國 立 交 通 大 學

電 控 工 程 研 究 所

碩 士 論 文

應用廣義加權平均集成運算與學習法則於影像邊緣偵測

Applying Weighted Generalized Mean Aggregation and Learning Rule to Edge Detection of Images

研 究 生：沈 煜 倫

指 導 教 授：張 志 永

中 華 民 國 一百零二 年 七 月

應用廣義加權平均集成運算與學習法則於影像邊緣偵測

Applying Weighted Generalized Mean Aggregation and Learning Rule to Edge Detection of Images

學　　生：沈煜倫　　Student : Yu-Lun Shen

指導教授：張志永　　Advisor : Jyh-Yeong Chang

國立交通大學

電機工程學系

碩士論文

A Thesis

Submitted to Department of Electrical Engineering

College of Electrical Engineering

National Chiao-Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master in

Electrical Control Engineering

July 2013

Hsinchu, Taiwan, Republic of China

中 華 民 國 一百零二 年 七 月

應用廣義加權平均集成運算與學習法則於影像邊緣偵測

學生:沈煜倫　　指導教授: 張志永博士

國立交通大學電控工程研究所

摘要

在這篇論文中，我們運用廣義加權平均建立區間值模糊關係，進行灰階影像邊緣偵測，並推導參數的學習法則以達成影像邊緣偵測。

我們的邊緣偵測方法包含三個部分。第一部分，在3×3滑動視窗中，我們利用上限與下限建構子計算中心像素和其八鄰域像素的加權平均集成運算，建立區間值模糊關係，及可指出相對應像素強度值變化程度的 W 模糊關係。第二部分，我們藉著離散型梯度演算法的概念進行加權平均參數的學習與更新，並引入口袋演算法(pocket algorithm)獲得最佳的參數集合。最後，我們運用後處理技術，包括增強邊緣的連接性並移除孤立的像素，以獲得較好的邊緣影像。從六張添加隨機雜訊的灰階合成影像訓練結果顯示，我們的方法產生較穩定且強健的邊緣偵測; 並且我們的方法對自然影像的邊緣偵測，比著名的 Canny 邊緣偵測器顯示出更清楚的細節。

Applying Weighted Generalized Mean Aggregation and Learning Rule to Edge Detection of Images

STUDENT: Yu-Lun Shen ADVISOR: Dr. Jyh-Yeong Chang

Institute of Electrical Control Engineering
National Chiao-Tung University

ABSTRACT

In this paper, we apply generalized weighted mean to construct interval-valued fuzzy relations for grayscale image edge detection and derive the learning formulas for parameters in order to decrease the edge detection error.

The proposed detector consists of three stages. In the first stage, we use the upper and lower constructors to calculate the weighted mean aggregations of the central pixel and its eight neighbor pixels in each 3×3 sliding window. Then we construct the interval-valued fuzzy relation and its associated W-fuzzy relation indicating the degree of intensity variation between the center pixel and its neighborhood. In the second stage, we update the weighting parameters of the mean which can be learned by the gradient method casted in discrete formulation and utilize pocket algorithm to obtain the optimal parameter set for all training images. Finally, we use post-processing techniques to strengthen the connectivity of edges and remove isolated pixels for obtaining better edge images. Our method produces a more stable and robust edge images on synthetic images and nature images as well, in comparison with the well-known Canny edge detector.

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

Digital images are valuable sources of information in many research and application areas including biology, material science, tracing, etc. Edge detection is a topic of continuing interest because it is a key issue in pattern recognition, image processing and computer vision. The edge detection process contributes to preserve useful structural information about object boundaries and, at the same time, simplify the analysis of images by reducing the amount of data extremely.

A great number of edge detectors lean their strategy on: 1) calculating a characteristic image (the characteristic image often means gradient image in many cases), and 2) thresholding the characteristic image to get the edge map [1], [3]. In grayscale edge detection, the Canny edge detector [1] which uses a multi-stage algorithm to detect a wide range of edges is famous for its high sensitivity and reliability. The three performance criteria he proposed for optimal edge detector are as follows: (1) Good detection, (2) Good localization, and (3) Only one response to a single edge.

To satisfy these requirements, Canny have used several techniques such as Gaussian filter, non-maximum suppression and hysteresis thresholding to let the output edge map more accurate and well-connected.

In this thesis, an improvement of weighted generalized aggregation algorithm is presented. The original weighted generalized aggregation algorithm only updates two designing parameters (α_S, α_T). Furthermore, it does not have any measure for continuity and threshold selecting. Our proposed method will overcome the drawbacks and obtain the best operating parameter set for edge detection in a limited number of iterative learning.

1.2 Fuzzy Set

1.2.1 Type 1 Fuzzy Set

Fuzzy logic theory has been successfully applied to many areas, such as image enhancement, classification and thresholding value selection [4]. The concept of fuzzy sets was first presented by Zadeh in 1965 [5], and was suitable for eliminating the grayness ambiguities/vagueness. In fuzzy set theory, a degree of membership in the interval [0, 1] is assigned to each element of the set. A flat membership function indicates the high image data vagueness, and hence a difficult thresholding.

For an $M \times N$ image subset $A \subseteq X$ with membership function $\mu_X(g)$, the most common measure of fuzziness is the linear index of fuzziness [7, 8], for the spatial case, which can be defined as follows:

$$\gamma_l = \frac{2}{MN} \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} \min[\mu_A(g_{ij}), 1-\mu_A(g_{ij})] \tag{1.1}$$

Many suitable membership functions $\mu_A(g)$ have been defined, such as standard S-function [8], the Huang and Wang function [9] and threshold as a fuzzy number used by Tizhoosh [10], to measure the global or local image fuzziness.

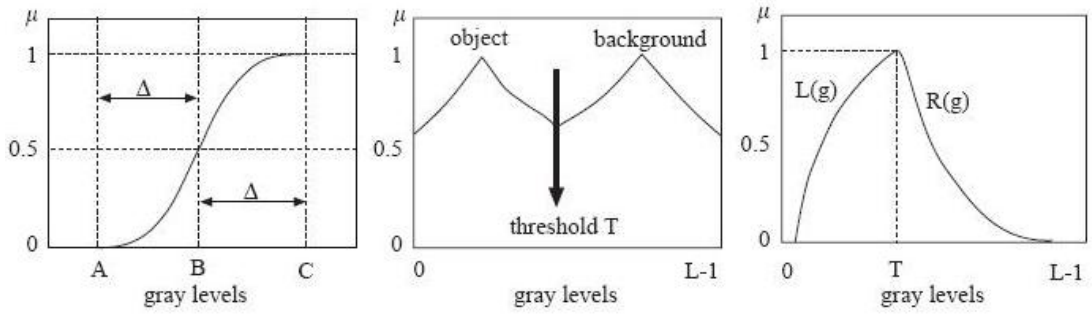


Fig. 1.1 Different membership functions. From left to right: S-function used by Pal and Rosenfeld [11], function used by Huang and Wang [9], and threshold as a fuzzy number used by Tizhoosh [10].

1.2.2 Type 2 Fuzzy Set

The problems with fuzzy sets type 1 are that it is not possible to say which membership function is the best one, and the assignment of a membership degree to a pixel is not certain. Because the experts define membership functions based on their knowledge due to the dilemma they met.

Therefore, Zedah presented the concept of type 2 fuzzy sets in 1975 [6] to find a more robust solution. In these sets, the membership function is itself represented by a fuzzy set on the interval [0, 1] and is able to model such uncertainties. Mendel and John [12] proved that one particular case of a type 2 fuzzy set is an interval type 2 set, which is equivalent to an interval-valued fuzzy set. The reason why we choose to work with interval-valued fuzzy set is that edge detection techniques attempt to find the pixel whose gray level intensity is very dissimilar to its neighbors. That means interval-valued fuzzy set is associated with not just a membership degree but also the length of its membership interval which can be used to indicate the difference of intensities related to a pixel and its neighbors.

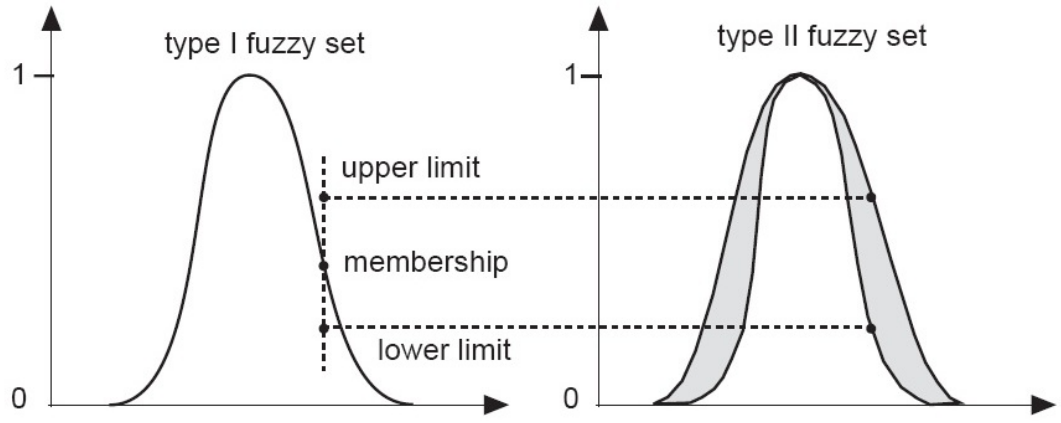


Fig. 1.2 A possible way to construct type 2 fuzzy set. The interval between lower and upper membership values (shaded region) should capture the vagueness.

1.3 Image Edge Detection

1.3.1 Image Edge

In view of imaging science, edge pixel represent a significant variation compared with neighbors in gray level (or intensity level of color). Edges are one of the most important visual clues because they can recognize obvious distinction between each geometric object and often used in subsequent image analysis operations for feature detection and object recognition.

1.3.2 Binary Edge Map

Generally, the simplest way to convert an image to a black-and-white image (binary image) is setting an appropriate threshold so that people can easily identify the boundary between object and background. But it is impossible to mark all edge points in a natural image which has complicated and blurred details corrupted by noise due to a number of imperfections in the imaging process. Only the edge point of synthetic image can be indicated exactly because of definitive variation of intensities. We can utilize basic operation like Sobel operator [2] to obtain the binary edge map of synthetic gray level image. Fig. 1.3(a) shows a synthetic image without noise with size 128×128. Fig. 1.3(b) indicates the binary edge map of synthetic image.

Obviously, the binary edge map is pure, clean and easy to calculate the correct rate. Therefore, we will do the test experiment with several synthetic images and compare the accuracy of our method with the accuracy obtained by benchmark Canny edge detector, to confirm whether our method is better than others.

(a) (b)

Fig. 1.3 The result of gray level edge detection. (a) The synthetic gray level image without noise; (b) The binary edge map obtained by Sobel operator [2] setting the threshold value $T = 0.1$.

1.4 Research Method

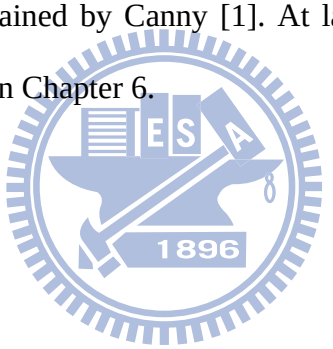
In this thesis, adopting the method proposed by Barrenechea et al. [13], our new edge detection method utilize generalized weighted mean aggregation algorithm to construct interval-valued fuzzy relation. We can obtain two new fuzzy relations to construct the interval-valued fuzzy relation by calculating the weighted mean difference of the central pixel and its 8-neighborhood pixels in a 3×3 sliding window across the image. To avoid the manual parameter selection as Canny edge detector does, we have derived the iterative learning mechanism for the two weighting parameters of the mean aggregation and the threshold T as well.

We make use of six grayscale synthetic images with adding different types and rates of random noises as the input images of the iterative learning mechanism. Besides, we also use pocket algorithm and several accuracy calculation metrics to obtain better accuracy so that we can extract the image edge map more precisely and reliably. We also have some unique post-processing techniques which can strengthen the continuity of the edge map as Canny edge detector does and obtain the best parameter set for the edge detection of all input images. Finally, we compare our edge

with Canny edge detector and discuss its merits and drawbacks.

1.5 Thesis Outline

This paper is organized as follows. Chapter 2 introduces the edge detection method for grayscale images adopting the concept of interval-valued fuzzy relation proposed by Barrenechea et al [13]. In chapter 3, we describe the concept of the perceptron learning algorithm and iterative learning mechanism on the best parameter selection. In chapter 4, some post-processing techniques enabling higher accuracy and continuity will be introduced. In chapter 5, we summarize all experiment results and compare them with those obtained by Canny [1]. At last, we conclude and suggest directions for future research in Chapter 6.



