Efficient techniques for face and eye detection

Deng, Xiaoyun

2005

Deng, X. (2005). Efficient techniques for face and eye detection. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46875>

<https://doi.org/10.32657/10356/46875>

Nanyang Technological University

Downloaded on 26 Apr 2023 13:17:30 SGT

Efficient Techniques for Face and Eye Detection

Deng Xiaoyun

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2005

TA
1650
D391
2005

Acknowledgement

The author would like to thank her supervisor, Assistant Professor Chang Chip Hong, for his initiated ideas, helpful suggestions, and consistent encouragement throughout the project.

The author also appreciates Mr. Erwin Braendle, Professor Guido M. Schuster, and the exchange students from *HSR Hochschule für Technik Rapperswil (HSR University of Applied Sciences)*, Switzerland, for their active participation and valuable contributions to the project.

The author is also grateful to the staffs and students at the *Center for High Performance Embedded Systems (CHiPES)* for their warm support and friendship that makes the center a pleasant working environment.

Table of Contents

ACKNOWLEDGEMENT i

TABLE OF CONTENTS..... ii

LIST OF FIGURES v

LIST OF TABLES vii

SUMMARY viii

CHAPTER 1 INTRODUCTION 1

 1.1 Motivation 1

 1.2 Objectives 4

 1.3 Major Contributions of the Thesis 4

 1.4 Organization of the Thesis 6

CHAPTER 2 LITERATURE REVIEW 8

 2.1 Existing Face Detection Methods 8

 2.1.1 Skin Color Detection..... 9

 2.1.2 Use of Texture..... 11

 2.1.3 Use of Shape 11

 2.2 Existing Eye Detection Techniques 12

 2.2.1 Knowledge-Based Approaches for Eye Detection 12

 2.2.2 Appearance-Based Approaches for Eye Detection..... 14

 2.3 Color Image Segmentation 14

 2.4 SOM..... 16

CHAPTER 3 SKIN COLOR DETECTION 19

 3.1 Discussion on Color Spaces and Classifiers 20

 3.2 Theory of SOTT 24

 3.3.1 Basic Self-Organizing Map (SOM) 24

3.3.2 Butterfly Permutation for Input Randomization	26
3.3.3 SOTT Algorithm	27
3.3 SOTT for Skin Color Detection	30
3.3.1 Scheme 1 – Fully Adaptive Scheme	30
3.3.2 Scheme 2 – Semi Adaptive Scheme	32
3.3.3 Scheme 3 – SOTT-Bayesian Scheme	32
3.4 Comparison of SOTT and Other Important Skin Color Classifiers	34
3.5 Summary	36
CHAPTER 4 ROI EXTRACTION AND VERIFICATION	37
4.1 Adaptive Center of Mass and Window Growing	37
4.1.1 Adaptive Center of Mass	37
4.1.2 Window Growing	43
4.1.3 Results and Discussions on Adaptive Center of Mass and Window Growing	49
4.2 Histogram Projection of Object Features	52
4.2.1 Selection of the Object Features	52
4.2.2 Locating Eyes by Projecting Their Edge Features	54
4.2.3 Results of Histogram Projection for Eye Detection	58
4.3 Principal Component Analysis for Template Matching	59
4.3.1 Computation of Eigeneyes	60
4.3.2 Verification of Eye	63
4.3.3 Selection of the Templates	65
4.3.4 Results of PCA	66
4.4 Summary	76
CHAPTER 5 TEST BED FOR APPLICATION DEVELOPMENT	77
5.1 Applied Technologies	78
5.1.1 TIFF	78
5.1.2 RAW File Format	78
5.1.3 DLL	79

5.2 GUI	80
5.3 Algorithms	82
5.3.1 Loading Files	82
5.3.2 Pre-Processing.....	83
5.3.3 Skin Color Classifiers	83
5.3.4 Face Extraction	86
5.3.5 Eye Detection.....	86
5.3.6 Post-Processing	93
5.4 System Evaluation	93
5.5 Summary	97
 CHAPTER 6 CONCLUSION AND RECOMMENDATIONS	98
6.1 Conclusion	98
6.2 Recommendations for Further Research.....	100
 APPENDICES	102
Appendix A Image Databases for Experiments and Benchmarking	102
A.1 AR Database	102
A.2 ECU Database	103
Appendix B TIFF Structure and Semantics	106
B.1 TIFF Structure	106
B.2 Bi-level Images	108
B.3 Grayscale Images	109
Appendix C GUI User Handbook.....	110
C.1 System Requirements	110
C.2 Installing the Application	113
C.3 How to create a DLL.....	114
 AUTHOR'S PUBLICATIONS	102
 BIBLIOGRAPHY	116

List of Figures

Figure 1.1 Text entry system ‘Dasher’	2
Figure 1.2 Off-road simulated drivers’ eye behavior.....	3
Figure 3.1 Distribution of skin colors (a) In RGB (b) In YCbCr	21
Figure 3.2 Distribution of skin (red) and non-skin (green) colors, and the probability of being skin color (blue) (a) In RGB (b) In YCbCr.....	21
Figure 3.3 ROC curves for Bayesian classifiers	23
Figure 3.4 SOM training algorithm	24
Figure 3.5 Butterfly permutation order for input of a 2-D coordinate system.....	27
Figure 3.6 2D and 1D SOTT structure.....	27
Figure 3.7 Filtered index maps incorporating spatial coordinates with weight $[Cb, Cr, x, y]$ of (a) $[255, 255, 0, 0]$; (b) $[255, 255, 31, 31]$; (c) $[255, 255, 63, 63]$	31
Figure 3.8 Codebook obtained after SOTT training	33
Figure 3.9 Comparison of skin color classifiers in their performances	34
Figure 4.1 Locating CoM in face detection application	39
Figure 4.2 Locating CoM in eye detection application.....	40
Figure 4.3 Adaptive CoM on image consisting of more than one faces.....	41
Figure 4.4 Pseudo code for Window Growing function.....	45
Figure 4.5 Window Growing process in eye extraction application.....	45
Figure 4.6 Cropped eye windows by Window Growing	46
(a) Original images	47
(b) Adaptive CoM on skin probability maps	47
(c) Binary skin images	48
(d) Cropped face windows.....	48
Figure 4.7 Results of Adaptive CoM and Window Growing for face detection	48
Figure 4.8 Sobel operator.....	53
Figure 4.9 Prewitt operator	53
Figure 4.10 Horizontal projections of G_x (vertical edges) for upright facial images.....	55

Figure 4.11 Horizontal projections of vertical edges and intensity of the same subject without and with the teeth shown	56
Figure 4.12 Vertical projections of horizontal edges for eye regions.....	57
Figure 4.13 Training set of eye images.....	60
Figure 4.14 Average intensity eye image	61
Figure 4.15 The first 30 eigeneyes.....	62
Figure 4.16 Test, average, projected and approximated eye images (from top to bottom)	64
Figure 4.17 Detected left eye images of women.....	67
Figure 4.18 Detected right eye images of women	68
Figure 4.19 Detected left eye images of men	69
Figure 4.20 Detected right eye images of men	70
Figure 4.21 Erroneously cropped eye regions	74
Figure 4.22 Poorly segmented eyes	74
Figure 4.23 Three eye regions for 6 previously failed subjects.....	75
Figure 5.1 Sample code for using DLL in C#.....	80
Figure 5.2 GUI of Eyetracking	81
Figure 5.3 Pseudo code: adaptive_threshold().....	87
Figure 5.4 Pseudo code: region_size_filter()	88
Figure 5.5 Pseudo code: region_growing().....	89
Figure 5.6 Between-region parameters	90
Figure 5.7 Pseudo code: region_distance_filter()	91
Figure 5.8 Structure of regions_ram[][]	92
Figure 5.9 Pseudo code: get_regions().....	92
Figure 5.10 Effect of distance to the camera	95
Figure 5.11 Effect of in-plane rotation	96
Figure 5.12 Effect of up and down rotation of the face	97
Figure 6.1 Three types of head rotations	100
Figure B.1 Structure of a TIFF image file	107

List of Tables

Table 3.1 Comparison of color classifiers in terms of speed and memory requirements. 35

Table 4.1 Results of Adaptive CoM and Window Growing for face detection..... 49

Table 4.2 Summary of other face detectors’ performance..... 51

Table 4.3 Failure rates for all images tested 58

Table 4.4 Failure rates for images without glasses 58

Table 4.5 The first 20 eigenvalues 62

Table 4.6 The original 20 images as the templates for PCA 71

Table 4.7 The re-elected set of templates containing 9 images 71

Table 4.8 The final 15 templates for PCA 71

Table 4.9 Euclidean distances of the detected eyes and their eigeneye approximations
(men) 72

Table 4.10 Euclidean distances of the detected eyes and their eigeneye approximations
(women) 73

Table 4.11 Calculated distance values for non-eye subjects..... 74

Table 4.12 Template matching results for the three potential eye regions 75

Table 5.1 Implemented Skin Color Classifiers 83

Table 5.2 Time measurements of the algorithms..... 93

Table A.1 ECU face detection database 105

Summary

Inspired by meaningful applications such as human computer interaction (HCI) based on gaze direction and the prevention of road accidents caused by drivers' drowsiness by monitoring their eyes dynamics, we propose in this thesis several methodologies for the progressive segmentation of the face and eyes from static images to aid the acquisition phase of these applications.

First, we have proposed three pixel-based skin color classification schemes based on Self Organizing Topological Tree (SOTT) to search for face from complex background in color images. Our proposed SOTT-based skin color classifiers provide alternative solutions to skin color detection with smaller amount of training data and memory requirement. The proposed SOTT schemes reduce the search time required by the basic SOM approach to this problem.

Second, we have developed several efficient techniques to segment the face/eye from gray level images. These techniques include an Adaptive Center of Mass method to locate the largest pixel cluster of object of interest, a Window Growing method to extract a rectangular window that is just sufficient to embrace the face, and a Histogram Projection method to locate the eyes from the facial images. Principal Component Analysis (PCA) with our suggested method for training samples selection is applied to verify the detected object of interest. For 445 images with complex background and only one face in the ECU database, our proposed methods can achieve the success rate of 87% for face detection, and achieve about 91% of successful eye detection rate for 281 upper face images without glasses from the AR database.

Lastly, a test bed with a user friendly Graphical User Interface (GUI) for convenient analysis and evaluation of the algorithms is designed. This interface eases the evaluation of the algorithms in real life environments and gives an insight of the performance and efficiency of the algorithm under test at different stages of an application. Upon determining the specifications of the application, this test bed can be used to investigate and select suitable methods for different processing requirements in face/eye detection.

Although the techniques and methodologies proposed and developed in this research are primarily meant for static images, they can augment HCI system's robustness greatly with the supplement of motion detection and video based processing techniques.

Chapter 1

Introduction

1.1 Motivation

Problems such as face recognition or eye tracking are the tasks that human are good at. However, to do these tasks automatically with a computer is not as easy as one would conceive. Such tasks are the most representative ones in image analysis, partly due to the challenges and complicated methodologies involved. Given a still image or a video sequence, the face or the facial features such as eyes and pupils are to be located for further processing in an application. Segmentation, localization, and verification are involved in the solutions to problems of this nature. The detection system should also be able to handle variations in illumination, changes in camera distance, faces of different races and expressions, etc.. In dealing with such problems, one would be amazed that human (even infants) are actually very good learners.

Despite the difficulties, automatic human face/eye detection has its widespread applications in biometric, security and intelligent human-computer interfaces (HCI). In 2002, a hands-free text entry system *Dasher* has been introduced by the famous science journals *Nature* [DAS02, DAV02]. This is a fast system that integrates a language model for word prediction and enables a user to write at up to 25 words per minute based on his

gaze direction (see Fig. 1.1). It greatly benefits people who have difficulty of using a standard keyboard or mouse. It can also be used for text entry with mobile phones and PDAs, on which a conventional keyboard is not convenient. Based on application systems such as *Dasher*, some researchers have built up the head mouse system to fully control the mouse [NIP03], while some others have designed the eyebrow-clicker as a binary switch to simulate mouse click, and use it to play simple games [JON02]. We can foresee that vision-based HCI will become more efficient and entertaining in the near future.

Figure 1.1 Text entry system ‘Dasher’

Other than applications such as HCI, eye detection can also be used to detect drivers’ drowsiness and prevent road accidents to a certain extent. According to report from IVI (Intelligent Vehicle Initiative), driver’s drowsiness is the most frequent form of human errors that leads to tragic traffic accidents [IVI01]. Research studies carried out by the Federal Highway Administration have proven a positive correlation between the ocular dynamics and the driver alertness and performance [IVI01]. Their driver simulator-based studies have shown that immediately prior to fatigue-induced accidents, the driver’s eye exhibits a change in blinking behavior and eyelid closure percentage over time (see Fig. 1.2). These results suggest the potential of an eye tracking system with an in-vehicle camera to help avoiding traffic accidents effectively. Reduced driver alertness can be detected by a monitoring system that measures the ratio of vertical to horizontal pupil diameter, and other ocular dynamics like blinking frequency and duration, saccade frequency (i.e., rapid eye movements), etc.. In anticipating potentially hazardous ocular parameters, an automatic response can be generated to wake up the driver, or an

“autopilot” can be activated if the automobile is equipped with such advanced capability [ERI97].

Figure 1.2 Off-road simulated drivers’ eye behavior

Inspired by the above important applications, we have researched into and developed several efficient techniques which are directly or indirectly applicable to face/eye detection. We are interested in image processing techniques that can identify the more meaningful regions from the cluttered background, in order to accelerate the searching process for the region of interest (RoI). To achieve this we exploit the color information to identify the skin regions. We have also devised several novel algorithms simply based on the gray level information to locate and extract eyes from facial images. We hope that the success in detecting and locating the eyes will assist in the measure of the eye occlusion and the registration of the fixation point for the gaze tracker. Inspired by the intriguing causality among different processing techniques, it is also desirable to develop a Graphical User Interface (GUI) for ease of testifying the intricate effects of different algorithms at various processing points of an application.

1.2 Objectives

The purpose of this thesis is to explore efficient techniques for an eye detection system used in vision-based text entry or a game console. Ideally, the system should be able to handle complex background, illumination variation, change in distance from a camera, and slight head tilting. Real-time eye tracking can be performed based on motion tracking from the previous state of the eye location. In practice, automated vision-based object detection and identification can never be error proof. Therefore, to design a fully automated system, the system needs to be reinitiated at intervals during the process of tracking upon loss of acquisition of the target. In this perspective, a robust approach to register the eyes on a still image is crucial to the success of the system. The primary objective of this thesis is to analyze solutions relevant and applicable to this problem and develop novel methods for this purpose, which have potential to evolve into an efficient, adaptive, and robust application. A comprehensive or complete HCI system is beyond the scope of this thesis, but a versatile graphical user interface (GUI) will be developed. The desiderata of the GUI is to provide a platform to plug and play software modules in order to evaluate and demonstrate the effectiveness of various algorithms for object of interest registration in a HCI application.

1.3 Major Contributions of the Thesis

With the aim of developing efficient techniques for object detection, especially for face and eye detection, the research reported in this thesis has led to the following findings and results:

1. *Proposal of three novel skin color detection schemes based on Self Organizing Topological Tree (SOTT).*

In order to cope with the complex background, robust skin detection is desirable for face segmentation to enable a fast search of eyes from within the potential face region dominated by skin pixels. Three pixel-based skin color classifiers based on SOTT are proposed for this purpose. Traditional skin color classifiers include parametric and non-parametric methods such as fixed range methods, Single Gaussian, Mixture of Gaussians, Multi-layer Perceptrons, Basic SOM and Bayesian classifier [PHU03, VEZ03]. Each of them has its advantages and disadvantages. Our proposed SOTT-based skin color classifier provides an alternative solution to skin color detection with less training data and memory requirement. The proposed SOTT schemes reduce the search time in comparison to the basic SOM approach to this problem [BRO01]. Part of this work has been published in a research paper titled “Self organizing topological tree for skin color detection” in the *Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 1097-1100 and presented at the conference held in Taiwan on December 6-9, 2004.

2. *Efficient techniques for face/eye segmentation.*

After the skin color pixels have been identified, the targeted Region of Interest (RoI) will be segmented from the image. We have proposed several new methods to locate and extract the face region and the eyes. By iteratively refining the center of mass of the RoI (*Adaptive Center of Mass method*), we are able to rapidly expand an appropriately sized window from the centre of mass to cover the RoI (*Window Growing method*). These techniques enable a fast segmentation of the face/eye regions for further processing, such as verification by template matching (*PCA for Template Matching*). We have also proposed another simple and elegant method, called the *Histogram Projection*, to locate the eyes in the face region. This method projects the gradient and intensity features that maximally discriminate the target object in the vertical and horizontal directions. The above mentioned methods have been tested with good results on images from the AR database [MAR98] and ECU database [PHU03]. Part of the work has been published in a research paper entitled “A new method for eye extraction from facial image” in the *Proceedings of 2nd IEEE International Workshop on Electronic Design, Test and Applications (DELTA2004)*, pp. 29-34, and presented at the workshop held in Perth, Australia on January 2004.

3. *A test bed with Graphical User Interface for algorithms testing and comparison.*

Besides the development of the above algorithms, a Graphical User Interface (GUI) has been built for the purpose of evaluating the performance of different algorithms. This test bed enables different algorithms to be tested in a real-life environment upon being converted to a .dll file and plugged into the GUI. Designers can exploit such interface to experience the effectiveness and evaluate the timing of various methods. With little modification, the test bed can also be transformed into an interface between the computer and a user by using the gaze direction for cursor control.

1.4 Organization of the Thesis

This thesis is organized into six chapters. Chapter 1 presents the motivations and the objectives of our research work, followed by a summary of the major contributions.

In Chapter 2, a brief survey of relevant literatures on face detection, eye detection, color image segmentation, as well as SOM and its applications in image segmentation, is provided. We have generalized the approaches to both the face and eye detection into two main categories, namely the knowledge-based and the appearance-based approaches. For face detection the discussion is focused more on the knowledge-based approaches making use of skin color, texture, and the shape of the face. Other color image segmentation methods for generic purpose are also studied to provide a good insight into the more sophisticated and advanced techniques. Literatures of SOM, which is the ancestor of SOTT are also reviewed.

Chapter 3 describes the proposed Self Organizing Topological Tree (SOTT) schemes for skin color classification. Our very own view on the controversial arguments pertaining to the effects of different color spaces and the existing color classifiers is presented before the introduction of the theory and operations of SOTT. The three proposed schemes are

described and discussed. The performances of the proposed schemes are analyzed in relation to other representative classifiers.

In Chapter 4, a few methodologies for RoI (i.e. face/eye) localization and extraction are proposed. The Adaptive Center of Mass method and the Window Growing are first presented. They are used to extract the target objects such as face and eyes but not limited to those. A Histogram Projection method is also introduced to rapidly locate the eyes at the facial images. This chapter also describes the use of Principal Component Analysis (PCA) for the verification of the segmented eyes. A systematic method to select a good training set to enhance the effectiveness of PCA is suggested. Towards the end of each section of this chapter, the test results on the AR and ECU databases are discussed.

Chapter 5 describes the test bed designed for algorithm comparison. The technologies employed are reviewed at the beginning of the chapter, followed by the description of the Graphical User Interface (GUI). The features of the GUI for pre-processing, post-processing, and the major steps of the algorithm are detailed in this chapter.

Lastly, Chapter 6 concludes our work by highlighting the virtues of the proposed methods and discussing some of the remaining problems. It also identifies the directions for future development.

Chapter 2

Literature Review

This chapter surveys the relevant and important literatures in four major topics pertaining to the approaches to the problems studied in this thesis. A critical review of various solutions and the recent development in the applications of interest shall provide the reflection necessary for improving existing approaches and a good yardstick for benchmarking our original methods.

2.1 Existing Face Detection Methods

The problem of automatic human face detection can be described as follows. Given an arbitrary image, the goal of face detection is to determine whether or not there is any face in the image and, if present, returns the image location and extent of each face [YAN02]. Such a problem is challenging because of the variability in the scale, location and orientation of the faces. Facial expression, occlusion, and lighting condition can also change the overall appearance of the faces, posing considerable challenge to their correct detection [YAN02].

There are a huge number of face detection approaches in literature. They could be grouped into two categories: knowledge-based and appearance-based approaches. Knowledge-based approaches make use of the explicit characteristics of a face,

including color, texture, shape, and facial features such as eyes, mouth, as well as their geometric relationship. Such characteristics are observed as “knowledge” about the face, and are expressed as rules used to distinguish between face and non-face patterns. Appearance-based approaches, on the other hand, do not care about such knowledge but treat the whole face as a pattern and employ machine-learning techniques to automatically extract the decision rules that differentiate between face and non-face patterns. Due to the nature of facial images of the intended applications, in this survey, we focus only on knowledge-based methods.

