standard.

The fiducial points estimation, the active texture extraction, compression and synthesis will be described in detail in the following sections.

## 5.2 Fiducial points estimation

In order to obtain the textures of interest, a 3D wireframe face model is first matched onto an individual face to track its different facial expressions. The correct estimation of the facial fiducial points produces a correct matching of a facial model with the face.

The accurate matching of the model leads to an accurate tracking of facial expres-



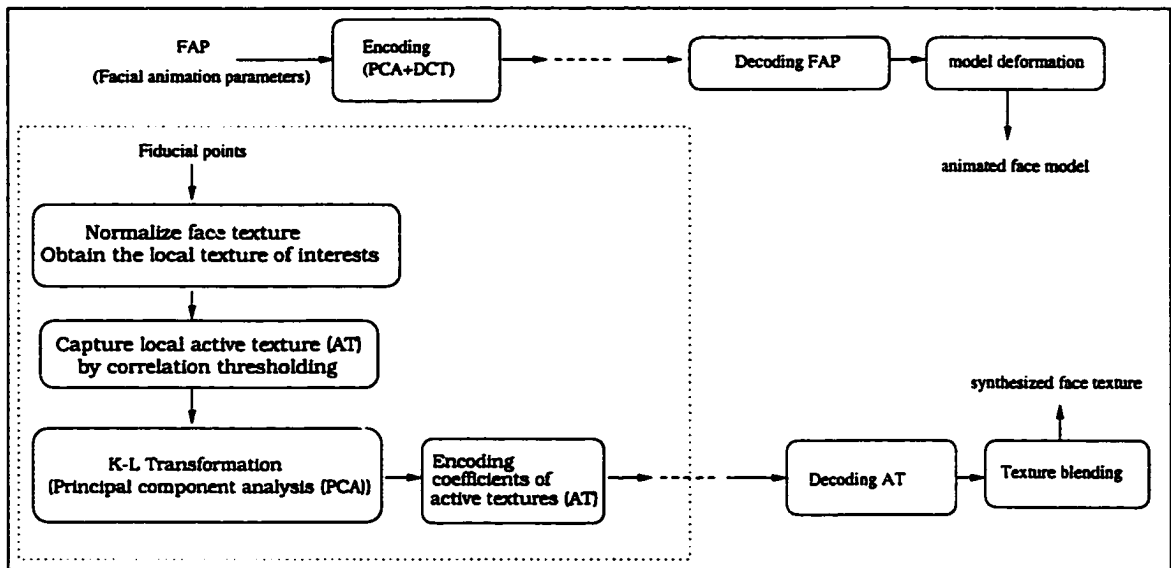


Figure 5.1: Flow chart of the active texture (AT) detection and synthesis

sions, gives a correct indication of the positions of the fiducial points on the facial features, and thus results in a correct selection of textures of interest (*e.g.*, wrinkle textures, eye and mouth textures).

### 5.2.1 Fiducial points

Twenty-seven fiducial points are defined on a human face – located on eyebrows, eye corners, nose sides, mouth corners, and hair contour – as shown in Figure 5.2(a). The corners of feature regions are the key information provided for forming the geometrical areas of the TOIs. For example, the inner corners of eyebrows and eyes are used for determining the location of the glabella region. Nose side is also an important geometric feature in our application which aids in locating the nasolabial wrinkle area effectively. This feature can be extracted using the template matching method as described in the previous chapters. The hair boundary, which is adjacent to the facial skin, can be estimated using the region growing method based on the distinctive color property between the hair and the facial skin so that these two regions can be distinguished. The shape of the eyebrow is determined by integral projection approach, which has been described in Chapter 3. After applying the active tracking algorithm

and color-based deformable template technique, all the feature shapes (such as eye and mouth) are extracted, then twenty-seven feature points (corners) distributed on the corresponding shapes can be obtained. Figure 5.2(b) illustrates the 27 feature points detected on a person's face. Figure 5.3 shows two adapted models, with a natural expression and a smiling expression, notice that the positions of fiducial points are changed accordingly in the nose side, mouth corner, eye corner, etc.

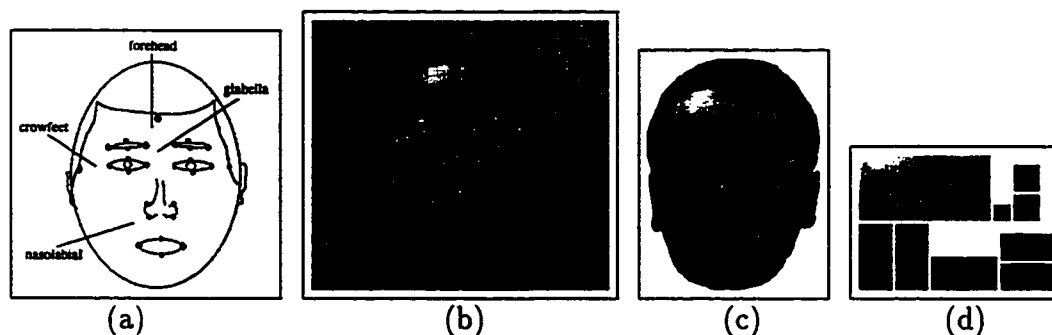


Figure 5.2: (from left to right) (a) Feature points defined on a human face; (b) An example of feature points extracted on a person's face; (c) textures of interest; (d) Normalization of textures of interest (from left to right, UP: forehead area, glabella area, and left-right crowsfeet area; DOWN: left-right nasolabial area, mouth, and left-right eye areas).

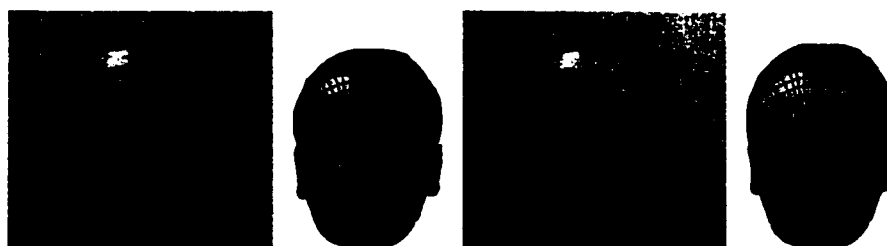


Figure 5.3: Adapted models: frames 1 and 21.

### 5.3 Active texture detection

After the facial model is fitted onto a face in a video sequence, the facial expressions are represented by a series of deformed facial models. As we discussed before, a deformed facial model represents only a geometric structure of a facial expression, which mainly reflects the expression of the eyes and mouth areas. This is not enough

to represent a realistic vivid expression of a face. In order to reconstruct a lifelike expression of a face, it is necessary to provide the texture information of the face corresponding to the individual expression. The fitted model sketches the accurate location of texture areas. Since the size and shape of textures of interest vary with different expressions, the adapted facial model must be warped to a standard shape and size (*i.e.*, the first frame). A face model consists of a number of triangle elements. Figure 5.4 shows the principle of the geometric transformation with a triangle element. A pixel  $k$  within a triangle element  $P$  is transformed into a new pixel position  $k'$  within the standard triangle element  $P'$ , as formulated in Equation 5.1.

$$k' = T \cdot k \quad (5.1)$$

where  $k = [x_k, y_k, 1]^T$ ,  $k' = [x_{k'}, y_{k'}, 1]^T$ . The transformation matrix  $T$  is derived from the relationship of the two triangle elements, as shown in Equations 5.2 and 5.3.

$$P' = T \cdot P \quad (5.2)$$

$$T = P' \cdot P^{-1} \quad (5.3)$$

where

$$P' = \begin{bmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ 1 & 1 & 1 \end{bmatrix}, T = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix}, P = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \quad (5.4)$$

Figure 5.5 shows an example of model transformation from  $k$ th ( $k = 15$ ) frame

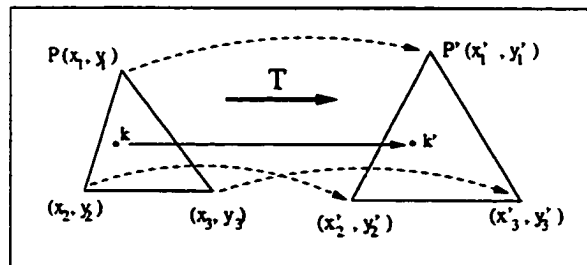


Figure 5.4: Texture transformation within a triangle

model to 1st frame model, and the corresponding textures. After the normalization,

the subsequent faces with different expressions have the same shape, but different texture than the first frame.

