國　立　交　通　大　學

電 機 與 控 制 工 程 研 究 所

碩 士 論 文

利用熵選取門檻之自動彩色邊緣偵測

Automatic Color Edge Detection by Entropic Thresholding

研　究　生：江宇洋

指 導 教 授：張 志 永

中 華 民 國 九十八 年 七 月

利用熵選取門檻之自動彩色邊緣偵測

Automatic Color Edge Detection by Entropic Thresholding

學　　生：江宇洋　　Student : Yu-Yang Jiang

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

June 2009

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 八 年 七 月

利用熵選取門檻之自動彩色邊緣偵測

學生：江宇洋　　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

由於邊緣偵測被廣泛應用在許多不同的影像處理上，例如：影像分割、物體辨識追蹤、立體分析等；這些影像處理任務的效能受到邊緣偵測結果好壞的巨大影響，所以邊緣偵測是個重要且不可忽視的基礎影像處理技術。為了得到更真實的邊緣，彩色邊緣偵測已經受到越來越多的重視。過去影像處理著重黑白影像邊緣偵測，不過灰階影像偵測邊緣時，往往不能偵測出具有相近灰階值但不同色彩的邊緣，同時也因為人類的視覺能區分出數千個不同的顏色卻只能區分出大約二十種的灰階，所以灰階影像失去許多彩色影像的邊緣資訊;近年來有越來越多關於彩色邊緣偵測的研究，不過這些彩色邊緣的研究的結果也只達到一定程度的效果，所以我們希望能夠提供比較有效的彩色邊緣偵測的方法。

本論文，我們提出基於向量階層統計與主要成分分析的彩色邊緣偵測技術，並且利用熵達到自動選取門檻的自動彩色邊緣偵測。利用我們提出的自動彩色邊緣偵測，不僅可以偵測到當相鄰物體具有相近的灰階值但不同色彩的邊緣，且門檻是依據影像內容所自動最佳調整，而不需要手動選取，增加使用者的方便性與信賴度。

Automatic Color Edge Detection by Entropic Thresholding

STUDENT: Yu-Yang Jiang ADVISOR: Dr. Jyh-Yeong Chang

Institute of Electrical and Control Engineering
National Chiao-Tung University

ABSTRACT

Edge detection is an important process in low level image processing because of its wide use in various tasks, including segmentation, object recognition, tracking, stereo analysis, image coding and many others. The performance of various subsequent image or video processing tasks is therefore greatly affected by the goodness of edge detection. To obtain the genuine edges, there has been an increased interest in color edge detection. Humans can differentiate thousands of colors compared to about two dozen shades of gray; hence, grayscale images do not carry all the edge information that human visual system (HVS) can detect.

In this thesis, we propose automatic color edge detection techniques based on vector order statistics and principal component analysis by entropic thresholding. Both methods employed improved entropic thresholding to determine the edge threshold. Our color edge detection techniques can detect edges when neighboring objects have different hues but with similar intensities, which cannot be detected by known grayscale or color edge detectors. Furthermore, by using entropic thresholding we can automatically determine an optimal threshold which is adaptive to different image contents without manual intervention. Edge detection by our proposed scheme is very user friendly and confident.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Jyh-Yeong Chang for valuable suggestions, guidance, support and inspiration he provided. Without his advice, it is impossible to complete this research. Thanks are also given to all of my lab members for their suggestion and discussion.

Finally, I would like to express my deepest gratitude to my family for their concern, supports and encouragements.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Motivation

Edge detection is an important process in low level image processing because of its wide use in several tasks such as segmentation, object recognition, tracking, stereo analysis, and image coding. The performance of these tasks is therefore tremendously affected by the goodness of edge detection. Conventionally, edge detectors use luminance component and locate changes in the intensity function. Pursuit of good edge detection algorithm led to such grayscale edge detectors as Canny, Cumani, and Compass [1]–[3]. Edges will not be detected in grayscale images when neighboring objects have different hues but equal intensities since the color cue is lost during grayscale conversion. Such objects cannot be distinguished in grayscale images. They are treated like one big object in the scene. This is not significant if the obstacle avoidance is the task of the vision system. Opposed to this, the capability of distinguishing between one big object and two (or several) objects may become crucial for the task of object grasping or even in 2-D image segmentation. Additionally, edge detection is sometimes difficult in low contrast images but rather sufficient results can be obtained in color images.

To obtain more meaningful edges, there has been an increased interest in color edge detection. Humans can differentiate thousands of colors compared to about two dozen shades of gray; hence, grayscale images do not carry all the edge information that human visual system (HVS) can detect. In [4], it is stated that luminance component makes up 90% of all edge points in a color image but the remaining 10% can be crucial for subsequent techniques that rely on edges in an image; in some cases the additional information provided by color is of utmost importance.

Multi-dimensional nature of color makes it more challenging to detect edges in color images, and often increases the computational complexity threefold compared to gray scale edge detection. Hence, color edge detection algorithms accept from the beginning that all of the efforts are to find the remaining 10% of the edges. Importance of color edge detection also becomes more apparent in low contrast images [5].

In this thesis, we propose two different approaches to the problem of color edge detection. The first approach is based on vector order statistics [13]–[15], [17]. In this approach, we calculate the local maximum edge response for every pixel, and threshold the local maximum edge response adaptively to the image content. The second approach is based on principal component analysis [18], [19]. In this approach, we keep the low frequency part and the high frequency part of the image, and we utilize Sobel operator to each of three color components separately to find edges.

1.2 Color Edge Detection

Color edge detection techniques fall into two main categories. Techniques in the first group [6]–[10] calculate gradients in each color component separately, and either fuses the gradients immediately or detect edges in each component separately before fusing to detect color edges. Techniques in the second group [2], [3], [11]–[15] treat each pixel as a tree-tuple vector and apply vector processing techniques without decoupling color components to obtain the edge map. A comprehensive analysis of color edge detectors can be found in [5], [16].

There is no universally accepted “color edge” definition. Literatures in this field suggest the following three definitions: (1) an edge exists if there is an edge in the corresponding grayscale image, (2) an edges exists if at least one of the color

components has an edge, (3) an edge exists if some norm (generally L_1, L_2, or L_∞) of the gradient from each color component exceeds a threshold value.

1.3 Automatic Thresholding Technique

In practice, edge detection is often done in an *ad-hoc* manner, frequently requiring user tuning of parameters. To enable the building of robust machine vision systems, it would be preferable to automate the edge thresholding process which is adaptive to different image contents without manual intervention.

The thresholding of edge detection involves the basic assumption that edges and non-edges pixels in the digital image have distinct edge response distributions. In [9], a *fast entropic thresholding technique* is used and shown to be highly efficient for the two-class classification problem. In [22], Joharmsen and Bille proposed a method using the entropy of the gray-level histrogram. This method divides the set of gray levels into two parts so as to minimize the interdependence (in information theoretic sense) between them. In [23], Wong and Sahoo proposed a thresholding method based on maximum entropy principle. The optimal threshold value is determined by maximizing the *a posteriori* entropy subject to certain inequality constraints which are derived by means of special measures characterizing uniformity and the shape of the regions in the image. For this purpose, the authors use both the gray-level distribution and the spatial information of an image.

When applying entropy to the thresholding of edge detection, the problem is that most pixels in an image are not edge pixels causing an erroneous bias to entropic thresholding. Therefore, we introduce a simple method to alleviate this bias to facilitate our automatic thresholding technique based on maximizing entropy.

1.4 Thesis Outline

The thesis is organized as follows. Before introducing the technique of our edge detection and entropic thresholding techniques, the basic concepts concerning the vector order statistics and principal component analysis are introduced in Chapter 2. Chapter 3 describes in details our vector-order statistics based and principal component analysis based edge detection techniques to calculate the edge response. Also, we describe in details our automatic thresholding techniques for these two detection techniques in Chapter 3. In Chapter 4, the experiment results of our automatic color edge detection techniques are shown and compared. At last, we conclude this thesis with a discussion in Chapter 5.

Chapter 2 Introduction to Vector Order Statistics and Principal Component Analysis

In this chapter, we briefly explain the basic concepts of vector order statistics and principal component analysis.

2.1 Vector Order Statistics

2.1.1 Vector Order Statistics Review

Scalar order statistics have played an important role in the design of robust signal analysis techniques. This is due to the fact that any outliers will be located in the extreme ranks in the sorted data. Consequently, these outliers can be isolated and filtered out before the signal is further processed. Ordering of univariate data is well defined and has been extensively studied [20]. Let the n random variables X_i, i = 1, 2, …, n, be arranged in ascending order of magnitude as

$$X_{(1)} \le X_{(2)} \le ... \le X_{(n)} \qquad (1)$$

Then the ith random variable $X_{(i)}$ is the so-called ith order statistic. The minimum $X_{(1)}$, the maximum $X_{(n)}$, and the median $X_{(n/2)}$ are among the most important order statistics, resulting the min, the max, and the median filters, respectively.

The concepts are, however, not straightforwardly expanded to multivariate data since there is not any universal way of defining an ordering in multivariate data. There has been a number of ways proposed to perform multivariate data ordering that

are generally classified into [17]: marginal ordering (M-ordering), reduced or aggregate ordering (R-ordering), partial ordering (P-ordering), and conditional ordering (C-ordering).

2.1.2 Characteristics of Vector Order Statistics

Let X represent a p-dimensional multivariate $X = [X_1, X_2, ..., X_p]^T$ where X_l, $l = 1, 2, \ldots, p$ are random variables and let X^i, $i = 1, 2, \ldots, n$ be an observation of X. Each X^i is a p-dimensional vector $X^i = [X_1^i, X_2^i, ..., X_p^i]^T$.

In M-ordering, the multivariate samples are ordered along each one of the p-dimensions independently. For color signals, this is equivalent to the separable method where each one of the colors is processed independently. The ith marginal order statistic is the vector $X^{(i)} = [X_1^{(i)}, X_2^{(i)}, ..., X_p^{(i)}]^T$, where $X_r^{(i)}$ is the ith largest element in the rth channel. The marginal order statistic $X^{(i)}$ may not correspond to any of the original samples $X^1, X^2, ..., X^n$ as it does in one dimension.

In R-ordering, each multivariate observation X^i is reduced to a scalar value d_i according to a distance criterion. A metric that is often used is the generalized distance to some point x. The samples are often arranged in ascending order of magnitude of the associated metric value d_i.

In P-ordering, the objective is to partition the data into groups or sets of samples, such that the groups can be distinguished with respect to order, rank, or extremeness. This type of ordering can be accomplished by using the notion of convex hulls. However, the determination of the convex hull is difficult to do in more than two dimensions. Other ways to achieve P-ordering are ad hoc partitioning procedures and

thus are not preferred. Another drawback associated with P-ordering is that there is no ordering within the groups and thus it is not easily expressed in analytical terms. These properties make P-ordering infeasible for implementation in digital image processing.

In C-ordering, the multivariate samples are ordered conditionally on one of the marginal sets of observations. This has the disadvantage in digital image processing that only the information in one component (channel) is used.

From the above, it is evident that R-ordering is more appropriate for color image processing than the other vector ordering methods. If we employ as a distance metric the aggregate distance of X^i to the set of vectors $X^1, X^2, ..., X^n$, then

$$d_i = \sum_{k=1}^{n} \| X^i - X^k \|, \quad i = 1, 2, ..., n \tag{2}$$

where $\| \cdot \|$ represents an appropriate vector norm. The arrangement of the d_is in ascending order $\left(d_{(1)} \le d_{(2)} \le ... \le d_{(n)}\right)$, associates the same ordering to the multivariate X^is.

$$X^{(1)} \le X^{(2)} \le ... \le X^{(n)} \tag{3}$$

In the ordered sequence, $X^{(1)}$ is the vector median of the data samples [21]. It is defined as the vector contained in the given set whose distance to all other vectors is a minimum. Moreover, vectors appearing in low ranks in the ordered sequence are vectors centrally located in the population, whereas vectors appearing in high ranks are vectors that diverge mostly from the data population. These samples are generally called “outliers.” It follows that this ordering scheme gives a natural definition of the median of a population and of the outliers of a population.