2.1.1 Skin Color Detection

Skin color is one of the most commonly used features for face detection. Most of the skin color based methods are pixel-based [VEZ03] while few others process the target image in macro-blocks [GAR99, WAN97]. Skin color based approach is fast and it is invariant to scale, orientation, and facial expressions. Two main issues to be addressed in skin color based approaches are (1) what color space is likely to enhance the detection and (2) what classifier should be used to deliver optimal performance.

The first issue is a topic of great controversy. Many researchers have investigated on whether the color space transformation eases the skin pixel detection problem through rigorous experimentations. Terrillon et al. [TER00] compared nine different color spaces using a single Gaussian model and a mixture of Gaussian models, and found out the preferred color spaces for each of the models. In [ZAR99], the performances of five color spaces were accessed and compared by using two histogram-based methods, the lookup table method and Bayesian method. The results show that for Bayesian method, using different color spaces gives little variation. These results have led Shin [SHI02] to question the effectiveness of color space transformation to skin detection problem. Four statistical measurements were proposed to examine if the transformation increases the separability between skin and nonskin classes while decreasing the intra-class difference. The results indicate that most of the color space transformations do not lead to the benefits that have been assumed. Such belief is also shared by Jones and Rehg [JON99], who have written a technical report which is believed to be the most comprehensive and detailed exploration of skin color models

to date. Another relevant but unanswered question pertains to whether the removal of the intensity information from the tri-chromatic color channels assists the detection. Some researchers believe that although different people have different skin colors, the major difference lies largely between their intensity rather than their chrominance [BAK99, CHA99, TAK98, SCH95, SPO01, YAJ96, YAJ98]. A different thought has also been raised that the removal of the intensity channel is essentially the expansion of the skin color decision boundary to cover the entire intensity channel. This comes at the cost of more false detections, and hence, all three color-channels should be kept for effective skin detection [JON99, PHU03, SOL03]. As universal consensus has been established among the researchers in this community, color spaces that have been used for skin detection include RGB [SOL03], normalized RGB [BAK99, HUN94, TAK01, WAN01, YAJ96], YCbCr [CHA99, CHA00, GAR99, KUC02, HSU02, PHU02, WAN97], HSV [GAR99, MCK98, SAN02, SIG03, WAN01], CIE XYZ [WU99], and CIE LUV [YAN98].

The most important skin color classifiers include fixed range decision boundary, methods based on Gaussian model, Self Organizing Maps (SOM), Multi-Layer Perceptrons (MLP), and Bayesian classifier. In fixed range decision boundary method, skin and non-skin colors are distinguished by a set of explicitly defined rules. For example, Chai and Ngan [CHA99] consider skin color range in YCbCr color space as $R_{Cb} = [77, 127]$, $R_{Cr} = [133, 173]$. Garcia and Tziritas proposed the decision boundary that is formed by eight planes in the YCbCr color space [GAR99]. Other fixed range skin color classifiers include [SOL03, TAK01, WAN01]. Such methods are definitely simple to implement. However, it is difficult to achieve high detection rate due to the difficulty to find both good color space and adequate decision rules empirically. Methods such as Self Organizing Maps [BRO01], Multi-Layer Perceptrons [PHU01], and Bayesian classifier [CHA00, JON99, KUC02, SPO01] are nonparametric skin color classification methods. They usually require construction of a skin probability map instead of deriving an explicit model of the skin color as in the parametric methods. Single Gaussian [YAJ96, YAN98] and mixture of Gaussians [PHU02] are used to model the skin color distribution as a single or several Gaussian functions. They are more compact skin color model representations that are able to generalize and interpolate the training data. Consequently, they rely less on the appropriate selection of the training data set. However, the goodness of fit of parametric models is

more susceptible to the skin color distribution shape, and hence the color space used. This dependency is especially pronounced for the single Gaussian model. For mixture of Gaussian models, much more expensive computation effort is required.

2.1.2 Use of Texture

Some researchers believe that human faces have distinct textures that can be used to separate them from different objects. Augusteijn and Skufca [AUG93] have adopted the second-order gray level statistical method which employs a set of co-occurrence matrices. Eight out of fourteen measures derived from these matrices were found adequate to represent the textures of hair and skin. Dai and Nakano [DAI96] also made use of the space gray-level dependence (SGLD) matrix together with the color information to detect faces. In Garcia and Tziritas's approach [GAR99], wavelet packet decomposition is applied to the rectangular face candidate regions found by skin color detection and region merging to classify the face and non-face patterns. In [SHA00], five different texture extraction methods including auto-correlation, edge frequency, primitive-length, Law's method, and co-occurrence matrices, are compared. The Law's method and co-occurrence matrix method yield the best results for the Meastex texture database [SMI97].

2.1.3 Use of Shape

Human face exhibits more or less elliptical shape. Based on this observation, Kim et al. [KIM00] searched around the candidate face regions obtained by template matching for the edge segments by using the SUSAN edge detector. After that, the parameters of the ellipse are calculated for each pair of edge segments to extract the face. Park and Aggarwal [PAR00] have proposed a partial-ellipse fitting algorithm that removes the bottom portion of the normal ellipse by 30 degrees of polar angle to exclude the shoulder pixels. With such a removal, the finally fitted ellipse is not biased towards the neck and shoulder region of the image. The least squares (LSQ) fitting [PIL96] and Hough transform [XIE02] are two of the common ellipse fitting algorithms surveyed.

2.2 Existing Eye Detection Techniques

There are two main categories in eye detection methods. Knowledge-based approaches require a priori knowledge about the eyes, such as the numerous edges they possess and their correspondence to the dark regions in a gray-level image. By utilizing such knowledge of the eyes, the candidate eyes could be located. Appearance-based approaches represent the eyes in a 2D intensity array. Instead of feature extraction and analysis, training and mapping are carried out for the classification of eye and non-eye images.

2.2.1 Knowledge-Based Approaches for Eye Detection

There are several observations about the eyes. D. Maio and D. Maltoni [MAI00] observed that the eyes possess strong horizontal edges, together with closed mouth. If the directional image vectors are projected horizontally, the local minima would correspond to the y coordinates of the eyes as well as the closed mouth. However, in their paper the directional image vectors are used for face localization and it has not been further elaborated and exploited in facial features segmentation.

The second observation about the eyes is that the pupils appear generally darker than the surrounding facial regions. Some facial feature extraction algorithm searches for local gray minima within the segmented facial regions [RIZ00]. Techniques such as contrast enhancement and intensity thresholding are involved to extract the dark regions. L. Zhang and P. Lenders [ZHA00] locate the eyes on the thresholded binary image based on the knowledge about the facial features, for instance, the hair region has the largest area in the binary image, the eyes are situated below the eyebrows, etc. Such algorithm is highly sensitive to the types of thresholding method used as well as the lighting conditions. To conclude, the gray level information would be helpful in detecting several eye candidates, but using it alone may not be sufficient to filter out other facial features such as eyebrows, which also appear as equally dark patches.

The third observation about the eyes is their shape. An artificial template could be built according to the rough shape of the eye and eyebrow such that the correlation coefficient between the template and the eye image could be calculated [BET00]. Alternatively, the shape can be represented by the boundary edges although the extraction of the feature boundary is not easy. In many cases the boundary of the eyes would be incomplete or broken when the intensity contrast around the feature is low. The deformable template based on the parameters of an eye model (i.e. the location and radius of the iris, the corners and height of the eye opening) [YUI92, TIA00a] will be able to extract the optimal eye boundaries once the template is initialized on the correct location of the eye. The problem is then the error minimization problem. However, this method is sensitive to the initial location of the template and it is computationally expensive. Snakes is another similar approach to the deformable template [PAM00].

Hough transform was also used for the detection of eyes, since it is feasible to detect shapes such as circle and ellipse by giving their parametric equations. T. Kawaguchi et al [KAW00] has proposed an algorithm to calculate the cost of using Hough transform to measure the fitness of the pairs of blobs to the image, and then select the pair of blobs with the smallest cost as the irises of both eyes. Robust edge detection is required before the application of Hough transform. In their experiment, Sobel edge detector [GON92] was used.

Recently there has been an increase in interest to apply Gabor Wavelets techniques for eye detection [SIR01, TIA00b, TIA02]. In the algorithm proposed by S.A. Sirohey and A. Rosenfeld [SIR01], Gabor wavelet-based linear filters are used for eye corner detection, and non-linear (Gaussian) filters are used for iris detection. By using Gabor wavelets, their energy could be localized in both the space and the frequency domains. However, using the linear filter alone gives many false alarms. The utilization of the non-linear filter enhances the performance of the detection.

2.2.2 Appearance-Based Approaches for Eye Detection

The basic idea of image-based approach is to generate a prototype image (template) and compare the 2D intensity array extracted from the input image with this prototype, so that the extracted object can be classified. The simplest way of forming the template would be to use the average eye. Correlation coefficients are then calculated as the measurement of the proximity between the template and the input image [DRO01].

Principal Component Analysis (PCA) is a fast, reasonably simple, and accurate method of template matching developed by Turk and Pentland [TUR91]. It transforms the eye images into a small set of characteristic images, called 'eigeneyes'. They are the principal components of the training set of eye images [TAL99, CAM00]. The main advantages of PCA are: (1) data compression is achieved by the low-dimensional subspace representation; (2) no knowledge of the geometry of eyes is required; and (3) verification is simple and efficient. This method is, however, not without limitations. First, most of the PCA implementations apply an exhaustive search for the targeted object by window scanning. The distance value for each possible location of the template would have to be calculated. Second, PCA is rather sensitive to the variation of translation, rotation and scaling of the targeted object in the image from the template.

2.3 Color Image Segmentation

Face detection based on skin color classification is an application of color image segmentation with a priori knowledge of the object of interest. Due to the close relationship between these two problems, a comprehensive survey has also been done on generic color image segmentation.

Image segmentation, according to [CHE01], is a process of dividing an image into different regions such that each region is, but the union of any two adjacent regions is not, homogeneous. Object detection sometimes can disregard the segmentation in the

background but just focus on segmenting the region of interest (RoI) from the background.

In the survey paper [CHE01], the selection of color space is discussed, and the color image segmentation approaches are categorized into (1) Histogram thresholding and color space clustering; (2) Region based approaches, such as region growing, and splitting and mergeing; (3) Edge detection approaches; (4) Fuzzy techniques; and (5) Neural Network approaches including Self Organizing Map (SOM).

Among these approaches, the color space clustering method is more appropriate than the 1D thresholding for color image segmentation, since color is multidimensional data [UCH94]. However, the clustering in color space does not consider the spatial information such as the connectivity among the object pixels. [PAU98] introduces the norm to handle the connectivity of the cluster data in the feature space, but not the spatial connectivity in the image. It is suggested that a low-pass filter be applied to handle the spatial information because they believe that introducing the spatial coordinates would destroy the cluster information.

Region growing is a natural way to handle connectivity [ADA94]. But it is difficult to determine the stopping criteria for the growing process, and the selection of the seeds affects the segmentation results. Despite these short-comings, region growing is still a popular method and [ADA94] is well cited perhaps due to its spontaneous attention to connectivity.

Edge detection is like applying a high-pass filter to the image to detect the high frequency feature – edges. Its difficulty lies at its incapability in closing the contour of the object. Thus post-processing is required in most cases. It is also relatively sensitive to noise, since the edge detector is usually applied locally. The JSEG concept by [DEN01] falls into this category.

Approaches like Snakes and active contours [CHA01] are to define the regions or their contours by minimizing a global energy function. This function usually consists of two terms, a data-driven term and a regularization term. The data-driven term usually aims to minimize the intra-class variance and maximize the inter-class

variance. It can be classified as contour-based [CHA01] or region-based [ZHU96]. The region-based methods compute for the entire area of the regions rather than their boundaries only. The regularization term is usually used to impose the smoothness of the contours on the segmentation. An example of regularization term is the length of the contours times a normalizing factor [CHA01]. The difficulties for this kind of approaches are (1) how to design a reasonable global energy function; and (2) how to solve the minimization problem. The level set method [SET97] has been noticed to be an advanced method to solve this problem.

2.4 SOM

Artificial neural networks are increasingly being employed in various fields such as signal processing and pattern recognition. They possess the information processing system that shares conceptual structures and characteristics with biological neural networks. The main characteristics of neural networks are that they have the ability to learn complex nonlinear input-output relationships, use sequential training procedures, and adapt themselves to the data. Thus, low dependence on domain-specific knowledge is required [JAI00]. Feed-forward networks such as Multi-Layer Perceptrons (MLP), Radial-Basis Function (RBF) network, and Self-Organizing Map (SOM) are considered to be the most popular neural networks for pattern classification tasks. Among them, SOM is the only one that possesses the topology preserving property, and hence, the spatial location of a neuron in the lattice corresponds to a particular feature, or group of features, of the input patterns [JAM97]. Due to this property and its unsupervised learning mode, SOM has been emerged in widespread applications that require clustering of unlabeled multivariate data.

With a little modification, SOM can also evolve into a classification algorithm, known as Learning Vector Quantization (LVQ) [KOH01]. Recently, some extensions of SOM, such as Bayesian Self-Organizing Map (BSOM) [YIN97a, YIN97b] and Self-Organizing Mixture Network (SOMN) [YIN01], are derived for performing density estimation and reported to outperform the classic expectation-maximization (EM) algorithm. In image processing field, Pei and Lo [PEI98] used SOM for color

quantization and image compression. They also suggested a butterfly-jumping sequence for inputting the training data to solve the input sequence dependent problem of SOM. Brown and Craw [BRO01] tried to perform skin color detection based on a single layer SOM with 16 to 256 neurons using the freely available package SOM_PAK written in C [KOH96]. They experimented with two detectors; one was trained with only skin pixels, and the other with mixture of skin and non-skin pixels. They noted that the skin-only SOM performed marginally (1-2%) but consistently better than the mixture model, and the sizes of SOM between 64 and 196 neurons give good results for both detectors with all color models. They tested their algorithm on their own data set and the Compaq image database [JON99], and reported the accuracy of 94% on average for different detector settings. Unlike Brown and Craw's approach, some researchers did the training and testing on the same image with SOM. Moreira and Costa [MOR96] have noticed SOM's use in image segmentation. They first detect the main chromatic features present in the image by using SOM (number of neurons they used for this stage is not specified), and then define the optimum number of clusters by incrementing the number of clusters from two and stopping when the quality measure reaches its maximum. However, only one single image with simple clusters is used for testing the algorithm and no further results are presented. Jiang et al. [JIA03] have the similar idea of utilizing SOM for image segmentation. The three color components (i.e. R, G, and B) as well as the spatial coordinates x and y are input as the feature vector to preserve some spatial relationship between the pixels. Then, the clustered blocks are merged into a specific number of regions. Again, no generalization of test images is presented. Wu et al. [WU00] have proposed an adaptive 1D SOM for color segmentation for human hand localization after analyzing the histogram-based approaches and parametric color models. Growing, pruning and merging are facilitated so that the number of clusters could be determined automatically. Supervised and unsupervised learning are combined as a transduction algorithm to adapt the color change of every frame. The skin regions are then localized by finding the centroid of the regions using SOM or region growing. Focused on the same color segmentation problem, Walder [WAL02] had an idea to detect objects such as a face by color vector quantization. The algorithm uses SOM to generate the code-book. An image is divided into blocks and hence some spatial information is incorporated. Each block is encoded with an index. (i.e., a 3D image of dimensions mh by mw by 3 is reduced in size by VQ to a 2D

index matrix of size h by w). The histograms of several vector quantized face images are plotted with respect to the index i . Object detection is carried out by scanning the image and histogram matching, while the similarity metric is the Mahalanobis distance which measures the intra-class variation (i.e. between the VQ-histograms). Tree structured VQ can be used to speed up the matching process. Rather than segmentation, Alba et al. utilized SOM together with PCA (Principal Component Analysis) for face identification [ALB01]. They first obtain the facial edge image with the ridge and valley operator. This shape information is then clustered in the face space by using SOM. The commonly used Euclidean distance is substituted by the modified Average Hausdorff Distance (AHD). Then the “shape-free” texture information is processed by PCA. The test results show that their SOM-PCA approach presents advantages over the classical PCA approach. More interestingly, inspired by the traveling salesman problem, Baraghimian [BAR89] proposed a new connected component labeling method based on SOM. The above research suggests the ubiquity and power of SOM if it is appropriately devised for a specific application.

Chapter 3

Skin Color Detection

Skin color detection is known to be an efficient preliminary process for face detection as it can eliminate irrelevant background information and enable the human computer interface (HCI) system to zoom in to the skin region rapidly. It also has the advantage of being invariant to scaling, orientation, and facial expressions [YAN02]. There are various existing skin color detectors such as the fixed range decision boundary [CHA99, SOL03, TAK01, WAN01], Gaussian and mixture of Gaussian model [MEN00, PHU02], Bayesian classifier [CHA00, JON99, KUC02, SPO01], and Self Organizing Maps (SOM) [BRO01]. Fixed range methods are simple to implement, but they suffer from the difficulty to find both good color space and adequate decision rules empirically. Therefore, they lack the robustness necessary to achieve high detection rate. Single Gaussian and mixture of Gaussian are compact skin color model representations that are able to generalize and interpolate the training data, so that they are less reliant on the characteristics of the training data set. On the other hand, the goodness of fit of parametric models is more susceptible to the shape of skin color distribution, and hence the color space used. This is especially true for single Gaussian model. For mixture of Gaussian, much higher computation effort is required. Methods such as SOM and Bayesian are nonparametric methods. They usually require the construction of a skin probability map instead of deriving an explicit model of the skin color as in the parametric methods. The performance of the nonparametric methods depends largely on the choice of the training data set. When the training set

is large enough these methods are superior to parametric methods in terms of the accuracy and computational cost [12].

Recently, Brown et al. [BRO01] proposed two SOM-based schemes for classifying skin and non-skin colors - one uses only skin samples, and the other uses both skin and non-skin samples. It is found that the latter performs better than the former scheme, and both schemes show no preference to the choice of the color space. Although Brown's method gives only moderate performance on skin color detection according to Phung's simulation [PHU03], it rekindles the interest of using SOM in color image segmentation and pattern recognition. SOM is deemed to be less resource intensive, while preserving the compact nature of the mixture model.

SOTT is a hierarchical SOM that possesses advanced features for online clustering [XU04]. In this chapter, three skin color detection schemes based on SOTT are proposed and evaluated. SOTT is preferable to SOM in practical clustering applications. SOTT reduces the computational cost significantly, especially for a large network. To some extent, its multi-resolution property provides the flexibility for the number of clusters to be adaptive to the input image. The discussion on the effect of the color space on the classifier performance is given in Section 3.1, followed by a review of the principle of SOTT in Section 3.2. Section 3.3 presents the three proposed schemes. The comparison of the proposed schemes and other important skin color detection methods in terms of performance, speed, memory requirement, and the size of the training set are presented in Section 3.4. The work in this chapter has also been revised and published in the *Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems* [DEN04b].

3.1 Discussion on Color Spaces and Classifiers

As reviewed in Section 2.2, there are two major issues to be considered before we deal with the skin color classification problem. First is the color space and second is the classifier. Different opinions, often controversial, on these problems exist and have been summarized in Section 2.2. However, we are of the opinion that these two

issues are closely related and cannot be separated from one another. The following paragraphs, with the aid of figures, explain our idea.

We collected the skin colors from the skin part of the first 100 images of the ECU database and projected them into the RGB and YCbCr spaces in Fig. 3.1 (a) and (b), respectively. The histograms along each dimension are also displayed. As can be seen, the skin colors form compact clusters in the chrominance planes of YCbCr while they occupy a much larger range of each channel of RGB. Hence, the removal of the intensity component helps to compact the skin color distribution.

