國　立　交　通　大　學

電 機 與 控 制 工 程 研 究 所

碩 士 論 文

自動眼睛偵測及眼鏡反光消除
Automatic Eye Detection and Reflection Separation within Glasses

研　究　生：林 祐 聖

指 導 教 授：張 志 永

中 華 民 國 九十五 年 七 月

自動眼睛偵測及眼鏡反光消除

Automatic Eye Detection and Reflection Separation within Glasses

學　　生：林祐聖　　　Student : Yu-Sheng Lin

指導教授：張志永　　　Advisor : Jyh-Yeong Chang

國立交通大學

電機與控制工程學系

碩士論文

A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master in

Electrical and Control Engineering

July 2006

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 五 年 七 月

自動眼睛偵測及眼鏡反光消除

學生: 林祐聖　　　　　　　　　　指導教授: 張志永博士

國立交通大學電機與控制工程研究所

摘要

眼睛偵測經常用在許多研究上，如：人臉辨識、瞌睡偵測、人眼注視追蹤等等；但是，當佩戴眼鏡時，眼睛偵測常會因為眼鏡本身顏色及鏡片上反光的干擾，造成誤判，因此需要對相關的干擾作處理。在本篇論文中，我們提出一套演算法，使其能夠在一張人臉影像中偵測眼睛的位置，並且在該人臉有戴眼鏡時，去除眼鏡及眼鏡內反光的干擾。此系統包含三個模組：臉部位置偵測，眼睛偵測，以及當受測者有戴眼鏡時，去除眼鏡反光的方法。首先，我們使用通用膚色圖偵測臉部區域，以確保在周遭光源條件變化時，仍有足夠的適應性。接著，我們採用邊緣及角的偵測再搭配非等向性擴散轉換，來偵測眼睛區域以及分離出眼鏡反光區域內關於眼睛的資訊。反光分離的原理是根據當反光影像可以成功的分離成兩張分別為前景層圖及反射層圖時，則該兩張成功分離影像的邊緣和角的總和，是所有可能分解的解中最小值。結果顯示，該反光原理用於鏡片上的反光分離處理是有效的且可以得到良好的反光分離。

Automatic Eye Detection and Reflection Separation within Glasses

STUDENT: YU-SHENG LIN ADVISOR: JYH-YEONG CHANG

Institute of Electrical and Control Engineering

National Chiao-Tung University

Abstract

Eye detection has been applied to many applications, for instance, human or faces recognition, eye gaze detection, drowsiness detection, and so on. However, eye detection often misdiagnoses for the interference caused by glasses when one wears spectacles. This thesis addresses an algorithm to automatically detect the eye location from a given face image and separate reflections within glasses while one has worn glasses. Our system consists of three modules: face segmentation, optic-area detection, and the separating of glasses reflections while one has worn glasses. First, we use the universal skin-color map to detect the face regions, which can ensure sufficient adaptability to ambient lighting conditions. Then, we proposed a novel method to detect the eye region and separate the reflection within glasses based on edge detection, corner detection, and anisotropic diffusion transform. The principle of separating reflection is based on that the correct decomposition of the reflection image whose summation of corners and edges is the smallest among all possible decompositions. The simulation and results demonstrate that the principle of separating reflection can be applied to the reflection within glasses effectively and result in good reflection separation.

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to my advisor, Dr. Jyh-Yeong Chang. Without his patient guidance and inspiration during the two years, it is impossible for me to overcome the obstacles and complete the thesis. In addition, I am thankful to all my lab members for their discussion and suggestion.

Finally, I would like to express my deepest gratitude to my parents. Without their fully support and encouragement, I could not go through these two years.

Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1 Motivation of This Research

Eye detection has been applied to many applications, for instance, human or face recognition, eye gaze detection, drowsiness detection, and so on. But eye detection often misdiagnoses for the interference caused by sunglasses or glass reflection when one wears glasses. There are a lot of methods proposed to solve the eye detection, yet the cases of people who wearing glasses seem to be omitted in these developments. There are several good reasons why people avoided dealing with eyeglasses, such as the following:

- The appearance of glasses frames is so diverse due to various material properties, such as metal and plastic.
- The reflectance distribution on glasses emerges randomly; for this reason, sometimes the reflection may be overlapped on the eye or eyebrows area.
- The reflectance brightness on glasses appears arbitrarily; consequently, sometimes the eye detection likely is mistaken for the erroneous eye location.
- Many faces are always well separated with the background whereas the glasses are stuck to the face and mixed by eyes or eyebrows.

Therefore, dealing with the wearing glasses cases in all of the application is important and instant.

To this end, we particularly aim at the drowsiness detection system and the camouflage of face recognition system dealing with eyeglasses interference problem. For the drowsiness detection system, we used image processing technique to measure the eye closure and calculate the PERCLOS and blinking rate. Then we use these two evidences above to estimate whether the consciousness state of a subject is drowsy or not [1]. The approach above offers the advantage of providing high detection accuracy and does not use any intrusive sensor the driver. When a driver wears eyeglasses, however, the system cannot detect the eye position accurately because the reflections within eyeglasses may be overlapped with eyes. Since there are almost one half people in Taiwan wearing glasses, the system must be able to accommodate the use of different kinds of glasses in individual driver to make the drowsiness detection system more practical. Moreover, wearing glasses is one of the most common camouflages to a human identification problem to face recognition system. The automatic glasses reflections separation method we will propose can helpfully remove this stumbling block.

1.2 Face Detection

There are many methods or algorithms have been proposed for face detection in the recent years. According to Hjelmas and Low [2], the major approaches are listed chronologically in Table I. These approaches utilize techniques such as principal component analysis, neural networks, machine learning, information theory, geometrical modeling, (deformable) template matching, Hough transform, motion extraction, and color analysis. Among these methods above, color analysis is a straightforward method and useful cue for face detection.

In this thesis, the method we adopted, is proposed by Chai and Ngan [3], which used color as a feature for identifying a human face in an image. This is feasible because human faces have a special color distribution that differs significantly (although not entirely) from those of the background objects. Hence this approach requires a color map that models the skin-color distribution characteristics. The skin-color reference map can be derived in two ways on account of the fact not all faces have identical color features. One approach is to predefine or manually obtain the map such that it suits only an individual color feature. In another approach, the skin-color map can be designed by adopting histogram technique on a given set of training data and subsequently used as a reference for any human face. Although the first approach is more accurate and has better segmentation results, the second approach is more practical and attempts to cater to all personal color features in an automatic manner. The remaining question is which color space should we use.

TABLE I
MAJOR FACE DETETION APPROACHES

Authors	Approach	Features Used
Féraud *et al.* [6]	Neural Networks	Motion; Color; Texture
Maio *et al.* [7]	Facial templates; Hough Transform	Texture; Directional images
Garcia *et al.* [8]	Statistical wavelet analysis	Color wavelet coefficients
Wu *et al.* [4]	Fuzzy color models; Template matching	Color
Sung *et al.* [9]	Learning	Texture
Yang *et al.* [5]	Multi-scale segmentation; color model	Skin Color; intensity
Yow *et al.* [10]	Feature; Belief networks	Geometrical facial features

Although RGB color space is suitable for display, it is not good for color scene segmentation and analysis because of the high correlation among the R, G, and B components. By high correlation, we mean that if the intensity changes, all the three components will change accordingly. Also, the measurement of a color in RGB space does not represent color differences in a uniform scale, hence it is impossible to evaluate the similarity of two colors from their distance in RGB space. Here, we use the YC_rC_b color space to be the color model. The details will be given in Chapter 2.

1.3 Eye Location and Glasses Existence Detection

One of the most essential pre-requisites in building an automated system for face recognition is eye detection. There are many algorithms being proposed to do this work. Rosenfeld *et al*. [11] used filers based on Gabor wavelets to detect eyes in gray level images. Feng *et al*. [12] employed multi cues for eye detection on gray images using variance projection function. Kumar *et al*. [13] used color cues and projection function to detect the eye position. Liu *et al*. [14] first used edge information to roughly segment the eye region and then used genetic algorithm to search the eye. Kawato *et al*. [15] proposed a new idea that uses a circle-frequency filter to search “between-eyes” rather than to detect the eye directly.

These various schemes that have been proposed for eye detection can be broadly categorized into two approaches. The first category assumes that rough eye regions have been located or there are some restrictions on the face image such that eye windows can be easily located. The detection of eyes in the face is then operated on the restricted windows. However, in practical, it is generally difficult to locate the eye windows in real world situations. In the second approach, a face detection algorithm is used to extract the approximate face region and the detection of eyes is carried out in the identified face area. But there is a problem that the eye detection accuracy depends on the robustness of the face detection algorithm. Moreover, unless the orientation of the face is known, it is very difficult to extract the eye pair.

In this thesis, we apply the corner operator, edge operator and anisotropic diffusion to the face segment that is extracted at the face detection stage. Then we calculate the response particularly to find out the eye region. We will present the details in Chapter 3.2.

1.4 Glasses Reflection Separation System

When a subject wears glasses or sunglasses the performance of estimating the state of eye will descend. Because the reflection will arise on the lens of glasses arbitrarily; hence, we need to diminish the side effect from the reflection. There are many algorithms being proposed to do this task. Levin and Weiss [16] use derivative filters, linear programming and some user assistance. [17, 18] use the feature which the reflection and non-reflection images have different motions to separate the two image. In this thesis, we adopt the method proposed by Levin *et al.* [19] to separate the reflection. We will show the details in Chapter 3.3.

1.5 Flowchart of Eye Detection and Glasses Reflections Separation System

Fig 1.1 exhibits the flowchart of eye detection and glasses reflections system. The system starts with a new image frame. After getting a new color frame, we search out the face region by extracting a skin-color area based on the skin-color reference map in [3]. Then we detected the appearance of glasses because we must preprocess the reflection within glasses if one wears glasses. Subsequently, we locate the area of eye for eye detection. Finally, we use the eye detection to determine that the state of eye is open or closed.

1.6 Thesis Outline

This thesis is organized as follows. Chapter 2 introduces the face location detection with some illustrative examples. Chapter 3 shows how we use edge detection, corner detection, and anisotropic diffusion transform to location eye area, estimate the presence of glasses, and then to separate the reflection within glasses. The technique that we use patch database and evaluation function to choose the best decomposition will also be shown. Several simulation examples and their results of each topic of chapters are provided in Chapter 4. We conclude this thesis with a discussion in Chapter 5.

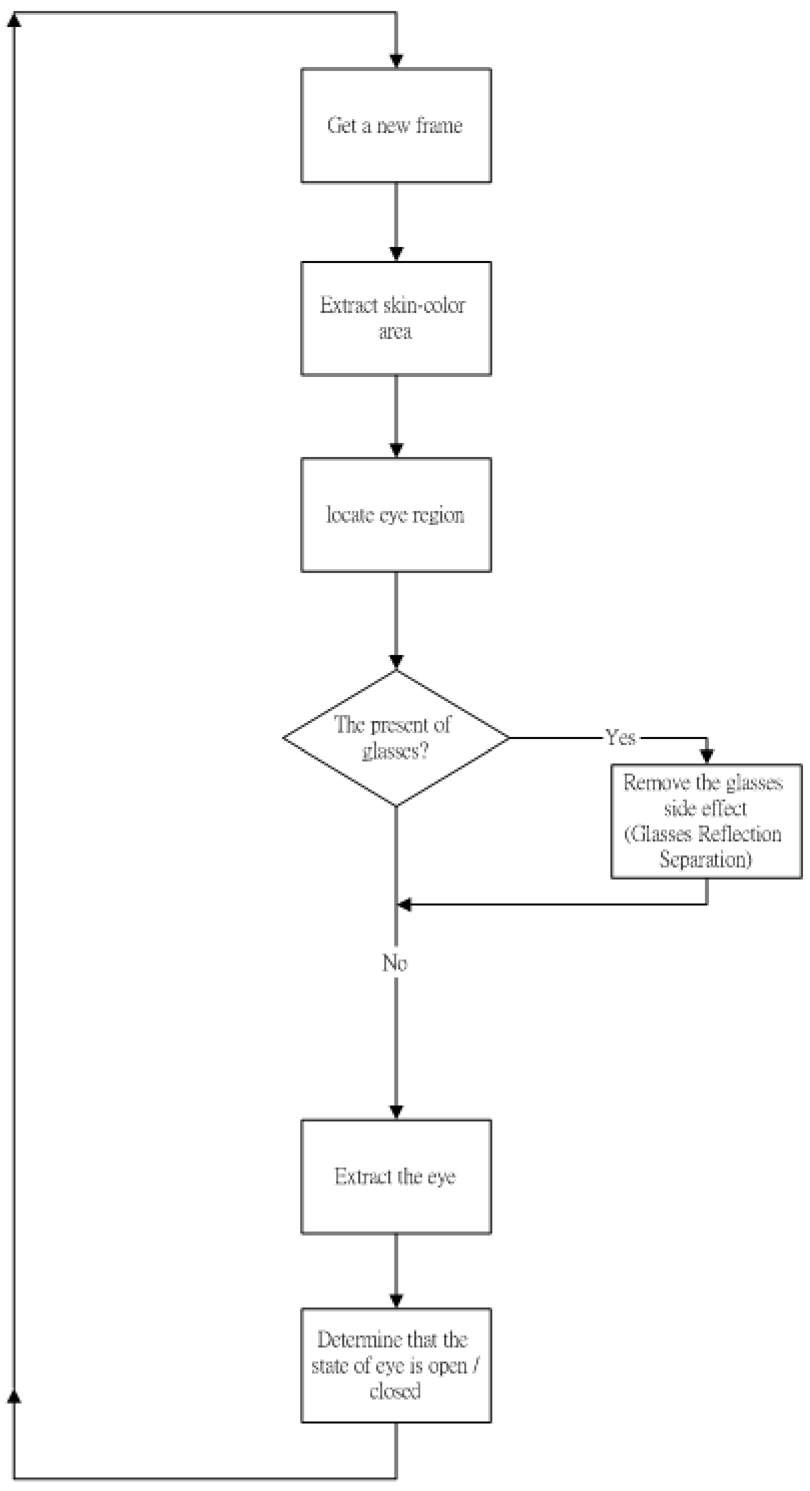

Get a new frame
Extract skin-color area
locate eye region
The present of glasses?
Yes
Remove the glasses side effect (Glasses Reflection Separation)
No
Extract the eye
Determine that the state of eye is open / closed

Fig 1.1. Flowchart of eye detection and glasses removal system.

Chapter 2. Face Segmentation

2.1 Introduction

In this section, we will show how to segment face area. Here we adopt a method proposed by Chai *et al*. [3]. We just used four stages of this method except the luminance regularization stage. The luminance regularity property is only appropriate for a simple background, but we will identify a person’s face in an image without any constraints. To this end, we skip the luminance stage, and use more constraints on geometry to distinguish the face region and background instead.

2.2 Face Segmentation Algorithm

The algorithm in [3] is an unsupervised segmentation algorithm, and hence no manual adjustment of any design parameter is needed in order to suit any particular input image. The only principal assumption is that the person’s face must be present in the given image, since we are locating and not detecting whether there is a face. The revised algorithm we used is consists of four stages, as depicted in Fig. 2.1.

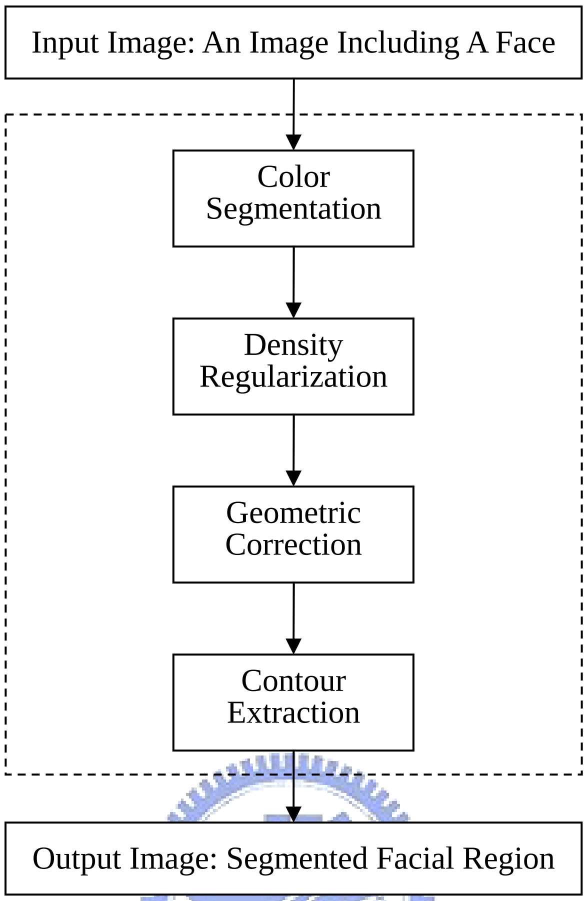


Fig. 2.1. Outline of face-segmentation algorithm.

A. Color Segmentation

The first stage of the algorithm is to classify the pixels of the input image to skin region and non-skin region. To do this, we reference a skin-color reference map in YC_rC_b color space.It has been proved that a skin-color region can be identified by the presence of a certain set of chrominance values (i.e., C_r and C_b) narrowly and consistently distributed in the YC_rC_b color space. We utilize R_{Cr} and R_{Cb} to represent the respective ranges of C_r and C_b values that correspond to skin color, which subsequently define our skin-color reference map. The ranges that the paper uses to be the most suitable for all the input images that they have tested are

$R_{C_r} = [133,\ \ 173]$, and $R_{C_b} = [77,\ \ 127]$.

The size of image we use is 640×480. In the cause of reducing the computing time, we downsample the image to become 320x240 and recover in the last stage. Therefore, for an image of *M*×*N* pixels and we downsample it to *M*/2×*N*/2. With the skin-color reference map, we got the color segmentation result O_A as

$$O_A(x,\ \ y) = \begin{cases} 1, & \text{if } [C_r(x,\ \ y) \in R_{C_r}] \bigcap [C_b(x,\ \ y) \in R_{C_b}] \\ 0, & \text{otehrwise} \end{cases} \tag{2.1}$$

where x = 0, … , *M*/2–1 and y = 0, … , *N*/2–1 and *M*, *N* are the height and width of the picture respectively. An example to illustrate the classification of the original image Fig. 2.2 is shown in Fig. 2.3.

Nevertheless, the result of color segmentation is the detection of pixels in a facial area and may also include other areas where the chrominance values coincide with those of the skin color (as is the case in Fig. 2.3). Hence the successive operating stages of the algorithm can be exploited to eliminate these misdiagnosed areas.

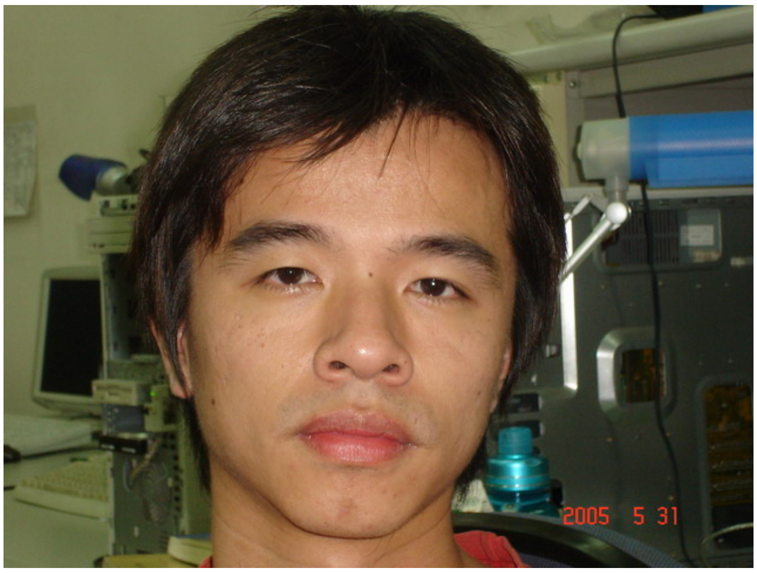


Fig. 2.2. Original image.

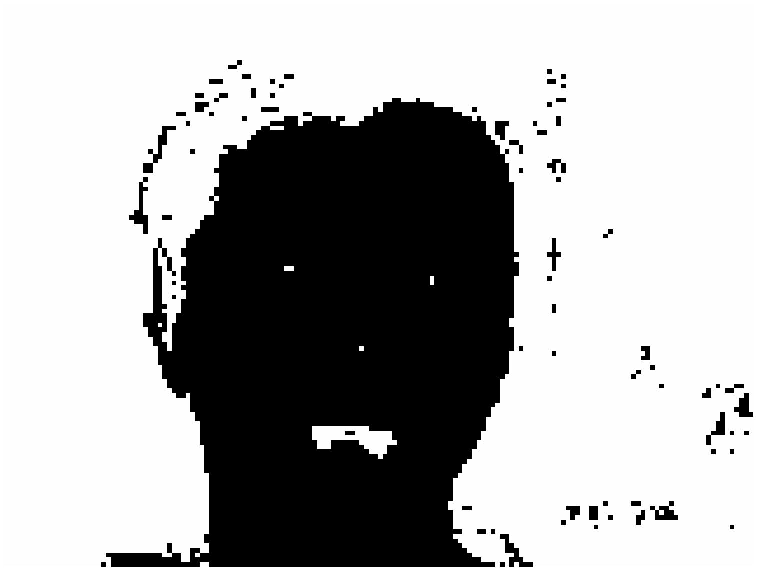


Fig. 2.3. Image after filtered by skin-color map in stage A.

B. Density Regularization

This stage considers the bitmap produced by the previous stages to contain the facial region that is corrupted by noise. The noise may appear as small holes on the facial region due to undetected facial features such as eyes, mouth, even glasses, or it may also appear as objects with skin-color appearance in the background scene.

Therefore, this stage pre-performs simple morphological operations such as dilation to fill in any small hole in the facial area and erosion to remove any small object in the background area. Nevertheless, the intention is not to remove the noise entirely but to reduce its amount and size.

To distinguish between facial and non-facial region more complete, we first need to identify regions of the bitmap that have higher probability of being the facial region. According to their observation in [3], it shows that the facial color is very uniform, and therefore the skin-color pixels belonging to the facial region will appear in a single large cluster, while the skin-color pixels belonging to the background may appear as many large clusters or small isolated objects. Thus, we study the density distribution of the skin-color pixels detected in stage A. A density map is calculus as follows.

$$D(x,\ y) = \sum_{i=0}^{3}\sum_{j=0}^{3} O_A(4x+i,\ 4y+j) \tag{2.2}$$

It first partitions the output bitmap of stage A $O_A(x,\ y)$ into non-overlapping groups of 4×4 pixels, then counts the number of skin-color pixels within each group and assigns this value to the corresponding point of the density map.