Chapter 2 Construction of Interval-Valued Fuzzy Relation From A Fuzzy Relation

In this chapter, we will introduce how to construct the interval-valued fuzzy relation image, abbreviated as IVFR, by applying the concepts of triangular norm (t-norm) and triangular conorm (t-conorm or s-norm). In each sliding window, we use the upper and lower constructors to calculate the intensity differences between the central pixel and its eight neighbor pixels so that we can construct the interval-valued fuzzy relation and its associated W-fuzzy relation.

Fig. 2.1 demonstrates the concepts and steps of the application of interval-valued fuzzy relations in edge detection of images. We apply the lower and upper constructor to obtain a darker image and brighter image from a grayscale image, respectively. Then, we can construct an IVFR such that the length of each interval represents the fuzzy edges, i.e., W-fuzzy edges. An appropriate threshold is selected to obtain crisp edges from the fuzzy edges.

2.1 Fuzzy Relation

For an image with dimensions of $\mathrm{M}\times\mathrm{N}$ and 256 gray levels, we have to execute the normalization step as follows. We divide the grayscale value of each pixel in the image by the gray-scale maximal intensity "255," so that the grayscale value of each pixel will be between interval of $[0,1]$. Next, we consider two finite universes $X=\{0,\ 1,...,\ M-1\}$ and $Y=\{0,\ 1,...,\ N-1\}$. Then, $R=\{((x,y),R(x,y))\,|\,(x,y)\in$

$X \times Y\}$ is called a FR from X to Y. FRs are described by matrices as follows:

$$R = \begin{pmatrix} R(0,0) & \cdots & R(0,N-1) \\ R(1,0) & \cdots & R(1,N-1) \\ \vdots & \ddots & \vdots \\ R(M-1,0) & \cdots & R(M-1,N-1) \end{pmatrix}. \tag{2.1}$$

Moreover, $F(X \times Y)$ represents the set of all fuzzy relations from X to Y [14], [15].

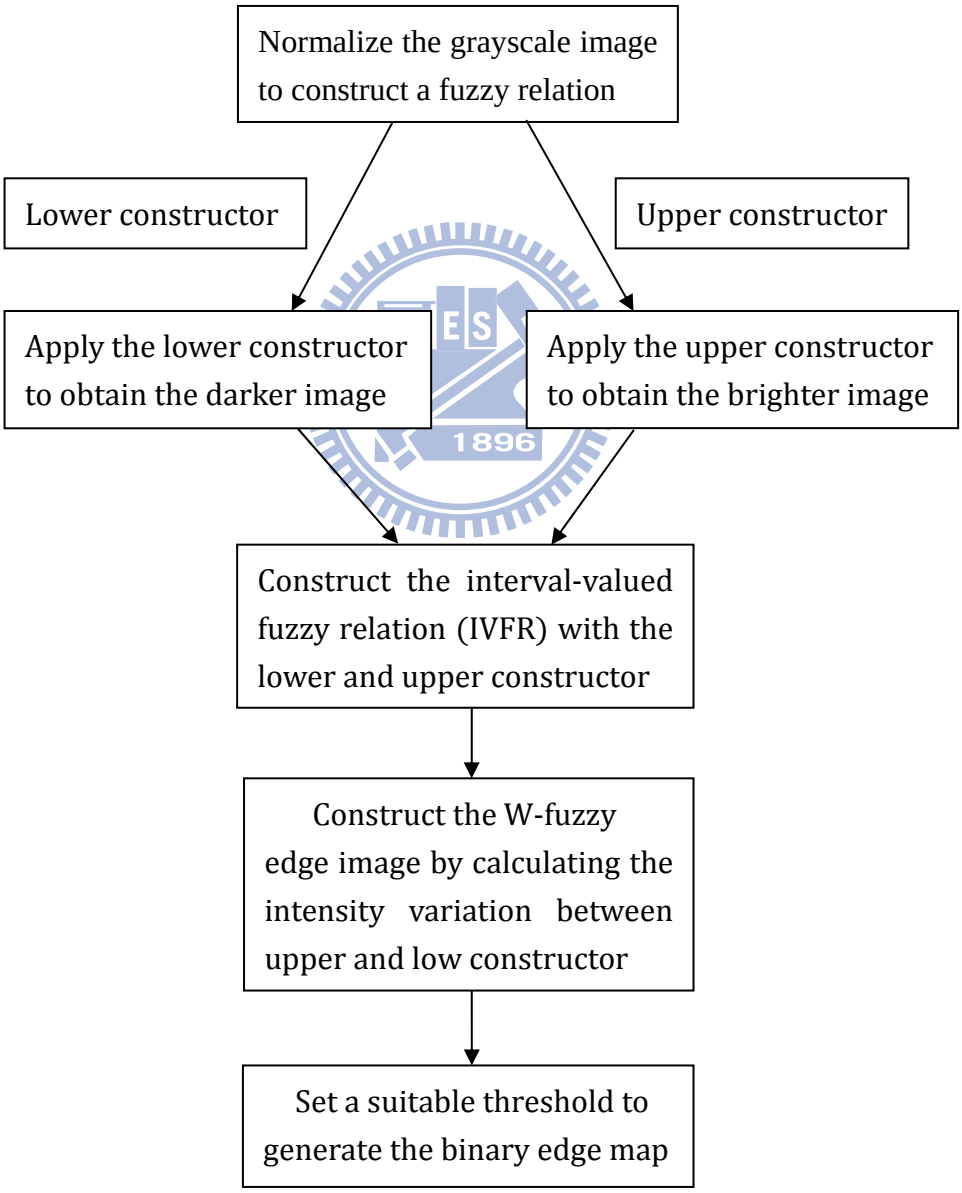


Fig. 2.1 The flow chart to obtain edges by using interval-valued fuzzy relation.

2.2 *Interval-Valued Fuzzy Relation*

Let us express by $L([0,1])$ the set of all closed subintervals of [0, 1], i.e.,

$$L([0,1]) = \{[\underline{x}, \overline{x}] \mid (\underline{x}, \overline{x}) \in [0,1]^2 \text{ and } \underline{x} \le \overline{x}\} \tag{2.2}$$

Then, $L([0,1])$ is a partially ordered set with regard to the relation $\le_L$ that is defined in the following way. Given $[\underline{x}, \overline{x}], [\underline{y}, \overline{y}] \in L([0,1])$,

$$[\underline{x}, \overline{x}] \le_L [\underline{y}, \overline{y}] \text{ if and only if } \underline{x} \le \underline{y}, \text{ and } \overline{x} \le \overline{y}. \tag{2.3}$$

2.2.1 Lower Constructor

A t-norm $T : [0,1]^2 \to [0,1]$ is an associative, commutative function which allows the extension of t-norm T to a k-ary operation by induction, defining for each k-tuple $(x_1, x_2, ..., x_k) \in [0,1]^k$ [16].

$$\underset{i=1}{\overset{k}{T}} x_i = T(\underset{i=1}{\overset{k-1}{T}} x_i, x_n) = T(x_1, x_2, ..., x_k). \tag{2.4}$$

The four basic t-norms are as follows:

(1) The minimum $T_M(x, y) = \min(x, y)$.

(2) The product $T_P(x, y) = x \cdot y$.

(3) The Lukasiewicz t-norm $T_L(x, y) = \max(x + y - 1, 0)$.

(4) The nilpotent minimum t-norm $T_{nM}(x, y) = \begin{cases} \min(x, y), & \text{if } x + y > 1 \\ 0, & \text{otherwise.} \end{cases}$

Let $R \in F(X \times Y)$ be an FR. Consider two t-norms T_1 and T_2 and two values $m, n \in \mathbb{N}$ so that $m < \frac{M-1}{2}$, and $n < \frac{N-1}{2}$. We define the lower constructor associated with T_1, T_2, m, and n in the following way:

$$L^{m,n}_{T_1,T_2}[R](x,y) = \overset{n}{\underset{\substack{i=-m\\ j=-n}}{\overset{m}{T_1}}} (T_2(R(x-i,y-j),\ R(x,y))) \tag{2.5}$$

For all $(x,y) \in X \times Y$, and where the indices i, j take values such that $0 \le x-i \le M-1$ and $0 \le y-j \le N-1$. The values of m and n indicate that the considered window centered at (x,y) is a matrix of dimension $(2m+1)\times(2n+1)$. We express $L^{m,n}_{T_1,T_2}$ as $L^{n}_{T_1,T_2}$, for simplicity, if $n = m$.

In Fig. 2.2, we graphically illustrate how the lower constructor operation works with $n = m = 1$, and $T_1 = T_2 = T_M$. For $(x_1, y_1) \in X \times Y$, we have:

$L^{1}_{T_M,T_M}[R](x_1, y_1)$ = min(min(0.23,0.44), min(0.42,0.44), min(0.49,0.44),
min(0.44,0.44), min(0.58,0.44), min(0.59,0.44),
min(0.56,0.44), min(0.77,0.44), min(0.56,0.44))
= 0.23

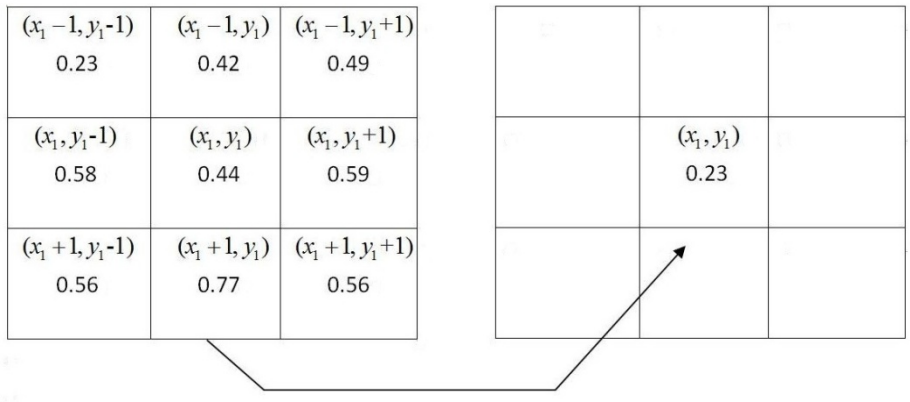


Fig. 2.2 Example of lower constructor operation.

2.2.2 Upper Constructor

Because the characteristics of t-conorms are similar to those of t-norms, we can, analogously, define another operator based upon t-conorms to construct the

upper bound of the intervals as the lower constructor does.

The four basic t-conorms are as follows:

(1) The maximum $S_M(x, y) = \max(x, y)$.

(2) The probabilistic sum $S_P(x, y) = x + y - x \cdot y$.

(3) The Lukasiewicz t-conorm $S_L(x, y) = \min(x + y, 1)$.

(4) $S_{nM}(x, y) = \begin{cases} \max(x, y), & \text{if } x + y < 1 \\ 0, & \text{otherwise.} \end{cases}$

Let $R \in F(X \times Y)$ be an FR. Consider two t-conorms S_1 and S_2 and two values $m, n \in \mathbb{N}$ so that $m < \frac{M-1}{2}$, and $n < \frac{N-1}{2}$. We define the upper constructor associated with S_1, S_2, m, and n in the following way:

$$U^{m,n}_{S_1,S_2}[R](x, y) = \underset{\substack{i=-m \\ j=-n}}{\overset{n}{\overset{m}{S_1}}} (S_2(R(x - i, y - j), R(x, y))) \tag{2.6}$$

For simplicity, we can also express $U^{m,n}_{S_1,S_2}$ as $U^{n}_{S_1,S_2}$, if $n = m$.

2.2.3 Construction of Interval-Value Fuzzy Relation

Let $R \in F(X \times Y)$ and consider a lower constructor $L^{m,n}_{T_1,T_2}$ and an upper constructor $U^{m,n}_{S_1,S_2}$. Then, the interval-value fuzzy relation $R^{m,n}$ can be denoted by

$$R^{m,n}(x, y) = [L^{m,n}_{T_1,T_2}[R](x, y), U^{m,n}_{S_1,S_2}[R](x, y)] \in L([0,1]) \tag{2.7}$$

for all $(x, y) \in X \times Y$ is an IVFR from X to Y.

If $n = m$, then we can denote $R^{m,n}$ as R^n.

2.2.4 W-Fuzzy Relations and W-Fuzzy Edges

In this section, we construct another FR, i.e., $W[R^{m,n}] \in F(X \times Y)$, so that

$$W[R^{m,n}] = \overline{R^{m,n}}(x,y) - \underline{R^{m,n}}(x,y) = U^{m,n}_{S1,S2}[R](x,y) - L^{m,n}_{T1,T2}[R](x,y) \tag{2.8}$$

for all $(x,y) \in X \times Y$, where X and Y are two sets $\{0,1,...,M\text{-}1\}$ and $\{0,1,...,N\text{-}1\}$. In image processing field, we call the fuzzy relation $W[R^{m,n}]$ a W-fuzzy edge image although the relation $W[R^{m,n}]$ does not show an edge image in the sense of Canny [1] due to the fact that $W[R^{m,n}]$ is a fuzzy one, not a sharp one.