Figure 3.1 Distribution of skin colors
(a) In RGB (b) In YCbCr

Figure 3.2 Distribution of skin (red) and non-skin (green) colors, and the probability of being skin color (blue) (a) In RGB (b) In YCbCr

Some researchers reported that the color space transformation makes no difference for skin color detection [SHI02, ZAR99]. We believe that it actually depends very much on the selection of the classifier. For histogram-based classifiers, since the histogram will record down the skin probability ‘everywhere’ in the color space, the compactness of the skin color distribution makes little difference to the performances of this type of classifiers. In fact, more details about the skin color distribution may be preferred so that more bins and more channels can be allocated to improve their performance (see Fig. 3.3 which shows the performances of several Bayesian classifiers with different resolutions and different number of channels). On the other hand, the fixed range methods and the parametric methods that generalize the skin color distribution as Gaussian functions would prefer the skin color Probability Density Function (PDF) to be not only Gaussian-like, but also compact and as maximally distinguishable from the non-skin colors as possible. This is because Gaussian-based methods do not consider the non-skin color distribution. They only assume that the skin color distribution has a parametric functional form. Thus separate skin and non-skin clusters in the chosen color space would help this type of methods to perform well. The advantages of Gaussian-based methods are that they require less training data than the histogram-based methods since they have the ability to generalize the distribution as a model, and they also require less memory as they have only a few parameters (such as mean and variance). Fig. 3.2 shows the statistics of 2,500 images in ECU database. We plotted the skin (shown in red) and non-skin (in green) PDF of each channel in RGB and YCbCr, which could be represented by the conditional probability $P(c|skin)$ and $P(c|nonskin)$, where c is the color vector and in this case a single element (i.e., R or G, etc.). The blue curves are the probabilities of being a skin color, obtained by $P = P(c|skin) / P(c|nonskin)$ (and multiplied by the same factor for all the plots, for the ease of displaying with the PDFs). If we draw a horizontal line at $P = 0.01$ for example, and define the colors in the range of $P > 0.01$ as skin colors, then the areas outside this range and under the red curve would correspond to $(1 - CDR)$, and the areas inside this range and under the green curve would correspond to FDR, where CDR and FDR are the **Correct Detection Rate** and the **False Detection Rate**, respectively and they are defined as

$$\begin{aligned}
 CDR &= \frac{\text{No.ofSkinPixelsDetected}}{\text{No.ofSkinPixels}} \times 100\% = \frac{TP}{TP + FN} \times 100\% \\
 FDR &= \frac{\text{No.ofNonskinPixelsFalse lyClassified}}{\text{No.ofNonskinPixels}} \times 100\% = \frac{FP}{TN + FP} \times 100\%
 \end{aligned} \tag{3.1}$$

where TP, TN, FP, and FN are acronyms for **True Positive**, **True Negative**, **False Positive**, and **False Negative**, respectively.

The abovementioned probability calculation, $P = P(c|skin) / P(c|nonskin)$ is actually the core idea of the Bayesian classifier, which is also known as the optimal classifier. For more details on the Bayesian classifier, please refer to Section 5.2.3 Implemented Skin Color Classifiers. Fig. 3.3 shows the **Receiver Operating Characteristic (ROC)** curves (a terminology borrowed from radar where a receiver is characterized by its ability to detect a transmitted signal) for the Bayesian classifiers with different number of bins in RGB and YCbCr spaces. The histograms are constructed based on the first 2,500 images of the ECU database, and the mapping tests are carried out with the next 500 images from image numbers 2,501 to 3,000.

Figure 3.3 ROC curves for Bayesian classifiers

Fig. 3.3 shows that the more the number of color channels, the better the performance. With the same number of channels, the higher the resolution (more bins), the better the performance.

After studying the pros and cons of existing skin color classifiers, we were inspired by the properties and merits of Self-Organizing Map (SOM) for the same objective. SOM by itself is not a classifier but can be considered a cluster organizer with learning ability. It has the property of preserving the topology and can map high-dimensional data into lower dimension. The Self-organizing Topological Tree (SOTT) algorithm and its use in our three proposed skin color detection schemes will be presented in the subsequent sections.

3.2 Theory of SOTT

3.3.1 Basic Self-Organizing Map (SOM)

The flow of the algorithm of basic SOM is shown in Fig. 3.4. At the first step, the weights of the neurons are initialized. Although arbitrary values can be assigned to the weights of the neurons because the SOM training algorithm is relatively insensitive to initialization, a roughly ordered initialization promises a faster convergence.

Figure 3.4 SOM training algorithm

After the initialization, the input vectors are fed into every neuron of the network. Then the neuron that has a closest matched weight to the input vector is selected as the winning neuron. Usually, the winning neuron is selected by

$$v(\mathbf{x}) = \arg \min_i \{\|\mathbf{x} - \mathbf{w}_i\|\} \quad (3.2)$$

where $\mathbf{w}_i$ is the weight of the neuron i . $v(\mathbf{x})$ is the label of the winning neuron which varies with the input vector $\mathbf{x}$.

The next step, which is also the core operation of SOM, is to update the weights of the winning neuron and its neighbors as follows:

$$\mathbf{w}_v(t+1) = \mathbf{w}_v(t) + h_{w_v(\mathbf{x}),i}(t)[\mathbf{x}(t) - \mathbf{w}_v(t)] \quad (3.3)$$

where t is the iteration step number and $\mathbf{x}(t)$ is the input vector which varies with t . The scalar multiplier $h_{w_v(\mathbf{x}),i}(t)$ is called the neighborhood function. It can be viewed as a smoothing kernel over the topology of the SOM which is defined by the algorithm. A commonly used neighborhood function is a Gaussian kernel:

$$h_{w_v(\mathbf{x}),i}(t) = h_0 \exp(-\|\mathbf{r}_i - \mathbf{r}_v\|^2 / \sigma^2) \quad (3.4)$$

where $\mathbf{r}_i$ and $\mathbf{r}_v$ are the coordinates of the neurons i and v , respectively. $h_0 = h_0(t)$ and $\sigma = \sigma(t)$ are some monotonically decreasing functions of time.

The input vectors from the training set will be fed into the network iteratively until the network converges. The convergent criterion is determined by the changes in the neuron weights between two successive iteration steps being asymptotically small.

3.3.2 Butterfly Permutation for Input Randomization

SOM algorithm can be used for on-line learning schemes, which is very useful for large training sets and real-time applications. However, online learning would cause the learning performance to be order dependent. For example, in image processing, the input data consists of pixel samples from a two dimensional coordinate system. Training the SOM with a raster scanning order of the input pixels has a strong tendency to over-train some groups of neurons. A block-based butterfly jumping sequence [PEI98] was used to sub-sample the pixels from each block to form different training sweeps. This fractal scanning process does not produce a sequence of inputs with maximal randomization of the input data. To accurately reflect the learning ability for a given source, a global butterfly permutation sequence is used in [XU04T] to present the spatially correlated input data from a multi-dimensional coordinate system. The aim is to let the neurons learn the global characteristics of the training source as early as possible to prevent the performance from being degraded by order dependent learning. The butterfly permutation is defined by a mapping, $\pi: Z \rightarrow Z^n$ of an input order number $r \in Z$ to a n -dimensional coordinate system, $(x_1, x_2, \dots, x_n) \in Z^n$ where Z is a finite integer space. For $x_i \in [0, N)$, the butterfly permutation π can be expressed mathematically as:

$$\pi(r) = \left\{ (x_1, x_2, \dots, x_n) \in Z^n \mid x_j = \sum_{i=0}^{\lceil \log_2 N \rceil - 1} 2^{\lceil \log_2 N \rceil - 1 - i} (r_{ni} \oplus r_{ni+j}) \forall j \neq n, x_n = \sum_{i=0}^{\lceil \log_2 N \rceil - 1} 2^{\lceil \log_2 N \rceil - 1 - i} r_{ni} \right\} \quad (3.5)$$

where r_i is the i -th bit of the binary representation of the decimal number r , and r_0 is the least significant bit.

The presentation order, r of the pixels for a two-dimensional ($n = 2$) coordinate system, with $x_1, x_2 \in [0, 8)$ is shown in Fig. 3.5. This corresponds to the special case of the localized butterfly jumping sequence for an image sub-block of size 8×8 defined by [PEI98].

$x_1 \backslash x_2$	0	1	2	3	4	5	6	7
0	0	32	8	40	2	34	10	42
1	48	16	56	24	50	18	58	26
2	12	44	4	36	14	46	6	38
3	60	28	52	20	62	30	54	22
4	3	35	11	43	1	33	9	41
5	51	19	59	27	49	17	57	25
6	15	47	7	39	13	45	5	37
7	63	31	55	23	61	29	53	21

Figure 3.5 Butterfly permutation order for input of a 2-D coordinate system

3.3.3 SOTT Algorithm

SOTT is a newly proposed SOM algorithm to accelerate the search procedure and hence enable on-line learning [XU04T]. Both intra-layer neighborhood relationship and inter-layer parent-child relationship coexist in SOTT. The multi-path search differentiates SOTT from other tree-structure SOM which are usually globally sub-optimal and fails to find the real best matching neurons during training. A 2D and a 1D quad-tree structured SOTT with three layers of neurons are shown in Fig. 3.6.

Figure 3.6 2D and 1D SOTT structure

The initialization and winning path search of SOTT are similar to those of the basic SOM except that it involves a multi-layer search from the root layer to the leaf layer. Once the winning leaf neuron is found, the winning path is traversed bottom-up via the parent neuron of the winning leaf neuron until the root neuron is reached. The updating rules are described by the following equations.

$$w_v(t+1) = w_v(t) + \alpha(m)[x(t) - w_v(t)] \quad \forall v \in \Gamma \quad (3.6)$$

$$w_u(t+1) = w_u(t) + \alpha(m)h(u, v, m)[x(t) - w_u(t)] \quad \forall v \in \Gamma \text{ and } u \in \mathcal{N}_v(m) \quad (3.7)$$

where v is the winning neuron, and $\mathcal{N}_v(m)$ denotes the neighborhood of v at the sweep number m . A sweep is a period during which the updating parameters remain constant. In our simulations, we set the number of input data in one sweep to be 12 times the number of neurons on the leaf layer. This means on average, each neuron on the leaf layer will be updated 12 times as the winning neuron in a sweep. The learning rate of the winning neurons at the m -th sweep, $\alpha(m) = \alpha(0)k_1^m$ is a monotonic decreasing gain function of sweep time. The initial learning rate, $\alpha(0)$ is usually set to a value slightly less than 1. The learning rate for the neighborhood of the winning neuron is scaled by the taper function $h(u, v, m)$. This neighborhood taper is usually defined as a time-decreasing Gaussian function of the spatial distance between the winning neuron and its neighbors, i.e.,

$$h(u, v, m) = e^{-\frac{\|u-v\|^2}{\sigma_i(m)^2}} \quad \forall v \in \Gamma \text{ and } u \in \mathcal{N}_v(m) \quad (3.8)$$

where $\sigma_i(m) = \sigma_i(0)k_2^m$ is the neighborhood width at the i -th layer at the sweep number m . It defines the boundary of the lateral interaction between the winning neuron v and other neurons on the i -th layer. The initial neighborhood width of the i -th layer, $\sigma_i(0)$ is usually set to a value between 1 and $N_i/2$ where N_i is the number of neurons at the i -th layer. Both the learning rate, $\alpha(m)$ and neighborhood width, $\sigma(m)$ decrease with m . The constant k_1 and k_2 are set to a typical value of 0.99.

To keep the neurons of the same family intact during the entire training process, the following updating rules are imposed:

- 1) The initialization of neighborhood widths, $\sigma_i(0)$ at adjacent layers is made to be proportional to the ratio of the numbers of neurons on those two layers:

$$\frac{\sigma_i(0)}{\sigma_{i-1}(0)} = \frac{N_i}{N_{i-1}} \quad (3.9)$$

- 2) The children neurons will only be updated if their parent neuron is also updated.
- 3) The neighborhood neurons will only be updated if it is sufficiently close to the winning neuron of their layer. This is to minimize the unnecessary computation with little or no effect as the neighborhood learning rate reduces exponentially with increasing $\sigma_i(m)$. A neighborhood neuron, u is considered to be close enough to the winning neuron, v_i at the i -th layer if

$$\frac{\|u - v_i\|}{\sigma_i(m)} \leq \delta \quad (3.10)$$

The value of δ is usually set to be between 1 and 10. In our experiments it is set to be 3 since the taper function $h(u, v, m)$ is very small and close to zero when $\frac{\|u - v_i\|}{\sigma_i(m)}$ is 3.

The neighborhood neurons satisfying both Rules 2) and 3) will be updated with Eq. (3.7).

At the end of each sweep, the weight vectors of the leaf neurons in the current and the previous sweeps are compared. If their difference is less than a small threshold value, then the training is completed. The convergence criterion can be stated as:

$$\frac{\sum_{j=1}^{N_L} \|w_{lj}(m) - w_{lj}(m-1)\|}{N_L} \leq \varepsilon \quad (3.11)$$

In our simulation, if the average square difference of the neuron weights, w_{Lj} at the leaf layer is less than 0.001, the training is terminated.

3.3 SOTT for Skin Color Detection

As discussed in Section 3.1, different skin color detection algorithms have their own advantages and disadvantages. The designers may choose the one that suits their available resources and applications. For example, if there is a large database and plenty of memory resource available and on-line trainability is not a mandatory criterion, then one may choose to implement the Bayesian method since it is robust and simple. In our project, we are exploring SOTT as a viable alternative to supplement the existing methods for skin color classification based on its advantages. First of all, SOTT is suitable for on-line training when it is used in an unsupervised mode. Its hierarchical structure further enhances this capability by reducing the computational complexity of time-consuming winning neuron search of the basic SOM from $O(N)$ to $O(\log N)$. Second, SOTT can cluster the colors in the color space and hence reduce the memory requirement. We proposed three experimental schemes to probe the use of SOTT in skin color classification. In the following three schemes to be presented, the SOTT used is a 2D network with 64 leaf neurons.

3.3.1 Scheme 1 – Fully Adaptive Scheme

In the fully adaptive scheme, the SOTT adapts its codebook to each image by initializing the network before a new image arrives. Since it is an unsupervised process without any training source available (the only available training data would be the test data itself), we must define some explicit rule to classify the pixels as skin or non-skin. Here we adopt the fixed range skin-color filter of D. Chai and K. N. Ngan [CHA99] as the rule. For those neurons with their weights fall into the range $R_{Cr} = [133, 173]$ and $R_{Cb} = [77, 127]$, they are classified as skin neurons.

Obviously there are advantages and disadvantages of this scheme. As its name suggests, this is a fully adaptive scheme, the clustering by the SOTT can be done

independently for every image, although it requires a slightly longer training time (the extra training time required is image-dependent but usually the number of training pixels needed is less than 1/10 of the image size thanks to the butterfly jumping sequence. It provides the global characteristics of the image early to allow the network to converge very fast). As the codebook is dynamically generated according to the incoming image, the neurons are distributed more reasonably for one particular image regardless of the variations in the illumination, background or any other situational changes. Unfortunately, as the classification rule is itself fixed, the performance of this scheme is at best, compatible to the fixed range classifier. We have observed that for an image of size 352×288 , the training time is about 1 second and the testing time is around 3.7 seconds on a Pentium III PC for a SOTT with 64 leaf neurons using two color channels.

We have also tried to incorporate other features besides color, such as the spatial coordinates x and y , and the texture information into the SOTT. We encountered two difficulties during our experiments. First, the optimal number of neurons is hard to determine. It must be large enough to cover the high-dimensional feature space, yet small enough to avoid inefficiency. Second, the weights of these features are difficult to adjust, especially when the number of neurons is inadequate. Fig. 3.7 shows the filtered images by using $[Cb, Cr, x, y]$ as the feature vector for a 16 leaf-neuron SOTT. By changing the weights of these features, the results vary significantly. Sometimes, large areas of non-skin pixels are also ‘dragged’ into the skin clusters due to the heavy weights of the spatial coordinates.

Figure 3.7 Filtered index maps incorporating spatial coordinates with weight $[Cb, Cr, x, y]$ of (a) $[255, 255, 0, 0]$; (b) $[255, 255, 31, 31]$; (c) $[255, 255, 63, 63]$.

3.3.2 Scheme 2 – Semi Adaptive Scheme

In the semi adaptive scheme, the training image set is separated from the testing set. We use the first 100 images of the ECU database for the SOTT training. Every time when the network converges in a training image, a new image will be input. A manually segmented skin image set is also required during training for the statistics computation in the classification process, i.e., during the entire training process, the number of skin and non-skin pixels won by each neuron will be recorded in addition to the codebook. After the training phase, the SOTT is ready to test any image based on the skin color probability of each neuron. The classification of skin and non-skin neurons are similar to the classification of skin and non-skin bins in Bayesian classifier. That is, if the ratio of skin pixels over non-skin pixels exceeds a certain threshold, the neuron is classified as a skin neuron, and hence, all test image pixels that are mapped to this neuron will be classified as skin pixels. The difference here is that SOTT does the self-organization (clustering) first before the classification process begins, and the number of the training images used is much less than that of the Bayesian method.

As can be seen in Fig. 3.9 in Section 3.4, the performance of this scheme of SOTT is not as good as Scheme 3 and the Bayesian method, though it is far better than the basic SOM [BRO01] and the single Gaussian model [MEN00]. Nevertheless, it uses much less training data and memory resource than the Bayesian classifier.

3.3.3 Scheme 3 – SOTT-Bayesian Scheme

This is the scheme evolved from the second scheme but has combined the clustering property of SOTT and the probabilistic feature of Bayesian. It is similar to Scheme 2 except that it divides the training into two phases. During the first phase, the SOTT is trained as in Scheme 2, but without recording any skin/non-skin statistics. In the second phase, a large number of training data is mapped to the neurons and the number of skin and non-skin pixels are accounted. This is done with the help of the

manually segmented skin image set of the ECU database. 2,500 of them are used in this phase. It is worth noted that during the second phase, the neuron weights are static and will not be updated any more. While mapping these data, the Bayesian method is utilized to determine the skin and non-skin neurons. Lastly, testing is done by mapping the color vectors to the neurons to decide the skin/non-skin classification of the pixel by the class of winning neuron to which it is mapped.

From Fig. 3.9 in Section 3.4, it can be seen that the performance of the SOTT-Bayesian scheme with 64 leaf neurons is comparable to the Bayesian classifier with 32×32 histogram bins based on the same CbCr space. The codebook obtained after the first phase of training is shown in Fig. 3.8. The skin neurons are represented by red '+' and non-skin neurons are represented by the black 'x' (the neurons are classified with the threshold set to 0.24). The network neighborhood is linked by the dotted lines. As can be seen, the neurons are more densely distributed at the skin color region (the blue rectangle of skin color region from [CHA99] is marked as a reference) than those outside the region. This is a good feature, since ideally we would like to have denser neuron distribution at the boundaries between the skin and non-skin clusters and maybe inside (but definitely not outside) the skin cluster. Such distribution can save the number of neurons required for good performance as a skin color classifier. This is because more neurons are allocated at the areas of high ambiguity while the clearly distinguishable areas are represented by reasonably small number of neurons.

Figure 3.8 Codebook obtained after SOTT training

3.4 Comparison of SOTT and Other Important Skin Color Classifiers

Fig. 3.9 compares the performance of the three proposed SOTT schemes with several other important skin color classifiers with the ROC curves. How we derive the ROC curve could be found in Section 3.1. As high *Correct Detection Rate* and low *False Detection Rate* are simultaneously preferred, the left upper corner in the diagram is the preferable area for the best performance of the ROC curves. It is evident that the Bayesian RGB with 64×64×64 bins gives the best performance among all the compared classifiers, followed by our SOTT scheme 3 with 64 leaf neurons. Bayesian CbCr 32×32×32 has performance comparable to SOTT Scheme 3, but requires much more memory (see Table 3.1). All SOTT Schemes outperforms the Gaussian [MEN00] and the Basic SOM [BRO01].

Figure 3.9 Comparison of skin color classifiers in their performances

Table 3.1 lists the comparison of the estimated memory requirements between the abovementioned classifiers. The speed of Bayesian is then taken as the reference for our estimation of speed, which takes around 0.6 second for an image of size 352×288 . In order to make a more platform-independent comparison, the processing time of the fixed range method is scaled to 1, and the processing time of other classifiers are normalized accordingly. In our experiments, the algorithms are coded with Visual C++ and run on a Pentium III 733MHz PC. For an image of size 352×288 , our Scheme 1 requires 1 second of on-line training time and 3.7 seconds for testing. Schemes 2 and 3 each requires about 4 seconds test time. All these three schemes have a great improvement in terms of speed compared to the basic SOM. From the memory perspective, fixed range method, Gaussian, and SOTT Scheme 1 use almost negligible memory space. Basic SOM and SOTT Schemes 2 and 3 require moderate amount of memory. The optimal classifier Bayesian in contrast needs 2MB of memory space for a histogram of $64 \times 64 \times 64$ bins.

Table 3.1 Comparison of color classifiers in terms of speed and memory requirements

Classifier	Specifications	Normalized Processing Time	Memory (Bytes)
Fixed range [CHA99]	CbCr	1.0	32
Gaussian [MEN00]	CbCr	35.1	50
Basic SOM [BRO01]	100 neurons, 2D space	186.1	3,200
Bayesian [PHU03]	RGB, 64 bins per channel	2.1	2,097,152
SOTT Scheme 1	CbCr, 2D network, 64 leaf neurons	15.7	32
SOTT Scheme 2	CbCr, 2D network, 64 leaf neurons	13.3	1,856
SOTT Scheme 3	CbCr, 2D network, 64 leaf neurons	13.3	1,856

In the above results, the performance, the processing time and memory comparisons of the Gaussian [MEN00] and Basic SOM [BRO01] are conducted by Phung [PHU03].

