ABSTRACT

IMAGE SEGMENTATION AND PATTERN CLASSIFICATION USING SUPPORT VECTOR MACHINES

by
Shouxian Cheng

Image segmentation and pattern classification have long been important topics in computer science research. Image segmentation is one of the basic and challenging lower-level image processing tasks. Feature extraction, feature reduction, and classifier design based on selected features are the three essential issues for the pattern classification problem

In this dissertation, an automatic Seeded Region Growing (SRG) algorithm for color image segmentation is developed. In the SRG algorithm, the initial seeds are automatically determined. An adaptive morphological edge-linking algorithm to fill in the gaps between edge segments is designed. Broken edges are extended along their slope directions by using the adaptive dilation operation with suitably sized elliptical structuring elements. The size and orientation of the structuring element are adjusted according to local properties.

For feature reduction, an improved feature reduction method in input and feature spaces using Support Vector Machines (SVMs) is developed. In the input space, a subset of input features is selected by the ranking of their contributions to the decision function. In the feature space, features are ranked according to the weighted support vectors in each dimension.

For object detection, a fast face detection system using SVMs is designed. Two-eye patterns are first detected using a linear SVM, so that most of the background can

be eliminated quickly. Two-layer 2nd-degree polynomial SVMs are trained for further face verification. The detection process is implemented directly in feature space, which leads to a faster SVM. By training a two-layer SVM, higher classification rates can be achieved.

For active learning, an improved incremental training algorithm for SVMs is developed. Instead of selecting training samples randomly, the *k*-mean clustering algorithm is applied to collect the initial set of training samples. In active query, a weight is assigned to each sample according to its distance to the current separating hyperplane and the confidence factor. The confidence factor, calculated from the upper bounds of SVM errors, is used to indicate the degree of closeness of the current separating hyperplane to the optimal solution.

IMAGE SEGMENTATION AND PATTERN CLASSIFICATION USING SUPPORT VECTOR MACHINES

by
Shouxian Cheng

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

January 2006

APPROVAL PAGE

IMAGE SEGMENTATION AND PATTERN CLASSIFICATION USING SUPPORT VECTOR MACHINES

Shouxian Cheng

Dr. Frank Shih, Dissertation Advisor — Date
Professor of Computer Science, NJIT

Dr. James A.M. McHugh, Committee Member — Date
Professor of Computer Science, NJIT

Dr. Denis Blackmore, Committee Member — Date
Professor of Mathematical Science, NJIT

Dr. Yun-Qing Shi, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Artur Czumaj, Committee Member — Date
Associate Professor of Computer Science, NJIT

Dr. Qun Ma, Committee Member — Date
Assistant Professor of Computer Science, NJIT

BIOGRAPHICAL SKETCH

Author: Shouxian Cheng

Degree: Doctor of Philosophy

Date: January 2006

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2006

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2001

- Bachelor of Engineering in Corrosion and Protection,
  Beijing University of Chemical Technology, Beijing, P. R. China, 1995

Major: Computer Science

Presentations and Publications:

Frank Y. Shih and Shouxian Cheng,
"Automatic seeded region growing for color image segmentation,"
Image and Vision Computing, vol. 23, no. 10, pp. 877-886, 2005.

Frank Y. Shih and Shouxian Cheng,
"Improved feature reduction in input and feature spaces,"
Pattern Recognition, vol. 38, no. 5, pp. 651-659, 2005.

Frank Y. Shih and Shouxian Cheng,
"Adaptive mathematical morphology for edge linking,"
Information Sciences, vol. 167, pp. 9-21, Dec., 2004.

Frank Y. Shih, Shouxian Cheng and Chao-Fa Chung,
"Face and eye detection using PCA and SVM in color images,"
7th Joint Conferences on Information Sciences, Cary, NC, September 2003.

N.M. Ravindra, B. Sopori, O.H. Gokce, S.X. Cheng, A. Shenoy, L. Jin, S. Abedrabbo, W. Chen, and Y. Zhang,

"Emissivity measurements and modeling of silicon-related materials: An overview,"
International Journal of Thermophysics, vol. 22, no. 5, pp. 1593-1611, 2001.

Shouxian Cheng and Frank Y. Shih,
"Face detection using two-layer support vector machines,"
IEEE Trans. Circuits and Systems for Video Technology, submitted, 2004.

Frank Y. Shih, Shouxian Cheng and Chao-Fa Chung,
"Human face and facial feature extraction,"
International Journal of Pattern Recognition and Artificial Intelligence, submitted, 2005.

Shouxian Cheng and Frank Y. Shih,
"An improved incremental training algorithm for support vector machines using active query,"
Pattern Recognition, submitted, 2005.

This dissertation is dedicated to
Linxi, my dear wife
and my beloved family

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor, Dr. Frank Shih, for his remarkable guidance, valuable insight, encouragement, and friendship throughout this research. Special thanks are given to Dr. James A.M. McHugh, Dr. Denis Blackmore, Dr. Yun-Qing Shi, Dr. Artur Czumaj, and Dr. Qun Ma for actively participating in my dissertation committee and providing valuable advice.

I extend my sincere thanks to my fellow graduate students, for all the help they have provided towards completion of this dissertation. I would especially like to thank Yi-Ta Wu, Chao-Fa Chuang, Kai Zhang, Yan-Yu Fu, Ming Qu, Peichung Shih, Gang Fu, and Yu Wang.

I wish to express my deepest gratitude to my parents for always supporting my academic pursuit and understanding me in all possible ways.

I also wish to thank my dear wife, Linxi Liao, for her love, patience, understanding and support throughout my study period.

I owe many thanks to all my friends for their friendship and support, especially to Changbo Zhou, Yanke Xu, Guilin Li, Guangjie Jiang, Bo Yu, and Xuan Zhou. I also owe limitless thanks to to Brothers and Sisters of the Church in Nutley for their unconditional and boundless love, especially to Joseph Tsai and Wenli Tsai.

TABLE OF CONTENTS

Chapter Page

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS (Continued)

.

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

Image segmentation and pattern classification have long been important topics in computer science research. Image segmentation is one of the basic and challenging lower-level image processing tasks. It is critical towards pattern classification. Feature extraction, feature reduction, and classifier design based on selected features are the three essential issues for the pattern classification problem. In this dissertation, several novel algorithms in image segmentation, feature reduction, object detection, and active learning are presented.

Image segmentation is a process of dividing an input image into subsets by classifying pixels according to their similarities (e.g., intensity, color, texture). There are primarily four types of segmentation techniques: thresholding, boundary-based, region-based, and hybrid techniques [18]. An automatic seeded region growing (SRG) algorithm for color image segmentation is developed. The initial seeds are automatically determined. A color image is segmented into regions where each region corresponds to a seed. Region merging is used to merge similar or small regions. An adaptive morphological edge-linking algorithm that fills in the gaps between edge segments is presented. Broken edges are extended along their slope directions by using the adaptive dilation operation with suitably sized elliptical structuring elements. The size and orientation of the structuring element are adjusted according to local properties.

Feature reduction is a process of selecting a subset of features with preservation or improvement of classification rates. In general, it intends to speed up the classification process by keeping the most important class-relevant features. In this dissertation, an improved feature reduction method for Support Vector Machines (SVMs) [5, 14, 67, 68] in the combinational input and feature space is proposed. In the input space, a subset of input features is selected by ranking their contributions to the decision function. In the feature space, features are ranked according to the weighted support vectors in each dimension. By combining feature selection in input and feature spaces, a fast non-linear SVM is designed without a significant loss in performance.

Automatic human face detection plays an important role in applications such as video surveillance, human computer interaction, face recognition, face tracking, and face image database management. Yang *et al.* [76] classified various face detection methods into four categories: knowledge-based, feature invariant, template matching, and appearance-based methods. In this dissertation, a fast face detection system using two-layer SVMs is designed. Using "integral images", each scanned window can be normalized quickly. Two-eye patterns are first detected using a linear SVM, so that most of the background can be eliminated quickly. Two-layer 2^{nd}-degree polynomial SVMs are trained for further face verification. The detection process is implemented directly in feature space, which leads to a faster SVM. By training a two-layer SVM, higher classification rates can be achieved.

The training procedure of SVMs usually requires huge amounts of memory space and significantly time-consuming computation because of the size of the training data and the involved quadratic programming problem. Some researchers have proposed incremental training or active learning algorithms to shorten the training time [2, 6, 39, 44, 57, 65]. In this dissertation, an incremental training algorithm for SVMs is presented.

The *k*-mean clustering algorithm is used to collect the initial training samples. In active querying, a weight is assigned to each sample according to its distance to the current separating hyperplane and a confidence factor indicating the degree that the current separating hyperplane is the optimal one. The confidence factor is obtained from the estimated generalization error of the current SVM which is calculated from the upper bound error of the SVM.

The rest of this dissertation is organized as following.

Chapter 2. Image Segmentation.

Chapter 3. Feature Reduction.

Chapter 4. Face Detection Using Two-layer Support Vector Machines.

Chapter 5. An Improved Incremental Training Algorithm for Support Vector Machines Using Active Query.

Chapter 6. Summary and Future Research.

CHAPTER 2

IMAGE SEGMENTATION

Image segmentation is one of the basic and challenging lower-level image processing tasks. An automatic seeded region growing (SRG) algorithm for color image segmentation and an adaptive morphological edge-linking algorithm are presented in this chapter.

2.1 Automatic Seeded Region Growing for Color Image Segmentation

In this section, an automatic seeded region growing algorithm for color image segmentation is presented. First, the input RGB color image is transformed into YCbCr color space. Second, the initial seeds are automatically selected. Third, the color image is segmented into regions where each region corresponds to a seed. Finally, region merging is used to merge similar or small regions. Experimental results show that this algorithm can produce better results as compared to some existing algorithms [15, 18].

2.1.1 Introduction

Image segmentation is a process of pixel classification. An image is segmented into subsets by assigning individual pixels to classes. It is an important step towards pattern detection and recognition. Pal and Pal [47] provided a review on various segmentation techniques. It should be noted that there is no single standard approach to segmentation. Many different types of scene parts can serve as the segments on which descriptions are based, and there are many different ways in which one can attempt to extract these parts from the image. Selection of an appropriate segmentation technique depends on the type of images and applications.

There are primarily four types of segmentation techniques: thresholding, boundary-based, region-based, and hybrid techniques. Thresholding is based on the assumption that clusters in the histogram correspond to either background or objects of interest that can be extracted by separating these histogram clusters [10, 11, 46, 54, 72]. Boundary-based methods assume that the pixel properties, such as intensity, color, and texture, should change abruptly between different regions [3, 36, 42, 59]. Region-based methods assume that neighboring pixels within the same region should have similar values (e.g., intensity, color, texture) [25, 27, 66]. Hybrid methods tend to combine boundary detection and region growing together to achieve better segmentation [1, 13, 15, 18, 21, 37, 41, 50, 55, 69].

Seeded Region Growing (SRG) is one of hybrid methods proposed by Adams and Bischof [1]. It starts with assigned seeds, and grows regions by merging a pixel into its nearest neighboring seed region. Mehnert and Jackway [37] pointed out that SRG has two inherent pixel order dependencies that cause different resulting segments. The first order dependency occurs whenever several pixels have the same difference measure to their neighboring regions. The second order dependency occurs when one pixel has the same difference measure to several regions. They used parallel processing and re-examination to eliminate the order dependencies. Fan *et al.* [18] presented an automatic color image segmentation algorithm by integrating color-edge extraction and seeded region growing on the YUV color space. Edges in Y, U, and V are detected by an isotropic edge detector, and the three components are combined to obtain edges. The centroids between adjacent edge regions are taken as the initial seeds. The disadvantage is that their seeds are over-generated.

In this section, the SRG is applied to color images with automatic seed selection. Strategies to avoid the two order dependencies are also developed. In Section 2.1.2, an overview of the proposed algorithm is given. In Section 2.1.3, the method for automatic seed selection is presented. Section 2.1.4 describes the proposed color SRG algorithm. Section 2.1.5 provides experimental results and discussions.

2.1.2 Overview of the Algorithm

Figure 2.1.1 presents the overview of the proposed algorithm. Firstly, the color image is transformed from RGB to YC_bC_r color space. Secondly, automatic seed selection is applied to obtain initial seeds. Thirdly, the seeded region growing algorithm is used to segment the image into regions, where each region corresponds to one seed. Fourthly, the region-merging algorithm is applied to merge similar regions, and small regions are merged into their nearest neighboring regions.

A color image is specified in RGB components. The RGB model is suitable for color display, but is not good for color analysis because of its high correlation among R, G, and B components. Besides, the distance in RGB color space does not represent the perceptual difference in a uniform scale. In image processing and analysis, these components are usually transformed into other color spaces. Cheng *et al.* [11] compared several color spaces including RGB, YIQ, YUV, normalized RGB, HIS, CIE $L^*a^*b^*$, and CIE $L^*u^*v^*$ for color image segmentation purposes. Every color space has its advantages and disadvantages. Garcia and Tziritas [19] noticed that the intensity value Y has little influence on the distribution in the C_bC_r plane and the sample skin colors form a small and very compact cluster in the C_bC_r plane. The YC_bC_r color space has been extensively used for skin color segmentation [19, 26]. In this dissertation, the YC_bC_r color space is used for segmentation due to three reasons: (1) YC_bC_r color space is widely used in video compression standards (e.g., MPEG and JPEG) [26]; (2) the color difference of human perception can be directly expressed by Euclidean distance in the color space; (3) the intensity and chromatic components can be easily and independently controlled.

The YC_bC_r color space is a scaled and offset version of YUV color space, where *Y*, *U* and *V* represent luminance, color and saturation, respectively. The C_b (C_r, respectively) is the difference between the blue (red, respectively) component and a reference value [7]. The transformation from RGB to YC_bC_r can be performed using Equation (2.1.1), where R, G, and B are in the range of [0, 1], Y is [16, 235], and C_b and C_r are in [16, 240].

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -39.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{2.1.1}$$

2.1.3 The Method for Automatic Seed Selection

For automatic seed selection, the following three criteria must be satisfied. First, the seed pixel must have high similarity to its neighbors. Second, for an expected region, at least one seed must be generated in order to produce this region. Third, seeds for different regions must be disconnected.

The similarity of a pixel to its neighbors is calculated as follows. Considering a 3×3 neighborhood, the standard deviations of Y, C_b and C_r components are calculated by using

$$\sigma_x = \sqrt{\frac{1}{9}\sum_{i=1}^{9}(x_i - \overline{x})^2}\ , \tag{2.1.2}$$

where x can be Y, C_b, or C_r, and the mean value $\overline{x} = \frac{1}{9}\sum_{i=1}^{9} x_i$. The total standard deviation is

$$\sigma = \sigma_Y + \sigma_{C_b} + \sigma_{C_r}. \tag{2.1.3}$$

The standard deviation is normalized to [0, 1] by

$$\sigma_N = \sigma / \sigma_{\max}\ , \tag{2.1.4}$$

where $\sigma_{\max}$ is the maximum of the standard deviation in the image. The similarity of a pixel to its neighbors is defined as

$$H = 1 - \sigma_N. \tag{2.1.5}$$

From the similarity, the first condition for the seed pixel candidate is defined as follows:

Condition 1: A seed pixel candidate must have a similarity higher than a threshold value.

Secondly, the relative Euclidean distances (in terms of YC_bC_r) of a pixel to its 8 neighbors are calculated as

$$d_i = \frac{\sqrt{(Y - Y_i)^2 + (C_b - C_{b_i})^2 + (C_r - C_{r_i})^2}}{\sqrt{Y^2 + C_b^2 + C_r^2}} \qquad i = 1,2,\ldots,8. \tag{2.1.6}$$

Experiments show that the performance of using relative Euclidean distance is better than using normal Euclidean distance. For each pixel, the maximum distance to its neighbors is calculated as

$$d_{\max} = \max_{i=1}^{8}(d_i). \tag{2.1.7}$$

From the maximum distance, the second condition for the seed pixel candidate is defined as

Condition 2: A seed pixel candidate must have the maximum relative Euclidean distance to its eight neighbors less than a threshold value.

A pixel is classified as a seed pixel if it satisfies the above two conditions. In order to choose the threshold value automatically for condition 1, Otsu's method [46] is used. The threshold is determined by choosing the value that maximizes the discrimination criterion σ_B^2/σ_w^2, where σ_B^2 is the between-class variance and σ_w^2 is the within-class variance. For condition 2, the value 0.05 is selected as the threshold based on the experiments.

Each connected component of seed pixels is taken as one seed. Therefore, the seeds generated can be one pixel or one region with many pixels. Condition 1 checks whether the seed pixel has high similarity to its neighbors. Condition 2 makes sure that the seed pixel is not on the boundary of two regions. It is possible that for one desired region, several seeds are detected to split it into several regions. The over-segmented regions can be merged later in the region-merging step. Figure 2.1.2(a) shows a color image, and (b) shows the detected seeds marked in red color. Note that the connected seed pixels are considered as one seed.

Regarding the threshold value for the relative Euclidean distance, if a lower value is used, a smaller number of pixels will be classified as seeds and some objects may be missed; conversely, a higher number of pixels will be classified as seeds and different regions may be connected. For example, in Figure 2.1.3, if the threshold is set as 0.04, one flower is missed in Figure 2.1.3(c) because there is no seed pixel on the flower as shown in Figure 2.1.3(b). On the other hand, if the threshold is set as 0.08, then Figure 2.1.3(f) shows that some part of the boat is merged with the water since these seed pixels are connected as shown in Figure 2.1.3(e). Note that due to variant background colors appearing in the figures, different colors are used to display the object boundaries clearly.

2.1.4 The Proposed Segmentation Algorithm

Let $A_1, A_2, \ldots, A_i$ denote initial seeds and S_i denote the region corresponding to A_i. The mean of all seed pixels in S_i in terms of *Y*, *C*

mean of all seed pixels in S_i in terms of Y, C_b and C_r components is denoted as $(\bar{Y}, \bar{C}_b, \bar{C}_r)$. The proposed segmentation algorithm is described as follows:

(1) Perform automatic seed selection.

(2) Assign a label to each seed region.

(3) Record neighbors of all regions in a sorted list *T* in a decreasing order of distances.

(4) When *T* is not empty, remove the first point *p* and check its 4-neighbors. If all labeled neighbors of *p* have a same label, set *p* to this label. If the labeled neighbors of *p* have different labels, calculate the distances between *p* and all neighboring regions and classify *p* to the nearest region. Then update the mean of this region, and add 4-neighbors of *p*, which are neither classified yet nor in *T*, to *T* in a decreasing order of distances.

(5) Perform region merging.

Note that in step 3, *T* denotes the set of pixels that are unclassified and are neighbors of at least one region S_i. The relative Euclidean distance d_i between the pixel *i* and its adjacent region is calculate by

$$d_i = \frac{\sqrt{(Y_i - \overline{Y})^2 + (C_{b_i} - \overline{C}_b)^2 + (C_{r_i} - \overline{C}_r)^2}}{\sqrt{Y_i^2 + C_{b_i}^2 + C_{r_i}^2}}, \qquad (2.1.8)$$

where $(\overline{Y}, \overline{C}_b, \overline{C}_r)$ are the mean values of Y, C_b, and C_r components in that region. In step 4, the pixel *p* with the minimum distance value is extracted. If several pixels have the same minimum value, the pixel corresponding to the neighboring region having the largest size is chosen. If *p* has the same distance to several neighboring regions, *p* is assigned to the largest region. Figure 2.1.2(c) shows the result of the proposed algorithm, where boundaries of regions are marked in white color.

Step 4 is explained graphically in Figure 2.1.4. Figure 2.1.4(a) is a small window of Figure 2.1.2(b), where the red pixels are the seed pixels and the green pixels are the 4-neighbor pixels of the seed regions. Therefore, the green pixels are actually the pixels stored in the sorted list *T*. In Figure 2.1.4(b), the white pixel is the one among all the green pixels that has the minimum distance to its adjacent regions. Therefore, the white pixel will be connected to its neighboring red seed region. In Figure 2.1.4(c), the black pixels are the three 4-neigbor pixels of the white pixel that are added to *T*. Note that these three black pixels are neither classified nor previously stored in *T*.

It is possible that several seeds are generated to split a region into several small ones. To overcome the over-segmentation problem, region merging is applied. Two criteria are used: one is the similarity and the other is the size. If the mean color difference between two neighboring regions is less than a threshold value, the two regions are merged. Unfortunately, the result of region merging depends on the order in which regions are examined. In the proposed method, the two regions having the smallest distance value among others are first examined. In each iteration, if this value is less than a threshold, the two regions are merged and the mean of the new region and the distances between the new region and its neighboring regions are recomputed. The process is repeated until no region has the distance less than the threshold. The color difference between two adjacent regions R_i and R_j is defined as the relative Euclidean distance

$$d(R_i,R_j)=\frac{\sqrt{(\overline{Y}_i-\overline{Y}_j)^2+(\overline{C}_{b_i}-\overline{C}_{b_j})^2+(\overline{C}_{r_i}-\overline{C}_{r_j})^2}}{\min(\sqrt{\overline{Y}_i^2+\overline{C}_{b_i}^2+\overline{C}_{r_i}^2},\sqrt{\overline{Y}_j^2+\overline{C}_{b_j}^2+\overline{C}_{r_j}^2})}. \qquad (2.1.9)$$

Based on experiments, the value 0.1 is selected as the threshold for color similarity measurement.

Next, the sizes of regions are checked. If the number of pixels in a region is smaller than a threshold, the region is merged into its neighboring region with the smallest color difference. This procedure is repeated until no region has size less than the threshold. Based on experiments, 1/150 of the total number of pixels in an image is selected as the threshold. Figure 2.1.2(d) shows the result of merging all similar neighboring regions, and Figure 2.1.2(e) shows the result of merging small regions.

Since variations of image complexity can be large, some cases have a high number of regions and some have small sized regions. In order to achieve better segmentation results, further merging is performed by controlling the size of regions and the color difference between regions. In the region merging, the relative Euclidean difference threshold 0.1 is set for initial merging. If this threshold is too high, some desired region may be merged with other regions; conversely, there will be too many regions. The size threshold of region is set as 1/150 of the image size. By using too high a threshold, some important objects may be merged to other regions, while too low a threshold may lead to over-segmentation. In the further merging step, the threshold value

0.2 is set for final merging because all the regions with large difference from their surroundings and with size larger than 1/150 of the image are desired to be kept. It is observed that two regions are obviously different if the relative Euclidean distance is over 0.2. Also the size threshold is set as 1/10 of the image. All the regions with difference higher than 0.1 and size larger than 1/10 of the image are desired to be kept. If this is not used, then the sky and water will be merged as shown in Figure 2.1.2(f). If the number of regions in an image is known, this information will help in improving the segmentation result; however, it is usually unknown. Therefore, in the proposed algorithm the final result is obtained by controlling the size of regions and the difference between regions.

The relative Euclidean distance is used as the merging condition because it outperforms the fixed Euclidean distance. Figure 2.1.2(d) shows that using 0.1 as the threshold value for initial merging can merge the most similar regions, while the water and sky can be separated. If 0.15 is used, then the sky and water are merged. Figure 2.1.5(b) shows the segmented result of using 0.1 as the distance threshold and 1/150 of the image size as the size threshold. Figure 2.1.5(c) shows the result of using the threshold 0.2 for further merging, which produces a reasonably good result. Figure 2.1.5(d) shows the result of using the threshold 0.15 for further merging. It can be seen that the two small regions on the right should be merged. Figure 2.1.5(e) shows the result of using the threshold 0.25 for further merging. It can be seen that the two different regions at the bottom are merged.

2.1.5 Experimental Results and Discussion

The proposed segmentation algorithm has been successfully performed on 150 color nature scene images which were randomly collected from the Internet. Figure 2.1.6

illustrates some of the results. Reasonably good results are obtained. Comparisons are also performed with some existing segmentation algorithms. Figure 2.1.7 shows the comparison with the JSEG algorithm developed by Deng *et al.* [15]. The first column shows the original color images. The second column shows the segmented results from the proposed SRG algorithm. The third column shows the results of the JSEG algorithm. Figures 2.1.7(a_3), (b_3) and (c_3) are directly downloaded from their website. Figures 2.1.7(d_3) and (e_3) are generated from the JSEG algorithm obtained from their website. Compared to Figure 2.1.7(a_2), Figure 2.1.4(a_3)'s nose of the baboon is separated into several regions and its background is over-segmented.

Figure 2.1.8 shows the comparison with the segmentation algorithm by Fan *et al.* [18]. Figures 2.1.8(a_3), (b_3), (c_3) and (d_3) are their respectively segmented results on Figures 2.1.8(a_1), (b_1), (c_1) and (d_1), and Figures 2.1.8(a_2), (b_2), (c_2) and (d_2) are the corresponding results by the proposed method. By observing Figure 2.1.8(a_3), their result is over-segmented and the hair is inappropriately merged to the background. By comparing Figures 2.1.8(b_2), (c_2), and (d_2) with Figures 2.1.8(b_3), (c_3), and (d_3), respectively, it is noticed their results are over-segmented.

The segmented results from the proposed method are less noisy as compared to Ref. [15, 18]. This is due to the fact that the proposed merging algorithm uses the dynamic control over the size of regions and the difference between regions. However, the merging step in Ref. [15, 18] uses a fixed threshold value. In Figure 2.1.7(d_3), two objects are missed in Ref. [18]. The reason is that either there is no seed generated in these objects or the merging threshold is too high, so that they are merged to the surroundings. Compared to Figure 2.1.7(e_3), Figure 2.1.7(e_2) produces more detailed and accurate boundaries. The reason is that in Ref. [15], color quantization is applied to form the class-map, so that some details of color information may be lost; however, the original color value is used by the proposed method.

There are mainly two disadvantages in the proposed algorithm. First, although using the fixed threshold values can produce reasonably good results as shown in Figures 2.1.6-2.1.9, it may not generate the best results for all the images. From Figures 2.1.7(b2), (c2), (d2), and Figure 2.1.8(a2), some small objects are missed. Some applications require more detailed information. Users can specify these threshold values according to their applications. For example, if the threshold values is adjusted for the size of objects and color differences, more detailed information can be segmented as shown in Figure 2.1.9. Second, when an image is highly color textured (i.e., there are numerous tiny objects with mixed colors), the proposed algorithm may fail to obtain satisfactory results because the mean value could not represent the property of the region well. Figure 2.1.10 shows an example where proposed algorithm performs worse than the method in Ref. [15]. Figure 2.1.10(b) shows that the tree is merged to the flower.

The time complexity for the proposed segmentation algorithm consists of three components: seed selection, region growing, and region merging. In automatic seed selection, calculating the standard deviation and maximum distance for each pixel takes $O(n)$, where n is the total number of pixels in an image. In region growing, each unclassified pixel is inserted into the sorted list exactly once. Checking neighboring regions and calculating distances can be done in constant time. Putting a pixel into the sorted list requires $\log n$. Therefore, it takes $O(n \log n)$ for region growing.

In region merging, calculating the differences between regions takes $O(m^2)$, where *m* is the number of regions. To calculate sizes for all the regions, it takes $O(n)$. Usually, *m* is much less than *n*. Merging two regions requires labeling the pixels in the two regions, calculating the mean for the new merged region, and calculating the distances between this and the other regions. Therefore, it takes $O(n+m) \approx O(n)$ to merge two regions. The complexity for region merging is $O(m^2+n+mn) \approx O(mn)$. Therefore, the total time complexity for the proposed algorithm is $O(n+n\log n+mn) \approx O((m+\log n)n)$.

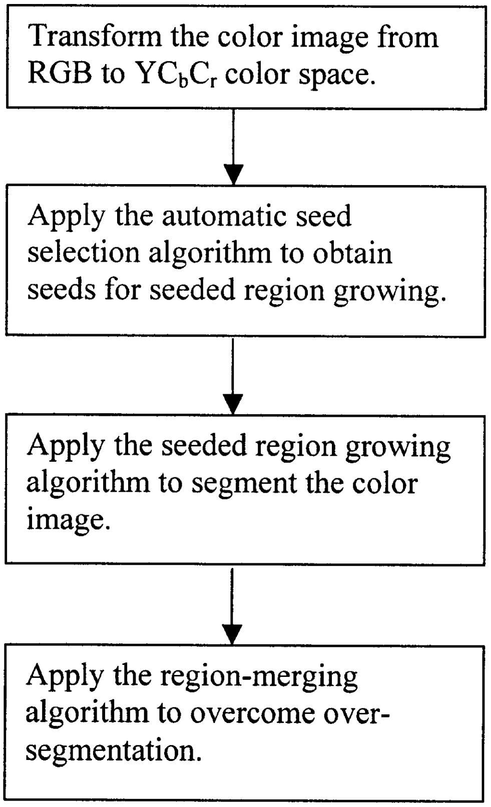

Transform the color image from RGB to YC_bC_r color space.
Apply the automatic seed selection algorithm to obtain seeds for seeded region growing.
Apply the seeded region growing algorithm to segment the color image.
Apply the region-merging algorithm to overcome over-segmentation.

Figure 2.1.1 Outline of the proposed automatic SRG algorithm.