2.2 Principal Component Analysis

2.2.1 Principal Component Analysis Review

Principal Component Analysis (PCA) is a useful statistical technique that has found application in fields such as face recognition and image compression, and is a common technique for finding patterns in data of high dimension. It is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analyzing data. The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data by reducing the number of dimensions without much loss of information. This technique used in the field of image compression.

PCA involves a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

PCA transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. PCA is theoretically the optimum transform for given data in least square terms.

In [18], the transformation that preserves chrominance edges and effectively reduces the dimensionality of color space from three to one for detecting color edges

is based on PCA method. This transformation generates monochrome images that carry the color edge information to facilitate the performances of conventional grayscale edge detectors.

In [19], in order to generate a set of eigenfaces, a large set of digitized images of human faces, taken under the same lighting conditions, are normalized to line up the eyes and mouths. They are then all re-sampled at the same pixel resolution. Eigenfaces is extracted by means of PCA. The eigenfaces that are created will appear as light and dark areas that are arranged in a specific pattern. This pattern is how different features of a face are singled out to be evaluated and scored.

2.2.2 Characteristics of Principal Components

The first component extracted in a principal component analysis accounts for a maximal amount of total variance in the observed variables. Under typical conditions, this means that the first component will be correlated with at least some of the observed variables. It may be correlated with many.

The second component extracted will have two important characteristics. First, this component will account for a maximal amount of variance in the data set that was not accounted for by the first component. Again under typical conditions, this means that the second component will be correlated with some of the observed variables that did not display strong correlations with component one.

The second characteristic of the second component is that it will be *uncorrelated* with the first component. Literally, if you were to compute the correlation between components 1 and 2, that correlation would be zero.

The remaining components that are extracted in the analysis display the same two characteristics: each component accounts for a maximal amount of variance in the

observed variables that was not accounted for by the preceding components, and is uncorrelated with all of the preceding components. A principal component analysis proceeds in this fashion, with each new component accounting for progressively smaller and smaller amounts of variance (this is why only the first few components are usually retained and interpreted). When the analysis is complete, the resulting components will display varying degrees of correlation with the observed variables, but are completely uncorrelated with one another.

Principal component analysis is sometimes confused with factor analysis, and this is understandable, because there are many important similarities between the two procedures: both are variable reduction methods that can be used to identify groups of observed variables that tend to hang together empirically. Both procedures can be performed with the SAS System's FACTOR procedure, and they sometimes even provide very similar results.

Nonetheless, there are some important conceptual differences between principal component analysis and factor analysis that should be understood at the outset. Perhaps the most important deals with the assumption of an underlying causal structure: factor analysis assumes that the co-variation in the observed variables is due to the presence of one or more latent variables (factors) that exert causal influence on these observed variables.

In contrast, principal component analysis makes no assumption about an underlying causal model. Principal component analysis is simply a variable reduction procedure that (typically) results in a relatively small number of components that account for most of the variance in a set of observed variables.

2.2.3 Principal Component Vectors Computation

Here, we demonstrate how to compute the principal components by a simple example.

Principal component analysis is based on the statistical representation of a random variable. Suppose we have a random vector population $X = \left(x_1, x_2, ..., x_n\right)^T$, and the mean of that population is denoted by $\mu_x = E\{X\}$. The covariance matrix of the same data set C_x is

$$C_x = E\left\{\left(X - \mu_x\right)\left(X - \mu_x\right)^T\right\} \tag{4}$$

The components of C_x, denoted by c_{ij}, represent the covariances between the random variable components x_i and x_j. The component c_{ii} is the variance of the component x_i. The variance of a component indicates the spread of the component values around its mean value. If two components x_i and x_j of the data are uncorrelated, their covariances c_{ij} and c_{ji} are zero. The covariance matrix is, by definition, always symmetric. From a symmetric matrix such as the covariance matrix, we can calculate an orthogonal basis by finding its eigenvalues and eigenvectors. The eigenvectors V_i and the corresponding eigenvalues λ_i are the solutions of the equation

$$C_x V_i = \lambda_i V_i, \quad i = 1, 2, ..., n \tag{5}$$

These values can be found, for example, by finding the solutions of the characteristic equation

$$\| C_x - \lambda_i I \| = 0, \quad i = 1, 2, ..., n \tag{6}$$

where I is the identity matrix having the same order of C_x and $\| \cdot \|$ denotes the determinant of the matrix. If the data vector has *n* components, the characteristic equation becomes of order *n*.

By ordering the eigenvectors in the order of descending eigenvalues (largest first), one can create an ordered orthogonal basis with the first eigenvector having the direction of the largest variance of the data. In this way, we can find directions in which the data set has the most significant amounts of energy.

Chapter 3 Color Edge Detection and Entropic Thresholding

3.1 Vector-Order Statistics based Color Edge detection

3.1.1 Confined Window Case

For a color image F of size $M \times N$, each pixel location (m,n) is represented by a three-tuple color vector $F(m,n) = (F_1(m,n), F_2(m,n), F_3(m,n))$, in which $F_i(m,n)$ denoting the i-th component of a color space, for $m = 1, 2, ..., M$ and $n = 1, 2, ..., N$. For each pixel location (m,n), by using a 3×3 window, we confine the pixel vectors in the window be the vectors $X^1, X^2, ..., X^9$. We employ the aggregate distance of X^i as a distance metric to the set of vectors $X^1, X^2, ..., X^9$, then

$$d_i = \sum_{k=1}^{9} \| X^i - X^k \|, \quad i = 1, 2, ..., 9 \tag{7}$$

where $\| \cdot \|$ represents a 2-norm. After we have computed the aggregate distances , we sort the distance values in ascending order $d_{(1)} \le d_{(2)} \le ... \le d_{(9)}$. $d_{(1)}$ and $d_{(9)}$ correspond to the minimum and the maximum of the nine distance values respectively.

By the concept of R-ordering, the ordering of $d_{(1)} \le d_{(2)} \le ... \le d_{(9)}$ associates the same ordering to the pixel vectors, $X^{(1)} \le X^{(2)} \le ... \le X^{(9)}$, which means that

$X^{(1)}$ is the vector median of vectors in the window and $X^{(9)}$ is the outlier of vectors in the window. Although we now obtain the information on the vector median and the outlier, the information contained among vectors $X^{(1)}, X^{(2)}, ..., X^{(9)}$ should also be captured and be useful for edge detection. The maximal variation among vectors is an indication of the distribution of the nine vectors. Since that vectors $X^{(1)}, X^{(2)}, ..., X^{(9)}$ correspond to the ordering of the aggregate distances, the confined maximal variation $\mathrm{MV_c}$ among these vectors can be simply defined as

$$MV_c = \max\left(\left\|X^{(i)} - X^{i+1}\right\|\right), \quad i = 1, 2, ..., 8 \tag{8}$$

When the value $\mathrm{MV_c}$ is determined, we can also determine the exact two vectors $X^{(i)}$ and $X^{(i+1)}$ which correspond to $\mathrm{MV_c}$. $X^{(i)}$ and $X^{(i+1)}$ further suggest that $X^{(1)}, X^{(2)}, ..., X^{(9)}$ can be classified into two clusters: (1) vectors, $X^{(1)}, X^{(2)}, ..., X^{(i)}$, from larger side of the edge, and (2) vectors, $X^{(i+1)}, X^{(i+2)}, ..., X^{(9)}$, from smaller side of the edge. Let M_s and M_l be the mean vector of the vectors $X^{(1)}, X^{(2)}, ..., X^{(i)}$, and the vectors $X^{(i+1)}, X^{(i+2)}, ..., X^{(9)}$ respectively. An edge detector can defined as a *Vector Mean Distance* (VMD) edge detector

$$VMD = \left\| M_l - M_s \right\| \tag{9}$$

VMD detect the variation between two sides of edge (larger and smaller side) by a distance measure. Consequently, in a uniform area, where all vector values are close to each other, the output of VMD will be small. On the other hand, the output of VMD will be large since M_s and M_l are the mean vectors of two sides of the edge. By thresholding the output of VMD, the actual edge can be obtained. Since the distribution of vectors $X^{(1)}, X^{(2)}, ..., X^{(9)}$ is an important source of information for

calculating the variation between two sides of an edge, a more discreet method for capturing the distribution of the vectors should be applied.

To capture the distribution of the vectors, we assume that vectors $X^{(1)}$, $X^{(2)}$, ..., $X^{(9)}$ can be classified into two clusters. Because that $X^{(9)}$ corresponds to the outlier of the vectors and could be affected, most probably one among the nine pixels in the window, by noise, $X^{(9)}$ should be discarded for a clearer analysis of the distribution of the vectors. We first take $X^{(1)}$ and $X^{(8)}$ as the representatives of two sides (larger side and smaller side) of an edge. Let d_{is} denote the distance between vector $X^{(i)}$ and $X^{(1)}$, and d_{il} denote the distance between $X^{(i)}$ and $X^{(8)}$. If $d_{is} < d_{il}$, $X^{(i)}$ belongs larger side of the edge. Otherwise, $X^{(i)}$ belongs to smaller side of the edge. After this procedure, we have a clear view of the distribution of vectors $X^{(1)}$, $X^{(2)}$, ..., $X^{(8)}$.

Since we know the distribution of vectors $X^{(1)}$, $X^{(2)}$, ..., $X^{(8)}$, the difference between two sides can reflect the variation of this area. We denote the mean vector of the smaller side of the edge as M_{ss} and the mean vector of the larger side of the edge as M_{ls}. Similar to VMD edge detector, an edge detector which capture the distribution of the vectors can be defined as a *Vector Distribution Difference* (VDD) edge detector

$$VDD = \| M_{ls} - M_{ss} \| \tag{10}$$

VDD detect the variation between two sides of an edge (larger and smaller side) in a similar way to VMD and capture the distribution of the vectors by a discreet method. Also by discarding the outlier $X^{(9)}$, VDD is more capable of alleviating the influence of noise.

From a different point of view, the aggregate distance d_i indicates how much a

vector X^i diverges from the other vectors in the window. If we modify the aggregate distance d_i to reflect the variation of every pixel vector to its corresponding neighboring area, we should be able to have a clear insight for the distribution of the vectors. This concept leads us to the following edge detection method.

3.1.2 Local Window Case

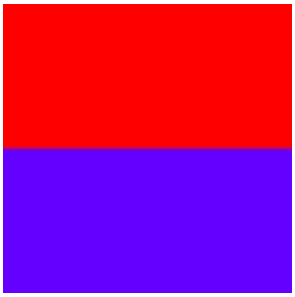
For each pixel location (m,n), by using a 3×3 window, we compute the local sum of distances to describe the relationship between the current pixel vector $F(m,n)$ and its neighboring pixel vectors. Let $d_l(m,n)$ be the local sum of distances for the current pixel vector $F(m,n)$, then

$$d_l(m,n) = \sum_{i=m-1}^{m+1} \sum_{j=n-1}^{n+1} \| F(m,n) - F(i,j) \| \tag{11}$$

where $\| \cdot \|$ represents a 2-norm. After we have computed the local sum of distances $d_l(m,n)$ of the current pixel location (m,n), we sort the distance values in the neighboring area in ascending order $d_{l(1)} \le d_{l(2)} \le ... \le d_{l(9)}$. The distance values $d_{l(1)}$ and $d_{l(9)}$ correspond to the minimum and the maximum of the nine distance values, respectively.