3.5 Summary

In this chapter, we proposed three pixel-based skin color classifiers based on the advanced features of SOTT. The three proposed schemes have their own characteristics. Scheme 1 trains with and tests on the same color image, and thus its codebook is adaptive to every new incoming image. The drawbacks are its slight increase in on-line training time, and the need of a set of explicit rules for classification, which greatly restrict its performance. Scheme 2 uses a small set of images for training. Comparing to the Bayesian method, which has been known to be the optimal classifier, it requires less amount of training data and memory space at the expense of slower processing speed than the Bayesian method. The performance is also inferior to the Bayesian method but it outperforms several other well-known methods such as the Gaussian method and the Basic SOM. Scheme 3 has comparable performance to Bayesian classifier when its number of neurons is also comparable to the Bayesian classifier. It requires a large training data set as the Bayesian method. It is slower in training and testing time but requires less memory than the Bayesian method for similar performance.

Upon exploring different pixel-based skin color detection methods, the next important processing task in HCI application is to locate the pixel clusters and extract the region of interest (RoI), which will be elaborated on in the next chapter.

Chapter 4

RoI Extraction and Verification

In this chapter, several methods for Region of Interest (RoI) extraction and verification have been proposed for the location and identification of the objects of interest. Among them, Adaptive Center of Mass (ACoM) and Window Growing have been tested on face and eye detection from captured images. Histogram Projection of object features is used to locate the eyes from the facial images. Principal Component Analysis is used for the verification of the detected eyes. However, the use of these methods is not restricted only to these applications. With appropriate modifications these methods could be applied to detect other objects of interest, depending on the specification of the problem. For this thesis, we confine our discussion and illustration of the proposed methods to only face and eye detection. A portion of this chapter has been published in the *Proceedings of 2nd IEEE International Workshop on Electronic Design, Test and Applications* [DEN04a].

4.1 Adaptive Center of Mass and Window Growing

4.1.1 Adaptive Center of Mass

The Center of Mass (CoM) of an image is an important affine-invariant-metric used as control point for many image-processing tasks, such as region growing, image

registration, motion detection and object recognition [SAD02]. We make the searching of CoM adaptive in this section. The aim is to accurately identify the single largest darkest region in gray-level images (in the case of binary images, the single largest region) and to locate the CoM *within* the region without being influenced by the scattered small dark regions and noise.

The calculated position of conventional one-pass CoM calculation has high potential to fall outside the region. This is because a single-pass CoM calculation is highly prone to erroneous estimation because of the influence of various clusters distributed in a complex image. To improve the robustness, we revise the CoM algorithm in such a way that the CoM is progressively locked into the targeted cluster irrespective of the distribution of the less important clusters in the image. The adaptation is achieved by narrowing the search area around the region of interest (RoI) iteratively as prompted by the displacement of successive CoM's. It is notable that the CoM located by our proposed adaptive CoM method significantly enhances the probability that the final determined CoM is well within the single important region. This property is extremely useful and important for the applications such as region growing and motion detection.

There are several important considerations for improving the accuracy and speeding up the adaptive CoM calculation, such as the location of the initial CoM, the distance metric for defining the distance of two successive CoMs and the search window, the coverage of the initial Region of Interest (RoI), and the convergence criterion to terminate the CoM operation. These considerations would be discussed further in this section.

In this thesis the Adaptive Center of Mass method has been experimented in two applications: (1) to locate the face in the complex-background image, and (2) to locate the eyes in the rectangular upper face image. By locating the object of interest in an image, further processing such as object tracking by motion estimation, or object extraction by region growing or window growing could be carried out. Both applications have been used to illustrate the method of Adaptive CoM. The ECU Database and the AR Database are used, respectively (see Appendix A for more information of these two databases).

For the face location application, the images have been down sampled and pre-processed by the Bayesian skin color classifier at RGB color space with 64 bins per channel. The probability maps, in which the intensity values represent the probabilities of being skin, are the input images for the Adaptive CoM (see Fig. 4.1). In the eye detection experiment the binary upper face region images pre-processed by iterative Otsu's thresholding [OTS78] are input to the Adaptive CoM process (see Fig. 4.2). Several important considerations of the Adaptive CoM for both experiments will be discussed. The rhombuses drawn on Fig. 4.1 and 4.2 shows the convergence of the search windows, with the CoMs marked by red dots in Fig. 4.1 and gray dots in Fig. 4.2.

Figure 4.1 Locating CoM in face detection application

Figure 4.2 Locating CoM in eye detection application

Location of Initial CoM

The starting pixel could be located at the spatial center of the image, or at some likely centroid decided based on the prior knowledge of the image.

In the face detection application, the spatial center of the image is taken as the initial CoM. If there are more than one skin regions of comparable sizes existing in the image, the one located in the central part of the image has higher probability to be located. That explains why in our experiment with the ECU database, sometimes only one face is found in an image consisting of several faces (see Fig. 4.3).

In the eye detection application, the location of the initial CoMs correspond to the eyes could be set to $(\frac{1}{4} w, \frac{1}{2} h)$ and $(\frac{3}{4} w, \frac{1}{2} h)$ where w and h are the width and height of the input upper face image, according to the knowledge of the most probable eye locations.

Figure 4.3 Adaptive CoM on image consisting of more than one faces

Distance Metric

Two commonly used distance metrics are Euclidean distance [GON92] or City Block distance [GON92]. The City Block distance metric is adopted here because it is computationally less expensive than the Euclidean distance. The City Block distance between two pixels with coordinates (x, y) and (m, n) is defined as:

$$CB = |x-m| + |y-n| \quad (4.1)$$

The locus of all points with a constant CB from the origin O appears to be a rhombus centered at O . Incidentally, in the eye detection application such shape is helpful in lowering the weight of the eyebrow in the calculation of CoM, since the eyebrows are always located near the uppermost part of the segmented input upper face images.

Coverage of the Initial RoI

The initial RoI could cover the entire image or just part of the image. This may vary for different classes of images.

In both applications, we define the *radius* of the rhombus as the distance between the center of the rhombus and any of its vertices. The radius is used to define a rhombus boundary within which the CoM is sought. The initial radius in the face detection application is set to half of the length of the longer boundary of the image, i.e., if the image width is larger than the length, then the initial radius is half of the width. This is to ensure that the whole or a large portion of the target is covered by the initial RoI (see Fig. 4.1). In the eye detection application, the initial radius is set to half of the height of the upper face image (see Fig. 4.2).

Computation of CoM at Every Iteration

The intensity value of each pixel within the rhombus (RoI) (The rhombus is defined by the present CoM and the radius) contributes directly to the probability of that pixel being the CoM. The coordinate of the CoM = (*cmRow*, *cmCol*) is calculated by:

$$\begin{aligned}
 cmRow &= \frac{\sum_{(x,y) \in RoI} I(x,y) \cdot x}{\sum_{(x,y) \in RoI} I(x,y)} \\
 cmCol &= \frac{\sum_{(x,y) \in RoI} I(x,y) \cdot y}{\sum_{(x,y) \in RoI} I(x,y)}
 \end{aligned} \tag{4.2}$$

Convergence of RoI

Let O (the CoM) be the centre of a rhombus, R with vertices $V_1, V_2, V_3, V_4 \in V$. Upon the computation of a new CoM, $O' = (O'_x, O'_y)$, a new rhombus, with vertex set V' centred at O' and radius r' can be constructed.

The radius of the RoI, r' converges according to the following rule.

$$r' = c \times (|O'_x - V_{ix}| + |O'_y - V_{iy}|) + d \quad (4.3)$$

where $V_i = (V_{ix}, V_{iy}) \in V$ is an vertex of the previous rhombus that produces the maximum projection from O along the direction of OO' , i.e.,

$$i = \arg \left\{ \min_{j=1}^4 \frac{\overrightarrow{OV_j} \cdot \overrightarrow{OO'}}{\overrightarrow{OO'}} \right\} = \arg \left\{ \min_{j=1}^4 \overrightarrow{OV_j} \cdot \overrightarrow{OO'} \right\} \text{ and the constants } c \text{ and } d \text{ are}$$

determined empirically by testing the algorithm on a large number of images. In our experiments, c is set to 0.6 and d to 0. The termination criterion is reached when the current CoM, O' is within one pixel distance from the previous CoM, O .

The final CoM is used as a registration point to frame the RoI for further processing. In our experiments, the CoM will be the starting point for the window framing process, which will be explained in Section 4.1.2.

4.1.2 Window Growing

Window Growing is a simplified version of Region Growing for regions of convex shapes. Unlike Region Growing, it does not need to grow the region pixel by pixel, but grow the frame that covers the Region of Interest (RoI). Hence, it is more efficient for convex-shape regions. Window Growing could be used in RoI extraction and object identification. In this thesis, it has been experimented on face and eye extraction before the post-processing such as template matching for the verification of the detected objects.

Traditionally, template matching requires several attempts of sliding the scaled windows around the potentially matched region for object identification. This process is also highly susceptible to the size of the target within the window. To speedup the template matching to ideally only one comparison for each test images, we need a

window that encompasses only the object for the correct test image. To enhance its robustness against scaling and translational differences, this window should be defined adaptively such that it fits the object well regardless of the scale and position of it in the original image. We call this newly developed process of framing the object *Window Growing* with the help of the Adaptive CoM.

In what follows, we will describe the method and use the application of eye extraction for its illustration. The input image for this application is the segmented upper face image which has been thresholded by Otsu's thresholding [OTS78]. It is assumed that the Adaptive CoM method has already located the CoM within the eyes.

Once the CoM is computed in the dark region, we could have a small rectangular window centered at the CoM, and grow the window adaptively such that it contains just the eye (it may cover parts of the eyebrow sometimes when the eyebrow and eye are adjoining). The black pixels are traced on the four sides of the window before it is expanded to a larger dimension along any side. If the center part of the strip (from $1/4$ to $3/4$ of the boundary length in this case) along any side is dominated by over $k\%$ of black pixels, the window boundary will be extended in that direction, otherwise it will be frozen. In the eye extraction experiment, k is 40, the initial window size is set to $1/20$ height by $1/20$ width of the input image, and the growing step is around $1/7$ of the initial window boundary length. The pseudo code and the intermediate growing process are illustrated in Fig. 4.4 and Fig. 4.5, respectively.

```

function [x1, x2, y1, y2] = WinGrowing (InputImage, CoM_x, CoM_y)

% Define parameters
set ini_window size
set step size of win grow y_step and x_step
set percentage k

% Grow the window
While (fU == 0 | fB == 0 | fL == 0 | fR == 0) (one or more boundaries was still expanding in the
last loop)

    set fU = 1, fB = 1, fL = 1, fR = 1

    U = central part of upper boundary (1 for RoI, 0 for non-RoI)
    B = central part of bottom boundary
    L = central part of left boundary
    R = central part of right boundary

    if (sum(U==1) > k% * length(U)) and (y1 > y_step)
        then fU = 0

    if (sum(B==1) > k% * length(B)) and (y2 < imgH - y_step)
        then fB = 0

    if (sum(L==1) > k% * length(L)) and (x1 > x_step)
        then fL = 0

    if (sum(R==1) > k% * length(R)) and (x2 < imgW - x_step)
        then fR = 0

    If fU == 0
        then y1 = y1 - y_step
    if fB == 0
        then y2 = y2 + y_step
    If fL == 0
        then x1 = x1 - x_step
    if fR == 0
        then x2 = x2 + x_step

end while

```

Figure 4.4 Pseudo code for Window Growing function

Figure 4.5 Window Growing process in eye extraction application

The cropped eye windows are shown in Fig. 4.6. In few cases the window contains the eyebrow instead of the eye due to the ill-defined starting point of the window growing process. Hence, template matching is necessary to authenticate and filter out the incorrect windowed images. If the windowed image is proven to be non-eye, another starting point for the window growing will be searched.

Figure 4.6 Cropped eye windows by Window Growing

The use of the Window Growing method in the face extraction would be similar. In such experiments, the initial window size is set to be a square of length equals to $1/10$ of the shorter dimension between the image height and width. This is because we assume the object of interest will not be smaller than this percentage of the image size. The growing step size is set to be $1/8$ of the initial window length. The central part of a boundary is defined as a strip from $1/5$ to $4/5$ of the length of the boundary. If above 35% of the central part of a particular boundary is occupied by the RoI pixels, that boundary would move outwards such that the RoI is expanded. The RoI is cropped until all the four boundaries are frozen. Some experimental results of the cropped faces are shown in Fig. 4.7, along with the original images, the Center of Mass images,

and the binary skin color images after Bayesian skin color classifier (which is the input image for Window Growing).

(a) Original images

(b) Adaptive CoM on skin probability maps

(c) Binary skin images

(d) Cropped face windows

Figure 4.7 Results of Adaptive CoM and Window Growing for face detection

4.1.3 Results and Discussions on Adaptive Center of Mass and Window Growing

For the face detection application, we simulated our proposed Adaptive Center of Mass and Window Growing methods on 500 ECU images preprocessed by Bayesian skin color classifier with Otsu’s thresholding. The results are summarized in Table 4.1.

Table 4.1 Results of Adaptive CoM and Window Growing for face detection

	More than one faces exist in the image (55 images are examined)		
Categories	Detected more than one faces	Detected one of the faces	Wrong detection
Number of Cases	16	22	17
Percentage	29.09%	40%	30.91%

	Only one face exists in the image (445 images are examined)			
Categories	Correct detection	Failures because of Bayesian	Failures due to Adaptive CoM	Failures due to Window Growing
Number of Cases	390	20	24	11
Percentage	87.64%	4.49%	5.39%	2.47%

We list the results for images with more than one faces and images with only one face into two separate tables. For the images consisting of more than one faces, our proposed methods is able to detect one or more of the faces in nearly 70% of them, though our focus is on the images with only one face. We analyzed all the failure cases and categorized the causes of failures into three main groups, namely failures due to the Bayesian skin color classifier, failures due to the Adaptive Center of Mass method, and failures due to the Window Growing.

The first category consists of the images that not only the skin but also the background exhibit high probability of being skin color. To correctly segregate them, we have to introduce some additional criteria in this case. However, this is beyond the scope of this thesis.

There are two main reasons for the failure attributable to the Adaptive Center of Mass operation. First, there are other larger skin parts exist in the image and draw the CoM

to those parts of the image. Second, even though other skin parts are of comparable weight to the face, they may be located near the center of the image while the face is situated towards the corner. Hence, the Adaptive CoM method, which initializes the CoM at the spatial center of the image, tends to capture these non-face skin parts.

The third reason for the failure is due to the Window Growing method. Sometimes, the eye line pixels can barely be classified as skin color pixels due to the shadow or the lighting condition. Therefore, the window can stop growing at the eye line and thus the cropped face window may not contain the eyes.

Despite all these failure cases, the proposed methods achieved 87.64% of correct detection rate among the 445 test images with only one face at any rotation in the ECU database.

Table 4.2 listed the results of some other face detectors as a reference for the comparison, which has also been summarized in [PHU03]. Among these face detectors, only [CHA99], [FER01], [GAR99], and [LUO00] are based on skin color detection. Hence, color images were selected for testing and benchmarking in these papers. Chapter 2 of this thesis has summarized different skin color detection approaches. [HUA02], [OSU97], and [ROW98] consider only intensity values and are appearance-based approaches wherein learning is involved. Which is remarkable is the AdaBoost algorithm by Viola & Jones [VIO02]. This is noted as one of the fastest detector so far. It can detect faces at 15 frames per second (frame size = 384x288) on a 700MHz Pentium III thanks to the fast scheme for feature computation, and the use of a boosted cascade for face/nonface classification. It is noticeable that the databases selected by the authors in these papers have been different. Hence, the summary only serves as a reference and should not be used for direct inference of their relative performance.

Table 4.2 Summary of other face detectors' performance

Authors	Descriptions	Image Type	Performance
Chai & Ngan [CHA99]	Fixed range skin color map in CbCr plane Face verification based on shape criteria	Color, mainly frontal	(82%, 60)
Feraud et al. [FER01]	Constrained generative model Multi-staged algorithm Window size = 15×20	Color, frontal / side views	(86%, CMU)
Garcia & Tziritas [GAR99]	Linear model of skin color in YCbCr and HSV Wavelets	Color, frontal / small in-plane rotation	(94.2%, 100)
Huang et al. [HUA02]	Radial basis function network, PCA Window size = 20×20	Intensity, frontal upright	(84.6%, MIT set B)
Luo & Eleftheriadis [LUO00]	Gaussian mixture skin color model Gaussian clustering of DCT-coefficient feature vectors	Color, frontal upright	(81%, 586)
Osuna [OSU97]	Support vector machines Window size = 19×19	Intensity, frontal upright	(97.1%, MIT set A), (74.2%, MIT set B)
Rowley et al. [ROW98]	Neural networks (MLP) Window size = 20×20	Intensity, frontal or in-plane rotated	(88.4%, CMU)
Sirohey [SIR93]	Edge-based ellipse-fitting of face boundary, Canny edge detector	Intensity, mostly frontal	(80%, 48)
Viola & Jones [VIO02]	Cascade of face / nonface classifiers AdaBoost algorithm for feature detection Harr-like features Window size = 24×24	Intensity, frontal & upright	(88.8%, CMU)
Note: Performance is given as (correct detection rate %, database name, or database size)			

4.2 Histogram Projection of Object Features

In this section, we would introduce another efficient object detection technique, namely Histogram Projection of the object features. This method, like Adaptive Center of Mass followed by Window Growing, is to rapidly locate the object of interest in the image, by projecting the significant features of the object horizontally and vertically. The experiment of locating the eyes in the upright frontal facial images by using this method has been carried out on the AR Face Database (see Appendix A for the information of the AR Database).

4.2.1 Selection of the Object Features

The selection of the object features is essential in the Histogram Projection method, since the projection of the most significant object feature gives least ambiguous indication of where the object is located in the image. In order to detect the eyes from the facial image, we have to select the unique characteristics of the eyes that are less dominant in other facial parts along the plane of projection. Intuitively, the intensity may be chosen but intensity alone may not be a good choice, since not only the eyes but also the eyebrows appear dark. In fact, the edges appearing in the eyes are found to be a good indicator through rigorous experimentations.

A feature extraction method is necessary for this purpose. We will first review the fundamentals of the two widely used feature extraction tools for edge detection, which are the Prewitt and the Sobel Operators [GON92].

The Prewitt operator is simpler to implement than the Sobel operator, but the latter has slightly superior noise-suppression characteristics, an important merit when dealing with derivatives. The Sobel operator performs a 2-D spatial gradient measurement on an image. So it emphasizes regions of high spatial gradient corresponding to edges. Typically it is used to find the approximate absolute gradient magnitude at each point of a grey scale image. In practice, these convolutional operators are applied through the use of masks, one mask for each of the two

perpendicular orientations. These masks are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid. The masks can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation. The horizontal mask, G_x and the vertical mask, G_y of Sobel and Prewitt operators are shown in Fig. 4.8 and Fig. 4.9, respectively.

For the Sobel masks shown in Fig. 4.8, the difference between the first and third columns of the 3×3 image region approximates the derivative in the x -direction, and the difference between the third and first rows approximates the derivative in the y -direction. A weight of 2 is attached to the central pixels along the edges to achieve some smoothing effect. It should be noted that the sum of coefficients for all masks will give a response of 0 in areas of constant gray level, as expected of a derivative operator.

Figure 4.8 Sobel operator

Figure 4.9 Prewitt operator

The results of applying the two masks can be combined together to find the absolute magnitude of the gradient at each point of an image and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (4.4)$$

where G_x and G_y represent the transition along the x direction (vertical edge) and the y direction (horizontal edge), respectively, and can be obtained by convolving each of the two masks with the 3×3 image block centered at every pixel of the image. To speed up the computation, sometimes the gradient magnitude is approximated by:

$$|G| = |G_x| + |G_y| \quad (4.5)$$

The angle of orientation of the edge relative to the pixel grid gives rise to the spatial gradient. This angle of the pixel is computed by:

$$\phi = \tan^{-1} \left(\frac{G_y}{G_x} \right) \pm 90^\circ \quad (4.6)$$

The eyes in a human facial image are observed to exhibit strong edges due to the existence of the iris and eye white. The next subsection would explain how the eyes are located by projecting their edges horizontally and vertically.

4.2.2 Locating Eyes by Projecting Their Edge Features

The horizontal projections of the vertical edges as shown in Fig. 4.10 are obtained by applying only one of the Sobel masks, i.e. the left one in Fig. 4.8, on the facial images. The vertical axis of the plot is the y coordinates of the image. The sum of the values of G_x (vertical edge) in each row is then horizontally plotted and could be read from the horizontal axis. The eye line is observed to have peak value in the projection plot. This might provide us a clue where the eyes lie in the facial image.