According to the density value, we classify each point into three types, namely, zero (D = 0), intermediate (0 < D < 16), and full (D = 16). A group of points with zero density value will represent a non-facial region, while a group of full density points will signify a cluster of skin-color pixels and a high probability of belonging to a facial region. Any point of intermediate density value will indicate the presence of noise. The density map of an example with three density classifications is depicted in Fig. 2.4. The point of zero density is shown in white, intermediate density in green, and full density in black.

Once the density map is derived, we can then begin the process that we termed as density regularization. This involves the following three steps.

1) Discard all points at the edge of the density map, i.e., set $D(0, y) = D(M/8–1, y) = D(x, 0) = D(x, N/8–1)$ for all $x = 0, …, M/8–1$ and $y = 0, …, N/8–1$.
2) Erode any full-density point (i.e., set to zero) if it is surrounded by less than five other full-density points in its local 3×3 neighborhood.
3) Dilate any point of either zero or intermediate density (i.e., set to 16) if there are more than two full-density points in its local 3×3 neighborhood.

After this process, the density map is converted to the output bitmap of stage B as

$$O_B(x,\ \ y) = \begin{cases} 1, & \text{if } D(x,\ \ y) = 16 \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

for all $x = 0, …, M/8–1$ and $y = 0, …, N/8–1$.

The result of the previous example is displayed in Fig. 2.5.

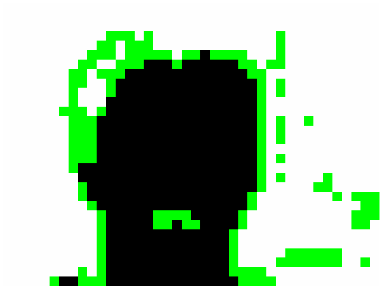


Fig. 2.4. Density map after classified to three classes.

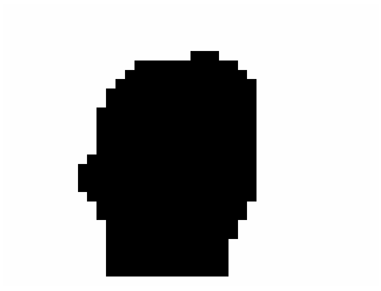


Fig. 2.5. Image produced by stage B.

C. Geometric Correction

In this stage, we first performed two simple procedures that are similar to that initially introduced in stage B to ensure that noise appearing on the facial region is filled in and that isolated noise objects on the background are removed. The two procedures are shown as followings, a pixel in $O_B(x, y)$ with the values of one will remain as a detected pixel if there are more than three other pixels, in its local 3×3 neighborhood, with the same value. At the same time, a pixel in $O_B(x, y)$ with a value of zero will be reconverted to a value of one (i.e., as a potential pixel of the facial region) if it is surrounded by more than five pixels, in its local 3×3 neighborhood, with a value of one.

We then commence the horizontal scanning process on the "filtered" bitmap. We search for any short continuous run of pixels that are assigned with the value of one. Any group of less than four horizontally connected pixels with the value of one will be eliminated and assigned to zero. A similar process is then performed in the vertical direction. As a result the output bitmap of this stage should contain the facial region with minimal or no noise, as demonstrated in Fig. 2.6.

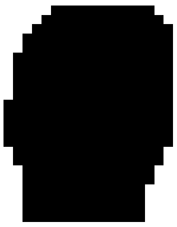


Fig. 2.6. Image produced by stage C.

D. Contour Extraction

In this final stage, we convert the $M/8 \times N/8$ output bitmap of stage C back to the dimension of $M/2 \times N/2$. To achieve the increase in spatial resolution, we utilize the edge information that is already made available by the color segmentation in stage A. Therefore, all the boundary points in the previous bitmap will be mapped into the corresponding group of 4×4 pixels with the value of each pixel as defined in the output bitmap of stage A. The representative output bitmap of this final stage of the algorithm is shown in Fig. 2.7.

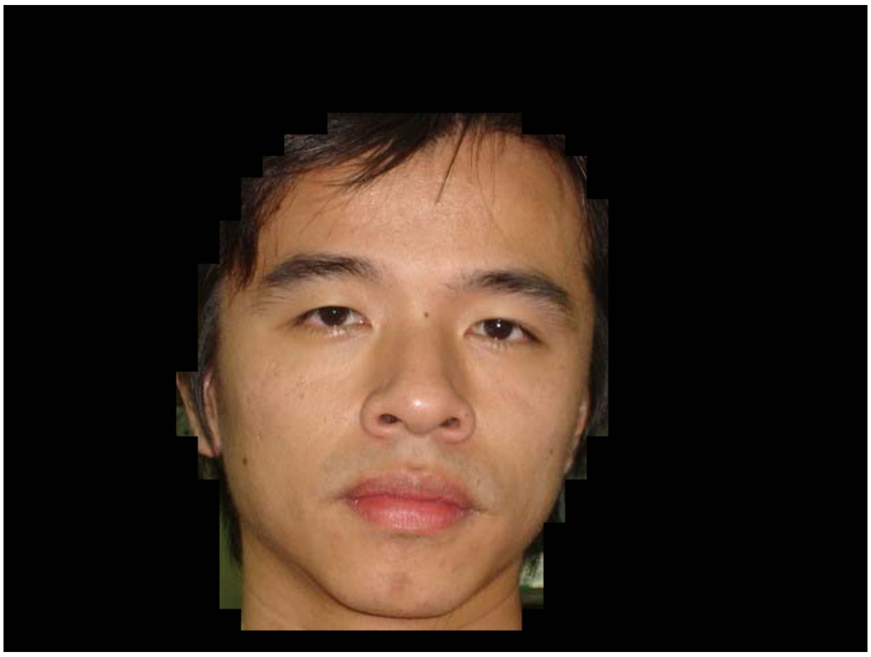


Fig. 2.7. Image produced by stage D.

Chapter 3. Eye Detection, Glasses Existence Detection and Reflection Separation of Glasses

3.1 Introduction

In this section, we first show three measures that were employed to eye location, glasses existence detection, and reflection separation. Then we describe how to locate the eye area, and then present the details of separating reflection. Our procedure is slightly different from the order of the flowchart mentioned in Fig 1.1. The task of glasses existence detection and eye position detection is done at the same stage, because the position of glasses always overlapped with eye or eyebrow.

3.1.1 Edge Detection

Image edges have already been defined as local variations of image intensity. Therefore, local image differentiation techniques [20]-[22] can produce edge detector operators. The image gradient $\nabla f(x,y)$

$$\nabla f(x,y) = \left[\frac{\partial f}{\partial x} \; \frac{\partial f}{\partial y} \right]^T = \left[f_x \; f_y \right]^T, \tag{3.1}$$

provides useful information about local intensity variations. Its magnitude,

$$\left| \nabla f(x,y) \right| = \sqrt{f_x^2(x,y) + f_y^2(x,y)}, \tag{3.2}$$

can be used as an edge detector. Alternatively, the sum of the absolute values of partial derivatives f_x, f_y can be employed by

$$\left|\nabla f(x,y)\right| = \left|f_x(x,y)\right| + \left|f_y(x,y)\right|, \tag{3.3}$$

for computational simplicity. Local edge direction can be described by the direction angle:

$$\phi(x,y) = \arctan(f_y/f_x). \tag{3.4}$$

Gradient estimates can be obtained by using gradient operators of the form:

$$\hat{f}_x = W_1^T X, \tag{3.5}$$

$$\hat{f}_y = W_2^T X, \tag{3.6}$$

where X is the vector containing image pixels in a local image neighborhood. Weight vectors W_1, W_2 are described by gradient masks. Such masks are shown in Fig. 3.1 for the Sobel edge detectors. Eqs. (3.5) and (3.6) are essentially two-dimensional linear convolutions with the 3x3 kernels shown in Fig. 3.1. They can be easily implemented in the spatial domain.

Edge templates are masks that can be used to detect edges along different directions. Such Modified Sobel edge detector masks of size 3x3 are shown in Fig. 3.2. They can detect edges at four directions (0, 45, 90, and 135 degrees).The resultant edge images processed through Sobel and Modified Sobel operator are shown in Figs. 3.3(b) and 3.3(c) respectively, over a test image “Baboon.” The Modified Sobel operator has better performance than the Sobel operator because it can produce more minute and subtle edge, such as slanted edges beside the noise.

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

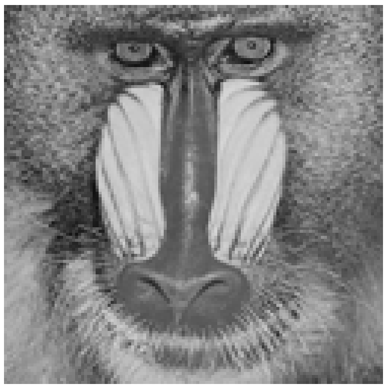
Fig. 3.1. Sobel edge detector masks for f_x and f_y, respectively.

-1	0	1
-2	0	2
-1	0	1

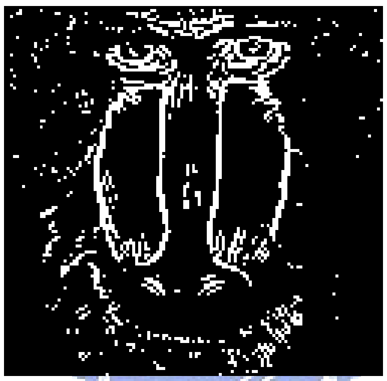
1	2	1
0	0	0
-1	-2	-1

2	0	2
0	0	0
-2	0	-2

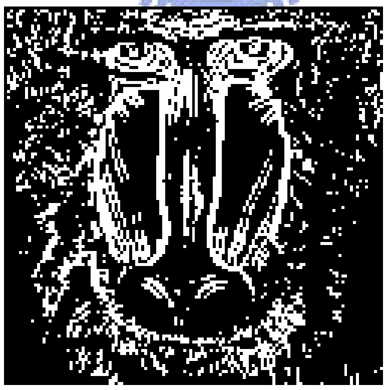
Fig. 3.2. Modified Sobel edge detector masks.



(a)



(b)



(c)

Fig. 3.3. The example of image applying edge detector. (a) Original image (Baboon); (b) Sobel edge detector output; and (c) Modified Sobel edge detector output.

3.1.2 Corner Detection

In this thesis, we utilize the Harris-like operator as a corner detection operator; therefore, we first describe the concept of Harris corner detector. In an arbitrary image, we can classify it to three kinds of regions with respect to Harris corner detector. Here, we show it as follows.

- The flat region: The intensity at this region changes scarcely at all directions.
- The edge region: The intensity at this region changes scarcely along the direction of edge, but it changes severely along the orthogonal direction of edge.
- The corner region: The intensity at this region changes significantly at all directions.

According to Harris [23], the Harris corner detector is based on the local auto-correlation function of a signal; where the local auto-correlation function measures the local changes of the signal with patches shifted by a small amount in different directions. Given a shift $(\Delta x, \Delta y)$ and a point (x, y), the auto-correlation function is defined as,

$$c(x,y) = \sum W(x,y)\left[I(x_i,y_i) - I(x_i+\Delta x, y_i+\Delta y)\right]^2 \tag{3.7}$$

where $I(\cdot,\cdot)$ denotes the image function and (x_i, y_i) are the points in window W (Gaussian) centered on (x, y). The shifted image is approximated by a Taylor expansion truncated to the first order terms,

$$I(x_i+\Delta x, y_i+\Delta y) \approx I(x_i, y_i) + \left[I_x(x_i, y_i)\, I_y(x_i, y_i)\right]\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \tag{3.8}$$

where $I_x(\cdot,\cdot)$ and $I_y(\cdot,\cdot)$ denote the partial derivatives in x and y, respectively.

Substituting approximation Eq. (3.8) into Eq. (3.7) yields,

$$\begin{aligned}
c(x,y) &= \sum W(x,y)\left[I(x_i,y_i) - I(x_i+\Delta x, y_i+\Delta y)\right]^2 \\
&= \sum W(x,y)\left(I(x_i,y_i) - I(x_i,y_i) - \left[I_x(x_i,y_i)\, I_y(x_i,y_i)\right]\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}\right)^2 \\
&= \sum W(x,y)\left(\left[I_x(x_i,y_i)\, I_y(x_i,y_i)\right]\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}\right)^2 \\
&= [\Delta x\ \Delta y]\ \sum W(x,y)\begin{bmatrix}\left(\mathrm{I}_x(x_i,y_i)\right)^2 & \mathrm{I}_x(x_i,y_i)\mathrm{I}_y(x_i,y_i)\\ \mathrm{I}_x(x_i,y_i)\mathrm{I}_y(x_i,y_i) & \left(\mathrm{I}_y(x_i,y_i)\right)^2\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \\
&= [\Delta x\ \Delta y]\,C(x,y)\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}
\end{aligned} \tag{3.9}$$

where matrix $C(x,y)$ captures the intensity structure of the local neighborhood. Let λ_1, λ_2 be the eigenvalues of matrix $C(x,y)$. The eigenvalues form a rotationally invariant description. There are three cases to be considered:

1. If both λ_1, λ_2 are small, then it indicates the windowed image region is of approximately constant intensity.
2. If one eigenvalue is high and the other is low, then it denotes local shifts along the edge direction cause little change and significant change in the orthogonal direction; this means an edge.
3. If both eigenvalues are high, then it indicates local shifts in any direction will result in a significant increase; this means a corner.

Harris [23] defined a measure of corner strength:

$$H(x,y) = \det C - \alpha (\text{trace } C)^2 , \tag{3.10}$$

and a corner is detected when

$$H(x,y) > H_{thr} \tag{3.11}$$

where H_{thr} is a parameter, a threshold on corner strength. In Harris corner detector, α plays a role to tune the sensitivity of corner. When α is larger, then $H(x,y)$ is smaller and less sensitive for corner detection; otherwise not. Fig. 3.4 and Fig. 3.5 shows Harris corner detector applied to some image with corners.

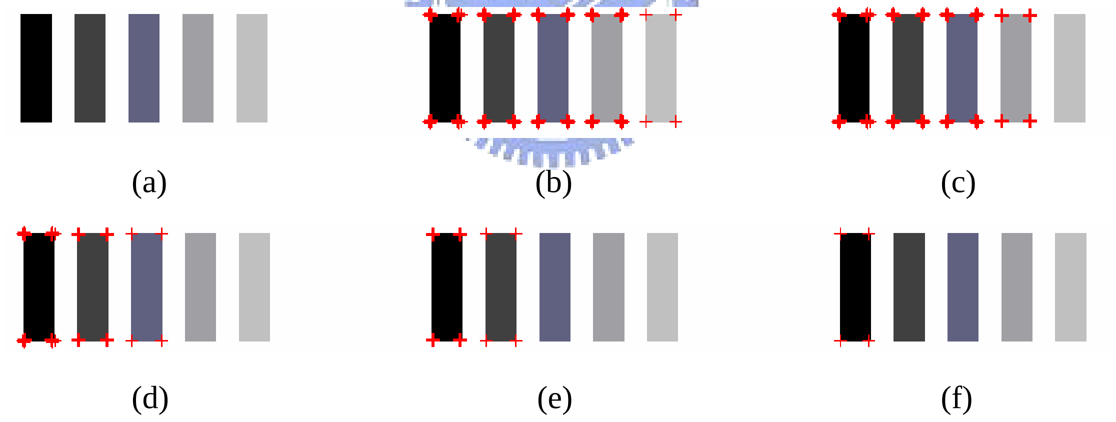
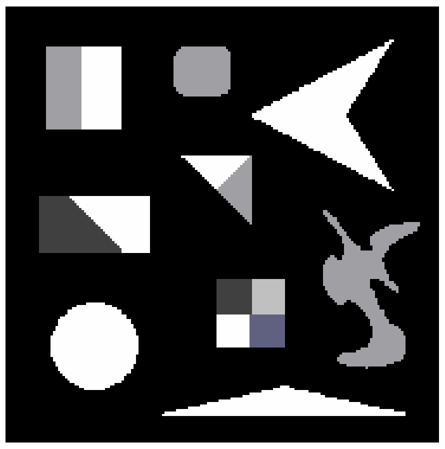
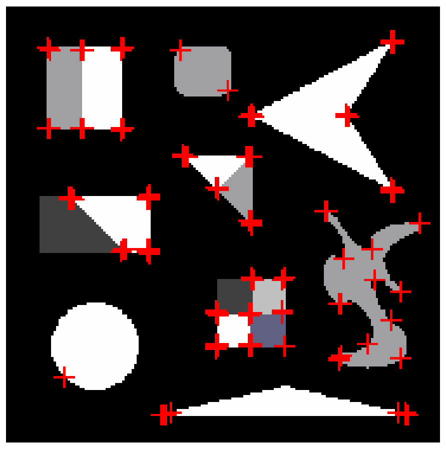


Fig. 3.4 The example 1 of corner detection. (a) Original image. (b) α =0.04 and H_{thr} =0.005. (c) α =0.04 and H_{thr} =0.01. (d) α =0.04 and H_{thr} =0.15. (e) α =0.04 and H_{thr} =0.6. (f) α =0.04 and H_{thr} =0.9.

(a) (b)

Fig. 3.5 The example 2 of corner detection. (a) Original image. (b) α =0.05 and H_{thr} =0.05.

Although the Harris corner detector provide good repeatability under varying rotation and illumination, it will be the best that remove noise before apply Harris corner detector to prevent from "spurious" corner. Therefore, the next section will introduce how anisotropic diffusion to remove noise.

3.1.3 Anisotropic Diffusion

In [24], Black mentioned diffusion algorithms could remove noise from an image by modifying the image via a partial differential equation (PDE). For example, consider applying the isotropic diffusion equation (the heat equation) given by $\partial I(x,y,t)/\partial t = \operatorname{div}(\nabla I)$, using the original (degraded/noisy) image $I(x,y,0)$ as the initial condition, where $I(x,y,0):\mathbb{R}^2 \to \mathbb{R}^+$ is an image in the continuous domain, (x,y) specifies spatial position, t is an artificial time parameter, and where ∇I is the image gradient. Modifying the image according to this isotropic diffusion equation is equivalent to filtering the image with Gaussian filter; however it result in blurring the edge.

Perona and Malik [25] proposed the anisotropic diffusion equation as follows:

$$\frac{\partial I(x,y,t)}{\partial t} = \operatorname{div}\left[g\left(\|\nabla I\|\right)\nabla I \right] \tag{3.12}$$

where $\|\nabla I\|$ is the gradient magnitude, and $g\left(\|\nabla I\|\right)$ is an "edge-stopping" function. This function is chosen to satisfy $g(x) \to 0$ when $x \to \infty$ so that the diffusion is "stopped" across edges as Fig. 3.6. The "edge-stopping" function adopted in [25] are

$$g(\nabla I) = e^{\left(-\left(\|\nabla I\|/K\right)^2\right)}, \tag{3.13}$$

and

$$g(\nabla I) = \frac{1}{1 + \left(\frac{\|\nabla I\|}{K}\right)^2}. \tag{3.14}$$

The constant K was fixed either by hand at some fixed value, or using the "noise estimator" described by Canny [22].

Perona and Malik discretized their anisotropic diffusion equation as follows:

$$I_s^{t+1} = I_s^t + \frac{\lambda}{|\eta_s|} \sum_{p \in \eta_s} g\left(\nabla I_{s,p}\right) \nabla I_{s,p} \tag{3.15}$$

where I_s^t is a discretely sampled image, *s* denotes the pixel position in a discrete, two-dimensional (2-D) grid, and *t* now denotes discrete time steps (iterations). The constant λ is a scalar that determines the rate of diffusion, η_s represents the spatial neighborhood of pixel *s*, and $|\eta_s|$ is the number of neighbors. Perona and Malik linearly approximated the image gradient (magnitude) in a particular direction as

$$\nabla \mathrm{I}_{s,\rho} = \mathrm{I}_\rho - \mathrm{I}_s^t, \qquad \rho \in \eta_s . \tag{3.16}$$

We show the local neighborhood of pixels at a boundary in Fig. 3.7. Fig. 3.8 shows the example of the noise image and its result image after anisotropic diffusion processing.

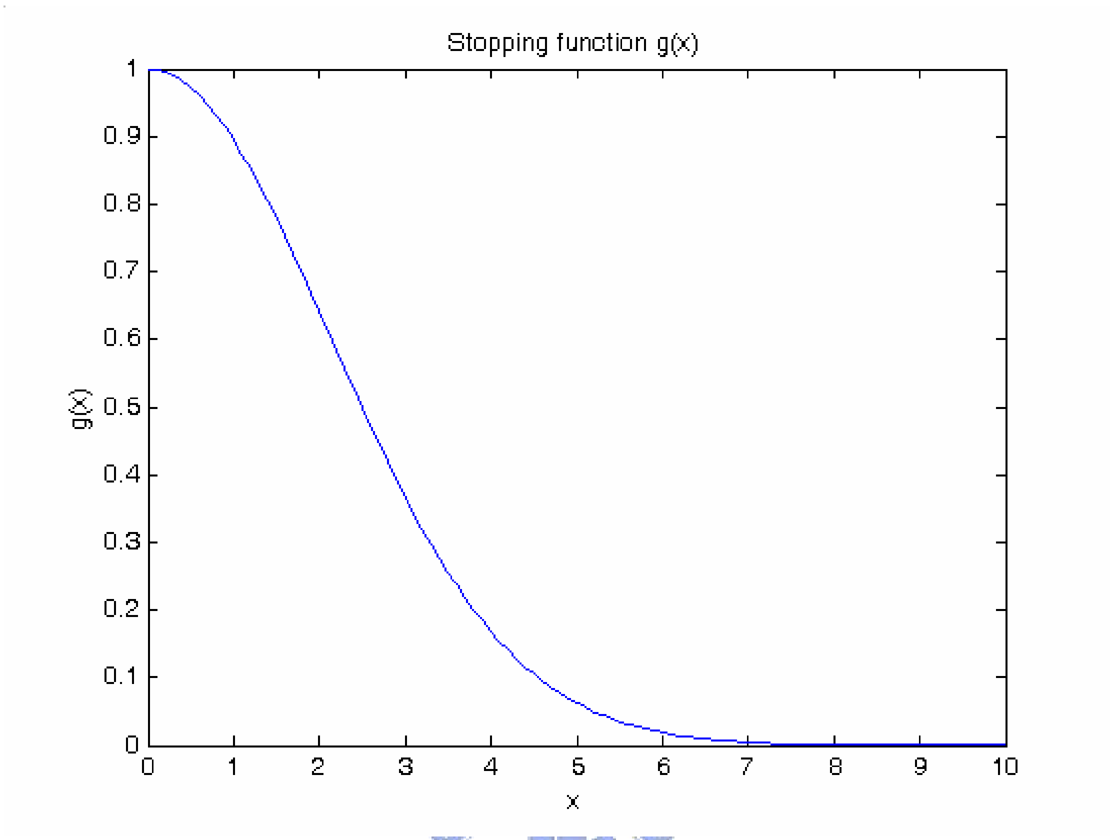


Fig. 3.6.　The stepping function g(.).