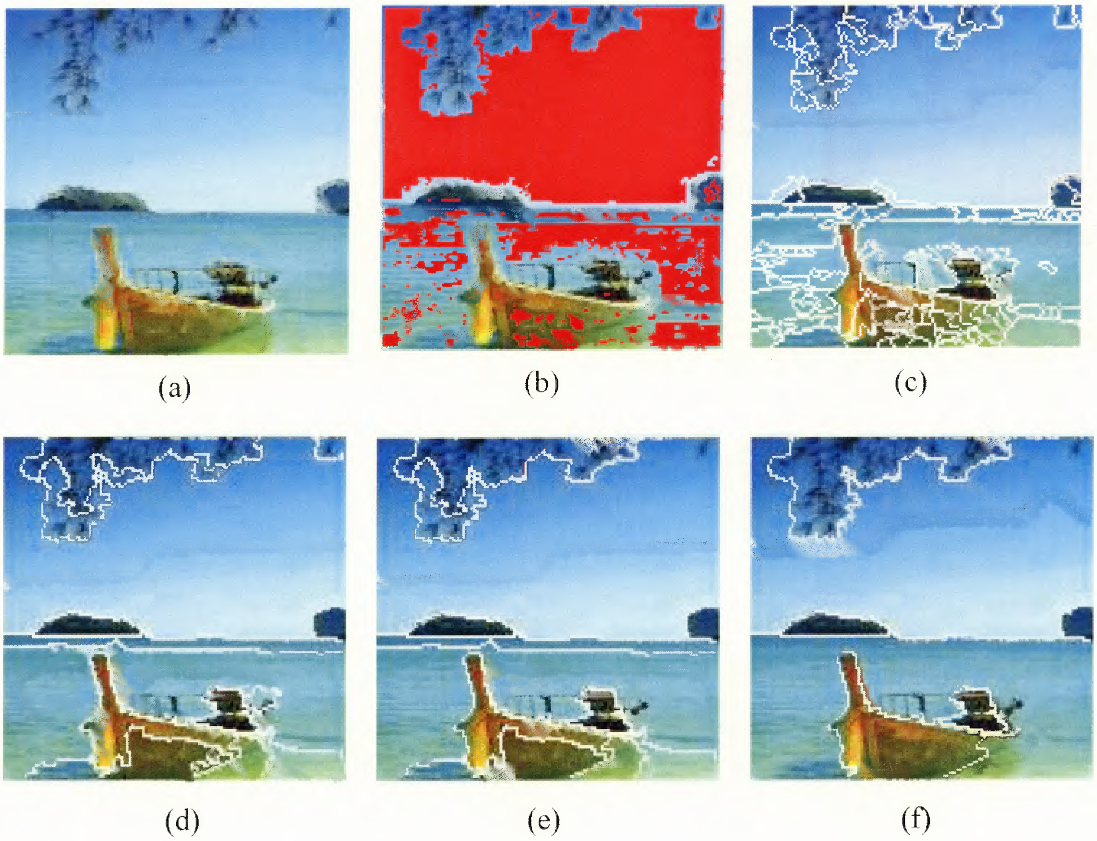


Figure 2.1.2 An example of SRG, (a) Original color image, (b) the detected seeds are shown in red color, (c) seeded region growing result, (d) the result of merging adjacent regions with relative Euclidean distance less than 0.1, (e) the result of merging small regions with size less than 1/150 of the image, and (f) final segmented result.

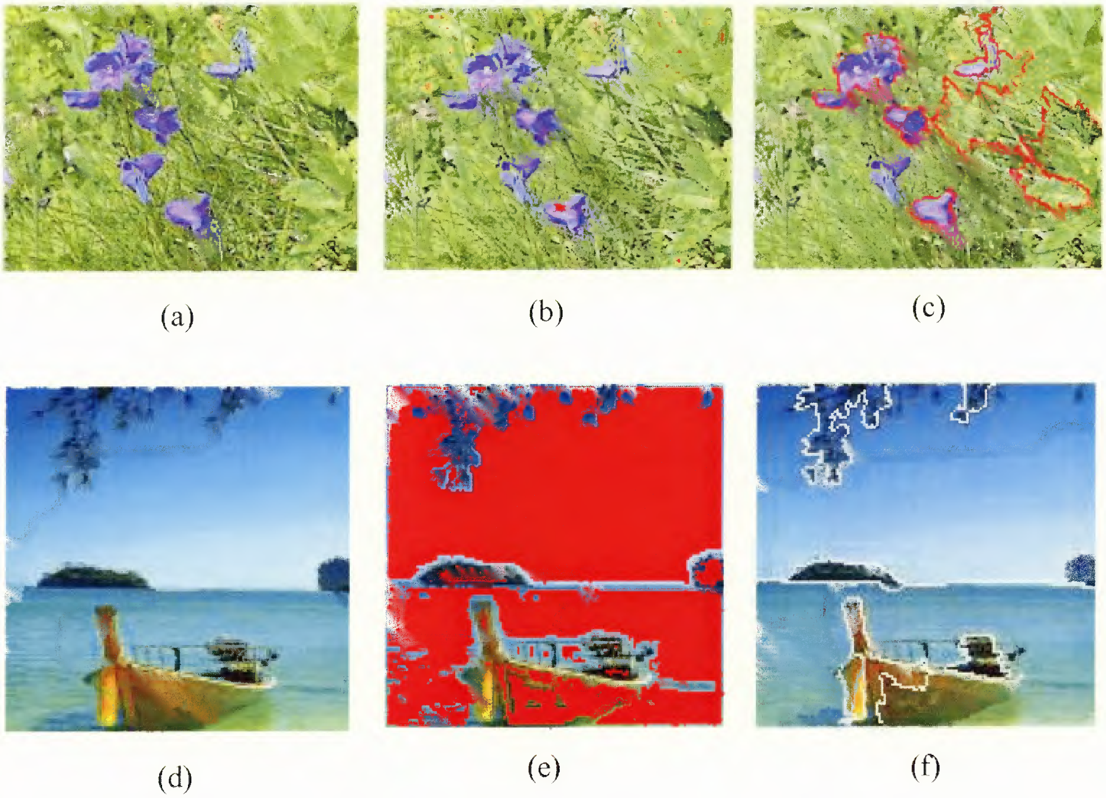


Figure 2.1.3 Examples of setting threshold for relative Euclidean distance. (a) A color image, (b) the red pixels are seed pixels generated using threshold 0.04, (c) the segmented result, (d) a color image with boat, water and sky, (e) the red pixels are the seed pixels generated using threshold 0.08, and (f) the segmented result.

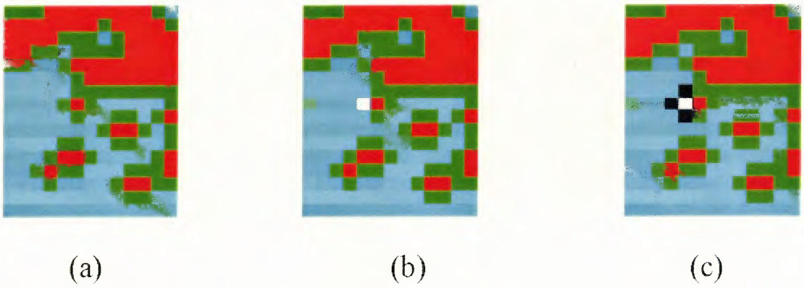
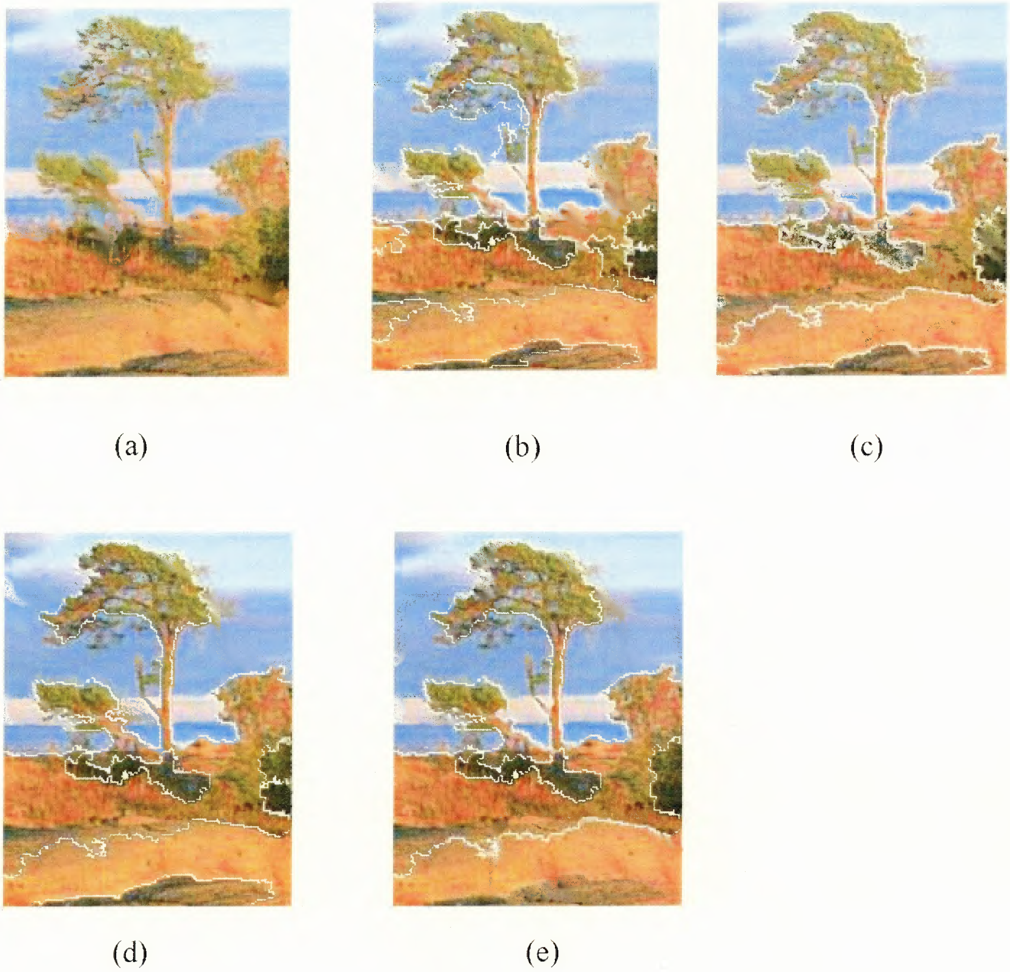


Figure 2.1.4 Step 4 of region growing. (a) A small window of Figure 2b, where the red pixels are the seeds and the green pixels are the pixels in the sorted list T, (b) the white pixel is the pixel with the minimum distance to the seed regions, and (c) the white pixel is connected to the neighboring red region and the black pixels are added to T.



(a) (b) (c)

(d) (e)

Figure 2.1.5 Example of choosing the relative Euclidean distance threshold for further region merging. (a) Original image, (b) the result after merging using 0.1 as the distance threshold value and 1/150 of the image size as the size threshold, (c) further merging using the relative Euclidean distance threshold 0.2, (d) further merging using threshold 0.15, and (e) further merging using threshold 0.25.

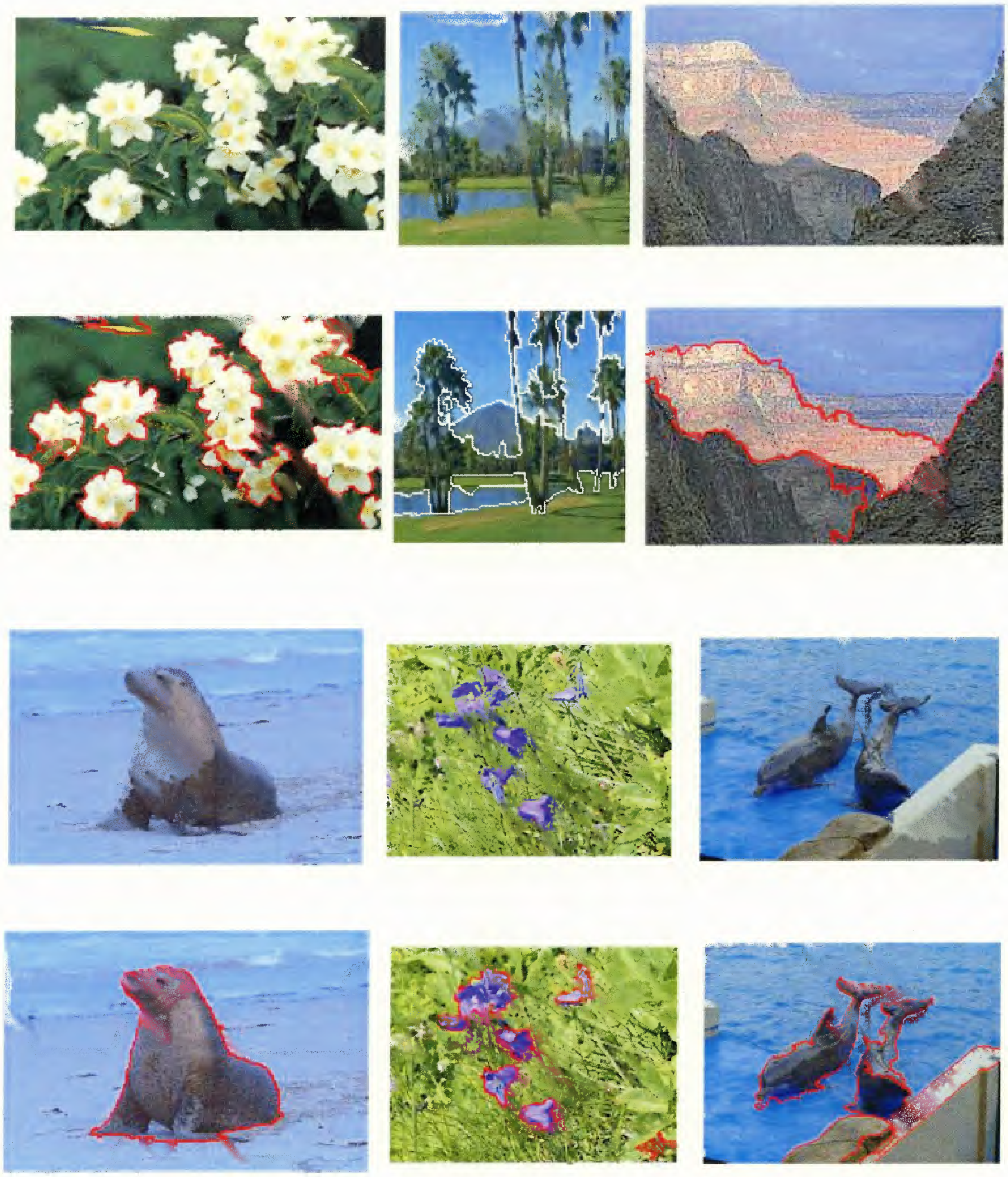
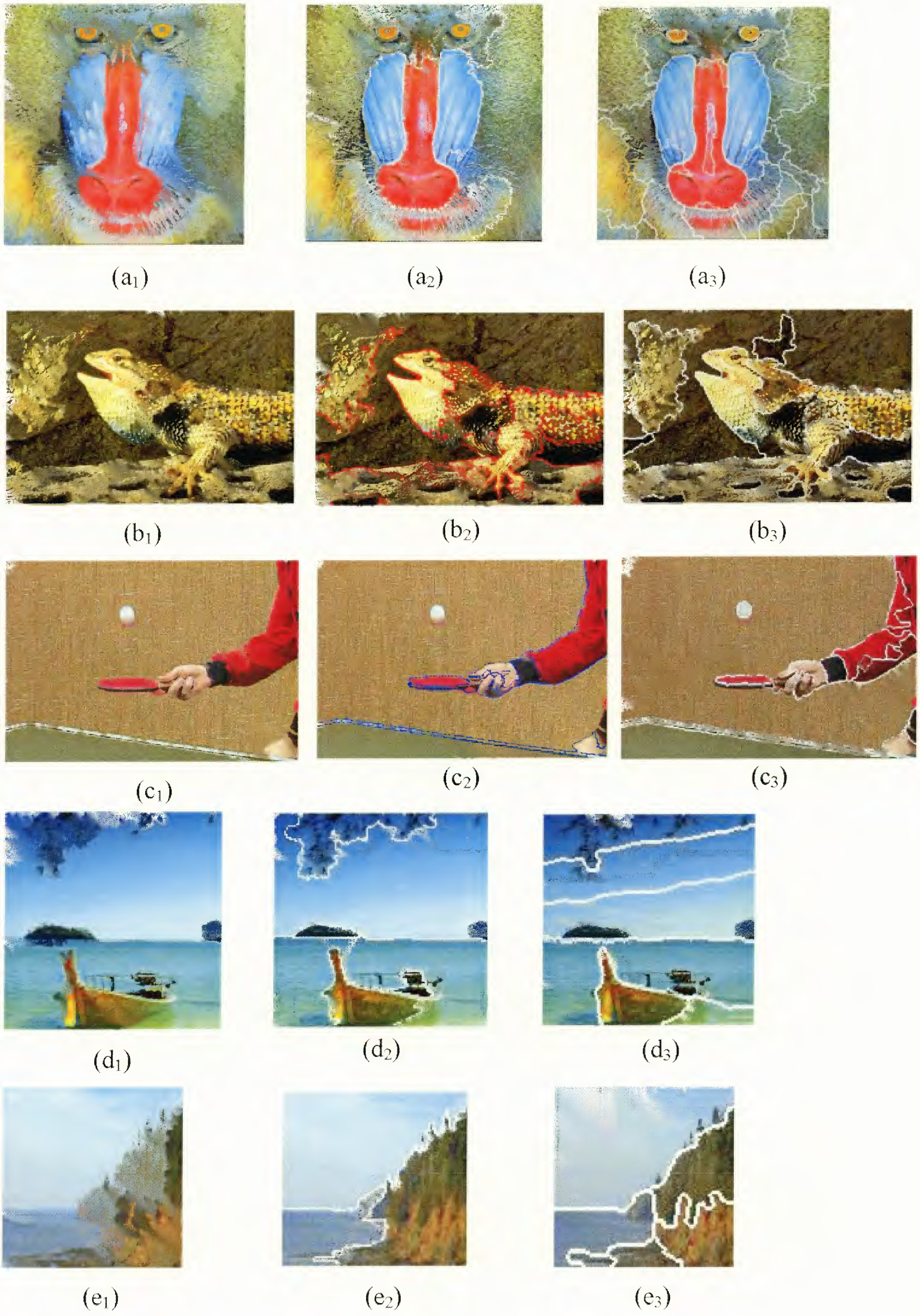


Figure 2.1.6 Some results of our color image segmentation algorithm.



(a$_1$) (a$_2$) (a$_3$)

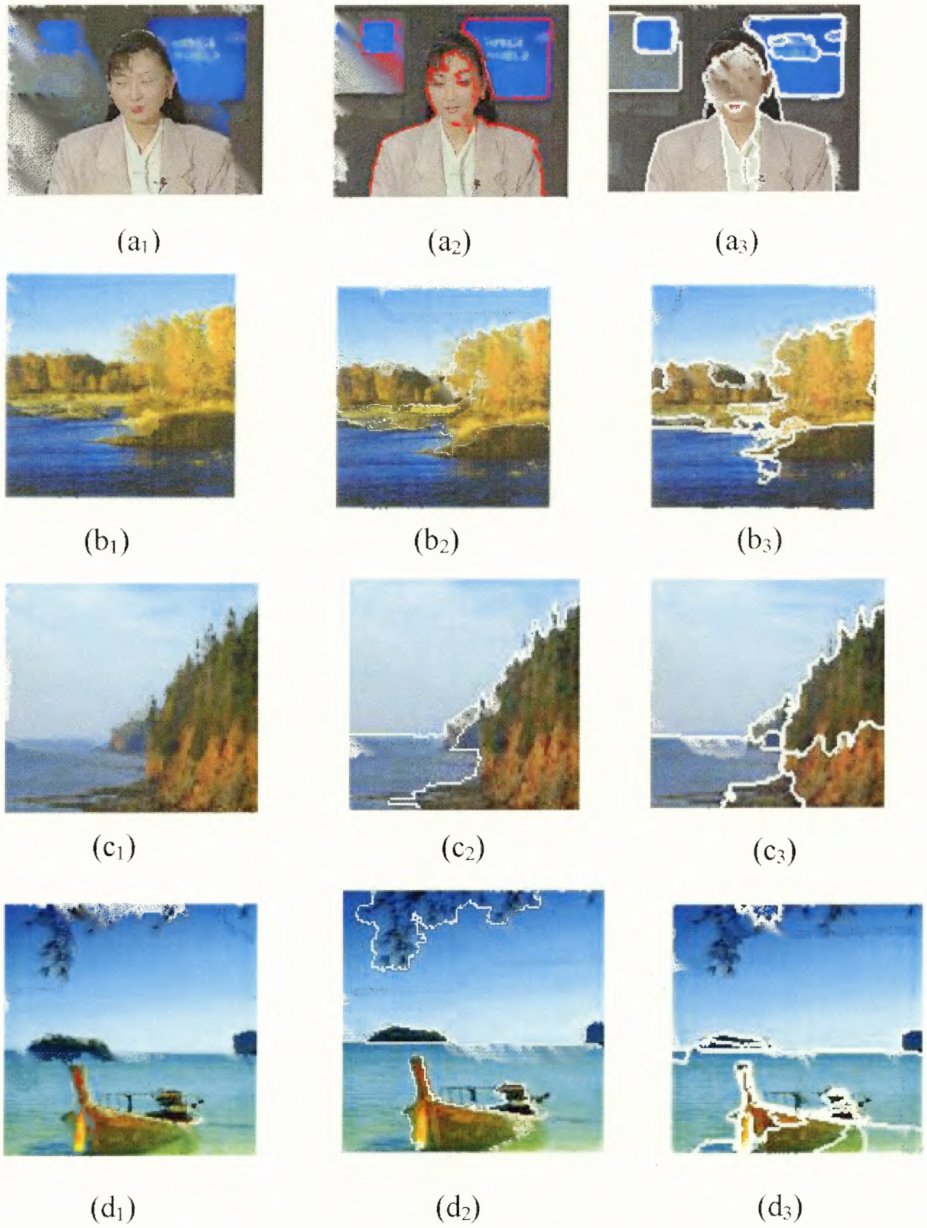
(b$_1$) (b$_2$) (b$_3$)

(c$_1$) (c$_2$) (c$_3$)

(d$_1$) (d$_2$) (d$_3$)

(e$_1$) (e$_2$) (e$_3$)

Figure 2.1.7 The first column shows the original images, the second column shows the segments of the proposed SRG method, and the third column shows the results of JSEG algorithm.



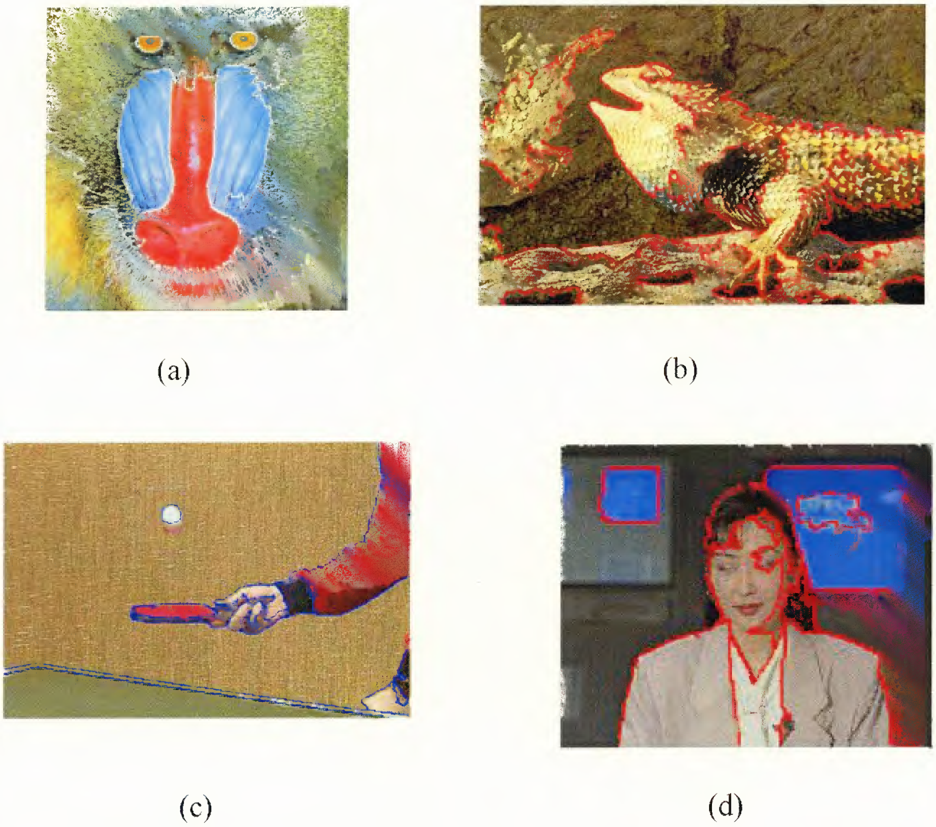
(a$_1$) (a$_2$) (a$_3$)

(b$_1$) (b$_2$) (b$_3$)

(c$_1$) (c$_2$) (c$_3$)

(d$_1$) (d$_2$) (d$_3$)

Figure 2.1.8 The first column shows the original images, the second column shows the segmented results of the proposed SRG method, and the third column shows the results of Fan's algorithm.



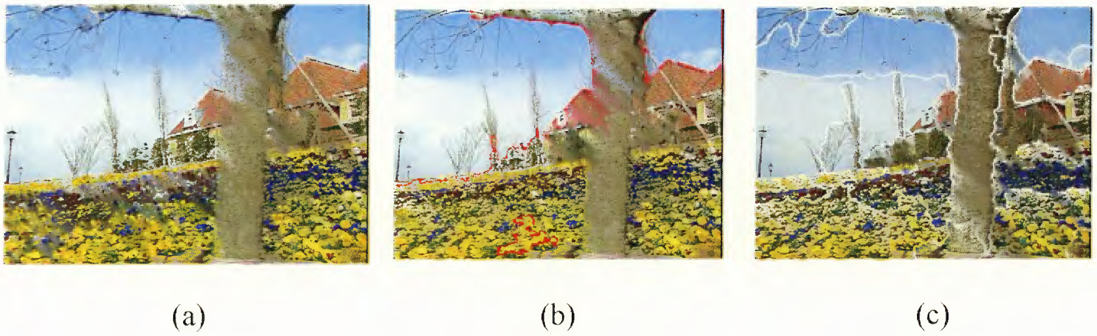
(a)

(b)

(c)

(d)

Figure 2.1.9 The segmented results with more details by adjusting the size of the minimum objects and the threshold value for the relative Euclidean distance.



(a)

(b)

(c)

Figure 2.1.10 An example shows that the proposed algorithm failed. (a) An image with highly colored texture, (b) our segmented result showing the tree is merged to the flower region, and (c) the results of Ref. [15].

2.2 An Adaptive Morphological Edge-Linking Algorithm

In this section, an adaptive morphological edge-linking algorithm is presented to fill gaps between edge segments. Broken edges are extended along their slope directions by using the adaptive dilation operation with suitably sized elliptical structuring elements. The size and orientation of the structuring element are adjusted according to local properties. Post-processes of thinning and pruning are applied. The edge-linking operation is performed in an iterative manner, and the broken edges can be linked up gradually and smoothly while the details of the object edge are preserved.

2.2.1 Introduction

Edge detection is part of processes in image segmentation. An edge is the boundary between an object and the background or between overlapping objects [49]. In an intensity image, an edge is the boundary between two adjacent, extensive regions where gray levels differ abruptly [3]. If continuous edges can be identified accurately, all the objects in the image can be recognized, and many properties such as area, perimeter and shape can be obtained.

There are tremendous a very large number of edge detection algorithms. Many edge detection methods apply neighborhood operations on each pixel to determine if it is an edge point. Some of these operations are very simple, like the gradient or Laplacian, whereas others are more complex and allow the elimination of most local noise [36]. Ideally, these operations should yield pixels which form continuous edges. In practice, this set of pixels seldom characterizes an edge completely because of noise, breaks in the edge due to non-uniform illumination, and other effects that induce spurious intensity discontinuities [20]. Thus, edge detection algorithms typically are followed by edge-linking procedures to assemble edge pixels into meaningful edges. In the approaches of edge or line linking, two assumptions are usually made [3]:

(1) True edge and line points in a scene follow some continuity pattern, whereas the noisy pixels do not follow any such continuity.

(2) The strengths of the true edge or line pixels are greater than those of the noisy pixels. The "strength" of pixels is defined differently for different applications.

Nevatia [43] presented an algorithm to link edge points by fitting straight lines. Linking direction is based on the direction of edge elements within a defined angular interval. However, portions of small curved edge segments may be neglected. Nalwa and Pauchon [42] presented an approach based upon local information. The defined *edgels* (i.e. short, linear edge elements, each characterized by a direction and a position) as well as curved segments are linked based solely on proximity and relative orientation. No global factor has been taken into consideration, and there is little concern about noise reduction. Liu *et al.* [34] proposed an edge-linking algorithm to fill gaps between edge segments. The filling operation is performed in an iterative manner rather than a single step. They first connect so-called *tip ends* (i.e., the set of edge pixels which are very likely to be the knots of some contour containing the one being considered) of two segments with a line segment and then try to modify the resulting segment by straight-line fitting. In each iteration, they define a dynamic threshold and the noises are removed gradually. For this method, it is difficult to get accurate tip ends. Another method [53] locates all of the end points of broken edges and uses a relaxation method to link them up such that line direction is maintained. Lines are not allowed to cross, and closer points are matched first. However, this suffers problems if unmatched end points or noises are present.

Mathematical morphology [22, 58, 60] has been extensively applied to image processing and analysis. Many morphological algorithms have been proposed in the areas of image enhancement, feature extraction, shape analysis, etc. A recursive soft morphological filter by Shih and Puttagunta [62] has been used to reduce noise while details are preserved. A noisy edge filtering algorithm using the so-called *space-varying* opening can be found in [1999]. A thinning algorithm using mathematical morphology is developed by Jang and Chin [28]. The algorithm is an iterative process based on the hit/miss operation. A simple approach to edge linking is a morphological dilation of endpoints by a circle followed by OR (a logic OR operation) of the boundary image with the resulting dilated circles, and skeletonization is finally applied [53]. This method, however, has the problem that some of the points may be too far apart for the circles to touch, while the circles maybe obscure details by touching several existing lines. An adaptive morphology for edge linking by using circular arcs to measure curvature can be found in [63].

Conventional morphological methods perform operations by using a fixed structuring element on all the image pixels although the size and shape of the structuring element can be arbitrarily designed. This presents problems since the local properties of input image pixels may not be identical everywhere and a fixed structuring element may not meet all purposes. In this section, an adaptive morphological edge-linking algorithm is presented. An adaptive dilation operation is applied at each endpoint with an adaptive elliptical structuring element. The size and orientation of the structuring element are adjusted according to local properties. Post-processes of thinning and pruning are finally applied.