Figure 4.10 Horizontal projections of G_x (vertical edges) for upright facial images

The projection of the horizontal transitions seems to work well with those images without the subjects' teeth shown. However, when the teeth are present, the horizontal transitions are stronger at the teeth lines instead of at the eyes (see Fig. 4.11). On the other hand, the intensity projection for the eyes is shown to be lower than the teeth. Hence, we propose to detect the top two peaks of the horizontal transition projection and select the one with the lower intensity as the y coordinate of the eyes in the upright frontal facial images.

Figure 4.11 Horizontal projections of vertical edges and intensity of the same subject without and with the teeth shown

Based on the y coordinate of the eyes, the eye region could be cropped from the face region. This is to lower the computational effort of successive operations. Since the eyes also possess strong horizontal edges, the horizontal edges are vertically projected by the use of the other Sobel mask (the right one in Fig. 4.8). The top two peaks of the vertical projection plot in Fig. 4.12 give the clue where the eyes are located at the horizontal axis.

Figure 4.12 Vertical projections of horizontal edges for eye regions

4.2.3 Results of Histogram Projection for Eye Detection

For ease of analysis of the experimental results, we selected the AR images with open eyes (with three different expressions, i.e., Neutral Expression, Smile, and Anger, for each person) are used. Altogether 398 images of 134 subjects are tested.

It is worth noting that the test images have the following characteristics. First, they are with uniform background. Second, all faces are either upright or with slight tilting. Third, some of the subjects are wearing glasses. The test results of 398 images are tabulated in Table 4.3. It is shown that the glasses severely degrade the detection accuracy. Hence, the failure rates for 281 test images without glasses are segregated and produced in Table 4.4.

Table 4.3 Failure rates for all images tested

	No. of failures	Failure Rate
Failure of both eyes	47	11.81%
Failure of one eye	6	1.51%
Total	53	13.32%

Table 4.4 Failure rates for images without glasses

	No. of failures	Failure Rate
Failure of both eyes	22	7.83%
Failure of one eye	2	0.71%
Total	24	8.54%

By comparing Tables 4.3 and 4.4, it can be seen that the reflection of the glasses accounts for almost half of the total number of failure cases (53 - 24 cases out of 53). For the rest of the failure cases, it is observed that men’s moustache and women’s long hair are the main causes of failure, since both the moustache and hair could generate vertical edges and possess low intensity values, and hence the eye locations are wrongly indicated. There are also a few failures occurred in the face region extraction stage, due to the imperfect coordination between the skin-color classifier and the Window Growing

technique, which is discussed earlier in Section 4.1. Given the simplistic algorithm and its amenability to hardware implementation, it would be a good method to improve upon for eyes registration or re-acquisition of lost track in a motion picture or video stream.

4.3 Principal Component Analysis for Template Matching

After the object candidates are detected through the image segmentation process, it is necessary to verify whether the candidates detected are the object of interest. Template matching is a technique for such verification.

Among the various template-matching techniques, Principal Component Analysis (PCA), also called the Karhunen Loeve Transformation [CAM00, TAL99, TUR91 & PET99], is a reasonably simple and the most popular method. In our eye detection application, we use PCA to examine the eye candidates. This process consists of two independent steps, which are the template building and the verification.

To construct an eye template, the eigenvectors of the covariance matrix of the training set images were computed. These eigenvectors are the principal components/features that characterize the variations among the eye images, and they make up the eigenspace that describes the eyes. The axes of the eigenspace are the directions of the largest spread of the point distribution. Each eye image could be represented as a linear combination of the eigeneyes and can be approximated using only the best eigeneyes.

When a new input image is encountered, a set of weights will be calculated by projecting the image onto each of the eigeneyes. If the image is sufficiently close to the eye space, it will be determined to be the eye. Otherwise it is treated as non-eye.

Although the PCA method has the advantages in terms of accuracy and simplicity, it is vulnerable to the scale and position variance of the object to be detected in the entire

image. This weakness has been greatly alleviated by the preprocessing steps of our proposed algorithms of Adaptive Center of Mass and Window Growing described in the earlier sections.

4.3.1 Computation of Eigeneyes

Let the training set of eye images be $eye_1, eye_2, eye_3, \dots, eye_M$. All images are of the same size of $h \times w$. The average eye intensity of the set is given by

$$\mathbf{ave} = \frac{1}{M} \sum_{i=1}^M \mathbf{eye}_i \quad (4.7)$$

For the purpose of illustration, 20 eye images (i.e. $M = 20$) were taken to be the training set. They are shown in Fig. 4.13, and the average intensity eye image is shown in Fig. 4.14.

Figure 4.13 Training set of eye images

Figure 4.14 Average intensity eye image

The difference between any eye image, eye_i and the average eye, ave is

$$I_i = eye_i - ave \quad (4.8)$$

I_i can be reshaped to be a column vector of length $N = h \times w$. Let A be the matrix containing all these vectors.

$$A = \{I_1 \ I_2 \ I_3 \ \dots \ I_M\} \quad (4.9)$$

The covariance matrix C of dimension $N \times N$ is defined as

$$C = \frac{1}{M} AA^T \quad (4.10)$$

The eigenvectors of C are calculated and they are sorted in descending order of their corresponding eigenvalues. There are altogether N eigenvectors. To have an insight to their visual interpretation, the first 30 eigenvectors were reshaped to be of the same size as the original training images. These representative eigenimages are shown in Fig. 4.15.

Figure 4.15 The first 30 eigeneyes

It is observed that the first $M-1$ (i.e. 19 in this case) eigeneyes carry significantly more feature information than the others. They resemble the eye more than the others. From entropy theory, this observation can be explained by their high eigenvalues, as shown in Table 4.5.

Table 4.5 The first 20 eigenvalues

No.	Eigenvalue ($\times 10^5$)	No.	Eigenvalue ($\times 10^5$)
1	4.2889	11	0.4108
2	2.8174	12	0.3649
3	1.9285	13	0.3088
4	1.4399	14	0.2673
5	1.2585	15	0.2398
6	1.0980	16	0.2129
7	0.8450	17	0.1934
8	0.6489	18	0.1217
9	0.5899	19	0.1039
10	0.4565	20	0.0000

All the eigenvalues after the $(M-1)^{\text{th}}$ (in our case the 19th) eigenvalue are negligibly small. Hence only the first $M-1$ eigenvectors are stored in the memory for the verification of eye when a test image is encountered.

4.3.2 Verification of Eye

The potential eye image, *eye* was preprocessed in the same manner as the training set images. These includes window cropping, image resizing, and contrast enhancement. Then the difference image is obtained by subtracting the average image from the preprocessed image.

$$I = \text{eye} - \text{ave} \quad (4.11)$$

The resulting test image, *I* is reshaped to be a column vector, and is projected into the eye space by

$$\mathbf{w} = \mathbf{u}^T I \quad (4.12)$$

where *u* is the matrix consists of columns of eigenvectors, and *w* is a column vector of length $(M-1)$ representing the projection weights of the image onto the eigeneye space.

The projection image is a linear combination of the eigenvectors which is calculated by

$$\begin{aligned} \mathbf{p} &= \mathbf{w}^T \mathbf{u}^T \\ \text{or } \mathbf{p}^T &= \mathbf{u} \cdot \mathbf{w} = \mathbf{u} \cdot \mathbf{u}^T \cdot I \end{aligned} \quad (4.13)$$

The obtained row vector *p* is reshaped to be of the same size as the original image, i.e., $h \times w$. The enhanced approximated image is obtained by

$$eye' = p + ave \quad (4.14)$$

Excerpts of the average, projected and approximated eye images for test images of three different subjects are shown in Fig. 4.16.

Figure 4.16 Test, average, projected and approximated eye images (from top to bottom)

The Euclidean distance between the actual input image and the approximated image can be used as the similarity measure for the eye filter.

$$d = \| eye - eye' \| \quad (4.15)$$

If d is smaller than a threshold value, it is determined that the image is an eye. Otherwise the test image is non-eye and is rejected. An appropriate threshold value will be empirically determined through rigorous experimentations. The decision of the threshold value will be presented with the results.

4.3.3 Selection of the Templates

How to select the templates for building the eigenspace is a major factor for the success of the Principal Component Analysis (PCA) method. The purpose of the PCA is to extract the variation of the images and represent them as the Principal Components. By the nature of the PCA, images containing the eye but with different appearances should be chosen as the training set, so that the template matching process will be more accurate and reliable. A set of varying eye images makes the eigeneyes span more subspace and cover eyes with different appearances, enabling the method to be more versatile in detecting human eyes of diversity.

The correctness of this proposition is proven by building different training sets and observing their PCA results on the same test images. It was found that for two training sets of images, the set with high variance of eye appearances is capable of giving more accurate results using fewer training images than the one constructed from comparatively uniform appearance of the eyes. From this experiment, we derived our unique method of selecting the training set images.

Assume we have L eye images in total. First we randomly choose M ($M < L$) of them as the templates from which the distance values of the remaining $L-M$ images are calculated. By the definition of the distance d , the larger the value of d , the more the image differs from the templates. Hence, we replace the training set by one image with the smallest distance value and $P-1$ ($P < M$) images with the largest distance values, thus we obtain a training set of P images. The reason behind this is the image with the smallest value represents the general eye that is of similar appearance as the original set of M templates, while the images with large distance values represent the eyes of substantially different appearances among the $L-M$ images. The new training set is comparatively smaller than the initial training set. The reduced cardinality training set will be expanded by recalculating the distance values for the $L-P$ images. This time, those images with the largest distance values (i.e., K of them) are added to the training set so that we have $P+K$ images as the templates where $(P+K) \leq M$.

By comparing the results of the Euclidean distance generated by the above algorithm and the results obtained by just using the original M images as the templates, we conclude that our proposed method is more efficient ($P+K$ could still be smaller than M) and robust as the calculated distances of the subjects under test match well with our subjective judgment of the eye and non-eye subjects. In our case, $M = 20$, $P = 9$ and $K = 6$. The size of the eye image is 16×24 pixels.

The experimental results illustrate the effectiveness of this algorithm in discriminating the eyes and non-eyes subjects.

4.3.4 Results of PCA

In this section, we will elaborate this Principal Component Analysis method for eye verification by a physical example using the 81 selected images from the AR Database. Figs. 4.17 – 4.20 show the segmented eyes obtained from these images by the methods described in previous sections.

Figure 4.17 Detected left eye images of women

Figure 4.18 Detected right eye images of women

Figure 4.19 Detected left eye images of men

Figure 4.20 Detected right eye images of men

First we randomly choose 20 segmented eye images as the templates for PCA. They are listed in Table 4.6.

Table 4.6 The original 20 images as the templates for PCA

w-015-1	w-016-1	w-017-1	w-019-1	w-020-1
w-030-1	w-031-1	w-032-1	w-034-1	w-036-1
m-001-1	m-002-1	m-004-1	m-005-1	m-009-1
m-010-1	m-012-1	m-013-1	m-014-1	m-015-1

From the calculated distance values of the remaining 61 images, we select the image w-048-1 with the smallest distance value (i.e., 560) and eight images with large distance values (>1000) as the training set. They are listed in Table 4.7.

Table 4.7 The re-elected set of templates containing 9 images

w-048-1	w-009-1	w-014-1
w-047-1	w-050-1	m-027-1
m-028-1	m-067-1	m-074-1

The distance values for the remaining $81-9 = 72$ images are obtained. Six more images with large distance values (>1100) are added to the training set. Hence, the 15 images listed in Table 4.8 are chosen as the final templates for the PCA.

Table 4.8 The final 15 templates for PCA

w-048-1	w-003-1	w-009-1
w-014-1	w-038-1	w-047-1
w-050-1	m-002-1	m-027-1
m-028-1	m-033-1	m-052-1
m-057-1	m-067-1	m-074-1

Tables 4.9 and 4.10 show the Euclidean distance between the right eye images and their approximated images in the eigenspace. The distances between the chosen templates of Table 4.8 and their approximated images are all extremely small, i.e., the values are in the order of 10^{-12} .

With reference to Table 4.9, Table 4.10 and the detected right eyes shown in Fig. 4.18 and 4.20, the number 13 improperly cropped right eye image of Fig. 4.20 has large

distance value. It is highlighted in bold and italic print in Table 4.9 for ease of comparison. From the calculated distance values, we set the threshold value to 1000 for the verification of eye and non-eye images.

Table 4.9 Euclidean distances of the detected eyes and their eigeneye approximations
(men)

No.	Facial Image	Distance	No.	Facial Image	Distance
1	m-001-1.raw.jpg	883.2752	23	m-033-1.raw.jpg	1.9839×10^{-12}
2	m-002-1.raw.jpg	1.0932×10^{-12}	24	m-035-1.raw.jpg	601.3012
3	m-004-1.raw.jpg	788.8191	25	m-037-1.raw.jpg	676.2507
4	m-005-1.raw.jpg	739.2764	26	m-042-1.raw.jpg	745.1024
5	m-009-1.raw.jpg	699.4774	27	m-048-1.raw.jpg	939.9574
6	m-010-1.raw.jpg	793.6457	28	m-050-1.raw.jpg	540.8449
7	m-012-1.raw.jpg	631.0079	29	m-051-1.raw.jpg	770.3578
8	m-013-1.raw.jpg	701.5832	30	m-052-1.raw.jpg	1.5636×10^{-12}
9	m-014-1.raw.jpg	703.0182	31	m-053-1.raw.jpg	767.6427
10	m-015-1.raw.jpg	699.1913	32	m-054-1.raw.jpg	941.1020
11	m-018-1.raw.jpg	773.8593	33	m-056-1.raw.jpg	825.9081
12	m-020-1.raw.jpg	625.9330	34	m-057-1.raw.jpg	1.8952×10^{-12}
13	<i>m-021-1.raw.jpg</i>	<i>1.2512×10^3</i>	35	m-059-1.raw.jpg	570.2435
14	m-022-1.raw.jpg	755.3942	36	m-060-1.raw.jpg	819.2254
15	m-023-1.raw.jpg	769.1208	37	m-065-1.raw.jpg	726.7543
16	m-024-1.raw.jpg	881.3322	38	m-066-1.raw.jpg	951.0407
17	m-025-1.raw.jpg	758.4045	39	m-067-1.raw.jpg	1.5378×10^{-12}
18	m-027-1.raw.jpg	1.8915×10^{-12}	40	m-068-1.raw.jpg	767.5181
19	m-028-1.raw.jpg	2.7131×10^{-12}	41	m-069-1.raw.jpg	705.7108
20	m-029-1.raw.jpg	716.1842	42	m-073-1.raw.jpg	817.3053
21	m-030-1.raw.jpg	874.0001	43	m-074-1.raw.jpg	2.2093×10^{-12}
22	m-031-1.raw.jpg	725.2823			

Table 4.10 Euclidean distances of the detected eyes and their eigeneye approximations
(women)

No.	Facial Image	Distance	No.	Facial Image	Distance
1	w-002-1.raw.jpg	915.2230	20	w-036-1.raw.jpg	591.6300
2	w-003-1.raw.jpg	1.6294×10^{-12}	21	w-037-1.raw.jpg	680.2184
3	w-005-1.raw.jpg	869.0303	22	w-038-1.raw.jpg	1.6183×10^{-12}
4	w-007-1.raw.jpg	713.7448	23	w-039-1.raw.jpg	678.3244
5	w-008-1.raw.jpg	739.9446	24	w-040-1.raw.jpg	724.9451
6	w-009-1.raw.jpg	2.0342×10^{-12}	25	w-042-1.raw.jpg	685.5643
7	w-010-1.raw.jpg	720.6841	26	w-043-1.raw.jpg	741.5776
8	w-011-1.raw.jpg	715.1254	27	w-045-1.raw.jpg	775.1992
9	w-013-1.raw.jpg	783.6635	28	w-046-1.raw.jpg	603.2430
10	w-014-1.raw.jpg	2.1775×10^{-12}	29	w-047-1.raw.jpg	1.8149×10^{-12}
11	w-015-1.raw.jpg	792.1787	30	w-048-1.raw.jpg	2.0437×10^{-12}
12	w-016-1.raw.jpg	898.2733	31	w-050-1.raw.jpg	2.8271×10^{-12}
13	w-017-1.raw.jpg	693.0702	32	w-051-1.raw.jpg	740.3953
14	w-019-1.raw.jpg	755.7028	33	w-052-1.raw.jpg	811.8595
15	w-020-1.raw.jpg	756.7662	34	w-054-1.raw.jpg	835.9951
16	w-030-1.raw.jpg	675.2521	35	w-056-1.raw.jpg	927.9553
17	w-031-1.raw.jpg	756.4487	36	w-057-1.raw.jpg	755.5235
18	w-032-1.raw.jpg	781.8779	37	w-059-1.raw.jpg	768.7882
19	w-034-1.raw.jpg	659.0942	38	w-060-1.raw.jpg	678.3250

With the expansion of the database and an appropriate choice of the templates, we expect a threshold value with a wider margin between the eyes and non-eyes subjects could be determined. This is important as the robustness of the overall scheme can be greatly enhanced by augmenting the role of coarse eye region detection by a reliable verification. In adverse situation that the location of the face and the eye region is difficult to determine, several segmented face and eye candidate regions may be passed on to this verification stage.

Even if the detected eye region does not contain the eyes, as long as the starting seeds for the Adaptive Center of Mass can be located and a window can be grown from the CoM, the eye detection process and the template matching will still be carried out. Fig. 4.21 and Fig. 4.22 show the erroneously cropped ‘coarse-eye-region’ and the ‘detected eye’, respectively. Table 4.11 shows the calculated distance values for these non-eye images.

Figure 4.21 Erroneously cropped eye regions

Figure 4.22 Poorly segmented eyes

Table 4.11 Calculated distance values for non-eye subjects

Facial Image Name	Euclidean Distance
w-041-1	1.5895×10^3
w-033-1	1.5357×10^3
w-018-2	860.9673
w-033-2	1.3074×10^3

From the calculated distance values, three of the tested images are classified as non-eyes. However, Image No. w-018-2 is a false acceptance, i.e., a non-eye is being classified as an eye. If it is validated by template matching that no eye exists in the eye region, this information will be feedback to the earlier stage to seek for the next likely eye region for detection and verification. Once the next most probable eye region is found, the eye would again be detected and verified. The following cases are those succeeded in finding the eyes from three potential eye regions. Those regions are displayed in Fig. 4.23. The results of template matching for the three eye regions are presented in Table 4.12.

Figure 4.23 Three eye regions for 6 previously failed subjects

Table 4.12 Template matching results for the three potential eye regions

Image Name	1 st region	2 nd region	3 rd region
w-018-1	654.4599	1.2597×10^3	1.3985×10^3
w-035-1	749.5739	1.7448×10^3	N.A.
w-049-1	1.6636×10^3	756.1216	N.A.
w-041-1	N.A.	1.2755×10^3	844.2590
w-055-1	1.4880×10^3	1.0224×10^3	N.A.
w-033-1	N.A.	1.0813×10^3	1.0579×10^3
N.A.: Not applicable – no starting seed for CoM or window growing			

From the above results, we observed that w-033-1 still fails despite two of its three eye regions contain the eyes. The false rejection is due to the search for the CoM is distracted by the hair. For w-055-1, although the obtained distance value from the region (the 2nd region) that contains the eyes exceeds the preset threshold value for PCA (i.e., 1000), it is still the smallest value among the three (ignoring the ‘N.A.’ cases). The saving grace is that for another four images, the eyes are finally located in one of the three eye regions. Ultimately, there are only 2 out of 87 cases (2.3%) failed in our tests.

4.4 Summary

In this chapter we proposed several efficient techniques for object localization, extraction and verification. These techniques are in many aspects, different from the existing ones reported in the literatures. First, the CoM of the RoI is now able to be located adaptively so that it is more likely to fall into the RoI. Second, the Window Growing method eliminates the need to grow the RoI pixel by pixel, and is able to rapidly frame the RoI as a window for post-processing. Third, for the first time a simple and elegant projection of selected features helps to locate the eyes. Finally, a systematic way of selecting the training set templates for PCA that maximizes its classification reliability has been proposed and implemented. The results of our original work have also been encouraging. Adaptive CoM and Window Growing is able to successfully extract the faces in 87.64% of the 445 images with only one face. The Histogram Projection can correctly locate the eyes in 91.46% of the 281 upper face images.

Chapter 5

Test Bed for Application Development

This chapter describes the test bed designed and built for algorithm evaluation and application development. The Graphical User Interface (GUI) is built with C# in the .NET environment. The application designer could design the algorithm in C++ in a DLL project. Upon compiling the project, the .dll file would be generated, and it is called in the GUI project.

In this chapter the relevant technologies are first introduced, followed by the description of the functionalities of the GUI. A series of preliminary algorithms is implemented for the functionality test. They are explained in details. Lastly, some observations and measurements of the system are given.