Since the fiducial points are extracted in the previous stage, the TOI areas can be determined by the geometric relationship of these fiducial points. Figure 5.2(c) shows

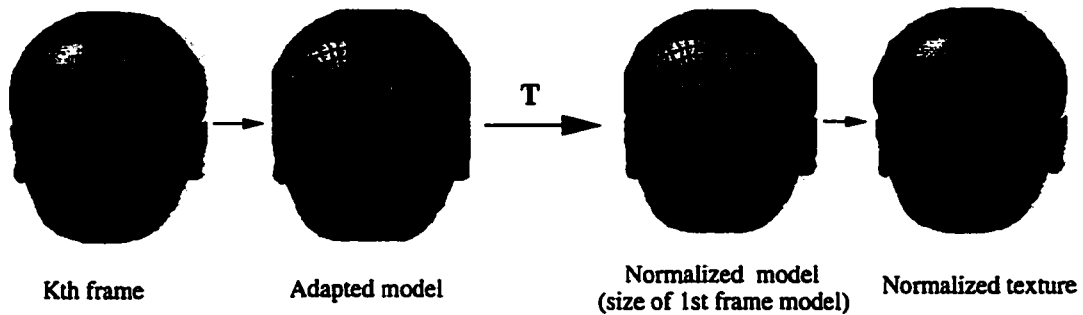


Figure 5.5: Example of whole face texture transform: from 15th model (texture) to 1st frame model (texture).

the textures of interest formed from the fiducial point locations. For example, a forehead wrinkle area is a quadrilateral, which is limited to the left-most and right-most areas of eyebrows in width, and the bottom-line of hair and top-line of eyebrows in height; a glabella wrinkle area is a block in between the inner eye corners in width, and the top-line of eyebrows and a line linking inner corners of eyes in height; a crows-feet wrinkle occurs between the outside corner of the eyebrow and top of the nose in height, and the outside corner of the eye and the hair contour in width; a nasolabial area is a polygon which is formed by the side of the nose, outside corner of the mouth, and outside corner of the eye. To reduce the computation time, the size and shape of the extracted textures are re-normalized to a standard size and shape, which is smaller than the actual size, using the geometric transformation method in Figure 5.4. Figure 5.2(d) shows the normalized TOI in wrinkle, eyes and mouth areas.

Although the textures of interest are extracted in each frame, not all the textures extracted contribute to the facial expression synthesis. To represent the facial texture

efficiently, only the *active textures* (AT), which are the typical textures representing significant facial expression, need to be transmitted. The active texture detection conforms to the following criteria: If we view a specific texture of interest (TOI) along consecutive frames – for example, the nasolabial wrinkle texture – each nasolabial wrinkle texture of the successive frame is correlated with the first frame to a certain degree. The higher the correlation computed between two nasolabial wrinkle textures, the closer their similarity is. In other words, the active wrinkle generated from the different expressions can be estimated by a lower correlation between the wrinkle of the current frame and the initial first frame. Similarly, the active textures in the mouth and eye areas can also be estimated using the same principle. The correlation coefficient  $C_w(w_{t_0}, w_{t_k})$  of a certain TOI  $w$  between the  $t_k$  frame and  $t_0$  frame is computed as follow:

$$C_w(t_0, t_k) = \frac{E_w(w_{t_0}w_{t_k}) - m_{w_{t_0}}m_{w_{t_k}}}{\sigma_w(t_0)\sigma_w(t_k)} \quad (5.5)$$

where  $E_w()$  is a mean operation,  $m_{w_{t_0}}$ ,  $\sigma_w^2(t_0)$  and  $m_{w_{t_k}}$ ,  $\sigma_w^2(t_k)$  are the means and variances of the normalized TOI at frame  $t_0$  and frame  $t_k$ , respectively. The higher the value of  $C_w(w_{t_0}, w_{t_k})$ , the lower the probability of the region being an active area. A TOI area showing a large difference from the initial frame (*i.e.*, low correlation value) is extracted as an active area. By doing this, a very limited number of active textures need updating, thereby reducing the amount of information that needs to be transmitted. Figure 5.6 shows the procedure of active texture detection.

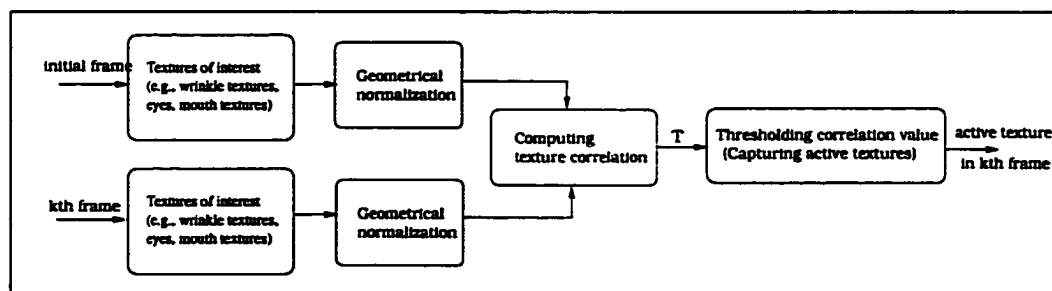


Figure 5.6: Diagram of active texture (AT) detection

## 5.4 Active texture compression

Each active area represents a class of image on a specific region of a face. When compressing such a clearly defined class of image (*e.g.*, active wrinkle textures, active mouth and eyes textures), all-purpose basis functions (such as FFT or DCT) are obviously not the best choice. A set of active textures of a person is a class of textures with the same statistical property, which in general represents a statistical resemblance determined by the action of different facial expressions. For example, normally the forehead wrinkle has horizontal folds, the crows-feet wrinkle has smile-shape oblique folds, the nasolabial wrinkle has parentheses-shape folds, etc. With these specific classes of textures, a transformation associated with a specific statistical property is obviously a good choice, *i.e.*, K-L transformation (principal component analysis). The principal component analysis method provides a more efficient and adaptive form to represent active textures, and therefore saves more bits while keeping the quality reasonable – as compared to DCT-based methods (such as JPEG, MPEG). By using this method, the output image can be described by the texture bases (called *eigen textures*). Since we only compute the textures of interest which have small areas, the computation load and storage capacity are greatly alleviated. This idea is similar to the strategy presented previously by Pentland [55], Strom [62] and Nishini [51] in which the whole face image is divided into blocks; however, our method has an even lower number of textures to compute.

We have defined nine active texture areas on a face. Suppose each area has  $M$  textures as an individual training set. First, the color texture image in RGB pixels with 24-bit depth is compressed into YUV format using 4:1:1 subsampling ( $U = C_r, V = C_b$ ). Each texture  $i$  in a training set is converted to a vector  $\mathbf{x}_i$ , with a dimension of  $1 \times 1.5N$ , by raster scanning the pixel values in each color band, as shown in Equa-

tion 5.6.  $N$  is the number of pixels in each texture.

$$\mathbf{x}_i = [x_{i,1}^Y, \dots, x_{i,N}^Y, x_{i,1}^U, \dots, x_{i,N/4}^U, x_{i,1}^V, \dots, x_{i,N/4}^V] \quad (5.6)$$

Then, the vectors in each training set forms an individual texture space  $\mathbf{x}$ ,

$$\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_M^T]^T \quad (5.7)$$

The mean texture of each space is computed from their own training set, *i.e.*,  $\mathbf{m}_\mathbf{x} = E\{\mathbf{x}\}$ , the average of the texture vectors in texture space  $\mathbf{x}$ . The transformation from a texture space  $\mathbf{x}$  into an eigen space  $\mathbf{y}$  is represented as follows:

$$\mathbf{y}_i = \mathbf{A}(\mathbf{x}_i - \mathbf{m}_\mathbf{x}) \quad (5.8)$$

where  $\mathbf{A}$  is a matrix whose rows are formed from the eigenvectors of  $\mathbf{C}_\mathbf{x}$ , ordered so that the first row of  $\mathbf{A}$  is the eigenvector corresponding to the largest eigenvalue, and the last row is the eigenvector corresponding to the smallest eigenvalue.  $\mathbf{C}_\mathbf{x}$  is a covariance matrix of the texture space  $\mathbf{x}$ , which is defined as:

$$\mathbf{C}_\mathbf{x} = E\{(\mathbf{x} - \mathbf{m}_\mathbf{x})(\mathbf{x} - \mathbf{m}_\mathbf{x})^T\} \quad (5.9)$$

After the K-L transform is performed,  $1.5N$  texture bases (eigen textures) are created and sorted in decreasing eigenvalue order. In order to reconstruct active texture with adequate accuracy, and efficiently compress the data,  $n$  texture bases ( $n \ll N$ ) are selected to form a transform matrix. The new texture to be encoded is projected onto the  $n$  texture bases. The active texture is now represented by  $n$  K-L coefficients and the corresponding set of texture bases. The  $n$  K-L coefficients can then be quantized to lower bit rates.

Because the transform matrix is derived from the statistical property of the texture, the population of the vectors in the training set directly affects the efficacy of the transformation. Increasing the size of the training set would increase the coding

efficiency and the image quality [62]. Therefore, besides inputting the active textures into the training set, we also add their mirror texture into it. For example, the left side of the nasolabial wrinkle texture would be more or less similar to the mirrored one of the right side, so the mirrored texture of the right side of the nasolabial wrinkle can be added to the training set for the left nasolabial wrinkle texture.

With the K-L transformation, the advantage of the partial texture update will be demonstrated in the later section with experiments.