By the concept of R-ordering, the ordering of $d_{l(1)} \le d_{l(2)} \le ... \le d_{l(9)}$ associates the same ordering to the pixel vectors, $X_l^{(1)} \le X_l^{(2)} \le ... \le X_l^{(9)}$, which means that $X_l^{(1)}$ is the pixel vector having the smallest local sum of distances and

$X_l^{(9)}$ is the pixel vector having the largest local sum of distances. If the current pixel location (m,n) has an edge, the vector $F(m,n)$ must have a larger response of $d(m,n)$. Therefore, the subsequent edge detection process executes when the vector $F(m,n)$ belongs to one of the vectors $X_l^{(4)}$ – $X_l^{(9)}$. Fig. 3.1 illustrates a simple edge example of the orderings of vectors based on the aggregate distances and the local sum of distances by the R-component only. Notice that the pixel in the center of the window has a smaller response of the aggregate distance in Fig. 3.1(c) denoted as $X^{(5)}$ in Fig. 3.1(d), and has a larger response of $d_l(m,n)$ in Fig. 3.1(e) denoted as $X_l^{(5)}$ in Fig. 3.1(f).



(a)

255	255	255
255	255	255
100	100	100

(b)

465	465	465
465	465	465
930	930	930

(c)

$X^{(1)}$	$X^{(2)}$	$X^{(3)}$
$X^{(4)}$	$X^{(5)}$	$X^{(6)}$
$X^{(7)}$	$X^{(8)}$	$X^{(9)}$

(d)

0	0	0
465	465	465
465	465	465

(e)

$X_l^{(1)}$	$X_l^{(2)}$	$X_l^{(3)}$
$X_l^{(4)}$	$X_l^{(5)}$	$X_l^{(6)}$
$X_l^{(7)}$	$X_l^{(8)}$	$X_l^{(9)}$

(f)

Fig. 3.1. A simple edge example. (a) single edge image, (b) R-component values in edge area, (c) the aggregate distances, and (d) sorted pixels vectors corresponding to the same ordering of the aggregate distances, (e) the local sum of distances, and (f) Sorted pixels vectors corresponding to the same ordering of the local sum of distances.

Next, the edge detector *Maximal Delta* (MD) is defined as

$$MD = \max\left(\left\| F(m,n) - X_l^{(j)} \right\|\right), \quad 4 \le j \le 9 \tag{12}$$

Because that the ordered vectors $X_l^{(4)}, X_l^{(5)}, ..., X_l^{(9)}$ denote the larger six sum of

distances $d_l(i,j)$, the positions of these ordered vectors could also be edge candidates. Because that an edge is two pixels wide in nature, it suggests that we can find the maximum variation existent in the two pixels of edge bank by utilizing MD edge detector.

3.2 PCA based Color Edge Detection

3.2.1 YC_bC_r Color Space

Before we introduce our PCA based color edge detection, we now briefly introduce YC_bC_r color space.

The color space of YC_bC_r can divided into luminance component (Y), and two chromatic blueness component (C_b), redness component (C_r). C_b and C_r are the difference of blue and the difference of red chromatic components. The following conversion matrix is used to convert RGB color space into YC_bC_r color space.

$$\begin{cases} Y = 0.2989 \times R + 0.5866 \times G + 0.1145 \times B \\ C_b = 0.5674 \times (B - Y) \\ C_r = 0.7132 \times (R - Y) \end{cases} \quad (13)$$

For a color edge detector, the goal is to detect edges that have changes in luminance or in chrominance. Our PCA based edge detection involves the reconstruction of the input image. We have to retain both chrominance and luminance information via the essential principal components of input image. Therefore, YC_bC_r color space is selected in our PCA based color edge detection theme.

3.2.2 Eigencolor Image Construction

For an input image F of size $M \times N$, each pixel location (m,n) is represented by a three-tuple color space vector $F(m,n)$ for $m = 1,2,...,M$ and $n = 1,2,...,N$. Each component of the color space in pixel location (m,n) can be represented as $F_i(m,n)$, for $i = 1, 2, 3$. A color component of input image F is denoted as F_i. Then, for the color component F_i, the estimate of the mean is calculated as

$$\overline{M_i} = \frac{1}{M \times N} \sum_{m=1}^{M} \sum_{n=1}^{N} F_i(m,n) \tag{14}$$

For $i = 1, 2,$ and 3, $\overline{M_i}$ is expanded to a matrix $\overline{F_i}$ of size $M \times N$ having the value of $\overline{M_i}$ for every entry of matrix $\overline{F_i}$. The covariance matrix estimate for the color component is

$$C_i = \left(F_i - \overline{F_i}\right)\left(F_i - \overline{F_i}\right)^T \tag{15}$$

Next, we compute for matrix C_i's eigenvalues λ_{ij}, with V_{ij} being the corresponding eigenvectors such that $\lambda_{i1} \geq \lambda_{i2} \geq ... \geq \lambda_{ij}$, for $j = 1, 2, ..., L$, where $L = \min(M, N)$. We select the first k_i terms and the last k_i terms of eigenvectors v_{ij} to retain the low frequency and the high frequency contents of the color image F. To obtain the compressed image for edge detection, we have to keep not only the low frequency principal components but also the high frequency components for detail and edge preservation, which is quite different from the conventional image

data compression approach. These eigenvectors are arranged in a matrix $E_i = \left[V_{i1},\ V_{i2},\ ...,\ V_{ik_i},\ V_{i(L-k_i)},\ V_{i(L-k_i+1)},\ ...,\ V_{iL} \right]$.

After obtaining the eigenvectors, we are now constructing an image called "eigencolor" image. First, the matrix $F_i^{'}$ of the color component i is obtained by

$$F_i^{'} = F_i - \overline{F_i} \tag{16}$$

And the weighting coefficients matrix W_i of selected eigenvectors can be computed by

$$W_i = E_i^T F_i^{'} \tag{17}$$

The i-th color component is then constructed by

$$Eigen_i = E_i W_i + \overline{F_i} \tag{18}$$

The other two color components are also constructed in the same procedure.

Finally, the eigencolor image can be obtained by combining three constructed color components $Eigen_i$ with the corresponding dimension order.

3.2.3 Edge Response Calculation

To obtain the edge response of the compressed eigencolor image, we simply apply the Sobel operator, as shown in Fig. 2, to each color component of eigencolor image, $Eigen_i$, and calculate the magnitude of edge response for each pixel location (m,n) by

$$ME_i(m,n) = \sqrt{g_{h_i}(m,n)^2 + g_{v_i}(m,n)^2},\ m = 1,\ 2,\ ...,M,\ n = 1,\ 2,\ ...,N \qquad (19)$$

where $g_{h_i}(m,n)$ and $g_{v_i}(m,n)$ are respectively the horizontal and vertical responses of the Sobel operator, shown in Fig. 3.2, in pixel location (m,n) for the i-th component.

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

(a)

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

(b)

Fig. 3.2. Sobel operator. (a) horizontal derivative approximation g_{h_i}, (b) vertical derivative approximation g_{v_i}.

By using a threshold T_i, the E_i function classifies the ME_i pixels into two classes: *edge pixels* and *non-edge pixels*, as

$$E_i(m,n) = \begin{cases} 1 \text{ (edge pixel)}, & \text{if } ME_i(m,n) \geq T_i \\ 0 \text{ (nonedge pixel)}, & \text{if } ME_i(m,n) < T_i \end{cases} \qquad (20)$$

The edge image is obtained by a *majority vote fusion rule* on the detected edge of three color components. Let *count* denote the number that $E_i(m,n)$ classifies pixel (m,n) as an edge for $i = 1,\ 2,$ and 3. Namely pixel (m,n) is classified as an edge if it is classified as an edge by at least two of its three color components, shown in (20). Otherwise, it is classified as a non-edge pixel and $E(m,n)$ is set to 0.

$$E(m,n) = \begin{cases} 1 \text{ (edge pixel)}, & \text{if } count \geq 2 \\ 0 \text{ (nonedge pixel)}, & \text{otherwise} \end{cases} \quad (21)$$

3.3 Entropic Thresholding

To automatically obtain an optimal threshold that is adaptive to the image contents, the entropic thresholding technique is adopted. In the following, we illustrate how to apply entropic thresholding technique to the vector-order statistics based and the PCA based edge detections.

3.3.1 Entropic Thresholding for Vector-Order Statistics based Edge Detection

We are now introducing the entropic thresholding technique. Given a threshold, e.g., T, the probability distributions for the edge and non-edge pixel classes can be defined, respectively. As they are to be regarded as independent distributions, the probability for the non-edge pixels in a $S \times S$ window having S^2 pixels $P_n(i)$ can be defined as

$$P_n(i) = \frac{l_n}{S^2} \quad (22)$$

where l_n indicates the number of pixels in the window that have the edge responses smaller than or equal to threshold T. The probability for the edge pixels $P_e(i)$ can be defined as

$$P_e(i) = \frac{l_e}{S^2} \quad (23)$$

where l_e indicates the number of pixels in the window that have the edge responses greater than threshold T. The entropies for these two pixel classes are then given as

$$H_n(T) = -P_n(i)\log P_n(i) \tag{24}$$

$$H_e(T) = -P_e(i)\log P_e(i) \tag{25}$$

The local optimal threshold T_l^* selected for performing the non-edge and edge pixel classification has to satisfy the following criterion function:

$$H(T_l^*) = \max_{T=T_1, T_2, ..., T_L} \left(H_n(T) + H_e(T)\right) \tag{26}$$

where $T_1, T_2, ..., T_L$ are the proportional ratio constants with respect to the norm of the mean vector of two sides (larger and smaller side) of the edge. In this way, the local optimal threshold T_l^* reflects the relative magnitude variation with respect to the mean magnitude of current processing area. In our experiments, we select $T_1, T_2, ..., T_L$ to be 0.2, 0.3, ..., 0.9.

3.3.2 Entropic Thresholding for PCA based Edge Detection

After obtaining the magnitude of edge response E_i, Sec. 3.2.2, of each color component for pixels in the image, we introduce the entropic thresholding technique globally. We utilize all the Sobel edge response values of pixels in the input image as the possible choices of the global optimal threshold T_g^* for each components *i*. For the *i*-th component of the color space, let the edge responses of the pixels have the range $[0, h_n]$ (the possible choices of the global optimal threshold), the probability

for the non-edge pixels in a $M \times N$ image $P_n(i)$ can be defined as

$$P_n(i) = \frac{l_n}{M \times N}, \tag{27}$$

where l_n indicates the number of pixels in the input image that have the edge responses smaller than or equal to threshold T. The probability for the edge pixels $P_e(i)$ can be defined as

$$P_e(i) = \frac{l_e}{M \times N}, \tag{28}$$

where l_e indicates the number of pixels in the input image that have the edge responses greater than threshold T.

The entropies $H_n(T)$ and $H_e(T)$ for these two pixel classes can be computed according to Eqs. (23) and (24). The global optimal threshold T_g^* selected for performing the non-edge and edge pixel classification has to satisfy the following criterion function:

$$H(T_g^*) = \max_{T=0, h_1, ..., h_n} \left(H_n(T) + H_e(T) \right) \tag{29}$$

However, the global optimal threshold T_g^* will be too small and too sensitive for edge detection because that most pixels in the input image are non-edge pixels causing a bias leaning to the non-edge pixels' edge strength. From Eq. (28), it is clear that maximum entropy occurs when the numbers of pixels in edge class and non-edge class are equal. Since most pixels are non-edge pixels, which have small magnitudes of edge response, T_g^* naturally is tuned to a small value among the pixels' edge response, which results in an erroneous edge/non-edge threshold selection. Therefore, we need to discard the pixels with small magnitudes of edge response in order to

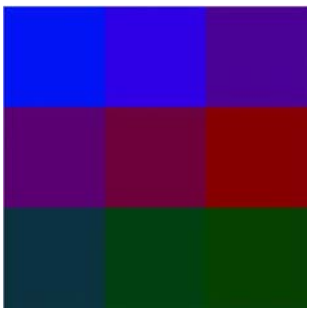
alleviate the bias caused from the usual uneven distribution between edge pixels and non-edge ones. To solve this, we introduce a sensitivity parameter γ, which is automatically calculated from the input image content. The pixels which have the magnitudes of edge response smaller than the sensitivity parameter γ are the *obvious non-edge* pixels and are discarded when applying our global entropic thresholding technique. On testing various images, we have found that the Sobel edge responses of obvious non-edge pixels usually have strengths smaller than five. To this end, we calculate the histograms of the edge response in the following ten intervals: $(0,\ 0.5]$, $(0.5,\ 1]$, ..., $(4.5,\ 5]$. The value of γ is chosen to be the larger boundary of the interval that has the largest drop in histogram value, which could be the boundary for the obvious non-edge pixels and *possible non-edge* pixels. This is because that the majority of pixels in an image belong to non-edge regions and these pixels have small yet compact magnitudes of edge response. Therefore, we can assume that the intervals with maximal drop in histogram values represent the possible boundary between obvious non-edge pixels and possible non-edge pixels and we set γ to be the right boundary value of this interval. For example, if the maximal drop in histogram values of pixels' edge strength occurs between $(2,\ 2.5]$ and $(2.5,\ 3]$, the value of γ is chosen as 2.5.