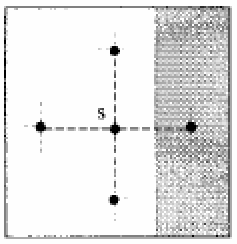
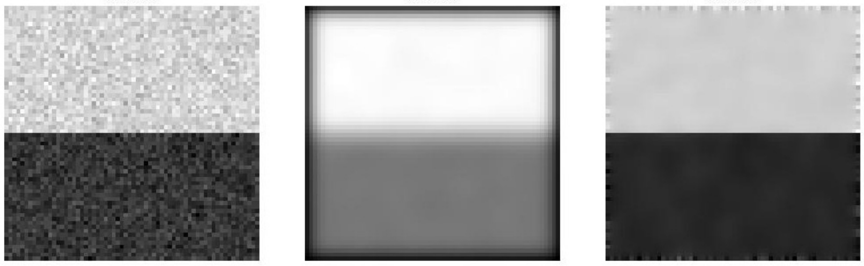


Fig. 3.7.　Local neighborhood of pixels at a boundary (intensity discontinuity).



(a) (b) (c)

Fig. 3.8. The example about anisotropic diffusion processing. (a) The input image(with noise). (b) The processed image after average mask 10 times. (c) The processed image after anisotropic diffusion 10 times.

3.2 Glasses Existence Detection and Eye Location

In this section, we discuss the glasses existence and eye location detection together, because the position of glasses always overlapped with eye or eyebrow. At first, we classify the face condition into three types:

- Face without glasses.
- Face with glasses (non-sunglasses).
- Face with sunglasses.

The typical results of face segment derived from above three types are shown in Figs. 3.9, 3.10, and 3.11, respectively.

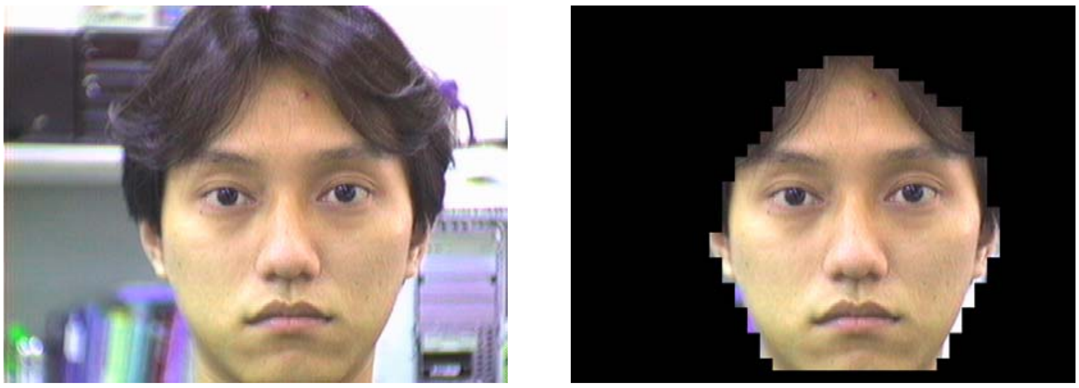


Fig. 3.9. The example of face without glasses. (a) The input face image without glasses. (b) The face segment derived from (a).

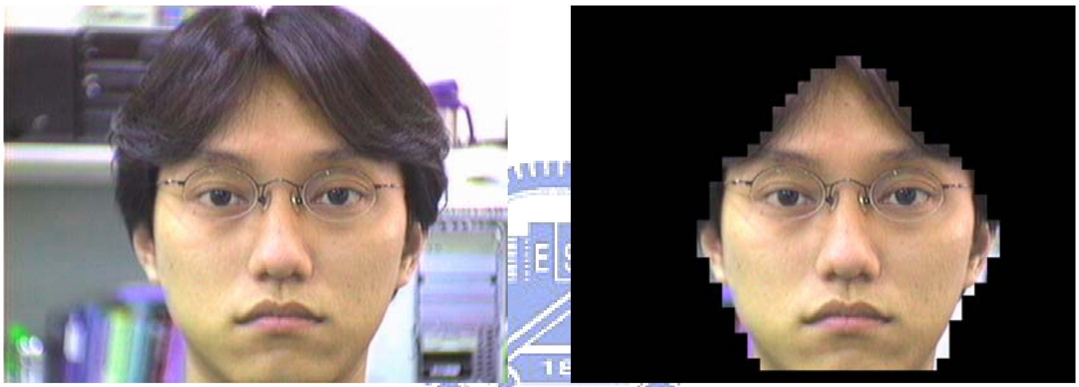


Fig. 3.10. The example of face with glasses. (a) The input face image with glasses. (b) The face segment derived from (a).

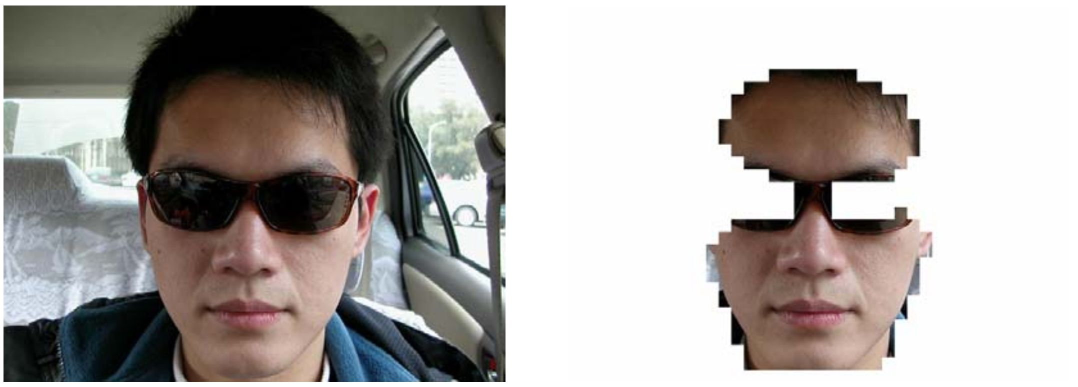
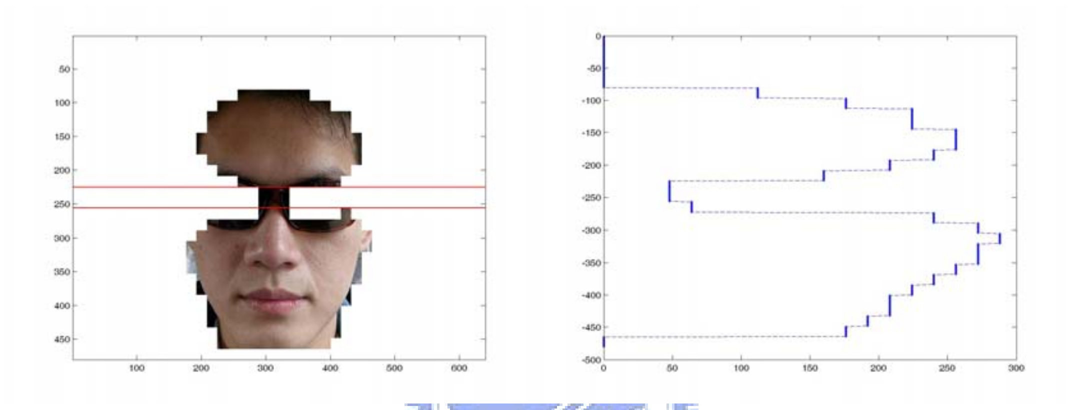


Fig. 3.11. The example of face with sunglasses. (a) The input face image with sunglasses. (b) The face segment derived from (a).

We can utilize the fact that the segment of the face with sunglasses has rapid gap at the position of eye caused from sunglasses to locate the eye wearing sunglasses, and hence to detect the existence of sunglasses. Therefore, we first assign the point of non-face region to zero, and assign the point of face region to the value of one. Then we sum up the face segment row by row into an 1-D vector that presents the quantity of face intensity to locate the eye position. Fig. 3.12 depicts the method above.

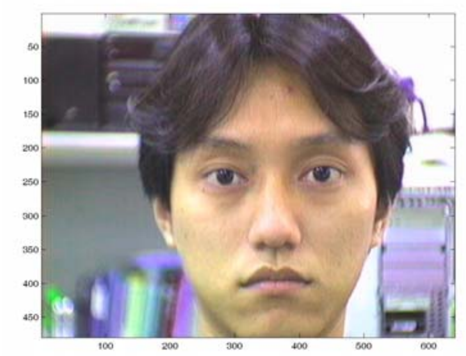


(a) (b)

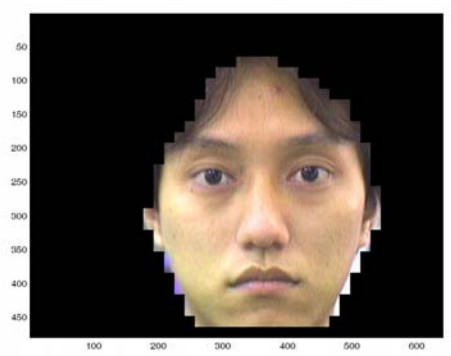
Fig. 3.12. The example of eye detection on sunglasses. (a) The face segment of a face with sunglasses. (b) The 1-D graph derived via summing the row-wise intensity face segment of (a).

In the bare face image, the eye region has more corners and gradient magnitude than other region. Moreover, the face image with glasses has more corners and horizontal edges than without glasses caused from the frame of glasses, nose-piece, and reflections within lens.

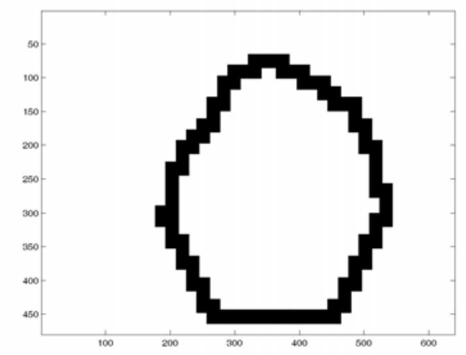
However, when the corner and edge detectors are applied to flat region of a face segment, many spurious responses will be generated. To avoid these responses, we first invoke anisotropic diffusion. Then we use the corner and edge detectors to detect the existence of glasses, and to locate the eye. Therefore, after anisotropic diffusion processing on the face segment, we then apply the corner operator and edge operator. We sum up the two operator’s output into a 1-D vector. Subsequently, we find the position that has peak value from the 1-D vector. The peak point indicates the horizontal position of eyes. Fig. 3.13 and Fig. 3.14 show the method above.


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

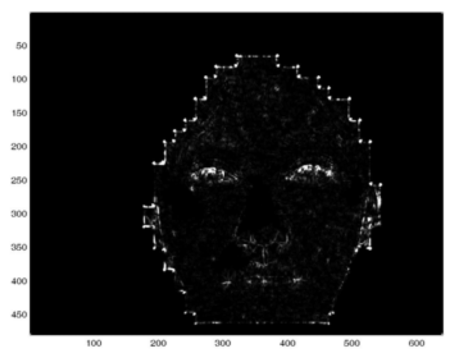
(a)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

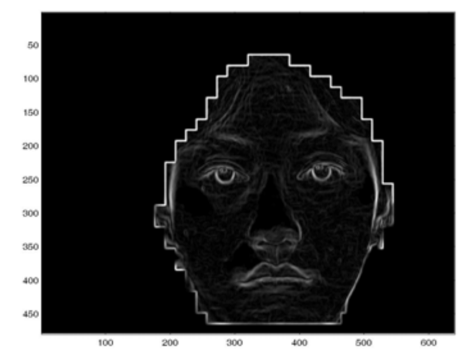
(b)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

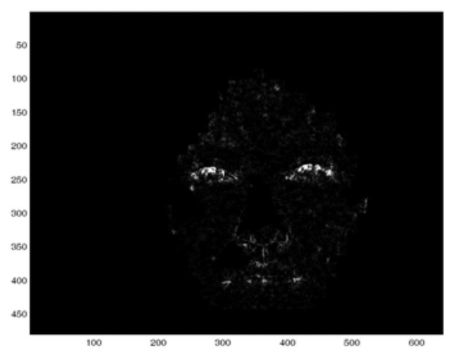
(c)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

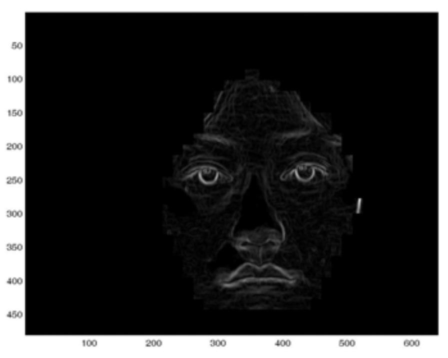
(d)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

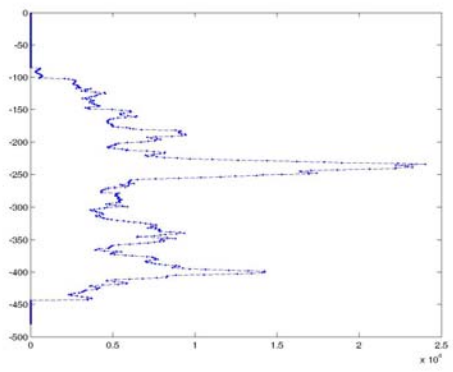
(e)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

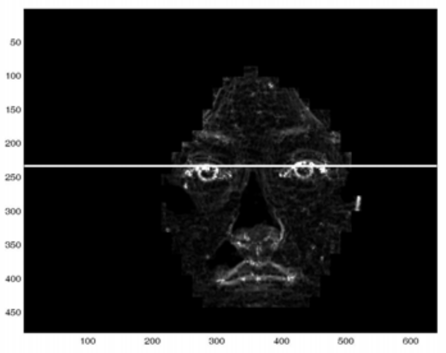
(f)



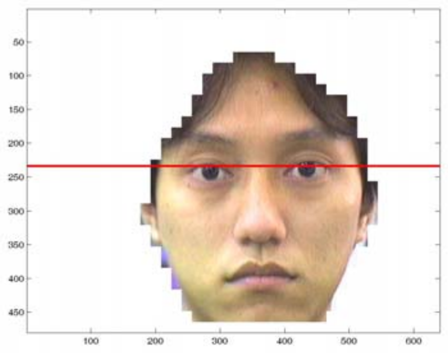
(g)



(h)

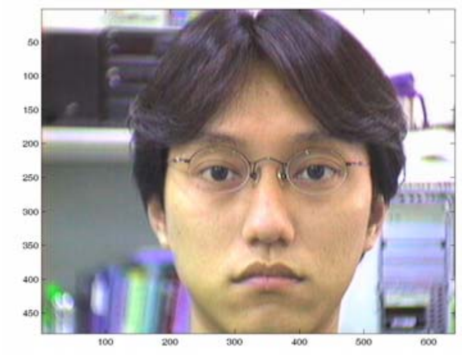


(i)

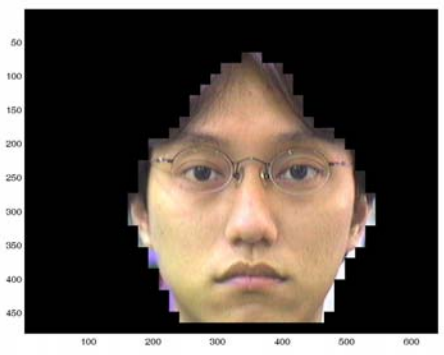


(j)

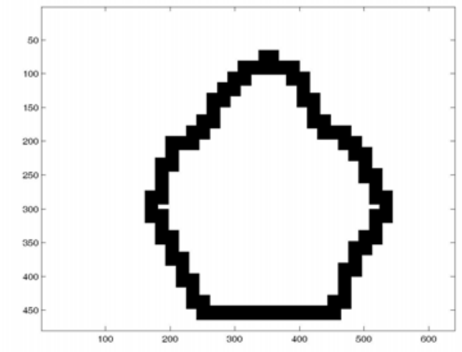
Fig. 3.13. The example of eye location determination on an image of face without glasses. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).


50 100 150 200 250 300 350 400 450
100 200 300 400 500 600

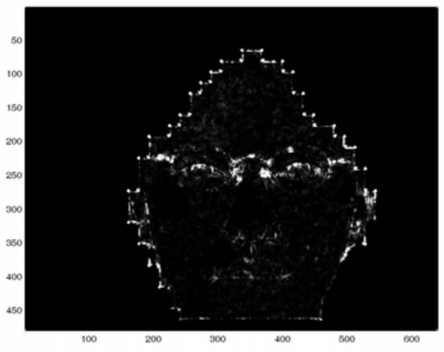
(a)


50 100 150 200 250 300 350 400 450
100 200 300 400 500 600

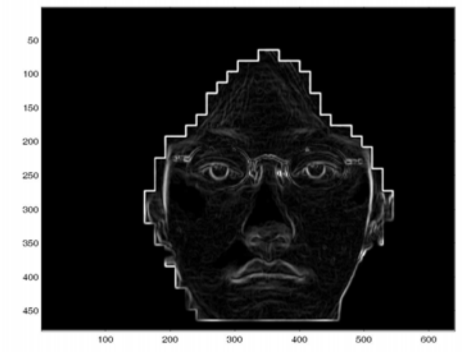
(b)


50 100 150 200 250 300 350 400 450
100 200 300 400 500 600

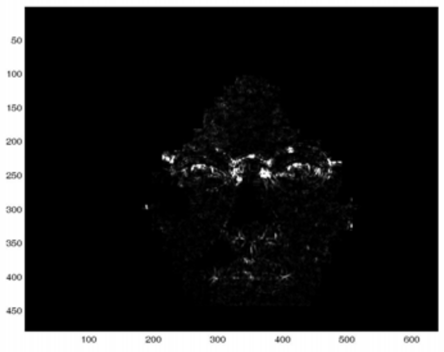
(c)


50 100 150 200 250 300 350 400 450
100 200 300 400 500 600

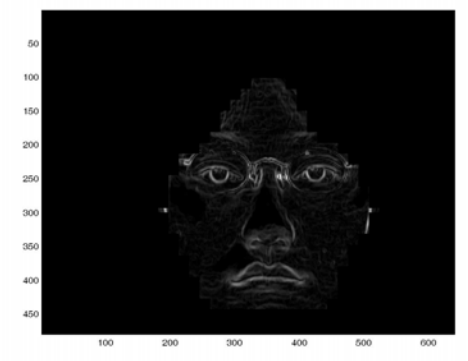
(d)


50 100 150 200 250 300 350 400 450
100 200 300 400 500 600

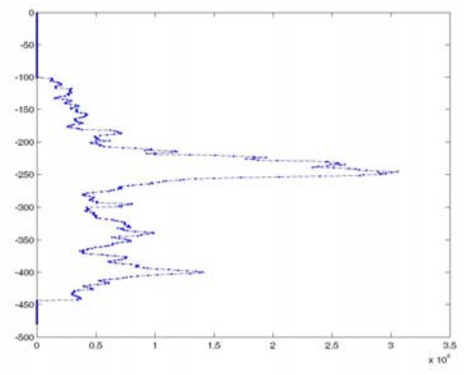
(e)


50 100 150 200 250 300 350 400 450
100 200 300 400 500 600

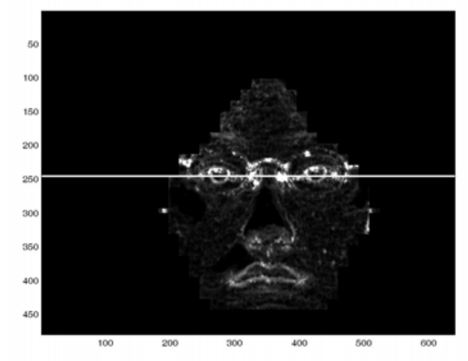
(f)



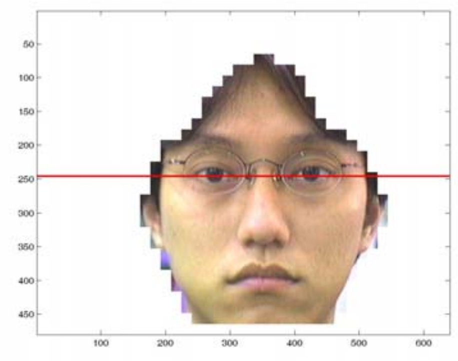
(g)



(h)



(i)



(j)

Fig. 3.14. The example of eye location determination on an image of face wearing glasses. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).

Comparing Figs. 3.13(h) and 3.14(h), it can be seen that the shapes and the maximum of the peak lobe are different. With these differences in the peak lobe, we can select suitable threshold value to determine whether the eyes of a face image wearing glasses or not.

3.2.1 Eyeball Extraction within Glasses

If the glasses are present, we have to perform some operation to extract the eye as below. First, we compute the color edge of the eye region in HSV color model and perform simple morphological operations to get the preliminary edge map of glasses. The color edge detection method proposed by Fan *et al.* [26] use entropic thresholding technique to obtain an optimal threshold that is adaptive to the image contents, and this technique has been proved to be highly efficient for two-class data classification problem [27]. Then, we convert the image of eye region from RGB color space to YC_rC_b color space because in YC_rC_b color model domain the intervals of the C_r and C_b components of skin-color are always very dissimilar from glasses and can easily be clustered to two classes. However, for kinds of glasses, such as metallic and thin-frame, the color of glasses frame sometimes lies in the skin-color interval in YC_rC_b color model because the metallic reflection and the noise caused by low resolution CCD. In order to solve this type problem, we make use of extra information from RGB gradient edge detector. Subsequently, we combine the three evidences to guarantee that the glasses have completely been extracted, despite some noises caused by hair or eyebrows to be included in the map. Fig. 3.15 show an example of edge detection while one wear glasses.