The length of the interval indicates the intensity variation and represents the membership degree of each element to the new FR. Because the W-fuzzy edge image visually captures the intensity changes, we can view this image as an edge image which represents edge in a fuzzy way.

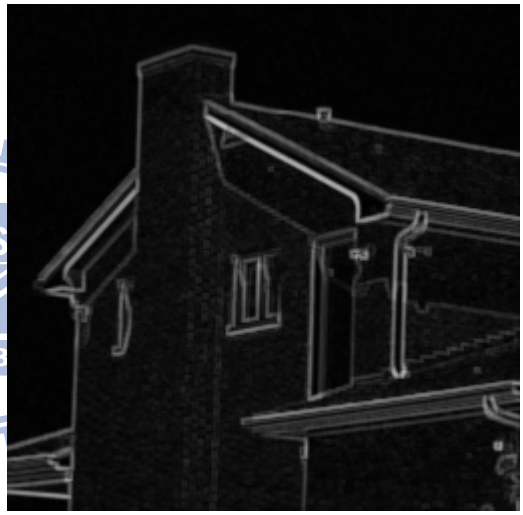
In Fig. 2.3, we show grayscale natural image and the images obtained by applying different lower and upper constructors to the original image and the W-fuzzy edge image which utilizes the constructors $U^1_{S_M,S_M}$ and $L^1_{T_M,T_M}$.

(a) (b)

(c)

(d)

(e)



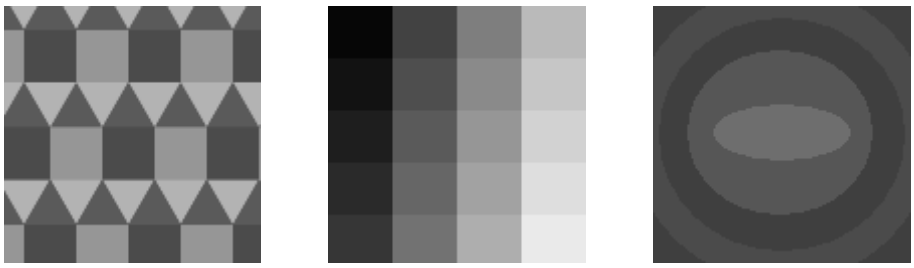
(f)

Fig. 2.3 (a) The "House" image. (b)−(c) Apply lower constructor $L^1_{T_M,T_M}$, $L^1_{T_P,T_P}$ to "House" image, respectively. (d)−(e) Apply upper constructor $\mathrm{U}^1_{S_M,S_M}$, $\mathrm{U}^1_{S_P,S_P}$ to "House" image, respectively.. (f) The W-fuzzy edge image obtained by using $\mathrm{U}^1_{S_M,S_M}$, $L^1_{T_M,T_M}$.

Chapter 3 Parameter Learning of Weighted Mean Based Edge Detection

In this chapter, we extend the concept of the W-fuzzy relation and propose the edge detection method through executing the weighted-mean aggregation to compute the difference between a central pixel and its 8-neighborhood pixels in a 3×3 sliding window across the image. In order to increase the edge detection accuracy, we have derived the parameter training formula which can be learned iteratively and the post-processing techniques, including non-maximum suppression and continuity reinforcement, introduced in the next chapter, to achieve better edge map.

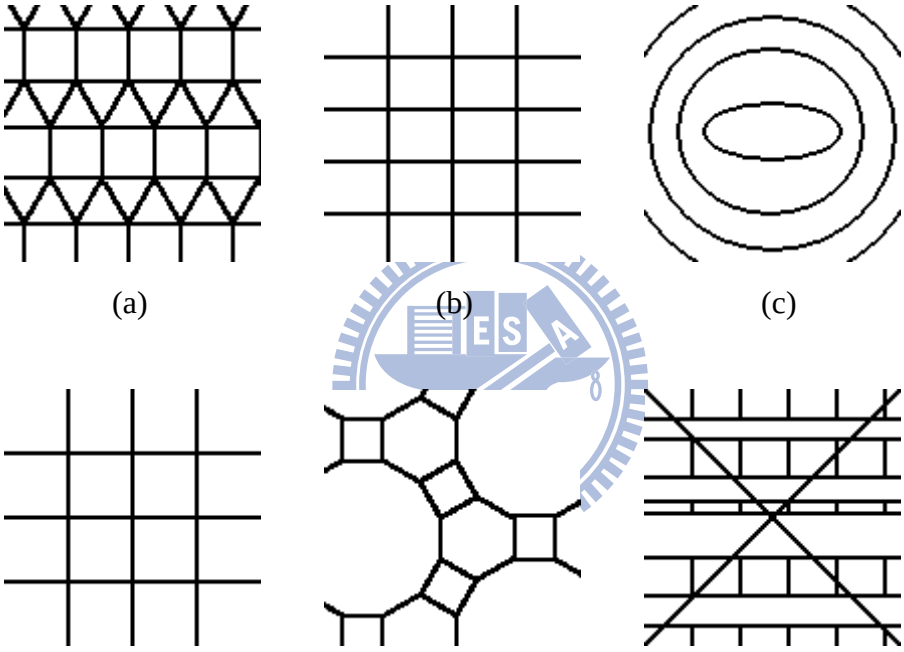
In the parameter learning phase, we use six synthetic 128×128 grayscale images, as shown in Fig. 3.1, as training input images for the parametric learning. Then, we update these three parameters to increase the edge accuracy of six images by the steepest decent method [17, 18] cast in discrete formulation, namely in a spirit similar to perceptron learning.



(a) (b) (c)

(d) (e) (f)

Fig. 3.1 Six gray-scale synthetic images with size of 128×128, for training.



(a) (b) (c)

(d) (e) (f)

Fig. 3.2 Edge ground truths of six gray-scale synthetic images.

3.1 Weighted Mean Based Interval-Valued Fuzzy Relation

For the method proposed by Barrenechea et al. [13, 14], there are totally sixteen combinations of choices to construct the interval-valued fuzzy relations due to four selections for the *t*-norm and *s*-norm operators, respectively. It is difficult to decide

which combination is the most suitable combination to fit the image edge extraction. Besides, the *t*-norm and *s*-norm operators are nonlinear functions and not easy to derive learning formula involving the "max" and "min" logical operators. To overcome these disadvantages, we will introduce the weighted-mean aggregation to generalize the *t*-norm and *s*-norm formulation in a continuous setting. To this end, two parameters in the weighted-mean aggregation, the *t*-type α_t and *s*-type α_s, are proposed to replace the *t*-norm and *s*-norm operators in constructing the interval-valued fuzzy relations for edge detection.

3.2 The Weighted Mean to Calculate the Difference of Neighbor Pixels of Images

In this section, we utilize the weight parameters α_t and α_s to determine the intensity of *t*-type and *s*-type operations and calculate the weighted mean difference for the pixel values of an image. Furthermore, we derive the iterative learning formula for weight parameters so that we can obtain the best parameter set with optimal capability in edge detection. The "linear" and "quadratic" types weighted mean difference calculation of each pixel in the image is introduced below.

3.2.1 The Linear Weighted Mean Edge Detection

For an $M \times N$ image I, by applying the generalized weighted mean operands α_t and α_s, as shown in Fig. 3.3, we calculate the linear weighted mean of the

grayscale intensity between pixel $I(m,n)$ and its 8-neighbor pixels in a 3×3 window centered at pixel $I(m,n)$.

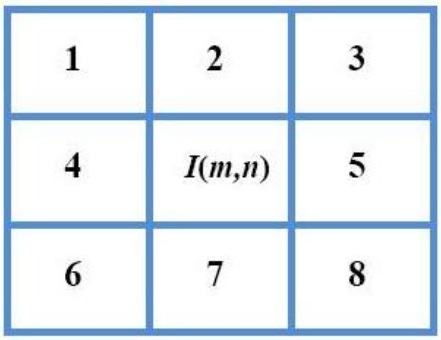


Fig. 3.3 The diagram of a 3×3 window centered at pixel $I(m,n)$ and its 8-neighborhood pixels

$$\begin{cases} y_{ti}(I(m,n)) = \alpha_t a_i(I(m,n)) + (1-\alpha_t) b_i(I(m,n)) \\ y_{si}(I(m,n)) = \alpha_s a_i(I(m,n)) + (1-\alpha_s) b_i(I(m,n)) \end{cases} \tag{3.1}$$

$\alpha_t \in [0, 0.5];\ \alpha_s \in [0.5, 1];\ i = 1, 2, ..., 8;\ m = 1, 2, ..., M;\ n = 1, 2, ..., N.$

Where i indicates the index of the eight neighbor pixels and the mean weighting parameters, α_t and α_s, satisfying $\alpha_t < \alpha_s$; Comparing the center pixel $I(m,n)$ with its i th neighbor pixels, $a_i(I(m,n))$ and $b_i(I(m,n))$ represent the larger and the smaller gray value, respectively. Moreover, we compute the average value of eight y_{ti} and y_{si} as follows:

$$\overline{y_t}(m,n) = \frac{\sum_{i=1}^{8} y_{ti}(I(m,n))}{8}, \qquad \overline{y_s}(m,n) = \frac{\sum_{i=1}^{8} y_{si}(I(m,n))}{8}. \tag{3.2}$$

Finally, for each image pixel, we can obtain an output value y_W which is similar to the meaning of W-fuzzy relation as follows:

$$y_W(m,n) = \overline{y_s}(m,n) - \overline{y_t}(m,n)$$

$$= \frac{(\alpha_s - \alpha_t)\sum_{i=1}^{8}(a_i(I(m,n)) - b_i(I(m,n)))}{8}. \tag{3.3}$$

Pixel $I(m,n)$ will be deemed as an edge pixel if $y_W(m,n)$ is larger than threshold T, otherwise, it is a non-edge pixel.

$$y(m,n) = \begin{cases} 1 \text{ (edge)}, & \text{if } y_W(m,n) - T \geq 0 \\ 0 \text{ (non-edge)}, & \text{otherwise} \end{cases} \tag{3.4}$$

3.2.2 The Quadratic Weighted Mean Edge Detection

To strengthen the effect of the weighted mean difference between the center pixel $I(m,n)$ and its neighbors, we replace $a_i(I(m,n))$ and $b_i(I(m,n))$ by $a^2{}_i(I(m,n))$ and $b^2{}_i(I(m,n))$, respectively, and rewrite Equations (3.1) – (3.4) so that the difference range of the weighted-mean aggregation can be broaden.

The quadratic weighted mean of the grayscale intensity between pixel $I(m,n)$ and its 8-neighbor pixels can be indicated as follows:

$$\begin{cases} y_{ti}(I(m,n)) = \alpha_t a_i{}^2(I(m,n)) + (1-\alpha_t) b_i{}^2(I(m,n)) \\ y_{si}(I(m,n)) = \alpha_s a_i{}^2(I(m,n)) + (1-\alpha_s) b_i{}^2(I(m,n)) \end{cases} \tag{3.5}$$

where $\alpha_t \in [0, 0.5]$; $\alpha_s \in [0.5, 1]$; $i = 1, 2, ..., 8$; $m = 1, 2, ..., M$; $n = 1, 2, ..., N$. Analogously, we calculate the average of the eight y_{ti} and y_{si} and define the output value y_W for each pixel $I(m,n)$ as follows:

$$\overline{y_t}(m,n)=\frac{\sum_{i=1}^{8} y_{ti}(I(m,n))}{8}, \qquad \overline{y_s}(m,n)=\frac{\sum_{i=1}^{8} y_{si}(I(m,n))}{8}. \tag{3.6}$$

$$y_W(m,n)=\sqrt{\overline{y_s}(m,n)-\overline{y_t}(m,n)}=\sqrt{\frac{(\alpha_s-\alpha_t)\sum_{i=1}^{8}(a_i^{\,2}(I(m,n))-b_i^{\,2}(I(m,n)))}{8}}. \tag{3.7}$$

$$y(m,n)=\begin{cases}1 \text{ (edge)}, & \text{if } y_W(m,n)-T\geq 0\\ 0 \text{ (non-edge)}, & \text{otherwise.}\end{cases} \tag{3.8}$$

3.3 Operating Parameter Learning Mechanism

Before processing the parameter learning mechanism, we have to give a set of three initial operating parameters (α_s, α_t and T) which can be learned iteratively. One of the merits of our learning mechanism is that we formulate the problem as the parameter training procedure leading to better edge map accuracy. The restrictions we must abide by in setting initial parameters are as follows:

(1) $\alpha_s=[0.5,1]$; $\alpha_t=[0,0.5]$

(2) $T\in\mathbb{R}^+$

In addition, we utilize the concept of perceptron learning procedure to update parameters and implement the pocket algorithm to ensure that we can obtain the best parameter set during the course of training.