Chapter 4 Experimental Results

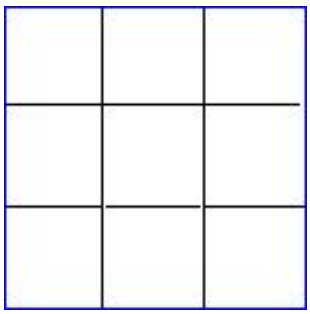
In chapter 4, we test our vector-order statistics based and PCA based edge detection techniques on synthetic images and real world images. The synthetic images include one image that contains nine different color blocks with the same intensity. These synthetic images are generated for assessing the performance comparison. For all the real world images are smoothed by the Gaussian filter with $\sigma = 1$ to alleviate the interference from noises. In the end of this chapter, we evaluate our edge detection techniques quantitatively by using Pratt's *Figure Of Merit* (FOM) [24] and TPR, TNR and ACC of *Receiver Operating Characteristic* (ROC) [25].

4.1 Vector-Order Statistics based Edge Detection

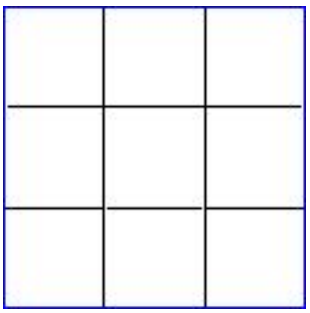
We utilize our vector-order statistics based edge detectors to compute edge response and entropic thresholding technique to determine a local optimal threshold T_l^*. The first image we test on is a image consisting of 9 different color blocks with the same intensity as shown in Fig. 4.1(a). This image is applied to verify that our edge detection techniques works when objects in image has the same intensity but different hues. Figs. 4.1(b)–(d) show the edge detection results obtained by VMD, VDD and MD, and proves that our edge detection techniques can work under this circumstance. Fig. 4.2 gives the corresponding histograms of local optimal threshold T_l^* of VMD, VDD and MD to Fig. 4.1(a), where the X-axis is the proportional ratio constants described in Sec. 3.1.1.



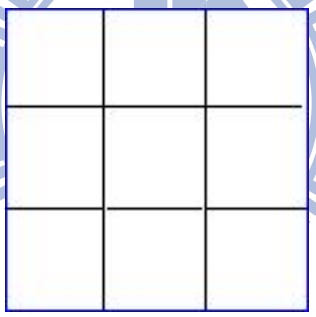
(a)



(b)

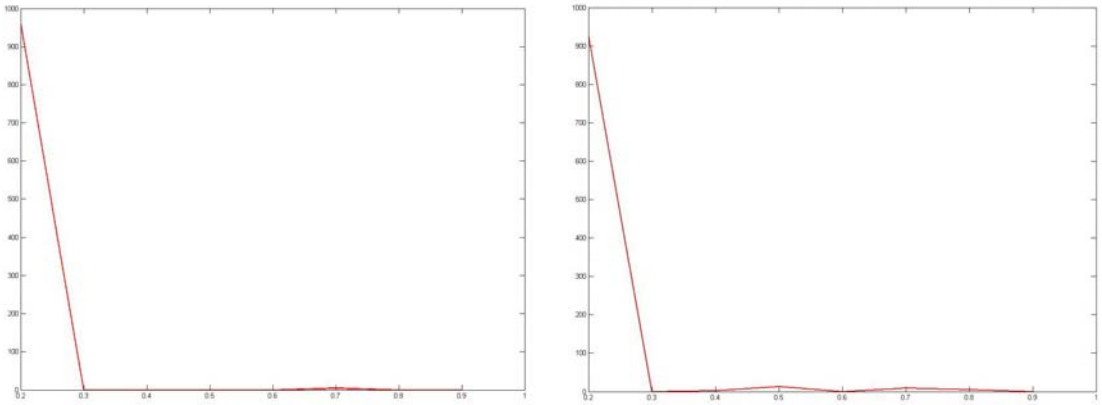


(c)

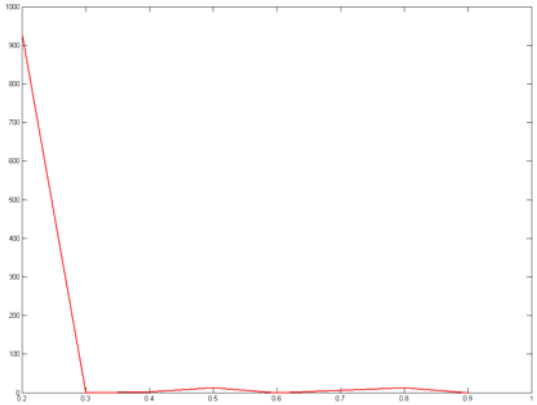


(d)

Fig. 4.1. Edge detection results of a color image with the same intensity. (a) original image, (b) edge detection by VMD, (c) edge detection by VDD, (d) edge detection by MD.



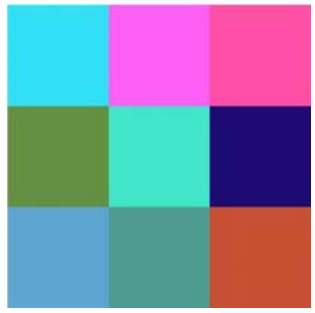
(a) (b)



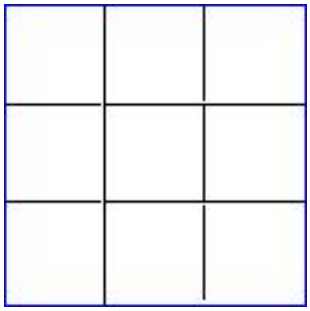
(c)

Fig. 4.2. The corresponding histograms of local optimal threshold T_l^* to Fig. 4.1(a). (a) histogram of T_l^* using VMD, (b) histogram of T_l^* using VDD, (c) histogram of T_l^* using MD.

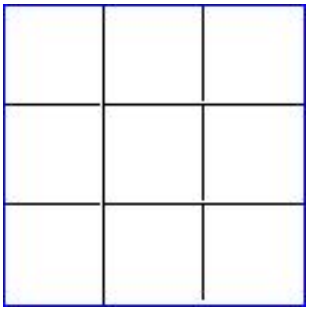
Next, we apply our edge detection techniques to images similar to Fig. 4.1(a) but with different intensities as shown in Fig. 4.3(a) and Fig. 4.5(a). Figs. 4.3(b)−(d) and Figs. 4.5(b)−(d) illustrate the edge detection result applying VMD, VDD and MD to Fig. 4.3(a). Fig. 4.4 and Fig. 4.5 give the corresponding histograms of local optimal threshold T_l^* of VMD, VDD and MD to Fig. 4.3(a) and Fig. 4.5(a) respectively.



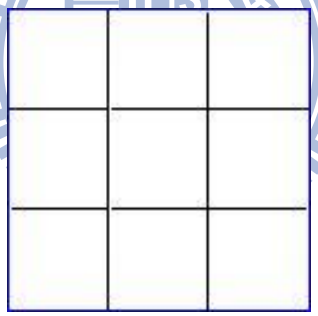
(a)



(b)

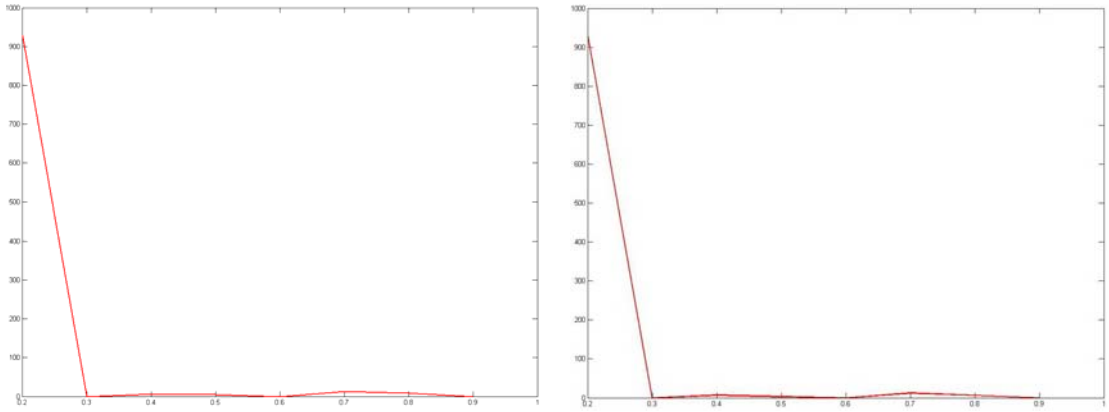


(c)

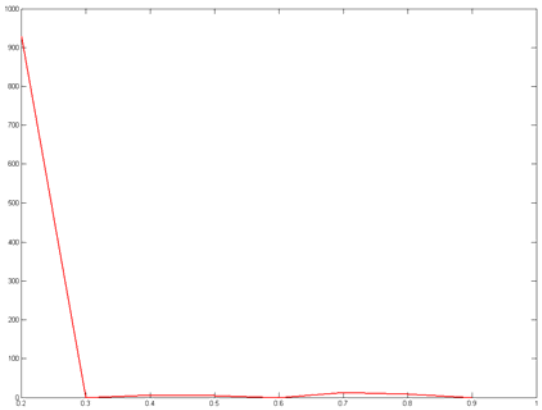


(d)

Fig. 4.3. Edge detection results of a color image with different intensities. (a) original image, (b) edge detection by VMD, (c) edge detection by VDD, (d) edge detection by MD.



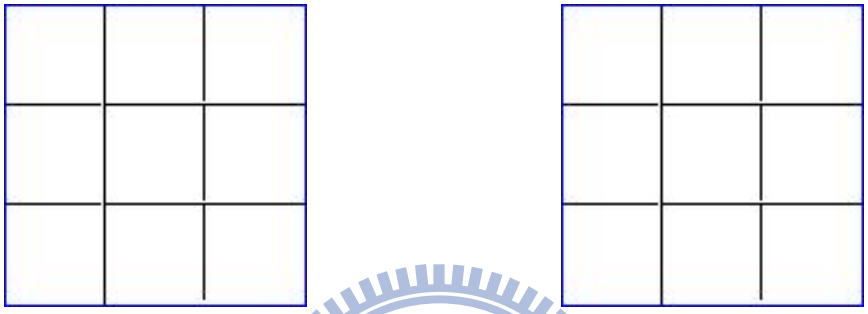
(a) (b)



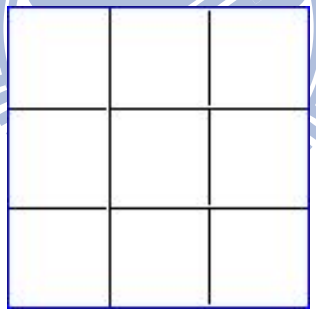
(c)

Fig. 4.4. The corresponding histograms of local optimal threshold T_l^* to Fig. 4.3(a). (a) histogram of T_l^* using VMD, (b) histogram of T_l^* using VDD, (c) histogram of T_l^* using MD.