5.1 Applied Technologies

5.1.1 TIFF

TIFF is an acronym for *Tag(ged) Image File Format*. TIFF is primarily designed for raster data interchange developed in 1986 by an industry committee chaired by the Aldus Corporation (now part of Adobe Software). Its main advantage is the highly flexible and platform independent format. Besides, it is well supported by numerous image processing software. Hence, it was chosen as the image file format for the captured images in our developed test bed.

TIFF uses 4-byte integer file offsets to store image data, with the consequence that a TIFF file cannot have more than 4 gigabytes of raster data. It supports both lossless and lossy compressions with 1 to 24 bits of color per pixel. A TIFF file is usually identified as a file with a '.tiff' or '.tif' file name suffix. TIFF files are used in many different image classes, including bi-level, grey-scale, palette-color, and full-color images. Interested reader can refer to Appendix B for the structure and semantics of TIFF.

5.1.2 RAW File Format

Images written in the raw file format are just a continuous string of numbers represented as a stream of bytes. The image pixels are encoded in row-by-row order, called *raster order*, perhaps with line-feeds separating rows. There is no compression, no encoding to eliminate blocks of zeros, etc.. All of the numbers are in the same format, such as integer, IEEE 8-byte real, 8-bit unsigned byte, etc.. Information such as image type, size, time taken and creation method is not part of a raw image.

5.1.3 DLL

Introduction of Dynamic Link Library

A Dynamic Link Library takes the idea of an ordinary library one step further. In Dynamic Linking, unlike Static Linking, references to function calls are not resolved until runtime. So the names of all the functions are put in a special table in executable form called function import table. This table contains all the names of the functions that need to be linked dynamically and the names of the libraries in which the function code lives.

How it works

When the program is running, the libraries (DLLs), whose names appear in the function import table, are loaded and referenced to the function calls in the DLL. They are resolved with the actual address of the function code. Only functions that appear in the DLL's export table can be linked dynamically, all other functions or data are private to the DLL. No external program has access to it.

Advantages

Modular development: A DLL increases the modularity of application development. Frequently changing code sequences, such as an algorithm, can easily be exchanged without reinstalling the entire application, just by exchanging the DLL file.

Performance: The power of C++ for calculating functions can be used in a Visual C# projects by creating a DLL file that includes the entire computational intensive part written in Visual C++.

Coding the DLL

To start, a project that would carry the DLL is needed.

Using the DLL in a C# project

Fig. 5.1 shows a short example of how to implement a new class with functionality from an external DLL.

```
public class Algorithm
{
    [DllImport(@"..\algorithm.dll")]
    public static extern int algorithm(ref int LPosX, ref int LPosY, ref int RPosX,
        ref int RPosY, byte[] img_RGB2, uint selectSkin,
        bool imageViaMemory, bool firstTimeAlgorithm,
        double threshold, bool skinPic, bool debugPic);

    //function
    private void RunAlgorithm ()
    {
        //start external algorithm
        algorithm(ref t_iLXPos, ref t_iLYPos, ref t_iRXPos, ref t_iRYPos, img_RGB2,
            m_SelectSkin, m_bImageViaMemory, m_bFirstTimeAlgorithm,
            m_threshold, m_bSkinPic, m_bDebugPics);
    }
}
```

Figure 5.1 Sample code for using DLL in C#

5.2 GUI

In this section, a description of the Graphical User Interface (GUI) will be given.

At the startup of the *Eyetracking* application and upon selection of a capturing device installed in the computer, the user interface appears as in Fig. 5.2.

Figure 5.2 GUI of Eyetracking

The first step is to start the Grabber by clicking the Start button. This action creates a new “Algorithm” thread in the class *algorithm.cs*. This thread grabs images from the webcam at the speed as indicated in the text box at the Grabber panel. When the checkbox Activ in the Algorithm panel is checked, the thread will run the algorithm and provide the detected eye position to the object *Position*. The thread can be stopped by pressing the Stop button. In the Grabber the captured image is also shown. If the eyes are found by the algorithm they would be marked by red and green crosses for the left and right eyes, respectively.

At the lower left corner of the GUI shows the computed positions of the eyes. The numbers are between 0 and 1000 proportional to the position of the eyes found on the captured picture. Two eye picture frames in this Result panel also show a closer view of the detected eyes. This function is active only when the checkbox Eye Pictures in the Algorithm panel is checked.

On the right hand side it is the Skin Color Classifier selector. The user could choose the Fixed Range method, Bayesian 2D, Bayesian 3D, and our three proposed SOTT schemes from the drop down list. The threshold value for the two Bayesian classifiers and SOTT Schemes 2 and 3 can be adjusted via the Threshold slide bar. A text box below the slide

bar indicates the current threshold value. By checking the **Show Skin Image** checkbox, a binary skin color image is shown on the left hand side of the check box.

In addition to the **Activ** and **Eye Pictures** checkboxes in the **Algorithm** panel, there is another useful development tool in the checkbox called **Debug Pictures**. Upon checking this box the intermediate processing results of eye detection are shown in the four pictures of the same panel. The status bar underneath the checkboxes indicates the processing status.

The **Cursor** panel is built for future applications. In order to control the cursor by the mouse or by the gaze direction, the *Cursor* thread has to be started by pressing the **Start** button. If the mouse should be used to control the cursor, the **Mouse Control** checkbox should be checked. The shortcut key to this function is **F5** since it might be difficult to control the cursor and move it to the desired position when the gaze direction is being used. Once the *Cursor* thread is started it takes data from the *Position* object and calculates the cursor position or movement.

The GUI is built with C# in the .NET environment. However, the algorithm part of determining the eye positions is written in C++, which is more efficient in computation and amendable to code optimization. In order to plug the algorithm into the GUI, a DLL project is created for the algorithm and the compiled *algorithm.dll* file would be called in the C# project, as shown in the sample code in Fig. 5.2. The implemented algorithms, the pre- and post-processing, and the assumptions made are discussed in the next section.

5.3 Algorithms

5.3.1 Loading Files

The function is called only once to load the necessary files for the algorithms when the GUI starts. By calling this function the pre-built 2D histogram in CbCr color space, 3D

histogram in RGB color space for the Bayesian skin color classifier, and the trained codebook for SOTT Schemes 2 and 3 in CbCr space, are loaded.

5.3.2 Pre-Processing

Cut the image to square

The size of the image captured by the webcam used is 320×240 pixels. The image would be cropped from the left and right sides by 40 pixels each.

Smoothing and Down-sampling

This step reduces the number of pixels to be processed by taking the average value of the 2x2 pixels in a block. The averaging process is like a low-pass filter to remove the noises.

5.3.3 Skin Color Classifiers

Table 5.1 shows the implemented skin color classifiers and their specifications.

Table 5.1 Implemented Skin Color Classifiers

Skin Color Classifier	Color Space	Specifications	File Required
Fixed Range	CbCr		
Bayesian 2D	CbCr	225x225 bins	Pre-built skin probability histogram in 2D
Bayesian 3D	RGB	64x64x64 bins	Pre-built skin probability histogram in 3D
SOTT Scheme 1	CbCr	2D SOTT network 64 neurons	
SOTT Scheme 2	CbCr	2D SOTT network 64 neurons	Codebook with skin probability for each neuron obtained in the training phase
SOTT Scheme 3	CbCr	2D SOTT network 64 neurons	Codebook with skin probability for each neuron obtained in the SOTT and Bayesian training phase

Fixed Range

The implemented Fixed Range method is based on Chai and Ngan [CHA99], in which a pixel is classified as skin pixel if it is within the range $Cr = [133, 173]$ and $Cb = [77, 172]$.

Bayesian

The Bayesian 2D and 3D methods are implemented in accordance to the maximum a posteriori (MAP) Bayesian rule [VEZ03]. Unlike the maximum likelihood (ML) rule, where $P(skin) = P(nonskin)$ is assumed, for the MAP rule, $P(skin)$ and $P(nonskin)$ are estimated from the training data. The probability of observing skin, given a color vector c is given by:

$$P(skin | c) = \frac{P(c | skin)P(skin)}{P(c | skin)P(skin) + P(c | nonskin)P(nonskin)} \quad (5.1)$$

$P(c | skin)$ and $P(c | nonskin)$ are directly computed from skin and nonskin color histograms. The prior probabilities, $P(skin)$ and $P(nonskin)$ can also be estimated from the overall number of skin and nonskin samples in the training set. An inequality $P(skin | c) \geq t$, where t is a threshold value, can be used as a skin detection rule.

One can avoid computing (5.1) exactly, if what is really needed is the relative probability comparison of $P(skin | c)$ and $P(nonskin | c)$. Using (5.1) the ratio of $P(skin | c)$ to $P(nonskin | c)$ can be written as:

$$\frac{P(skin | c)}{P(nonskin | c)} = \frac{P(c | skin)P(skin)}{P(c | nonskin)P(nonskin)} \quad (5.2)$$

Comparing (5.2) to a threshold produces the skin / nonskin decision rule:

$$\frac{P(c | skin)}{P(c | nonskin)} > \theta$$

$$\theta = K \times \frac{1 - P(skin)}{P(skin)} \quad (5.3)$$

This shows that the choice of the a priori probabilities does not affect the overall detector behavior –for any a priori probability, $P(skin)$, it is possible to choose an appropriate value of K , that gives the same detection threshold θ .

The Bayesian skin color classifiers have two phases, i.e., the training and testing phases. During training, two histograms are built to record the number of skin and nonskin pixels observed in each color bin, respectively. The first 2,500 images in the ECU database are used for the Bayesian training, with their manually segmented skin templates to decide whether the pixel is of skin color or not. With these two histograms, we are able to compute $P(skin | c)$:

$$P(skin | c) = \frac{N_{skin @ c}}{N_{skin @ c} + N_{nonskin @ c}} \quad (5.4)$$

where $N_{skin @ c}$ and $N_{nonskin @ c}$ denote respectively, the number of skin and nonskin pixels observed in c . In other words, (5.4) gives the fraction of the number of skin pixels observed in c over the total number of pixels observed in c .

These probabilities are then saved as a probability histogram. The implemented Bayesian 2D in CbCr space has the histogram of size 225×225 . This is because the range of Cb or Cr is [16, 240]. The implemented Bayesian 3D in RGB has a $64 \times 64 \times 64$ histogram. These histograms can be loaded and used directly for testing when the GUI starts.

In the testing phase, a proper threshold value should be chosen to determine if the observed color of a particular pixel is skin color.

SOTT Schemes

Our proposed three SOTT schemes are also implemented. Please refer to Chapter 3 for more details of the three schemes.

5.3.4 Face Extraction

After the pixel-based skin color detection is performed, our proposed Adaptive Center of Mass and Window Growing method in Chapter 4 are applied to extract the face. The first debug image in the Algorithm panel shows the extracted face from the binary skin color image shown in the SkinColorClassifier panel.

5.3.5 Eye Detection

In order to detect the eyes from the cropped facial image, we further limit our search within the upper half part of the face, which is shown in the second debug image at the Algorithm panel.

The upper facial image in gray would then be processed by the *Adaptive Threshold* function. The *Region Size Filter* and the *Region Distance Filter* are also applied to the thresholded binary image to eliminate the false eye candidates. These functions are described as follows.

5.3.5.1 Adaptive Threshold – `adaptive_threshold( )`

This function converts a gray level image into a binary image by thresholding. The algorithm is adaptive in the sense that it chooses a threshold value depending on the overall brightness of the image. The calculation of the threshold value is given by:

$$Threshold = \frac{\sum_{n=0}^{N-1} (n+1)K_n}{\sum_{n=0}^{N-1} K_n} \div FACTOR_X \quad (5.5)$$

where N is the `NUMBER_OF_GRAYSCALES`, n is the computed gray-scale, and K_n is the histogram's value of the gray-scale n (`Histogram[n]`). `FACTOR_X` is used to scale the calculated value down to a pragmatic value - it was determined empirically and set to 4 in our experiments. For simplification, the denominator in (5.5) is set equal to: `ImageLength × ImageWidth`. The third debug image shown in the Algorithm panel of the GUI is obtained after the adaptive thresholding.

Parameter: *TiffFile: image

Return value: -

Pseudo code: Figure 5.3

```
Function adaptive_threshold( )
{
    for (all gray-scale values)
    {
        calculate numerator;
        calculate denominator;
    }
    threshold = numerator / denominator / FACTOR_X;
    run threshold_image( );
}
```

Figure 5.3 Pseudo code: `adaptive_threshold( )`

5.3.5.2 Region Size Filter – `region_size_filter( )`

Regions which are too big are removed by `region_size_filter()`. It goes through all regions of an image and determines its size. If the size in the x -direction or y -direction is bigger than the pre-determined ratio of the `ImageWidth` of the upper half facial image, the region is being removed (set to `WHITE`). In our experiment the width of the eye is assumed to be less than 0.5 of the `ImageWidth` of the upper half facial image, and the

height of the eye should be less than 0.25 of the ImageWidth. The fourth debug image in the GUI is the result of the region size filter.

Parameter: *TiffFile: image

Return value: -

Pseudo code: Figure 5.4

```
Function region_size_filter( )
{
    for (all pixels of the image)
    {
        if (pixel is BLACK)
        {
            region_growing(coordinates of actual pixel);
            for (all pixels in the coordinate arrays)
            {
                find max value in Y-Direction;
                find min value in Y-Direction;
                find max value in X-Direction;
                find min value in X-Direction;
            }
            object_height = max value in Y-Direction - min value in Y-Direction;
            object_wide = max value in X-Direction - min value in X-Direction;
            if (object_height OR object_width too big)
            {
                for (all pixels in the coordinate arrays)
                {
                    remove pixel (set region to WHITE);
                }
            }
        }
    }
}
```

Figure 5.4 Pseudo code: region_size_filter()

5.3.5.3 Region Growing – region_growing()

The algorithm of region_growing() is used to grow the region starting from a black seed-pixel in a binary image. The seed-pixel has the coordinates [rCenter, cCenter]. All pixels within the region are stored in the two coordinate arrays ramr[] and ramc[].

Therefore, after region growing, these two arrays store all pixel's coordinates within the region.

Parameter: *TiffFile: image, rCenter: row center, cCenter: column center

Return value: -

Tip: Make sure a threshold is set before you do region growing.

Pseudo code: Figure 5.5

```
Function region_growing( )
{
    if (seed-pixel is within image)
    {
        if (seed-pixel = BLACK)
        {
            store coordinates of seed-pixel in the coordinate arrays;
            while (there are new pixels in coordinate arrays for further growing)
            {
                set start-pixel to the next new pixel in the coordinate arrays;
                if (seed-pixel != WHITE)
                {
                    if (pixel to the left = BLACK) {
                        save left pixel in coordinate arrays for further growing;
                        set left pixel to GRAY;    }
                    if (pixel to the top = BLACK) {
                        save top pixel in coordinate arrays for further growing;
                        set top pixel to GRAY;    }
                    if (pixel to the bottom = BLACK) {
                        save bottom pixel in coordinate arrays for further growing;
                        set bottom pixel to GRAY;    }
                    if (pixel to the right = BLACK) {
                        save right pixel in coordinate arrays for further growing;
                        set right pixel to GRAY;    }
                    set start-pixel to WHITE;
                }
            }
            set all pixels stored in the coordinate arrays to BLACK;
        }
    }
}
```

Figure 5.5 Pseudo code: region_growing()

5.3.5.4 Region Distance Filter – `region_distance_filter()`

This filter finds two regions which are separated by a distance and an angle within a given range. The `region_distance_filter()` compares all the regions in the image among each other as long as no appropriate pair of regions is found - hence the `get_regions()` function has to be run first. A lot of error checking of size and distance features for the compared regions can also be included in this function. The regions are compared from right to left and from bottom to top. This is because the lowest two regions which fit all the attributes are likely to be the pupils.

Fig. 5.6 illustrates the calculated parameters between two CoMs. The distance, C is given by $C = \sqrt{A^2 + B^2}$. For simplicity, we use the city block distance $C \approx A + B$.

The surviving pair of regions must satisfy the following conditions:

$0.3 \times \text{ImageWidth} < C < 0.7 \times \text{ImageWidth}$ (5.6)

$-15^\circ \leq \alpha \leq +15^\circ$ (5.7)

Figure 5.6 Between-region parameters

Parameter: *TiffFile: image
Return value: -
Tip: Function `get_regions()` has to be run in advance.

Pseudo code: Figure 5.7

```

Function region_distance_filter( )
{
  for (compare all regions among each other as long as no appropriate regions found)
  {
    if (0.3 ImageWidth <= A+B <= 0.7 ImageWidth)
    {
      if (the angle is within a certain range)
      {
        if (seed-pixel of region 1 is BLACK)
        {
          run region_growing( ) from seed 1;
          for (all pixels in region 1) {
            make pixels GRAY; }
        }
        if (seed-pixel of region 2 is BLACK)
        {
          for (all pixels in region 2) {
            make pixels GRAY; }
        }
        store left and right eye coordinates and leave for-loop;
      }
    }
  }
  for (all pixels of the image)
  {
    make BLACK pixels WHITE;
    make GRAY pixels BLACK;
  }
}

```

Figure 5.7 Pseudo code: region_distance_filter()

5.3.5.5 Acquire Region's Data – get_regions()

This algorithm finds regions in an image and stores several parameters (for further processing) of the regions in the 2D-array `regions_ram[ ][ ]`. The array is of the structure in Fig. 5.8. The seed-pixels are used for region growing in subsequent functions. The seed pixels have to be saved, because the center of mass (CoM) is not necessarily a BLACK pixel and therefore would not be within the region. The CoM itself is used for algorithms which make use of the distance between regions. The regions are searched from left to right and from bottom to top of the image. Row 5 in the array is used to mark

a region for later identification. The `get_regions()` marks all regions which contain more pixels than `MIN_NUM_PIX`, which is $100 / (2 \times 2)$ in our experiments, where 2×2 is the block size used for the down-sampling.

	0	1	2	...
Seed (row), which to grow from:	0			
Seed (column), which to grow from:	1			
Center of mass (row):	2			
Center of mass (column):	3			
Number of pixels within region:	4			
Marked region:	5			
Region's height:	6			
Region's width:	7			

Figure 5.8 Structure of `regions_ram[ ][ ]`

Parameter: *TiffFile: image
Return value: -
Pseudo code: Figure 5.9

```
Function get_regions( )
{
  for (all pixels of the image; from right to left and bottom to top)
  {
    if (pixel is BLACK)
    {
      run region_growing() from that pixel;
      if (number of pixel within region > certain value)
      {
        mark region as large region: regions_ram[5][ ]=1;
      }
      run center_of_mass( );
      store start seed in regions_ram[0][ ] and regions_ram[1][ ];
      store center of mass (CoM) in regions_ram[2][ ] and regions_ram[3][ ];
      store number of pixels within region in regions_ram[4][ ];
      compute height of region and store in regions_ram[6][ ];
      compute width of region and store in regions_ram[7][ ];
      set all pixels of located region to GRAY;
    }
  }
  set all GRAY pixels of the image to BLACK
}
```

Figure 5.9 Pseudo code: `get_regions()`

5.3.6 Post-Processing

After the relative eye positions in the upper facial image are obtained by running the eye detection algorithms, the absolute location of the eyes in the whole image is re-calculated and returned to the GUI for the marking of the eyes.

5.4 System Evaluation

It should be noted that the test bed has been developed for the purpose of proof of concept, and no specific emphasis and effort has been made to optimize the latency of the algorithms plugged in the GUI at this juncture. The timing analysis is carried out on a HP PC with Intel Pentium(R) 4 CPU 1.6GHz and 1GB of RAM. Table 5.2 gives a brief idea of the computation time needed by each function. The timing information has to be regarded as a rudimentary appraisal. The actual computation times for many functions are dependent on the content of the input image.

Table 5.2 Time measurements of the algorithms

Function	Time (ms)	Comment
LoadFile()	56.4	Load histogram and codebook files; Called only once at the start of the GUI.
ReadImage()	212.9	Copy image via parameter
QuadImage()	5.2	Crop the image to a square
DownSampling()	3.3	Smooth the image and down-sample with factor 2x2
SkinColorClassifier:		Only one SkinColorClassifier is needed
FixedRange()	2.5	Refer to Section 5.3.3 Fixed Range
Bayesian2()	13.9	Refer to Section 5.3.3 Bayesian
Bayesian3()	26.7	Refer to Section 5.3.3 Bayesian

SOTTScheme1()	13754.0	Refer to Chapter 3
SOTTScheme2()	460.6	Refer to Chapter 3
SOTTScheme3()	592.5	Refer to Chapter 3
AdpativeCoM()	0.25	Refer to Chapter 4
WindowGrowing()	0.26	Refer to Chapter 4
EyeDetection()	34.0	Refer to Section 5.3.5

The system can operate smoothly at 30ms interval of image capturing with the Fixed Range and Bayesian skin color classifiers. The distance between the user and the camera does not affect the detection much (see Fig. 5.10). This is because the system adopts the progressive segmentation approach. First, it extracts the face, and then zooms into the face to search for the eyes. Hence, as long as the size of the face is not unreasonably small, the distance to the camera has no observable effect on the detection results. Currently, the system can only handle the in-plane rotation to within 15 degrees (Fig. 5.11). It can handle the up and down rotation of the face to some extent (Fig. 5.12).