## 5.5 Facial texture synthesis and expression generation

Since the facial texture for each frame is synthesized by the first frame image (for example, showing a natural expression), and the corresponding decoded active textures, simply texture-mapping is not good enough on a final appearance, especially on the boundary of the wrinkle texture; the sharp brightness transition is obviously seen. To overcome this drawback, a temporal blending method is applied to smooth the texture boundary, in which the brightness of the active texture is linearly blended with the image of the first frame on the boundary. The blending weight is in the range between 0 and 1, the value being a function of the pixel position in the active texture. In the large central area of the texture, the weight is set to 1. In the transition area the value decreases from 1 to approaching 0, as the texture gets closer to the boundary. The blended texture value is computed as follows:

$$b_k = (1 - w) \cdot f_1 + w \cdot f_k \quad (5.10)$$

$$w = \begin{cases} 1 & (x, y) \text{ within a central area;} \\ \frac{r}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x_c)^2 + (y-y_c)^2}{2\sigma^2}} + c & (x, y) \text{ within transition area.} \end{cases} \quad (5.11)$$

where  $b_k$  is the blended value of the  $k$ th frame active texture at the location  $(x, y)$ .  $f_1$  and  $f_k$  are the active texture values in frame 1 and frame  $k$ , respectively.  $w$  is



the blending weight as depicted in Figure 5.7.  $(x_e, y_e)$  is a variable position on the edge (borderline between the central area and the transition area).  $r$  is a factor for normalizing the values; here  $r$  is set as  $\sqrt{2\pi}\sigma$ . The response of the human visual

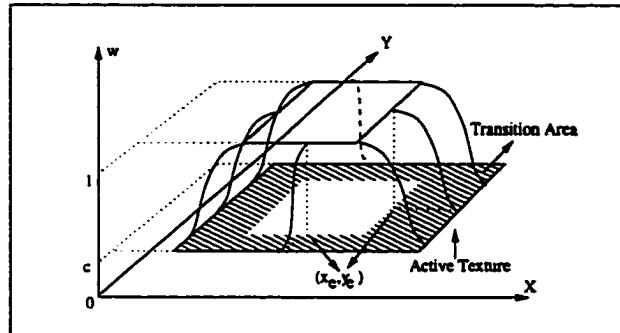


Figure 5.7: The weight value selection for texture blending

system tends to “overshoot” around the boundary of regions of different intensity. The result of this brightness perception is to make areas of constant intensity appear as if they had varying brightness. This is a well known *Mach-band effect*. To alleviate this effect, in the transition area the weight value is determined by the Gaussian distribution centered at variable positions of  $(x_e, y_e)$  to make texture fusion smoother and the intensity difference around the boundary region smaller. The variance  $\sigma$  and constant  $c$  are adjustable, and are set here as  $\sigma = 1.0$ ,  $c = 0$ . Moreover, a spatial low-pass filter with a template  $3*3$  is applied to smooth the border. This temporal blending plus spatial filtering process results in the two textures being synthesized in a visually smoother manner in our experiment. Some examples in close view of the blending effect are shown in Fig. 5.8. Figure 5.9 shows the sample results of 2D texture synthesis and 3D expression synthesis of whole face on frame 15 and frame 19.

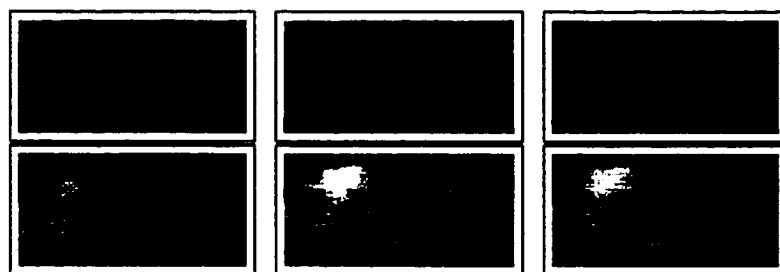


Figure 5.8: Close view of texture synthesis. (top) *Nasolabial*: (original frame25; no Blend(B)&Filter(F); with B&F). (bottom) *Forehead*: (original frame63; no B&F; with B&F).

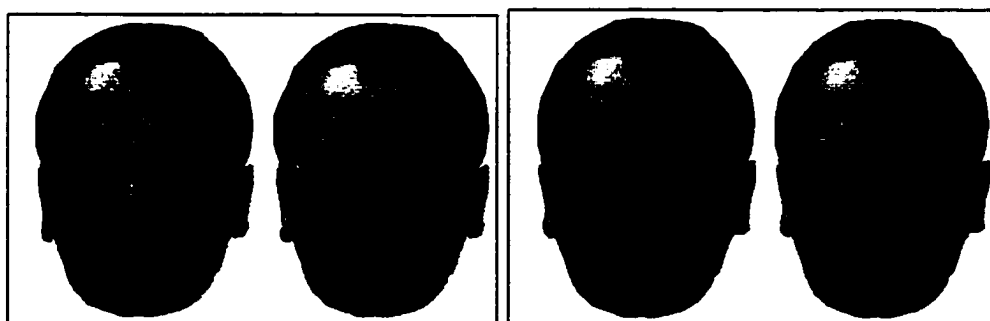


Figure 5.9: Expression synthesis using partial texture update and blending technique. Left two (frame 15 and frame 19): synthesized 2D facial texture using the 1st frame texture and the active textures of frame 15 and frame 19, respectively; Right two: texture-mapped 3D model using the left synthesized textures on frame 15 and frame 19.

## 5.6 Experimental results

Video sequences showing the different facial expressions of a person are the input for our experiment<sup>2</sup>. The resolution of each frame is  $550 \times 700$  pixels, the model resolution is 2954 vertices. Figure 5.10 (Row 1) shows the results of model adaptation using the extended dynamic mesh matching algorithm. Row 2 shows the original face textures. The results of face image normalization, and extracted textures of interest (*e.g.*, wrinkle, eye, mouth) are shown as polygons in Row 3. Figure 5.10 (Row 4) shows the group of textures of interest, which are geometrically normalized into the size of  $50 \times 90$  for forehead texture,  $30 \times 30$  for glabella texture,  $40 \times 40$  for both left and right crows-feet textures,  $80 \times 40$  for both left and right nasolabial textures,  $50 \times 80$  for mouth texture, and  $40 \times 80$  for both left and right eye textures.

To detect the active texture among the TOI, the correlation values of the TOI between the first frame and the subsequent frames are computed. The temporal correlation of six wrinkle textures and eye-mouth textures are changed along with the different expressions; the textures whose correlation values are below the threshold line are detected as active textures. The threshold for individual texture of interest is obtained by the statistical result, in which five sets of video sequences showing a person demonstrating different expressions, are used as the training set. We choose the average correlation value, which indicates the obvious change in the individual texture of interest, as a selected threshold for this type of texture. The threshold value obtained is 0.8 (forehead), 0.85 (glabella), 0.7 (crowsfeet), 0.5 (nasolabial), 0.45 (mouth), and 0.35 (eye). Figures 5.11 and 5.12 show examples of the active texture detection, in which an original video showing a person's different expressions is input to our system (as shown on the top row of the figure). The person's expression changes

---

<sup>2</sup>Please see <http://www.cs.ualberta.ca/~lijun/Demo.html> for color demo clips on various parts of our implementation [85] [84] [83] [81] [82].

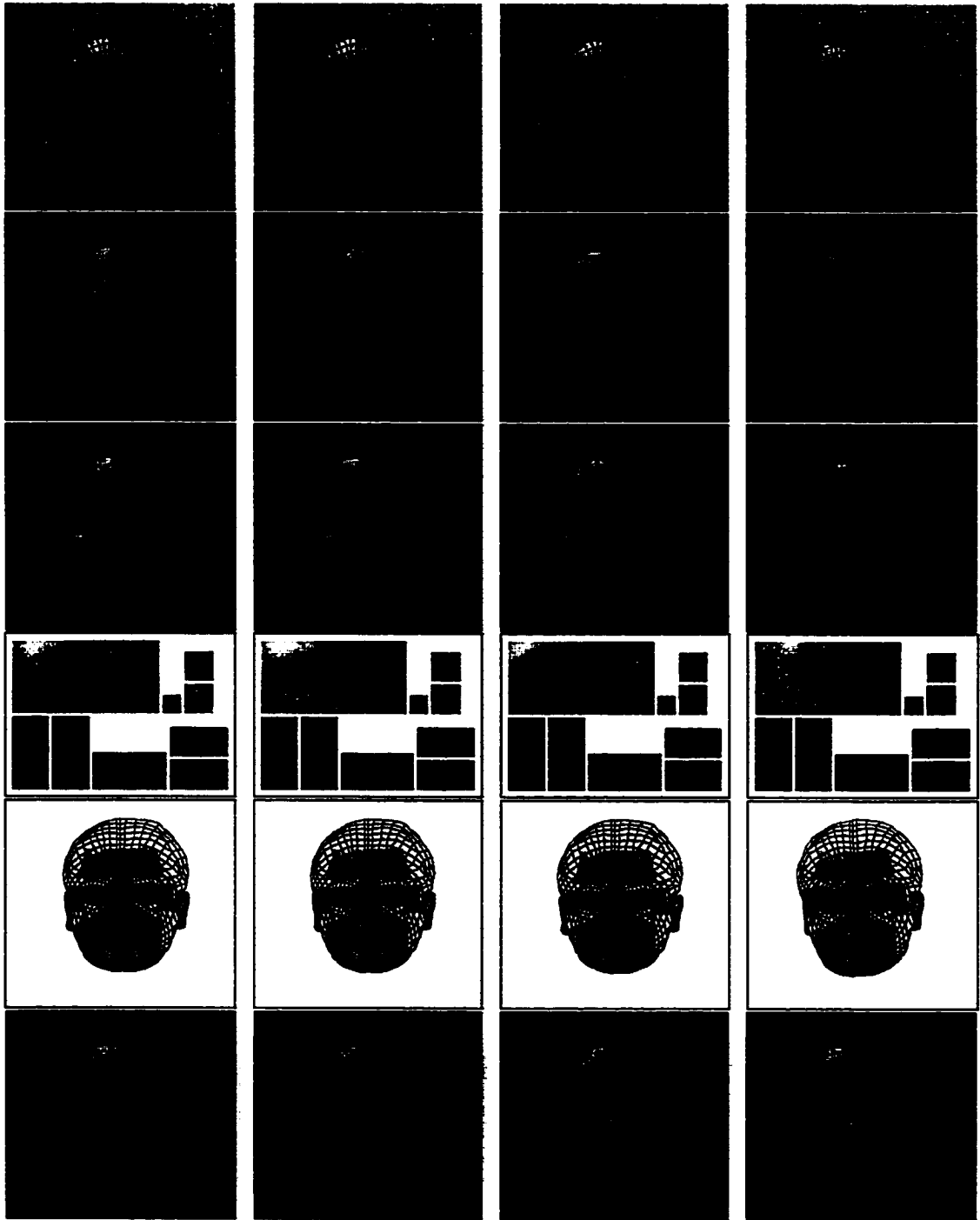


Figure 5.10: Facial texture synthesis: Row1: fitted model; Row2: original texture; Row3: normalized face texture; Row4: normalized textures of interest; Row5: updated textures of interest; Row6: synthesized face; (from left to right: frame 20, 32, 42, 62 of video Mario).