(a)

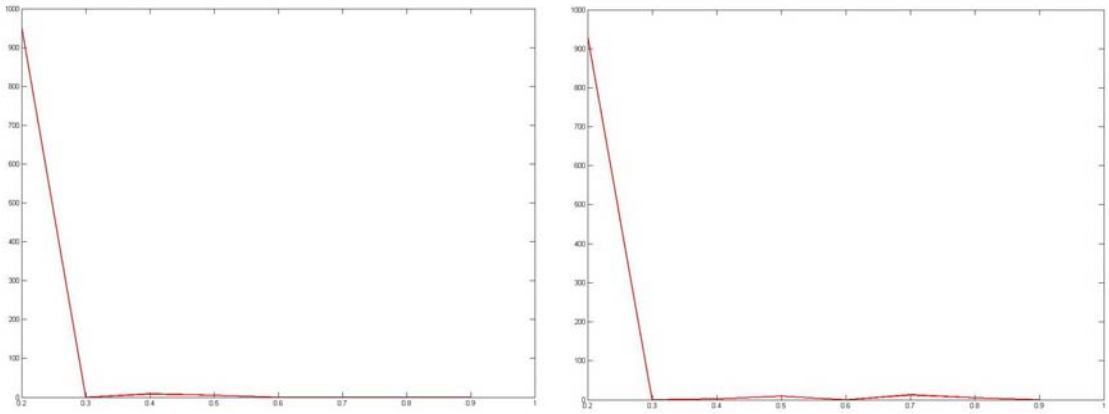


(b) (c)

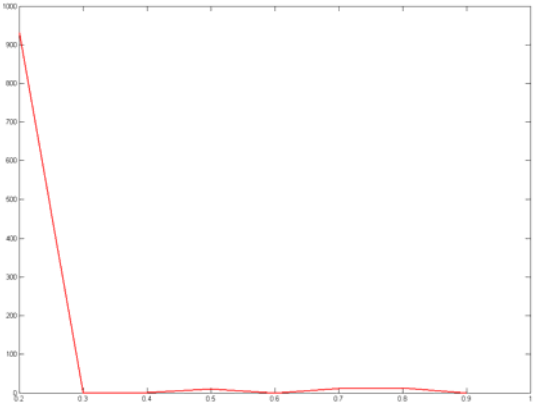


(d)

Fig. 4.5. Edge detection results of a color image with different intensities. (a) original image, (b) edge detection by VMD, (c) edge detection by VDD, (d) edge detection by MD.



(a) (b)

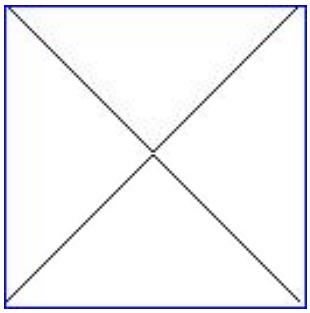


(c)

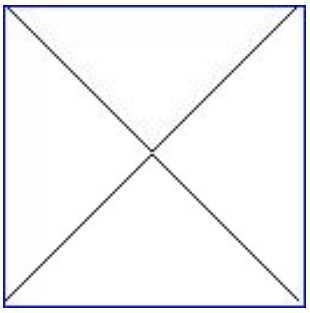
Fig. 4.6. The corresponding histograms of local optimal threshold T_l^* to Fig. 4.5(a). (a) histogram of T_l^* using VMD, (b) histogram of T_l^* using VDD, (c) histogram of T_l^* using MD.

From the above results, our vector-order statistics based edge detection techniques are capable of detecting vertical and horizontal edges. Therefore, we are interested in their ability to detect diagonal edges. Fig. 4.7 shows the result of applying our edge detection techniques to an image with diagonal edges. Fig. 4.8 shows the corresponding histograms of local optimal threshold T_l^* of VMD, VDD and MD to Fig. 4.7(a). At last, we apply our techniques to real world images, as shown in Fig. 4.9(a) and Fig. 4.10(a). The edge detection results are given in Figs. 4.9(b) – (d) and Figs. 4.10(b) – (d).

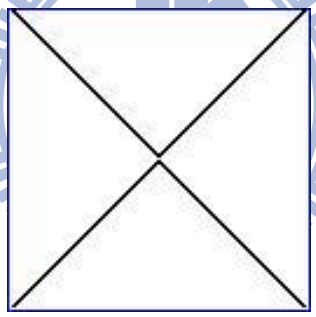
(a)



(b)

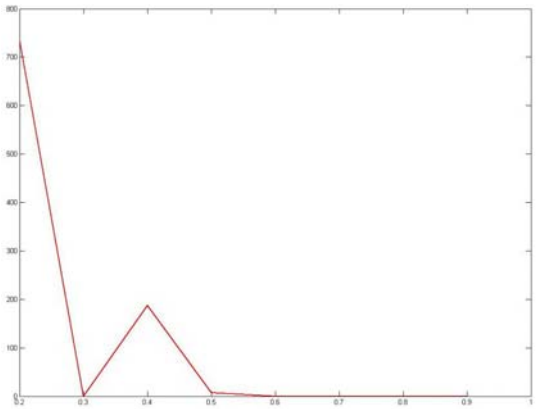


(c)

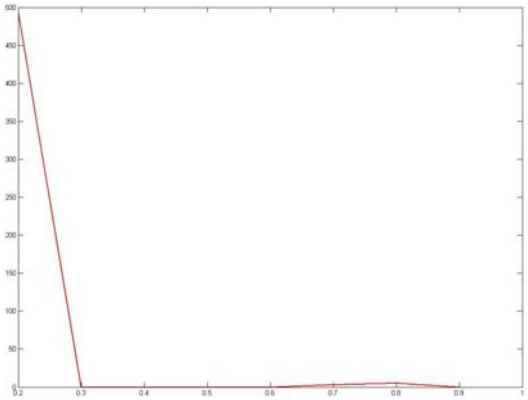


(d)

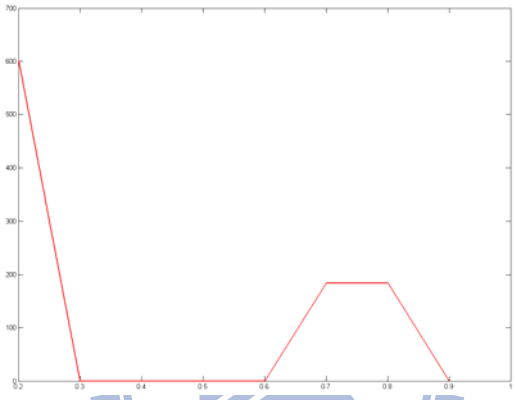
Fig. 4.7. Edge detection results of a color image with diagonal edges. (a) original image, (b) edge detection by VMD, (c) edge detection by VDD, (d) edge detection by MD.



(a)



(b)

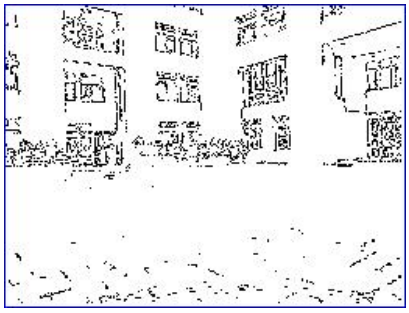


(c)

Fig. 4.8. The corresponding histograms of local optimal threshold T_l^* to Fig. 4.7(a). (a) histogram of T_l^* using VMD, (b) histogram of T_l^* using VDD, (c) histogram of T_l^* using MD.

(a)

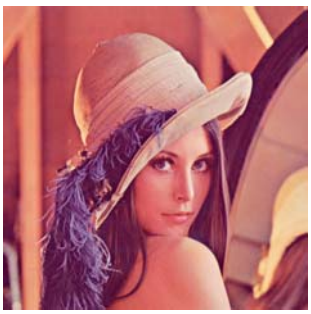
(b)



(c)

(d)

Fig. 4.9. Edge detection results of a real world image. (a) original image, (b) edge detection by VMD, (c) edge detection by VDD, (d) edge detection by MD.



(a)

(b) (c)

(d)

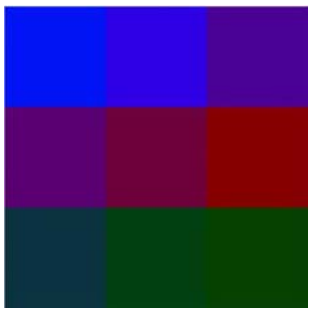
Fig. 4.10. Edge detection results of a human face image. (a) original image, (b) edge detection by VMD, (c) edge detection by VDD, (d) edge detection by MD.

4.2 PCA based Edge Detection

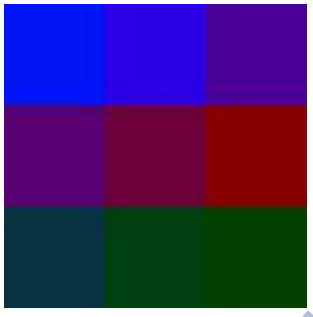
Here, we test our PCA based edge detection technique. If an input image has size $M \times N$, we select the minimum between M and N, as mentioned in Sec. 3.2.2., to be L. We select k_i to be 16% of L for Y color component and 8% of L for C_b and C_r components. In Sec. 3.2.2., k_i represents how many eigenvectors we select to retain the low and high frequency contents of an input image. After we obtained the eigencolor image, Sobel operator is applied to obtain edge responses. The global entropic thresholding technique is then applied to generate the global optimal threshold T_g^*. The sensitivity parameter γ is generated automatically to facilitate the global entropic thresholding technique.

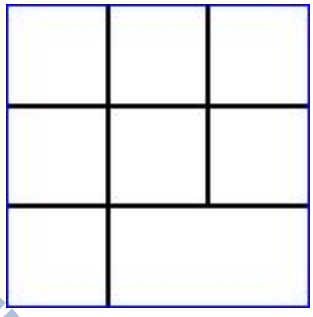
We first apply the image in Fig. 4.1(a) to test our PCA based edge detection technique, as shown in Fig. 4.11(a). Fig. 4.11(b) shows the corresponding eigencolor image in RGB color space. Fig. 4.11(c) is the corresponding edge detection result. From Fig. 4.11(b), we can observe that PCA can capture the low and high frequency image contents. Fig. 4.12 and Fig. 4.13 show the results of applying PCA based edge detection techniques to the color images in Fig. 4.5(a) and Fig. 4.7(a). Fig. 4.14 shows the probability density function (PDF) plots and cumulative distribution function (CDF) plots of T_g^* taking the image with diagonal edges as an example. We can observe clearly that the maximum of entropy occurs when the numbers of pixels in edge class and in non-edge class are equal.

At last, we apply our techniques to real world images, as shown in Fig. 4.15(a) and Fig. 4.16(a). Fig. 4.15(b) and Fig. 4.16(b) show the corresponding eigencolor images in RGB color space. Fig. 4.15(c) and Fig. 4.16(c) are the corresponding edge detection results.



(a)



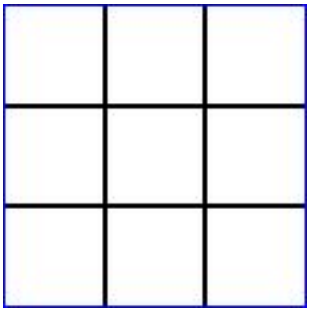


(b) (c)

Fig. 4.11. Edge detection result of a color image with the same intensity. (a) original image, (b) the eigencolor image in RGB color space, (c) the edge detection result.

(a)

(b)

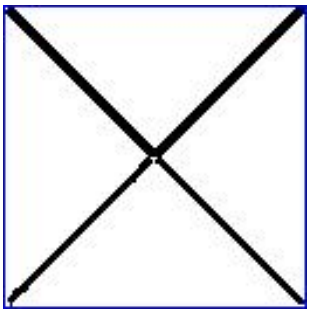


(c)

Fig. 4.12. Edge detection result of a color image with the different intensities. (a) original image, (b) the eigencolor image in RGB color space, (c) the edge detection result.

(a)

(b)



(c)

Fig. 4.13. Edge detection result of a color image with diagonal edges. (a) original image, (b) the eigencolor image in RGB color space, (c) the edge detection result.