Figure 5.10 Effect of distance to the camera

Figure 5.11 Effect of in-plane rotation

Figure 5.12 Effect of up and down rotation of the face

5.5 Summary

In this chapter we described the Graphical User Interface (GUI) built for the purpose of evaluating different algorithms. Currently six skin color classifiers have been implemented for the testing. The eye detection part of the algorithm could also be changed easily in the DLL project to generate the *algorithm.dll* file. The debug images in the GUI enable the designer to see the intermediate results of the algorithm. This GUI eases the algorithm design and development processes considerably. The instructions on how to use the GUI for testing or designing algorithms are given in the user manual in Appendix C.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

Technologies are changing people's life and their ways of communication. We envisage the widespread use of automatic face/eye detection in biometrics, surveillance, and intelligent human computer interaction to greatly enhance the security and the quality of life. To realize this capability with reliable performance at reasonable cost, the ability to automatically detect the face/eye from the snapshots of a digital video camera is a pre-requisite. In this thesis, we have studied and proposed several methodologies for the progressive segmentation of the face and eyes from static images.

First, we explored the feasibility of employing SOTT for skin color detection. Based on the advanced features of SOTT, we experimented with the three proposed schemes of color classifiers and compared them with the existing methods. Scheme 1 trains and tests on the same image, thus it possesses an adaptive codebook for every new incoming image. However, it requires a small surplus of on-line training time and a set of explicit rules for classification. These rules would also restrict the performance of this scheme. Scheme 2 uses only a small set of images for training. Such training

could be carried out when the environment changes so that the codebook could be adapted to the new environment with different lighting condition and background, etc.. It requires less training data and memory space than the Bayesian method, but its testing process is slower than Bayesian. Its performance is not as good as Bayesian although it still outperforms several other provably good methods such as Gaussian and basic SOM. Scheme 3 has comparable performance with Bayesian when its number of neurons is also comparable to the Bayesian. It requires training data set as large as the Bayesian method. It is slower in training and testing but requires less memory than the Bayesian. These schemes provide alternatives to the existing skin color detectors for the designers to tailor to their application requirements.

After the skin color pixels have been detected, we proposed an Adaptive Center of Mass method to rapidly locate the largest pixel cluster. A Window Growing method is applied to extract a rectangular window that is just sufficient to embrace the face. For 445 images with complex background and only one face, the rate of success exceeds 87%. The same methodology could be applied to extract the eyes from the facial images. Another proposed Histogram Projection method is able to detect the eye coordinates by projecting the edge and intensity features of the eyes horizontally and vertically. The merits of this method are its simplicity and elegance. It achieved about 91% of successful detection rate for 281 upper face images without glasses. Finally, the Principal Component Analysis is applied for template matching to verify the detected eye candidates. A heuristic method for selecting an optimal set of training samples has been proposed and it has been proven to be effective in the experiments.

Besides the algorithmic development, a Graphical User Interface, which serves as a test bed to evaluate different image processing techniques for the above purposes has also been designed. This interface eases the evaluation of the algorithms in real life environments and exposes the weaknesses of some algorithms in adverse conditions. Hence, it would be useful for identifying potential trade-offs in HCI research. With little modification, it could be converted to a cursor controller for text entry or a game console by detecting and tracking the user's gaze direction.

6.2 Recommendations for Further Research

In this thesis we proposed a couple of efficient techniques for face/eye detection. Admittedly, this humble work is still far from being a complete and robust system. As usual, a few assumptions have been made to simplify the problem. Albeit the challenges, the removal of some or all of these restrictions prompts interesting future work.

(1) Detection and correction of the face rotation.

One assumption we made for the eye detection from facial images is the face is upright, though the skin color approach for face detection is rotation-invariant. The head is a three dimensional object and hence there are three types of rotation that need to be considered as shown in Fig. 6.1.

(1) **Roll** is the rotation about the axis which is perpendicular to the image plane

(2) **Yaw** is the left/right rotation.

(3) **Pitch** is the up/down rotation.

Figure 6.1 Three types of head rotations

To detect the 3D head rotation, more than one cameras might be used. Otherwise the facial features such as the eyes, the mouth, etc. have to be tracked and the changes of their relative distance have to be calculated in order to detect the rotation angle.

Mastering the rotation information not only helps detect the face and eyes, but also gives additional cue for the computer to understand the user's instructions and better control of the cursor can be supplemented by tracking human head poses.

(2) Use of the motion information for real-time tracking.

We have not considered the motion information in this thesis. In the application of Human Computer Interaction (HCI), we can assume that the background is static and the object of interest is moving. However, this is not true (the movement of the subject of interest is relatively slow compared to the background) for the driver's drowsiness monitoring system. Current face/eye detection work on the static images could be used as the re-initialization if the tracking is lost for both of the above mentioned applications. By incorporating the motion information, real-time object tracking become feasible for the HCI application. Motion estimation and motion compensation methods utilized in MPEG4 are good references for this purpose.

(3) Integration of the algorithms and the completion of the system.

The work presented in this thesis focuses more on the image processing techniques and their evaluation. If a specific application has been identified with known resource constraints and background features, appropriate system components can be chosen for the implementation based on our study. To form an efficient, robust and complete HCI system, it involves the investigation of existing algorithms, the selection of suitable techniques to fulfill the constraints, and the integration of the GUI and the logic units. The existing GUI could then be converted to serve as an interface between the human and computer with minimum modification.

Appendices

Appendix A Image Databases for Experiments and Benchmarking

A.1 AR Database

For benchmarking purpose, we use the AR face database for our experiment on eye detection problem. The AR face database was created by Aleix Martinez and Robert Benavente in the Computer Vision Center (CVC) at the U.A.B. It contains over 4,000 color images corresponding to 126 faces (70 men and 56 women). These images feature front view of faces with different facial expressions, illumination conditions, and occlusions (i.e., wearing sun glasses and scarfs) with uniform white background. The pictures were taken at the CVC under strictly controlled conditions. No restrictions on wear (clothes, glasses, etc.), make-up, hair style, etc. were imposed on the participants. Each person participated in two sessions, separated by two weeks (14 days) time frame. The same pictures were taken in both sessions. This face database is publicly available and can be obtained from the website given in [MAR98]. The database can be downloaded upon request by academics who wish to test their systems solely for educational and research purposes.

The images in the AR database are organized using the following nomenclatures: images of male and females are stored as: M-xx-yy.raw and F-xx-yy.raw, respectively, where 'xx' is a unique person identifier (from "00" to "70" for males and from "00" to "56" for females) and 'yy' indicates the session and the expression, illumination or occlusion of each image. The meanings of the number assigned to 'yy' are described as follows:

- 1 : Neutral expression
- 2 : Smile
- 3 : Anger
- 4 : Scream
- 5 : left light on
- 6 : right light on
- 7 : all side lights on
- 8 : wearing sun glasses
- 9 : wearing sun glasses and left light on
- 10 : wearing sun glasses and right light on
- 11 : wearing scarf
- 12 : wearing scarf and left light on
- 13 : wearing scarf and right light on
- 14 to 26 : second session (same conditions as 1 to 13)

For ease of exposition and analysis of the experimental results obtained in this thesis, only those AR images with open eyes and without sun glasses are used. For each person, only the first three expressions, i.e., neutral expression, smile, and anger, are selected.

A.2 ECU Database

There are a few databases available for benchmarking the performance of face detection and face recognition algorithms (usually they are with simple background for face recognition). Besides the AR Database, AT&T [SAM94], FERET [PHI00], UMIST [GRA98], and CMU [ROW98] are some popular databases used in published literatures on the face detection and recognition algorithms. However, some of them are with simple background, some provide only gray-level images, and others are either too small in terms of the number of test images, or not comprehensive or representative enough to cover most applications' need. For our experiments on face detection we have chosen the

ECU Face Detection Database [PHU03], which is unique in three aspects compared to existing face databases.

First, the database consists of color images, and therefore supports both color-based and intensity-based face detection algorithms. Second, the database consists of a large number of images (over 3,300 images) that are widely varied in terms of the image size, the background, the person, the lighting condition, the facial expression, and the face pose. Third, the database contains ground-truth images that have been manually produced.

The ECU face detection database consists for the following datasets:

- **Set 1** consists of *original color images*. The images are stored in the JPEG format. The image files are named sequentially in the form for *im#####.jpg*, where *#####* is a five-digit sequence number starting from 1.
- **Set 2** consists of the *skin segmented images* for the images in Set 1. The segmentation is done manually. In Set 2, nonskin regions are replaced with white pixels ($R = G = B = 255$). Note that we segment not only the facial skin regions but also all other exposed skin regions such as the neck, the hands, and the arms. This set is very valuable for skin color models and skin segmentation algorithms.
- **Set 3** consists of the *face segmented images* for the images in Set 1. In Set 3, nonface regions are replaced with white pixels. The face is segmented manually by following closely the face contours and not including any parts such as the hair, the ears, or background regions. This dataset can be used to evaluate face detection algorithms.
- **Set 4** consists of the *eye segmented images* for the images in Set 1. In Set 4, noneye regions are replaced with white pixels. The eye region is deemed to include the iris, the pupil, the eyelashes, and the white of eye.
- **Set 5** consists of *face patterns*. These face patterns are obtained by manually cropping the central region of the face in images obtained from various online sources. The face patterns in this set are mainly for face/nonface classification task.
- **Set 6** consists of *nonface patterns*. These patterns are extracted from nonface images in Set 7. The nonface patterns are needed for the task of classifying face and nonface.

• **Set 7** consists of *nonface images*. These images are known to contain no faces. This set consists of mostly landscape and scenery photos.

Table A.1 summarizes the datasets.

Table A.1 ECU face detection database

Set	Description	Size	File Name	File Format	Other Info.
1	Original color images	3,300	imnnnnnn.jpg	JPEG	
2	Skin segmented images	3,300	imnnnnnn_s.bmp	Bitmap*	
3	Face segmented images	3,300	imnnnnnn_f.bmp	Bitmap*	
4	Eye segmented images	3,300	imnnnnnn_e.bmp	Bitmap*	
5	Face patterns	9,125	Face_nnnnnn.bmp	Bitmap	Image size = 64 x 64
6	Nonface patterns	260000	Nonface_nnnnnn.bmp	Bitmap	Image size = 64 x 64
7	Nonface images	500	LSnnnnn.jpg	JPEG	Landscapes

* To save storage, these ground-truths can be converted into binary images.

Appendix B TIFF Structure and Semantics

B.1 TIFF Structure

A TIFF file begins with an 8-byte image file header that points to an *image file directory* - *IFD*. An IFD contains information about the image, as well as pointers to the image data itself.

Image File Header:

Please refer to Fig. B.1 for the following exposition.

Byte 0-1: The byte order used within the file. 'II' for *little-endian* notation and 'MM' for *big-endian* notation.

Byte 2-3: An arbitrary but carefully chosen number (42) that identifies the file as a TIFF file.

Byte 4-7: The offset (in bytes) of the first IFD.

Image File Directory:

There is at least one IFD in a TIFF file and each IFD has at least one entry. Each IFD defines a sub-file. Therefore it is possible to handle multiple images in one TIFF File. If there is only one IFD in a TIFF file, the *Offset to next IFD* is equal to 0.

Figure B.1 Structure of a TIFF image file

Each 12-byte IFD Entry has the following format:

- Byte 0-1: Tag that identifies the field.
- Byte 2-3: The field type.
- Byte 4-7: The number of values, count of the indicated type.
- Byte 8-11: The value offset.

To save time and space, if the value fits into 4 bytes, *the value offset* contains the value itself instead of pointing to the value. Otherwise, the *value offset* points to the value which is stored in the *Values longer than 4 bytes* section.

B.2 Bi-level Images

Bi-level images consist of only of black and white pixels. The *Field Entries* (Figure B.1 Structure of a TIFF image file) required to describe a bi-level TIFF image are listed in this subsection.

Color

The *Photometric Interpretation* field defines if the image is in the white-is-zero or black-is-zero format:

0: WhiteIsZero, 0 is imaged as white.

1: BlackIsZero, 0 is imaged as black.

Compression

Data can be stored either compressed or uncompressed:

- 1: No compression.
- 2: 1-Dimensional Modified Huffman run length encoding.
- 32773: PackBits compression, a simple byte-oriented run length scheme.

Rows and Columns

A TIFF image is organized as a rectangular array of pixels:

ImageLength: Number of rows in the image (y-coordinate).

ImageWidth: Number of columns in the image (x-coordinate).

Physical Dimensions

The size of the picture represented by the image:

- ResolutionUnit: 1: No absolute unit of measurement, 2: Inch, 3: Centimeter.
XResolution: The number of pixels per ResolutionUnit in x-coordinate.
YResolution: The number of pixels per ResolutionUnit in y-coordinate.

Location of the Data

The image data can be stored almost anywhere in a TIFF file. TIFF also supports breaking an image into separate strips for increased editing flexibility and efficient I/O buffering. The location and size of each strip is given by the following fields:

- RowsPerStrip: 1: The number of rows in a strip.
StripOffsets: The offset in bytes for a strip.
StripByteCounts: The number of bytes for a strip (after any compression).

B.3 Grayscale Images

Gray-scale images are a generalization of bi-level images. Bi-level images can only store black and white image data, but grey-scale ones can also store shades of grey. The following field is different from a bi-level image:

- BitsPerSample: The number of bits per pixel (4 or 8).

Appendix C GUI User Handbook

C.1 System Requirements

C.1.1 .NET Framework

To run this Eyetracker Interface the .NET Framework is required. For the .NET Framework Microsoft provides the following system requirements on their webpage at <http://msdn.microsoft.com/netframework/technologyinfo/sysreqs/default.aspx> (29.11.2003)

Processor

Client (a computer not working in a server capacity): 90-megahertz (MHz) Intel Pentium class processor

Server (a computer working in a server capacity): 133-MHz Intel Pentium-class processor

Operating System

The .NET Framework 1.1 Redistributable is supported on the following platforms:

- Microsoft Windows® Server 2003 (.NET Framework 1.1 is installed as part of the operating system)
- Windows XP Professional
- Windows XP Home Edition
- Windows 2000
- Windows ME
- Windows 98

- Windows NT (NT 4.0 requires Service Pack 6a)

The .NET Framework 1.1 Redistributable cannot be installed on 64-bit computers; Windows NT 4.0 Terminal Server is not supported.

Memory

Client: 32 megabytes (MB) of RAM, 96 MB recommended

Server: 128 MB of RAM, 256 MB recommended

Hard Disk

110 MB of hard disk space required, 40 MB additional hard disk space is required for installation (150 MB total)

Display

800 × 600 or higher-resolution display with 256 colors

Input Device

Microsoft mouse or compatible pointing device

Others

Microsoft Internet Explorer 5.01 or later is required.

Microsoft Data Access Components 2.6 is required for data scenarios (Microsoft Data Access Components 2.7 is recommended on a server).

Install the latest Windows service packs and critical updates from the Windows Update site.

Installation of the .NET Framework 1.1 is split into two parts: the core and language packs. The core contains everything you need to run .NET Framework applications; all dialog boxes and error messages will be in English. If you want dialog boxes and error messages in another language, you must also install the corresponding language pack.

For more information, see the .NET Framework Downloads page.

C.1.2 DirectX 9

For the DirectX 9 Microsoft provides the following system requirements on their webpage at

<http://www.microsoft.com/windows/directx/default.aspx?url=/windows/directx/productinfo/sysreqs/default.htm> (29.11.2003)

Operating System

Microsoft Windows® 98, Windows Millennium Edition (Windows Me), Windows 2000, Windows Server 2003, or Windows XP.

Hard Disk

Approximately 65 megabytes (MB) of available space is needed for installation. Once installed, the installation files can be deleted. The remaining DirectX files use approximately 18 MB of hard drive space. If an earlier version of DirectX is installed on the computer, little difference in used space on the hard drive is visible. DirectX 9.0 will overwrite the earlier versions.

C.1.3 Eyetracking Interface

This application requires a general purpose webcam, about 1MB of disk space and the .NET Framework described above. The webcam should have the resolution of 320×240 pixels for the current algorithm described in Chapter 5.

C.2 Installing the Application

Note: Install first the .NET Framework and DirectX 9!

To run the application the following 11 files are needed:

- Eyetracker.exe
- algorithm.dll
- DShowNET.dll
- bayes_2D_hist_nonskin.raw in subfolder CB_HIST
- bayes_2D_hist_skin.raw in subfolder CB_HIST
- bayes_3D_hist_nonskin.raw in subfolder CB_HIST
- bayes_3D_hist_skin.raw in subfolder CB_HIST
- HSOM2_CB.raw in subfolder CB_HIST
- HSOM3_CB.raw in subfolder CB_HIST
- HSOM3_hist_nonskin.raw in subfolder CB_HIST
- HSOM3_hist_skin.raw in subfolder CB_HIST

Copy these 11 files into any folder on a PC.

After the first run of the software the file *Settings_Data.xml* will be created. It contains all the settings for the cursor processing.

To use the "Map" function a .BMP picture has to be saved and selected in the "Settings Menu - Map".

C.3 How to create a DLL

The body of the DLL in C++ is given in the file *algorithm.cpp*. The body contains the .tiff file handler which can be used to read the grabbed image (.tiff file) into the algorithm.

Pre-condition:

- Path of the grabbed image in String / Char * format. (char *path)
- X- & Y-coordinates of every eye as reference-parameters in Integer format. (int &rLPosX, int &rLPosY, int &rRPosX, int &rRPosY)
- Flag whether or not the algorithm should write the pictures of every calculated step in the whole calculation process in Boolean format. (bool pDebugPics)

Post-condition:

- Position of the eyes in the grabbed picture in X- & Y-coordinates returned in the reference-parameters.
- Depending on the flag parameter, picture of the every intermediate algorithm step in the path of the picture string parameter.

Author's Publications

X. Deng, P. Xu, and C. H. Chang, "Self organizing topological tree for skin color detection," in Proc. of *IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 2004)*, Tainan, Taiwan, pp. 1097-1100, December 6-9, 2004.

X. Deng, C. H. Chang, and E. Braendle, "A new method for eye extraction from facial image," in Proc. of *IEEE International Workshop on Electronic Design, Test and Applications (DELTA2004)*, Perth, Australia, pp. 29-34, January 2004.