2.2.2 The Adaptive Mathematical Morphology

A. Traditional Mathematical Morphology

Mathematical morphology provides an effective tool for image processing and analysis. It has been used for extracting image components that are useful in the representation and description of shape, such as boundaries, skeletons, and the convex hull [20]. The language of mathematical morphology is set theory. Sets in mathematical morphology represent objects in an image. The morphological operators deal with two images. The image being processed is referred to as the *active image*, and the other image being a kernel is referred to as the *structuring element*. Two basic morphological operations, dilation and erosion, are defined below:

Definition 1: Let A and B be subsets of N-dimensional Euclidean space E^N. The *dilatio*n of A by B is defined by

$$A \oplus B = \{c \in E^N \mid c = a + b \text{ for some } a \in A \text{ and } b \in B\}.$$

Definition 2: Let A and B be subsets of E^N. The *erosion* of A by B is defined by

$$A \ominus B = \{x \in E^N \mid x + b \in A \text{ for every } b \in B\}.$$

Another important operation, translation, is defined below:

Definition 3: Let A be a subset of E^N and $x \in E^N$. The *translation* of A by x is defined by

$$(A)_x = \{c \mid c = a + x \text{ for some } a \in A\}.$$

In binary image processing, the dilation of pattern A by structuring element B is to grow a translated version of B centered at each point of A and union all of the translated versions. The erosion of pattern A by B is to check the points of A where the translated version of B is completely contained in A. Dilation and erosion are fundamental to morphological processing. Many other operations are defined based on the combination of these two operations, including Opening, Closing, Hit-or-Miss Transform, Thinning, and Pruning [22, 58].

B. Adaptive Mathematical Morphology

Traditional morphological operators process an image by using a structuring element of fixed shape and size over the entire collection of image pixels. In adaptive morphology by incorporating rotation and scaling factors, the structuring elements are adjusted according to the local properties. In order to introduce the adaptive morphology, several terminologies are described [63].

Assume that the sets in consideration are always connected and bounded. Let the boundary ∂B of a set B be the set of points, all of whose neighborhoods intersect both B and its complement B^c. If a set B is connected and has no holes, it is called *simply connected.* If it is connected but has holes, it is called *multiply connected.* In this paper, the concept of ∂B is defined as the continuous boundary of a structuring element in Euclidean plane and only simply connected structuring elements are considered. Therefore, the transformations performed on the structuring element B are replaced by the transformations on its boundary ∂B in the Euclidean plane and followed by a positive filling operator. The positive filling $[\partial B]_+$ of a set B is defined as the set of points that are inside ∂B.

The adaptive morphological dilation and erosion can be defined by slightly modifying definitions 1 and 2 of the traditional operations.

Definition 4: Let A and B be subsets of E^N. The *adaptive morphological dilation* of A by a structuring element B is defined by

$$A \hat{\oplus} B = \{c \in E^N \mid c = a + \hat{b} \text{ for some } a \in A \text{ and } \hat{b} \in [(R(a)S(a)\partial B]_+\},$$

where S and R are the scaling and rotation matrices respectively.

Definition 5: Let A and B be subsets of E^N. The *adaptive morphological erosion* of A by a structuring element B is defined by

$$A \hat{\Theta} B = \{c \in E^N \mid c + \hat{b} \in A \quad \text{for every } b \in [(R(a)S(a)\partial B]_+\}.$$

The broken edge segments can be linked gradually by using adaptive morphological dilation as shown in Figure 2.2.1. Figure 2.2.1 (a) is the input signal with gaps. Figure 2.2.1 (b) shows an adaptive dilation at the endpoints. The reason for choosing the elliptical structuring element is that by using appropriate major and minor axes, all kinds of curves can be linked smoothly. Figure 2.2.1(c) shows one stage of adaptive dilation of the program.

2.2.3 The Adaptive Mathematical Morphology Edge-linking Algorithm

Removing noisy edge segments, checking endpoints, adaptive dilation, thinning, and pruning constitute a complete edge-linking algorithm. If large gaps occurr, this algorithm can be applied several times until no more endpoints exist or a predefined number of iterations has been reached.

Step 1: Removing noisy edge segments

If the length of one edge segment is shorter than a threshold value, then this segment is removed. The value 3 is used in the program.

Step 2: Detecting all the endpoints

Endpoint is defined as any pixel having only one 8-neighborhood. All the edge points are checked to extract the entire endpoint set.

Step 3: Applying adaptive dilation operation at each endpoint

From each endpoint, a range of pixels along the edge is chosen. From the properties of this set of pixels, the rotation angle and size of the elliptical structuring element are obtained. In the proposed program, increasing sizedranges of pixels are used for each iteration. This is indicated by the adjustable parameter *s* in this dissertation, where *s* denotes the size of the range of pixels from the endpoint. Two pixels from this range of pixels are taken: the left-most side p_1 (x_1, y_1) and the right-most side p_2 (x_2, y_2). The equation $slope = (y_2 - y_1) / (x_2 - x_1)$ is used to compute the slope. The rotation angle of the elliptical structuring element can be obtained by using the equation $\theta = \tan^{-1}(slope)$. The size of the elliptical structuring element is adjusted according to the number of the pixels in the set, the rotation angle θ, and the distance between p_1 and p_2. In the propsed program, the elliptical structuring elements are used with fixed $b = 3$ and a changing from 5 to 7. These values are based on experimental results. For a big gap, by using $a = 7$ the gap can be linked gradually in several iterations. For a small gap, using $a = 5$ is better than using 7. If $a = 3$ or 4 is used for a small gap, the structuring element will be almost a circle, and the thinning algorithm does not work well on the circular shape.

As known, the ellipse can be represented as

$$x^2 / a^2 + y^2 / b^2 = 1,$$

Where *a* and *b* denote, respectively, the semi-major and semi-minor axes. If the center of the ellipse is shifted from the origin to (x_0, y_0), the equation becomes

$$(x-x_0)^2/a^2+(y-y_0)^2/b^2=1.$$

If the center is at (x_0, y_0) and the rotation angle is θ ($-\pi/2<=\theta<=\pi/2$), the equation becomes:

$$((x-x_0)\cos\theta-(y-y_0)\sin\theta)^2/a^2+((x-x_0)\sin\theta-(y-y_0)\cos\theta)^2/b^2=1.$$

Because the image is discrete, the rounded off values are used in defining the elliptical structuring element. At each endpoint, an adaptive dilation is performed by using the elliptical structuring element. Therefore, the broken edge segment will be extended along the slope direction by the shape of the ellipse.

Step 4: Thinning

After applying the adaptive dilation at each endpoint, the edge segments are extended in the direction of the local slope. Because the elliptical structuring element is used, the edge segments grow a little fat. Morphological thinning is used to obtain on edge with one pixel of width.

Step 5: Branch pruning

The adaptively dilated edge segments after thinning may have noisy, short branches. These short branches must be pruned away. The resulting skeletons after thinning are one pixel in width. A *root point* is defined as a pixel having at least three pixels in an 8-neighborhood. From each endpoint, the program traces back along the existing edge. If the length of this branch is shorter than a threshold value after it reaches a root point, the branch is pruned.

Step 6: Decision

The program terminates when no endpoints exists or a predefined number of iterations have reached. Eight iterations is used as a limit in the program.

2.2.4 Experimental Results

Figure 2.2.2(a) shows an original elliptical edge and Figure 2.2.2(b) shows its randomly discontinuous edge. The edge-linking algorithm is demonstrated in Figure 2.2.2(b).

1. Using circular structuring elements

Figures 2.2.3 show the results of using circular structuring elements with r=3, r=5 and r=10, respectively, in 5 iterations. Compared with the original ellipse in Figure 2.2.2(a), it can be seen that if the gap is larger than the radius of the structuring element, it is difficult to link the gap smoothly. However, if a very big circular structuring element is used, the edge will look hollow and protuberant. Also, using a big circle can obscure the details of the edge.

2. Using a fixed sized elliptical structuring element and a fixed range of pixels to measure the slope

In this experiment, elliptical structuring elements are used with fixed semi-minor axis b=3 and semi-major axis a=5, 7, 9, respectively. A fixed range of pixels is used to measure the local slope for each endpoint. That is, the same range of pixels counting from the endpoint is used in each iteration. The parameter s denotes the range of pixels from an endpoint.

Figure 2.2.4 shows the result of using an elliptical structuring element a=5, b=3, s=7. The result is not good because a=5 is too small for some big gaps. Figure 2.2.5 shows the result of using a=7, b=*3*, and s=9. Figure 2.2.6 shows the result of using a=7, b=*3*, and s=11. There is not much difference between Figures 2.2.5 and 2.2.6. Compared with the original ellipse in Figure 2.2.2(a), Figures 2.2.5 and 2.2.6 have a few shortcomings, but they are much better than Figure 2.2.4. Figure 2.2.7 shows the result of using *a*=9, *b*=3, and *s*=11. Therefore, for this broken edge by using *a*=7 or *a*=9, a reasonably good result can be obtained except or fvery few shifted edge pixels.

3. Using a fixed sized elliptical structuring element for every endpoint but using adjustable sized ranges of pixels to measure local slope in each iteration

Figure 2.2.8 shows the result of using a=5, b=3, and adjustable s. Figure 2.2.9 shows the result of using a=7, b=3, and adjustable s. Compared with Figures 2.2.6 and 2.2.7, Figures 2.2.8 and 2.2.9 are better in terms of elliptical smoothness. This is true because after each iteration, the range of pixels used to measure local slope is increased, and more information from the original edge is taken into account.

4. Using adjustable sized elliptical structuring elements for every endpoint and using adjustable sized ranges of pixels to measure local slope in each iteration (the adaptive morphological edge-linking algorithm)

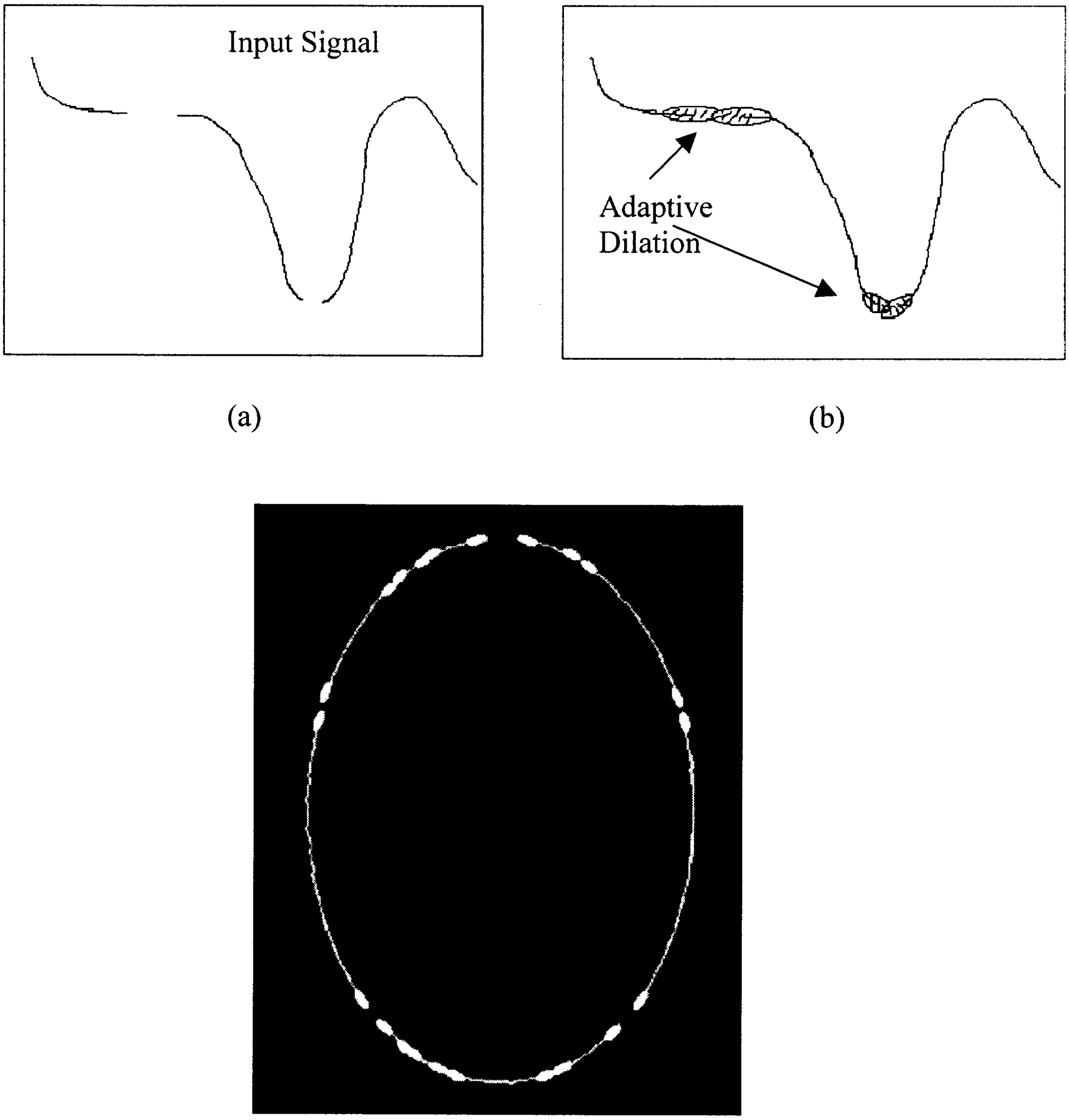
In this experiment, adjustable sized elliptical structuring elements are used with *a* changing from 5 to 7, b=3, and adjustable *s*.

Figure 2.2.10 shows the result of using the adaptive morphological edge-linking algorithm. Compared with Figures 2.2.6 and 2.2.9, Figure 2.2.10 is not obviously better. The advantage of the adaptive method is that it can adjust the parameters *a* and *s* automatically, while the parameters for Figure 2.2.6 and 2.2.9 are predefined, i.e., they work well for this case, but may not for other cases. This algorithm is also applied to many other broken elliptical edges, and for most cases the results are good. Figures 2.2.11 and 2.2.12 show the results of using this algorithm on other two randomly discontinuous elliptical edges. The shortcomings occur at places where the original broken edge segments are not smooth.

Figure 2.2.13(a) shows the elliptical edge with added uniform noise. Figure 2.2.13(b) shows the edge after removing the noise. Figure 2.2.13(c) shows the result of using the adaptive morphological edge-linking algorithm. Compared with Figure 2.2.10, Figure 2.2.13(b) has several shortcomings. This is because after removing the added noise, the edge is changed at some places.

Figure 2.2.14(a) shows the edge of an industrial part and Figure 2.2.14(b) shows its randomly discontinuous edge. Figure 2.2.14(c) shows the result of using the adaptive morphological edge-linking algorithm. Figure 2.2.15(a) shows the edge with added uniform noise and Figure 2.2.15(b) shows the edge after removing the noise. Figure 2.2.15(c) shows the result of using the adaptive morphological edge-linking algorithm.

Figure 2.2.16(a) shows a face image with the originally detected broken edge. Figure 2.2.16(b) shows the face image with the edge linked by the adaptive morphological edge-linking algorithm.

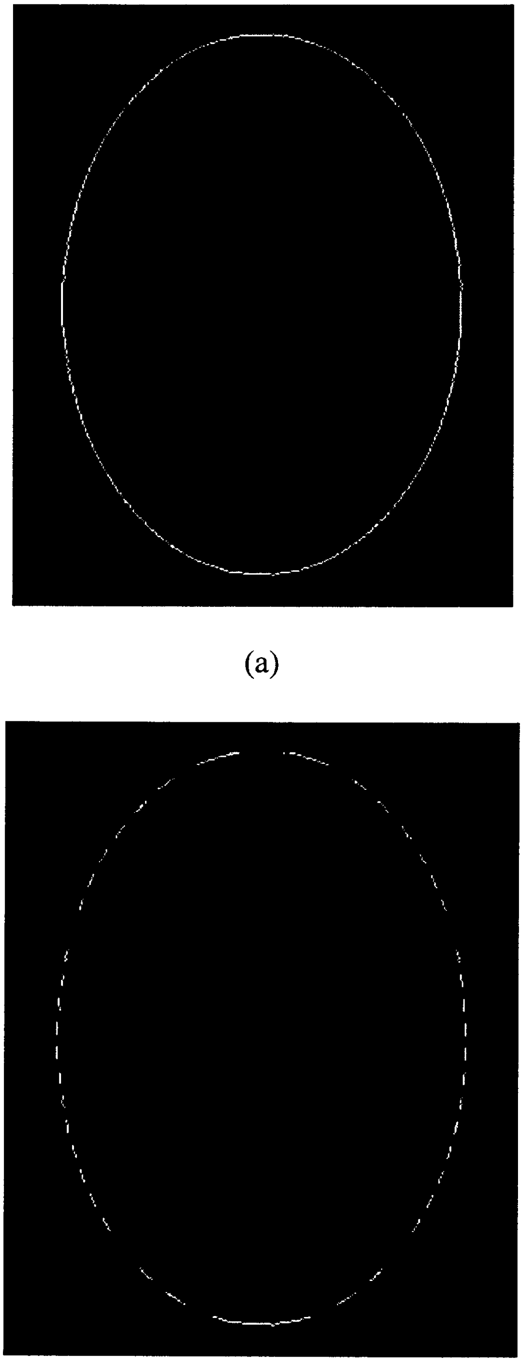


(a)

(b)

(c)

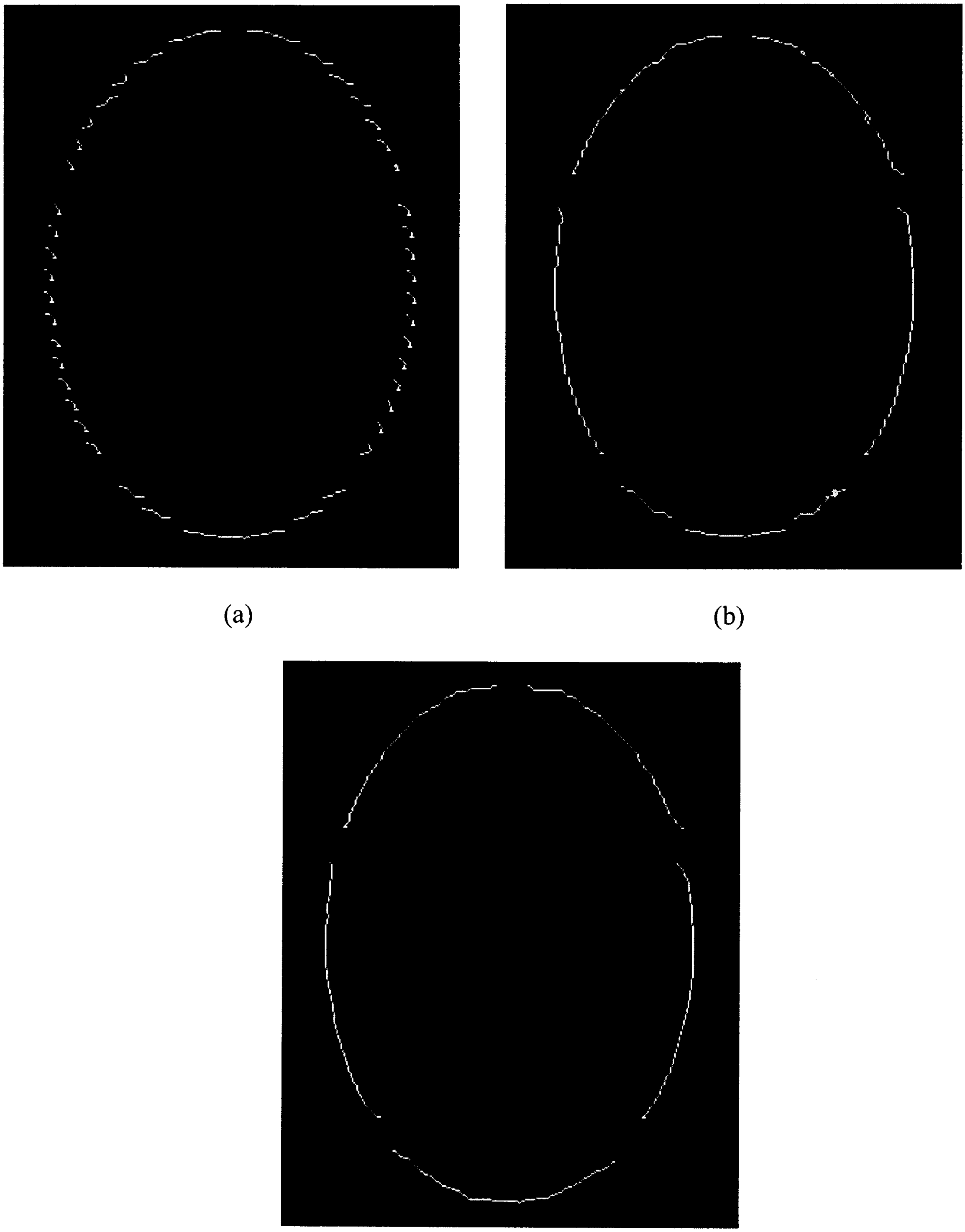
Figure 2.2.1 The broken edge segments can be linked gradually by using the adaptive morphological dilation. (a) Input signal with gaps, (b) Adaptive dilation using elliptical structuring elements, and (c) One stage of adaptive dilation of our program.



(a)

(b)

Figure 2.2.2 (a) Original elliptical edge, and (b) Its randomly discontinuous edge.



(a)

(b)

(c)

Figure 2.2.3 Using circular structuring elements in 5 iterations with (a) *r*=3, (b) *r*=5, and (c) *r*=10.

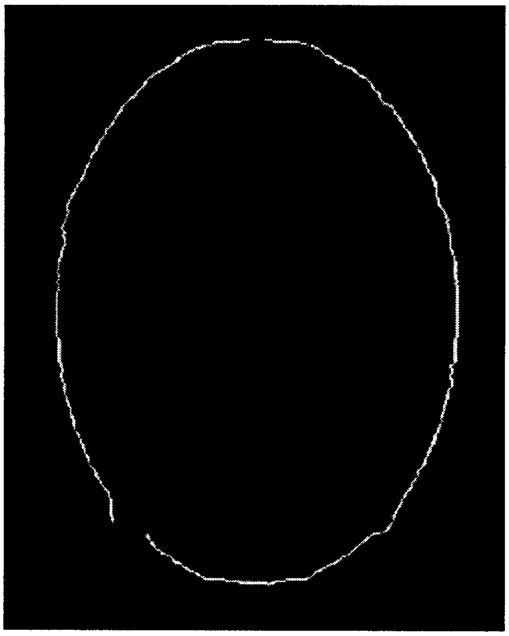


Figure 2.2.4 An elliptical structuring element with *a*=5, *b*=3, and *s*=7.

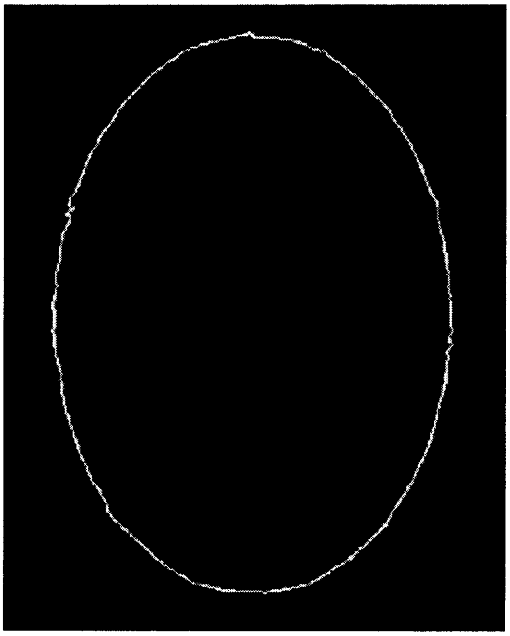


Figure 2.2.5 An elliptical structuring element with *a*=7, *b*=3, and *s*=9.

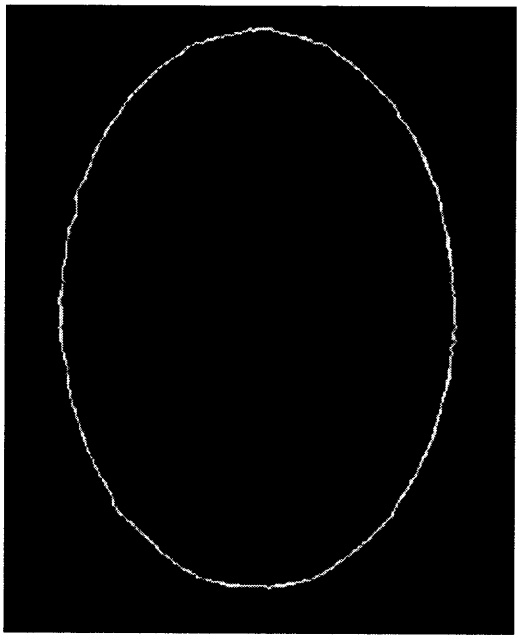


Figure 2.2.6 An elliptical structuring element with *a*=7, *b*=3, and *s*=11.

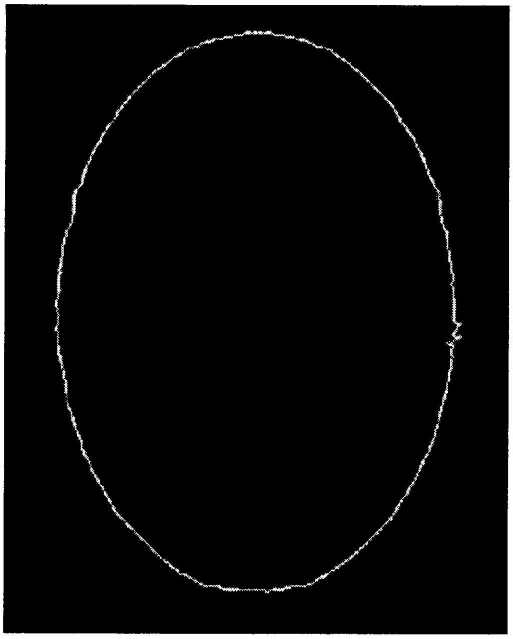


Figure 2.2.7 An elliptical structuring element with *a*=9, *b*=3, and *s*=11.

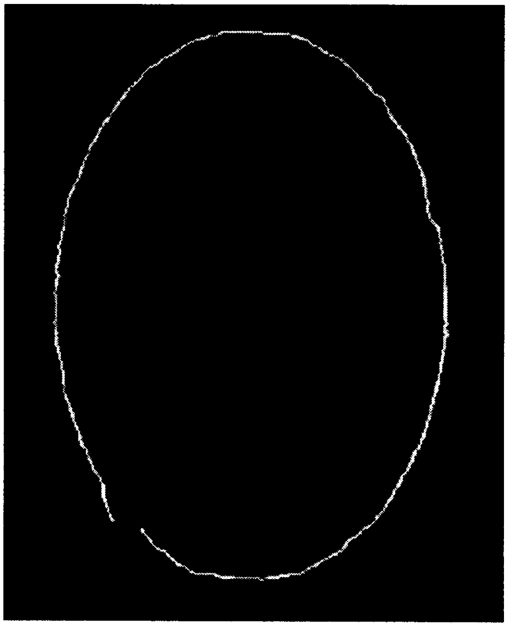


Figure 2.2.8 Using an elliptical structuring element with a=5, b=3, and adjustable s.

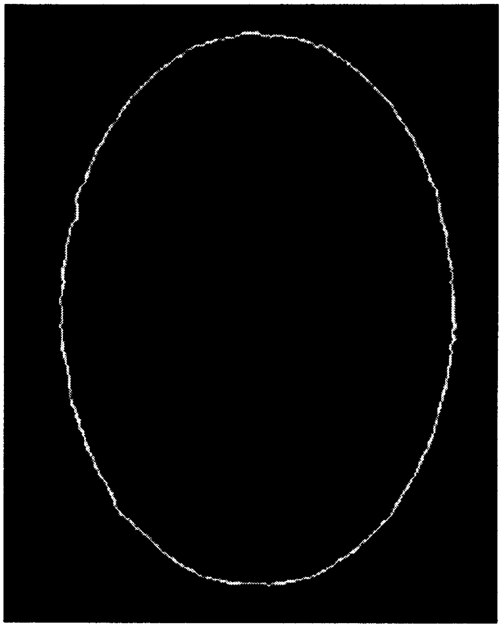


Figure 2.2.9 Using an elliptical structuring element with a=7, b=3, and adjustable s.

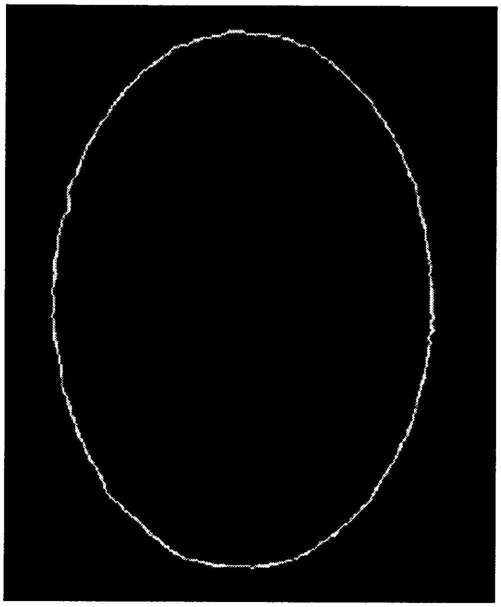
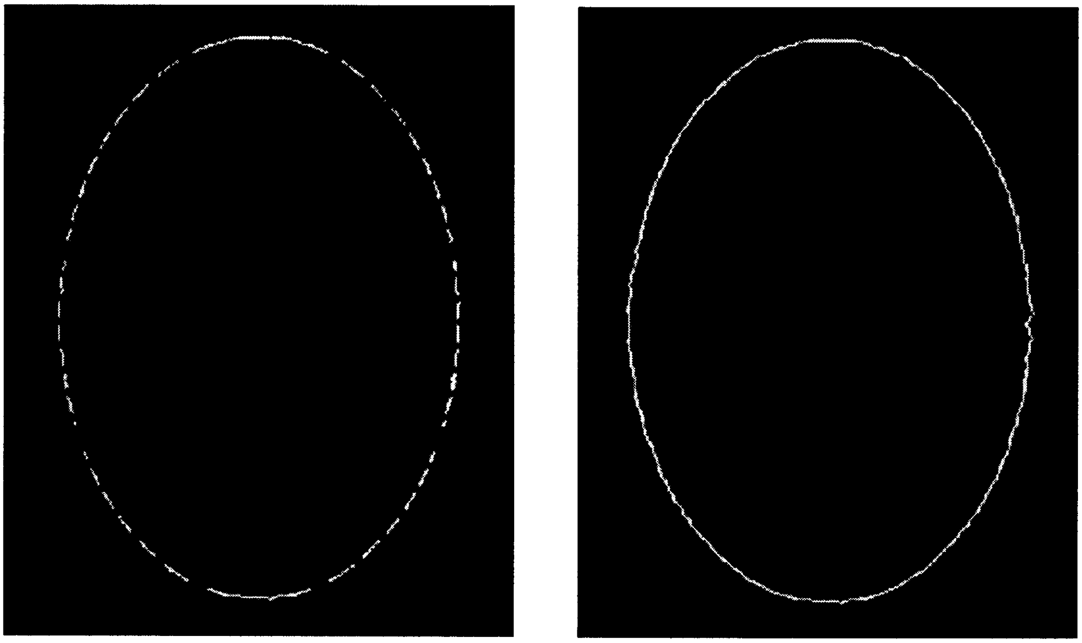


Figure 2.2.10 Using the adaptive morphological edge-linking algorithm.