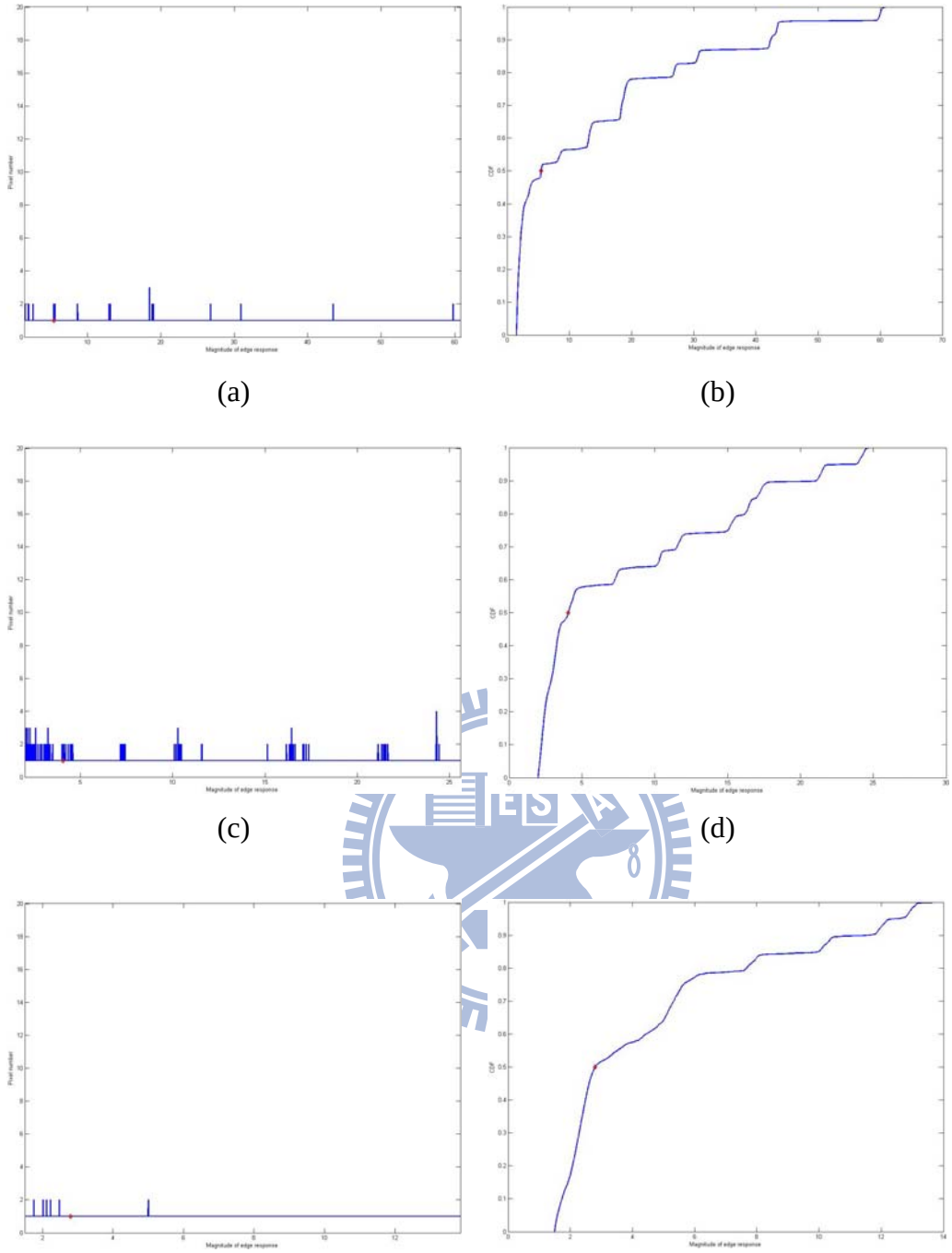


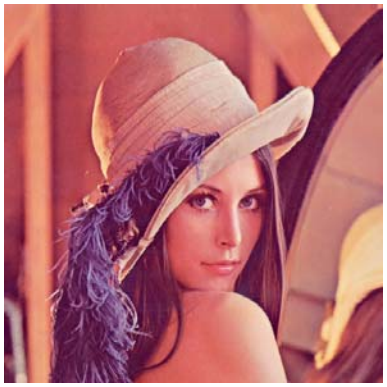
Fig. 4.14. The PDF and CDF of T_g^* of the color image with diagonal edges. (a) PDF of Y, (b) CDF of Y (CDF of T_g^* =0.499), (c) PDF of C_b, (d) CDF of C_b (CDF of T_g^* =0.504), (e) PDF of C_r, (f) CDF of C_r (CDF of T_g^* =0.501).

(a)

(b)

(c)

Fig. 4.15. Edge detection result of a real world image. (a) original image, (b) the eigencolor image in RGB color space, (c) the edge detection result.



(a)

(b)

(c)

Fig. 4.16. Edge detection result of a human face image. (a) original image, (b) the eigencolor image in RGB color space, (c) the edge detection result.

4.3 Comparing the Experimental results

First, we would like to introduce Pratt's *Figure Of Merit* (FOM) and TPR, TNR and ACC of *Receiver Operating Characteristic* (ROC) as performance measures. FOM is defined as

$$FOM = \frac{1}{\max\{I_D, I_I\}} \sum_{i=1}^{I_D} \frac{1}{1+\alpha(d_i)^2} \times 100\% \tag{30}$$

where I_D and I_I are the number of detected and number of ideal edge points respectively, α (>0) is a calibration constant, and d_i is the edge deviation for the ith detected edge pixel. In all cases $0 < \text{FOM} \le 1$; for a perfect match between the detected and the ideal edges FOM = 1 whereas the detected edges deviate more and more from the ideal ones FOM goes to zero. The scaling constant $\alpha = 1/9$ proposed in [14] has been adopted. Next, *True Positive Rate* (TPR) is defined as

$$TPR = \frac{TP}{(TP + FN)} \times 100\% \tag{31}$$

where *TP* (*true positive*) represents the number of pixels which are detected as an edge pixel and belong to an ideal edge pixel, and *FN* (*false negative*) represents the number of pixels which are detected as an non-edge pixel but belong to an ideal edge pixel.

On the other hand, *true negative rate* (TNR) is defined as

$$TNR = \frac{TN}{TN + FP} \times 100\% \tag{32}$$

where *TN* (*true negative*) represents the number of pixels which are detected as an non-edge pixel and belong to an ideal non-edge pixel, and *FP* (*false positive*) represents the number of pixels which are detected as an edge pixel but belong to an

ideal non-edge pixel.

At last, *accuracy* (ACC) is defined as

$$ACC = \frac{TP + TN}{P + N} \times 100\% \tag{33}$$

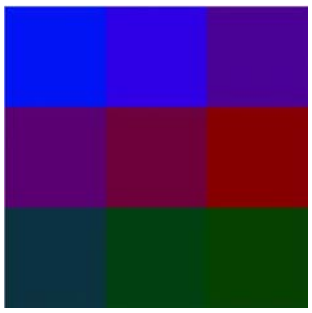
where P (*positive*) represents the total number of ideal edge pixels, and N (*negative*) represents the total number of ideal non-edge pixels. We also calculate the *normalized accuracy* by

$$NACC = \frac{TPR + TNR}{2} \times 100\% \tag{34}$$

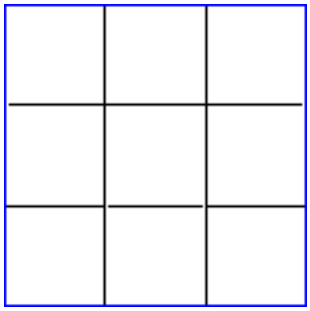
The performance of our automatic color edge detection techniques are compared to those by color Canny edge detector [1], MVD edge detector [14], directional operator [15], Dikbas *et al*. [18], and automatic isotropic color edge detector [9]. Figs. 4.17 – 4.19 show the edge detection results of synthetic images for comparison. Tables I – III show the corresponding FOM, TPR, TNR, ACC and NACC of various edge detection schemes, with the script representing the rank of the method, of Figs. 4.17 – 4.19. From Tables I – III, we can see that our edge detection techniques perform well compared to other edge detection techniques. Figs. 4.20 and 4.21 show the edge detection results of real world images for comparison.

From the above results, we can see that our techniques perform consistently well compared to other edge detection techniques. Comparing the results by our PCA based technique with the results by automatic isotropic color edge detector, we can see that our global entropic thresholding technique performs better because that the sensitivity parameter γ alleviates the bias leaning to the non-edge pixels' edge strength, which is mentioned in Sec. 3.3.2. The results of automatic isotropic color edge detector are too sensitive because that the edge detector utilizes all the pixels' edge responses to determine the optimal threshold without alleviating the bias caused

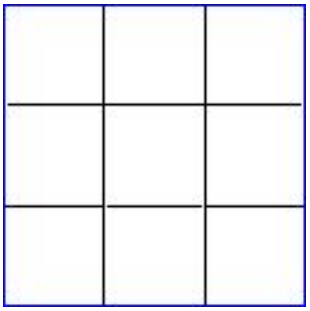
from dominant non-edge pixels in number. From the results of our vector-statistics based edge detectors, VDD edge detector is more capable of alleviating the influence from noises but is too insensitive to real edges because that VDD detector discards the outlier $X^{(9)}$ while calculating the edge response. In general, MD detector is sensitive to small texture variation since it detects the genuine local maximal variance within the processing area.



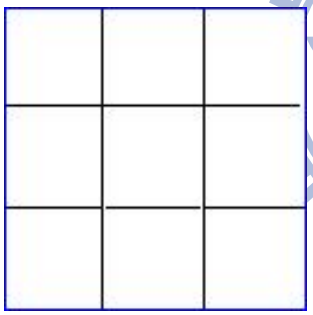
(a)



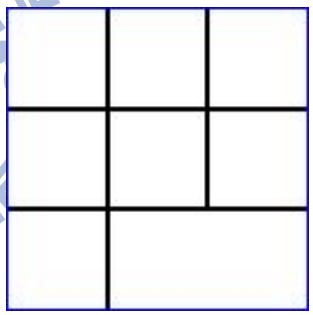
(b)



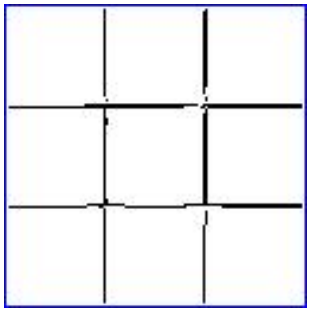
(c)



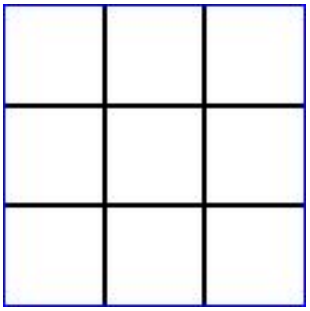
(d)



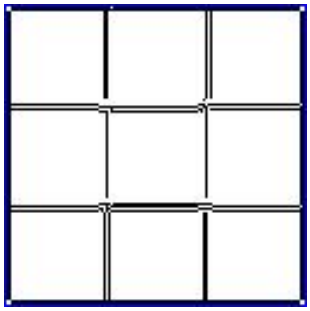
(e)



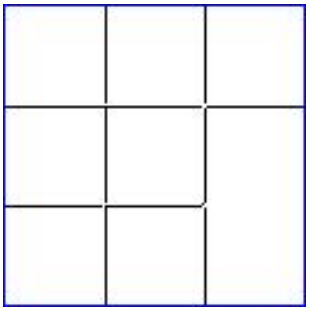
(f)



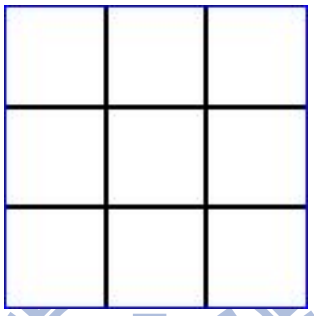
(g)



(h)



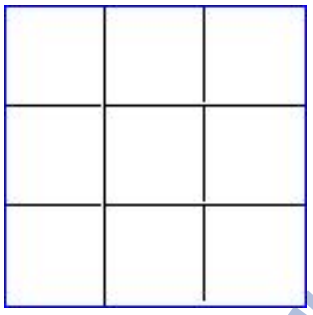
(i)



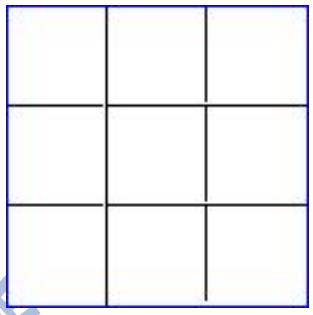
(j)

Fig. 4.17. Edge detection result of a color image with the same intensity. (a) original image, (b) VMD, (c) VDD, (d) MD, (e) eigencolor, (f) color Canny, (g) MVD, (h) directional operator, (i) Dikbas *et al*. [18], (j) automatic isotropic color edge detector.