After getting the edge map of detected eye region, we use geometry and projections to eliminate the glasses region and locate accurately the eyes position. When we apply erosion twice, twice better than once empirically, to the edge image of wearing glasses image, the edge will break into small pieces and then the eye can be separated from glasses contour easily by selecting the largest connected component which has the smallest standard deviation to each center of the component. The hair and eyebrows components also can be recognize because they are always from the top to the bottom and begin with the top of the eye region we have set. The extracted eye position result from edge map of wearing glasses example of Fig. 3.15 is demonstrated in Fig. 3.16. Finally, we can estimate the state of eyes by its vertical length and area, and determine whether the driver is drowsy or not in drowsiness detection using PERCLOS and blinking rate, PERCLOS and blinking rate are, respectively, the duration and frequency of eye closure.

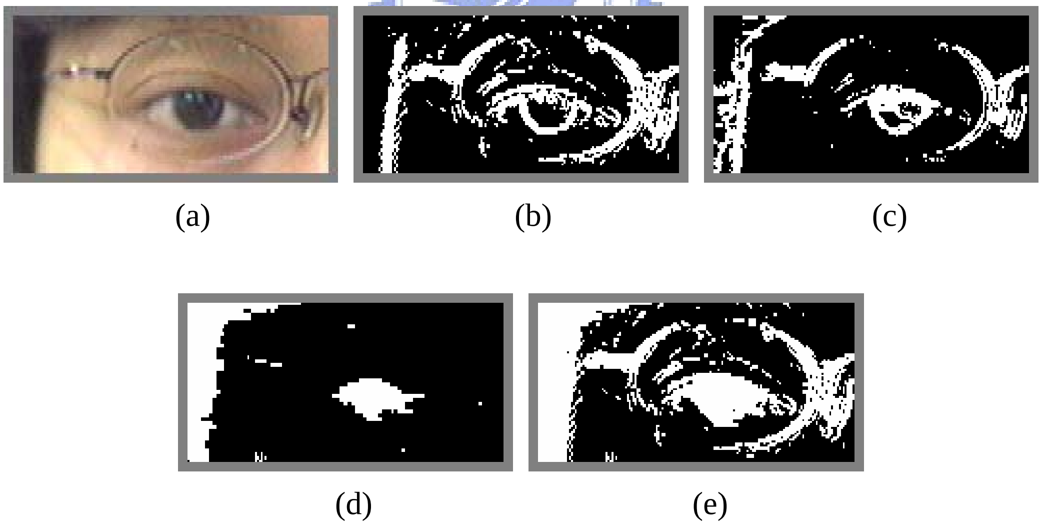


Fig. 3.15. An example of edge detection on eye region. (a) Original image. (b) Edge detection in RGB color space. (c) Edge detection in Hue component of HSV color space. (d) Non-skin-color region. (e) The resultant edge map union the previous three edge map.

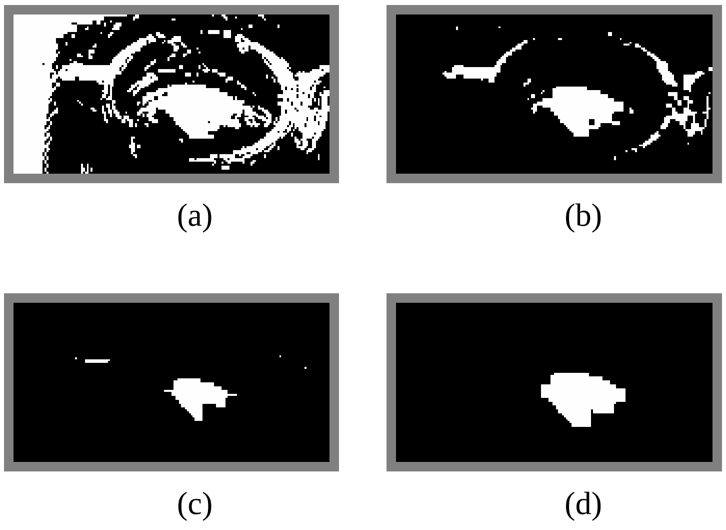


Fig. 3.16. Eye extraction from edge map of Fig. 3.15. (a) Edge map of eye region. (b) Edge map with eliminating the hair region. (c) Edge map eliminating hair region with twice erosion. (d) Extraction the pupil position.

3.3 Reflection Separation within Glasses

When we take a picture from a subject who wears glasses, the image unavoidably catches some reflection within glasses. Therefore, we need to remove the side effect of glasses reflection if we want to detect the open or close degree of eyes.

3.3.1 Introduction to Reflection Separation

As we take a picture through a glass, we often get an image that seems a linear superposition of two images: the image of the scene beyond the glass plus the image of the scene reflected by the glass. Our algorithm for reflection separation was mentioned by Levin *et al.* [19], which assumed the image with reflection can be decomposed into two transparent layers.

Mathematically, the decomposition problem can be posed as follows. We are given an image $I(x,y)$ and wish to find two layers I_1 and I_2 such that

$$I(x,y) = I_1(x,y) + I_2(x,y) \tag{3.17}$$

Obviously, in the absence of additional prior knowledge there are a huge number of possible decompositions. Fig. 3.17 depicts a number of possible decompositions; all of them satisfy Eq. (3.17). In order to choose the "correct" decomposition, we need additional assumptions or conditions.

One might think that in order to correctly decompose such images, an algorithm would need to recognize all objects in images. The algorithm above would possess such high level knowledge to prefer the correct decomposition. But can the "correct" decomposition be chosen without such high level knowledge? In [19], Levin *et al*. [19] propose an algorithm that can decompose reflection images *using a single input image* and without any high level knowledge. The algorithm is based on a very simple cost function: it favors decompositions which have a small number of edges and corners.

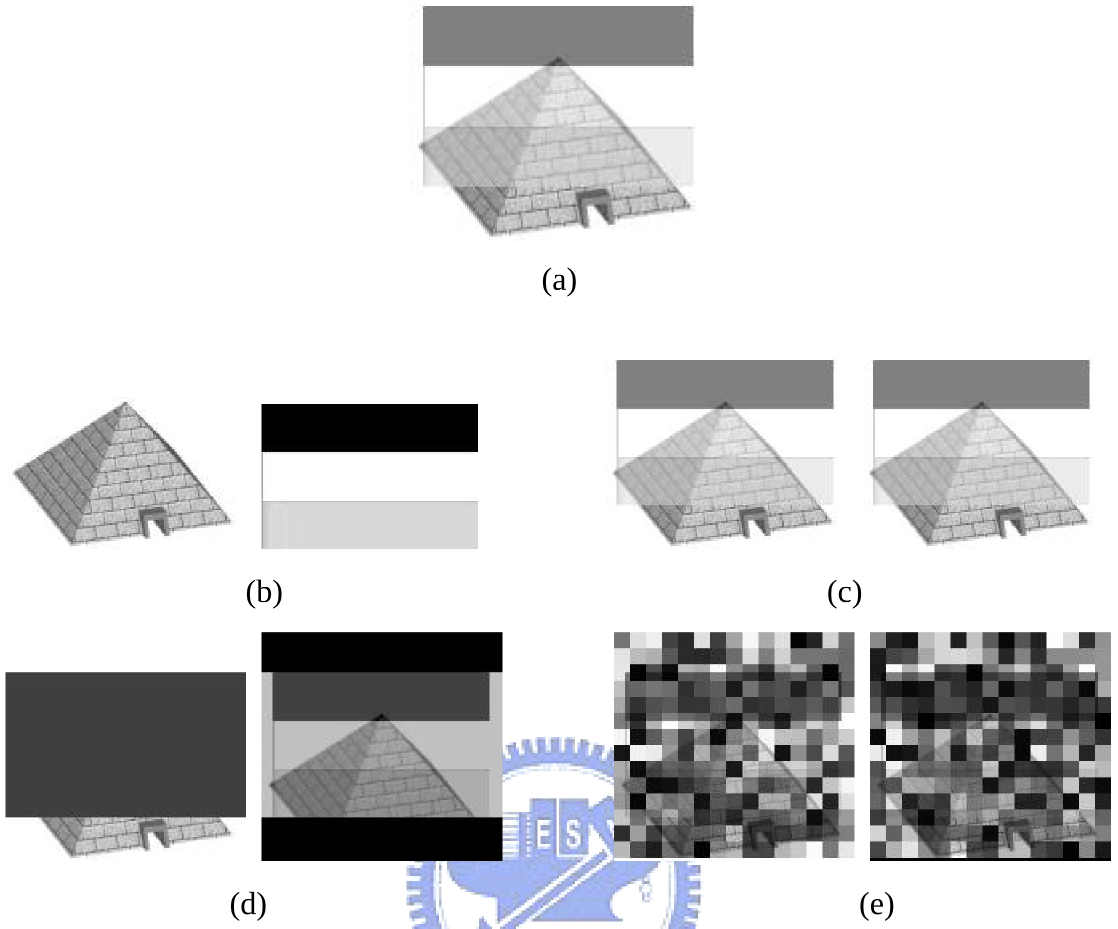


Fig. 3.17. The example of reflection image and it's decomposition. (a) The input image (generated by summing the two images of (b)). (b) The correct decomposition. (c)－(e) alternative possible decompositions.

3.3.2 Edges, Corners, and Cost Function

What is the reason we make use of edges and corners to estimate the "correct" decompositions? At first, consider the simple image in Fig. 3.18(a), the image can be decomposed into an infinite number of possible two layer decompositions. Fig. 3.18(b－e) show some possible decompositions including the decomposition into two squares (the perceptually "correct" decomposition).

Why should the “correct” decomposition be favored? One reason is that out of the decompositions shown in the figure, it minimizes *the total number of edges and corners*. The original image has 10 corners: 4 from each square and two “spurious” corners caused by the superposition of the two images. When we separate the image into two squares we get rid of the two spurious corners and are left only with eight corners. The decomposition shown in the third row increases the number of corners (it has 14 corners) while the bottom decomposition has 8 corners but increase the number of edges.

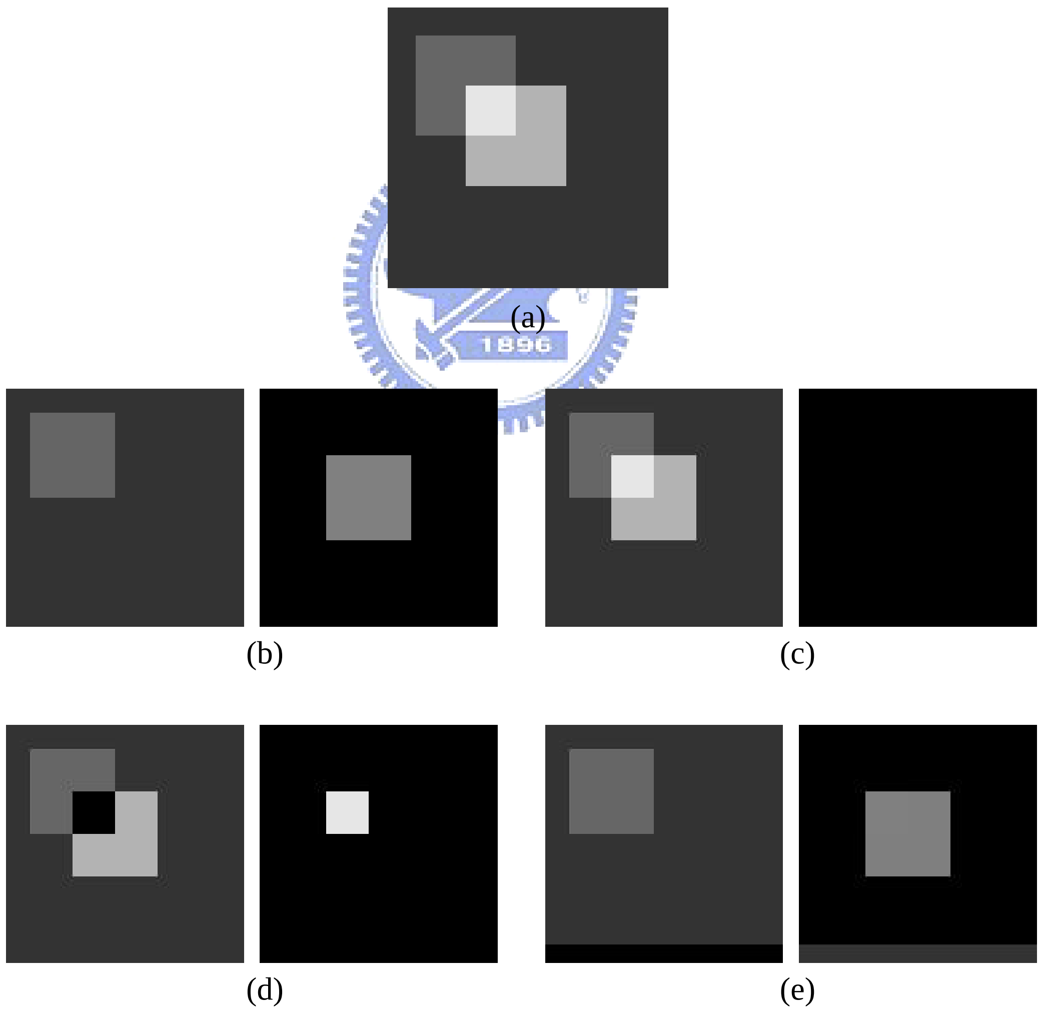


Fig. 3.18. An example of an input image and it’s possible decompositions .

How can we translate the preference for a small number of edges and corners into cost function? We need to make two decisions: (1) what operators to use for edge and corner detectors and (2) what mathematical form to give the cost. We adopt the expression proposed by Levin *et al.* [19], who use the gradient magnitude $\left|\nabla I(x,y)\right|$ as an edge operator and Harris-like operator $c(x,y)$ as corner operator:

$$c(x_0,y_0;I) = \det\left(\sum w(x,y)\begin{pmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y)\end{pmatrix}\right) \tag{3.18}$$

For cost function, we motivate by the qualitative statistics observed from natural images mentioned by Levin *et al.* [28]; it leads to the following cost function for a single layer:

$$cost_1(I) = \sum_{x,y}\left|\nabla I(x,y)\right|^{\alpha} + \eta c(x,y;I)^{\beta} \tag{3.19}$$

with $\alpha = 0.7$, $\beta = 0.25$, $\eta = 15$. The values are obtained from the histograms of the corner and edge operators in natural images and were shown to be critical for the successful decomposition [28]. The cost of two layer decomposition is simply the sum of the costs for each layer separately:

$$cost(I_1,I_2) = cost(I_1) + cost(I_2) \tag{3.20}$$

In real images, however, Detecting edges by edge magnitude and corner via a simple Harris detector arise many spurious "edges" and "corner" in many seemingly flat regions of image. Therefore, we first apply a nonlinear smoothing separately to each layer to diminish the number of spurious "edges" and "corner" found by the gradient and Harris operators. Then we apply Eq. (3.19) to the smoothed layers. In this thesis, we adopt anisotropic diffusion to achieve the task of nonlinear smoothing.

Thus our cost function for a single layer now is:

$$cost_2(I) = \sum_{x,y} \left|\nabla \hat{I}(x,y)\right|^{\alpha} + \eta c\left(x,y;\hat{I}\right)^{\beta} \tag{3.21}$$

where $\hat{I}$ is the layer after applying anisotropic diffusion [25].

3.3.3 Oriented filters

In this section, we introduce oriented filters because they will be utilized as the measure to find the correct decomposition. There are many approaches proposed in the computer vision literature by convolving the image with a bank of linear spatial filters f_i tuned to various orientation and spatial frequencies. The oriented filter we exploit is introduced in Malik *et al*. [29]. The oriented filterbank we used in this thesis, depicted in Fig. 3.19, is based on rotated copies of Gaussian derivative and its Hilbert transform. More precisely, let $f_1(x,y) = G''_{\sigma 1}(y)G_{\sigma 2}(x)$ and $f_2(x,y)$ equal the Hilbert transform of $f_1(x,y)$ along the *y* axis:

$$f_1(x,y) = \frac{d^2}{dy^2}\left(\frac{1}{C}\exp\left(\frac{y^2}{\sigma^2}\right)\exp\left(\frac{x^2}{\ell^2\sigma^2}\right)\right) \tag{3.22}$$

$$f_2(x,y) = Hilbert\left(f_1(x,y)\right) \tag{3.23}$$

where σ is the scale, ℓ is the aspect ratio of the filter, and *C* is a normalization constant.

Now assume that the image is convolved with such a bank of linear filters. We will refer to the collection of response images $I * f_i$ as the *hypercolumn transform* of the image. Malik *et al*. [29] state the hypercolumn transform provides a convenient front end for contour and texture analysis; therefore, it is the reason why we utilize oriented filter as the measure to find the correct decomposition.

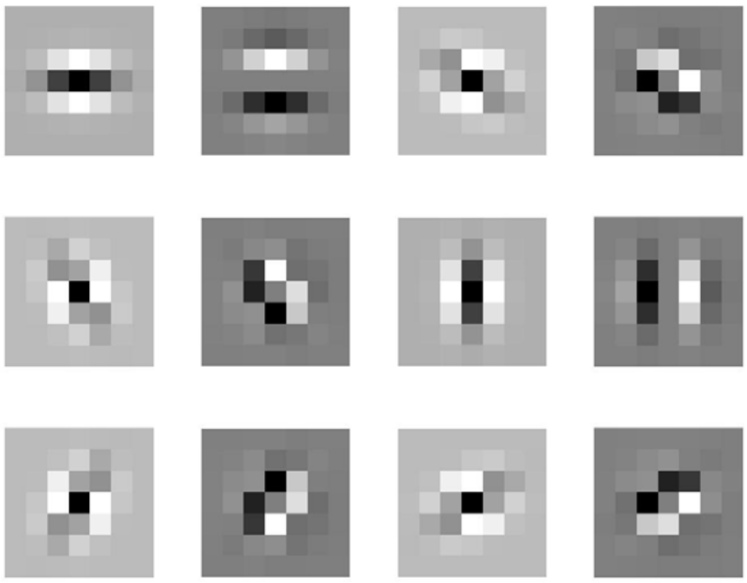


Fig. 3.19. The example of A filterbank. Filter set consisting of 2 phase (even and odd) and 6 orientations (equally spaced from 0 to π). The first and third column is the image of $f_1(x, y)$ with size 7x7; the second and forth column is the correspond Hilbert transform of first and third column.

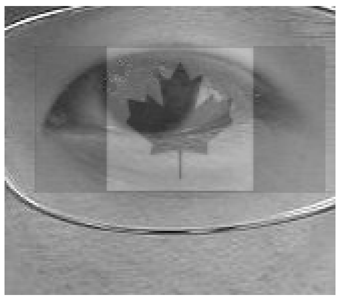
3.3.4 Implementation

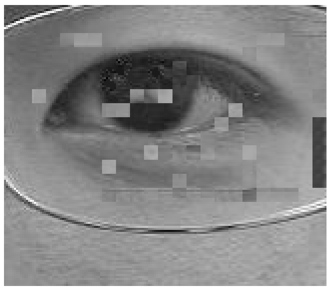
In this section, we describe the detail of implementation because we have some processes different to the one proposed by Levin *et al*. [19]. In [19], in order to optimizing possible decomposition, Levin *et al*. [19] discretized the problem by dividing the image into small 7×7 overlapping patches and restrict the search to 20 possible decompositions for each patch. Therefore, in this thesis we divide the image into small 7×7 non-overlapping patches p to reduce the computational time. Then we build the patches database for searching the possible decompositions for each patch of the input image. Our patches database just contains the eye area in the face image with glass; because we only are interest in the layer that possesses the eye information, not the reflection layer.

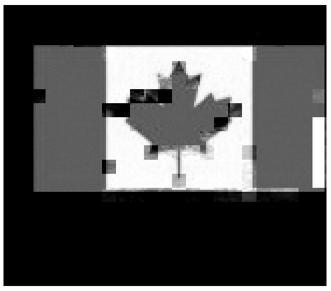
To find out the candidate patches $p \approx p_1 + p_2$ we searched the database for the patches q minimizing

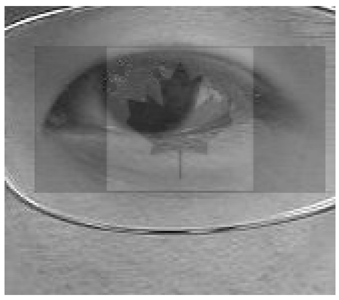
$$\sum_i \left| f_i * p - f_i * q \cdot s \right| \tag{3.24}$$

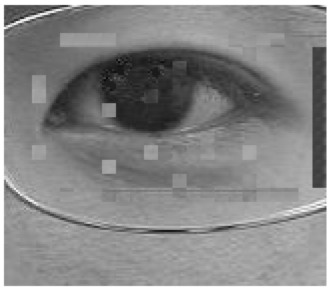
where f_i is the oriented filterbank mentioned above. For any candidate patch q and the optimal contrast s were found via minimizing this equation. Where optimal contrast s is exploited to conquer the change of intensity at patches as patches is located at the doubtful reflection region. However, we just search candidate patches to reconstruct the non-reflection layer from the patches database. Since, it arises some misdiagnosed patches at the recovery image when the reflection layer has more strong intensity of feature than the non-reflection layer. Fig. 3.20 depicts the example we separate reflection by find out the patches by Eq. (3.24).


(a)


(b)


(c)


(d)


(e)

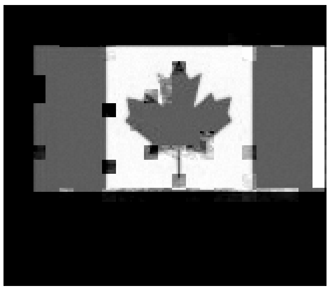

(f)

Fig. 3.20. Example of separation results of images with reflections using discretization. (a) The input image consisting of the foreground image multiplied by 0.8 and the reflection image multiplied by 0.2. (b) Separated foreground layer image of (a). (c) Separated reflection layer image of (a). (d) The input image consisting of the foreground image multiplied by 0.85 and the reflection image multiplied by 0.15. (e) Separated foreground layer image of (d). (f) Separated reflection layer image of (d).

Chapter 4. Simulation and Results

We have tested the eye-detection algorithm on a number of people in order to confirm the validity and stability. First, we obtain frontal face images of people from a CCD camera, and then we demonstrate the algorithm via these images. In Section 4.1, we will show the step-by-step result of finding the eyes. Subsequently, we will show the experiment result of reflection separation within glasses in Section 4.2. The size of images is 640×480 and the simulation is processing on a Pentium IV 3.2GHz personal computer.