3.3.1 Perceptron Learning Algorithm

The perceptron learning is an algorithm which can iteratively learn the weight of a linear prediction function of the feature vector to best dichotomize a two-class problem. It is a type of linear classifier for supervised classification of an input into one of two classes, and can be easily extended to multiple class classification. The learning algorithm for perceptrons was developed in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt [19] and it processes elements in the training set one at a time. The perceptron is a binary classifier which maps its input x (a real-valued vector) to an output value $f(x)$ (a signum function):

$$f(x) = \begin{cases} 1, & \text{if } w \cdot x + b > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

where w is a vector of real-valued weights, $w \cdot x$ is the dot product (which computes a weighted sum), and b is the "bias", a constant term which is independent of any input value.

The perceptron algorithm is also termed the single-layer perceptron that is the simplest feedforward neural network and the learning step can not stop if the learning set is not linearly separable. The most famous example with linearly non-separable vectors is the Boolean exclusive-or problem [20].

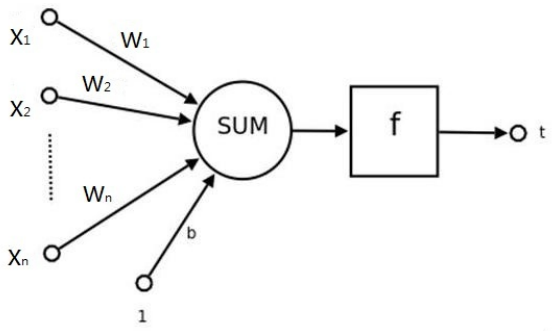


Fig. 3.4 The example of a learning algorithm for a single-layer perceptron.

For a single-layer perceptron with *n*-dimensional inputs, $f(\cdot)$ is the activation function and the output can be defined as follows:

$$t = f[w_1x_1 + w_2x_2 + \cdots + w_nx_n + b] = f(\sum_{i=1}^{n} w_ix_i + b) \tag{3.10}$$

$$= f(\mathrm{w}^T\mathrm{x}).$$

where $\mathrm{w} = [w_1\ w_2\ \ldots\ w_n\ b]^T$; $\mathrm{x} = [x_1\ x_2\ \ldots\ x_n\ 1]^T$; $f(n) = \begin{cases} +1, & \text{if } n \geq 0 \\ -1, & \text{otherwise.} \end{cases}$

The error function e is defined by

$$e = \frac{1}{p}\sum_{i=1}^{n}(d_i - t_i)^p, \tag{3.11}$$

where $p = 1$ for L_1 norm and d_i is the desired output value of the i th input x_i

Then, we update the weights by:

$$w_{i\_new} = w_{i\_old} + \alpha(d_i - t_i)x_i, \quad i = 1,\ldots,n. \tag{3.12}$$

The algorithm updates the weights immediately after Eqs. (3.10) and (3.12) are applied to a parameter set in the training set.

3.3.2 Learning Rule of Linear Mean Weighting Parameters

For a given $M \times N$ image, in order to obtain the smallest number of wrongly classified edge and non-edge pixels, we calculate the total error $e(m,n)$ of incorrect edge and non-edge pixels at (m,n) as follows.

$$e(m,n) = \frac{1}{2}\sum_{p=1}^{M}\sum_{q=1}^{N}(d(p,q) - y(p,q))^2 \tag{3.13}$$

where $d(p,q)$ indicates the ground truth edge map at (p,q) of the input image, and $y(p,q)$ denotes the output edge map at (m,n) of the input image.

To begin with, the derivative of e with regard to α_s is given by

$$\frac{\partial e(m,n)}{\partial \alpha_s} = \frac{\partial e(m,n)}{\partial y(m,n)} \frac{\partial y(m,n)}{\partial y_W(m,n)} \frac{\partial y_W(m,n)}{\partial \alpha_s}$$

$$= \frac{\partial \left(\frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{N}(d(m,n)-y(m,n))^2\right)}{\partial y(m,n)} \frac{\partial \left(\frac{(\alpha_s-\alpha_t)\sum_{i=1}^{8}(a_i(I(m,n))-b_i(I(m,n)))}{8}-T\right)}{\partial \alpha_s}$$

$$= -(d(m,n)-y(m,n))\frac{\sum_{i=1}^{8}(a_i(I(m,n))-b_i(I(m,n)))}{8}. \quad (3.14)$$

Therefore, the weighting parameter α_s can be learned iteratively by equation (3.12).

$$\alpha_{s\_new} = \alpha_{s\_old} - \eta_s \frac{\partial e(m,n)}{\partial \alpha_s}$$

$$= \alpha_{s\_old} + \eta_s (d(m,n)-y(m,n))\frac{\sum_{i=1}^{8}(a_i(I(m,n))-b_i(I(m,n)))}{8}, \quad (3.15)$$

where η_s is the learning constant of α_s. Similarly, a perceptron learning of the other two parameters, α_s and T, iteratively, can also be given by

$$\frac{\partial e(m,n)}{\partial \alpha_t} = \frac{\partial e(m,n)}{\partial y(m,n)} \frac{\partial y(m,n)}{\partial y_W(m,n)} \frac{\partial y_W(m,n)}{\partial \alpha_t}$$

$$= (d(m,n)-y(m,n))\frac{\sum_{i=1}^{8}(a_i(I(m,n))-b_i(I(m,n)))}{8}, \quad (3.16)$$

$$\alpha_{t\_new} = \alpha_{t\_old} - \eta_t \frac{\partial e(m,n)}{\partial \alpha_t}$$

$$= \alpha_{t\_old} - \eta_t (d(m,n)-y(m,n))\frac{\sum_{i=1}^{8}(a_i(I(m,n))-b_i(I(m,n)))}{8}, \quad (3.17)$$

$$\frac{\partial e(m,n)}{\partial T} = \frac{\partial e(m,n)}{\partial y(m,n)} \frac{\partial y(m,n)}{\partial T}$$

$$= -(d(m,n) - y(m,n))(-1) = d(m,n) - y(m,n), \tag{3.18}$$

$$T_{new} = T_{old} - \eta_T \frac{\partial e(m,n)}{\partial T} = T_{old} - \eta_T (d(m,n) - y(m,n)). \tag{3.19}$$

where η_t and η_T represent the learning constant of α_t and T, respectively.

3.3.3 Learning Rule of Quadratic Mean Weighting Parameters

Analogously, from Eq. (3.8) and Eq. (3.13), we can derive the learning formulas for quadratic mean weighting form as follows:

$$\frac{\partial e(m,n)}{\partial \alpha_s} = \frac{\partial e(m,n)}{\partial y(m,n)} \frac{\partial y(m,n)}{\partial y_W(m,n)} \frac{\partial y_W(m,n)}{\partial \alpha_s}$$

$$= \frac{\partial \left(\frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{N}(d(m,n) - y(m,n))^2\right)}{\partial y(m,n)} \frac{\partial \left(\sqrt{\frac{(\alpha_s - \alpha_t)\sum_{i=1}^{8}(a_i^2(I(m,n)) - b_i^2(I(m,n)))}{8}} - T\right)}{\partial \alpha_s}$$

$$= \frac{-(d(m,n) - y(m,n))}{2\sqrt{\frac{(\alpha_s - \alpha_t)\sum_{i=1}^{8}(a_i^2(I(m,n)) - b_i^2(I(m,n)))}{8}}} \frac{\sum_{i=1}^{8}(a_i^2(I(m,n)) - b_i^2(I(m,n)))}{8}$$

$$= -\frac{1}{4}(d(m,n) - y(m,n))\sqrt{\frac{\sum_{i=1}^{8}(a_i^2(I(m,n)) - b_i^2(I(m,n)))}{2(\alpha_s - \alpha_t)}}, \tag{3.20}$$

$$\frac{\partial e(m,n)}{\partial \alpha_t} = \frac{\partial e(m,n)}{\partial y(m,n)} \frac{\partial y(m,n)}{\partial y_W(m,n)} \frac{\partial y_W(m,n)}{\partial \alpha_t}$$

$$= \frac{1}{4}(d(m,n) - y(m,n)) \sqrt{\frac{\sum_{i=1}^{8} (a_i^2(I(m,n)) - b_i^2(I(m,n)))}{2(\alpha_s - \alpha_t)}}, \tag{3.21}$$

$$\frac{\partial e(m,n)}{\partial T} = \frac{\partial e(m,n)}{\partial y(m,n)} \frac{\partial y(m,n)}{\partial T} = d(m,n) - y(m,n). \tag{3.22}$$

A perceptron learning of the three parameters can be given by

$$\alpha_{s\_new} = \alpha_{s\_old} - \eta_s \frac{\partial e(m,n)}{\partial \alpha_s}$$

$$= \alpha_{s\_old} + \eta_s \frac{d(m,n) - y(m,n)}{4} \sqrt{\frac{\sum_{i=1}^{8} (a_i^2(I(m,n)) - b_i^2(I(m,n)))}{2(\alpha_s - \alpha_t)}}, \tag{3.23}$$

$$\alpha_{t\_new} = \alpha_{t\_old} - \eta_t \frac{\partial e(m,n)}{\partial \alpha_t}$$

$$= \alpha_{t\_old} - \eta_t \frac{d(m,n) - y(m,n)}{4} \sqrt{\frac{\sum_{i=1}^{8} (a_i^2(I(m,n)) - b_i^2(I(m,n)))}{2(\alpha_s - \alpha_t)}}, \tag{3.24}$$

$$T_{new} = T_{old} - \eta_T \frac{\partial e(m,n)}{\partial T} = T_{old} - \eta_T (d(m,n) - y(m,n)). \tag{3.25}$$

3.3.4 Pocket Algorithm

It is understandable that the learned edge map will be different from the ground truth edge map of synthetic images, i.e., different edge/non-edge pixel classification for whole training synthetic images. With this concept in mind, the error $e(m,n)$ defined in Eq. (3.13) can not be zero, and hence the perceptron learning can not stop by itself. Instead, we utilize the pocket algorithm in the course of edge map training with the best solution set of α_t and α_s being confined in the interval [0, 1] for synthetic training images. For each training image, we train each pixel in a raster-scan

fashion, and then test the parameter set in the end of row. If the current trained parameters α_t, α_s and T can result in higher edge average accuracy than the best average accuracy of the parameter set stored in the pocket, then the current parameter set will be stored in the pocket to replace the worse old ones. The current parameters are the initial parameters for the beginning pixel of the next row. On the other hand, the current parameter set does not produce a better accuracy of edge maps of all training images, then the best parameter set saved in the pocket need not change, and this pocket parameter set is used for the initial value of the first pixel of the next row. In this way, we can find the final optimal parameter set after a long enough learning epochs.