in several stages: *natural* → *eye-closing* → *smiling* → *laughing* → *back-to-natural* → *worried or sad* → *surprised*. In Chart I, the active texture of the forehead wrinkle is captured when the expression goes to "surprise", and the forehead horizontal-shape wrinkle is formed. In Chart II, the active glabella texture (in-between eyebrows) is detected when the "worry or sad" expression is demonstrated. In Chart III and IV, the curves of the texture correlation in the crowsfeet area (outer corners of eyes) and the nasolabial regions go down to the threshold value when the "smiling" and "laughing" expressions are shown, the eye-corner wrinkles with smile-shape oblique folds and mouth-nose side wrinkles with parentheses-shape folds are generated. Chart V indicates the active mouth textures detected when the mouth is opening to expose the teeth. Active eye textures are estimated by the correlation values lower than the threshold in Chart VI. There are three occasions showing the obvious change with eye textures, which are in the stages of "eye closing", "sad" and "surprised".

The low temporal correlation value distinguishes the active texture among the textures of interest. For example, the wrinkle corresponding to a lower value shows an obviously different wrinkle shape than the others. Hence, the correlation method performs well in detecting the active texture on an expressive face.

After obtaining the active textures on an individual face, the principal component transformation is applied to map the active texture into the eigen space. In our experiment, each active area has 200 textures as training vectors. Because we take into account only nine active areas, and each one has small dimensions, a small number of eigen textures is sufficient to describe the principal characteristics of each active texture, and to reconstruct the active texture with adequate fidelity. That is, 25 eigen textures (texture bases) are selected for forehead texture, nasolabial texture (both left side and right side), mouth texture, and eye texture (left side and right side), individually; 12 eigen textures are selected for crowsfeet texture (left and right) and

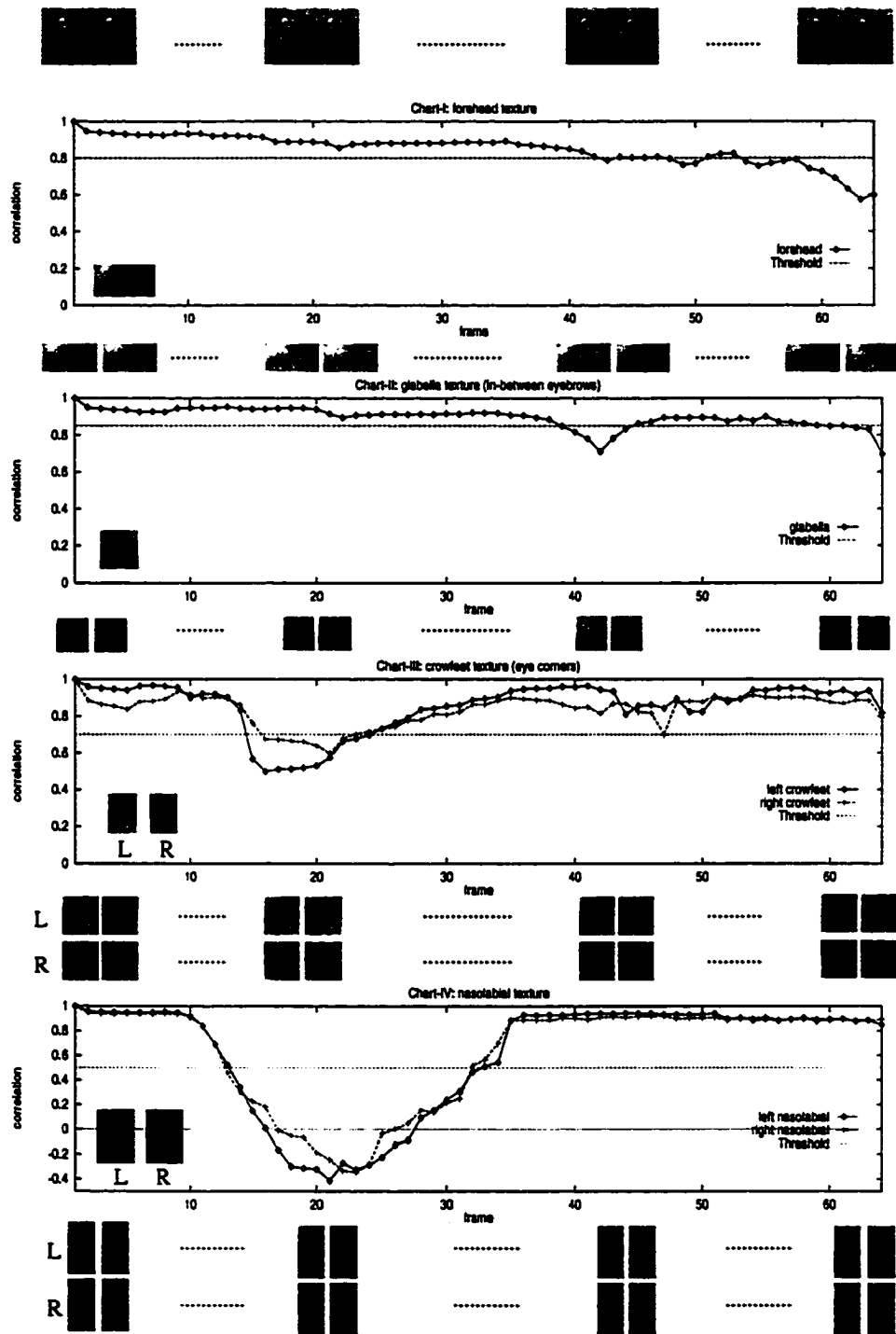


Figure 5.11: Chart of temporal correlation of wrinkle textures between the first frame (left-most face of the top row) and the subsequent frames. The input video shows a person's face demonstrating different expressions from left to right, *i.e.*, initial natural expression, eyes closing, smiling, laughing, worried and sad, surprised

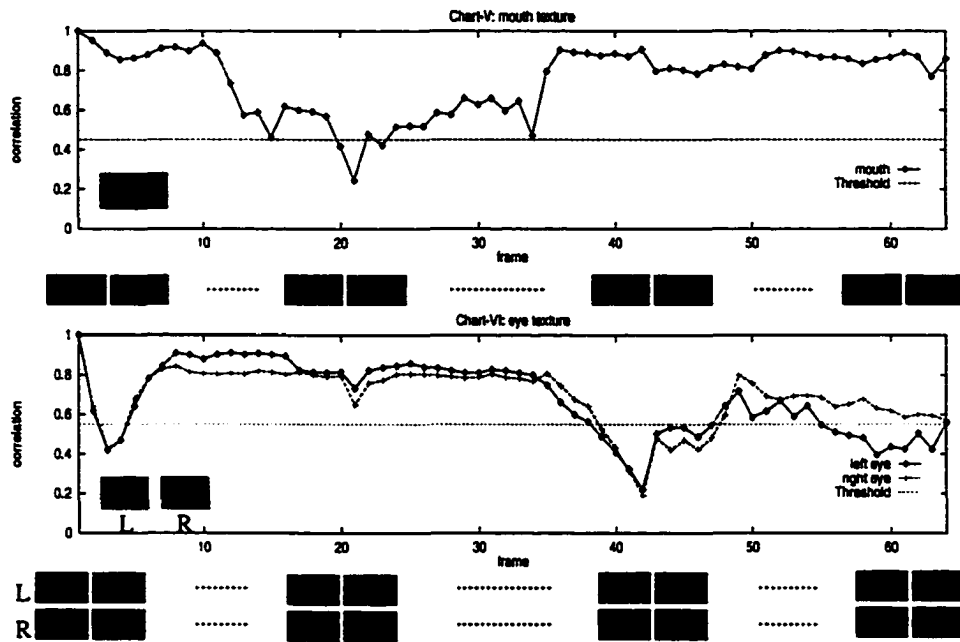


Figure 5.12: Chart of temporal correlation of mouth and eyes textures between the first frame and the subsequent frames. The input video shows a person's face demonstrating different expressions from left to right, *i.e.*, initial natural expression, eyes closing, smiling, laughing, worried and sad, surprised

glabella texture, respectively. Thus, if all nine TOI are detected to be active textures in a frame ( $550 \times 700$  pixels), only 186 data ( $= 25 \times 6 + 12 \times 3$ ) in maximum need to be transmitted, each datum can be quantized with 5bits; and the produced data for each frame is about 116 bytes, to achieve a SNR of 34.6dB. In the case of a frame rate of 30Hz, the bit cost for texture update is about 27kbit/s to the maximum. In an actual situation, the texture update does not occur on every TOI, nor on every frame. Therefore, the bit cost is much less than the above maximum estimation. If the animation parameters and geometric information are counted, the transmission rate will increase no more than 10kbit/s when using the "PCA+DCT" compression approach [66]. So, in total, the bit cost would be in a range of less than 40kbit/s. As compared with the full face texture update approach [62], (where the data encoded for each frame is about 1632 bytes in the case of  $600 \times 600$  pixels textures, and the quantizer with 5 bits per coefficient for yielding a SNR of 35.5 dB, the bit-rate is over 200kbit/s with 25 frame rate), the computation intensity is greatly reduced and the