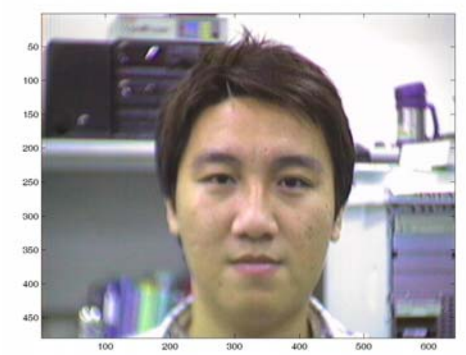
4.1 Eye detection

We show five example of eye detection on the images of bare face, three examples of eye detection on the images of face wearing glasses, and one example of eye detection on the image of face wearing light sunglasses in Section 4.1.1. Then We show two example of eye detection on the images of face wearing sunglass in Section 4.1.2.

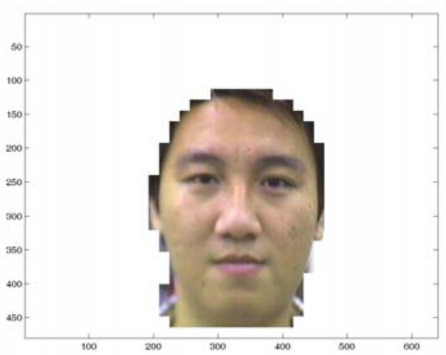
4.1.1 Eye detection on the image of bare face, face wearing glasses, and face wearing light sunglasses

In Figs. 4.1–4.5, we show five example of eye detection on the images of bare face. In Figs. 4.6–4.8 we show three examples of eye detection on the images of face wearing glasses. In Fig. 4.9 we show one example of eye detection on the image of face wearing light sunglasses.

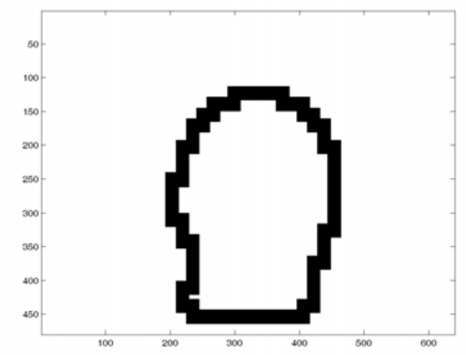
In each example, we will show the original face image in (a), and face extraction by skin-color map in (b). The boundary of face segment produced by erosion operation is depicted in (c). The corner response and edge response of (b) are depicted in (d) and (e), respectively. The corner response and edge response inside the boundary of face segment are depicted in (f) and (g), respectively. (h) is the 1-D graph showing the sum of (f) and (g). (i) shows the eye location is detected by the position that has the peak value at the 1-D graph of (h). (j) locates the detected eye position in the face segment of (b).


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

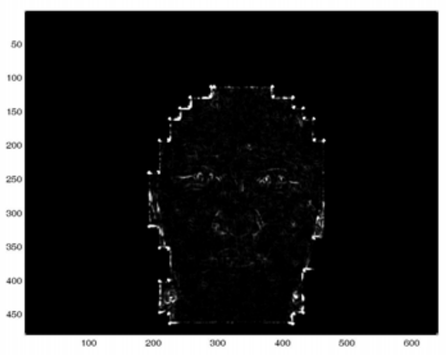
(a)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

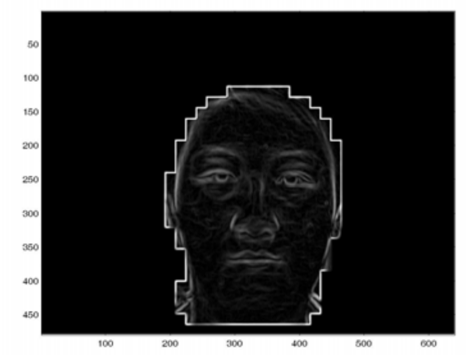
(b)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

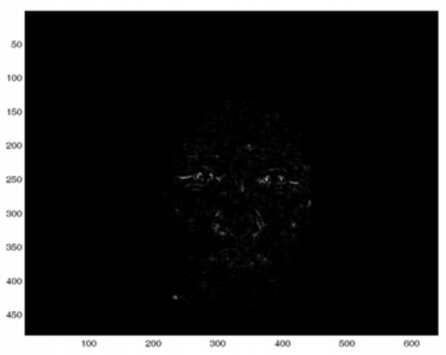
(c)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

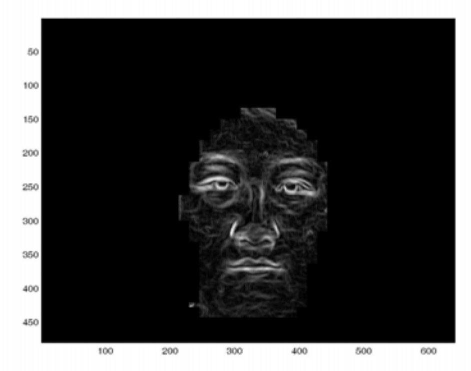
(d)

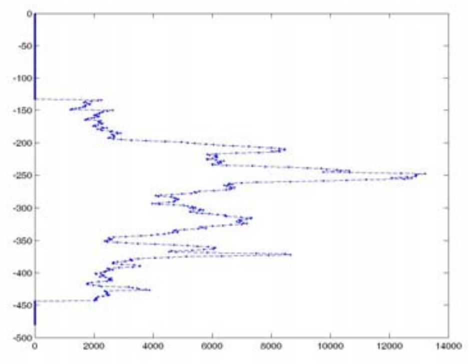

50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

(e)

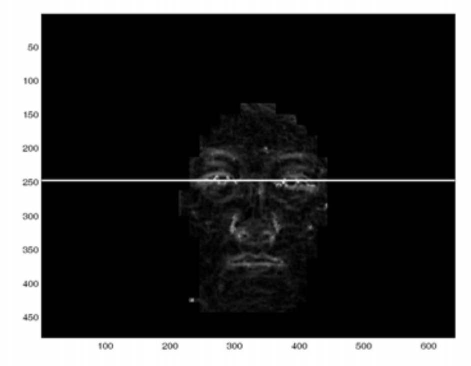

50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

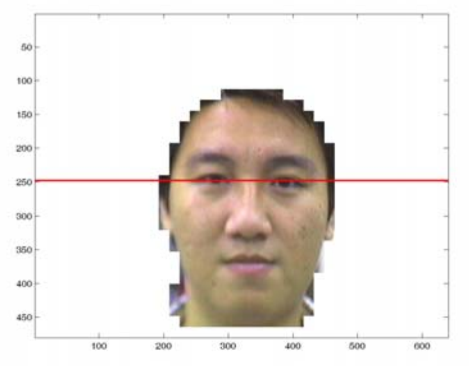
(f)





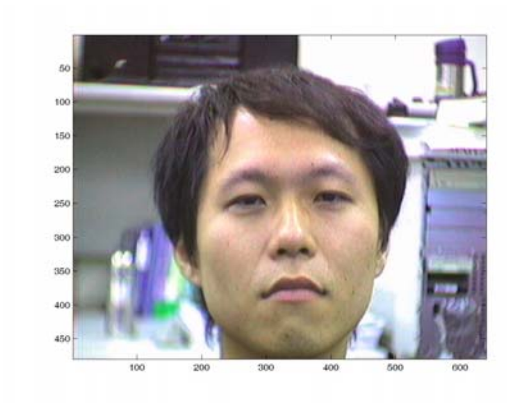
(g) (h)



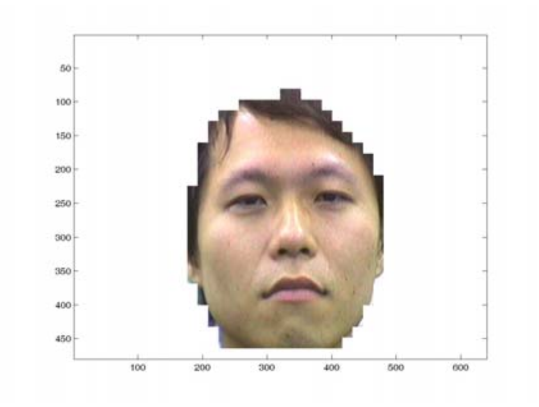


(i) (j)

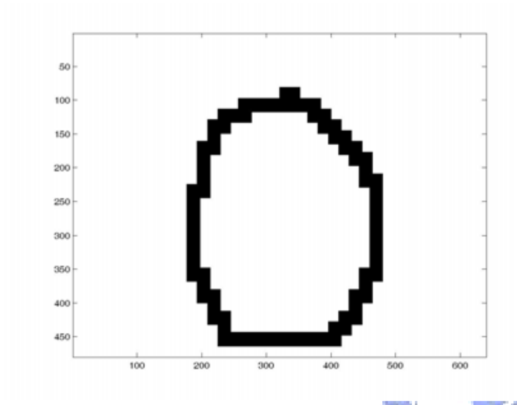
Fig. 4.1. The example 1 of eye location on the image with bare face. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

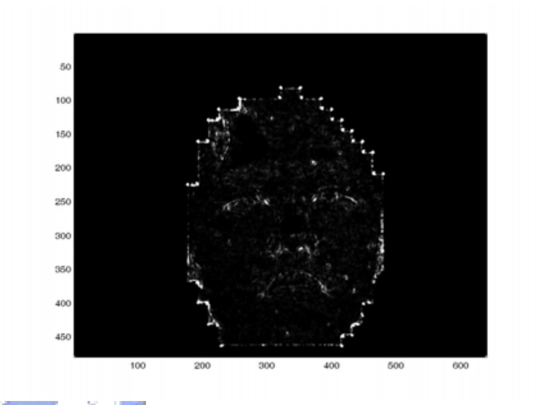
(a)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

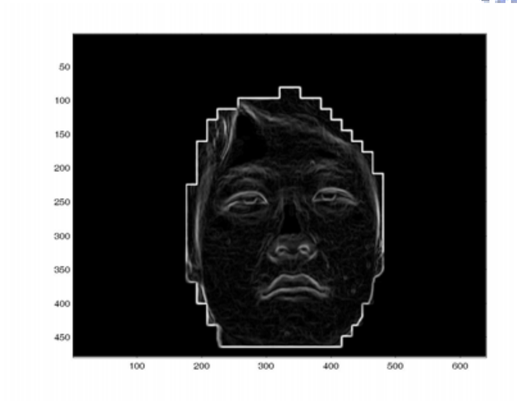
(b)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

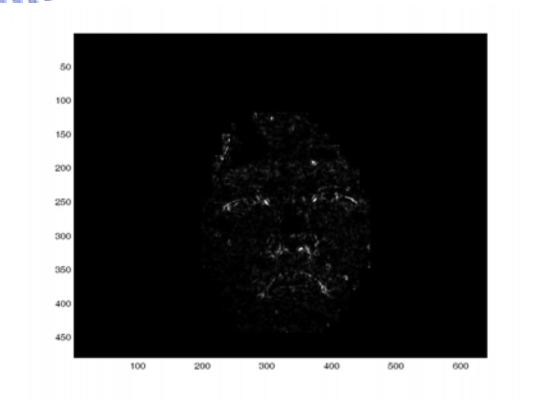
(c)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

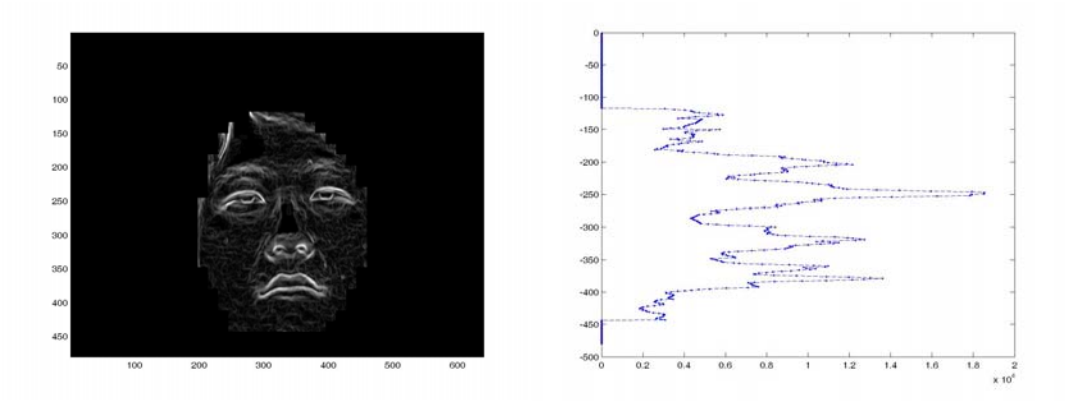
(d)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

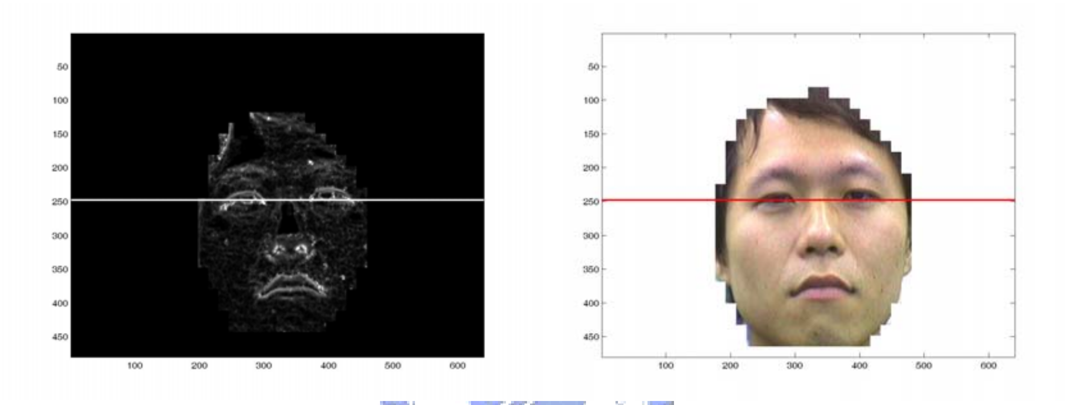
(e)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

(f)

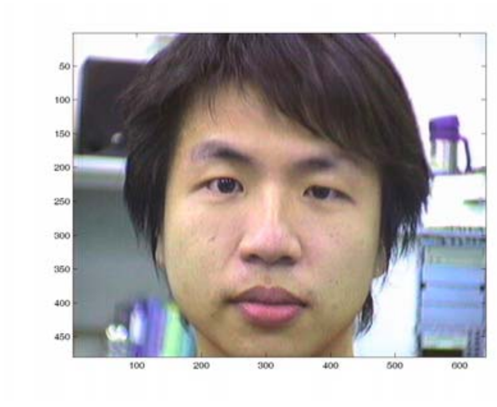


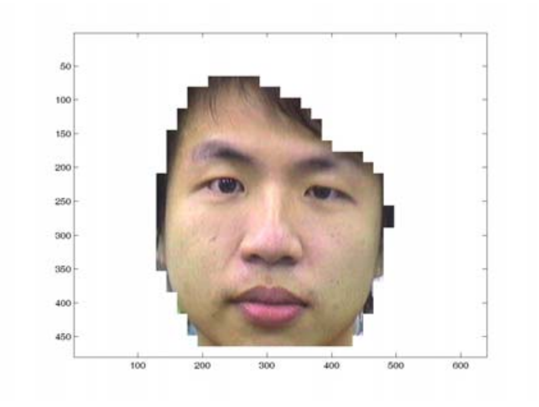
(g) (h)

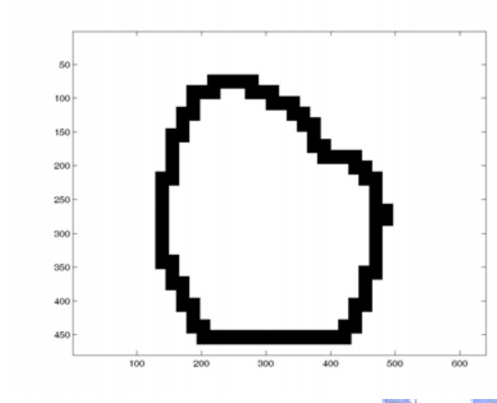


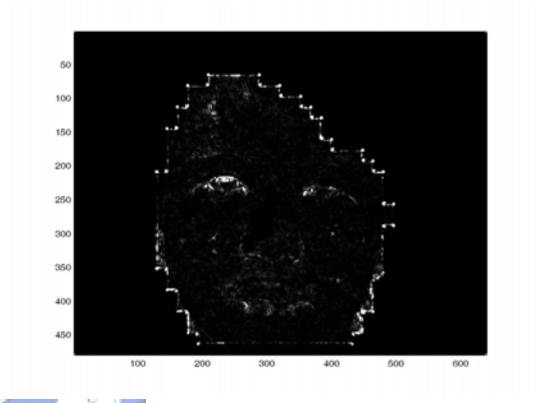
(i) (j)

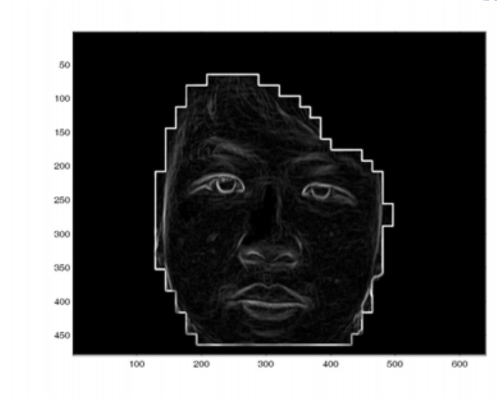
Fig. 4.2. The example 2 of eye location on the image with bare face. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).

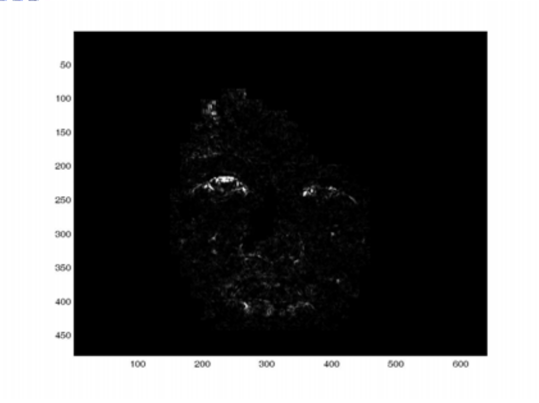

(a)

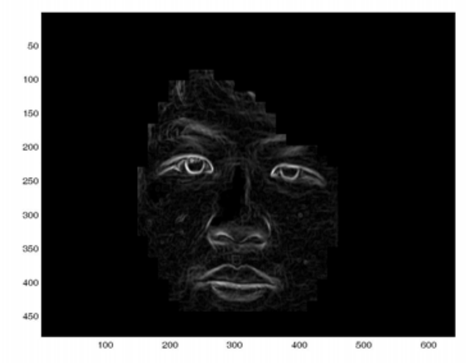

(b)


(c)

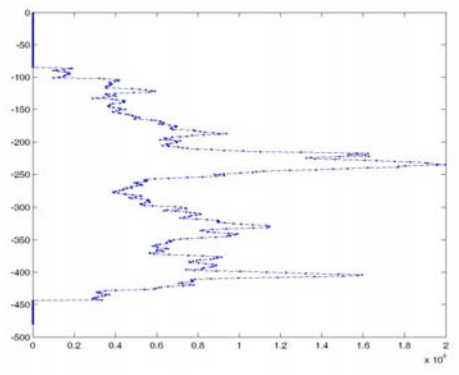

(d)


(e)

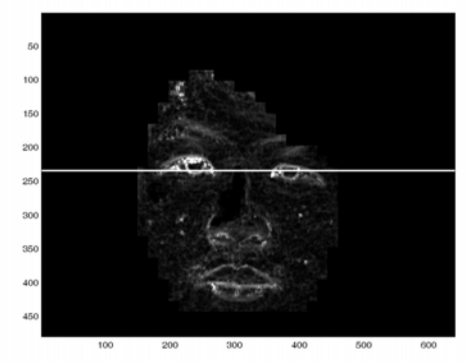

(f)



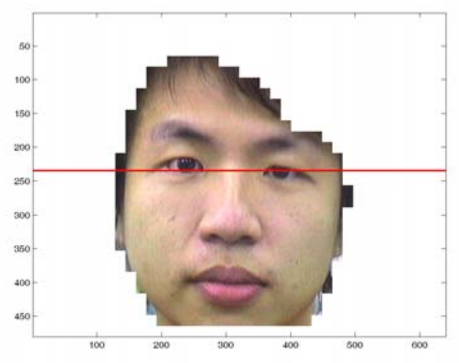
(g)



(h)

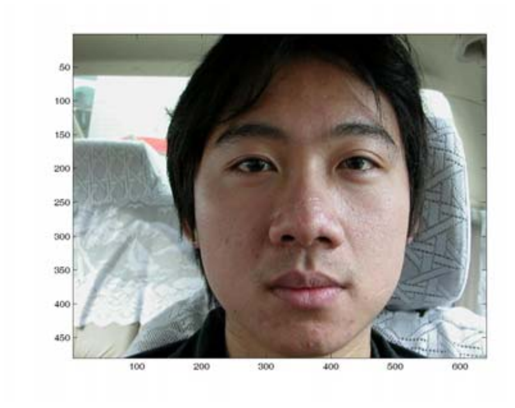


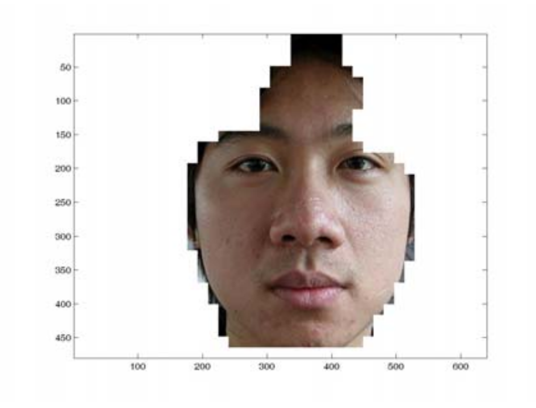
(i)

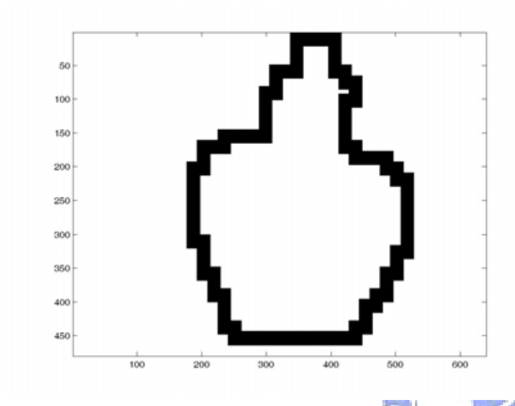


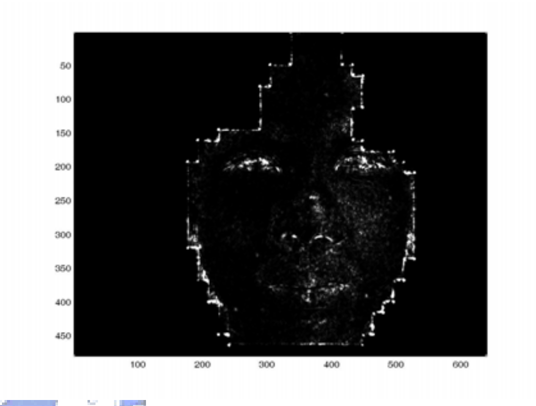
(j)

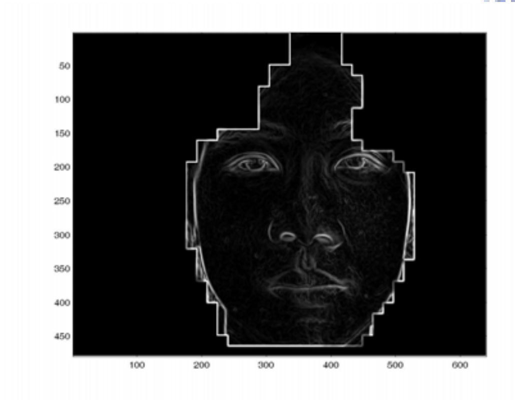
Fig. 4.3. The example 3 of eye location on the image with bare face. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).