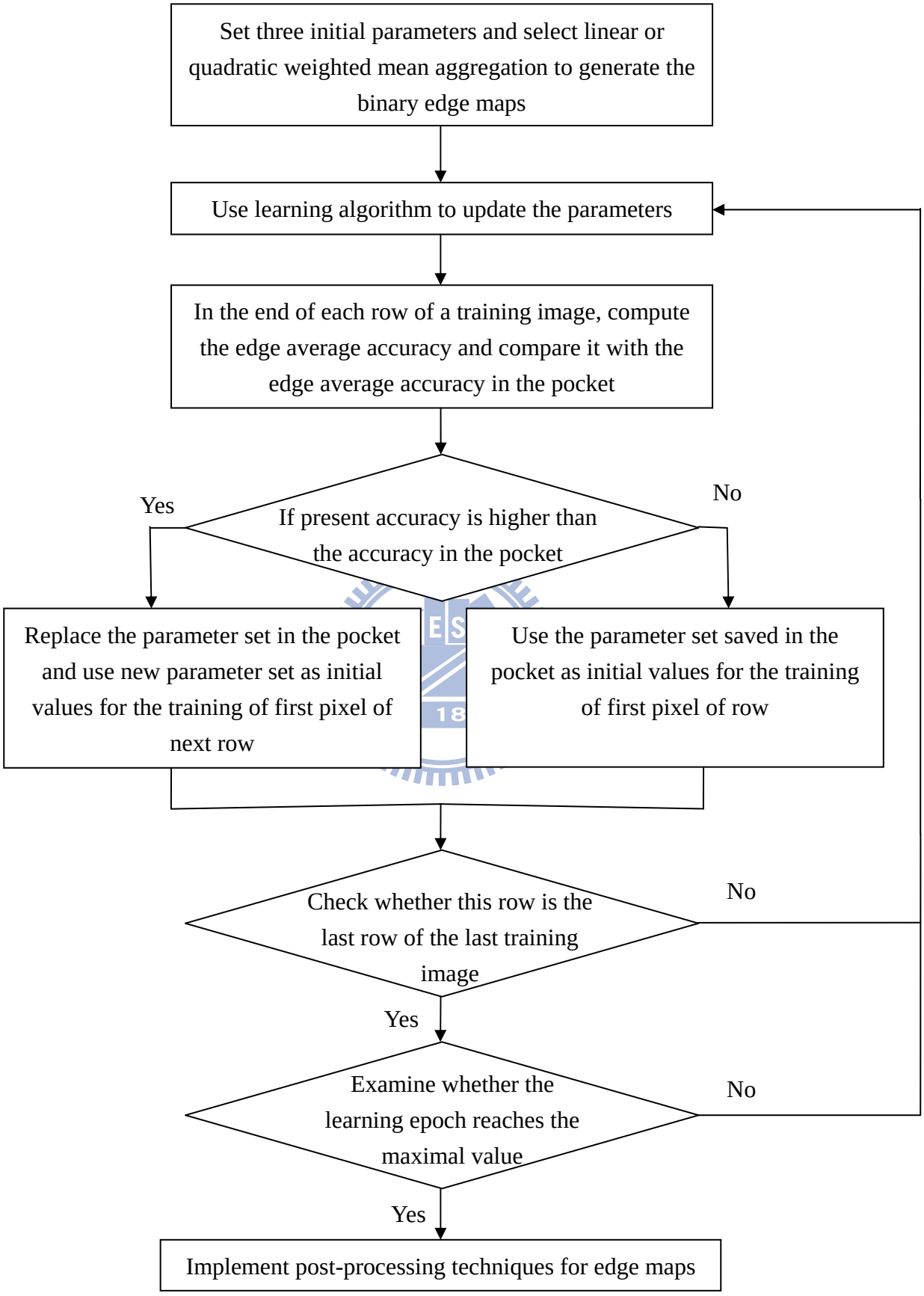


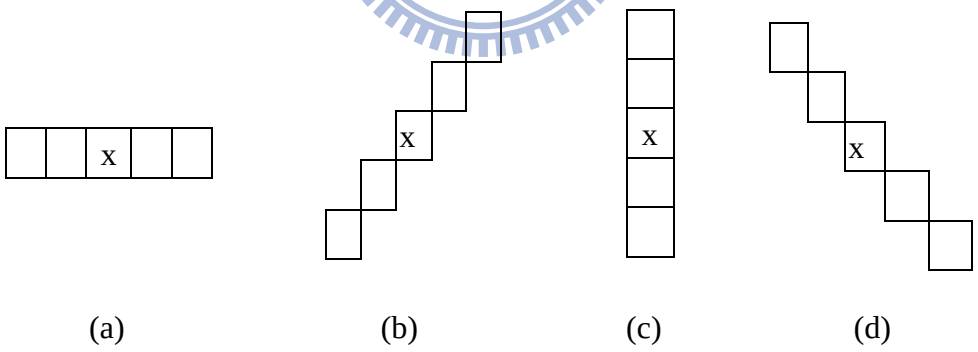
Fig. 3.5 The flow chart of parameter learning algorithm.

Chapter 4 Post-Processing Techniques

After obtaining the best parameter set which generates the edge map, we apply three special methods, including new directional non-maximum suppression, continuity reinforcement and isolated pixel removal, to enhance the accuracy and continuity of our weighted mean aggregation in edge detection.

4.1 Directional Non-Maximum Suppression

Non-maximum suppression is a process which makes all pixels, whose gray level intensity is not maximal, as zero within neighbors in a certain direction. Fig. 4.1 shows four directional arrays of linear windows at angles $0^{\circ}, 45^{\circ}, 90^{\circ}$ and 135°.



(a) (b) (c) (d)

Fig. 4.1 Illustration of several linear windows at different directions. "x" indicates the center pixel of the linear window.

4.1.1 Direction of Image Edge

In order to decide the direction that the edge pixels belong to, we apply Sobel operator [2], shown in Fig. 4.2, to the fuzzy image y_W which is obtained by Eqs. (3.3)

and (3.7). For each pixel, we can obtain two vector components and compute the corresponding normal angle so that we determine the direction of pixel that is perpendicular to its normal vector.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

(a)

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

(b)

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

(c)

Fig. 4.2 (a) The corresponding coordinate of matrix (z_5 is the working pixel), (b) The Sobel operator for horizontal edge, (c) The Sobel operator for vertical edge

By using Sobel operator, we can obtain two vector components, g_x and g_y, of each pixel as follows:

$$g_x = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{4.1}$$

$$g_y = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{4.2}$$

For a $M \times N$ image, we calculate the normal angle of edge pixels by

$$\alpha(m,n) = \tan^{-1}[\frac{g_y}{g_x}] \quad \text{(unit: radian)} \tag{4.3}$$

Then, we convert radian to degree by

$$\phi(m,n) = \frac{180}{\pi}\tan^{-1}[\frac{g_y}{g_x}] \quad \text{(unit: degree)} \tag{4.4}$$

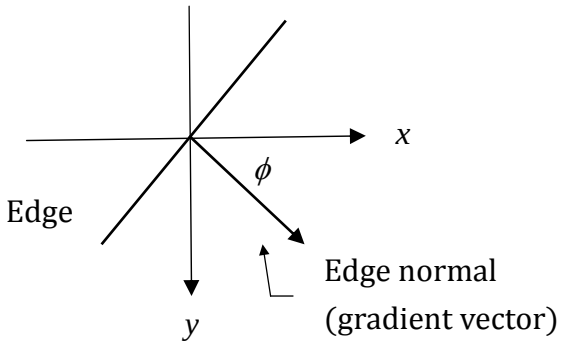


Fig. 4.3 The edge and its gradient vector.

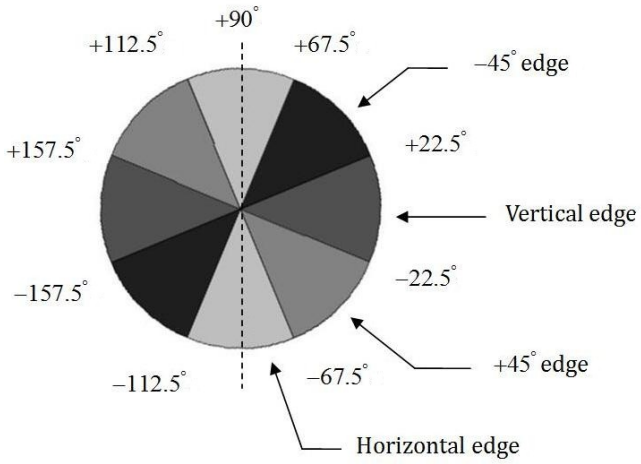


Fig. 4.4 The corresponding group for the angle ϕ of gradient vector.

From Fig. 4.4 [2], we divide the directions of edge pixels into four groups as follows:

(1) The pixel belongs to horizontal edge when its normal angle is in the interval $[-67.5^{\circ}, -112.5^{\circ}]$ or $[+67.5^{\circ}, +112.5^{\circ}]$.

(2) The pixel belongs to vertical edge when its normal angle is in the interval $[0^{\circ}, +22.5^{\circ}]$ or $[-22.5^{\circ}, 0^{\circ}]$ or $[+157.5^{\circ}, +180^{\circ}]$ or $[-157.5^{\circ}, -180^{\circ}]$.

(3) The pixel belongs to $+45^{\circ}$ edge when its normal angle is in the interval $[-22.5^{\circ}, -67.5^{\circ}]$ or $[+112.5^{\circ}, +157.5^{\circ}]$.

(4) The pixel belongs to -45° edge when its normal angle is in the interval $[-112.5^{\circ}, -157.5^{\circ}]$ or $[+22.5^{\circ}, +67.5^{\circ}]$.

4.1.2 Additional Parameter for Directional Non-Maximum Suppression

Directional Non-maximum suppression, abbreviated as DNMS, maintains the y_W of the pixel which assumes a local maximum in the gradient direction, i.e., if the central pixel has the largest y_W value in the corresponding gradient direction of linear window, we keep its y_W value. Otherwise, we set its value to zero. The effect of non-maximum suppression is stronger if the size of linear window array is larger.

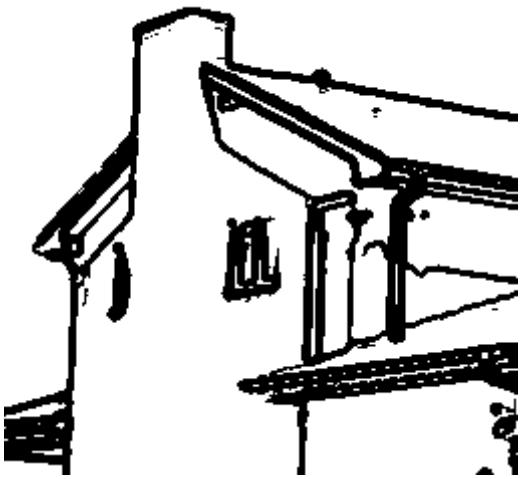
Although the non-maximum suppression can thin the edges, it destroys the line connectivity of image containing complicated and concentrated edges. This phenomenon becomes more severe if the size of the linear window array is large. So we set up an additional parameter, R_{NMS}, and define the formula as follows:

$$y_{strong}(m,n) = \begin{cases} y_W(m,n), & \text{if } y_W(m,n) \geq T \text{ and } y_W(m,n) \geq R_{NMS} \cdot M(y_W(m,n)) \\ 0, & \text{otherwise.} \end{cases} \quad (4.5)$$

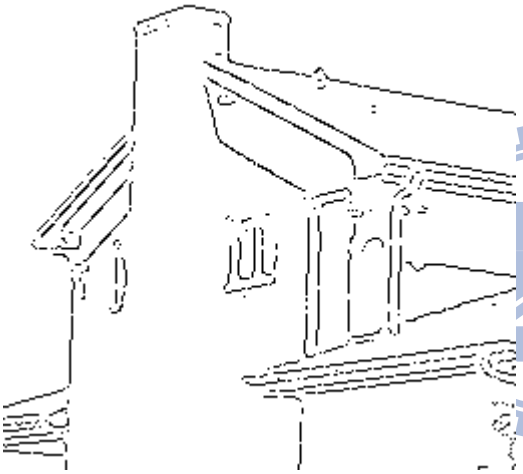
where $M(y_W(m,n))$ indicates the maximal y_W in the corresponding gradient direction of linear window centered at $y_W(m,n)$.

Fig. 4.5(b) shows that the edge map obtained by y_W is too sensitive to distinguish the thick edge map. Fig. 4.5(c)−(f) shows the difference between the original and new non-maximum suppression with different linear window. In Fig. 4.5(c) and (e), although the original DNMS can thin the edges into one pixel width, it destroys edge connectivity seriously. But the edge connectivity in Fig. 4.5(d) and (f) performs better due to the additional parameter R_{NMS}. In contrast, the width of edge is much thicker in Fig. 4.5(d) and (f), so we have to use thinning algorithm after the continuity reinforcement to get a proper image with thinner edge.

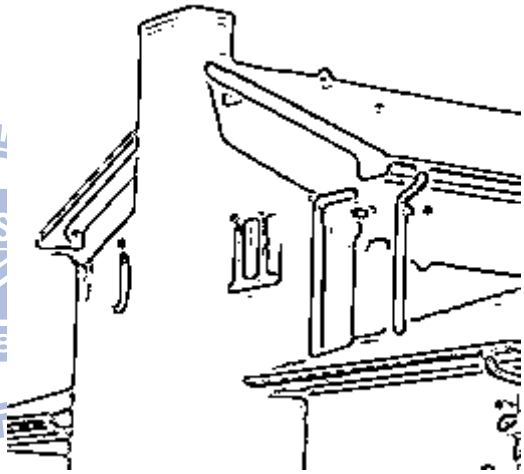
(a)



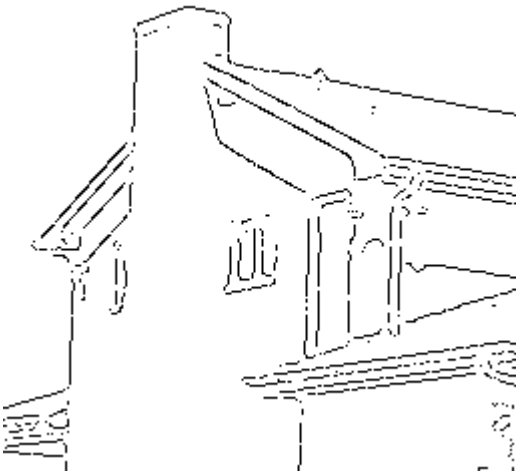
(b)



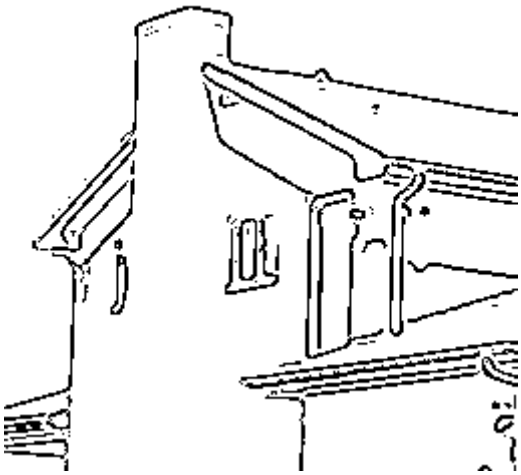
(c)



(d)



(e)



(f)

Fig. 4.5 (a) The “House” image. (b) Binary edge map obtained by thresholding the fuzzy image y_W with T=0.662, (c) Edge map obtained by original DNMS with $R_{NMS}=1$, window size=3, (d) Edge map obtained by new DNMS with $R_{NMS}=0.8$, window size=3, (e) Edge map obtained by original DNMS with $R_{NMS}=1$, window size=5, (f) Edge map obtained by original DNMS with $R_{NMS}=0.8$, window size=5.