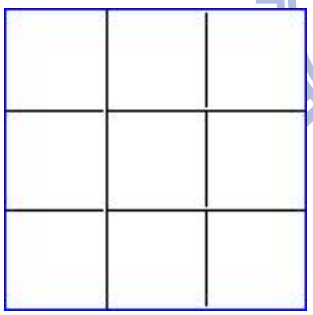
(a)



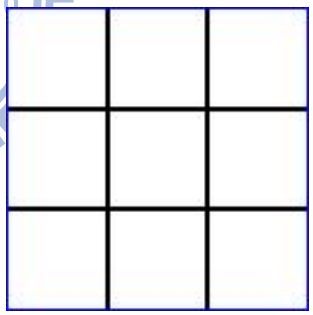
(b)



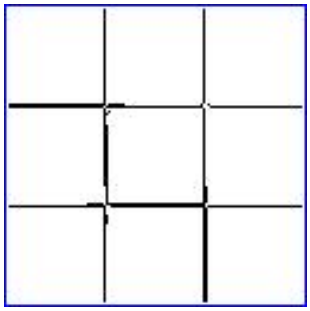
(c)



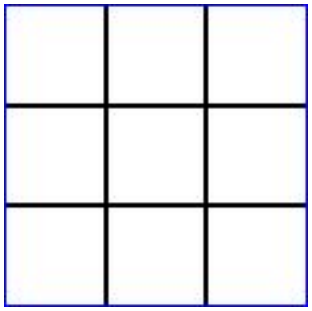
(d)



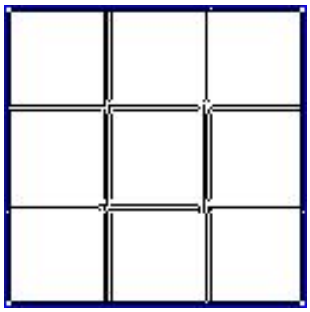
(e)



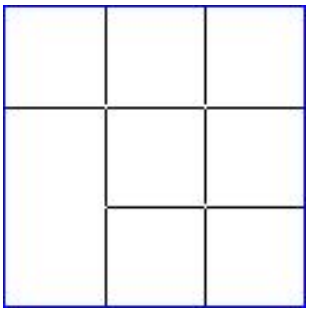
(f)



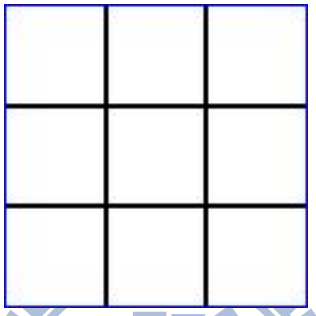
(g)



(h)

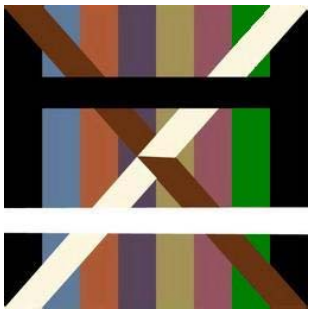


(i)

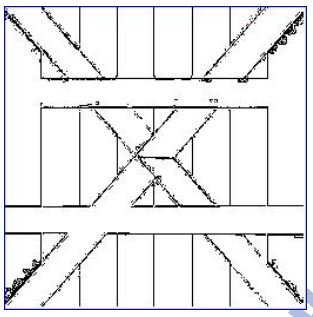


(j)

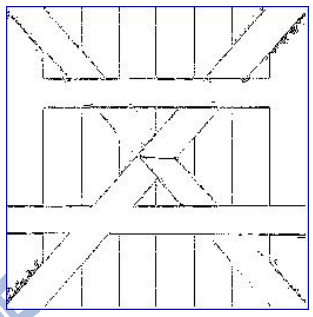
Fig. 4.18. Edge detection result of a color image with the different intensities. (a) original image, (b) VMD, (c) VDD, (d) MD, (e) eigencolor, (f) color Canny, (g) MVD, (h) directional operator, (i) Dikbas *et al*. [18], (j) automatic isotropic color edge detector.



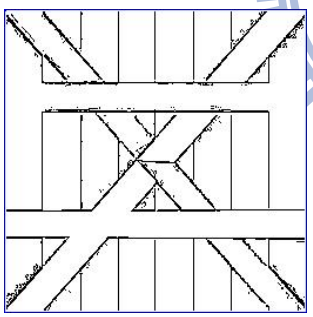
(a)



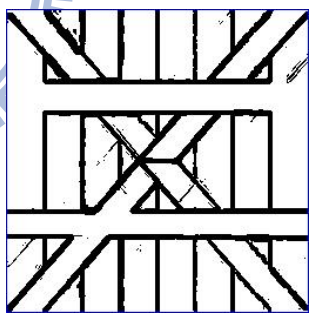
(b)



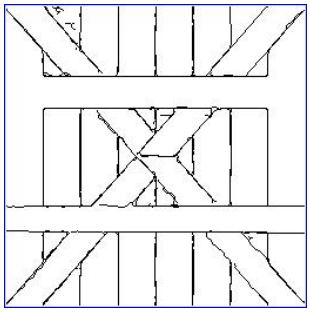
(c)



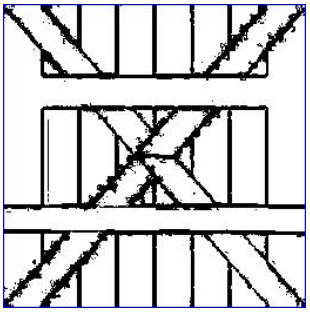
(d)



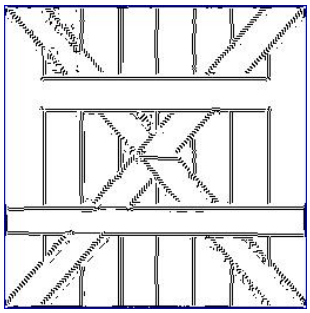
(e)



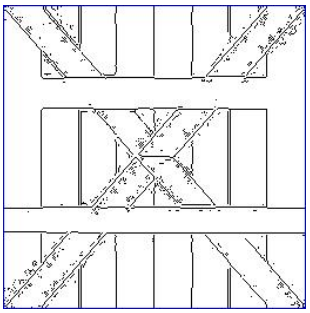
(f)



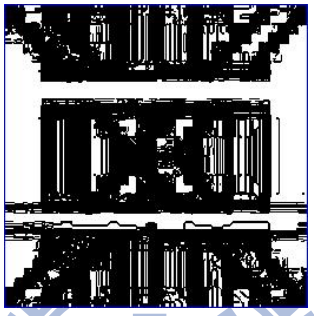
(g)



(h)



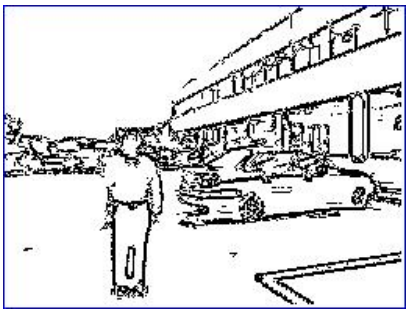
(i)



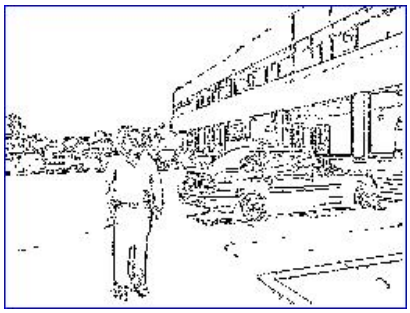
(j)

Fig. 4.19. Edge detection result of a synthetic color image. (a) original image, (b) VMD, (c) VDD, (d) MD, (e) eigencolor, (f) color Canny, (g) MVD, (h) directional operator, (i) Dikbas *et al*. [18], (j) automatic isotropic color edge detector.

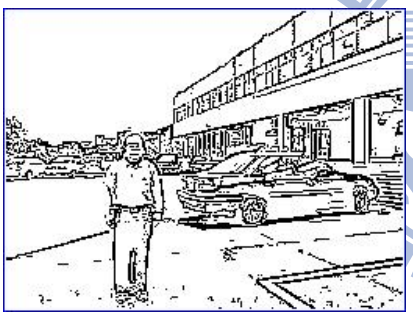
(a)



(b)

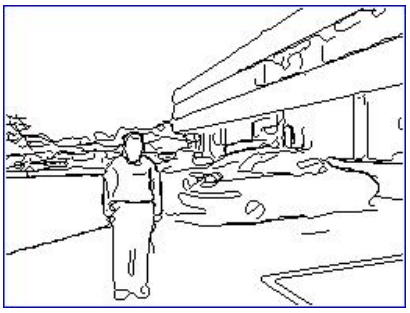


(c)



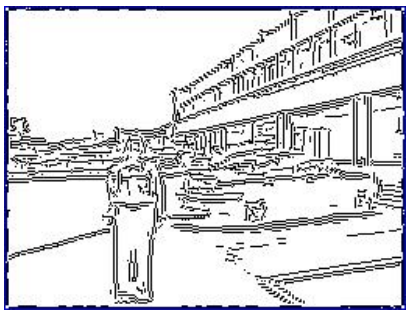
(d)

(e)

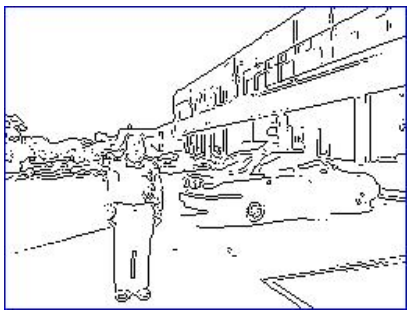


(f)

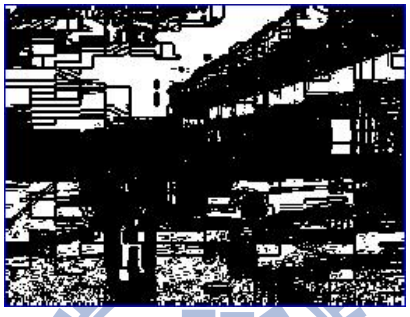
(g)



(h)



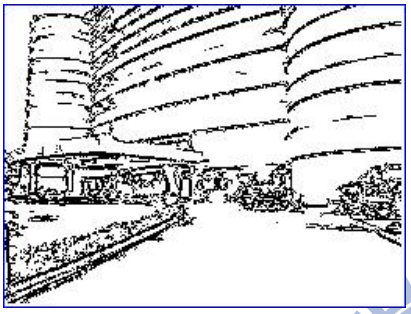
(i)



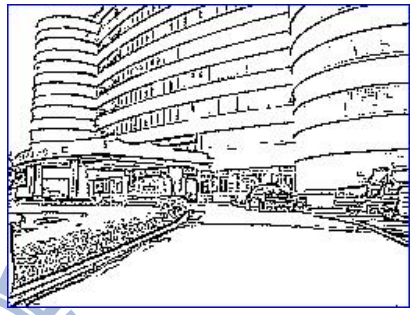
(j)

Fig. 4.20. Edge detection result of a real world image. (a) original image, (b) VMD, (c) VDD, (d) MD, (e) eigencolor, (f) color Canny, (g) MVD, (h) directional operator, (i) Dikbas *et al*. [18], (j) automatic isotropic color edge detector.