(a) (b)

Figure 2.2.11 (a) Another randomly discontinuous elliptical edge, and (b) Using the adaptive morphological edge-linking algorithm.

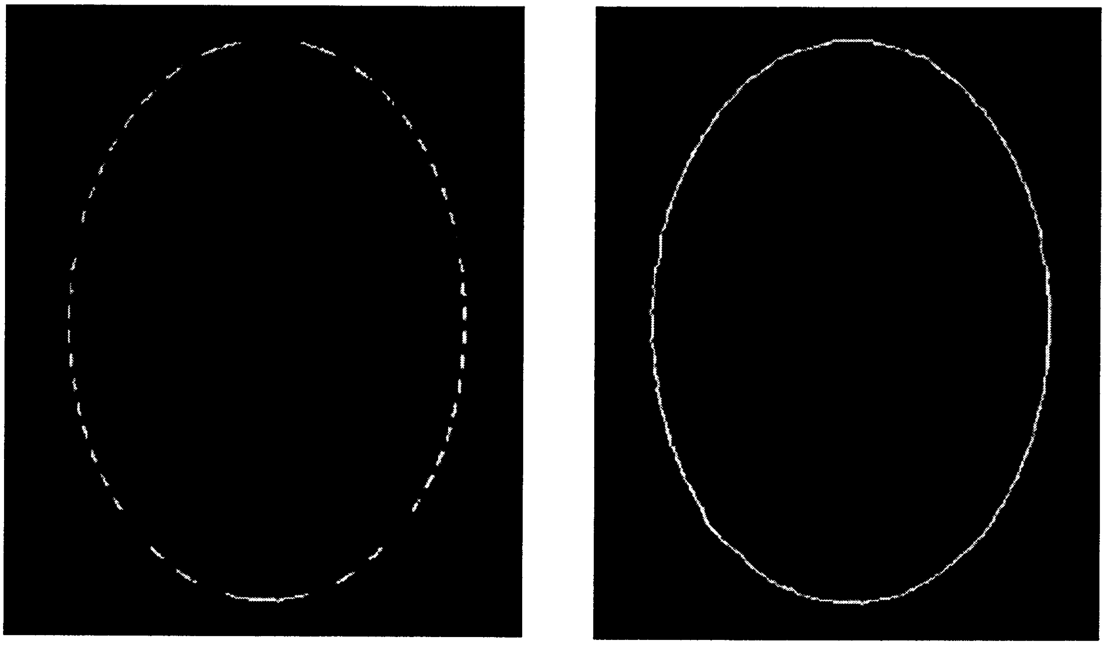
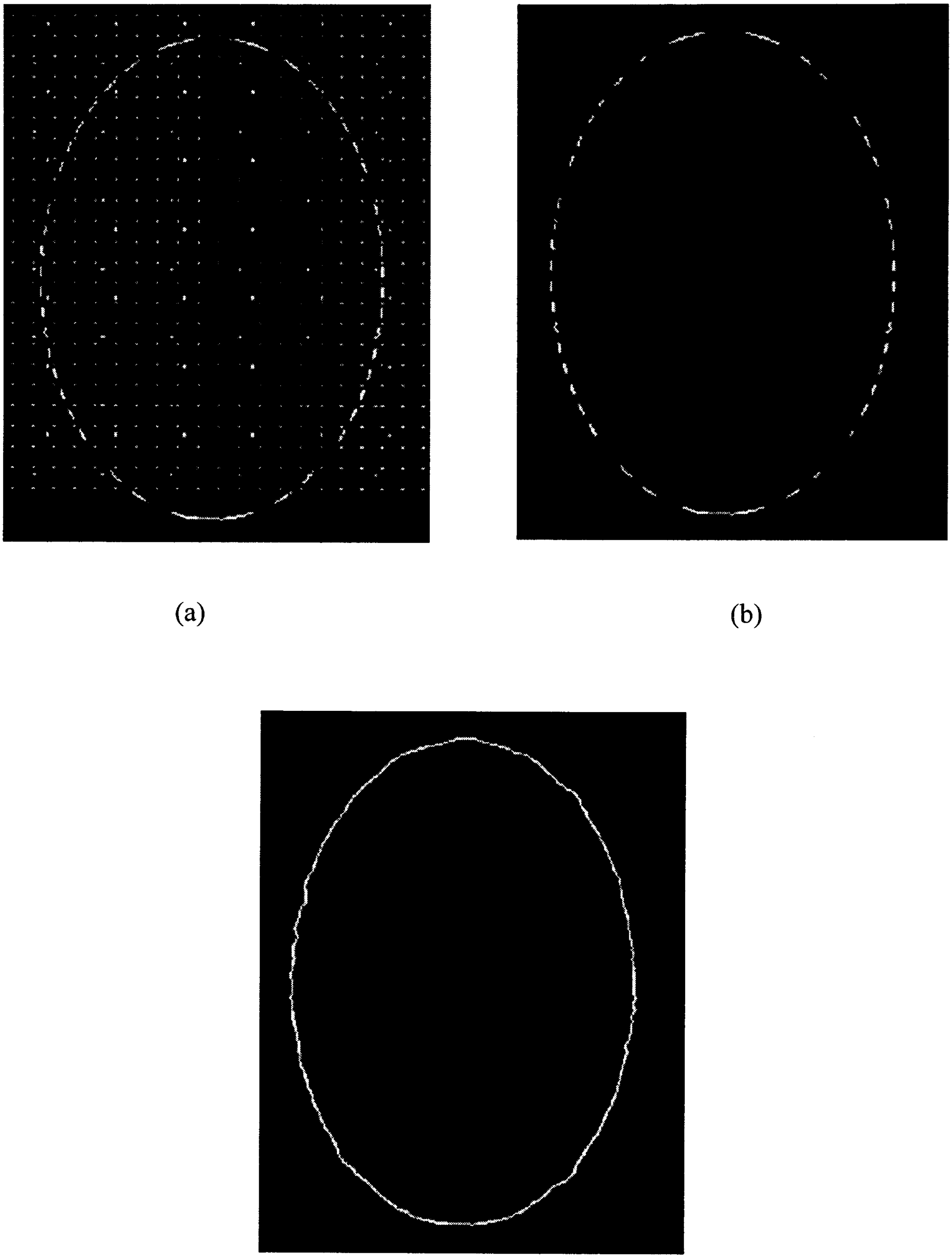


Figure 2.2.12 (a) Another randomly discontinuous elliptical edge, and (b) Using the adaptive morphological edge-linking algorithm.



(a)

(b)

(c)

Figure 2.2.13 An example with noise. (a) The elliptical edge with added uniform noise, (b) The edge after removing noise, and (c) Using the adaptive morphological edge-linking algorithm.

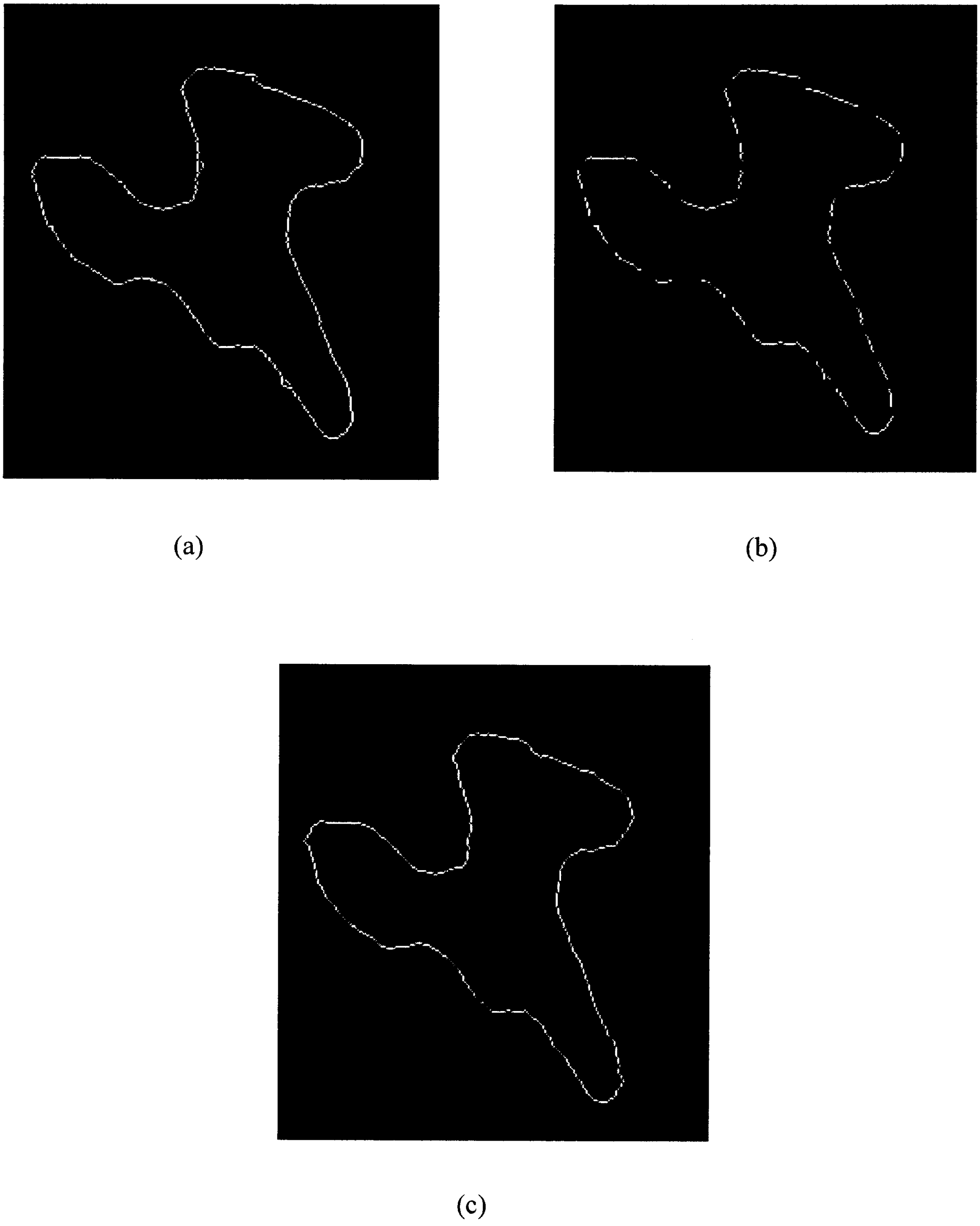
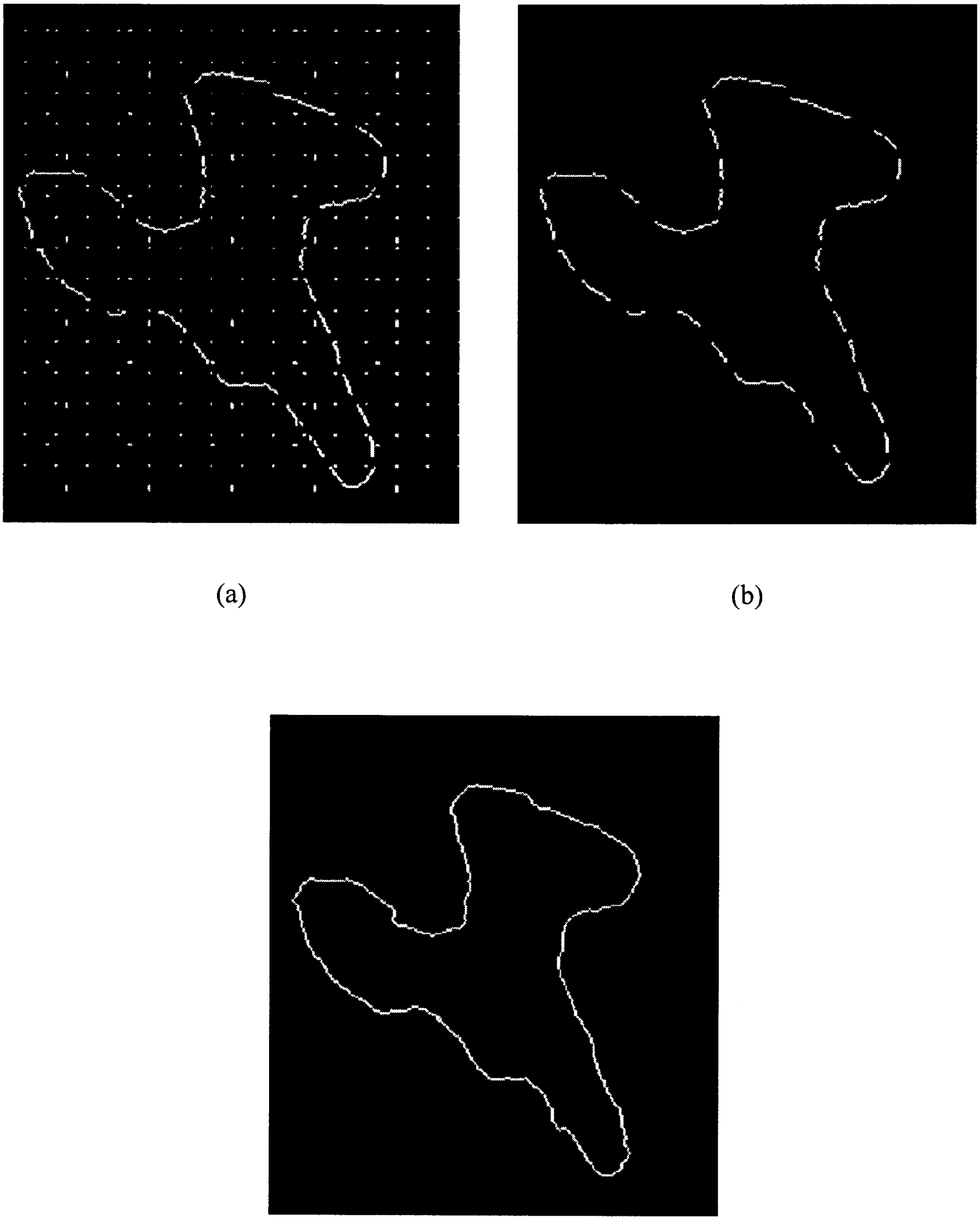


Figure 2.2.14 An example of an industrial part. (a) The edge of an industrial part, (b) Its randomly discontinuous edge, and (c) Using the adaptive morphological edge-linking algorithm.



(a)

(b)

(c)

Figure 2.2.15 An example of part with added noise. (a) Part edge with added uniform noise, (b) Part edge after removing noise, and (c) Using the adaptive morphological edge-linking algorithm.

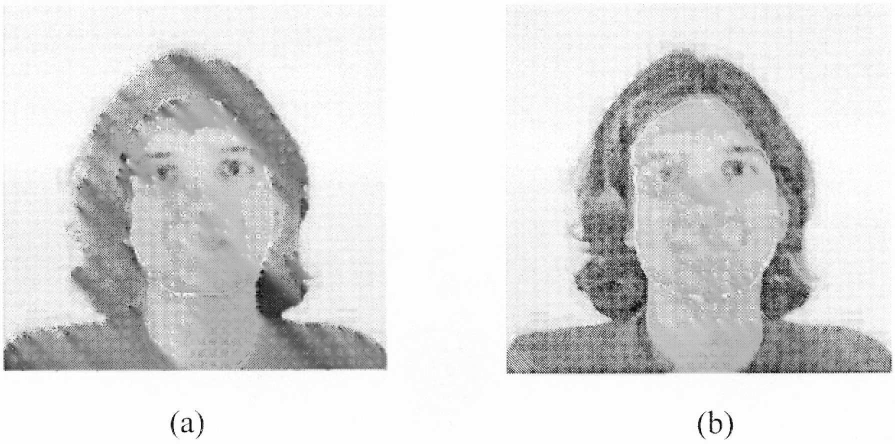


Figure 2.2.16 An example of face boundary linking. (a) Face image with the originally detected broken edge, and (b) Face image with the edge linked by the adaptive morphological edge-linking algorithm.

CHAPTER 3

FEATURE REDUCTION

In this chapter, an improved feature reduction method in the combinational input and feature space for Support Vector Machines (SVMs) is presented. In the input space, a subset of input features is selected by ranking their contributions to the decision function. In the feature space, features are ranked according to the weighted support vector in each dimension. By combining both input and feature space, a fast non-linear SVM is developed without a significant loss in performance. The proposed method has been tested on the detection of a face, person, and car. Subsets of features are chosen from pixel values for face detection and from Haar wavelet features for person and car detection. The experimental results show that the proposed feature reduction method works successfully. In fact, this method performs better than the methods of using all the features and the Fisher's features in the detection of a person and car. Also a speed advantage is gained.

3.1 Introduction

Feature extraction and reduction are two primary issues in feature selection that are essential in pattern classification. Feature extraction is used to achieve high classification rates by extracting features to represent objects from raw data. Feature reduction is used to select a subset of features with preservation or improvement of classification rates. In general, it tends to speed up the classification process by keeping the most important class-relevant features.

Support Vector Machines (SVMs) are formulated from a mathematical point of view. While most classifiers (e.g. Bayesian, neural networks, and Radial Basis Function (RBF)) are trained to minimize the empirical risk, SVMs are implemented to minimize the structural risk. Osuna *et al.* [45] applied SVMs to face detection. Heisele *et al.* [23] trained a 2nd-degree polynomial SVM using 10,038 faces and 36,220 non-faces, and achieved a higher detection rate than in [45]. However, they need to spend several minutes to search an image for faces at different scales. Heisele *et al.* [24] presented the two following methods to speed up face detection using SVMs: hierarchical classification and feature reduction.

Principal Component Analysis (PCA) features are used to reduce the dimensionality in input space. Weston *et al.* [71] developed a feature reduction method by minimizing the bounds on the leave-one-out error. Evgenious *et al.* [17] introduced a method for feature selection based on the observation that the most important features are the ones that separate the hyperplane the most.

In this chapter, the method of feature reduction in the input and feature spaces to achieve a fast non-linear SVM without a significant loss in performance is presented. The rest of this chapter is organized as follows. In Section 3.2, a brief introduction to SVMs is given. Sections 3.3 and 3.4 describe the feature reduction methods in input and feature space, respectively, for face detection. In Section 3.5, a fast non-linear SVM is developed by a combination of input and feature space for face detection. In Section 3.6, the proposed method is applied for person and car detection. Finally, discussions are provided in Section 3.7.

3.2 Support Vector Machines (SVMs)

In this section, the SVMs [14, 67] that are designed for two-class classification by using a hyperplane that maximizes the margin or the distance between the hyperplane and the nearest data point in each class is briefly introduced. The hyperplane is viewed as an optimal separating hyperplane (OSH). These nearest points of the training set are called *support vectors*.

Consider a set of l labeled training patterns $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2), ..., (\mathbf{x}_i, y_i), ..., (\mathbf{x}_l, y_l)$, where $\mathbf{x}_i$ denotes the i-th training sample and $y_i \in \{1, -1\}$ denotes the class label. If the data are not linearly separable in the input space, a non-linear transformation function $\Phi(\cdot)$ is used to project $\mathbf{x}_i$ from the input space to a higher dimensional feature space. An OSH is constructed in the feature space by maximizing the margin between the closest points $\Phi(\mathbf{x}_i)$ of two classes. The inner-product between two projections is defined by a kernel function $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$. The commonly used kernels include polynomial, Gaussian RBF, and Sigmoid kernels.

The decision function of the SVM is defined as

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b, \tag{3.1}$$

where $\mathbf{w}$ is the support vector, α_i is the Lagrange multiplier, and b is a constant. The optimal hyperplane can be obtained by maximizing

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{3.2}$$

subject to $\sum_{i=1}^{l} \alpha_i y_i = 0$ and $0 \le \alpha_i \le C$, where C denotes a positive value determining the constraint violation during the training process.

3.3 Feature Reduction in Input Space

3.3.1 Projection Algorithm

Principal Component Analysis (PCA) is widely used in image representation for dimensionality reduction. To obtain *m* principal components, a transformation matrix of $m \times N$ is multiplied by an input pattern of $N \times 1$. The computation is costly. In this section, a method of feature reduction is proposed in the input space in order to save computational time.

One way of feature reduction is using Fisher's criterion to choose a subset of features that possess a large between-class variance and a small within-class variance. For face detection, the within-class variance is calculated as

$$\sigma_i^2 = \frac{\sum_{j=1}^{l} (g_{j,i} - m_i)^2}{l-1}, \tag{3.3}$$

where l is the total number of samples, $g_{j,i}$ is the i-th dimensional gray value of sample j, and m_i is the mean value of the i-th dimension. Fisher's score for between-class measurement is calculated as

$$S_i = \left| \frac{m_{i,face} - m_{i,nonface}}{\sigma_{i,face}^2 + \sigma_{i,nonface}^2} \right|. \tag{3.4}$$

By selecting the features with the highest Fisher's scores, the most discriminating features between face and non-face classes can be retained.

To improve Fisher's method, a 2nd-degree polynomial SVM with kernel $K(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}\cdot\mathbf{y})^2$ is proposed. The decision function for a pattern **x** is defined as

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{s} \alpha_i y_i (1+\mathbf{x}_i \cdot \mathbf{x})^2 + b \\ &= \sum_{i=1}^{s} \alpha_i y_i (1 + x_{i,1}x_1 + x_{i,2}x_2 + \ldots + x_{i,k}x_k + \ldots + x_{i,N}x_N)^2 + b, \end{aligned} \tag{3.5}$$

where s is the total number of support vectors, $\mathbf{x}_i$ is the i-th support vector, and $x_{i,k}$ and x_k are respectively the k-th componenta for the support vector $\mathbf{x}_i$ and the pattern **x**. The k-th of (3.5) (where $k = 1, 2, \ldots, N$) is

$$\begin{aligned} f(\mathbf{x},k) &= \sum_{i=1}^{s} \alpha_i y_i \left[2x_k x_{i,k}(1 + x_{i,1}x_1 + \ldots + x_{i,k-1}x_{k-1} + x_{i,k+1}x_{k+1} + \ldots + x_{i,N}x_N) + x_k^2 x_{i,k}^2\right] \\ &= \sum_{i=1}^{s} \alpha_i y_i \left[2x_k x_{i,k}(1 + x_{i,1}x_1 + \ldots + x_{i,N}x_N) - x_k^2 x_{i,k}^2\right]. \end{aligned} \tag{3.6}$$

The largest m contributions to the decision function out of the original N features are selected. The contribution can be obtained by

$$F(k) = \int_V f(\mathbf{x},k)dP(\mathbf{x}), \tag{3.7}$$

where V denotes the input space and $P(\mathbf{x})$ denotes the probability distribution function. Since $P(\mathbf{x})$ is unknown, $F(k)$ is approximated using a summation over the support vectors as

$$F(k) = \sum_{i=1}^{s} \left| \sum_{j=1}^{s} \alpha_j y_j [2x_{i,k}x_{j,k}(1 + x_{j,1}x_{i,1} + \ldots + x_{j,N}x_{i,N}) - x_{i,k}^2 x_{j,k}^2] \right|. \tag{3.8}$$

3.3.2 Experimental Results

A face image database from the Center for Biological and Computational Learning at Massachusetts Institute of Technology (MIT), which contains 2,429 face training samples, 472 face testing samples, and 23,573 non-face testing samples, is adopted. 15,228 non-face training samples are randomly collected from the images that do not contain faces. The size of all these samples is 19 × 19. A 2nd-degree polynomial SVM with kernel $K(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$ is used in the experiments.

In order to remove background pixels, a mask is applied to extract only the face. Prior to classification, image normalization and histogram equalization are performed. The image normalization is used to normalize the gray-level distribution by the Gaussian function with mean zero and variance one. The histogram equalization is performed using a transformation function equal to the cumulative distribution to produce an image whose gray levels have a uniform density. In Figure 3.1, (a) shows a face image, (b) shows the mask, and (c) and (d) show the images after normalization and histogram equalization, respectively.

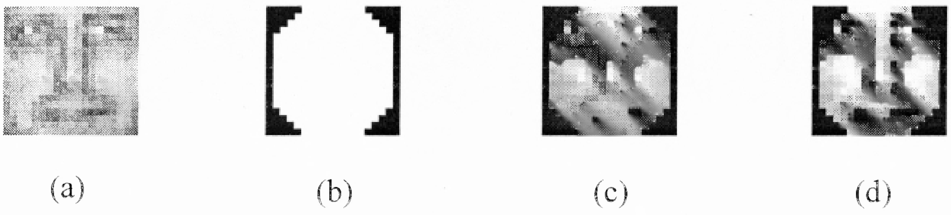


Figure 3.1 Face image preprocessing. (a) Original face image, (b) the mask, (c) normalized image, and (d) histogram equalized image.

The following two methods are used to calculate PCA values: one is to combine face and non-face training samples, and the other is to use face training samples only.

Figure 3.2 shows the Receiver Operating Characteristic (ROC) curves of using different numbers of principal components. The ROC curve is defined as shifting the SVM hyperplane by changing the threshold value *b*. Face classification is performed on the testing set and the false positive and the detection rates are calculated. The horizontal axis shows the false positive rate over 23,573 non-face testing samples. The vertical axis shows the detection rate over 472 face testing samples. In this example, the PCA values are calculated using only the face training samples. It is observed that using 20 or 40 PCA values obtains worse results than using all the 283 grays. While using 60 or 100 PCA values can achieve almost the same performance as using all the 283 gray values.

Figure 3.3 shows the ROC curves of using the two following methods to obtain PCA features: one uses both face and non-face training samples, and the other uses only face training samples. It is observed that using only positive training samples performs better than using both positive and negative training samples. This is also tested on the 3,600 faces extracted from FERET database and the same result is obtained. The reason is that only training face samples are used in the calculation of the transformation matrix and later for testing the input samples are projected on the face space, so that better classification results can be achieved in separating face and non-face classes.

Figure 3.4 shows the ROC curves for different preprocessing methods: image normalization, histogram equalization, and without preprocessing. It is observed that using normalization or equalization can produce better results than without preprocessing. Therefore, in the following experiments, image normalization is utilized as the pre-processing method and positive training samples are used to calculate PCA values.

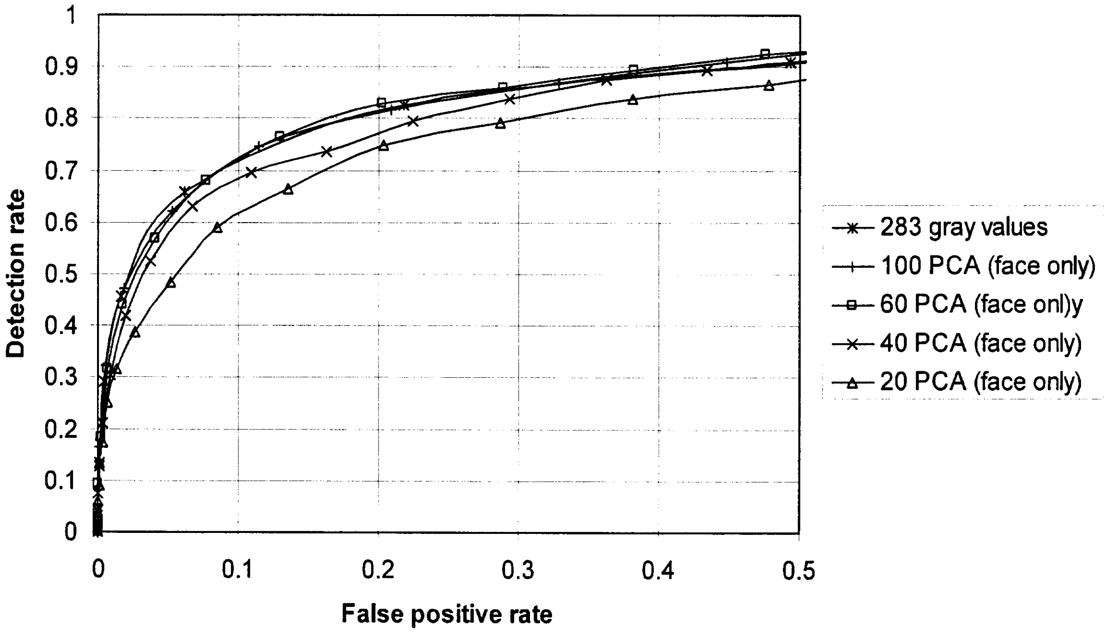


Figure 3.2 ROC curves for different numbers of PCA values.