Bibliography

- [ADA94] R. Adams and L. Bischof, "Seeded region growing," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641-647, Jun. 1994.
- [ALB01] J. L. Alba, A. Pujol, "Novel SOM-PCA network for face identification," in *Proc. of Applications and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation*, San Diego, USA, pp. 186-194, Jul. 2001.
- [AUG93] M. F. Augusteijn and T. L. Skufca, "Identification of human faces through texture-based feature recognition and neural network technology," in *Proc. IEEE Conf. Neural Networks*, pp. 392-398, 1993.
- [BAK99] V. Bakic and G. Stockman, "Menu selection by facial aspect," in *Proc. Int. Conf. on Vision Interface*, Trois Rivieres, Quebec, Canada, May 1999.
<http://www.gel.ulaval.ca/~parizeau/vi99/Proceedings/node54.html#SECTION00012400000000000000>
- [BAR89] G. A. Baraghimian, "Connected component labeling using self-organizing feature maps," in *Proc. 13th Annual Int. Computer Software and Applications Conference (COMPSAC 89)*, Los Alamitos, CA, pp. 680-684, 1989.
- [BET00] M. Betke and W. J. Mullally, "Preliminary investigation of real-time monitoring of a driver in city traffic," in *Proc. IEEE Int. Symp. on Intelligent Vehicles*, Orlando, FL, pp. 563-568, Jul. 2000.
- [BRO01] D. Brown, I. Craw and J. Lewthwaite, "A SOM based approach to skin detection with application in real time systems" in *Proc. British Machine Vision Conf.*, Manchester, UK, pp. 491-500, Sep. 2001.
- [CAM00] T. E. Campos, R. S. Feris and R. M. Cesar Jr., "Eigenfaces versus eigeneyes: first steps towards performance assessment of representations for face recognition," *Lecture Notes in Artificial Intelligence*, vol. 1793, pp. 197-206, Springer-Verlag Press, April 2000.
<http://www.springer.de/comp/lncs/index.html>
- [CHA99] D. Chai and K. N. Ngan, "Face segmentation using skin color map in

- videophone applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 551-564, Jun. 1999.
- [CHA00] D. Chai and A. Bouzerdoum, "A Bayesian approach to skin color classification in YCbCr color space," in *Proc. of IEEE Region Ten Conf.*, vol. II, Kuala Lumpur, Malaysia, pp. 421-424, Sep. 2000.
- [CHA01] T. F. Chan, L. A. Vese, "Active contours without edges," *IEEE Trans. on Image Processing*, vol. 10, no. 2, pp. 266-277, Feb 2001.
<http://www.math.ucla.edu/~lvese/Publications.html>
- [CHE01] H. D. Cheng, X. H. Jiang, Y. Sun, and J. Wang, "Color image segmentation: advances and prospects," *Pattern Recognition*, vol. 34, no. 12, pp. 2259-2281, 2001.
- [DAI96] Y. Dai and Y. Nakano, "Face-texture model based on SGLD and its application in face detection in a color scene," *Pattern Recognition*, vol. 29, no. 6, pp. 1007-1017, 1996.
- [DAS02] Inference Group, Dasher Project (text-entry system).
<http://www.inference.phy.cam.ac.uk/dasher/SpecialNeeds.html>
- [DAV02] D. J. Ward and D. J. C. MacKay, "Fast hands-free writing by gaze direction," *Nature*, vol. 418, no. 6900, pp. 838, Aug. 2002.
- [DEN01] Y. Deng, and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800-810, Aug 2001.
<http://vision.ece.ucsb.edu/publications/>
- [DEN04a] X. Deng, C. H. Chang, E. Braendle, "A new method for eye extraction from facial image," in *Proc. IEEE International Workshop on Electronic Design, Test and Applications (DELTA2004)*, Perth, Australia, pp. 29-34, Jan. 2004.
- [DEN04b] X. Deng, P. Xu, and C. H. Chang, "Self organizing topological tree for skin color detection," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS2004)*, Tainan, Taiwan, pp. 1097-1100, Dec 6-9, 2004.
- [DRO01] <http://web.media.mit.edu/~nathan/research/drowsiness.htm>
- [ERI97] M. Eriksson and N. Papanikotopoulos, "Eye tracking for detection of driver fatigue," in *Proc. IEEE Int. Conf. on Intelligent Transportation Systems*, Boston,

- MA, pp. 314-319, Nov. 1997.
- [FER01] R. Feraud, O. J. Bernier, J. E. Viallet, and M. Collobert, "A fast and accurate face detector based on neural networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 42-53, Jan. 2001.
- [GAR99] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions merging and wavelet packet analysis," *IEEE Trans. on Multimedia*, vol. 1, no. 3, pp. 264-277, 1999.
- [GON92] R. Gonzalez and R. Woods, *Digital Image Processing*. Addison Wesley, 1992.
- [HSU02] R.L. Hsu, M. Abdel-Mottaleb, and A.K. Jain, "Face detection in color images," *IEEE Trans. of Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696-706, May 2002.
http://www.cse.msu.edu/~hsureinl/facloc/index_facloc.html
- [HUA02] L. L. Huang, A. Shimizu, Y. Hagihara, and H. Kobatake, "Robust face detection using a modified radial basis function network," *IEICE Trans. on Information and Systems*, vol. E85-D, no. 10, pp. 1654-1662, Oct. 2002.
- [HUN94] M. Hunke, and A. Waibel, "Face locating and tracking for human-computer interaction," in *Proc. 28th Asilomar Conf. on Signals, Systems and Computers*, Monterey, California Pittsburgh, pp. 1277-1281, 1994.
<http://citeseer.nj.nec.com/hunke94face.html>
- [GAR99] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions merging and wavelet packet analysis," *IEEE Trans. on Multimedia*, vol. 1, no. 3, pp. 264-277, Sep. 1999. <http://www.dcs.ex.ac.uk/people/wangjunl/tracking.htm>
- [GON92] R. Gonzalez and R. Woods, *Digital Image Processing*. Addison Wesley, 1992.
- [GRA98] D. B. Graham and N. M. Allinson, "Characterizing virtual eigensignatures for general purpose face recognition," in *Face Recognition: From Theory to Applications*, H. Wechsler, P. J. Phillips, V. Bruce, F. Fogelman-Soulie and T. S. Huang, Eds, NATO ASI Series F, Computer and Systems Sciences, vol. 163, pp 446-456, 1998. <http://images.ee.umist.ac.uk/danny/database.html>
- [IVI01] Intelligent Vehicle Initiative. <http://www.its.dot.gov/ivi/ivi.htm>
- [JAI00] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp.

- 4-37, Jan. 2000.
- [JAM97] C. J. James, "Detection of epileptiform activity in the electroencephalogram using artificial neural networks," Ph. D thesis, University of Canterbury, Christchurch, New Zealand, 1997.
- [JIA03] Y. Jiang, K. J. Chen, Z. H. Zhou, "SOM based image segmentation," in *Lecture Notes in Artificial Intelligence* 2639. pp. 640-643, Springer Verlag, 2003.
<http://citeseer.nj.nec.com/jiang03som.html>
- [JON99] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Ft. Collins, CO, pp. 274-280, Jun. 1999.
- [JON02] J. Lombardi and M. Betke, "A self-initializing eyebrow tracker for binary switch emulation," Boston University Computer Science Technical Report No. 2002-023. <http://www.cs.bu.edu/techreports/>
- [KAW00] T. Kawaguchi, D. Hidaka and M. Rizon, "Detection of eyes from human faces by Hough transform and separability filter," in *Proc. Int. Conf. on Image Processing*, Vancouver, BC, Canada, pp. 49-52, Sep. 2000.
- [KIM00] H. Kim, W. Kang, J. Shin and S. Park, "Face detection using template matching and ellipse fitting," *IEICE Trans. Inf. & Syst.*, vol. E38-D, no. 11, pp. 2008-2011, Nov. 2000. http://www.dcs.ex.ac.uk/people/wangjunl/e83-d_11_2008.pdf
- [KOH96] T. Kohonen, J. Hymminen, J. Kangras, and J. Laaksonan, "SOM_PAK: The self-organizing map program package," Technical Report A31, Helsinki University of Technology, 1996.
- [KOH01] T. Kohonen, *Self-Organizing Maps*, Berlin, New York: Springer-Verlag, 2001.
- [KUC02] P. Kuchi, P. Gabbur, S. Bhat, S. David, "Human face detection and tracking using skin color modeling and connected component operators", *IETE Journal of Research*, Special issue on *Visual Media Processing*, vol. 38, no. 3&4, pp. 289-293, May-Aug. 2002.
- [LUO00] H. Luo and A. Eleftheriadis, "On face detection in the compressed domain," in *Proc. ACM Int. Conf. on Multimedia*, California, USA, pp. 285-294, 2000.
- [MAI00] D. Maio and D. Maltoni, "Real-time face location on gray-scale static images," *Pattern Recognition*, vol. 33, no. 9, pp. 1525-1539, Sep. 2000.

-
- [MAR98] A. M. Martinez and R. Benavente, "The AR face database," CVC Tech. Report #24, 1998.
- [MCK98] S. McKenna, S. Gong and Y. Raja. "Modelling facial colour and identity with Gaussian mixtures," *Pattern Recognition*. vol. 31, no. 12, pp. 1883-1892, Dec. 1998. http://www.dcs.qmul.ac.uk/research/vision/face_recog.html
- [MEN00] B. Menser and M. Wien, "Segmentation and tracking of facial regions in color image sequences," in *Proc. SPIE Visual Comm. and Image Processing*, Perth, Australia, pp. 731-740, 2000.
- [MOR96] J. Moreira and L. D. F. Costa, "Neural-based color image segmentation and classification using self-organizing maps," in *Proc. Brazilian Symposium on Computer Graphics and Image Processing (IX SIBGRAPI)*, Caxambu, MG, Brasil, pp. 47-54, 1996.
- [NIP03] Neural Information Processing Group, Eötvös Loránd University, Head Mouse v.1.5 alpha. <http://nipg.inf.elte.hu/headmouse/headmouse.html>
- [OSU97] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Puerto Rico, pp. 130-136, Jun. 1997.
- [OTS78] N. Otsu, "A threshold selection method from gray level histogram," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 8, no. 1, pp. 62-66, Jan. 1979.
- [PAM00] M. Pardas, "Extraction and tracking of the eyelids," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-2000)*, Istanbul, Turkey, pp. 2357-2360, Jun. 2000.
- [PAR00] S. Park and J. K. Aggarwal, "Head segmentation and head orientation in 3D space for pose estimation of multiple people," in *Proc. 4th IEEE Southwest Symposium on Image Analysis and Interpretation*, Austin, Texas, pp. 192-196, Apr. 2000. <http://www.ece.utexas.edu/projects/cvrc/selected.htm>
- [PAU98] E. Pauwels and G. Frederix, "Finding salient regions in images: Nonparametric clustering for image segmentation and grouping," *Journal of Computer Vision and Image Understanding*, vol. 75, pp. 73-85, 1999. <http://citeseer.ist.psu.edu/pauwels98finding.html>
- [PEI98] S. C. Pei and Y. S. Lo, "Color image compression and limited display using self-

- organizing Kohonen maps,” *IEEE Trans. on Circuits and Systems for Video Techonology*, vol. 8, no. 2, pp. 191-205, 1998.
- [PET99] M. Petrou and P. Bosdogianni, *Image Processing: Fundamentals*. John Wiley & Sons, 1999.
- [PHI00] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, “The FERET evaluation methodology for face-recognition algorithms,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090-1034, 2000.
<http://www.nist.gov/humanid/feret>
- [PHU01] S.L. Phung, D. Chai, and A. Bouzerdoum, “A universal and robust human skin color model using neural networks,” in *Proc. INNS-IEEE Int. Joint Conf. on Neural Networks*, Washington DC, USA, pp. 2844-2849, Jul. 2001.
- [PHU02] S. L. Phung, D. Chai, and A. Bouzerdoum, “A distribution-based face/non-face classifier using Gaussian mixtures,” in *Proc. IASTED Int. Conf. on Signal and Image Processing*, Hawaii, USA, pp. 485-490, Aug. 2002.
- [PHU03] S. L. Phung, “Automatic human face detection in color images,” Ph. D Thesis, Edith Cowan University, Perth, Australia, 2003.
- [PIL96] M. Pilu, A. Fitzgibbon, and R. Fisher. “Ellipse-specific direct least-square fitting,” in *Proc. IEEE Int. Conf. on Image Processing*, Lausanne, Switzerland, pp. 599-602, Sep. 1996.
<http://citeseer.nj.nec.com/pilu96ellipsespecific.html>
- [RIZ00] M. Rizon and T. Kawaguchi, “Automatic eye detection using intensity and edge information,” in *Proc. IEEE TENCON 2000*, Kuala Lumpur, Malaysia, vol. 2, pp. 415-420, Sep. 2000.
- [ROW98] H.A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-28, Jan. 1998. <http://vasc.ri.cmu.edu/idb/html/face/index.html>
- [SAD02] G. Sadana, “Adaptive center of mass location in multi-clutural images,” Internal report, NTU overseas student exchange program, 2002.
- [SAM94] F. S. Samaria, “Face recognition using Hidden Markov Models,” Ph. D Thesis, Univ. of Cambridge, 1994.
- [SAN02] K. Sandeep and A.N. Rajagopalan, “Human face detection in cluttered color

- images using skin color and edge information,” in *Proc. Indian Conf. on Computer Vision, Graphics and Image Processing (ICVGIP 2002)*, India, Dec. 2002.
- [SCH95] B. Schiele and A. Waibel, “Gaze tracking based on face-color,” in *Proc. Int. Workshop on Automatic Face and Gesture Recognition*, pp. 344--349, Zurich, Switzerland, pp. 344-349, Jun. 1995.
<http://citeseer.nj.nec.com/schiele95gaze.html>
- [SET97] J. A. Sethian, “Level set methods: an act of violence,” *American Scientist*, vol. 85, no. 3, May-Jun. 1997.
- [SHA00] M. Sharma, M. Markou, and S. Singh, “Evaluation of texture methods for image analysis,” in *Proc. 7th Australian and New Zealand Intelligent Information Systems Conference*, pp. 117-121, Perth, Western Australia, 2001.
<http://www.eng.iastate.edu/ee528/Papers/texturecomparison.pdf>
- [SHI02] M. C. Shin, K. I. Chang, and L. V. Tsap, “Does colorspace transformation make any difference on skin detection?” in *Proc. IEEE Workshop on Applications of Computer Vision*, Orlando, FL, USA, pp. 275-279, Dec. 2002.
- [SIG03] L. Sigal, S. Sclaroff and V. Athitsos, “Skin color-based video segmentation under time-varying illumination,” Boston University Computer Science Technical Report No. 2003-006, March 2003, Submitted to *IEEE Trans. on Pattern Analysis and Machine Intelligence*. <http://www.cs.bu.edu/techreports/>
- [SIR93] S. A. Sirohey, “Human face segmentation and identification,” University of Maryland, Center for Automation Research, Technical Report CS-TR-3176, 1993.
- [SIR01] S. A. Sirohey and A. Rosenfeld, “Eye detection in a face image using linear and nonlinear filters,” *Pattern Recognition*, vol. 34, no. 7, pp. 1367-1391, Jul. 2001.
- [SMI97] G. Smith, Meastex Database. <http://www.cssip.uq.edu.au/meastex/meastex.html>
- [SOL03] F. Solina, P. Peer, B. Batagelj, S. Juvan, and J. Kovac, “Color-based face detection in the ‘15 seconds of fame’ art installation,” in *Proc. Mirage 2003*, INRIA Rocquencourt, France, pp. 38-47, Mar. 2003.
- [SPO01] S. Spors and R. Rabenstein, “A real-time face tracker for color video,” in *Proc. IEEE Int. Conf. on Acoustics, Speech & Signal Processing*, Utah, USA, pp. 1-4,

May 2001.

- [TAK98] K. Takaya and K. Y. Choi, "Detection of facial components in a video sequence by independent component analysis," in *Proc. 3rd Int. Conf. on Independent Component Analysis and Blind Signal Separation*, San Diego, California, USA, pp. 266-271, December 9-12, 2001.
http://ica2001.ucsd.edu/index_files/pdfs/098-takay.pdf
- [TAK01] K. Takaya and K. Y. Choi, "Detection of facial components in a video sequence by independent component analysis," in *Proc. 3rd Int. Conf. on Independent Component Analysis and Blind Signal Separation*, California, USA, pp. 266-271, December 9-12, 2001.
- [TAL99] K. Talmi and J. Liu, "Eye and gaze tracking for visually controlled interactive stereoscopic displays," *Signal Processing: Image Communication*, no. 14, pp. 799-810, 1999.
- [TER00] J. C. Terrillon, M. N. Shirazi, H. Fukamachi, and S. Akamatsu, "Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images," in *Proc. IEEE Int. Conf. on Face and Gesture Recognition*, Washington, DC, USA, pp. 54-61, Mar. 2000. <http://www.dcs.ex.ac.uk/people/wangjunl/skindetection.htm>
- [TIA00a] Y. Tian, T. Kanade and J. F. Cohn, "Dual-state parametric eye tracking," in *Proc. Int. Conf. on Face and Gesture Recognition*, Grenoble, France, pp. 110-115, Mar. 2000.
- [TIA00b] Y. Tian, T. Kanade and J.F. Cohn, "Eye state action unit detection by Gabor wavelets," in *Proc. 3rd Int. Conf. on Multi-modal Interfaces (ICMI'00)*, Beijing, China, pp. 143-150, Oct. 2000.
- [TIA02] Y. Tian, T. Kanade and J. F. Cohn, "Evaluation of Gabor-wavelet-based facial action unit recognition in image sequences of increasing complexity," in *Proc. 5th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, Washington, DC, USA, pp. 229-234, May 2002.
- [TUR91] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*, Maui, Hawaii, pp. 586-591, 1991.

- [UCH94] T. Uchiyama, M.A. Arbib, "Color image segmentation using competitive learning," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.16, no.12, pp. 1197 – 1206, Dec. 1994.
- [VEZ03] V. Vezhnevets, V. Sazonov, A. Andreeva, "A survey on pixel-based skin color detection techniques," in *Proc. Graphicon-2003*, Moscow, Russia, pp. 85-92, Sep. 2003. <http://graphics.cs.msu.su/ru/publications/>
- [VIO02] P. Viola and M. Jones, "Fast and robust classification using asymmetric AdaBoost and a detector cascade," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., Cambridge, MA: MIT Press, pp. 1311-1318, 2002.
- [WAL02] C. J. Walder, and B. C. Lovel, "Face and object recognition and detection using color vector quantisation," in *Proc. 4th Australasian Workshop on Signal Processing Applications (WOSPA-2002)*, Brisbane, Australia, pp. 27-30, Dec. 2002. <http://eprint.uq.edu.au/archive/00000203/>
- [WAN97] H. Wang, S. F. Chang, "A highly efficient system for automatic face region detection in MPEG video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 615-628, 1997.
- [WAN01] Y. Wang and B. Yuan, "A novel approach for human face detection from color images under complex background," *Pattern Recognition*, vol. 34, no. 10, pp. 1983-1992, Oct. 2001.
- [WU99] H. Wu, Q. Chen, and M. Yachida, "Face detection from color images using a fuzzy pattern matching methods," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 557-563, Jun. 1999.
<http://citeseer.ist.psu.edu/482989.html>
- [WU00] Y. Wu, Q. Liu, and T. S. Huang, "An adaptive self-organizing color segmentation algorithm with application to real-time human hand localization," in *Proc. Asian Conf. on Computer Vision*, Taiwan, pp. 1106-1111, 2000.
<http://www.ifp.uiuc.edu/~yingwu/papers/ACCV00.pdf>
- [XIE02] Y. Xie and Q. Ji, "A new efficient ellipse detection method," in *Proc. Int. Conf. on Pattern Recognition*, vol. II, pp. 957-960, 2002.
http://www.ecse.rpi.edu/homepages/qji/Papers/ellipse_det_icpr02.pdf

-
- [XU04T] P. Xu, C. H. Chang and A. Paplinski, "Self-organizing topological tree for on-line vector quantization and data clustering," Special Issue on Learning in Computer Vision and Pattern Recognition, *IEEE Transactions on Systems, Man and Cybernetics*, Part B, Accepted for publication in 2004.
- [XU04] P. Xu and C. H. Chang, "Self-organizing topological tree," in *Proc. Int. Symp. on Circuits and Systems*, Vancouver, Canada, vol. V, pp. 732-735, May 23-26, 2004.
- [YAJ96] J. Yang and A. Waibel, "A real-time face tracker," in *Proc. 3rd IEEE Workshop on Application of Computer Vision (WACV-96)*, Sarasota, Florida, USA, pp. 142-147, 1996.
http://www.ri.cmu.edu/pubs/pub_2007_text.html
- [YAJ98] J. Yang, W. Lu, and A. Waibel, "Skin-color modeling and adaptation," in *Proc. Asian Conf. on Computer Vision (ACCV-98)*, Hong Kong, vol. II, pp. 687-694, 1998. http://www.ri.cmu.edu/pubs/pub_2001.html
- [YAN98] M. H. Yang and N. Ahuja, "Detecting human faces in color images," in *Proc. Int. Conf. on Image Processing*, Chicago, Canada, vol. 1, pp. 127--130, 1998.
<http://citeseer.nj.nec.com/yang98detecting.html>
- [YAN02] M. H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34-58, Jan. 2002.
- [YIN97a] H. Yin and N. M. Allinson, "Comparison of a Bayesian SOM with the EM algorithm for Gaussian mixtures," in *Proc. of Workshop Self-Organizing Maps*, Helsinki, Finland, pp. 304-305, Jun. 1997.
- [YIN97b] H. Yin and N. M. Allinson, "Bayesian learning for self-organizing maps," *Electron. Lett.*, vol. 33, pp. 304-305, 1997.
- [YIN01] H. Yin and N. M. Allinson, "Self-organizing mixture networks for probability density estimation," *IEEE Trans. on Neural Networks*, vol. 12, no. 2, pp. 405-411, Mar. 2001.
<http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=19749&page=1>
- [YUI92] A. L. Yuille, P. W. Hallinan and D. S. Cohen, "Feature extraction from faces using deformable templates," *Int. Journal of Computer Vision*, vol. 8, no. 2, pp.

99-111, 1992.

- [ZAR99] B. D. Zarit, B. J. Super, and F. K. H. Quek, "Comparison of five color models in skin pixel classification," in *Proc. Int. Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, Corfu, Greece, pp. 58-63, Sep. 1999. <http://vislab.cs.wright.edu/Publications/>
- [ZHA00] L. Zhang and P. Lenders, "Knowledge-based eye detection for human face recognition," in *Proc. 4th Int. Conf. on Knowledge-Bases Intelligent Systems & Allied Technologies*, Brighton, UK, pp. 117-120, Aug. 2000.
- [ZHU96] S. C. Zhu and A. L. Yuille, "Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation," *IEEE Trans. on PAMI*, vol.18, no.9, pp. 884-900, 1996.
http://civs.stat.ucla.edu/Segmentation/Region_competition/region_competition.htm