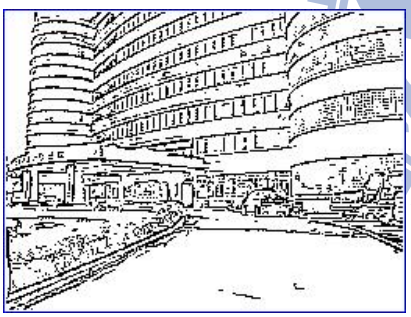
(a)



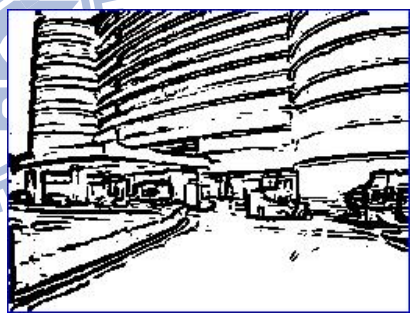
(b)



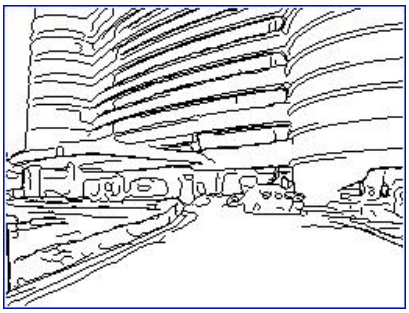
(c)



(d)

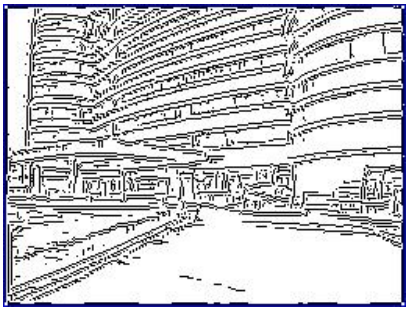


(e)

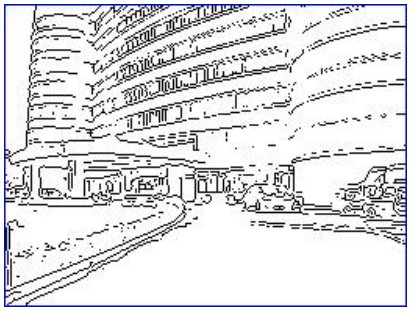


(f)

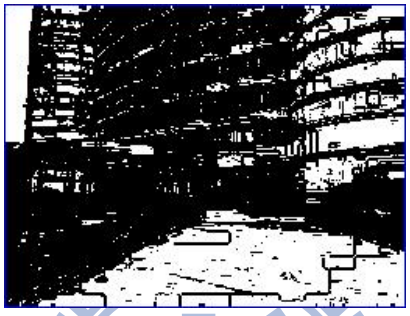
(g)



(h)



(i)



(j)

Fig. 4.21. Edge detection result of a real world image. (a) original image, (b) VMD, (c) VDD, (d) MD, (e) eigencolor, (f) color Canny, (g) MVD, (h) directional operator, (i) Dikbas *et al*. [18], (j) automatic isotropic color edge detector.

TABLE I

EVALUATION RESULT OF FOM, TPR, TNR, ACC AND NACC OF FIG. 4.17.

Method	FOM (%)	TPR (%)	TNR (%)	ACC (%)	NACC (%)
VMD	98.82 $_{2}$	98.82	100.00	99.96 $_{2}$	99.41 $_{2}$
VDD	98.82 $_{2}$	98.82	100.00	99.96 $_{2}$	99.41 $_{2}$
MD	98.82 $_{2}$	98.82	100.00	99.96 $_{2}$	99.41 $_{2}$
Eigencolor	92.00 $_{4}$	92.00	100.00	99.48 $_{4}$	96.00 $_{3}$
Color Canny	95.28 $_{3}$	89.80	99.72	94.69 $_{5}$	94.76 $_{5}$
MVD	**100.00** $_{1}$	100.00	100.00	**100.00** $_{1}$	**100.00** $_{1}$
Directional operator	53.78 $_{6}$	62.84	91.65	89.72 $_{6}$	77.25 $_{6}$
Dikbas *et al*. [18]	90.57 $_{5}$	90.57	99.25	99.74 $_{3}$	94.91 $_{4}$
Automatic isotropic	**100.00** $_{1}$	100.00	100.00	**100.00** $_{1}$	**100.00** $_{1}$

TABLE II

EVALUATION RESULT OF FOM, TPR, TNR, ACC AND NACC OF FIG. 4.18.

Method	FOM (%)	TPR (%)	TNR (%)	ACC (%)	NACC (%)
VMD	98.43 $_{3}$	98.43	100.00	99.95 $_{2}$	99.21 $_{2}$
VDD	98.43 $_{3}$	98.43	100.00	99.95 $_{2}$	99.21 $_{2}$
MD	98.43 $_{3}$	98.43	100.00	99.95 $_{2}$	99.21 $_{2}$
Eigencolor	**100.00** $_{1}$	100.00	100.00	**100.00** $_{1}$	**100.00** $_{1}$
Color Canny	98.61 $_{2}$	95.56	99.84	99.71 $_{3}$	97.70 $_{3}$
MVD	**100.00** $_{1}$	100.00	100.00	**100.00** $_{1}$	**100.00** $_{1}$
Directional operator	53.47 $_{5}$	65.56	91.72	89.55 $_{5}$	78.64 $_{5}$
Dikbas *et al*. [18]	90.62 $_{4}$	89.74	99.71	99.65 $_{4}$	94.73 $_{4}$
Automatic isotropic	**100.00** $_{1}$	100.00	100.00	**100.00** $_{1}$	**100.00** $_{1}$

TABLE III

EVALUATION RESULT OF FOM, TPR, TNR, ACC AND NACC OF FIG. 4.19.

Method	FOM (%)	TPR (%)	TNR (%)	ACC (%)	NACC (%)
VMD	90.26 $_{3}$	88.52	97.85	96.74 $_{3}$	93.19 $_{2}$
VDD	89.12 $_{5}$	70.20	99.10	**97.35** $_{1}$	84.65 $_{6}$
MD	91.25 $_{2}$	90.65	97.44	96.77 $_{2}$	**94.05** $_{1}$
Eigencolor	**91.68** $_{1}$	97.26	88.68	89.75 $_{6}$	92.97 $_{3}$
Color Canny	91.05 $_{4}$	83.33	95.68	95.15 $_{5}$	89.50 $_{5}$
MVD	85.43 $_{6}$	94.32	87.64	88.78 $_{7}$	90.98 $_{4}$
Directional operator	77.78 $_{8}$	42.44	92.15	88.62 $_{8}$	67.30 $_{9}$
Dikbas *et al*. [18]	84.37 $_{7}$	70.57	95.25	95.74 $_{4}$	82.91 $_{7}$
Automatic isotropic	43.51 $_{9}$	99.47	36.21	41.54 $_{9}$	67.84 $_{8}$

Chapter 5 Conclusion and Future Work

In this thesis, we have proposed automatic color edge detection schemes by entropic thresholding. By using our edge detection and automatic thresholding techniques, we can detect not only edges with different intensities but also edges with minor color difference and similar intensity. In this way, the performances of higher level image processing tasks such as segmentation and object recognition can be improved because of the improvement of edge detection result. Experimental results have shown that our automatic color edge detection techniques are good and reliable.

In the future study, we can further improve our automatic thresholding technique by applying texture analysis to the input image. By texture analysis, we can obtain the information concerning the amount of edges in the input image. Hence, we can have a more discreet method to obtain good γ parameter and then alleviate the bias resulting from dominant non-edge pixels.

References

[1] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 8, no. 6, pp. 679–698, 1986.

[2] M. A. Ruzon and C. Tomasi , “Color edge detection with the compass operator,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1999, vol. 2, pp. 160–166.

[3] A. Cumani, “Efficient contour extraction in color images,” in *ACCV ’98: Proceedings of the Third Asian Conference on Computer Vision*, London, UK, 1998, vol. 1, pp. 582–589, Springer-Verlag.

[4] C. L. Novak and S. A. Shafer, “Color edge detection,” in *Proc. of DARPA Image Understanding Workshop*, 1987, pp. 35–37.

[5] A. Koschan and M. Abidi, “Detection and classification of edges in color images,” *IEEE Signal Processing Mag.*, vlo. 22, no. 1, pp. 64–73, January 2005.

[6] R. Nevatia, “A color edge detector and its use in scene segmentation,” *IEEE Trans. Syst., Man, Cybernetics*, vol. 7, no. 11, pp. 820–826, 1977.

[7] G. Robinson, “Color edge detection,” *Optical Eng.*, vol. 16, no. 5, pp. 479–484, September 1977.

[8] A. Shiozaki, “Edge extraction using entropy operator,” *CVGIP*, vol. 36, pp. 1–9, 1986.

[9] J. Fan, W. G. Aref, M. S. Hacid, and A. K. Elmagarmid, “An improved automatic isotropic color edge detection technique,” *Pattern Recognition Letters*, vol. 22, no. 13, pp. 1419–1429, November 2001.

[10] A. Koschan, “A comparative study on color edge detection,” in *Proc. 2nd Asian Conf. Computer Vision-ACCV’95*, 1995, vol. III, pp. 574–578

[11] R. Machuca and K. Phillips, “Applications of vector fields to image processing,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 5, no. 3, pp. 316–329, May 1983.

[12] S. Di Zenzo, “A note on the gradient on of a multi-image,” *CVGIP*, vol. 33, pp. 116–125, 1986.

[13] P.E. Trahanias and A.N. Venetsanopoulos, “Color edge detection using vector order statistics,” *IEEE Trans. Image Process.*, vol. 2, no. 2, pp. 259–264, April 1993.

[14] P.E. Trahanias and A.N. Venetsanopoulos, “Vector order statistics operators as color edge detectors,” *IEEE Trans. Syst., Man, Cybern., B*, vol. 26, no. 1, pp. 135–142, February 1996.

[15] J. Scharcanski and A.N. Venetsanopoulos, “Edge detection of color images using directional operators,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 1, pp. 135–142, February 1996.

[16] K.N. Plataniotis, A.N. Venetsanopoulos and S.Y. Zhu, “Comprehensive analysis of edge detection in color image processing,” *Opt. Eng.*, vol. 38, no. 4, pp. 612–625, April 1999.

[17] V. Barnett, “The ordering of multivariate data,” *J. Royal Statistical Society A*, vol. 139, Part 3, pp. 318–343, 1976.

[18] S. Dikbas, T. Arici, and Y. Altunbasak, “Chrominance edge preserving grayscale transformation with approximate first principal component for color edge detection,” *Proc. IEEE Int’l. Conf. on Image Processing*, pp. 261–264, 2007.

[19] M. Turk and A. Pentland, “Eigenfaces for recognition,” *Journal of Cognitive Neuroscience*, vol. 19, pp. 743–756, 1997.

[20] H. A. David, *Order Statistics*. New York: Wiley, 1980.

[21] J. Astola, P. Haavisto and Y. Neuvo, “Vector median filters,” *Proc. IEEE*, vol. 78, pp. 678–689, April 1987.

[22] G. Joharmsen, and J. Bille, “A threshold selection method using information measures,” *6th Int. Conf. Pattern Recognition*, Munich, Germany, 1982.

[23] A. K. C. Wong and P. K. Sahoo, “A gray-level thresholding selection method based on maximum entropy principle,” *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 866–871, 1989.

[24] I. Abdou. And W. Pratt, “Quantitative design and evaluation of enhancement/thresholding edge detectors,” *Proc. IEEE*, vol. 67, pp. 753–763, 1979.

[25] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.