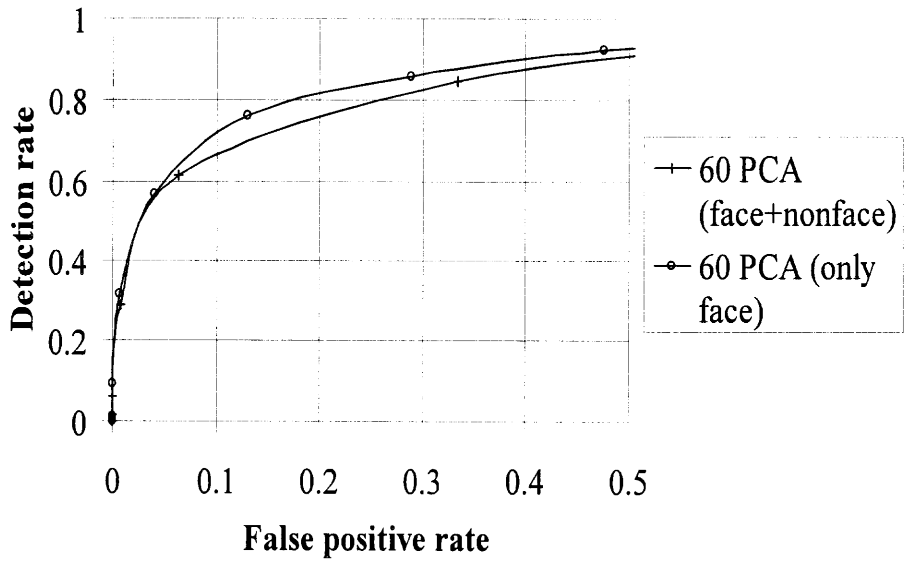


Figure 3.3 ROC curves of using the following two methods to obtain PCA features: one uses both face and non-face training samples, and the other uses only face training samples.

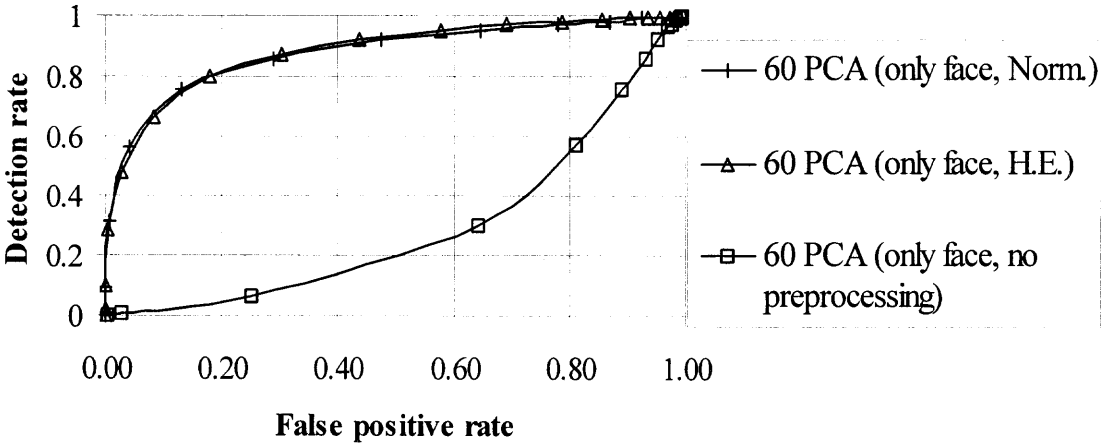


Figure 3.4 ROC curves for different preprocessing methods: image normalization, histogram equalization, and without preprocessing.

Using the normalized 2,429 face and 15,228 non-face training samples and taking all the 283 gray values as input to train the 2^{nd}-degree SVM, 252 and 514 support vectors for face and non-face classes are obtained, respectively. Using these support vectors in Equation (3.8), $F(k)$ is obtained, where k = 1, 2, ..., 283. Figure 3.5 shows the ROC curves for different features in input space. This ranking method of using 100 features is compared with the methods of using all the 283 gray values, 100 PCA features, Fisher's scores, and the 100 features selected by Evgenious *et al.* [17]. It is observed that using this method performs similarly as using all the 283 features and using 100 PCA values; however, it is better than using the 100 features by Fisher's scores and by Evgenious *et al.* [17].

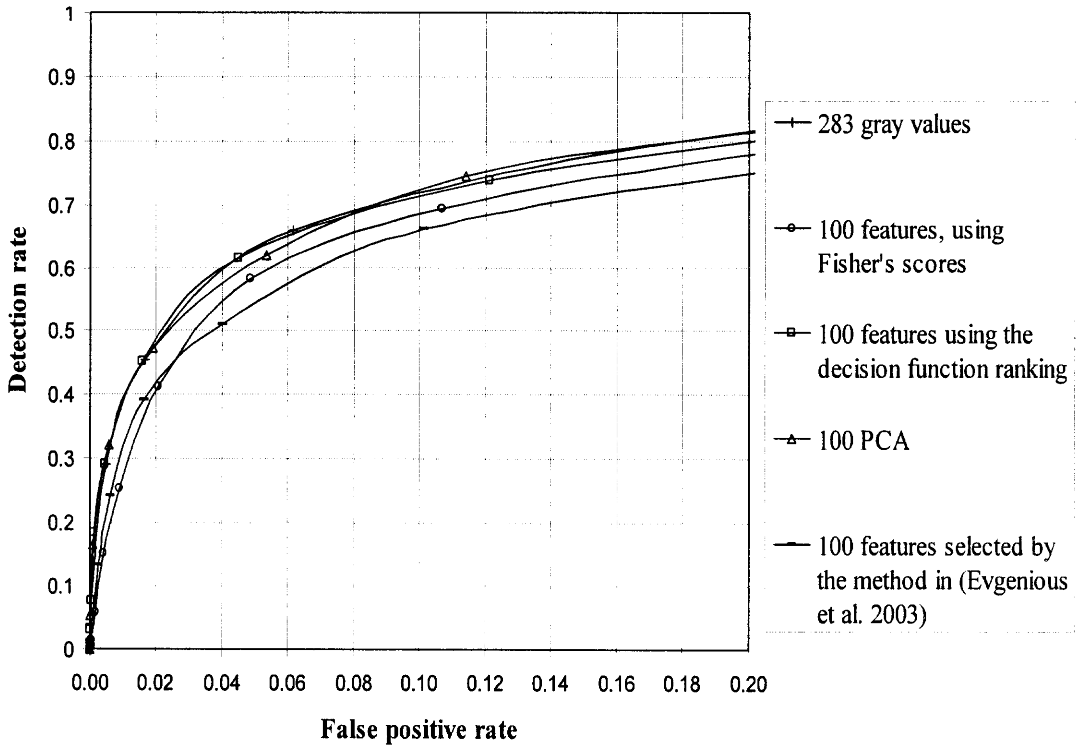


Figure 3.5 ROC curves for comparisons of using different features in input space.

3.4 Feature Reduction in Feature Space

3.4.1 Feature Ranking

In feature space, the decision function *f*(**x**) of SVMs is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{S} \alpha_i y_i (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)) + b = \mathbf{w} \cdot \Phi(\mathbf{x}) + b, \tag{3.9}$$

where **w** is the support vector. For a 2$^{\text{nd}}$-degree polynomial SVM with the input space of dimension *N* and kernel $K(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2$, the feature space is given by

$$\Phi(\mathbf{x}) = (\sqrt{2}x_1,\ldots,\sqrt{2}x_N, x_1^2,\ldots,x_N^2, \sqrt{2}x_1x_2,\ldots,\sqrt{2}x_{N-1}x_N) \tag{3.10}$$

of dimension $P = N(N+3)/2$.

Suppose that a 2nd-degree SVM is trained using face and non-face samples to obtain *s* support vectors. The support vector in the feature space can be represented as

$$\mathbf{w} = \sum_{i=1}^{s} \alpha_i y_i \Phi(\mathbf{x}_i) = (w_1, w_2, ..., w_P). \tag{3.11}$$

One way to select a subset of features is to rank $|w_k|$, for $k = 1,2,...,P$. In this section, an improved method of using $\left| w_k \int_V | x_k^* | \, dp(x_k^*) \right|$ is proposed, where x_k^* denotes the *k*-th component of **x** in the feature space *V*. Since the distribution function $dp(x_k^*)$ is unknown, a ranking function *R*(*k*) is defined as

$$R(k) = \left| w_k \sum_{i=1}^{s} | x_{i,k}^* | \right|, \tag{3.12}$$

where $x_{i,k}^*$ denotes the *k*-th component of $\mathbf{x}_i$. The decision function of *q* features is calculated as

$$f(\mathbf{x}, q) = \mathbf{w}(q) \cdot \Phi(\mathbf{x}, q) + b, \tag{3.13}$$

where $\mathbf{w}(q)$ is the selected *q* features in **w** and $\Phi(\mathbf{x}, q)$ is the corresponding *q* features in **x**.

For a pattern **x**, the difference of two decision values of using all the features and using the subset of *q* features is calculated as

$$\Delta f_q(\mathbf{x}) = | f(\mathbf{x}) - f(\mathbf{x}, q) |. \tag{3.14}$$

The differences over all the support vectors are summed up as

$$\Delta F_q = \sum_{i=1}^{s} \Delta f_q(\mathbf{x}_i). \tag{3.15}$$

3.4.2 Experimental Results

In the experiments, a 2nd-degree polynomial SVM is trained using 60 PCA values in the input space. The training samples are the same as in Section 3.2, and 289 and 412 support vectors for face and non-face classes are obtained, respectively. The 1,890 features in the feature space can be calculated by Equation (3.10). The support vector in the feature space, $\mathbf{w} = (w_1, w_2, \ldots, w_{1890})$, can be calculated by Equation (3.11). The value of ΔF_q for different q can be obtained by Equation (3.15). Figure 3.6 shows the variance of ΔF_q by varying the number of feature subsets using the following two ranking methods: one by $|w_k|$ and the other by the proposed method. It is observed that the proposed method leads to a faster decrease in the difference.

Given a pattern $\mathbf{x}$, the decision value of using the selected q features can be calculated by Equation (3.13). When $q = 300, 500, \text{ and } 1000$, the results are illustrated in Figure 3.7. It is observed that using the selected 500 or 1,000 features can achieve almost the same performance as using all the 1,890 features. However, using 300 features is insufficient to achieve a good performance.

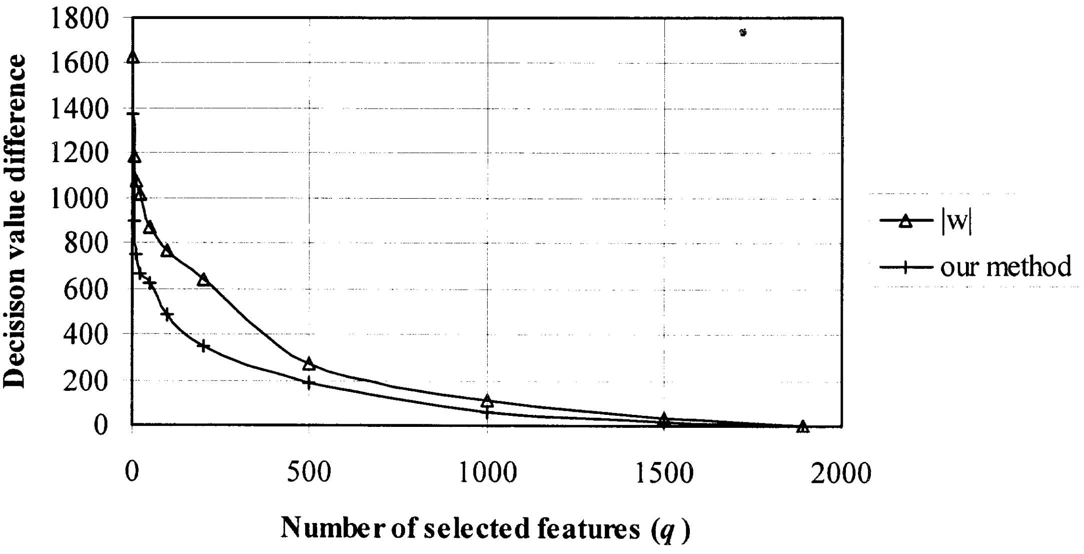


Figure 3.6 Decision value differences relative to the subset of q features in the feature space selected by two ranking methods.

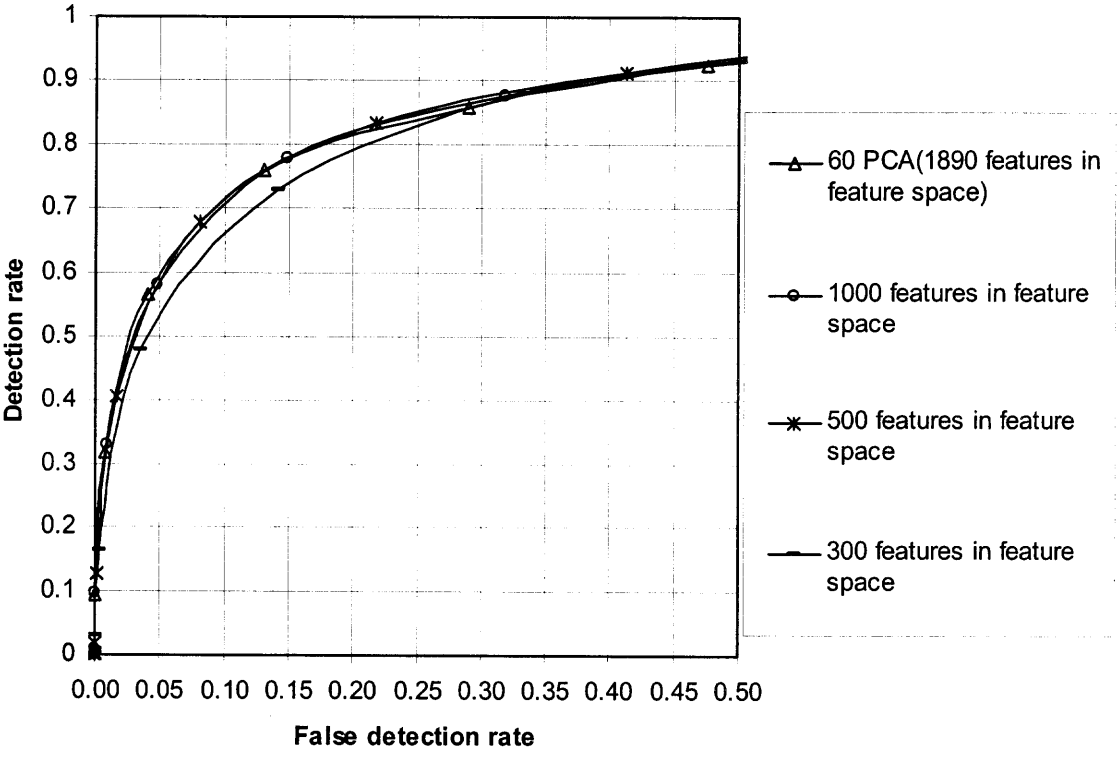


Figure 3.7 ROC curves for different numbers of features in the feature space.

3.5 Combinational Input and Feature Space

In this section, the method of feature reduction by combining the input and feature space is presented. First, m features are chosen from the N input space as described in Section 3.3, the 2^{nd}-degree polynomial SVM is trained. Next, q features are selected from $P = m(m+3)/2$ features in the feature space to calculate the decision value. The two following methods are used to compute the decision values: one by Equation (3.5) in input space and the other by Equation (3.9) in feature space. In Equation (3.5), the number of multiplications required to calculate the decision function is $(N+1)s$. Note that s is the total number of support vectors. In Equations (3.9) and (3.10), the total number of multiplications required is $(N+3)N$. If $(N+1)s > (N+3)N$, it is more efficient to implement the 2^{nd}-degree polynomial SVM in the feature space; otherwise, in the input space. This is evidenced by the experiments because the number of support vectors is more than 700, which is much larger than N. Note that N = 283, 60, or 100 indicates all the gray-value features, 60 PCA values, or 100 features, respectively.

The SVM is trained using the selected 100 features as described in Section 3.2 to obtain 244 and 469 support vectors for face and non-face classes, respectively. From Figure 3.8, it can be seen that using the 3,500 features selected by this method can produce the similar performance as using all the 5,150 features by Equation (3.10). Figure 3.9 shows the comparisons of using the combinational method, 60 PCA values, and all the 283 gray values. It is observed that using the combinational method can obtain competitive results with using 60 PCA values or all 283 gray values. Apparently, this method gains has a speed advantage.

Table 3.1 lists the number of features used in the input and feature space and the number of multiplications required in calculating the decision values for comparing the proposed method with the methods of using all the gray values and using PCA features.

Table 3.1 Comparisons of the Number of Features and the Number of Multiplications

Methods	Number of features in input space	Number of features in feature space	Number of multiplications
All gray values	283	40,469	80,938
PCA	60	1,890	20,760
Proposed method	100	3,500	8,650

From Table 3.1, it is observed that by using PCA for feature reduction in the input and feature space, a speed-up factor of 4.08 can be achieved. However, by using the proposed method, a speed-up factor of 9.36 can be achieved. Note that once the features in the feature space are determined, projecting the input space on the whole feature space is not needed; it can be done on the selected feature subspace. This can further reduce the computation, i.e., only 7,000 multiplications are required instead of 8,650.

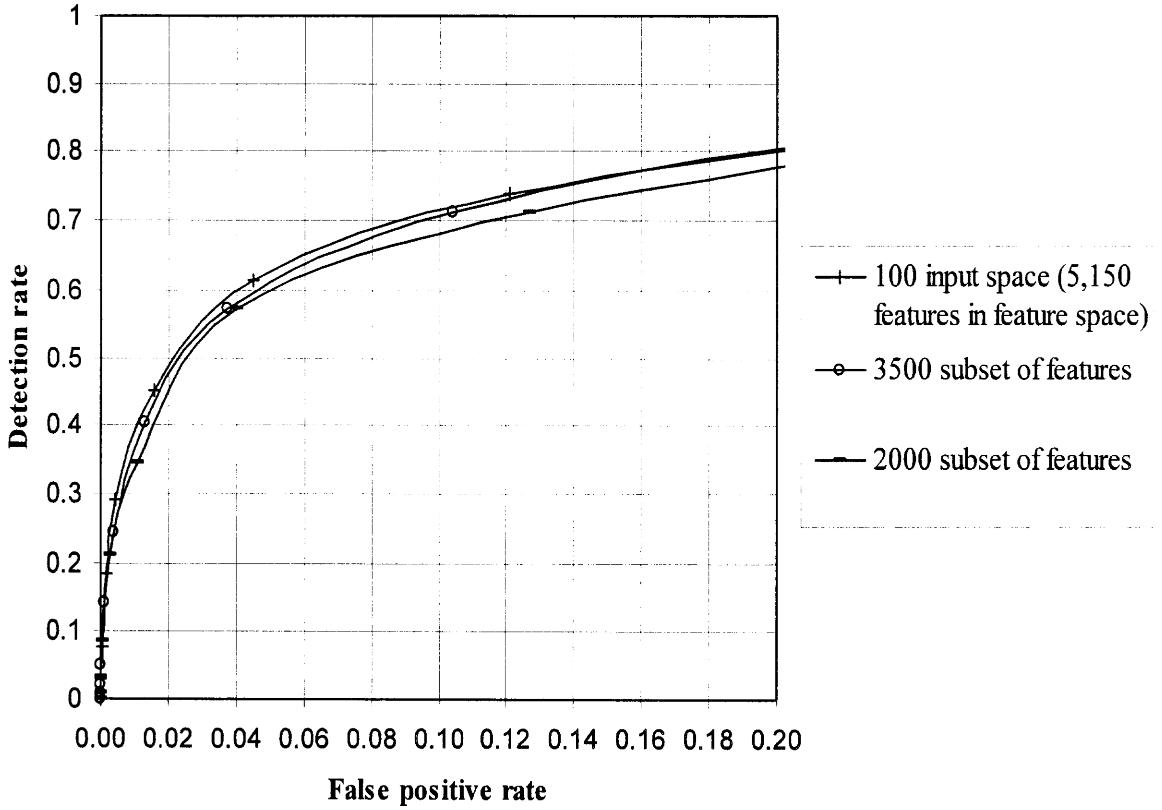


Figure 3.8 ROC curves of using different numbers of features in feature space. The 100 features in the input space are selected using the proposed ranking method.

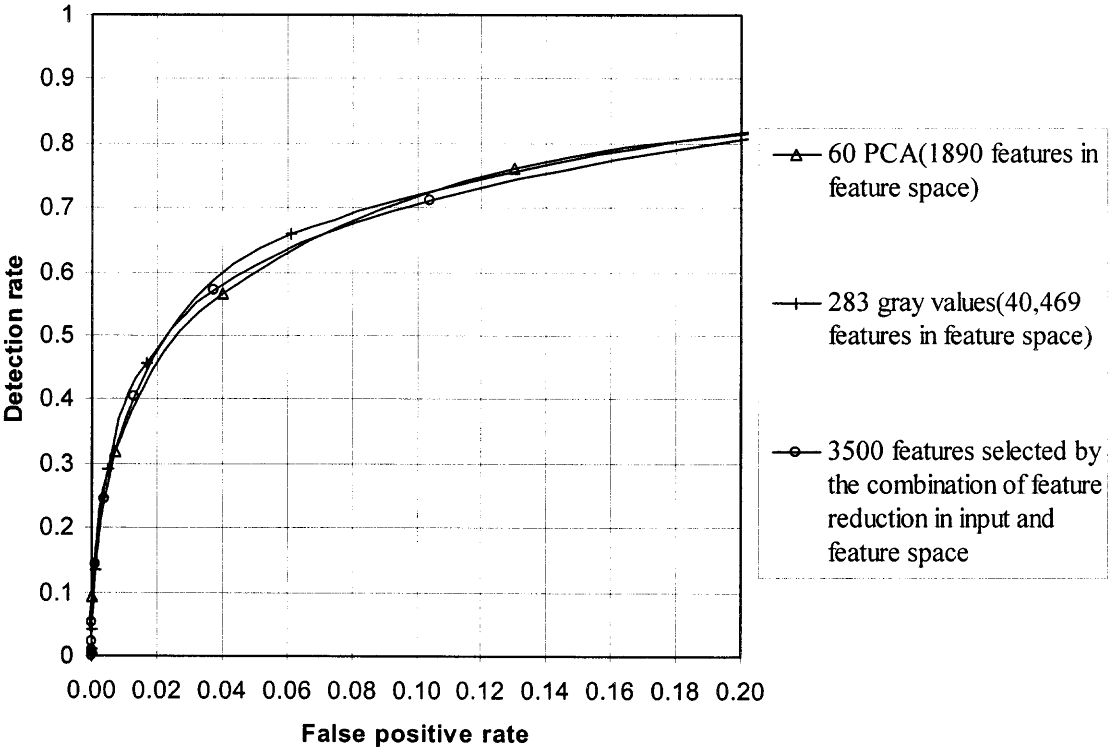


Figure 3.9 ROC curves of using the proposed feature reduction method, 60 PCA, and all the 283 gray values.

3.6 Performance on Person and Car Detection

In this section, the experiments are extended to person and car detection to further justify the validity of the proposed feature reduction method. Instead of selecting features from gray values in face detection, Haar wavelet features are used in person and car detection since the positive samples of person and car contain complex backgrounds, while the face samples contain only the face part.

3.6.1 Person Detection

The 924 person images of size 128 × 64 used in [17] are adopted. Some examples are shown in Figure 3.10; 700 images are used for training and the remaining 224 images are used for testing. Also 6,000 non-person training samples and 3,000 non-person testing images are randomly extracted.

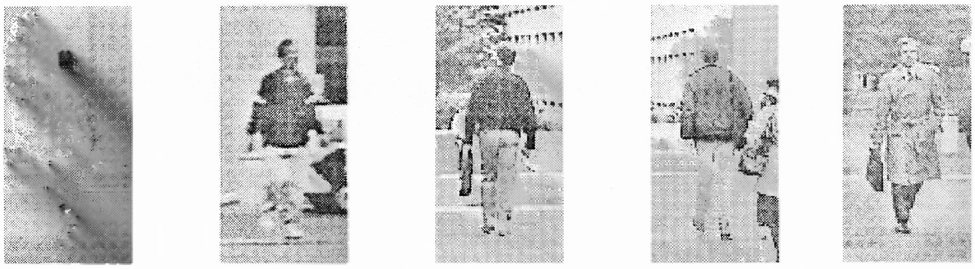


Figure 3.10 Some examples of person images.

Haar wavelet representation is used in person detection since it can produce better results than using pixels or PCA [17]. The three types of Haar wavelets used are shown in Figure 3.11. The Haar feature is defined as the difference between two sums of pixel values in white rectangular and gray rectangular areas. Haar features are calculated at scales of 32 × 32 and 16 × 16, respectively. In total, 1,326 Haar features are obtained for each person image. The Haar features are normalized between 0 and 1.

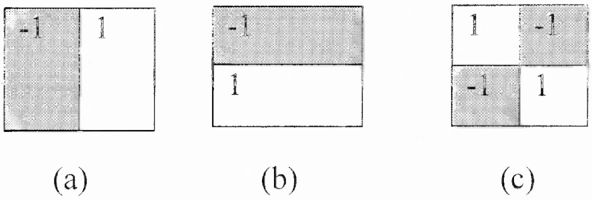


Figure 3.11 The three types of Haar wavelet features: (a) vertical, (b) horizontal, and (c) diagonal.

Figure 3.12 shows the results of using the three methods: all the 1,326 Haar features, the subset of 100 features selected using the proposed method, and the 100 features using Fisher's score. The *x*-axis indicates the false positive rate over 3,000 non-person testing samples. The *y*-axis indicates the detection rate over 224 person images. It is observed that when the false positive rate > 0.004, using 100 Fisher's features can obtain slightly better results than using all the 1,326 features, while using 100 features selected by the proposed method can always perform better than using all the 1,326 features.

Figure 3.13 shows that using the subset of 1,000 features in feature space, the results are obviously worse than using all the 5,150 features. Using the subset of 2,000 features, at very low false detection rates (i.e., less than 0.002), the performance is worse than using all the 5,150 features, while using the subset of 3,500 features, almost the same results can be obtained as using all the 5,150 features.

For person detection, after training a 2nd-degree SVM using all the 1,326 Haar features, 486 support vectors are obtained, among them 136 for person and 350 for non-person classes. It is more efficient to implement the 2nd-degree SVM in input space since $(N+1)s < (N+3)N$, where $N = 1{,}326$ and $s = 486$. Therefore, it takes about 644,922 multiplications to classify a pattern. If the proposed feature reduction method is used in input space to select 100 Haar features and in feature space to select 3,500 features, it takes about 8,650 multiplications. Thus it is about 74.5 times faster.

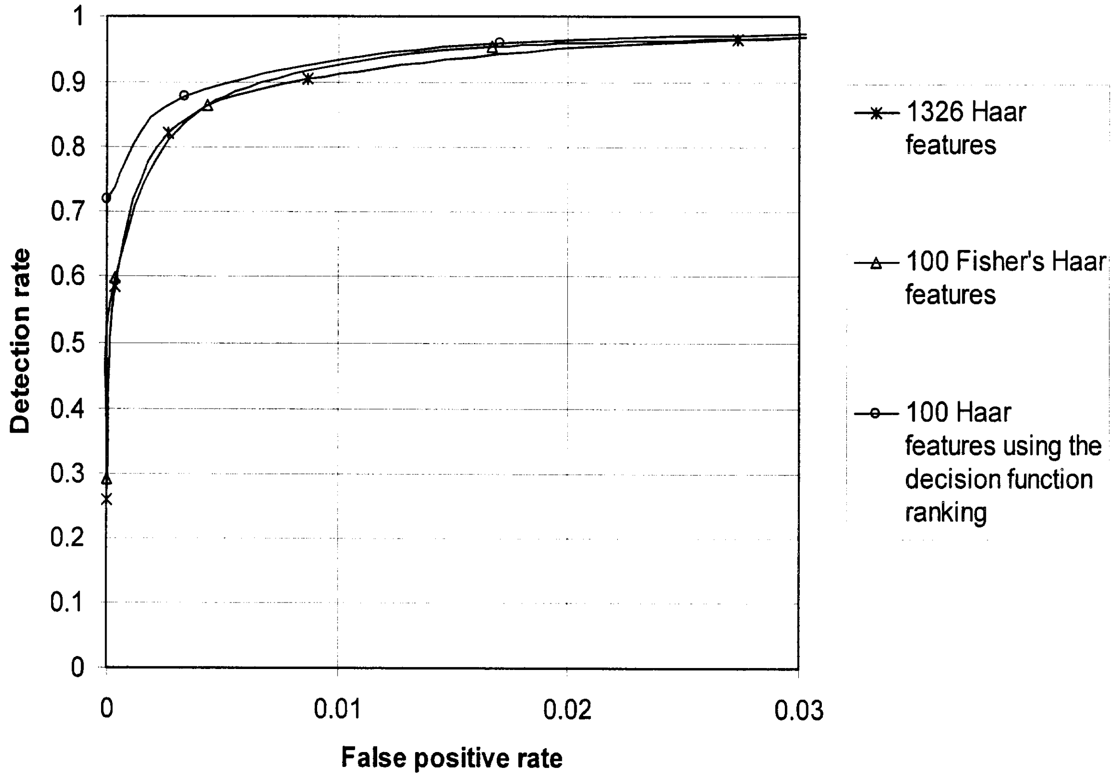


Figure 3.12 ROC curves for person detection using the proposed feature reduction method in input space, Fisher's method and all the 1,326 Haar features.