<i>Property</i>	<i>Partial texture</i>	<i>Whole texture</i>
Quality	34.6dB	35.5dB
Subjective	4/5 (good)	not available
Compression	<40kbits/sec	>200kbits/sec
Computation	dimension (<50*90)	<600*600

Table 5.1: Comparison of the partial texture update and whole texture update

coding efficiency is obviously improved to a large degree, using our active texture update method. The comparison of the coding efficiency, the computation load, and the quality evaluation are illustrated in Table 5.1. Notice that a subjective evaluation is performed by 25 people, who were asked to do the rating after comparatively looking the original video sequences and the synthesized video sequences. The rate is ranked from 1 to 5, which corresponds to 1 (bad, totally different), 2 (poor, big different), 3 (fair, can tell some difference, but acceptable), 4 (good, very subtle different) and 5 (excellent, no different), respectively. After the test, 19 people rated as '4', 6 people rated as '5'. To synthesize the realistic facial expression, the decoded textures are repaired in the transition border area using the temporal blending technique, and finally the textures are mapped onto the individual models in the subsequent frames, as shown in Figure 5.10 (row 6). In comparison with the original video sequence, the recovered images have high visual fidelity, and the SNR (signal-noise ratio) values have over 34dB as measured by the ratio of the mean pixel value over the mean error value. It is obvious that the resulting synthesis, with texture updating and blending, is much better than the one without texture updating and blending (as seen in Figure 5.13 and Table 5.2 for comparison). Notice in this figure that the third face from the left, the one without texture update, does not seem to capture the smiling expression as accurately as the rightmost one – the one with texture update and blending. Also, patches around the cheeks are evident in the fourth face from the left, the one without blending.



<i>Scheme</i> (partial texture)	<i>Subjective</i> ( <i>Rating</i> )	<i>Objective</i> <i>SNR (dB)</i>
No update	2/5 (poor)	25.7 (average)
update/no blending	3/5 (fair)	33.1 (average)
update+blending	4/5 (good)	34.6 (average)

Table 5.2: Comparison of with-or-without texture update and blending

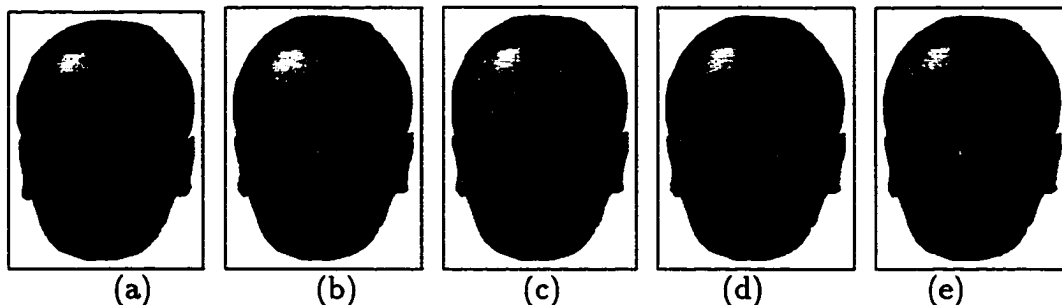


Figure 5.13: (from left to right) (a) Original 1st frame; (b) Original 15th frame; (c) Synthesized face (15th frame) without wrinkle texture update (only 1st frame texture used); (d) Synthesized face with wrinkle texture update but no blending of the edges; (e) Synthesized face (frame 15) with texture update and texture blending.

## 5.7 Discussion

In this chapter we proposed a face texture update and compression strategy for model based coding. The experimental results show that this strategy, using the partial wrinkle texture update method, has certain advantages over the entire texture update method; more specifically, the number of coefficients that need to be transmitted is reduced significantly from the original 1632 bytes per frame to the present 116 bytes per frame. Perceptually, the high fidelity of the image quality is sustained. Notice that the temporal correlation method to detect the active texture can be potentially used for facial expression recognition. We believe that the different composition of active textures in the temporal domain has a certain relationship with representation of different expressions. Further investigation into this relationship would be a promising direction for facial expression recognition, although this is not the purpose of the work presented in this chapter. In the future, a variable resolution model update strategy will be investigated to further improve the texture mapping results. To deal with the

arbitrary pose of head motion, multiple-view active textures detection and synthesis are also needed. In a real application, the computation load needs to be further reduced so that the system can be implemented in real time.

# Chapter 6

## Conclusions and future work

3-D model-based image sequence coding is a challenging research area, because it combines research efforts in Computer Vision, Image Coding and Computer Graphics. In this thesis, issues about realistic facial expression generation from real video sequences are addressed. Four major issues are investigated: 3-D face modeling, face feature extraction with active tracking, face model adaptation automatically, and face active texture synthesis with wrinkles. Several new approaches are proposed in these areas. The techniques have been applied successfully to each sub-problem, we have made the first step toward automatically synthesizing the *realistic* facial expressions through the video sequences. The schemes and algorithms presented here from different sub-problems impose fewer restrictions on the application than the methods reported in literature. The improvements include the following: no special markers on the face are required; active motion of the camera is allowed; there is no restriction on the background; and a higher compression ratio is achieved without greater loss of fidelity. Note that subtle texture information from the expressive face – such as wrinkles – are first investigated in our thesis. We made an important first step toward life-like real facial expression generation. However, our methods still pose some limitations for real applications. Following is a discussion of these limitations, as well as suggestions for future work.

## 6.1 Limitations of this thesis

### 1. Face modeling limitation

Our face modeling algorithm is working at the set-up stage (*i.e.*, off-line). This cannot satisfy the requirements of a real application. An on-line face modeling technique needs to be investigated so that the system is less constrained and more feasible. One way of resolving this problem would be to use *structure from motion* techniques to extract the 3-D structure of the face using only one camera. Another way is to use two or more cameras for an on-line stereo set-up.

### 2. Rigid motion and orientation

We have not concerned ourselves with large rigid motion (translation and rotation) of the head during expression analysis and synthesis. We have constrained our study to a talking face with front view in the video sequence. Only minimal motion is allowed. This strict assumption is not the case desired for a spontaneous and dynamic setting, and is not expected in a real conversation. In addition to large motions of a head, scaling (*i.e.*, moving towards or away from the camera) is also a major concern. Consider that some emotions, such as surprise, fear, and laughter, are normally accompanied by a backward and forward motion of the head, we need to deal with this type of motion. Therefore, it is necessary to address this issue of extracting the *large and free* global motion of the head. An interesting future extension would be the addition of either a feature-based or a template-based method to extract such global motions. Another way of resolving this problem is to use the simultaneous estimation of global rigid motion and local non-rigid motion using an affine model of motion and spatial-temporal constraints.

### 3. Extracting accurate shapes of the object regions

We have applied successfully the concept of *deformable templates* to extract the

shapes of the facial feature regions. However, this task becomes increasingly difficult when the complexity of the object increases. The decrease in performance is due either to a difference between the assumed image characteristics of the object region and the measured ones, or to the fact that neighboring object regions have similar image characteristics. Changes in scale, rotations, and lighting conditions or occlusions may cause a mismatch between the assumed image characteristics and the actual ones, and thus influence the performance of the shape extraction scheme. Therefore, a more flexible and adaptive template needs to be designed so that the multiple templates for one object region or a higher order template can be used for improving the performance of feature extraction.

#### **4. Real-time aspects**

When applying the proposed methods within a model-based coding scheme, the execution time becomes important. One of our future tasks is to study a more efficient method for achieving a fast procedure for model adaptation. Currently the two-step model adaptation algorithm is very time-consuming, especially in the course of fine-adaptation. One way to resolve this problem would be to simplify the model and use the technique of local surface fitting, instead of pixel-by-pixel movement. Another way to meet the real-time requirement is to parallelize the proposed adaptation methods. There are numerous possibilities for parallelization of the problems, *e.g.*, extracting the shape of different object regions, calculation of the different energy terms, or updating the equations of motion for each nodes. Additional research in this direction should reveal how much computation time can be gained by parallelization, and whether real-time implementation is conceivable.

#### **5. Realistic texture synthesis**

Currently, our texture synthesis technique is based on the video images. The

textures of interest are extracted, and the transmission of the texture information is greatly reduced. However, in order to further reduce the bit rates (*e.g.*, to less than 10kbits/s), a more complicated physics-based computer graphics technique needs to be investigated – in which the environment lighting and the physical property of skin texture will be explored for the skin texture synthesis with active wrinkles. Moreover, in order to deal with the case of various head poses, a multiple-view texture synthesis technique needs to be investigated as well.

## 6. Compressions of animation parameters

In this thesis, the issue of animation parameter compression is not addressed. There is some existing work aimed at the facial motion parameters compressions for MPEG-4, in which the PCA plus DCT method seems to be the best approach to deal with this issue as seen [66, 61]. Our future work in this area will be to integrate our system with the existing MPEG-4 algorithm for face animation parameter compression, and make our system compatible with the requirements of the MPEG4-SNHC specification.