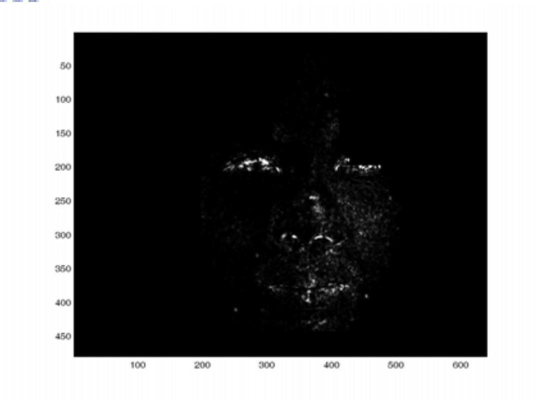

(a)

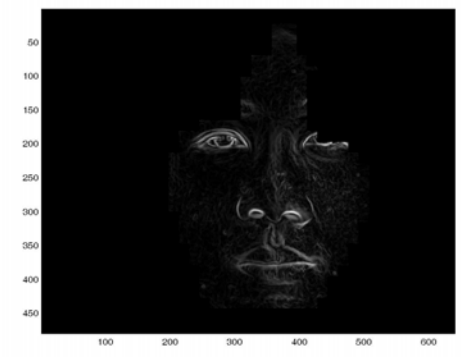

(b)


(c)

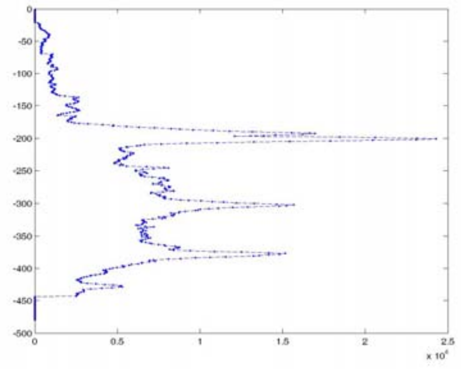

(d)


(e)

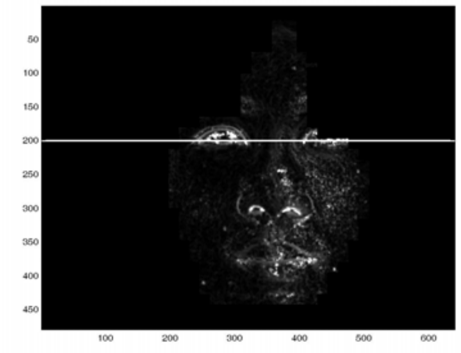

(f)



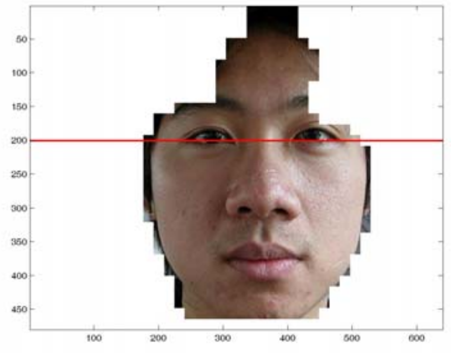
(g)



(h)

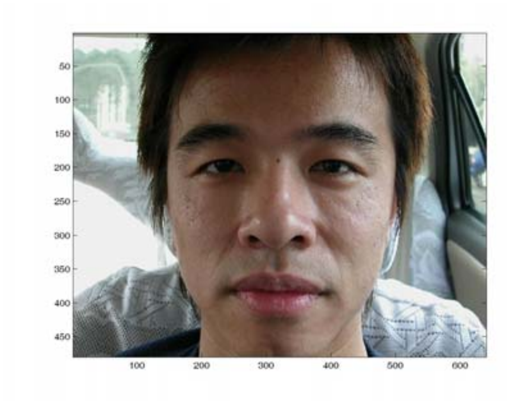


(i)

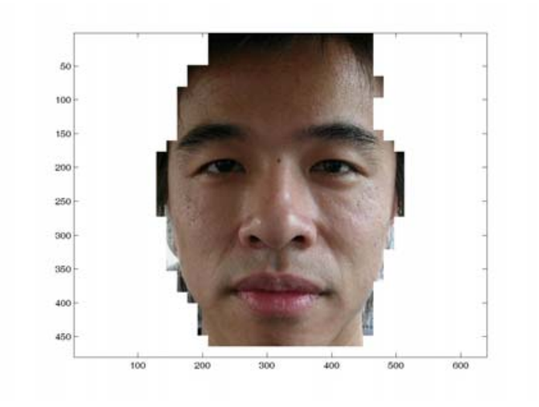


(j)

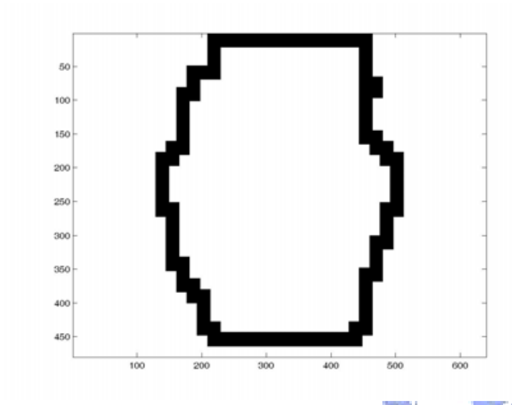
Fig. 4.4. The example 4 of eye location on the image with bare face. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

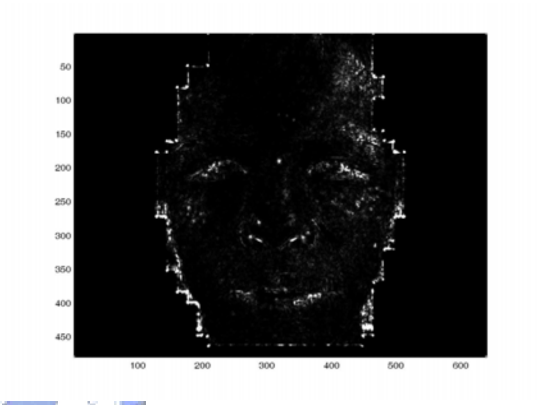
(a)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

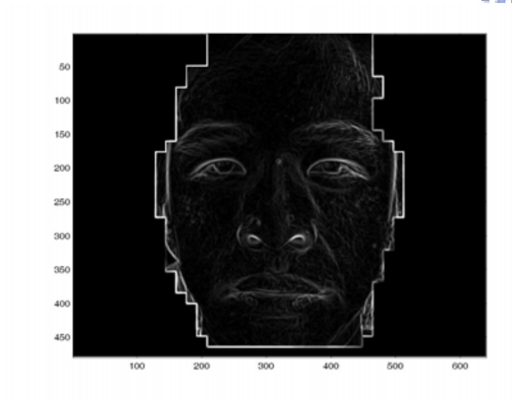
(b)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

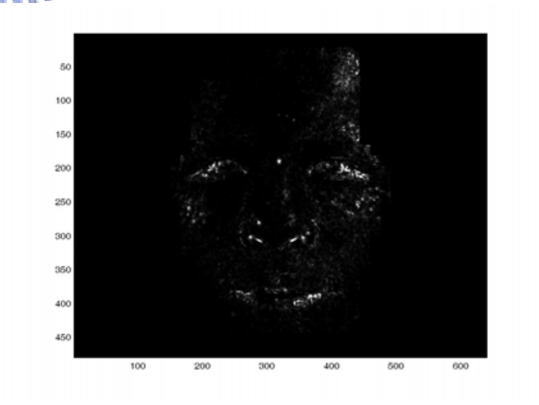
(c)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

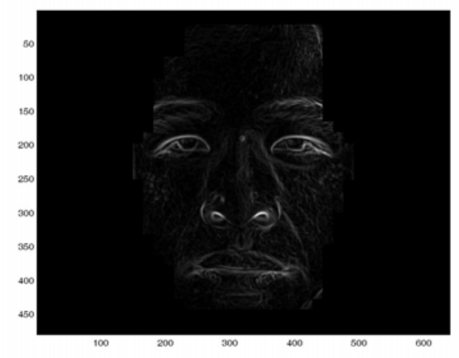
(d)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

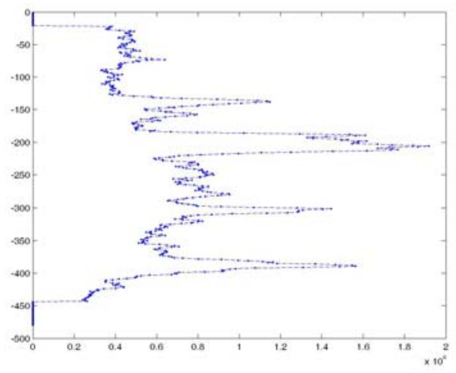
(e)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

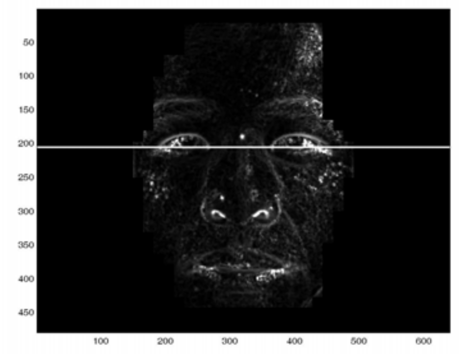
(f)



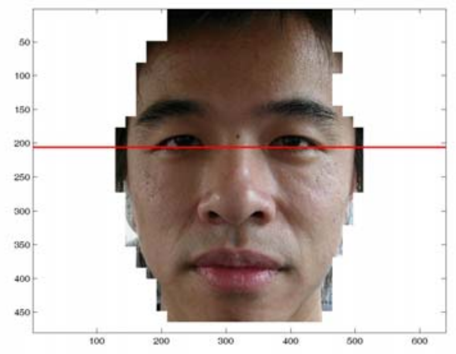
(g)



(h)



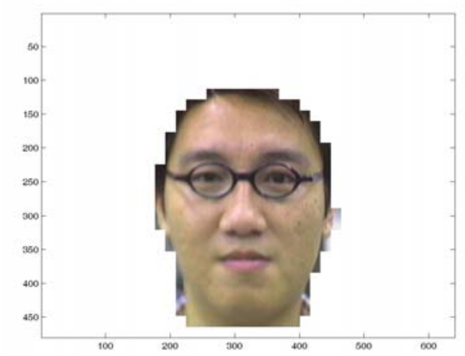
(i)



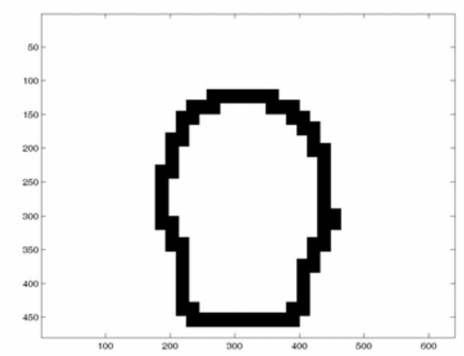
(j)

Fig. 4.5. The example 5 of eye location on the image with bare face. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).

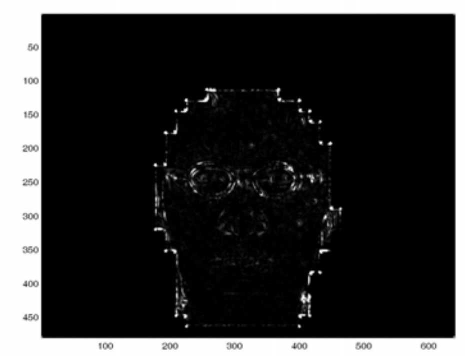
(a)



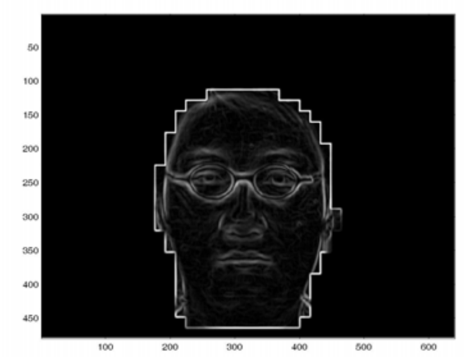
(b)



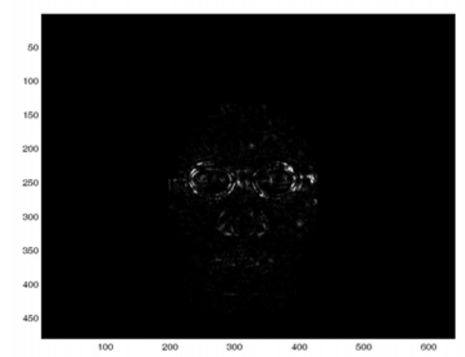
(c)



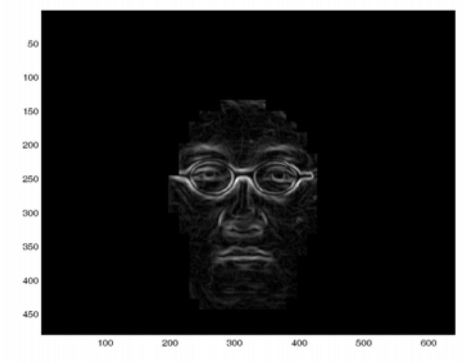
(d)



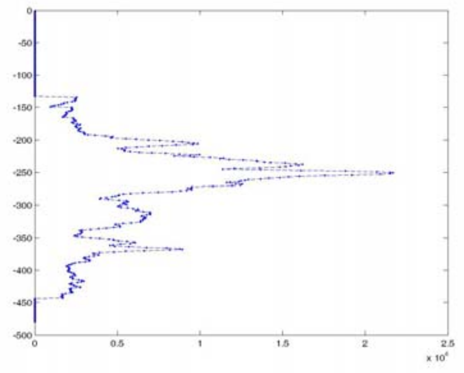
(e)



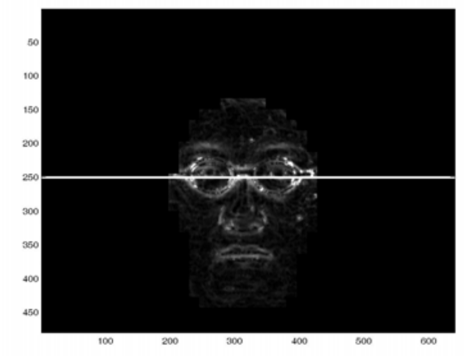
(f)



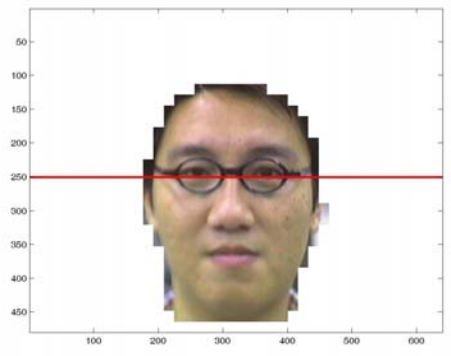
(g)



(h)

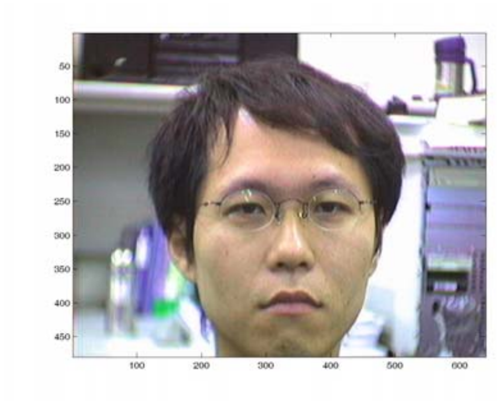


(i)

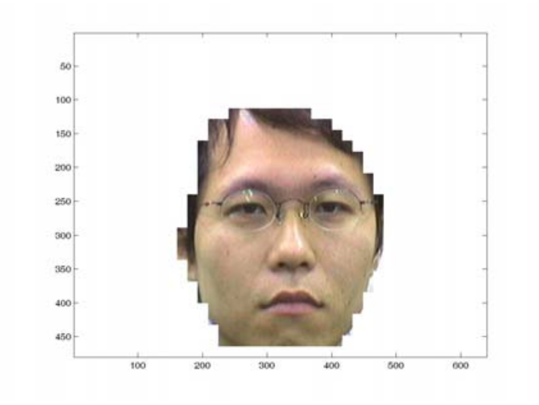


(j)

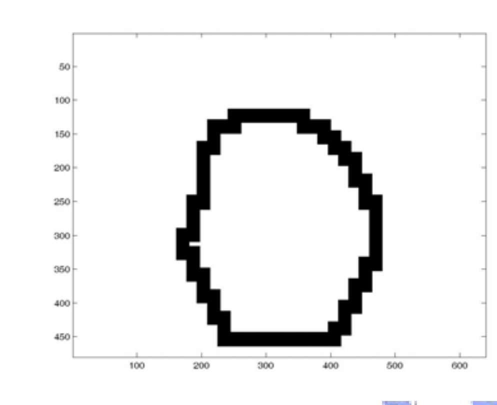
Fig. 4.6. The example 1 of eye location on the image of face wearing glasses. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).



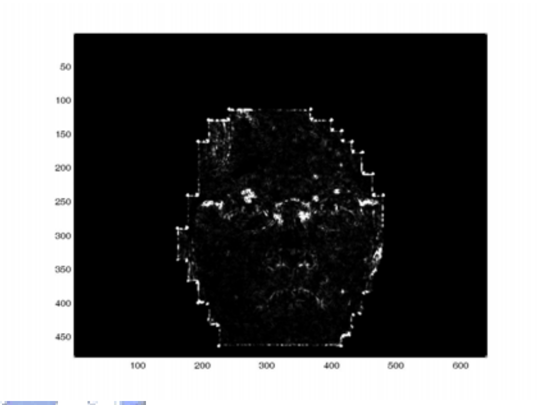
(a)



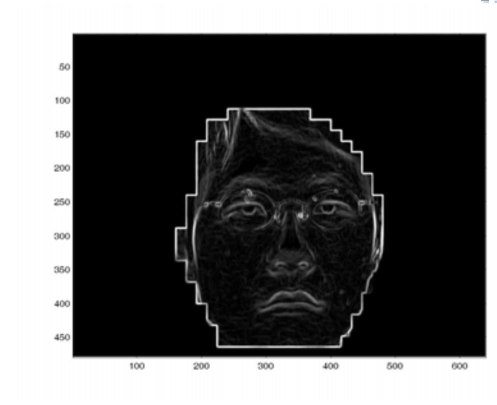
(b)



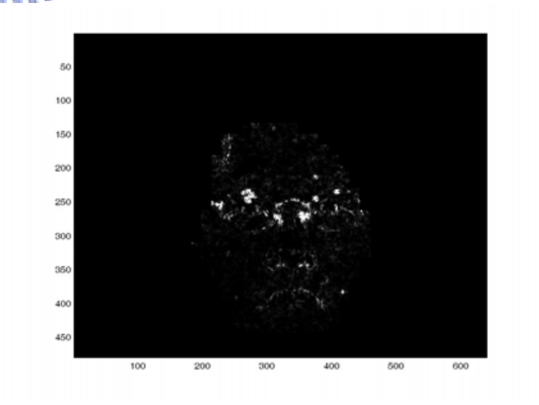
(c)



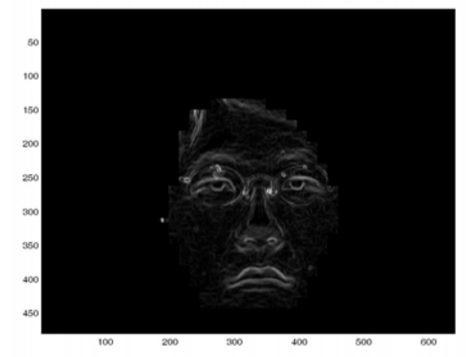
(d)

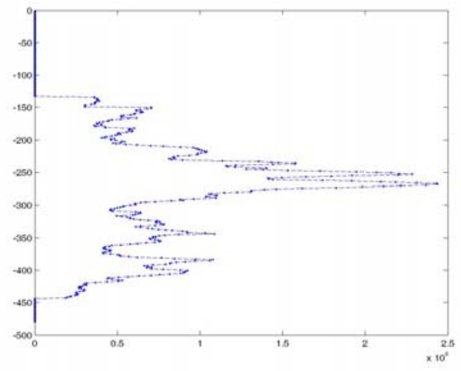


(e)

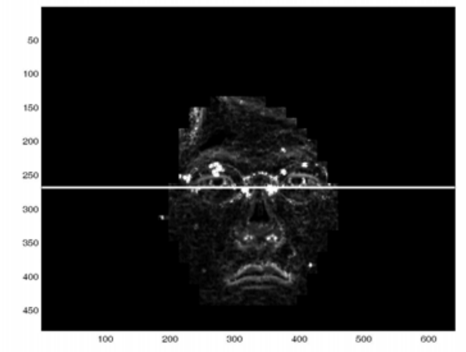


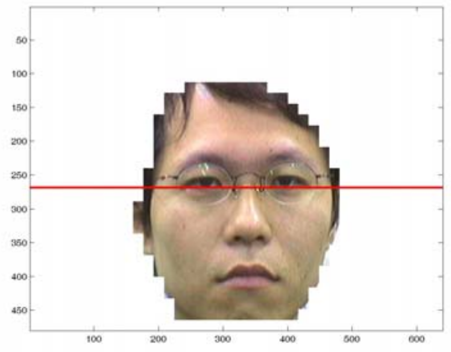
(f)