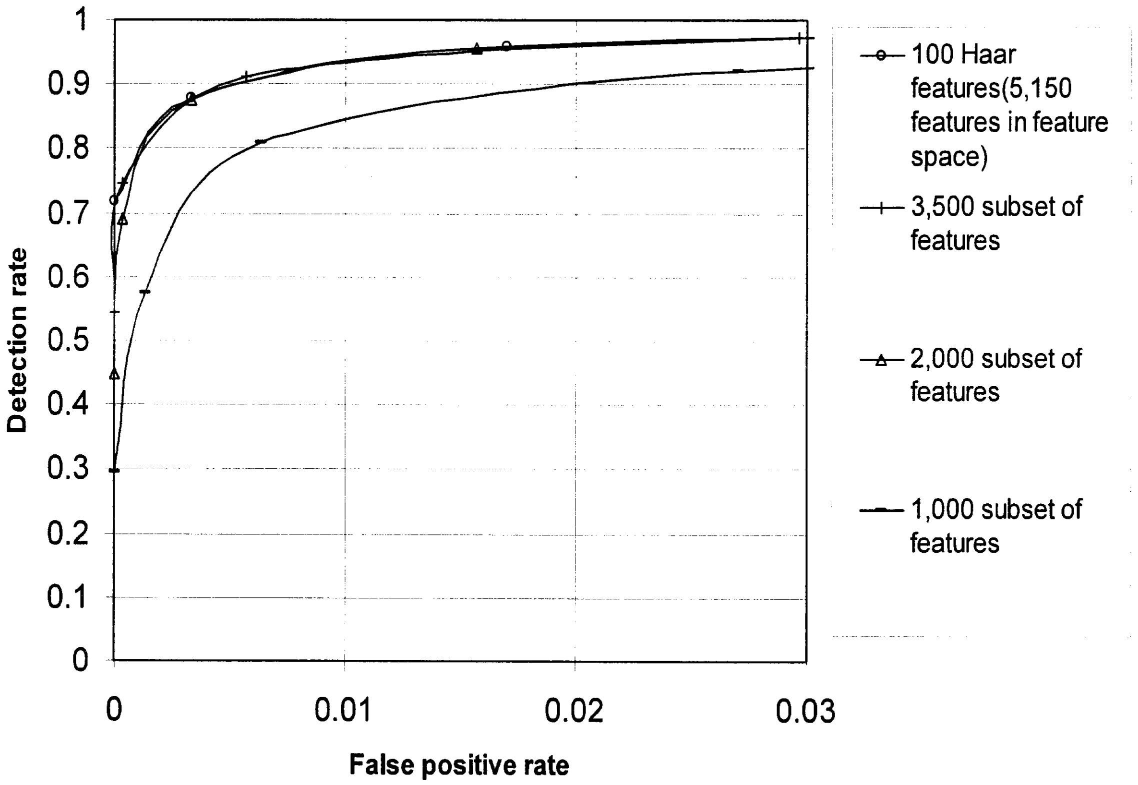


Figure 3.13 ROC curves for person detection using the proposed feature reduction method in feature space.

3.6.2 Car Detection

In the experiment, 516 car images of size 128 × 128 used in [48] are adopted. Some examples are shown in Figure 3.14. For each car image, its mirror image is also constructed. Totally, 1,032 car images are obtained. 700 car images are used as positive training samples and the remaining 332 images are used as testing samples. Also 6,000 non-car training samples and 3,000 non-car testing images are randomly extracted.

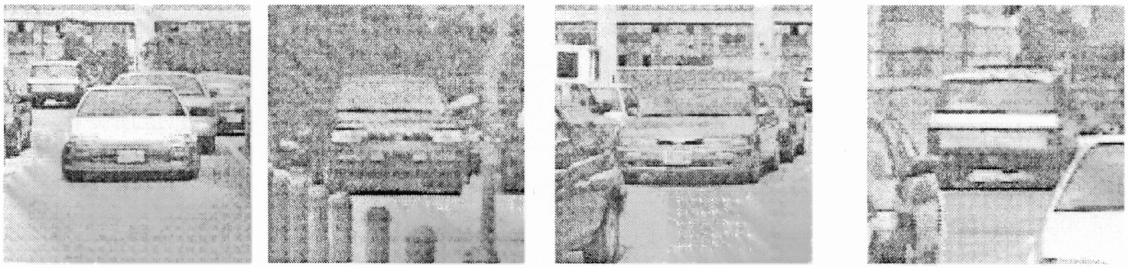


Figure 3.14 Some examples of car images.

Haar wavelet features are also used for car detection and 3,030 Haar features are obtained for each car image. The Haar features are normalized between 0 and 1. Figure 3.15 shows the results of using the three methods: all the 3,030 Haar features, the subset of 100 features selected by the proposed method, the 100 features by Fisher's score. It is observed that using the subset of 100 Fisher's features can produce a little better result than using all the 3,030 features. Using the subset of 100 features from the proposed method in input space can produce better results than both the Fisher's method and all the 3,030 features.

Figure 3.16 shows that using the subset of 1,000 features in feature space, the result is obviously worse than using all the 5,150 features, while using the subsets of 2,000 and 3,500 features, similar results can be obtained as using all the 5,150 features.

For car detection, after training a 2^{nd}-degree SVM using all the 3,030 Haar features, 260 support vectors are obtained, among them 88 for car and 172 for non-car classes. It is more efficient to implement the 2^{nd}-degree SVM in input space since $(N+1)s < (N+3)N$, where $N = 3{,}030$ and $s = 260$. Therefore, it takes about 788,060 multiplications to classify a pattern. If the proposed feature reduction method is used in input space to select 100 Haar features and in feature space to select 3,500 features, it takes about 8,650 multiplications. So it is about 90 times faster.

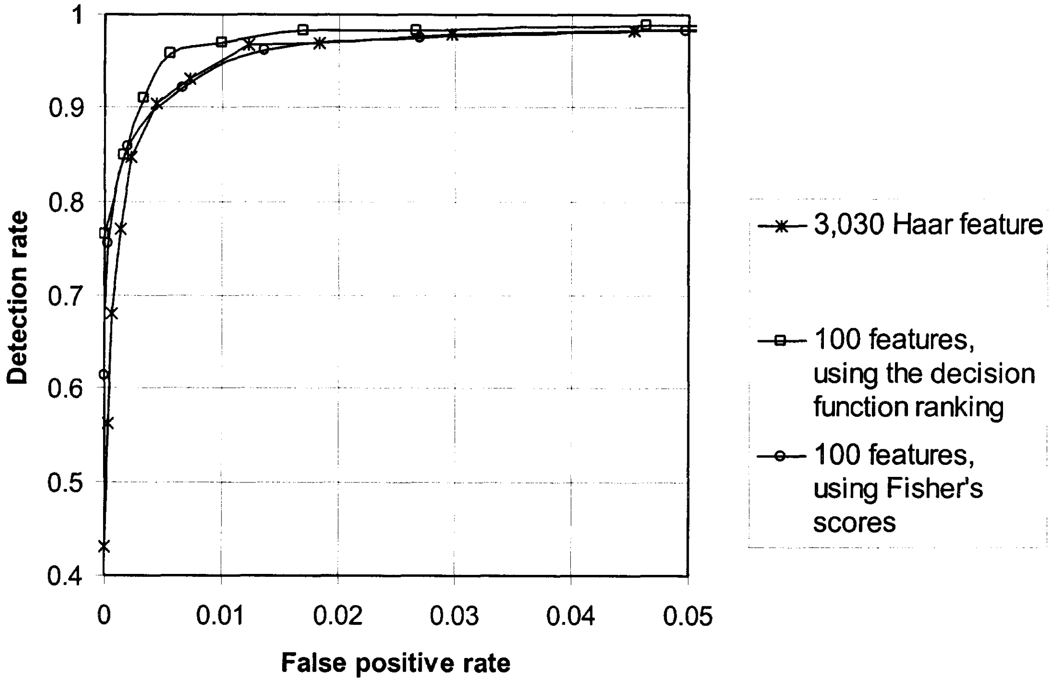


Figure 3.15 ROC curves for car detection using the proposed feature reduction method in input space, Fisher's method, and all the 3,030 Haar features.

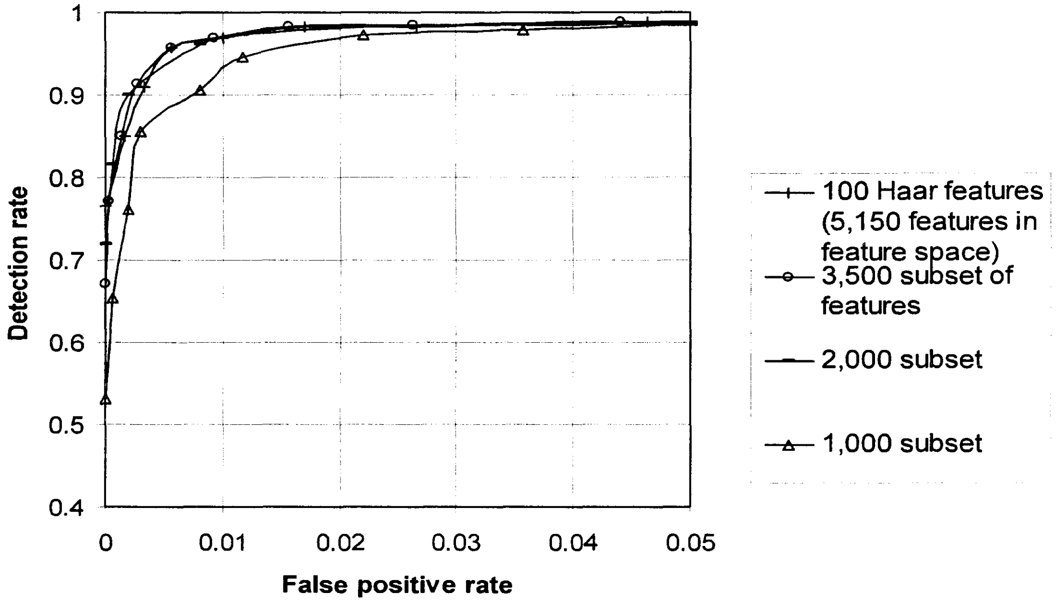


Figure 3.16 ROC curves for car detection using the proposed feature reduction method in feature space.

3.7 Discussions

In this chapter it is intended to select a subset of features by ranking their contributions to the decision function of SVMs. The features that have large contributions indicate the importance of relevance in classification. This leads to a fast 2nd-degree SVM without a significant loss in performance. The proposed method has been tested on detection of a face, person, and car. The experimental results show that the proposed feature reduction method works successfully. In fact, the proposed method performs better in the detection of a person and car. This is because they contain complex background and the proposed method only selects the majority of target features rather than the background features. The proposed method also gains on the advantage in speed. For face detection, the subset is chosen directly from 283 gray values instead of the PCA features that require matrix computation. For person detection, 100 Haar features are selected instead of all the 1,032 Haar features. For car detection, 100 Haar features are selected instead of all the 3,030 Haar features. Furthermore, in the feature-space method, a subset of 3,500 features is selected instead of all the 5,150 features.

For the feature reduction method in input space, the number of features required to obtain good results is problem dependent. Experiments indicate that it is best to select 100 gray values from all the 283 gray values for the face detection, and 100 Haar features from 1,326 and 3,030 Haar features for person and car detection, respectively. For face detection problem, the selected features are shown in Figure 3.17 as white pixels. It can be seen that most of these features are located at eyes, nose, cheek, and the corner of the mouth. These pixels are the most relevant ones that distinguish face from non-face images. For car and person detection, the features near the boundary capture the difference between the object and background, while the features on the object capture the object property. This is the reason why better results can be obtained by using a subset of features than using all the Haar features. Six examples of the selected Haar features are shown in Figure 3.18. For a domain problem, if all the features in input space are necessary to obtain good results, then the proposed feature reduction method in input space cannot be applied. On the other hand, if the dimension of the input space is very large and it is possible to obtain good results using only a subset of features, then this feature reduction method cannot applied directly. In this case, other feature reduction methods should be used to reduce the dimension first.

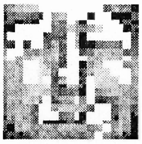


Figure 3.17 The 100 selected pixels are shown as white pixels.

For the feature reduction method in feature space, experiments have been performed on selecting 1,000 features from 1,890 features corresponding to 60 PCA values in input space and on selecting 3,500 features from 5,150 features corresponding to 100 gray values in input space for face detection. For person and car detection, experiments have been performed on selecting 3,500 features from 5,150 features corresponding to 100 Haar features in input space. Experimental results show that the proposed method works successfully. However, there is one limitation to applying the feature reduction method in feature space. The feature reduction method in feature space can only be applied if it is more efficient to implement the 2^{nd}-degree SVM in feature space than in input space, i.e., the condition $(N+1)s > (N+3)N$ must be satisfied, where N is the dimension of input space and s is the total number of support vectors.

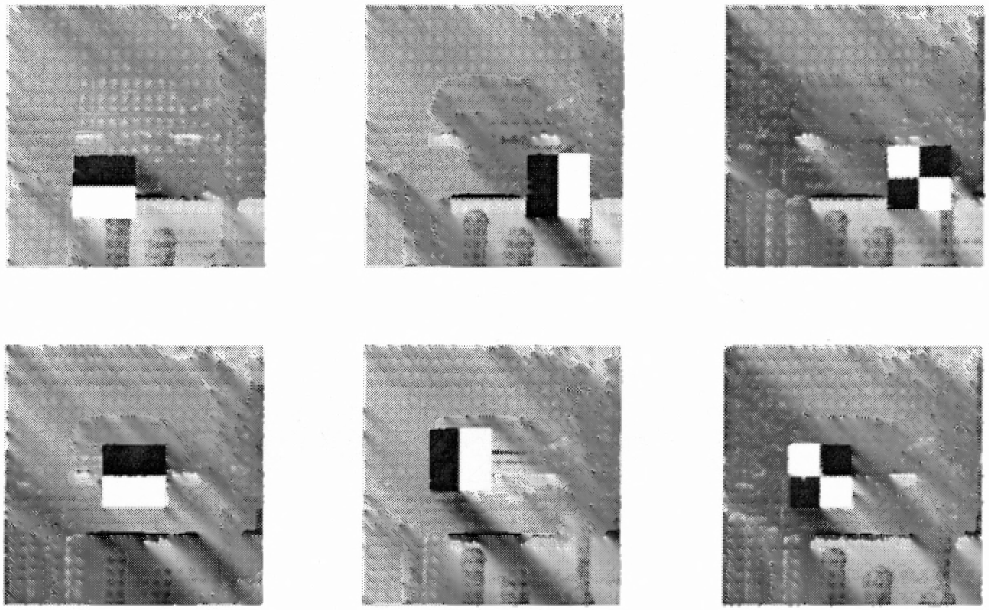


Figure 3.18 Six examples of selected Haar features.

CHAPTER 4

FACE DETECTION USING TWO-LAYER SUPPORT VECTOR MACHINES

In this chapter, a two-layer support vector machines (SVMs) method for face detection is presented. Two integral images are used for obtaining the variance quickly in each scanned window. By detecting two-eye patterns using a linear SVM, most of the background can be eliminated quickly. Two-layer second-degree polynomial SVMs are developed for face verification. The detection process is implemented directly in feature space leading to efficient face classification. Experimental results show that the proposed method can achieve better performance than other face detection methods.

4.1 Introduction

Automatic human face detection plays an important role in applications such as video surveillance, human computer interaction, face recognition, face tracking, and face image database management. Many variations contribute to the challenges of face detection problems, such as pose, structural components, facial expression, occlusion, image orientation, lighting condition, etc. Various approaches to face detection have been proposed. Yang *et al.* [76] classified face detection methods into four categories: knowledge-based, feature invariant, template matching, and appearance-based methods. Face color is also taken into consideration [26, 74].

Sung and Poggio [64] used a number of Gaussian clusters to model the distribution of face and non-face patterns. Rowley *et al.* [51] developed a neural network-based algorithm to detect upright, frontal view of faces in gray-scale images.

Schneiderman *et al.* [56] applied a Bayes classifier to estimate the joint probability of local appearance and the position of face patterns. Yang *et al.* [77] proposed a method that uses SNoW (Sparse Network of Winnows) to detect faces with different features and expressions. Viola and Jones [70] proposed a face detection algorithm using AdaBoost to train a cascade of linear classifiers and select features from an over-complete set of rectangular features. Although rectangular features can be quickly computed using integral images, it takes several weeks for training [24].

Support Vector Machines (SVMs) [14] are developed from a mathematical point of view. While most classifiers (e.g. Bayesian, neural networks, and RBF) are trained to minimize the empirical risk, SVMs are implemented to minimize the structural risk. Osuna *et al.* [45] first applied SVM to face detection. Heisele *et al.* [23] trained a 2nd-degree polynomial SVM using 10,038 faces and 36,220 non-faces. Two heuristics are used to suppress false positives. Although they achieved a high detection rate, it takes several minutes to search an image for faces at different scales as noted by Heisele *et al.* [24] who used the two following methods to speed up face detection by SVMs: hierarchical classification and feature reduction. Linear SVMs are used to reject large parts of the background, and feature reduction is achieved by ranking the contribution of features in feature space. Ma and Ding [35] proposed hierarchical SVMs which are composed of linear and nonlinear SVMs.

In this chapter, an efficient face detection system using two-layer SVMs is presented. Two-eye patterns are first detected using a linear SVM to remove large parts of background. Then two-layer 2nd-degree SVMs are used for further face verification. This chapter is organized as follows. Section 4.2 presents an overview of the proposed face detection system. The detailed steps in the face detection system are described in Section 4.3. Experimental results are shown in Section 4.4.

4.2 Overview of the Proposed Face Detection System

The overview of the proposed face detection system is illustrated in Figure 4.1. Integral images are used to quickly calculate the variance of each rectangular area, so that gray-level normalization can be performed. The variance is also used as a criterion to eliminate uniform background. In each window of size 19×19, it is verified whether its upper 7×19 part contains a two-eye pattern using a linear SVM. If the window passes the two-eye pattern verification, two-layer 2^{nd}-degree SVMs are used for further verification. Principle component analysis (PCA) is adopted to reduce dimensionality in input space.

A fixed window of size 19×19 is used for face verification. To detect faces of different sizes, the image is scaled in multi-layers. In each layer, the locations and sizes of all the positive detections are recorded. The decision values are also recorded from all the detections. The detections from all layers are combined to form the final result.

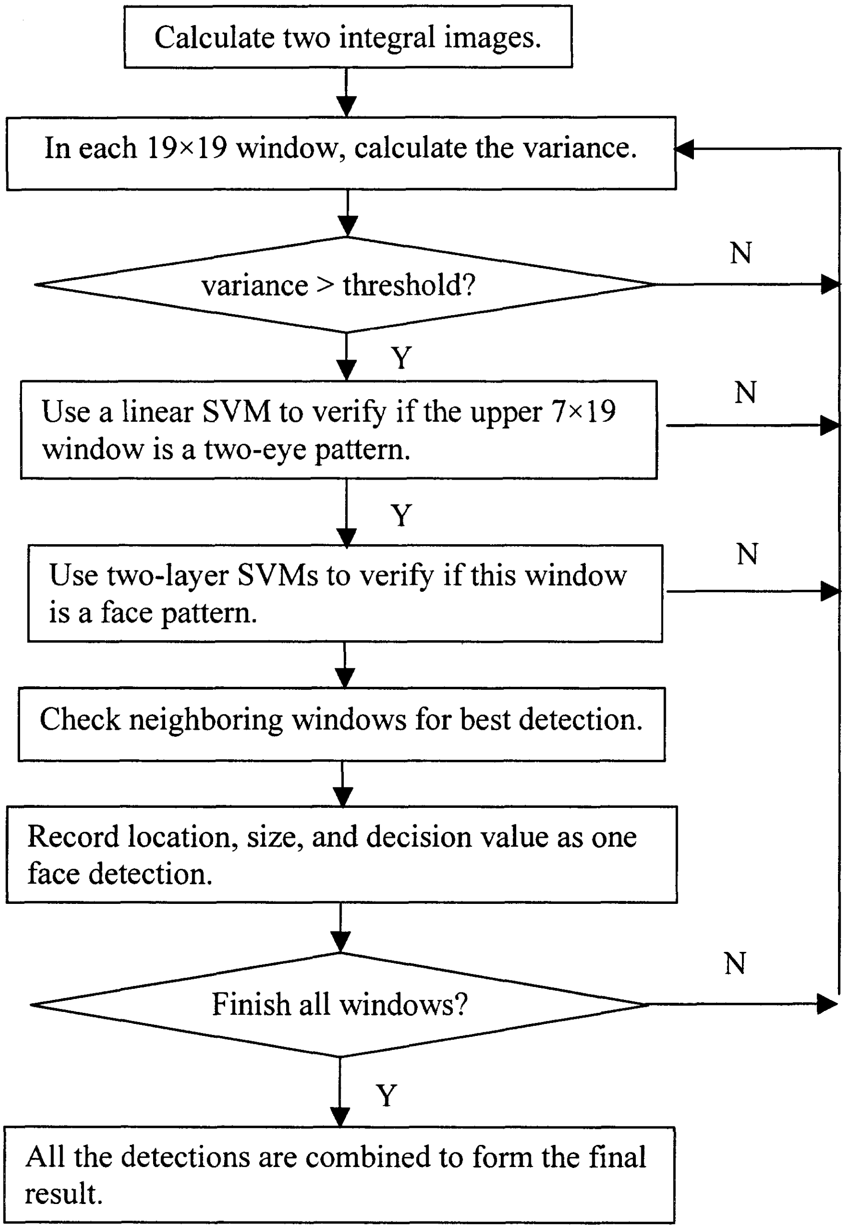

Calculate two integral images.
In each 19×19 window, calculate the variance.
variance > threshold?
N
Y
Use a linear SVM to verify if the upper 7×19 window is a two-eye pattern.
N
Y
Use two-layer SVMs to verify if this window is a face pattern.
N
Check neighboring windows for best detection.
Record location, size, and decision value as one face detection.
Finish all windows?
N
Y
All the detections are combined to form the final result.

Figure 4.1 Overview of the face detection system.

4.3 The Proposed Face Detection System

In this section, the calculation of integral images for variance, the detection of two-eye patterns, and the two-layer 2nd-degree SVMs for verifying face patterns are presented.

4.3.1 Calculation of Integral Images

In order to calculate the variance quickly in each detected window, two integral images are used. The first integral image at location (*x, y*) containing the sum of the pixels preceding it is computed as:

$$I_{sum}(x,y)=\sum_{x'=1}^{x}\sum_{y'=1}^{y}I(x',y'), \qquad (4.1)$$

where I_{sum} is the sum-integral image and $I(x',y')$ is the gray value of the original image *I* at location(x',y'). The second integral image, called the *squared-sum-integral image*, containing the sum of the squared values of the pixels preceding it is computed as:

$$I_{sqsum}(x,y)=\sum_{x'=1}^{x}\sum_{y'=1}^{y}I(x',y')^2. \qquad (4.2)$$

Two integral images can be used to quickly calculate the variance of any rectangle *ABCD* as illustrated in Figure 4.2. The sum of *ABCD* can be computed in four parts as:

$$SUM_{ABCD}=I_{sum}(x_2,y_2)-I_{sum}(x_2,y_1-1)-I_{sum}(x_1-1,y_2)+I_{sum}(x_1-1,y_1-1). \qquad (4.3)$$

The squared sum of *ABCD* can also be calculated in four parts as:

$$SQSUM_{ABCD}=I_{sqsum}(x_2,y_2)-I_{sqsum}(x_2,y_1-1)-I_{sqsum}(x_1-1,y_2)+I_{sqsum}(x_1-1,y_1-1).(4.4)$$

Therefore, the variance of *ABCD* can be easily obtained as:

$$\sigma^2_{ABCD} = \frac{SQSUM_{ABCD}}{N} - (\frac{SUM_{ABCD}}{N})^2, \tag{4.5}$$

where N denotes the total number of pixels in $ABCD$. The mean of $ABCD$ is calculated as $m_{ABCD} = SUM_{ABCD} / N$. The uniform background whose variance is smaller than a threshold is excluded and the non-uniform regions where a face pattern may exist are kept. If an input image is in the range [0, 1], the threshold is set as 0.0025. The normalized value of the pixel (x, y) in $ABCD$ can be quickly calculated as $\frac{I(x,y) - m_{ABCD}}{\sigma_{ABCD}}$.

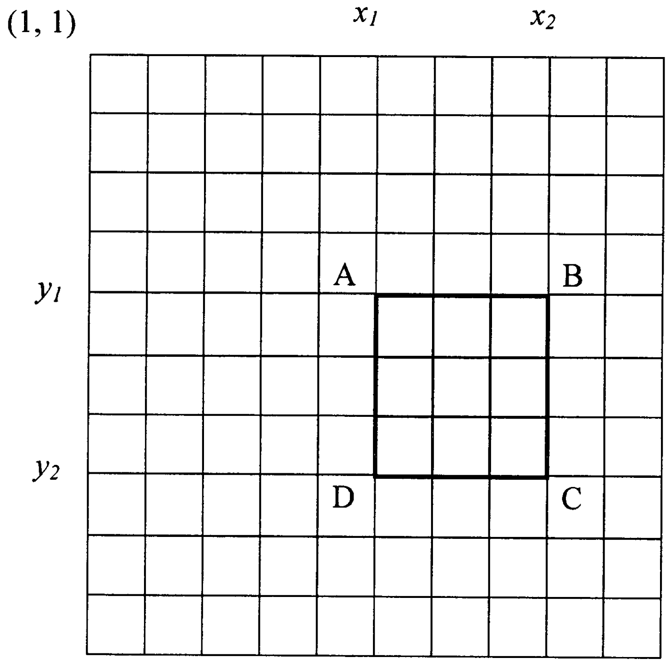


Figure 4.2 Calculation of a rectangle *ABCD*.

4.3.2 Detection of Two-Eye Patterns

The detection of two-eye patterns is applied to eliminate most of background. In each window of size 19×19, it is verified if its upper 7×19 subimage is a two-eye pattern. There are three advantages in doing this. First, the two-eye pattern contains fewer features than the face pattern, so a linear SVM can quickly achieve the classification. Second, if a one-eye pattern is detected instead of a two-eye pattern, the resolution is too low to yield good classification. Third, the geometric property of two eyes is taken into consideration in classification.

The mask for extracting the two-eye pattern pixels within a 19×19 window is shown in Figure 4.3 as the black region. In experiments, a linear SVM is trained using 1,200 positive and 6,718 negative two-eye patterns. Figure 4.4 shows the result of the two-eye pattern detector. The processing speed is about 3×10^5 windows per second when the program is implemented in a PC with a 1.4 GHz Intel Pentium-M processor. It is observed that there are some false positives, so face detection is applied on these windows containing two-eye patterns.

Figure 4.3 The mask used to extract the two-eye pattern from a 19×19 window.

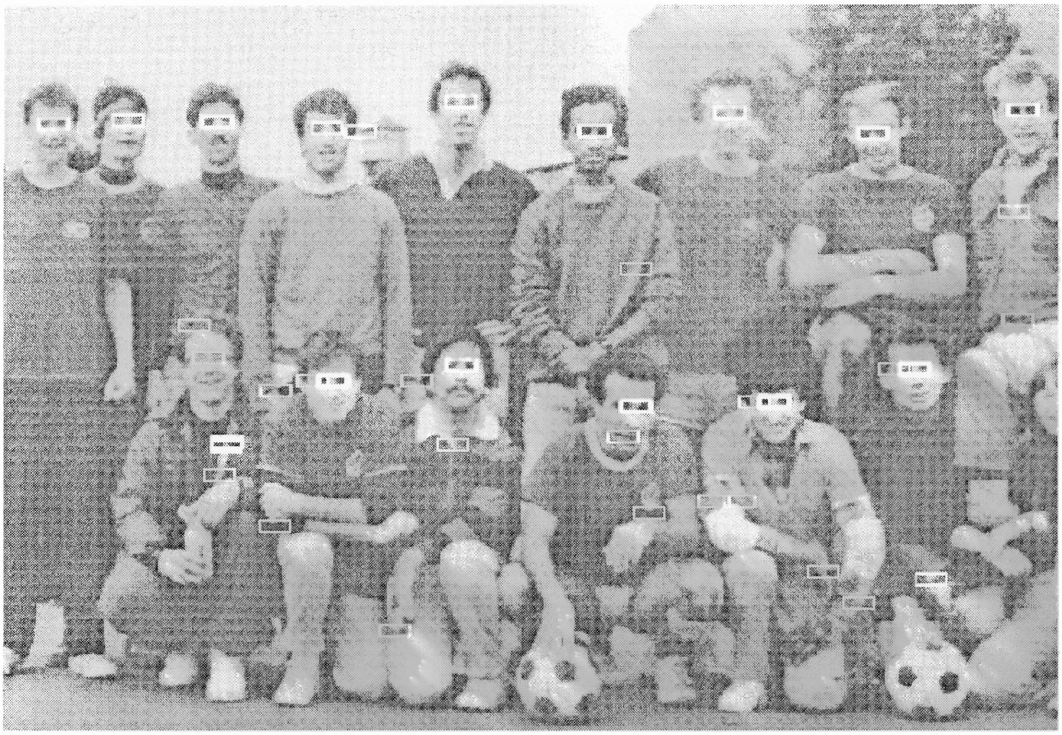


Figure 4.4 The result of two-eye detection.

4.3.3 Two-Layer 2nd-Degree SVMs

The SVMs, intended for two-class classification, use a hyperplane to maximize the margin between the hyperplane and the nearest data in each class. This hyperplane determined by training samples is called an *Optimal Separating Hyperplane* (OSH) as illustrated in Figure 4.5.