## 6.2 Directions for Future Work

The technique proposed in this thesis has made a significant step towards an application of model-based coding in real situations. We have also shown that a number of fundamental problems are still unanswered. We believe that the most difficult problem is, and will continue to be, the accurate and robust measurement of object characteristics from image data *without* imposing restrictions on the object's appearance or on the imaging conditions. Application of this coding principle in a real, unrestricted environment is, therefore, expected to be part of our future work. Issues, such as the imaging condition, algorithm robustness, realistic synthesis, and real-time aspects, will be the major directions for further investigation. Only when remarkable breakthroughs have been achieved within the field of Computer Vision and

Graphics will these coding schemes begin to prove their right to existence. However, the proposed techniques can be applied for purposes which reach beyond efficient compression, such as computer-based film-making, developing new user interfaces, applications in cognitive research, identification systems, and teaching or speech aids.

After decades of effort, facial analysis and synthesis is still a challenging research topic. We hope this thesis will provide insight into some parts of the difficult problems.

# Bibliography

- [1] G.A. Abrantes and F. Pereira. An MPEG-4 SNHC compatible implementation of a 3D facial animation system. In *IWSNHC3DI'97, Rhodes, Greece*, pages 12–15, September 1997.
- [2] K. Aizawa, C.S. Choi, H. Harashima, and T.S. Huang. Human facial motion analysis and synthesis with application to model-based coding. In M. I. Sezan and R.L. Lagendijk, editors, *Motion Analysis and Image Sequence Processing*. Norwell, MA: Kluwer, 1993.
- [3] K. Aizawa, H. Harashima, and T. Saito. Model-based analysis-synthesis image coding (MBASIC) system for a person's face. *Signal Processing: Image Communication*, 10:138–152, October 1989.
- [4] K. Aizawa and T. Huang. Model-based image coding: Advanced video coding techniques for very low bit-rate application. *Proceedings of the IEEE*, 2(2):259–271, February 1995.
- [5] G. Bernd, P. Eisert, M. Magnor, E. Steinbach, and T. Wiegand. 3D Imaging and Compression - Synthetic, Hybrid or Natural Fit? In *International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI'99), Santorini, Greece*, September 1999.
- [6] S. Bernoegger, Lijun Yin, Anup Basu, and A. Pinz. Eye tracking and animation for MPEG-4 coding. In *ICPR'98: Proceedings 14<sup>th</sup> IAPR International Conference on Pattern Recognition, Australia*, volume II, pages 1281–1284, August 1998.
- [7] M. Bichsel. Strategies of robust object recognition for the automatic identification of human faces. *Ph.D. Thesis*, ETH Zurich, Zurich, 1991.
- [8] R. Brunelli and T. Poggio. Face recognition: features versus templates. *IEEE Trans. PAMI*, 15(10):1042–1052, October 1993.
- [9] T.C. Chang and T.S. Huang. Facial feature extraction from color images. In *ICPR'94: Proceedings 12<sup>th</sup> IAPR International Conference on Pattern Recognition, Jerusalem, Israel*, volume II, pages 39–43, October 9-13 1994.
- [10] Y. Chang and X. Li. Adaptive image region growing. *IEEE Transactions on Image Processing*, 3(6):868–872, November 1994.
- [11] Y. Chang and X. Li. Fast image region growing. *Image and Vision Computing*, 13(7):559–571, September 1995.



- [12] R. Chellappa, C. Wilson, and A. Sirohey. Human and machine recognition of faces: a survey. *Proceedings of the IEEE*, 83(5):705–740, May 1995.
- [13] H.H. Chen, T. Ebrahimi, G. Rajan, C. Horne, P.K. Doenges, and L. Chiariglione. Special issue on synthetic/natural hybrid video coding. *IEEE Trans. CAS for Video Tech.*, 9(2), March 1999.
- [14] C. Choi, K. Aizawa, H. Harashima, and T. Takebe. Analysis and synthesis of facial image sequences in model-based image coding. *IEEE Trans. CAS for Video Tech.*, 4(6):257–275, June 1994.
- [15] G. Chow and X. Li. Towards a system for automatic facial feature detection. *Pattern Recognition*, 26(12):1739–1755, December 1993.
- [16] T. Darrel, B. Moghaddam, and A. Pentland. Active face tracking and pose estimation in an interactive room. In *Proceedings of IEEE CVPR'96*, 1996.
- [17] E.R. Davies. A modified hough scheme for general circle location. *Pattern Recognition*, 7:37–43, January 1988.
- [18] D.J. Dawe. *Matrix and Finite Element Displacement Analysis of Structures*. Clarendon Press, Oxford, UK., 1984.
- [19] J.Y. Deng and F. Lai. Region-based template deformation and masking for eye-feature extraction and description. *Pattern Recognition*, 30(3):403–419, March 1997.
- [20] G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski. Classifying facial actions. *IEEE Trans. PAMI*, 21(10):974–989, October 1999.
- [21] P. Ekman and W. Friesen. *Facial Action Coding System*. New York: Consulting Psychologists Press, 1977.
- [22] P. Ellis. Recognizing faces. *British Journal of Psychology*, 66(4):409–426, 1975.
- [23] I. Essa, S. Basu, and A. Pentland. Motion regularization for model-based head tracking. In *ICPR'96: Proceedings 13<sup>th</sup> IAPR International Conference on Pattern Recognition, Vienna, Austria*, volume 3, pages 611–616. IEEE Computer Society Press, August 1996.
- [24] I. Essa and A. Pentland. Facial expression recognition using a dynamic model and motion energy. In *ICCV'95: Proceedings 5<sup>th</sup> International Conference on Computer Vision, MIT, Cambridge, MA*, pages 360–367. IEEE Computer Society Press, June 1995.
- [25] I. A. Essa and A. P. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. *IEEE Trans. PAMI*, 19(7):757–763, July 1997.
- [26] C. Gu and M. C. Lee. Semiautomatic segmentation and tracking of semantic video objects. *IEEE Trans. CAS for Video Tech.*, 8(5):572–584, September 1998.
- [27] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *SIGGRAPH'98, Orlando, FL.*, pages 55–66, July 1998.

- [28] B. Haskell, P. Howarda, Y. LeCun, A. Puri, J. Ostermann, M. Civanlar, L. Rabiner, L. Bottou, and P. Haffner. Image and video coding – Emerging standards and beyond. *IEEE Trans. CAS for Video Tech.*, 8(7):814–837, November 1998.
- [29] T. Huang and A. Netravali. Motion and structure from feature correspondences: a review. *Proceedings of the IEEE*, 82(2):252–268, 1994.
- [30] W. Huang and D. Goldgof. Adaptive-size meshes for rigid and non-rigid shape analysis and synthesis. *IEEE Trans. PAMI*, 15(6):611–616, June 1993.
- [31] Anil K. Jain. *Fundamentals of digital Image Processing*. Prentice Hall Information and system science series, 1989.
- [32] T. Kanade. Picture processing by computer complex and recognition of human faces. *Tech. Rep.*, Kyoto Univ., Dept. Inform. Sci., 1973.
- [33] P. Kierkegaard. A method for detection of circular arcs based on the hough transform. *Machine Vision and Applications*, 5:249–263, 1992.
- [34] R. Koenen, F. Pereira, and L. Chiariglione. MPEG-4: Context and objectives. *Signal Processing: Image Communication*, 9(4):295–304, May 1997.
- [35] K. Komatsu. Surface model of face for animation. *Trans. IPSJ*, May 1996.
- [36] I. Kompatsiaris, D. Tzovaras, and M.G. Strintzis. 3-D model-based segmentation of videoconference image sequences. *IEEE Trans. CAS for Video Tech.*, 8(5):547–561, September 1998.
- [37] Y. Kunita, M. Inami, T. Maeda, and S. Tachi. Real-time rendering system of moving objects. In *Proceedings of IEEE Computer Society Workshop on Multi-View Modeling & Analysis of Visual Scenes*, pages 81–88, Fort Lollins, Colorado, June 1999.
- [38] T. Kurihara and K. Arai. A transformation method for modeling and animation of the human face from photographs. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Computer Animation'91*, pages 45–58. Springer-Verlag, 1991.
- [39] H. Li and R. Forchheimer. 3-D motion estimation in model-based facial image coding. *IEEE Trans. PAMI*, 15(6):545–555, June 1993.
- [40] H. Li and R. Forchheimer. Two-view facial movement estimation. *IEEE Transactions on Circuits and System for Video Technology*, 4(6):276–287, June 1994.
- [41] H. Li, A. Lundmark, and R. Forchheimer. Image sequence coding at very low bitrates: A review. *IEEE Transactions on Image Processing*, 3(5):589–609, September 1994.
- [42] Y. Li and Y. Chen. A hybrid model-based image coding system for very low bit-rate coding. *IEEE Journal on Selected Areas in Communications*, 16(1):28–42, January 1998.
- [43] K. Mase. Recognition of facial expression from optical flow. *IEICE Transactions*, E74(10):3474–3483, October 1991.
- [44] T. Meier and K. N. Ngan. Automatic segmentation of moving objects for video object plane generation. *IEEE Trans. CAS for Video Tech.*, 8(5):525–538, September 1998.