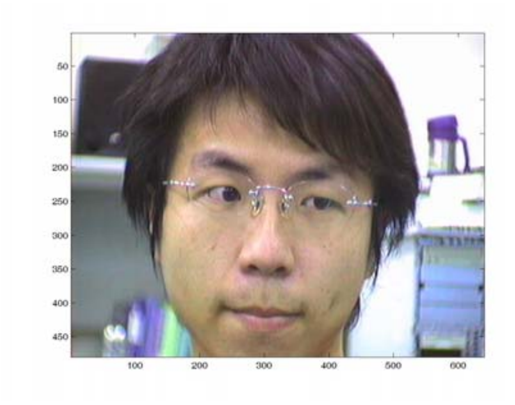
(g) (h)



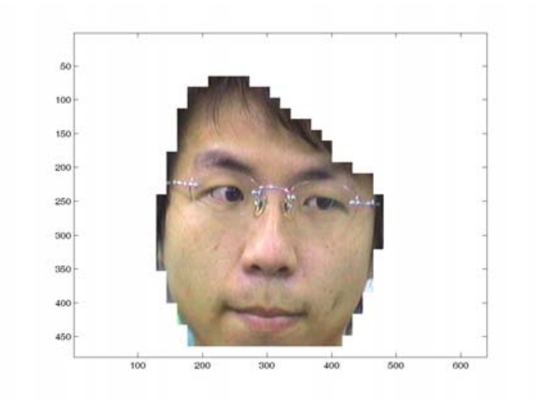


(i) (j)

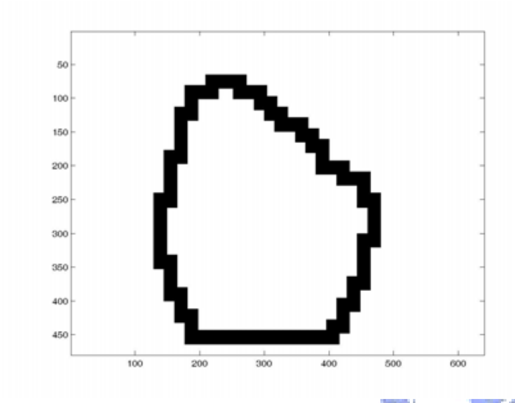
Fig. 4.7. The example 2 of eye location on the image of face wearing glasses. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

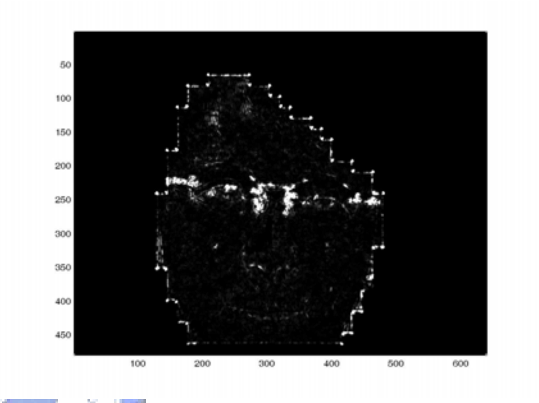
(a)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

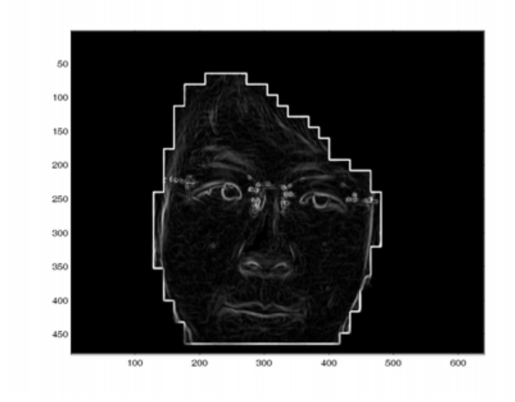
(b)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

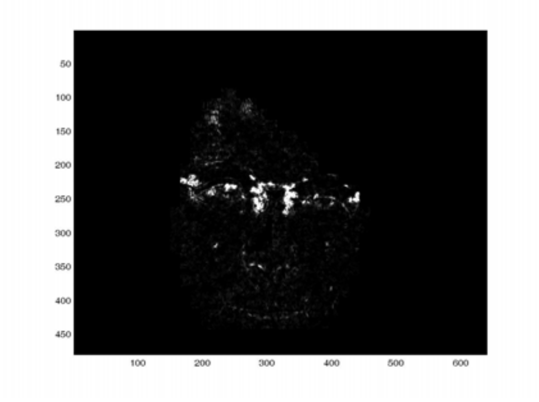
(c)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

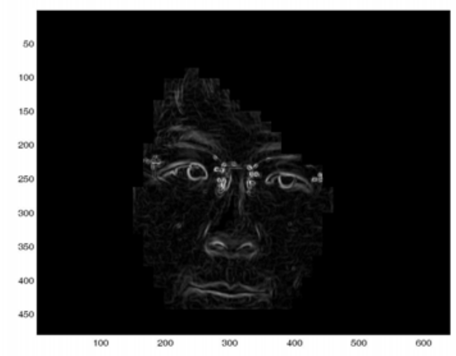
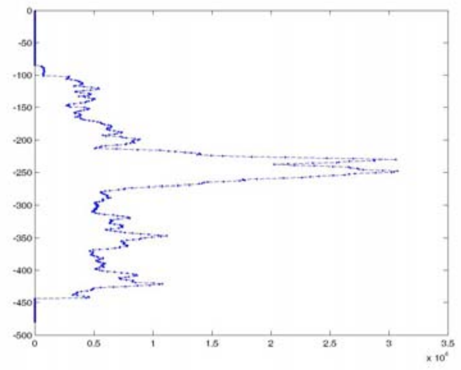
(d)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

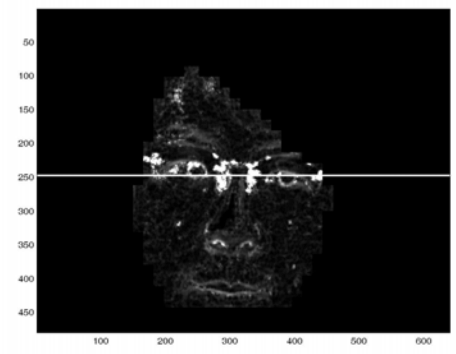
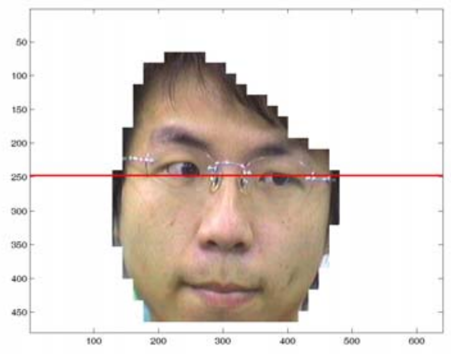
(e)


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

(f)

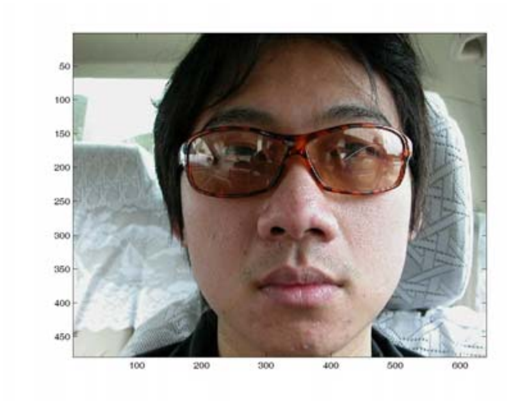



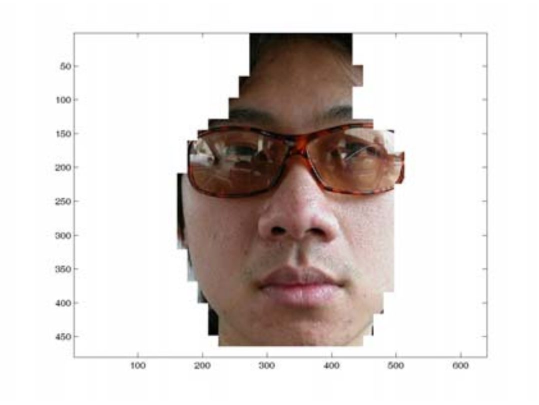
(g) (h)

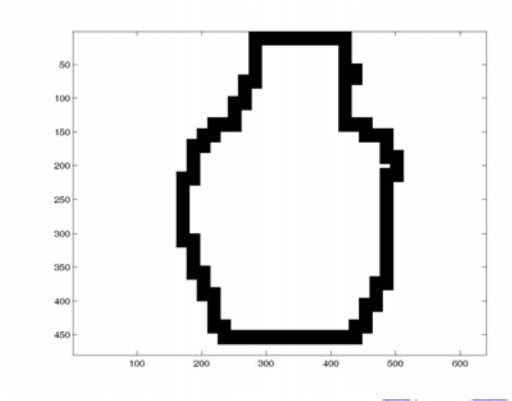



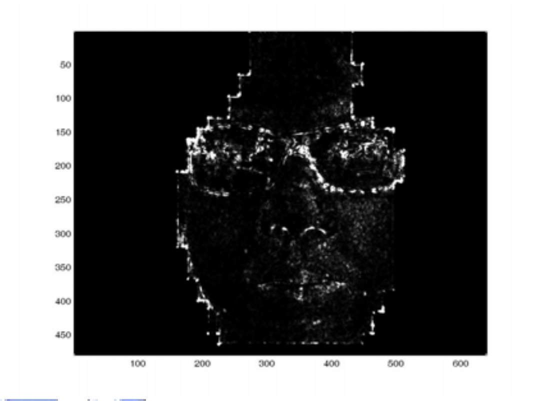
(i) (j)

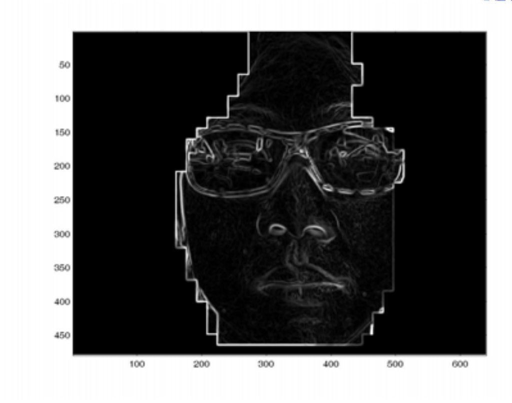
Fig. 4.8. The example 3 of eye location on the image of face wearing glasses. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).

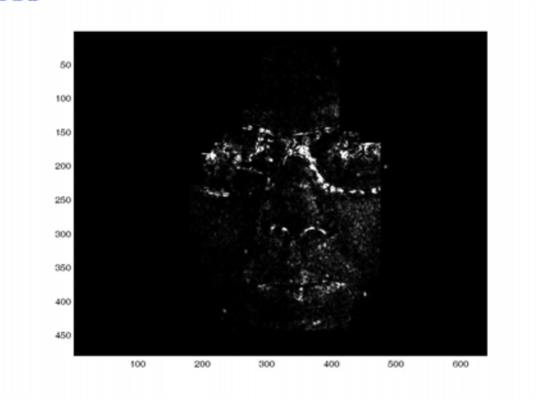

(a)

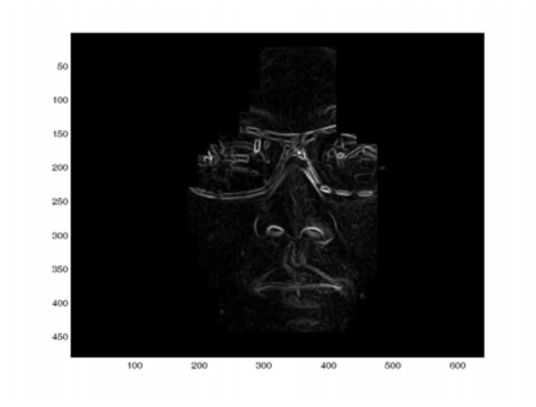

(b)

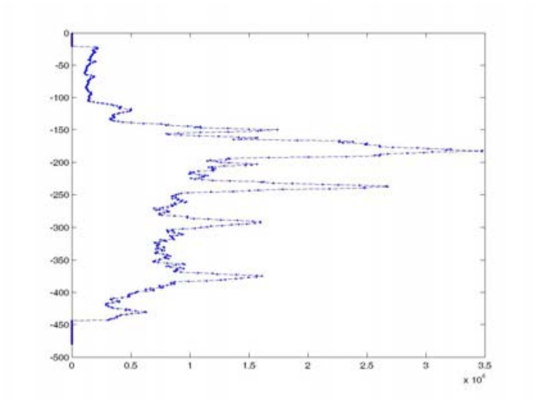

(c)

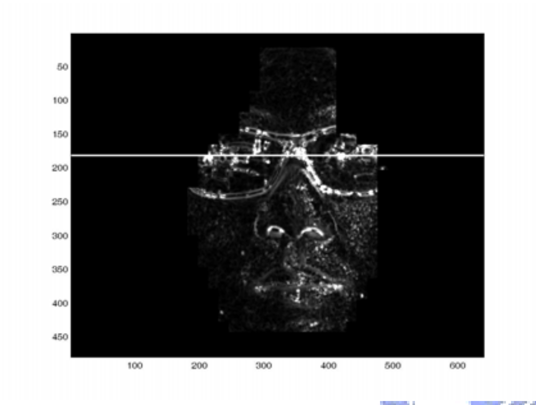

(d)


(e)


(f)


(g)


(h)


(i)

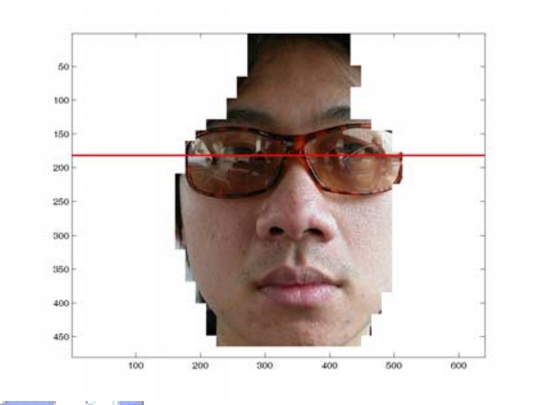
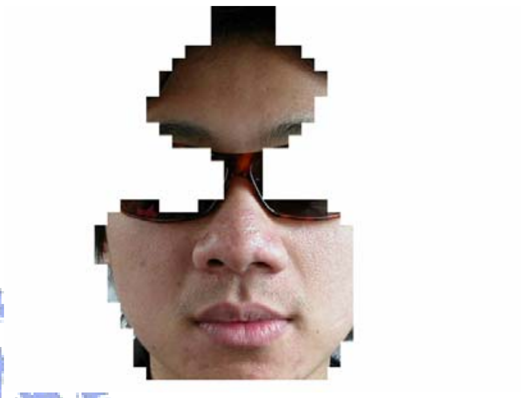

(j)

Fig. 4.9. The example of eye location on the image of face wearing light sunglasses. (a) The input face image. (b) Face extraction by skin-color map. (c) The boundary of face segment by erosion operation. (d) The corner response of (b). (e) The edge response of (b). (f) The corner response inside the boundary of face segment. (g) The edge response inside the boundary of face segment. (h) The 1-D graph showing the sum of (f) and (g). (i) The eye location detected by the position that has the peak value at the 1-D graph of (h). (j) The detected eye position in the face segment of (b).

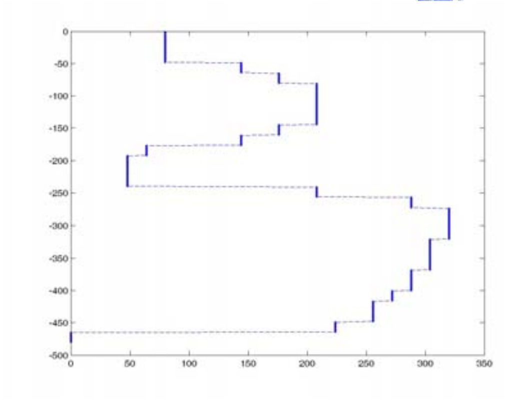
4.1.2 Eye detection on the image of face wearing dark sunglasses

In Figs. 4.10–4.11, we show two examples of eye detection on the images of face wearing sunglasses. (a) shows the input face image; (b) is the face segment derived from (a); (c) is the 1-D graph derived via summing the row-wise intensity of face segment of (b); (d) Locates the eye location at (b).

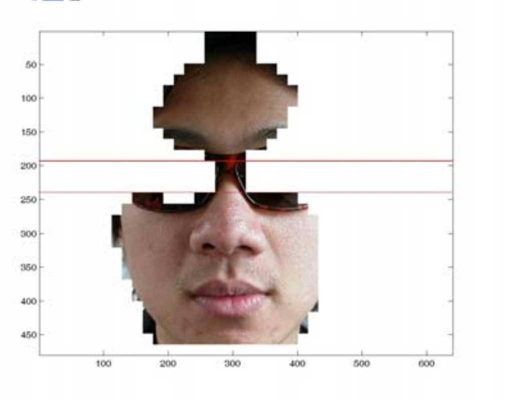
(a)



(b)



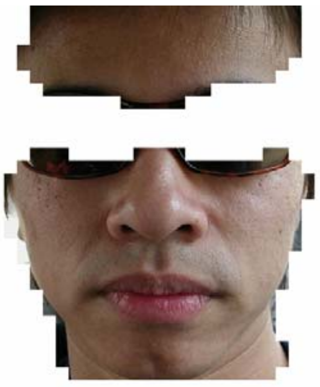
(c)



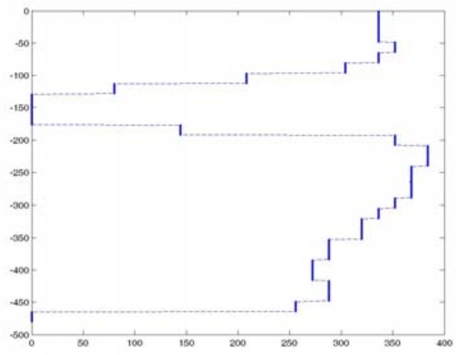
(d)

Fig. 4.10. The example 1 of location on face wearing dark sunglasses. (a) The input face image with sunglasses. (b) The face segment derived from (a). (c) The 1-D graph derived via summing the row-wise intensity of face segment of (b). (d) Locating the eye location at (b).

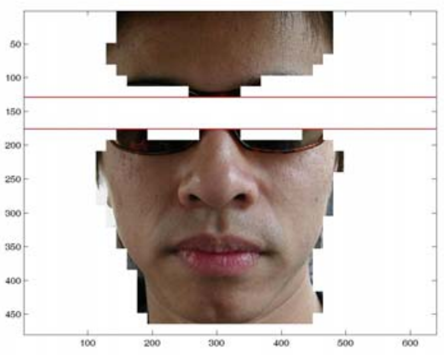
(a)



(b)



(c)



(d)

Fig. 4.11. The example 2 of location on face wearing dark sunglasses. (a) The input face image with sunglasses. (b) The face segment derived from (a). (c) The 1-D graph derived via summing the row-wise intensity of face segment of (b). (d) Locating the eye location at (b).

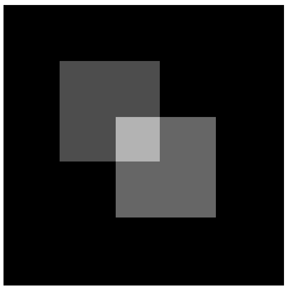
4.2 Reflection Separation

In Section 4.2.1, we show two example of one dimensional reflection separation with cost value to confirm that the cost function Eqs. (3.20) and (3.21) works well. In Section 4.2.2, we first show three examples of reflection separation by discretization to interpret the important of feature intensity between the reflection layer and the foreground layer. Then we show the result of four examples with different reflection straightly.

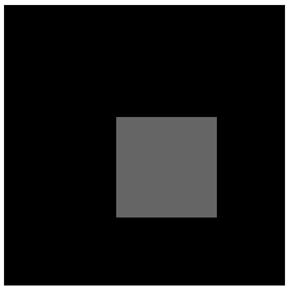
4.2.1 One Dimensional Reflection separation

In this section, we demonstrate a one dimensional family of solutions according to [19]. First, we define $s(x,y)$, shown in Fig. 4.12(b), which is the image of the "correct" decomposition for Fig. 4.12(a). We consider decompositions of the expression $I_1 = \gamma s(x,y)$, $I_2 = I - I_1$ and evaluated the cost for different values of r by Eq. (3.19). Fig. 4.12 indeed shows the minimum in this one dimensional subspace of solution is obtained at the "correct" solution.

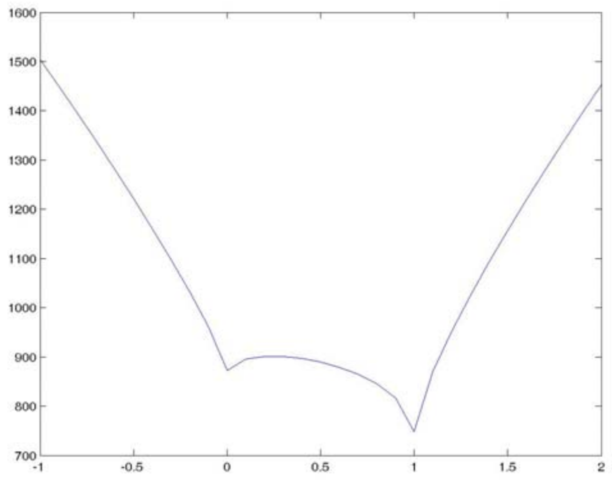
However, when we apply the cost function Eq. (3.19) at real image, it does not favor the "correct" decomposition; the reason is the "spurious" response as mention at Section 3.3.2. We should apply the refine cost function Eq. (3.21) to natural image and we can see the "correct" decomposition is indeed favored in the one dimension subspace (the minimum is obtained at γ=1) at Fig. 4.13.