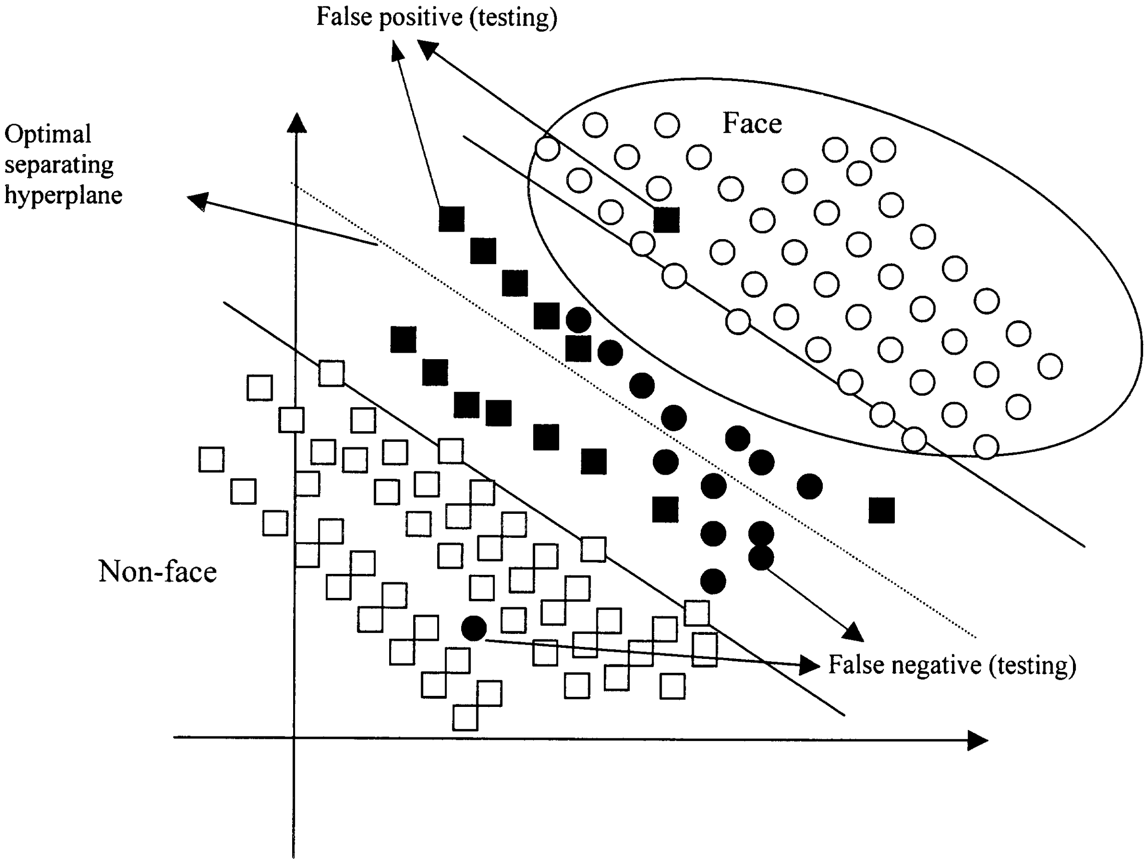


Figure 4.5 The optimal separating hyperplane (OSH) of SVM. The filled patterns are testing samples. False positive and false negative are respectively shown as a filled square and a filled circle.

It is observed that in SVM, only a small portion of training samples are support vectors, and most training samples have the decision value greater than 1. If the distribution of testing data is not very different from the training samples, it can be concluded that most of the errors in classifying testing data are generated from the patterns near OSH. Conceptually, there are two strategies to improve classification rates. The first strategy is collecting the positive and negative samples near the OSH of the first SVM, and re-training a second SVM to classify them. The second strategy is re-projecting these patterns on a second feature space that may separate them better. However, since the projection is time-consuming, it will significantly slow down the detection process. Besides, a second better feature space is difficult to find.

Principle component analysis (PCA) is adopted to reduce dimensionality in input space. In experiments, the top 60 PCA values are used to represent each sample. There are total of 2,429 face samples used: 2,000 for training and 429 for testing. There are a total of 38,801 non-face samples used: 15,228 for training and 23,573 for testing. The 2,429 face and 23,573 non-face testing samples were established by the *Center for Biological and Computational Learning* at *MIT*. In addition, 15,228 non-face training samples are randomly collected. In the following experiments, three different training methods are used for performance evaluations.

Method 1: A 2nd-degree polynomial SVM is trained using 2,000 face and 15,228 non-face samples, and then it is tested it on 429 face and 23,573 non-face images. A *Receiver Operating Characteristic* (ROC) curve is used to evaluate performance. The ROC curve is generated by shifting OSH with varying the threshold value *b*. Face classification is performed on testing samples and false positive and detection rates are calculated. The resulting ROC curve is shown in Figure 4.6, where the horizontal axis indicates the false positive rate over 23,573 non-face testing samples and the vertical axis indicates detection rate over 429 face testing samples.

Method 2: A 2nd-degree polynomial SVM is trained using 1,000 face and 10,228 non-face samples which are subsets of the training samples used in Method 1. The resulting ROC curve is also shown in Figure 4.6. It is observed that Method 2 performs worse than Method 1.

Method 3: This is the method that uses the aforementioned first strategy. From the 2,000 face training samples, the 870 face patterns are collected that are near the OSH of Method 2 (i.e., $-1 \leq f(\mathbf{x}) \leq 1$), where $f(\mathbf{x})$ denotes the decision value of pattern $\mathbf{x}$. From the 15,228 non-face training samples, the 1,715 non-face patterns are collected that are near the OSH of Method 2. Using these collected samples, a second SVM is trained. In testing, a pattern is first classified using the first SVM in Method 2. If it is near the OSH, the second SVM is used for re-classification. The resulting ROC curve is also shown in Figure 4.6. From experimental results, it is observed that the two-layer SVMs (Method 3) outperform Method 2 significantly and perform slightly better than Method 1.

In general, to construct a SVM for face detection, positive and negative samples are taken to train the SVM. The performance could be improved by adding false positives and false negatives to re-train a new SVM. However, because too many variations exist in face and non-face patterns, this method does not guarantee a better classification rate; sometimes it could be worse. Instead of adding more training samples, the training samples near the OSH of the first SVM can be collected to re-train a second SVM to improve the classification. This does not slow down the classification process much because no new features need to be extracted, and the number of patterns near the OSH of the first SVM is relatively small compared to the whole testing set.

Given a 2nd-degree polynomial SVM with a kernel $K(\mathbf{x},\mathbf{y}) = (1+\mathbf{x}\cdot\mathbf{y})^2$, the decision function for a given pattern **x** is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{s} \alpha_i y_i (1+\mathbf{x}_i \cdot \mathbf{x})^2 + b, \tag{4.6}$$

where s is the number of support vectors, $\mathbf{x}_i$ is the i-th support vector, and b is the threshold value. If N denotes the dimensionality of the input feature **x**, the number of multiplications required to calculate the decision function is $(N+1)s$.

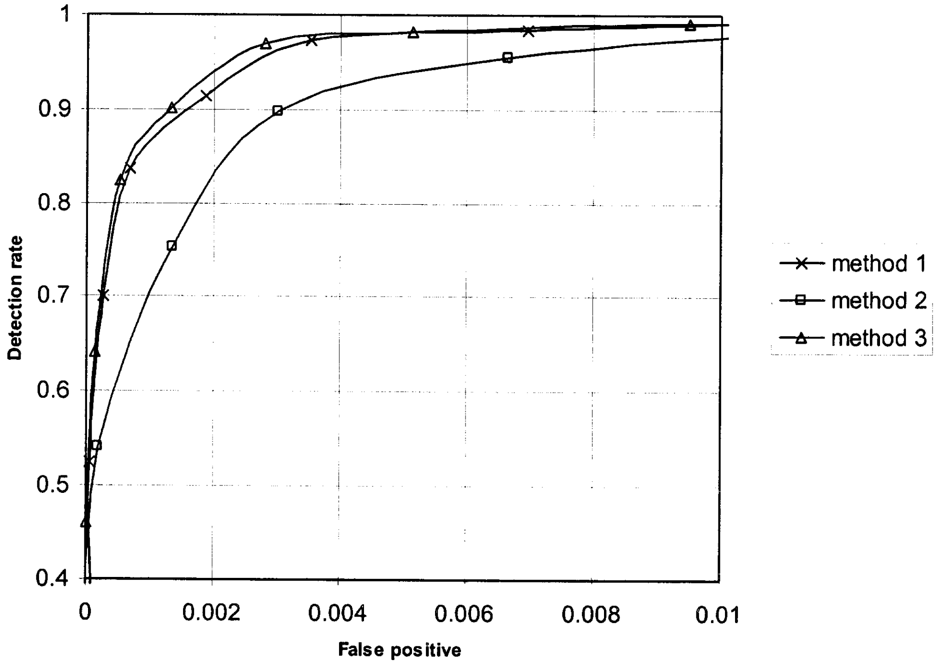


Figure 4.6 ROC curves of three training methods. The testing samples include 429 faces and 23,573 non-faces.

In feature space, the decision function $f(\mathbf{x})$ of SVM can be calculated as

$$f(\mathbf{x}) = \sum_{i=1}^{s} \alpha_i y_i (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)) + b = \mathbf{w} \cdot \Phi(\mathbf{x}) + b, \tag{4.7}$$

where

$$\mathbf{w} = \sum_{i=1}^{s} \alpha_i y_i \Phi(\mathbf{x}_i) = (w_1, w_2, \ldots, w_P) \tag{4.8}$$

is the support vector and P is the dimension in feature space. The feature space is given by

$$\Phi(\mathbf{x}) = (\sqrt{2}x_1, \ldots, \sqrt{2}x_N, x_1^2, \ldots, x_N^2, \sqrt{2}x_1 x_2, \ldots, \sqrt{2}x_{N-1}x_N), \tag{4.9}$$

of dimension $P = N(N+3)/2$. The number of multiplications required to calculate the decision function is $(N+3)N$. If $(N+1)s > (N+3)N$, it is more efficient to implement the 2nd-degree polynomial SVM in feature space. In experiments, $N = 60$ and $s > 1{,}000$, so it is faster to implement the SVM in feature space. The processing speed is about 1.0×10^4 windows per second when the program is implemented on a PC with a 1.4 GHz Intel Pentium-M processor.

If the detected patterns are different from faces, the decision values will be small. As shown in Figure 4.7, there are three faces detected. Note that the incomplete-face images cannot be detected. The decision values of an 8×8 region at the uppermost face are shown in Table 4.1, and at the image center (where a face is not present) are shown in Table 4.2. From Table 4.1, it is observed that four positive values appear on the center of the uppermost face, and the values decrease as the pixels move away from the center of the face. From Table 4.2, because no face is present, the decision values are all negative. Therefore, non-face patterns can be eliminated quickly to speed up the detection process.

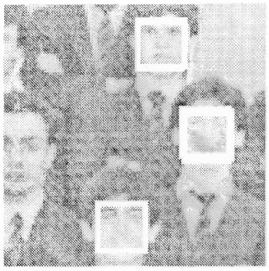


Figure 4.7 An example of three faces.

Table 4.1 The Decision Values of an 8×8 Region at the Uppermost Face

-8.9708	-7.8361	-7.2915	-7.2165	-7.7274	-8.4624	-8.9582	-8.5790
-9.6290	-8.3149	-6.7118	-6.2958	-6.9922	-8.3298	-9.4971	-9.0373
-10.1120	-8.9096	-5.8226	-3.9230	-4.8727	-7.6357	-11.1600	-11.7300
-9.7307	-9.1459	-3.9018	1.4697	1.3517	-2.5977	-9.3476	-12.7520
-7.3816	-7.0983	-3.3209	1.2614	1.8612	-1.6981	-7.9363	-12.3280
-5.7875	-4.8310	-2.4996	0.1457	-0.7831	-4.5756	-9.0572	-13.6200
-5.8639	-4.8173	-3.8848	-2.8779	-4.5553	-7.1203	-9.9464	-13.6260
-6.5973	-4.2707	-2.3995	-2.2787	-4.4652	-5.6277	-8.1469	-12.0170

Table 4.2 The Decision Values of an 8×8 Region at the Image Center

-5.7541	-6.8029	-9.0473	-11.4970	-10.5690	-8.8814	-8.2135	-9.5539
-6.3197	-6.3491	-8.8044	-12.0910	-10.8470	-8.2184	-6.4213	-7.2828
-7.1943	-6.5021	-8.2114	-10.5800	-9.8959	-6.9134	-5.8686	-6.2791
-9.3905	-7.3555	-7.7201	-8.7436	-8.3543	-6.1919	-6.0401	-6.3148
-18.2860	-15.8100	-11.4560	-7.4753	-6.7675	-5.8251	-6.0124	-6.0964
-20.1950	-18.7740	-16.9520	-14.3150	-12.4140	-10.8270	-8.7379	-7.2143
-17.7550	-16.6750	-16.1570	-16.4940	-17.2500	-17.2340	-16.5530	-14.9390
-15.7940	-14.6680	-15.0050	-16.8380	-17.8560	-18.8550	-21.1670	-21.6420

4.4 Experimental Results

In experiments, the first layer SVM is trained using 6,029 face samples; among which 2,429 are from MIT and the others are from FERET database. The 18,065 non-face training samples were collected using the bootstrap strategy. After training the 2nd-degree SVM using 60 PCA values, 457 and 1,015 support vectors, respectively, for face and non-face classes are obtained.

From 2,032 face samples, 483 faces near the OSH of the first SVM are collected, in which 9,253 non-face samples are also collected. The second layer SVM is trained using these collected face and non-face samples and the support vectors from the first SVM.

A test set containing 155 faces in 23 images is used. By applying only the first SVM, the detection rate of 83.23% with 12 false positives is achieved. Using the two-layer SVM system, the detection rate of 89.68% with 13 false positives can be achieved. Therefore, the two-layer SVM system works better than just one SVM. Figure 4.8 shows three examples. In the first image, all the faces are detected correctly. In the second image, all the faces are detected with one false positive. In the third image, one face is missed because it is too dark. Table 4.3 lists the performances of several face detection systems on the same test set. The proposed method achieves better performance than the distribution-based [64], neural network [51], and other SVM [23, 45] methods. The results of Bayes [56] and SNoW-based [77] methods seem better than the proposed method; however, they only tested on 20 images with 136 faces, and it is unclear what heuristics they used to suppress false positives.

The second layer SVM will not significantly slow down the algorithm as compared with only one layer SVM. First, the number of patterns needed for re-verification using the second SVM is relatively small. Second, 60 PCA features are used as the representation corresponding to 1890 features in feature space. When the second SVM is used for classification, the 1890 features from the first SVM are ready to use. To obtain the decision values, 1890 extra multiplications are needed. This is relatively small compared with the number of multiplications needed for computing the 60 PCA values, which is $283\times60=16,980$.

As aforementioned in Section 4.3.2, the speed of the linear SVM for two-eye detection is about 3.0×10^5 windows/sec, and of the 2^{nd}-degree SVM is about 1.0×10^4 windows/sec. In the proposed system, the variance is used to eliminate background and the decision values are used to eliminate adjacent windows. It takes 0.3 second to process an image of 320×240 using a 19×19 window. In [24], an average of 4 frames/sec is achieved for an image of 320×240 at five different scales from 30×30 to 70×70. However, their system produces a high false positive rate. The proposed system can achieve 3 frames/sec. An additional condition such as the two-eye region is often darker than the region under it can be used to achieve 5 frames/sec with only a slight decrease in the classification rate.

Table 4.3 Performance Comparisons of Several Face Detection Systems

System	Detection Rate	False Positives
[Sung 96] MIT Distribution-based	81.9%	13
[Osuna 97] MIT SVM	74.2%	20
[Rowley 98] CMU Neural Network	84.5%	8
[Schneiderman 98] CMU Bayes*	91.2%	12
[Yang 2000] UIUC SnoW*	94.1%	13
[Heisele 2000] MIT SVM	84.7%	11
The proposed system	89.68%	13

Note: "*" indicates that the images of hand-drawn faces and cartoon faces are excluded, using a total of 20 images with 136 faces.

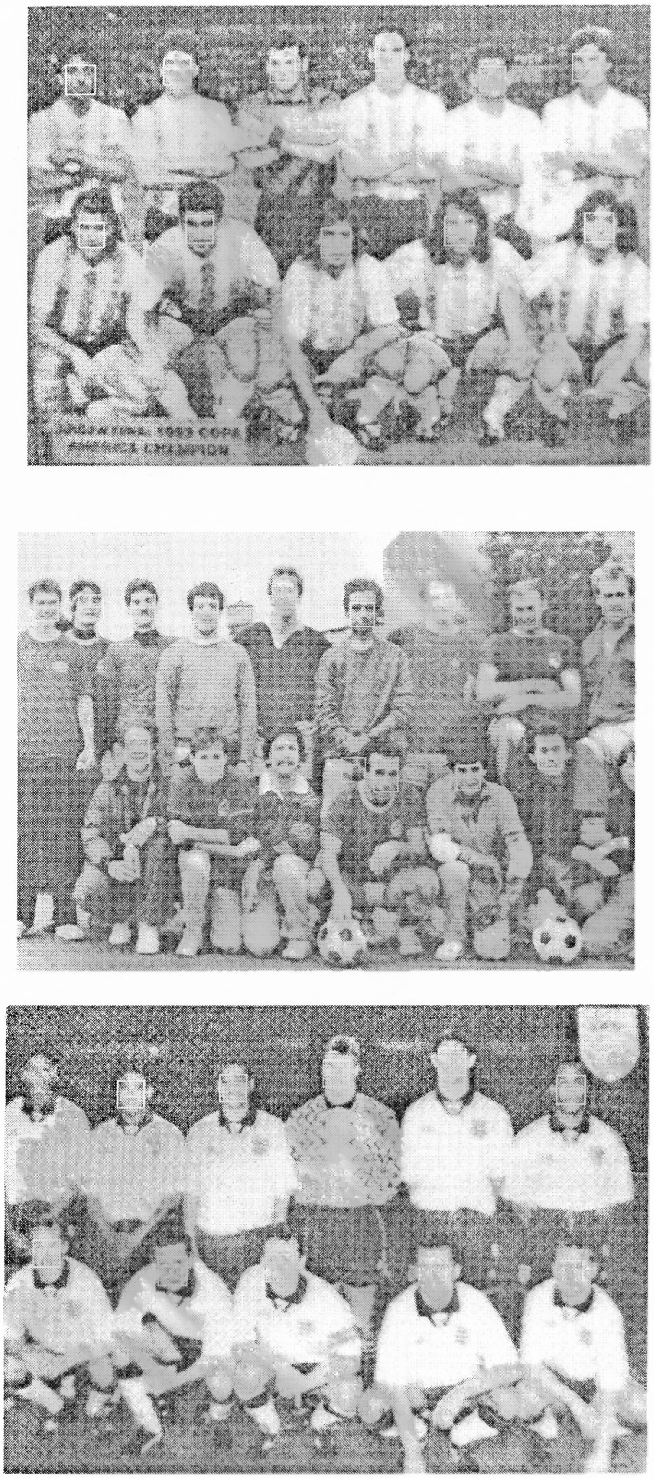


Figure 4.8 Face detection examples.

CHAPTER 5

AN IMPROVED INCREMENTAL TRAINING ALGORITHM FOR SUPPORT VECTOR MACHINES USING ACTIVE QUERY

In this chapter, an improved incremental training algorithm is proposed for support vector machines. Instead of selecting training samples randomly, they are divided into groups and the *k*-mean clustering algorithm is applied to each group to collect the initial set of training samples. In active query, a weight is assigned to each sample according to its distance to the current separating hyperplane and the confidence factor. The confidence factor, calculated from the upper bounds of errors of the SVM, is used to indicate the degree of closeness of the current separating hyperplane to the optimal solution. A criterion is proposed to incrementally eliminate some non-informative training samples. The proposed algorithm has been successfully applied on artificial and real data sets. The results show its robustness by comparing with other methods.

5.1 Introduction

The *Support Vector Machine* (SVM) is intended to generate an optimal separating hyperplane by minimizing the generalization error without using the assumption of class probabilities as does a Bayesian classifier [5, 14, 67, 68]. Its training procedure usually requires huge amounts of memory space and significantly time-consuming computation because an enormous amount of training data and quadratic programming are used. The decision hyperplane of SVM is determined by the most informative data instances, called *Support Vectors* (SVs). In practice, these SVs are only a small subset of the entire training data. By applying feature reduction in both input and feature space, a fast non-linear SVM can be designed without a significant loss in performance [24, 61]. Therefore, some researchers proposed incremental training or active learning to shorten the training time [2, 6, 39, 44, 57, 65].

Syed *et al.* [65] developed an incremental learning procedure by partitioning the training data set into subsets. At each incremental step, only the support vectors are added to the training set in next step. Cambell *et al.* [6] proposed a query learning strategy by iteratively requesting the label of a sample which is the closest to the current hyperplane. Schohn and Cohn [57] presented a greedy optimal strategy by calculating class probability and expected error. A simple heuristic was used by selecting the training examples according to their proximity to the dividing hyperplane.

Moreover, An *et al.* [2] developed an incremental learning algorithm by extracting initial training samples as margin vectors of the two classes, selecting new samples according to their distances to the current hyperplane, and discarding training samples without contribution to the separating plane. However, only the distance factor is considered. Nguyen and Smeulders [44] pointed out that prior data distribution is useful for active learning, and new training data are selected from the samples having the maximal contribution to the current expected error.

Most existing methods of active learning perform sample based on its proximity to the current separating hyperplane. Mitra *et al.* [39] presented probabilistic active learning using a distribution determined by the current separating hyperplane and a confidence factor. They did not provide the criteria for selecting test samples for calculating the confidence factor. There are two issues in their algorithm. First, the initial training samples are selected randomly which may cause the initial hyperplane to be far away from the optimal solution. Second, using only the support vectors from the previous training set may lead to bad performance in some cases.

In this chapter, an improved incremental training algorithm is presented for SVMs using active query. The rest of this chapter is organized as follows. In Section 5.2, the proposed incremental training algorithm is presented. In Section 5.3, experimental results and comparisons with other active learning methods are provided.

5.2 The Improved Incremental Training Algorithm

In this section, the methods of the initial training samples selection using the *k*-mean clustering algorithm, the active query of most informative samples, and the criterion for eliminating non-informative training samples are presented.

5.2.1 Selection of Initial Training Samples

In most of the incremental training methods for SVMs [6, 39, 57], the initial training samples are selected randomly. In a simple case of the positive and negative training samples being two separable clusters, there will be no problem using the random selection. However, in a complex case of the samples of one class being distributed crossing several clusters, it is possible to obtain a final hyperplane that is far away from the optimal solution if the random selection is used.

Figure 5.1 shows an artificial data set containing two classes: 1,000 points marked as "+" and 1,100 points marked as "o". The "o" points are distributed over two clusters: one with 1,000 points and the other with 100 points. The dashed line indicates the decision boundary of the SVM training using 1,000 "o" and 1,000 "+" training samples. The solid line indicates the decision boundary of the SVM training using 1,100 "o" and

1,000 "+" training samples. It is observed that the solid line represents the optimal hyperplane.

In incremental training, the samples near the hyperplane of the current SVM are queried and added to the current training set to form the new training set for the next step. If the initial training samples are selected randomly, a poor decision boundary may be obtained as the dashed line in Figure 5.1. This experiment has been conducted for 10 times by taking a random 10% of the samples in each class as the initial training samples. Results show the dashed line is chosen for four times and the solid line for six times. Obviously, it is unreliable for using a random initial set of training samples to train the SVM incrementally.

Instead, a method is designed to select the initial training samples by considering the distribution of the training samples. The positive and negative training samples are randomly separated into K groups respectively. Each group is separated into k sub-clusters (such as k=10) using a k-mean clustering algorithm. From each of the k clusters, one sample that is nearest to the mean of each cluster is chosen. In this way, around $1/k$ of the total training samples are collected as the initial training samples. The variables K and k are selected according to the size of the total training samples. In this experiment, the values of K=10 and k=10 are used. Using this method, generating poor decision boundary is avoided and almost the same boundary is obtained as using batch training, that is the solid line in Figure 5.1.

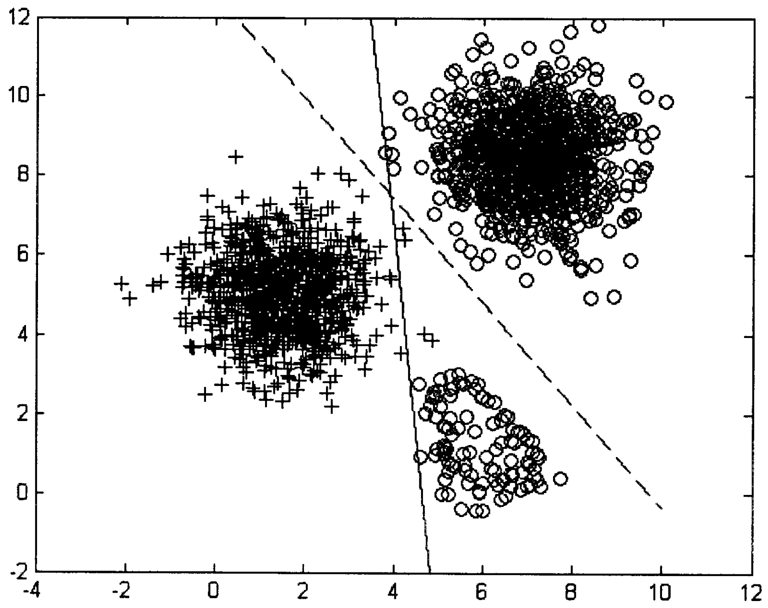


Figure 5.1 An artificial data set containing two classes: 1,000 "+" and 1,100 "o". The solid line indicates the decision boundary training using all samples; the dashed line indicates the decision boundary training using 1,000 "+" and 1,000 "o" samples.

5.2.2 Active Query of New Training Samples

Suppose there are l patterns, and each pattern consists of a pair: a vector $\mathbf{x}_i \in \mathbf{R}^n$ and the associated label $y_i \in \{-1, 1\}$. Let $X \in \mathbf{R}^n$ be the space of patterns, $Y = \{-1, 1\}$ be the space of labels, and $p(\mathbf{x}, y)$ be a probability density function on $X \times Y$. The machine learning problem consists of finding a mapping: $sign(f(\mathbf{x}_i)) \to y_i$, which minimizes the error of misclassification. Let f be the decision function. The $sign$ function is defined as

$$sign(f(\mathbf{x})) = \begin{cases} 1 & f(\mathbf{x}) \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{5.1}$$

The expected value of the actual error is given by

$$E[e] = \int \frac{1}{2} \left| y - sign(f(\mathbf{x})) \right| p(\mathbf{x}, y) d\mathbf{x} dy. \tag{5.2}$$

However, $p(\mathbf{x}, y)$ is not readily known in practice, and the suitable function f must be inferred from the training set. The generalization performance of the function f is controlled by two factors: the training error and the capacity of the machine which is measured by its Vapnik Chervonenkis (VC) dimension [5]. Let us choose some η with $0 \leq \eta \leq 1$, then with probability 1- η the following bound holds [68]:

$$e \leq e_t + \sqrt{(\frac{h(\log(2l/h)+1) - \log(\eta/4)}{l})}, \tag{5.3}$$

where h is the VC dimension, and the e_t is the error tested on the training set

$$e_t = \frac{1}{2l}\sum_{i=1}^{l} | y_i - sign(f(\mathbf{x}_i)) |. \tag{5.4}$$

For two-class classification, SVMs use a hyperplane that maximizes the margin (i.e., the distance between the hyperplane and the nearest data point of each class). This hyperplane is viewed as an *Optimal Separating Hyperplane* (OSH). If the data are not linearly separable in the input space, a non-linear transformation function $\Phi(\cdot)$ is used to project $\mathbf{x}_i$ from the input space $\mathbf{R}^n$ to a higher dimensional feature space $\mathbf{R}^d$. An OSH is constructed in the feature space by maximizing the margin between the closest points $\Phi(\mathbf{x}_i)$. The inner-product between two projections is defined by a kernel function $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$. The commonly used kernels include polynomial, Gaussian *Radial Basis Function* (RBF), and a Sigmoid kernels.

Constructing an optimal hyperplane for SVM is a quadratic programming problem which maximizes

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{5.5}$$

subject to $\sum_{i=1}^{l} \alpha_i y_i = 0$, and $0 \le \alpha_i \le C$, where α_i is Lagrange multiplier for pattern $\mathbf{x}_i$, and C is a positive value determining the constraint violation during the training process. After training, the decision function of the SVM can be written as

$$f(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b\,, \tag{5.6}$$

where $\mathbf{w}$ is the constructed support vector in feature space, which can be obtained by

$$\mathbf{w} = \sum_{i=1}^{s} \alpha_i y_i \Phi(\mathbf{x}_i) = (w_1, w_2, \ldots, w_d)\,. \tag{5.7}$$

where s is the number of support vectors and d is the dimension in feature space.

An upper bound for the expected error of an SVM classifier is given by [67]

$$E[e] \le \frac{E[D^2 / M^2]}{l}\,, \tag{5.8}$$

where D is the diameter of the minimum sphere enclosing all the training samples in the feature space, and M is the margin of the SVM classifier. If the training samples are separated without errors by an optimal hyperplane, another upper bound for the expectation value of error for SVM on the test examples is given by the ratio between the expectation value of the number of support vectors and the number of training samples [14]

$$E[e] \le \frac{E[s]}{l}\,, \tag{5.9}$$

where s is the number of support vectors.

Therefore, there are three ways to estimate the test error for SVM by using Equations (5.3, 5.8, and 5.9). Using Equation (5.3), the values of η and the VC dimension h are needed. The value of η is from 0~1, and usually it is a small number. The value

$\eta = 0.01$ is used, which means with probability 99% Equation (5.3) holds. Since the VC dimension of the set of oriented hyperplanes in $\mathbf{R}^n$ is n+1 [5], the VC dimension for SVM in feature space $\mathbf{R}^d$ is $h = d + 1$. If the dimension d can be obtained explicitly, then the Equation (5.3) can be used to calculate the upper bound of the actual error. For a linear SVM, it is obvious that the VC dimension is$h = n + 1$. For a second-degree polynomial SVM: with kernel

$$K(\mathbf{x},\mathbf{y}) = (1 + \mathbf{x} \cdot \mathbf{y})^2, \tag{5.10}$$

the corresponding vector in feature space is given by

$$\Phi(\mathbf{x}) = (\sqrt{2}x_1,\ldots,\sqrt{2}x_n, x_1^2,\ldots,x_n^2,\sqrt{2}x_1x_2,\ldots,\sqrt{2}x_{n-1}x_n), \tag{5.11}$$

with dimension of$d = n(n+3)/2$. Therefore the VC dimension for the second-degree polynomial SVM is$h = n(n+3)/2+1$. For a Gaussian RBF SVM with kernel

$$K(\mathbf{x},\mathbf{y}) = \exp(-\frac{(\mathbf{x}-\mathbf{y})^2}{\sigma^2}), \tag{5.12}$$

the value for h could not be obtained, since the pattern in the input space could not be projected to the feature space explicitly and the VC dimension can be infinite.