- [45] F. Moscheni, S. Bhattacharjee, and M. Kunt. Spatiotemporal segmentation based on region merging. *IEEE Trans. PAMI*, 20(9):897–915, September 1998.
- [46] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Trans. PAMI*, 16(5):449–459, May 1994.
- [47] ISO/IEC JTC1/SC29/WG11 N2205. Text of IS 14496-5 (MPEG 4 simulation) committee draft. *International Organization for Standardization, Coding of Moving Picture and Audio, Working Group 29, N2205*, March 1998.
- [48] Y. Nagashima, H. Agawa, and F. Kishino. 3D face model reproduction method using multi view images. In *Proceedings SPIE Visual Communication and Image Processing, Boston, Massachusetts*, volume 1606, pages 566–573. SPIE, 1991.
- [49] M. Nahas, H. Huitric, M. Rioux, and J. Domey. Facial image synthesis using skin texture recording. *The Visual Computer—The International Journal of Computer Graphics*, 6:337–343, June 1990.
- [50] Y. Nakaya and H. Harashima. Model-based/Waveform hybrid coding for low-rate transmission of facial images. *IEICE Transactions on Communication*, E75-B(5):377–384, May 1992.
- [51] K. Nishini, Y. Sato, and K. Ikeuchi. Eigen-texture method: Appearance compression based on 3D model. In *CVPR'99*, pages 618–624, Fort Lollins, Colorado, June 1999.
- [52] F. Parke. Parameterized models for facial animation. *IEEE Computer Graphics and Applications*, 2(9):61–68, November 1982.
- [53] F. Parke and K. Waters. *Computer Facial Animation*. A K Peters, Wellesley, MA, 1996.
- [54] D.E. Pearson. Developments in model-based video coding. *Proceedings of the IEEE*, 83(6):892–906, June 1995.
- [55] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1994.
- [56] F. Pighin, J. Hecker, D. Linchinski, R. Szeliski, and D. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH'98, Orlando, FL.*, pages 75–84, July 1998.
- [57] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C; The Art of Scientific Computing*. Cambridge University Press, 1988.
- [58] M. Reinders, P. Beek, B. Sankur, and J. Lubbe. Facial feature localization and adaptation of a generic face model for model-based coding. *Signal Processing: Image Communication*, 7:57–74, July 1995.
- [59] MPEG Requirements. Overview of MPEG-4 profiles and levels. *ISO/JTC1/SC29/WG11 N2325, Dublin MPEG Meeting*, July 1998.
- [60] A. Samal and P.A. Iyengar. Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern Recognition*, 25:65–77, January 1992.

- [61] MPEG Video & SNHC. Text of ISO/IEC FDIS 14496-2: visual, Doc. ISO/IEC JTC1/SC29/WG11 MPEG N2501. *Proc. Atlantic City MPEG Meeting*, October 1998.
- [62] J. Strom, F. Davoine, J. Ahlberg, H. Li, and R. Forchheimer. Very low bit rate facial texture coding. In *IWSNHC9DI'97, Rhodes, Greece*, pages 237–240, September 1997.
- [63] L. Strub. Automated facial conformation for model-based videophone coding. In *IEEE ICIP*, October 1995.
- [64] Y. Suenaga and Y. Watanabe. A method for the synchronized acquisition of cylindrical range and color data. In *IAPR Workshop on MVA*, November 1990.
- [65] MPEG Systems. Text of ISO/IEC FDIS 14496-1: systems, Doc. ISO/IEC JTC1/SC29/WG11 MPEG N2501. *Proc. Atlantic City MPEG Meeting*, October 1998.
- [66] H. Tao, H. Chen, W. Wu, and T. Huang. Compression of MPEG4 facial animation parameters for transmission of talking heads. *IEEE Trans. CAS for Video Tech.*, 9(2):264–276, March 1999.
- [67] H. Tao and T. Huang. Explanation-based facial motion tracking using a piecewise bézier volume deformation model. In *CVPR'99*, pages 611–617, Fort Lollins, Colorado, June 1999.
- [68] D. Terzopoulos and M. Vasilescu. Sampling and reconstruction with adaptive meshes. In *CVPR'91*, pages 70–75, 1991.
- [69] D. Terzopoulos and M. Vasilescu. Adaptive mesh and shells: irregular triangulation, discontinuities and hierarchical subdivision. In *Proceedings of IEEE CVPR'92*, pages 829–832, 1992.
- [70] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Trans. PAMI*, 15(6):569–579, June 1993.
- [71] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuro science*, 3(1):71–86, 1991.
- [72] M. Vannier, T. Pilgram, G. Bhatia, and B. Brundsen. Facial surface scanner. *IEEE Computer Graphics and Applications*, 4:72–80, April 1991.
- [73] MPEG Video&SNHC. Final text for FCD 14496-2:visual. *Doc. ISO/MPEG N2202, Tokyo MPEG Meeting*, March 1998.
- [74] J. Waite and W. J. Welsh. Head boundary location using snakes. *British Telecom Technology Journal*, 8(3):127–136, 1990.
- [75] W. J. Welsh. Model-based coding of video-phone images. *Electronics and Communication Engineering Journal*, pages 29–36, 1991.
- [76] L. Williams. 3D paint. *Computer Graphics*, 24(2):225–233, March 1990.
- [77] H. Wu, T. Yokoyama, D. Pramadihanto, and M. Yachida. Face and facial feature extraction from color image. In *ICAFGR'96: Proceedings 2<sup>nd</sup> IEEE International Conference on Automatic Face and Gesture Recognition, Killington, Vermont, USA*, p345-350, Oct. 1996.

- [78] G. Xu. Three-dimension face modeling for virtual space teleconferencing systems. *Trans. IEICE*, October 1990.
- [79] Y. Yacoob and L. S. Davis. Recognizing human facial expressions from long image sequences using optical flow. *IEEE Trans. PAMI*, 18(6):636–642, June 1996.
- [80] Lijun Yin and Anup Basu. MPEG-4 face modeling using fiducial points. In *Proceedings of IEEE International Conference on Image Processing, Santa Barbara, CA*, pages 109–112, October 1997.
- [81] Lijun Yin and Anup Basu. Integrating active face tracking with model based coding. *Pattern Recognition Letters*, 20(6):651–657, 1999.
- [82] Lijun Yin and Anup Basu. Realistic animation using extended adaptive mesh for model based coding. In *2nd International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR'99), Springer-Verlag Lecture Notes Series on Computer Science. York, UK.*, pages 189–201, July 1999.
- [83] Lijun Yin and Anup Basu. Synthesizing a realistic facial animation using energy minimization for model based coding. *accepted by Pattern Recognition*, 2000.
- [84] Lijun Yin and Anup Basu. Texture decomposition and correlation thresholding for realistic low-bitrate model-based coding. In *IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP'2000)*, pages 1943–1946, June 2000, Istanbul, Turkey.
- [85] Lijun Yin and Anup Basu. Partial update of active textures for efficient expression synthesis in model-based coding. In *IEEE International Conference on Multimedia and Exposition (ICME'2000)*, July 2000. New York City, NY. (IEEE Computer Society, Circuit and System Society, Communications Society and Signal Processing Society).
- [86] A.L. Yuille, P.W. Hallinan, and D.S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.
- [87] L. Zhang. Automatic adaptation of a face model using action units for semantic coding of videophone sequences. *IEEE Trans. CAS for Video Tech.*, 8(6):781–795, October 1998.
- [88] Y.Q. Zhang, F. Pereira, T. Sikora, and C. Reader. Special issue on MPEG-4. *IEEE Trans. CAS for Video Tech.*, 7, February 1997.

# Appendix A

## Proof of pin-hole camera model of Section 3.2.1

### Proof:

As shown in Figure 3.2, assuming perspective projection, the relationships between points in the image plane and points in 3-D with respect to the camera coordinate system are:

$$x = f \frac{X}{Z} \quad (\text{A.1})$$

$$y = f \frac{Y}{Z} \quad (\text{A.2})$$

where  $P(X,Y,Z)$  is a 3-D location of a point in the camera coordinate system, and  $p(x,y)$  is its projection on the image plane;  $f$  is the focal length.

Consider a camera rotation around  $O$  (lens center),  $\alpha$  is a small angle of rotation about the pan axis ( $Y$ -axis);  $\gamma$  is a small angle of rotation about the tilt axis ( $X$ -axis). A rotation of the camera by  $R$  can be regarded as the rotation of the scene in the opposite sense. If the scene is rotated by  $R^{-1}$  ( $R^{-1} = R^T$ ), where  $T$  denotes transpose, point  $P = (X, Y, Z)$  moves to point  $P' = (X', Y', Z')$ , as shown in below,

$$\begin{bmatrix} (P')^T \\ 1 \end{bmatrix} = R^T \begin{bmatrix} (P)^T \\ 1 \end{bmatrix} \quad (\text{A.3})$$

where  $R^T$  consists of two rotations, which are pan rotation  $R_Y(\alpha)$  and tilt rotation  $R_X(\gamma)$ ,  $R^T = R_Y(\alpha)R_X(\gamma)$ ,

$$R_Y(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A.4})$$

$$R_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) & 0 \\ 0 & \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A.5})$$

Assuming a small rotational angle between consecutive frames, we use the approximations

$$\cos(\gamma) \approx 1, \cos(\alpha) \approx 1, \sin(\gamma) \approx \gamma, \sin(\alpha) \approx \alpha \quad (\text{A.6})$$

then

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} X + \alpha\gamma Y + \alpha Z \\ Y - \gamma Z \\ -\alpha X + \gamma Y + Z \\ 1 \end{bmatrix} \quad (\text{A.7})$$

Since the point  $P'$  is projected to  $(x', y')$  on the image plane, where

$$x' = f \frac{X'}{Z'} \quad (\text{A.8})$$

$$y' = f \frac{Y'}{Z'} \quad (\text{A.9})$$

Combining this with (A.7), (A.1) and (A.2), the relationships between  $(x', y')$  and  $(x, y)$  can be established by Equation (A.10) and (A.11).