(a)

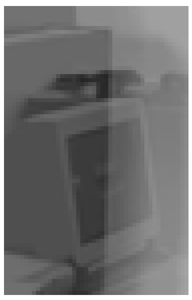


(b)

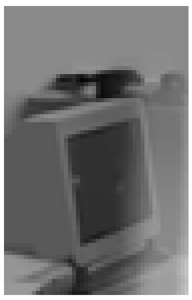


(c)

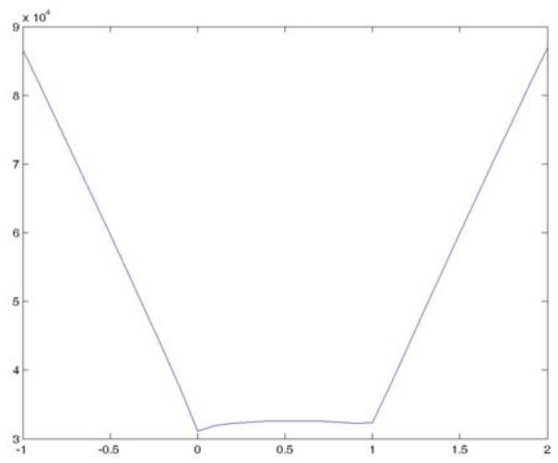
Fig. 4.12. Example 1 for testing an one dimension family of solutions. (a) Original image I. (b) The correct foreground layers. (c) The cost value of one dimension family of solutions for (b).



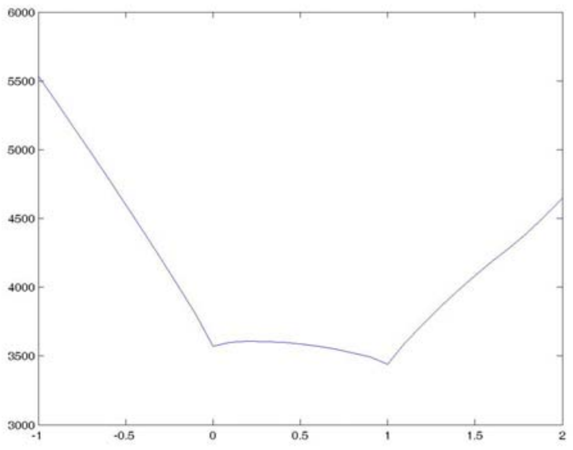
(a)



(b)



(c)



(d)

Fig. 4.13. Example 2 for one dimension family of solutions. (a) Reflection images. (b) The correct foreground layers. (c) The cost value of one dimension family of solutions for (b), calculated by (3.19). (d) The cost value of one dimension family of solutions for (b) , calculated by (3.21).

4.2.2 Reflection Separation by Discretization

In Figs. 4.14–4.16, we show three examples of reflection separation by discretization to interpret the important of feature intensity between the reflection layer and the foreground layer. Then we show the result of four examples with different reflection straightly in Fig. 4.17.

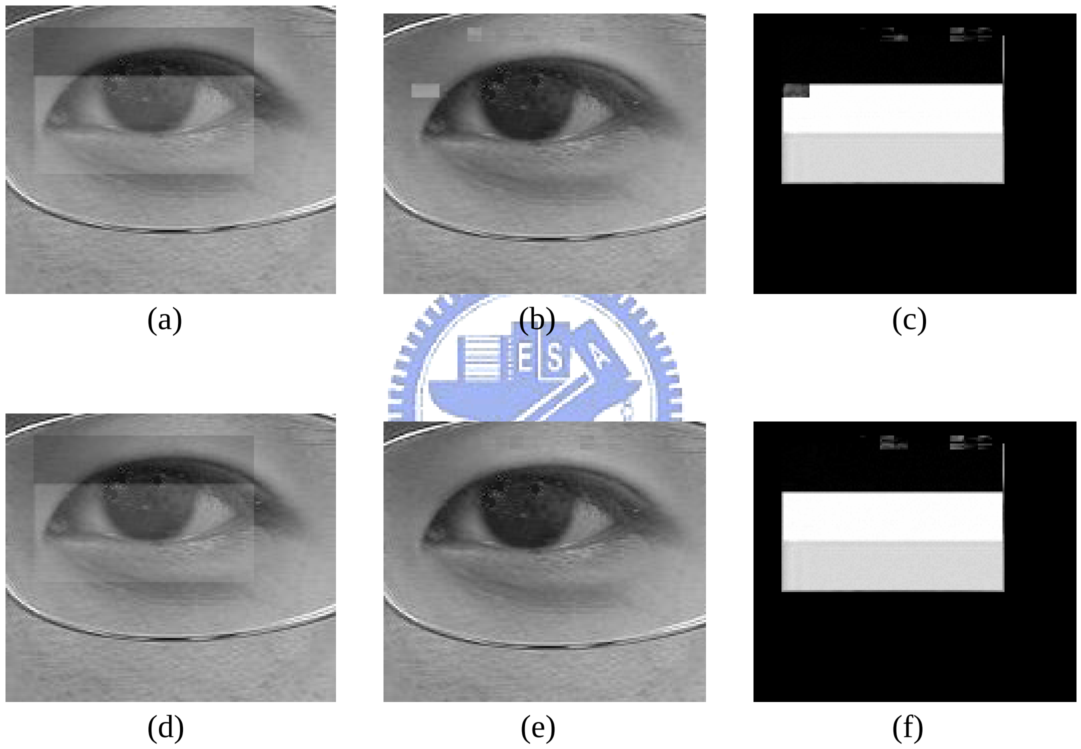


Fig. 4.14. Example 1 of separation results of images with reflections using discretization. (a) The input image consisting of the foreground image multiplied by 0.8 and the reflection image multiplied by 0.2. (b) Separated foreground layer image of (a). (c) Separated reflection layer image of (a). (d) The input image consisting of the foreground image multiplied by 0.85 and the reflection image multiplied by 0.15. (e) Separated foreground layer image of (d). (f) Separated reflection layer image of (d).

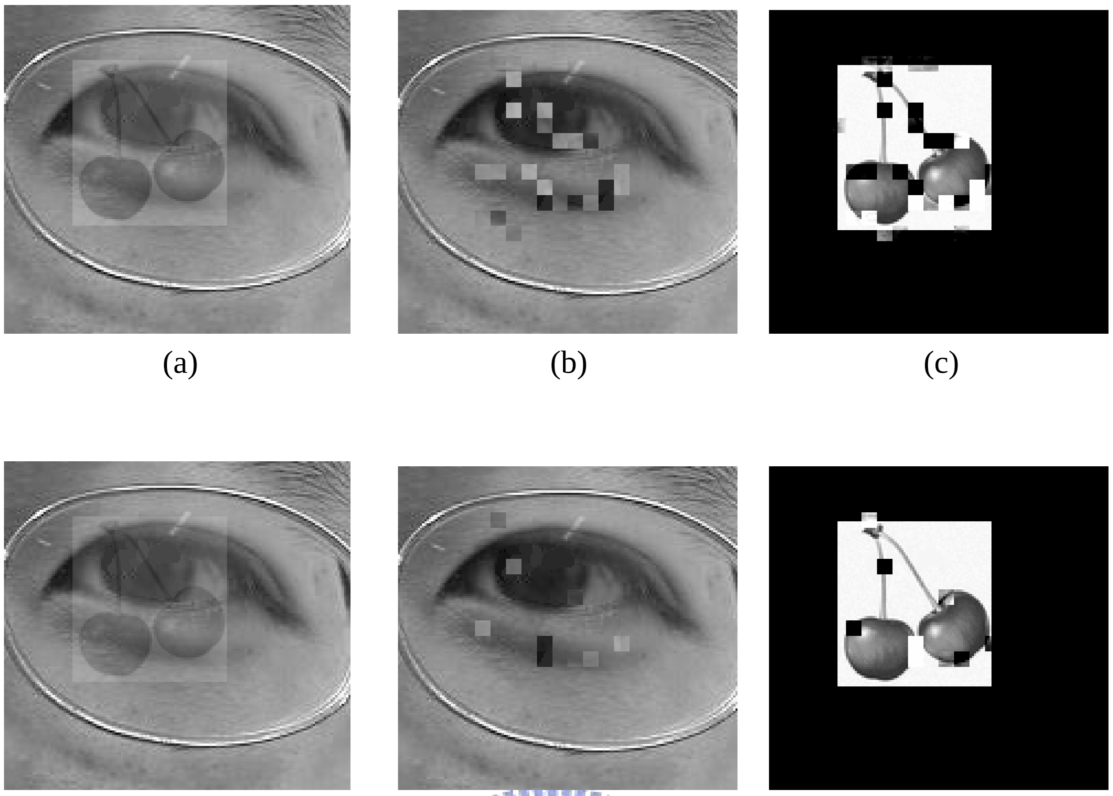


Fig. 4.15. Example 2 of separation results of images with reflections using discretization. (a) The input image consisting of the foreground image multiplied by 0.8 and the reflection image multiplied by 0.2. (b) Separated foreground layer image of (a). (c) Separated reflection layer image of (a). (d) The input image consisting of the foreground image multiplied by 0.85 and the reflection image multiplied by 0.15. (e) Separated foreground layer image of (d). (f) Separated reflection layer image of (d).

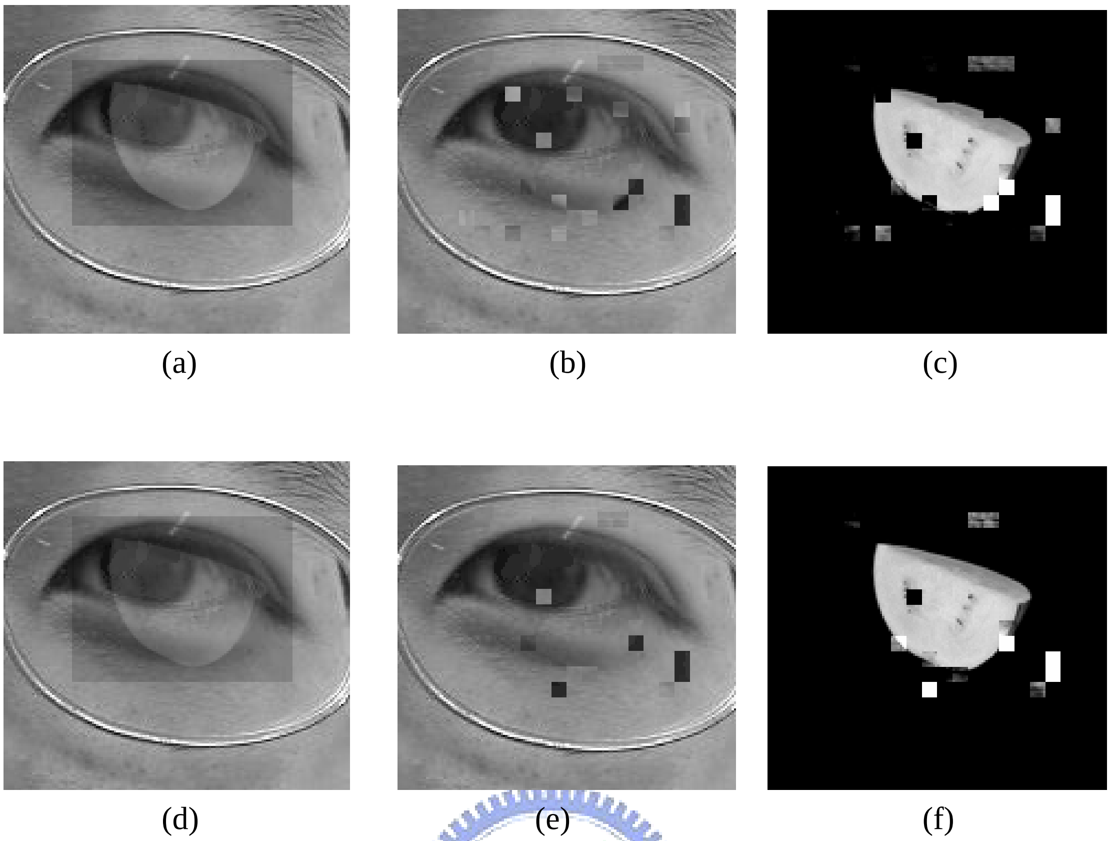


Fig. 4.16. Example 3 of separation results of images with reflections using discretization. (a) The input image consisting of the foreground image multiplied by 0.8 and the reflection image multiplied by 0.2. (b) Separated foreground layer image of (a). (c) Separated reflection layer image of (a). (d) The input image consisting of the foreground image multiplied by 0.85 and the reflection image multiplied by 0.15. (e) Separated foreground layer image of (d). (f) Separated reflection layer image of (d).

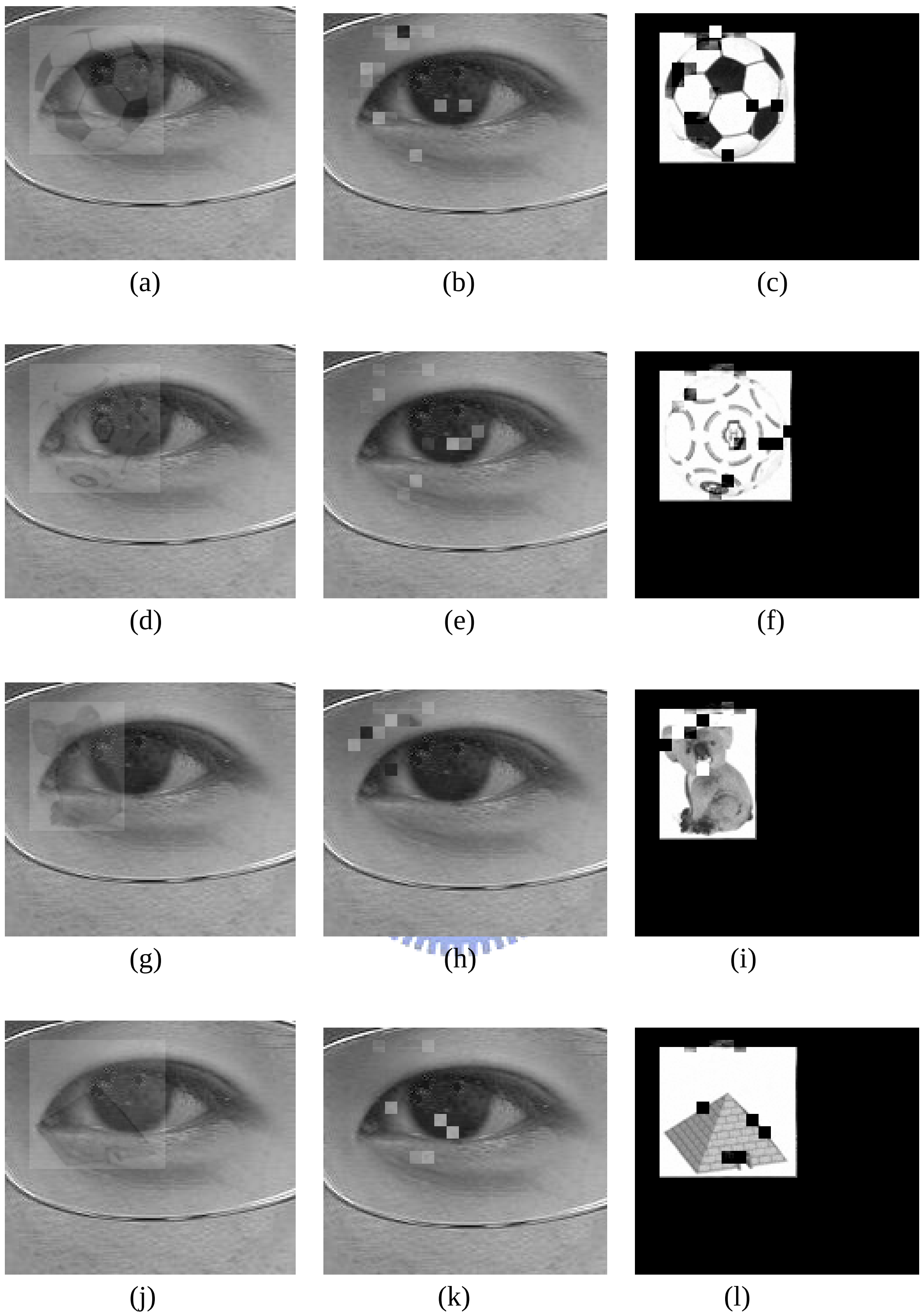


Fig. 4.17. Example of separation results of images with different reflections using discretization. (a), (d), (g), and (j) The input image consisting of the foreground image multiplied by 0.85 and the reflection image multiplied by 0.15. (b), (e), (h), and (k) Separated foreground layer images for (a), (d), (g), and (j), respectively. (c), (f), (i), and (l) Separated reflection layer images for (a), (d), (g), and (j), respectively.

Chapter 5. Conclusion

In this thesis, we develop eye detection algorithm to locate the eye region. We also introduce reflection separation algorithm to reduce the side effect of reflection arisen from glasses or sunglasses. In the relevant applications we are concerned about drowsiness detection system that provides an early detection and warning of fatigue at the wheel. Our eye detection algorithm can provide high eye accurate location; especially it is applicable to the case when a subject wears glasses or sunglasses. Our proposed reflection separation algorithm can retrieve and recover the eye information behind glasses.

In our eye detection system, we first utilize the universal skin-color map to extract the face region. Then we use corner operator, edge operator, and anisotropic diffusion to locate the eye region, detect the existence of glasses and separate reflection. The anisotropic diffusion plays an important role at corner and edge finding stage above because it can reduce noise and reserve the feature at same time.

For future work, we shall increase the number of patches in patches database. Besides, in order to optimize the result of recover image, we should not only pick up the correct patch based on the error of Eq. (3.24), but also take the neighbor cost into account value will be. The optimization procedure could be improved by the techniques such as belief propagation (BP), max-product BP, and graph cuts.

References

[1] C. H. Chang, “Drowsiness detection using fuzzy integral based information fusion,” Master Thesis, Department of Electrical and Control Engineering, National Chiao Tung University, Taiwan, June 2005.

[2] E. Hjelmas and B. K. Low, “Face detection: a survey,” *Computer Vision and Image Understanding*, vol. 83, pp. 236–274, 2001.

[3] D. Chai and K. N. Ngan, “Face segmentation using skin-color map in videophone applications,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, pp. 551–564, 1999.

[4] H. Wu, Q. Chen, and M. Yachida, “Face detection from color images using a fuzzy pattern matching method,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, pp. 557–563, 1999.

[5] J. Yang and A. Waibel, “A real-time face tracker,” in *Proc. 3rd IEEE Workshop on Application of Computer Vision*, 1996, pp. 142–147.

[6] R. Féraud, O. J. Bernier, J. E. Viallet, and M. Collobert, “A fast and accurate face detection based on neural network,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, pp. 42–53, 2001.

[7] D. Maio and D. Maltoni, “Real-time face location on gray-scale static images,” *Pattern Recognition*, vol. 33, pp. 1525–1539, 2000.

[8] C. Garcia and G. Tziritas, “Face detection using quantized skin color regions merging and wavelet packet analysis,” *IEEE Trans. Multimedia*, vol. 1, pp. 264–27, 1999.

[9] K. K. Sung and T. Poggio, “Example-based learning for view-based human face detection,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 39–51, 1998.

[10] K. C. Yow and R. Cipolla, “Feature-based human face detection,” *Image and Vision Computing*, vol. 15, pp. 713–735, 1997.

[11] S. A. Sirohey and A. Rosenfeld, “Eye detection in a face image using linear and nonlinear filters,” *Pattern Recognition*, vol. 34, pp. 1367–1391, 2001.

[12] G. C. Feng and P. C. Yuen, “Multi-cues eye detection on gray intensity image,” *Pattern Recognition*, vol. 34, pp. 1033–1046, 2001.

[13] R. T. Kumar, S. K. Raja, and A. G. Ramakrishnan, “Eye detection using color cues and projection functions,” in *Proc. IEEE Int. Conf. Image Processing*, 2002, vol. 3, pp. 24–28.

[14] Z. Liu, X. He, J. Zhou, and G. Xiong, “A novel method for eye region detection in gray-level image,” in *Proc. IEEE Int. Conf. Communications, Circuits and Systems and West Sino Expositions*, 2002, vol. 2, pp. 1118–1121.

[15] S. Kawato and J. Ohya, “ Two-step approach for real-time eye tracking with a new filtering technique,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2000, vol. 2, pp. 1366–1371.

[16] A. Levin and Y. Weiss, “User assisted separation of reflections from a single Image using a sparsity prior,” in *Proc. of the European Conference on Computer Vision (ECCV), Prague, May 2004.*

[17] M. Irani and S. Peleg, “Image sequence enhancement using multiple motions analysis,” in *Proc. IEEE Conf. Comput. Vision Pattern Recog., Champaign, Illinois*, 1992, pp. 216–221.

[18] R. Szeliksi, S. Avidan, and P. Anandan, “Layer extraction from multiple images containing reflections and transparency,” in *Proceedings IEEE CVPR*, 2000.

[19] A. Levin, A. Zomet, and Y. Weiss, “Separating reflections from a single image using local features,” *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2004, Washington DC

[20] D. C. Marr and E. Hildreth, “Theory of edge detection,” *Proc. Roy. Soc. London.*, vol. B 207, pp. 187-217, 1980.

[21] A. Rosenfeld and M. Thurston, “Edge and curve detection for visual scene analysis,” *IEEE Trans. Comput.*, vol. C-20, no. 5, pp. 562-569, 1971.

[22] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.

[23] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proc. 4th Alvey Vision Conference*, pp. 147–151, 1988.

[24] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger. “Robust anisotropic diffusion,” *IEEE Trans. on Image Processing*, vol. 7, no. 3, MARCH 1998.

[25] P. Perona and J. Malik, “Scale space and edge detection using anisotropic diffusion,” *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 12(7):629–639, July 1990.

[26] J. Fan, D. K. Y. Yau, A. K. Elmagarmid, and W. G. Aref, “Automatic image segmentation by integrating color-edge extraction and seeded region growing,” *IEEE Trans. Image Processing*, vol. 10, pp. 1454–1466, 2001.

[27] J. Fan, R. Wang, L. Zhang, D. Xing, and F. Gan, “Image sequence segmentation based on 2-D temporal entropy,” *Pattern Recognition Lett.*, vol. 17, pp. 1101–1107, 1996.

[28] A. Levin, A. Zomet, and Y. Weiss, “Learning to perceive transparency from the statistics of natural scenes,” in S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, 2002.

[29] J. Malik, S. Belongie, T. Leung, and J. Shi, “Contour and texture analysis for image segmentation,” in K.L. Boyer and S. Sarkar, editors, *Perceptual Organization for artificial vision systems*. Kluwer Academic, 2000.

[30] J.S. Yedidia, W.T. Freeman, and Y. Weiss, “Constructing free energy approximations and generalized belief propagation algorithms,” MERL Technical Report TR2002-35, 2002, available online at http://www.merl.com/papers/TR2002-35/.