Using Equation (5.8), the values of D and M are needed. To compute the diameter D of the smallest sphere in feature space which encloses the mapped training data, $(D/2)^2$ is needs be minimized subject to:

$$\left|\Phi(\mathbf{x}_i) - \mathbf{O}\right|^2 \le (D/2)^2 \quad \forall i, \tag{5.13}$$

where $\mathbf{O}$ is the unknown center of the sphere. This is a convex quadratic programming problem. For large data sets with high dimensionality, it requires large memory space and it is time consuming. To avoid solving the quadratic programming problem, each attribute of the training data is normalized in the range between 0 and 1. For linear SVM, if each sample has n normalized attributes, all of the training set will be within a n-dimensional cube of length 1. The diameter D can be approximated as $\sqrt{n}$. For a second-degree polynomial SVM with the input space of dimension n and kernel as Equation (5.10), the dimension in feature space will be $d = n(n+3)/2$ as shown in Equation (5.11). Since the original input space is normalized to be in the range between 0 and 1 for each dimension, in feature space all the training samples will be within a d-dimensional cube of length $\sqrt{2}$. Therefore, the diameter D can be approximated as $\sqrt{2d}$. The margin M in Equation (5.8) can be computed as [14]

$$M = \frac{2}{\|\mathbf{w}\|}, \tag{5.14}$$

where $\|\mathbf{w}\|$ is the norm of the support vector $\mathbf{w}$. For linear SVM and second-degree polynomial SVM, the support vector $\mathbf{w}$ can be constructed using Equation (5.7). For Gaussian RBF SVM, since the projected vector in feature space could not be obtained explicitly and the values for D and M could not be calculated.

The calculation of Equation (5.9) is simple; it is not related to the dimension of the feature space. Therefore, it is easily applicable to linear, polynomial and Gaussian RBF SVMs. In the training process, the purpose is to select a subset of the whole training set and achieve similar performance as using the whole training set. To estimate the actual error of the SVM in each incremental step using only a selected subset of training samples, Equation (5.8) is adjusted as

$$e \le e_t + \frac{D^2 / M^2}{l}, \tag{5.15}$$

and Equation (5.9) as

$$e \leq e_t + \frac{s}{l}. \tag{5.16}$$

In Equations (5.3, 5.15, and 5.16), the e_t is calculated from the whole training set, and l is the number of the set of training samples. In Equation (5.16), s is the number of support vectors for the current training set. The term e_t is added in Equations (5.15 and 5.16) because only a subset of the whole training set is used at each incremental training step and the assumption on Equations (5.8 and 5.9) is that the training samples can be separated by the optimal hyperplane without error.

The minimum of Equations (5.3, 5.15 and 5.16) is taken as the actual error, that is

$$e = \min(e_t + \sqrt{(\frac{h(\log(2l/h)+1)-\log(\eta/4)}{l})},\ e_t + \frac{(D^2/M^2)}{l},\ e_t + \frac{s}{l},\ 0.5). \tag{5.17}$$

Another term e_{opt} is defined as the error for the optimal hyperplane based on the whole training data set estimated from the current SVM using only a subset of the whole training set. Since the hyperplane of SVM is obtained by minimizing the train error and maximizing the margin, the e_{opt} is estimated by

$$e_{opt} = \frac{m}{l}, \tag{5.18}$$

where m is the number of samples tested as errors from the current training sample set, and l is the number of training samples of the whole training set. The observation is used that the optimal hyperplane based on the whole training set could not be better than the current one using a subset assuming that all the unused samples are classified correctly by the current SVM. Note that if the current SVM can classify the current training samples without error, then $e_{opt} = 0$.

A confidence factor to control the selection of new training samples is defined as

$$C = \frac{1-e}{1-e_{opt}}. \tag{5.19}$$

Define a parameter as

$$\alpha = \frac{1}{2}\ln[C/(1-C)] \tag{5.20}$$

Figure 5.2 shows how α varies with *C*. The value 0.5 is put in Equation (5.17), which means the worst performance of the trained classifier is 50%. This can ensure the confidence factor $C > 0.5$ and the parameter $\alpha > 0$.

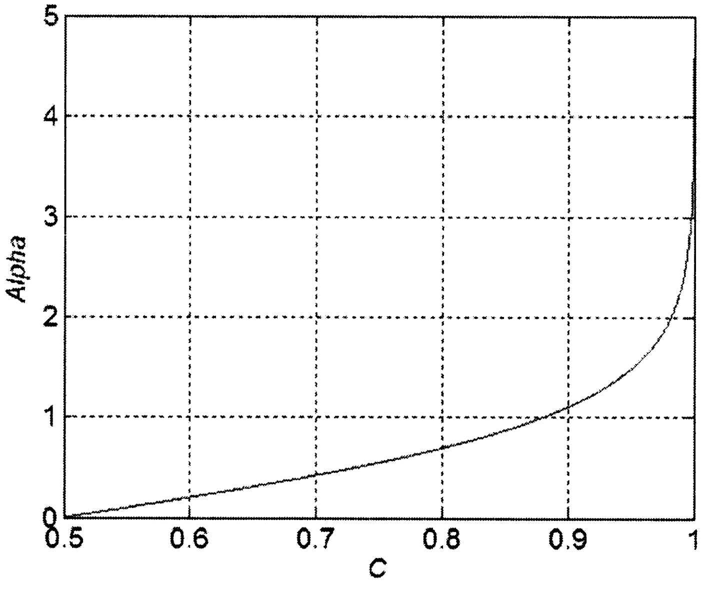


Figure 5.2 α varies with *C*.

After each incremental training step, every unused sample receives a weight that determines its probability of being selected for the next training set. The weight is defined as

$$W(\mathbf{x}_i) = \begin{cases} e^{-|\alpha \times f(\mathbf{x}_i)|} & \text{if pattern } \mathbf{x}_i \text{ is correctly classified;} \\ e^{(1-\alpha)\times|f(\mathbf{x}_i)|} & \text{otherwise.} \end{cases} \tag{5.21}$$

where $f(\mathbf{x}_i)$ is the decision value of pattern $\mathbf{x}_i$ using the trained SVM by Equation (5.6). From Equation (5.21), the chance of a pattern being selected for training is controlled by the α and decision value. From Equation (5.20) and Figure 5.2, it can be seen that the value of α increases as C increases, while high C value means high confidence for the current classification, such as when α =0.5, C=0.7311, when α =1, C=0.8808, when α =3, C=0.9975. If a pattern is classified correctly, its weight is low if both α and the decision values are high. Otherwise, if a pattern is not correctly classified, when α <1, a high value will be assigned, which means that the wrongly classified patterns are more important when the confidence is low. When α >1, the wrongly classified patterns near the OSH are more important. New training samples will be selected according to their weights.

In AdaBoost learning algorithm [16], a similar strategy is used to assign weights to each training sample. The difference between the proposed method and the AdaBoost is that in AdaBoost the final classification decision is based on the combination of the outputs of a chain of different classifiers, while the proposed method obtains only one final classifier. The proposed method also differs from other near-boundary methods. The near-boundary methods select the samples nearest the current separating hyperplane, while the proposed selection method is controlled by the confidence factor. From Equation (5.21), it can be seen that when α <1, the wrongly classified patterns are first selected, with the higher the $|f(\mathbf{x})|$ value, the higher the weight. When α >1, the wrongly classified patterns near the OSH are selected first, then the correctly classified patterns near the OSH, and the weights are also controlled by the decision values. The training process will stop until no more informative training samples are available. For informative training sample, the threshold value for weight is set to be e^{-2} according to the experimental results by considering both the confidence factor and the distance between the sample and the current separating hyperplane.

5.2.3 Elimination of Non-informative Training Samples

In the incremental training step of active learning for the SVM, keeping only the support vectors of previous training samples for next step may lead to bad performance. On the other hand, keeping all the previous samples may lead to large active training samples. A method is designed to actively eliminate some training samples that are not informative. Firstly, a threshold value is set using the decision value. If for one sample with $|f(\mathbf{x})| < 1.1$, it is near the hyperplane and it is kept. Secondly, according to the Equation (5.21), a weight is assigned to each sample in the current training set. A threshold value $t = e^{-2}$ is set. If the weight of one training sample is higher than *t*, then it is kept for the next step training. By this strategy, these samples with decision value higher than 1.1 and weight less than e^{-2} are eliminated from the active training samples. Note that the second condition is not enough to eliminate samples. In some cases, during the training, high α value may be obtained when the confidence factor *C* is high, such as when *C* > 0.9975, α > 3. If only $t = e^{-2}$ is used as the threshold value, all the current training samples may be eliminated.

5.3 Experimental Results

In this section, the proposed improved incremental training algorithm is applied on the SVMs with three types of kernels: linear, second-degree polynomial SVM as shown in Equation (10), and the Gaussian RBF SVM as shown in Equation (12). Note that $\sigma = 1$ is used in Equation (12).

The test sets are listed in Table 5.1. The face data set contains 2,429 faces and 15,228 non-faces. The 2,429 faces are downloaded from the *Center for Biological and Computational Learning* (CBCL) at *Massachusetts Institute of Technology* (MIT). Totally 15,228 non-face training samples are randomly collected from the images that do not contain faces. The size of all these samples is 19 × 19. Each sample is represented as the top 60 PCA values. All the other data sets are available in the *University of California, Irvine* (UCI) machine learning repositories [4].

Table 5.1 Number of Attributes and Number of Samples for the Data Sets

Data Set	Num of Attributes	Num of Class 1	Num of Class 2	Total Num
Face	60	2,429	15,228	17,657
Cancer	9	444	239	683
Diabetes	8	500	268	768
Ionosphere	34	225	126	351
Heart	13	150	120	270
German	24	700	300	1,000

The proposed method is compared with five other methods. The first method is batch learning which is using all the available positive and negative training samples to train SVM. It is indicated as BatchSVM in the tables. Notice that in real world, if there are too many training samples, this method could be used. The second method is that at each incremental step, a number of new training samples are just randomly selected. It is indicated as IncrSVMRandom in the tables. The third method is to query new training samples by collecting the samples near the current separating hyperplane. It is indicated as QuerySVM in the tables. The fourth one is the method in [65], it is indicated as SubsetSVM in the tables. The fifth one is the method proposed in [39]. The whole training samples are used to calculate the confidence factor. It is indicated as QueryStat in the tables.

As shown in Table 5.1, each dada set contains two classes. The value of each attribute is normalized in the range between 0 and 1. The experiment is performed 10 times, with each time take different 90% samples from Class 1 and Class 2 as the training samples, and the rest of 10% samples are taken as test samples. The average of the 10 classification rates is taken as the overall classification rate. As described in Section 5.2.2, for linear and second-degree polynomial SVM, the VC dimension h, the diameter D enclosing the training set, and the margin M can be calculated. Therefore Equation (5.17) could be used to calculate the error. The comparison of performance of different methods of using linear SVM to do the training and testing is shown in Table 5.2. The results of using second-degree polynomial SVM are shown in Table 5.3. For Gaussian RBF SVM, the values for h, D and M could not be obtained, since the pattern in the input space could not be projected to the feature space explicitly and the VC dimension can be infinite. Therefore, only Equation (5.16) is used to estimate the actual error. The results of using radial basis function SVM are shown in Table 5.4.

The BatchSVM is used as the basis to do comparison. For each other method, the average of the differences of the classification rates of all the data sets compared with the BatchSVM is calculated. The average values of differences are shown at the last rows of Table 5.2, 5.3, and 5.4. The results show that the overall performance of the proposed methods outperforms other methods. For IncrSVMRandom method, the same number of training samples as QuerySVM method are used. It can be seen that in most cases, IncrSVMRandom method performs worse than QuerySVM method. For linear SVM and Gaussian RBF SVM, the proposed method even outperforms the BatchSVM slightly. The QueryStat performs badly in the experiments because this algorithm only keeps the support vectors of the previous training samples. For some cases, this may perform well, such as the Cancer data set using linear SVM, the Diabetes data set using Gaussian RBF SVM. However, for other cases, it may perform badly, such as Face data set using linear SVM and the German data set using polynomial SVM.

Table 5.2 Comparison of Classification Rates of Different Methods Using Linear SVM

Method Data Set	BatchSVM	IncrSVM Random	QuerySVM	SubsetSVM	QueryStat	The Proposed Method
Face	0.9618	0.9392	0.9594	0.9447	0.9096	0.9616
Cancer	0.9716	0.9701	0.9716	0.9687	0.9716	0.9731
Diabetes	0.7684	0.7524	0.7605	0.7711	0.7763	0.7711
Ionosphere	0.8824	0.8559	0.8706	0.85	0.8676	0.8824
Heart	0.8407	0.8222	0.8444	0.8407	0.8407	0.8407
German	0.7660	0.7440	0.7510	0.7510	0.7060	0.7720
Ave. Diff	0	-0.0178	-0.0056	-0.0108	-0.0198	+0.0017

Table 5.3 Comparison of Classification Rates of Different Methods Using 2nd-degree Polynomial SVM

Method Data Set	BatchSVM	IncrSVM Random	QuerySVM	SubsetSVM	QueryStat	The Proposed Method
Face	0.9858	0.9077	0.9815	0.9689	0.9605	0.9863
Cancer	0.9716	0.9701	0.9701	0.9716	0.9642	0.9716
Diabetes	0.7789	0.7611	0.7671	0.7724	0.7303	0.7763
Ionosphere	0.9176	0.8647	0.8618	0.8676	0.8824	0.9176
Heart	0.8148	0.8037	0.8037	0.7741	0.7852	0.8259
German	0.7460	0.7030	0.7280	0.7030	0.6850	0.7310
Ave. Diff	0	-0.0341	-0.0171	-0.0262	-0.0345	-0.0010

Table 5.4 Comparison of Classification Rates of Different Methods using Gaussian RBF SVM

Method Data Set	BatchSVM	IncrSVM Random	QuerySVM	SubsetSVM	QueryStat	The Proposed Method
Face	0.9864	0.9533	0.9854	0.9789	0.9334	0.9863
Cancer	0.9716	0.9701	0.9731	0.9731	0.9687	0.9731
Diabetes	0.7750	0.7684	0.7789	0.7645	0.7763	0.7750
Ionosphere	0.9380	0.9312	0.9412	0.9324	0.9265	0.9382
Heart	0.8185	0.8037	0.8148	0.8000	0.8111	0.8185
German	0.7340	0.7290	0.7310	0.7380	0.7290	0.7340
Ave. Diff	0	-0.0113	+0.0002	-0.0061	-0.0131	+0.0003

To justify the strategy to eliminate non-informative training samples presented in Section 5.2.3, the experiment is performed on the face data set that contains 2,429 face and 15,228 non-face samples. Linear SVM is used to do training and testing. Around 10% of samples are taken as testing samples. The initial training samples are selected according to the method in Section 5.2.1, and new samples are queried according the strategy in Section 5.2.2. In each incremental training step, 200 new samples are selected. Table 5.5 shows the comparison of the numbers of training samples for each step of keeping all and eliminating the non-informative training samples in the experiment. It can be seen that some non-informative training samples can be eliminated in each step, and this leads to smaller training subsets and a faster training process. Note that both methods achieved the same performance.

Table 5.5 The Numbers of Training Samples of Each Incremental Training Step by Keeping All the Previous Samples, the Newly Queried Samples, and by Eliminating the Non-informative Samples

Step	Num of Keeping All Samples	Num of Newly Queried Samples	Num of Eliminating Non-informative Samples
1	1579		1579
2	1779	200	458
3	1979	200	658
4	2179	200	856
5	2379	200	1020
6	2579	200	1119
7	2779	200	1261
8	2979	200	1432
9	3179	200	1587
10	3379	200	1761
11	3579	148	1961
12	3739		2109

CHAPTER 6

SUMMARY AND FUTURE RESEARCH

6.1 The Summary of the Contributions of his Dissertation

In this dissertation, several important issues in image processing and pattern classification areas are explored. The summary of the contributions is given as follows.

An efficient seeded region growing method for color image segmentation is developed. The seeds are automatically selected according to the properties of images. In region growing, strategies are applied to avoid order dependencies. By controlling the size of regions and the color difference of regions, the region merging method can adjust dynamically according to different images to obtain satisfactory segmentation results. Experimental results show that our algorithm can produce better results as compared to some existing algorithms.

An adaptive morphological edge-linking algorithm using the elliptical structuring element has been presented. Adaptive dilation is applied at each endpoint. The orientation and size of the elliptical structuring element are changed according to local properties. Gaps between broken edges can be filled gradually by extending along their slop directions. The experimental results show the success of the proposed algorithm.

An improved feature reduction method in the combinational input and feature space for Support Vector Machines (SVMs) is presented. In the input space, a subset of input features is selected by ranking their contributions to the decision function. In the feature space, features are ranked according to the weighted support vector in each dimension. Combining input and feature space feature reduction, a fast non-linear SVM is developed without a significant loss in performance.

An efficient face detection algorithm using two-layer support vector machines is designed. Integral images are used to normalize each scanned window quickly. Two-eye patterns are firstly detected using a linear SVM to eliminate most background. Two-layer 2nd-degree polynomial SVMs are developed for further face verification to achieve higher accuracy than a single SVM. The detection process is implemented directly on feature space that leads to a fast SVM.

An incremental training algorithm for support vector machines has been presented. A weight is assigned to each sample according to its distance to the current separating hyperplane and a confidence factor indicating the degree that the current separating hyperplane is the optimal one. The performance of the proposed method is compared with several other methods using several data sets, and the linear SVM, the 2nd-degree polynomial SVM, and the radial basis function SVM are tested. The experimental results showed the robustness of the proposed method compared with other methods.

6.2 Future Research

There are many actively ongoing research areas in image segmentation and pattern classification. To extend the research work that has been done in this dissertation, the following future research work is proposed.

The SRG algorithm proposed in this dissertation is for color image segmentation based on color similarity. In many applications, texture information is needed to obtain satisfactory segmentation results. Chen *et al.* [9] proposed an approach for image segmentation based on low-level features for color and texture. It is an interesting research topic to extend the SRG algorithm for texture-based segmentation. It is also attractive to improve the segmentation results by combing different features, such as color, texture, shape, etc.

Researchers have shown that combining the results of multiple classifiers can improve the classification performance [29, 31, 32, 38, 40]. It is interesting to investigate how to combine the results of different classifiers and to combine the results using different features to improve the performance. The research issues are how to do the combination and how to prove theoretically that the performance is improved by the combination.

Speed and accuracy are two main factors to judge an object detection algorithm. For face detection, the research in this dissertation is limited to frontal view face detection. Algorithms to detect multi-view faces [33, 75] and faces with in-plane rotation [30, 52, 73] have been proposed. Faces in different views need to be detected individually, and all the results are combined to form the final result. This will slow down the detection process and the false positive rate will be increased since the detection results from different views are combined to form the final result. Future research will explore how to design an efficient face detector for different views and investigate how to extract more efficient features for face detection.

REFERENCES

1. Adams, R. and Bischof, L., "Seeded region growing," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641-647, 1994.

2. An, J.-L., Wang, Z.-O., and Ma, Z.-P., "An incremental learning algorithm for support vector machine," *Proc. Second Int'l Conf. Machine Learning and Cybernetics*, Xi'an, China, pp.1153-1156, November, 2003.

3. Basak, J. and Chanda, B., "On edge and line linking with connectionist model," *IEEE Trans. System, Man, and Cybernetics*, vol. 24, no. 3, pp. 413-428, 1994.

4. Blake, C.L., and Merz, C.J., UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, Department of Information and Computer Science, Univ. of California, Irvine, CA,1998.

5. Burges, C. J.C., "A Tutorial on support vector machines for pattern recognition," *IEEE Trans. Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

6. Campbell, C., Cristianini, N., and Smola, A., "Query learning with large margin classifiers," *Proc. 17th Int'l Conf. Machine Learning*, Stanford University, CA, pp. 111-118, June, 2000.

7. Chai, D. and Bouzerdoum, A., "A Bayesian approach to skin color classification in YCbCr color space," *Proceedings: TENCON 2000*, vol. 2, pp. 421-424, 2000.

8. Chen, C.S., Wu, J.L., and Hung, Y.P., "Theoretical aspects of vertically invariant gray-level morphological operators and their application on adaptive signal and image filtering," *IEEE Trans. Signal Processing*, vol. 47, no.4, pp. 1049-1060, April 1999.

9. Chen, J, Pappas, T. N., Mojsilovic, A., and Rogowitz, E., "Adaptive perceptual color-texture image segmentation," *IEEE Trans. Image Processing*, vol. 14, no. 10, pp. 1524-1536, 2005.

10. Cheng, H. D., Chen, C. H., Chiu, H. H., and Xu, H., "Fuzzy homogeneity approach to multilevel thresholding," *IEEE Trans. Image Processing*, vol. 7, no. 7, pp. 1084-1088, 1998.

11. Cheng, H. D., Jiang, X. H., Sun, Y., and Wang, J., "Color image segmentation: advance and prospects," *Pattern Recognition*, vol. 34, pp. 2259-2281, 2001.

12. Cheng, H. D., Jiang, X. H., Wang, J., "Color image segmentation based on homogram thresholding and region merging," *Pattern Recognition*, vol. 35, pp. 373-393, 2002.

13. Chu, C. and Aggarwal, J. K., "The integration of image segmentation maps using region and edge information," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 2, pp. 1241-1252, 1993.

14. Cortes, C. and Vapnik, V., "Support-vector networks," *Machine Learning*, vol. 20, pp. 273-297, 1995.

15. Deng, Y., Manjunath, B.S., "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800-810, 2001.

16. Duda, R., Hart, P., and Stork, D., *Pattern Classification*, Wiley-Interscience Publication, New York, 2001.

17. Evgenious, T., Pontil, M., C. Papageorgiou, and T. Poggio, "Image representations and feature selection for multimedia database search," *IEEE Trans. Knowledge and Data Engineering*, vol. 15, no. 4, pp. 911-920, 2003.

18. Fan, J., Yau, D. K. Y., Elmagarmid, A. K. and Aref, W. G., "Automatic image segmentation by integrating color-edge extraction and seeded region growing," *IEEE Trans. Image Processing*, vol. 10, no. 10, pp. 1454-1466, 2001.

19. Garcia, C. and Tziritas, G., "Face detection using quantized skin color regions merging and wavelet packet analysis," *IEEE Trans. on Multimedia*, vol. 1, no. 3, pp.264-275, 1999.

20. Gonzalez, R. C. and Woods, R. E., *Digital Image Processing*, Prentice Hall, 2002.

21. Haddon, J. F. and Boyce, J. F., "Image segmentation by unifying region and boundary information," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 929-948, 1990.

22. Haralick, R.M., S.R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 532-550, Apr. 1987.

23. Heisele, B., Poggio, T., and Pontil, M., "Face detection in still gray images," A. I. Memo 1687, Center for Biological and Computational Learning, MIT, Cambridge, MA, 2000.

24. Heisele, B., Serre, T., Prentice, S., and Poggio, T., "Hierarchical classification and feature reduction for fast face detection with support vector machines," *Pattern Recognition*, vol. 36, no.9, pp. 2007-2017, 2003.

25. Hojjatoleslami, S. A. and Kittler, J., "Region growing: a new approach," *IEEE Trans. Image Processing*, vol. 7, no. 7, pp. 1079-1084, 1998.

26. Hsu, R.-L., Abdel-Mottaleb, M. and Jain, A. K., "Face detection in color image," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696-706, May 2002.

27. Ikonomakis, N., Plataniotis, K. N., Zervakis, M., and Venetsanopoulos, A. N., "Region growing and region merging image segmentation," *Proc. IEEE Conf. Digital Signal Processing*, vol. 1, pp. 299-302, 1997.

28. Jang, B.-K. and Chin, R. T., "Analysis of thinning algorithms using mathematical morphology," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 541-551, June 1990.

29. Kim, H.-C., Pang, S., Je, H.-M., Kim, D., "Constructing support vector machine ensemble," *Pattern recognition*, vol. 36, pp. 2757-2767, 2003.

30. Kim, H.I., Lee, S.H., Cho, N.I., "Rotation-invariant face detection using angular projections," *Electronics Letters*, vol. 40, no. 12, pp. 726-727, 2004.

31. Kittler J., Hatef M., Dulin, R. P.W., and Matas, J., "On combining classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226-239, 1998.

32. Kuncheva, L., "A theoretical study on six classifier fusion strategies," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 281-286, 2002.

33. Li, S.Z., and Zhang, Z.Q., "FloatBoost learning and statistical face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1112-1123, 2004.

34. Liu, S.-M., Lin, W.-C., and Liang, C.-C., "An iterative edge linking algorithm with noise removal capability," *9th International Conference on Pattern Recognition*, vol. 2, pp. 1120-1122, 1988.

35. Ma, Y., and Ding, X., "Face detection based on hierarchical support vector machines," *Proc. IEEE Conf. Computer Vision*, pp. 222-225, 2002.

36. Marteli, A., "An application of heuristic search methods to edge and contour detection," *Graphics and Image Processing*, vol. 19, no. 2, pp. 73-83, 1976.

37. Mehnert, A. and Jackway, P., "An improved seeded region growing algorithm," *Pattern Recognition Letters*, vol. 18, pp. 1065-1071, 1997.

38. Melnik, O., Vardi, Y., and Zhang, C.-H., "Mixed group ranks: preference and confidence in classifier combination," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 973-981, 2004.

39. Mitra, P., Murthy, C.A., and Pal, S.K., "A probabilistic active support vector learning algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 3, pp. 413-418, 2004.

40. Murua, A., "Upper bounds for error rates of linear combinations of classifiers," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 591-602, 2002.

41. Najman, L. and Schmitt, M., "Geodesic saliency of watershed contours and hierarchical segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1163-1173, 1996.

42. Nalwa, V. S. and Pauchon, E., "Edgel aggregation and edge description," *Computer Vision, Graphics, and Image Processing*, vol. 40, no. 1, pp. 79-94, 1987.

43. Nevatia, R., "Locating objects boundaries in textured environments," *IEEE Trans. Computers*, pp. 1170-1175, November 1976.

44. Nguyen, H. T., and Smeulders, A., "Active learning using pre-clustering," *Proc. 21st Int'l Conf. Machine Learning*, Alberta, Canada, July, 2004.

45. Osuna, E., Freund, R., and Girosi, F., "Training support vector machines: an application to face detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 130-136, 1997.

46. Otsu, N., "A threshold selection method from gray-level histogram," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62-66, 1979.

47. Pal, N. R. and Pal, S. K., "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, no. 9, pp. 1277-1294, 1993.

48. Papageorgiou, C. and Poggio, T., "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, pp. 15-33, 2000.

49. Parker, J,.R., *Algorithms for Image Processing and Computer Vision*, Wiley Computer Publishing, 1997.

50. Pavlidis, T. and Liow, Y. T., "Integrating region growing and edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 3, pp. 225-233, 1990.

51. Rowley, H., Baluja, S., and Kanade, T., "Neural network-based face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, 1998.

52. Rowley, H.A., Baluja, S., and Kanade, T., "Rotation invariant neural network-based face detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 38-44, 1998.

53. Russ, J.C., *The Image Processing Handbook*, CRC Press, 1992.

54. Saha, P. K. and Udupa, J. K., "Optimum image threshold via class uncertainty and region homogeneity," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 23, no. 7, pp. 689-706, 2001.

55. Salembier, P., "Morphological multiscale segmentation for image Coding," *Signal Processing*, vol. 38, no. 3, pp. 359-386, Sept. 1994.

56. Schneiderman, H. and Kanade, T., "Probabilistic modeling of local appearance and spatial relationships for object recognition," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 45-51, 1998.

57. Schohn, G., and Cohn, D., "Less is more: active learning with support vector machines," *Proc. 17th Int'l Conf. Machine Learning*, Stanford University, CA, pp. 839-846, June, 2000.

58. Serra, J., *Image Analysis and Mathematical Morphology*, New York: Academic, 1982.

59. Shih, F. Y. and Cheng, S., "Adaptive mathematical morphology for edge linking," *Information Sciences*, vol. 167, no. 1-4, pp. 9-21, Dec. 2004.

60. Shih, F. Y. and Mitchell, O. R., "Threshold decomposition of grayscale morphology into binary morphology," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 1, pp. 31-42, Jan. 1989.

61. Shih, F. Y., and Cheng, S., "Improved feature reduction in input and feature spaces," *Pattern Recognition*, vol. 38, no. 5, pp. 651-659, 2005.

62. Shih, F.Y. and Puttagunta, P., "Recursive soft morphological filters," *IEEE Trans. Image Processing*, vol. 4, no. 7, pp. 1027-1032, July 1995.

63. Shih, F.Y., Pei, S.-C., and Horng, J.-H., "Edge linking using adaptive morphology," technical report, National Taiwan University.

64. Sung, K.-K. and Poggio, T., "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39-51, Jan. 1998.

65. Syed, N. A., Liu, H., and Sung, K. K., "Incremental learning with support vector machines," *Proc. Int'l Joint Conf. Artificial Intelligence*, Stockholm, Sweden, July, 1999.

66. Tremeau, A. and Bolel, N., "A region growing and merging algorithm to color segmentation," *Pattern Recognition*, vol. 30, no. 7, pp. 1191-1203, 1997.

67. Vapnik, V., *Statistical Learning Theory*, Wiley, New York, 1998.

68. Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.

69. Vincent, L. and Soille, P., "Watershed in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583-598, June 1991.

70. Viola, P. and Jones, M., "Rapid object detection using a boosted cascade of simple features," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 511-518, 2001.

71. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., and Vapnik, V., "Feature selection for support vector machines," *Advances in Neural Information Processing System*, vol. 13, MIT Press, Cambridge, MA, 2000, pp. 668-674.

72. Wong, A. K. C. and Sahoo, P. K., "A Gray-level threshold selection method based on maximum entropy principle," *IEEE Trans. Systems, Man and Cybernetics*, vol. 19, no. 4, pp. 641-647, 1989.

73. Wu , B., Ai, H., Huang, C., and Lao, S., "Fast rotation invariant multi-view face detection based on real Adaboost," *Proc. IEEE Conf. Automatic Face and Gesture*, pp. 79-84,, 2004.

74. Wu, H., Chen, Q., and Yachida, M., "Face detection from color images using a fuzzy pattern matching method," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 557-563, 1999.

75. Xiao, R., Li, M-J., and Zhang, H-J., "Robust multipose face detection in images," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 31-41, 2004.

76. Yang, M.-H., Kriegman, D. J., and Ahuja, N., "Detecting faces in images: a survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34-58, 2002.

77. Yang, M.-H., Roth, D., and Ahuja, N., "A SNoW-based face detector," *Advances in Neural Information Processing Systems*, vol. 12, pp. 855-861, 2000.