$$x' = f \frac{x + \alpha\gamma y + \alpha f}{-\alpha x + \gamma y + f} \quad (\text{A.10})$$

$$y' = f \frac{y - \gamma f}{-\alpha x + \gamma y + f} \quad (\text{A.11})$$

# Appendix B

## Proof of pan-tilt camera model of Section 3.2.1

### Proof:

In the case of pan/tilt model, the origin of the camera coordinate system is located at the lens center, the origin of the device coordinate system is at the device center. If we define a reference frame as the intersecting axes of rotation (device Y-axis and X-axis), whose origin is located at the device center, any 3D point in the reference frame can be expressed as a vector  $P_r$ , which has the following relationship with its position in the camera frame by a homogeneous transformation  $T_c$ ,

$$P_r = T_c P_c \quad (\text{B.1})$$

where  $P_c$  is the 3D point in the camera frame, whose origin is located at the lens center.  $T_c$  is the  $4 \times 4$  transform matrix relating the camera frame to the reference frame. For an arbitrary initial tilt angle  $\theta$ , the camera transformation  $T_c$  is

$$T_{c|\theta} = R_X(\theta) T_{c|\theta=0} \quad (\text{B.2})$$

where

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{B.3})$$



$$T_{c|\theta=0} = \begin{bmatrix} 1 & 0 & 0 & d_X \\ 0 & 1 & 0 & d_Y \\ 0 & 0 & 1 & d_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{B.4})$$

The current camera frame,  $T_c(t)$ , is the result of a pan/tilt rotation ( $\alpha$  and  $\gamma$ ) of the previous camera position  $T_c(t-1)$ ,

$$T_c(t) = R_Y(\alpha)R_X(\gamma)T_c(t-1) \quad (\text{B.5})$$

Any point in the reference frame can be represented in terms of camera frames before and after motion, such as

$$P_r = T_c(t)P_c(t) = T_c(t-1)P_c(t-1) \quad (\text{B.6})$$

According to the above relationship, the position of a point in one camera frame can be derived by its position in the other as:

$$P_c(t-1) = T_c(t-1)^{-1}T_c(t)P_c(t) \quad (\text{B.7})$$

Combine Equation B.5 and B.7, we obtain

$$P_c(t-1) = T_c(t)^{-1}R_Y(\alpha)R_X(\gamma)T_c(t)P_c(t) \quad (\text{B.8})$$

Assuming a small rotational angle between consecutive frames, we use the approximations

$$\cos(\gamma) \approx 1, \cos(\alpha) \approx 1, \sin(\gamma) \approx \gamma, \sin(\alpha) \approx \alpha \quad (\text{B.9})$$

From above Equation B.2 to Equation B.9, the relationship between every pixel position in two consecutive images can be derived as follows [46]:

$$x_{t-1} = f \frac{x_t + \alpha \sin(\theta)y_t + f \alpha \cos(\theta) + \frac{f d_Z}{Z_t}}{-\alpha \cos(\theta)x_t + \gamma y_t + f + f \frac{\gamma \cos(\theta)d_Y - \gamma \sin(\theta)d_Z}{Z_t}} \quad (\text{B.10})$$

$$y_{t-1} = f \frac{-\alpha \sin(\theta)x_t + y_t - f \gamma + \frac{\gamma \sin(\theta)d_Y - \gamma \cos(\theta)d_Z}{Z_t}}{-\alpha \cos(\theta)x_t + \gamma y_t + f + f \frac{\gamma \cos(\theta)d_Y - \gamma \sin(\theta)d_Z}{Z_t}} \quad (\text{B.11})$$

Assuming the depth is large compared to the other parameters, we can neglect the last terms of both numerator and denominator and obtain following approximate relationship:

$$x_{t-1} = f \frac{x_t + \alpha \sin \theta y_t + f \alpha \cos \theta}{-\alpha \cos \theta x_t + \gamma y_t + f} \quad (\text{B.12})$$

$$y_{t-1} = f \frac{-\alpha \sin \theta x_t + y_t - f \gamma}{-\alpha \cos \theta x_t + \gamma y_t + f} \quad (\text{B.13})$$

where  $f$  is the focal length. With knowledge of  $f$ , initial inclination  $\theta$ , and pan/tilt rotation  $\gamma$  and  $\alpha$ , for every pixel position  $(x_t, y_t)$  in the current image we can calculate the position  $(x_{t-1}, y_{t-1})$  of the corresponding pixel in the previous image.

## Appendix C

### Sample frames of eight video sequences of Table 3.1



Figure C.1: “Mario” (550\*700 pixels). left: original frame; right: feature shape detected



Figure C.2: “Guan” (480\*640 pixels). left: original frame; right: feature shape detected

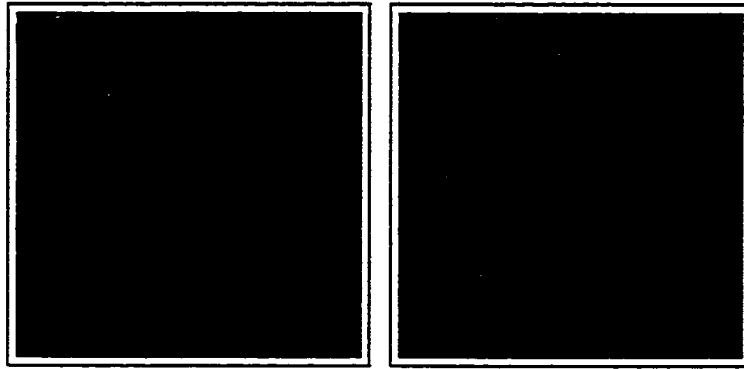


Figure C.3: "Dima" (480\*480 pixels). left: original frame; right: feature shape detected

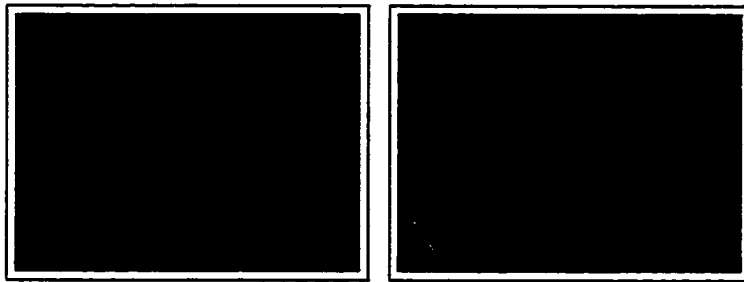


Figure C.4: "Stan" (480\*640 pixels). left: original frame; right: feature shape detected

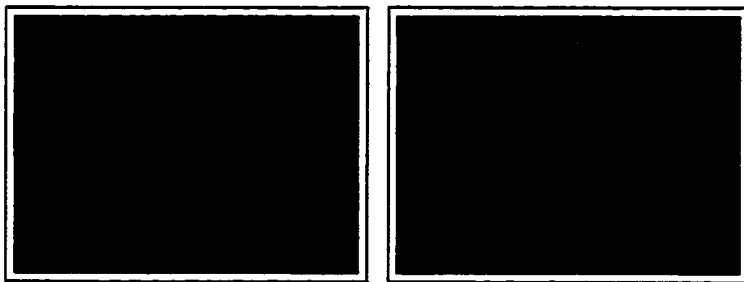


Figure C.5: "Dana" (480\*640 pixels). left: original frame; right: feature shape detected

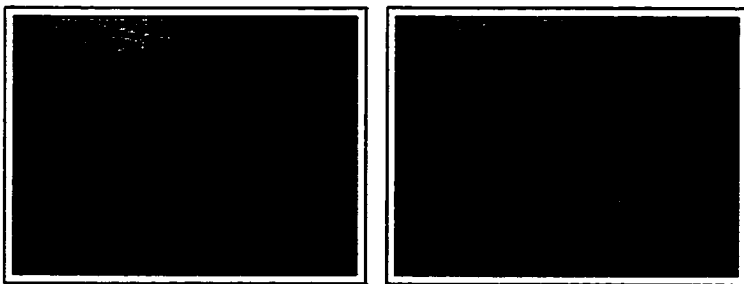


Figure C.6: "Xun" (480\*640 pixels). left: original frame; right: feature shape detected

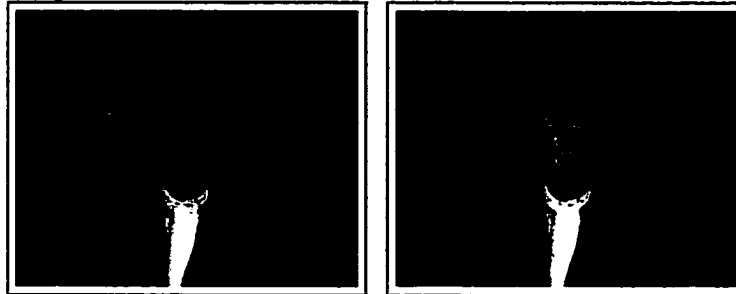


Figure C.7: "Claire" (288\*360 pixels). left: original frame; right: feature shape detected

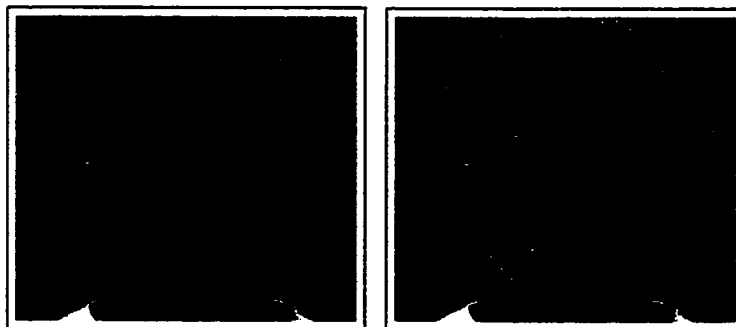


Figure C.8: "Alau" (512\*512 pixels). left: original frame; right: feature shape detected