4.2 Continuity Reinforcement

After executing the new DNMS, we use another technique to enhance the continuity due to the disconnecting edges, especially in edge intersections. The concept of this algorithm is that, for those pixels eliminated owing to non-maximum suppression, the probability to convert them to edges has a great relationship to their neighborhood. For the pixel removed by DNMS, if the number of surrounded edge pixels is larger, the limit to change it into edge point is lower. The following procedure will invoke some non-maximal suppressed pixels back to its original $y_W(m,n)$ value. Since a higher number in the surrounding edge pixel will have a good chance to invoke the pixel $y_W(m,n)$ value back, and it can run the procedure for whole image *maxNum* times.

Here are the steps of our continuity reinforcement algorithm.

Step 1: Set initial values: *num* =1, *contact* = 8, $R_{con} = 0.8$.

Step 2: For each pixel which $y_{strong}(m,n) = 0$, if the number of nonzero y_{strong} value in its 8-neighbors is equal to *contact*, we calculate the average y_{strong} value of those nonzero pixels and define the average value as *mean*.

Step 3: If $y_W(m,n) \geq R_{con} \cdot mean$, we mark this pixel.

Step 4: Do Step 2 and Step 3 until all pixels in the image are executed, then

contact = *contact* – 1, $R_{con} = R_{con} + 0.02$.

Step 5: If *contact* > 0, go to Step 2.

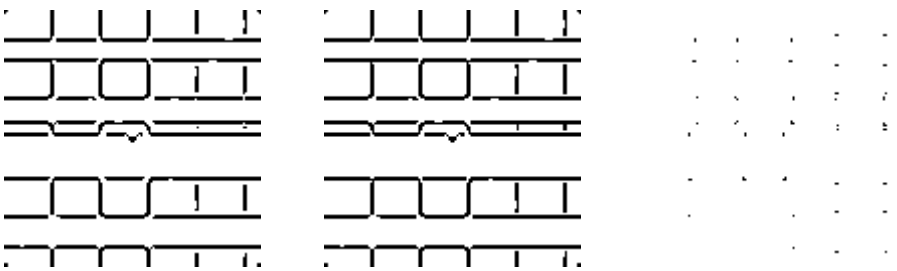
Else, $y_{strong}(m,n) = y_W(m,n)$, for all marked pixels.

Step 6: If *num* < *maxNum*, *num* = *num* + 1, *contact* = 8, $R_{con} = 0.8$, and go to Step 2.

Step 7: Convert the pixels whose y_{strong} is not zero to 1 (edge), others to 0 (non-edge).

where *maxNum* indicates the total executive times of this algorithm on the target image. Because of the different types of input images, we can freely regulate the two parameters, *maxNum* and R_{con}, to obtain the best result.

Fig. 4.6 (c) shows that the continuity reinforcement which strengthens the edge is effective especially in edge intersections.



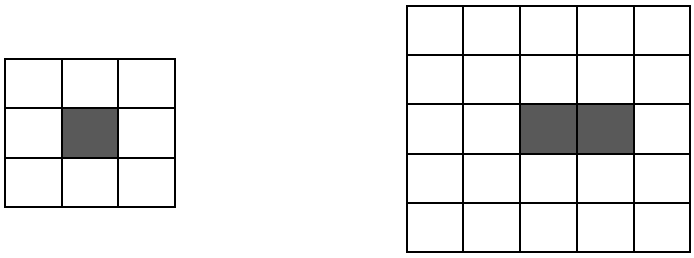
(a) (b) (c)

Fig. 4.6 (a) Image obtained only by new DNMS ($R_{NMS} = 0.8$), (b) Image obtained by new DNMS and continuity reinforcement ($R_{NMS} = 0.8$, *maxNum*=5). (c) Image obtained by subtracting (a) from (b).

4.3 Isolated Pixel Removal

In order to eliminate the isolated pixels, we first mark the edge pixels surrounded by non-edge pixels in a 3×3 sliding window, as shown in Fig. 4.7(a), and then label the edge pixels having only one edge pixels in its neighborhood in a 5×5 sliding window, as shown in Fig. 4.7(b). Finally, we convert all marked pixels to non-edge point after scanning the whole binary image obtained by continuity reinforcement.

In summary, we use thinning algorithm [21] so as to obtain the simplified edge image as canny edge detector does. Fig. 4.8 shows the flow chart of all the steps to obtain binary edge images by using the parameter learning of weighting mean and the post-processing techniques.



(a) (b)

Fig. 4.7 Examples of central isolated edge pixels which must be eliminated. (a) Isolated pixel in 3×3 window. (b) Isolated pixels in 5×5 window.

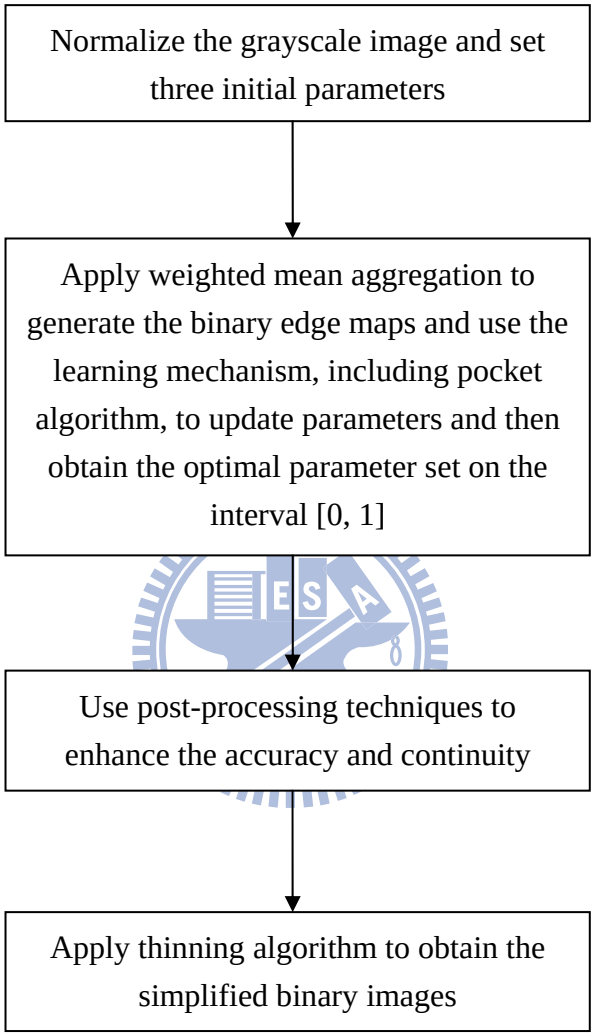
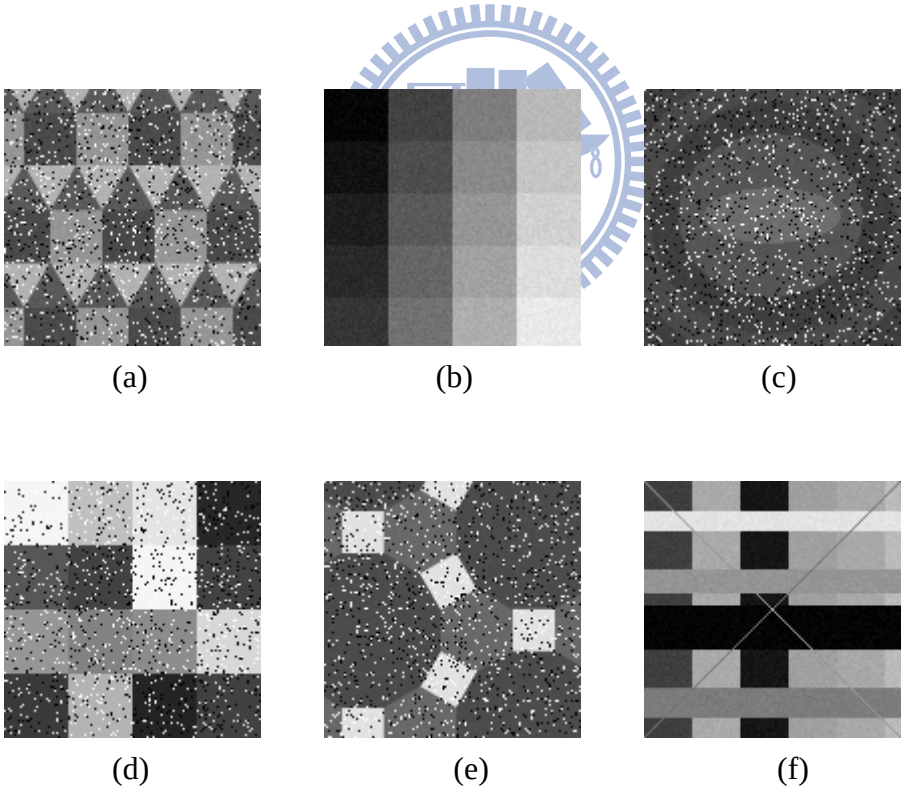


Fig. 4.8 The flow chart of all the steps to obtain binary edge images.

Chapter 5 Experimental Results

In our experiment, in order to imitate the natural images, we add different types and proportions of random noises, as shown in Fig. 5.1 with noisy corrupted strength shown in caption figure, to six grayscale synthetic images of Fig. 3.1, respectively. Before proceeding to the edge detection, we first executing the preprocessing procedures including a 3×3 median filter and than a Gaussian filter of $\sigma=1.2$ to remove the noise pixels. We apply our parameter learning of weighted mean introduced in chapter 3 to the six filtered synthetic noisy images, as shown in Fig. 5.2.



(a) (b) (c)

(d) (e) (f)

Fig. 5.1 Six grayscale synthetic images mixed with different types and proportions of random noise, (a) 10% impulse noise and Gaussian noise ($\mu=0,\ \sigma=4$), (b) Gaussian noise ($\mu=0,\ \sigma=5$), (c) 10% impulse noise and Gaussian noise ($\mu=0,\ \sigma=3$), (d) 10% impulse noise, (e) 10% impulse noise

and Gaussian noise ($\mu = 0,\ \sigma = 2$), (f) Gaussian noise ($\mu = 0,\ \sigma = 4$).

(a) (b) (c)

(d) (e) (f)

Fig. 5.2 The results of six noisy synthetic images through a 3×3 median filter and than a Gaussian filter of $\sigma = 1.2$.

5.1 Accuracy Calculation

Because we can accurately define the ground truth images of synthetic images, as shown in Fig. 3.2, we calculate the edge average accuracy of result images obtained by our parameter learning of *s*-type and *t*-type weighted mean algorithm. Fig. 5.3 shows the edge classification of each pixel in the image, which may be differentiated into four groups concerning accuracy.

		Ground Truth	
		Edge	Non-edge
Edge Outcome	Edge	True Positive (*TP*)	False Positive (*FP*)
	Non-edge	False Negative (*FN*)	True Negative (*TN*)

Fig. 5.3 The edge classification result of each pixel in the image.

From Fig. 5.3, we can obtain the average accuracy $\bar{a}_{cc}^{(r)}$ of the synthetic images by

$$\text{Average Accuracy } \bar{a}_{cc}^{(r)} = \frac{(\text{Edge accuracy}) + r(\text{Non-edge accuracy})}{1+r} \times 100\% \qquad (5.1)$$

$$\text{Edge accuracy} = \frac{\text{the number of } TP \text{ pixels}}{\text{the number of } TP \text{ pixels} + \text{the number of } FN \text{ pixels}},$$

$$\text{Non-edge accuracy} = \frac{\text{the number of } TN \text{ pixels}}{\text{the number of } TN \text{ pixels} + \text{the number of } FP \text{ pixels}}$$

where r indicates the weighting number for non-edge accuracy.

The average ratio of non-edge points to edge points in the ground truth images, as shown in Fig. 3.2, is 7.34 to 1, i.e., the importance of one edge pixel is 7.34 times important than the non-edge pixel if we consider the average accuracy with $r = 1$. To counter-balance the difference in the numbers of edge and non-edge pixels, we compute the average accuracy with different r to increase the weighting of edge pixels for all training images and use the pocket algorithm to select the best parameter set in the parameter learning algorithm. Moreover, we assume the width of edges in ground truth images to be two pixels wide in this study. Thus, we compute the accuracy with the standard of two-pixel width in the learning process; but thinning to

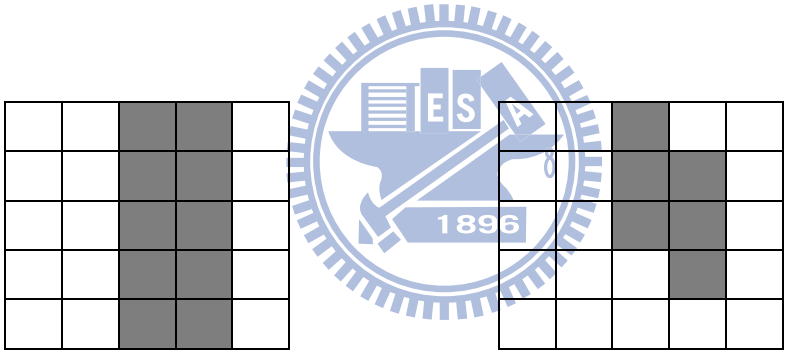
one-pixel width in the final step of thinning algorithm.

When we calculate the accuracy of one-pixel width in the final step, we only consider one side of the edge which is two-pixel width in the ground truth image. For the image map Fig. 5.4(b) obtained from ground truth image Fig. 5.4(a), the average accuracy $\bar{a}_{cc}^{r=1}$ can be computed by

$TP = 4; \quad FP = 2; \quad TN = 18; \quad FN = 1;$

$$\text{Edge accuracy} = \frac{4}{4+1} = 0.8, \quad \text{Non-edge accuracy} = \frac{18}{18+2} = 0.9,$$

$$\text{Average Accuracy } \bar{a}_{cc}^{r=1} = \frac{0.8+1\times 0.9}{1+1} \times 100\% = 85\%$$



(a) (b)

Fig. 5.4 (a) The ground truth image, (b) The edge map obtained.

5.2 The Result of Parameter Learning Algorithm

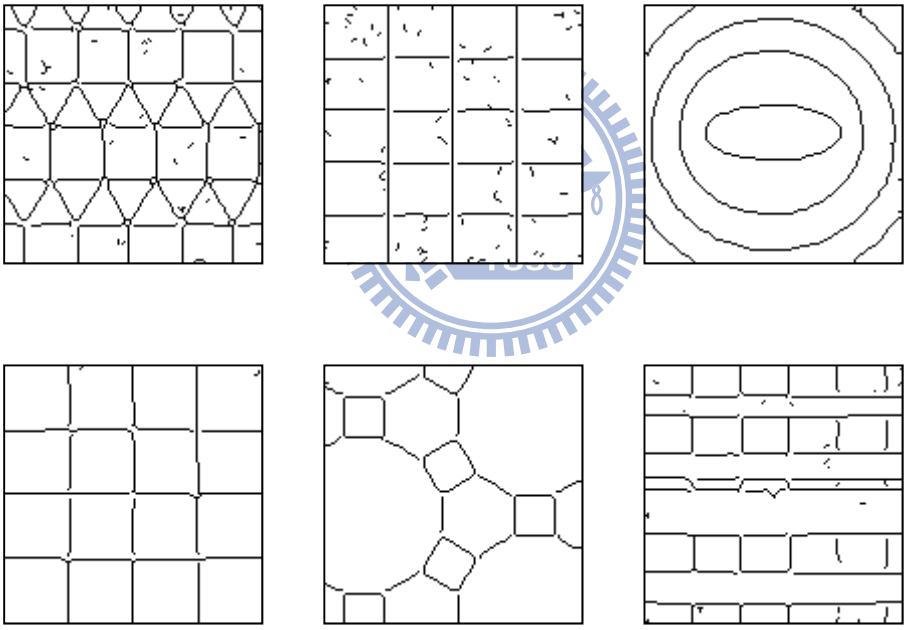
To verify the effectiveness of our proposed weighted mean based interval-valued fuzzy relations for edge detection, we define three sets of different initial parameters to demonstrate the stability and correctness of our method. By computing the average accuracy with different strength of counter-balance parameter r, we can control the sensitivity of parameter set to edge map images obtained from the parameter learning

process.

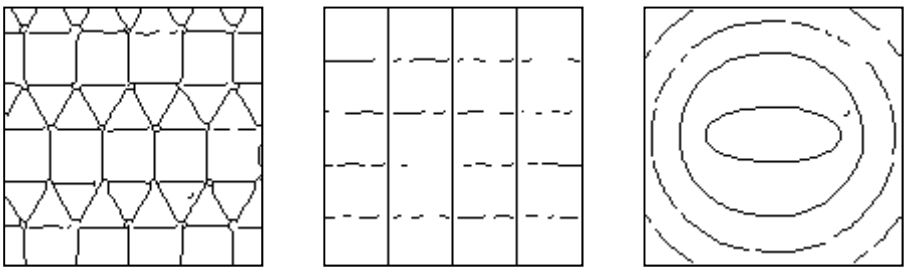
5.2.1 The Result of Linear Weighted Mean Edge Detection

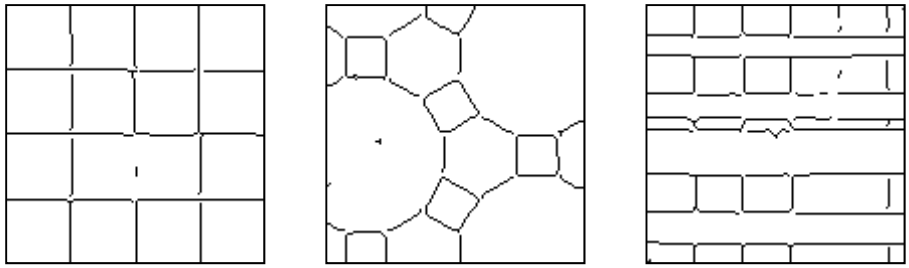
Figs. 5.5−5.7 shows the result images with different initial parameters of linear weighted mean edge detection through 40 learning epoches. Table 5.1 indicates the exact values of accuracy and parameters. The parameters used in post-processing are $R_{NMS} = 0.8$, window size =5 and *maxNum* =5. The three sets of initial parameters are defined as follows:

A. $\alpha_t = 0.25,\ \alpha_s = 0.75,\ \eta_t = \eta_s = 0.0035,\ \eta_T = 0.05,$ and $T = 20$

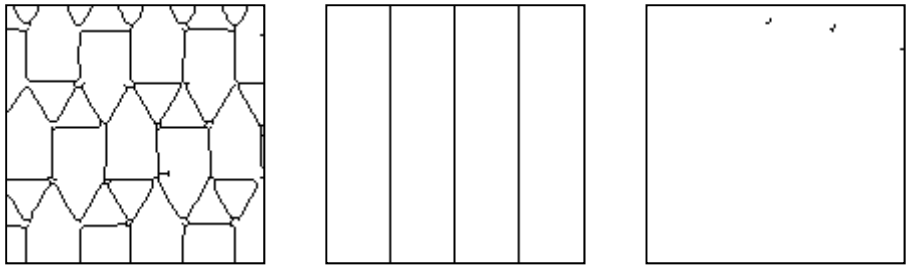


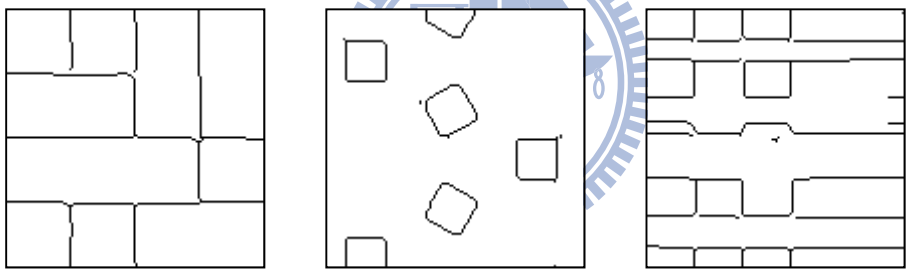
(a)





(b)

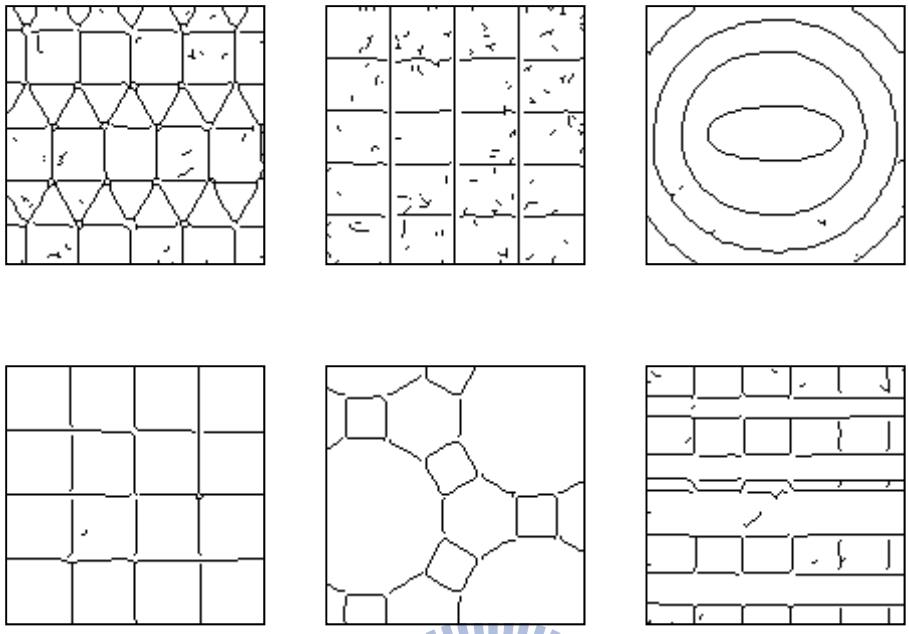




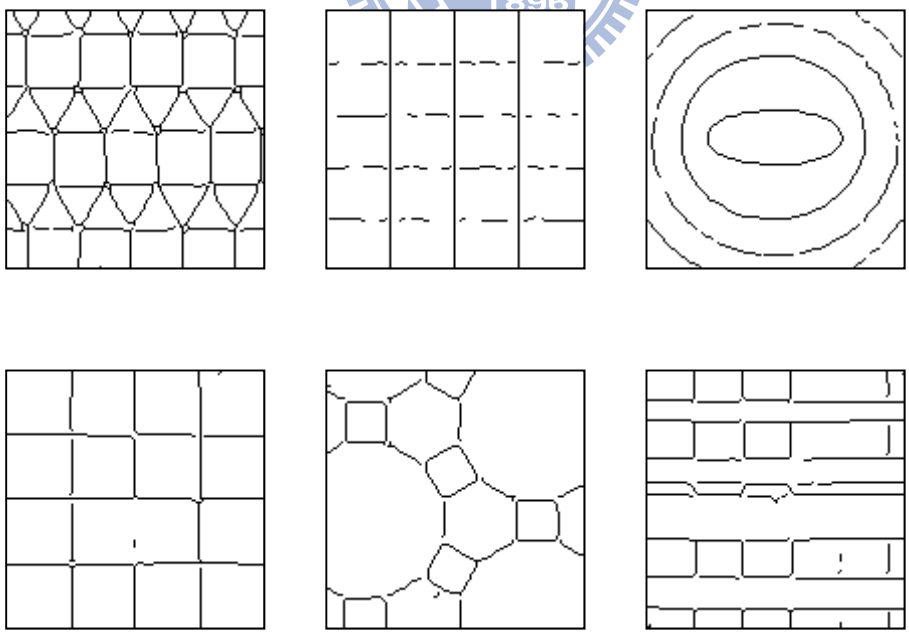
(c)

Fig. 5.5 The result images which use different r in the learning algorithm of linear weighted mean edge detection with initial values $\alpha_t = 0.25$, $\alpha_s = 0.75$, $T = 20$. (a) Images obtained by computing accuracy based on $r = 1$ with learned parameters $\alpha_t = 0.2805$, $\alpha_s = 0.7203$, T=0.662, (b) Images obtained by computing accuracy based on $r = 2$ with learned parameters $\alpha_t = 0.2813$, $\alpha_s = 0.7244$, T=1.2541, (c) Images obtained by computing accuracy based on $r = 4$ with learned parameters $\alpha_t = 0.3741$, $\alpha_s = 0.6299$, T=2.2146.

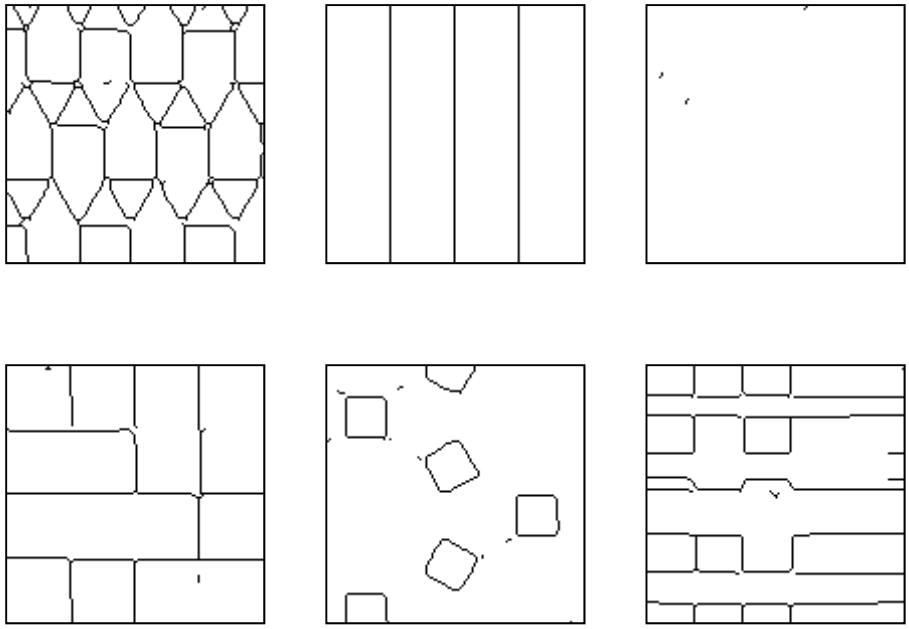
B. $\alpha_t = 0.2,\ \alpha_s = 0.7,\ \eta_t = \eta_s = 0.0035,\ \eta_T = 0.05,$ and $T = 20$



(a)



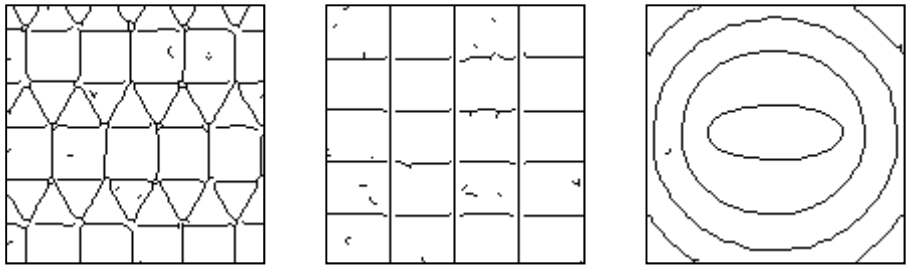
(b)

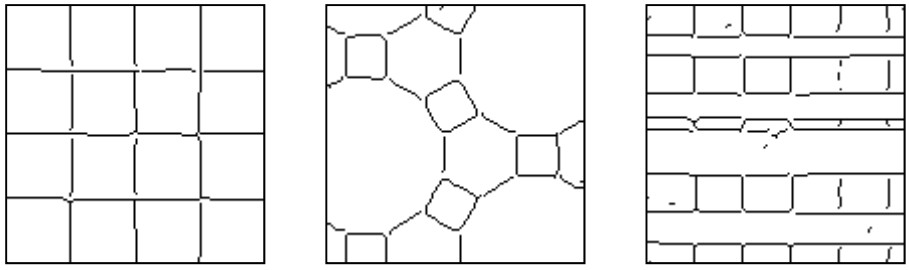


(c)

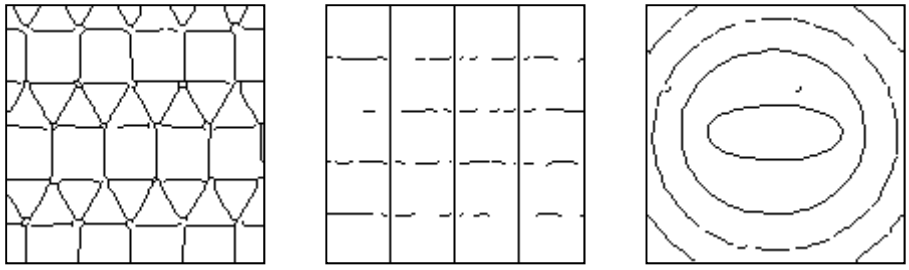
Fig. 5.6 The result images which use different r in the learning algorithm of linear weighted mean edge detection with initial values $\alpha_t = 0.2$, $\alpha_s = 0.7$, $T = 20$. (a) Images obtained by computing accuracy based on $r = 1$ with learned parameters $\alpha_t = 0.2392$, $\alpha_s = 0.7604$, T=0.7203, (b) Images obtained by computing accuracy based on $r = 2$ with learned parameters $\alpha_t = 0.3855$, $\alpha_s = 0.6166$, T=0.6590, (c) Images obtained by computing accuracy based on $r = 4$ with learned parameters $\alpha_t = 0.3641$, $\alpha_s = 0.6462$, T=2.3773.

C. $\alpha_t = 0.3$, $\alpha_s = 0.65$, $\eta_t = \eta_s = 0.0035$, $\eta_T = 0.05$, and $T = 50$

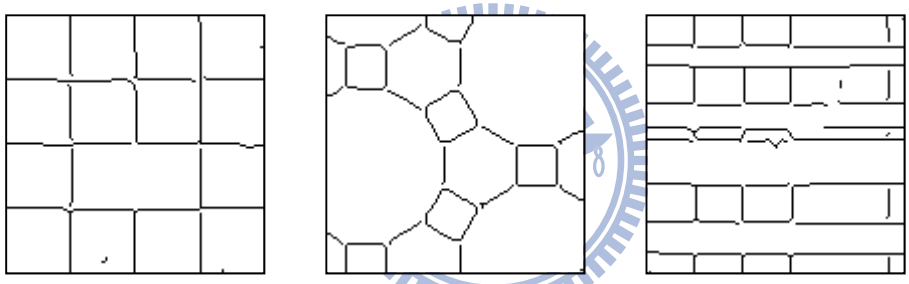




(a)



(b)



(c)

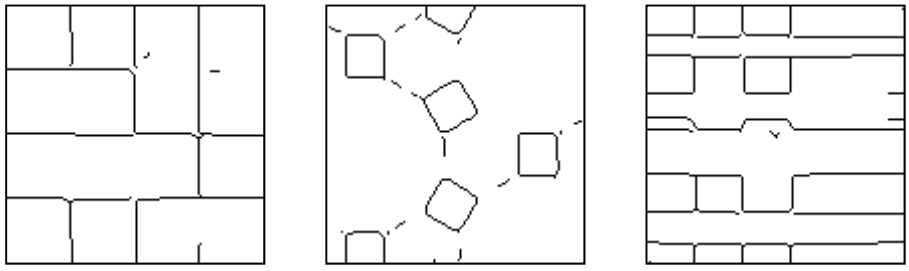


Fig. 5.7 The result images which use different r in the learning algorithm of linear weighted mean edge detection with initial values $\alpha_t = 0.3$, $\alpha_s = 0.65$, $T = 50$. (a) Images obtained by computing accuracy based on $r = 1$ with learned parameters $\alpha_t = 0.3204$, $\alpha_s = 0.6798$, T=0.6025, (b) Images obtained by computing accuracy based on $r = 2$ with learned parameters $\alpha_t = 0.3492$, $\alpha_s = 0.6519$, T=0.8369, (c) Images obtained by computing accuracy based on $r = 4$ with learned parameters $\alpha_t = 0.3637$, $\alpha_s = 0.6346$, T=2.1899.

TABLE I

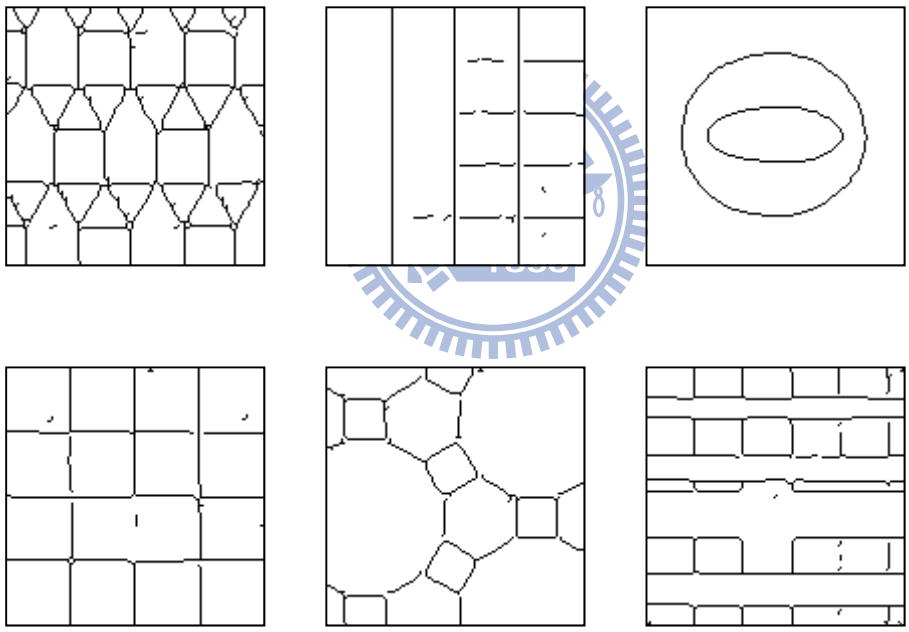
THE BEST OPERATING PARAMETERS AND AVERAGE ACCURACY OF LINEAR WEIGHTED MEAN EDGE DETECTION

Initial Values	Parameter r Used in the Learning Algorithm	The Best Learned Parameters	Accuracy $\bar{a}_{cc}^{r=1}$ of Edge map Images Obtain
$\alpha_t = 0.25$ $\alpha_s = 0.75$ $T = 20$	$r = 1$	$\alpha_t = 0.2805$, $\alpha_s = 0.7203$, $T = 0.6620$	90.44%
	$r = 2$	$\alpha_t = 0.2813$, $\alpha_s = 0.7244$, $T = 1.2541$	88.53%
	$r = 4$	$\alpha_t = 0.3741$, $\alpha_s = 0.6299$, $T = 2.2146$	72.39%
$\alpha_t = 0.2$ $\alpha_s = 0.7$ $T = 20$	$r = 1$	$\alpha_t = 0.2392$, $\alpha_s = 0.7604$, $T = 0.7203$	90.26%
	$r = 2$	$\alpha_t = 0.3855$, $\alpha_s = 0.6166$, $T = 0.6590$	87.91%
	$r = 4$	$\alpha_t = 0.3641$, $\alpha_s = 0.6462$, $T = 2.3773$	72.31%
$\alpha_t = 0.3$ $\alpha_s = 0.65$ $T = 50$	$r = 1$	$\alpha_t = 0.3204$, $\alpha_s = 0.6798$, $T = 0.6025$	90.12%
	$r = 2$	$\alpha_t = 0.3492$, $\alpha_s = 0.6519$, $T = 0.8369$	88.50%
	$r = 4$	$\alpha_t = 0.3637$, $\alpha_s = 0.6346$, $T = 2.1899$	73.18%

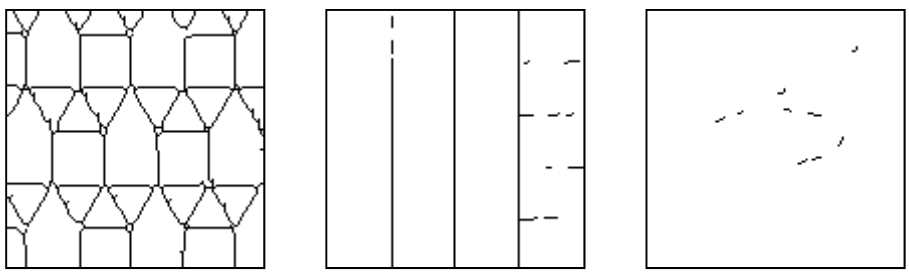
5.2.2 The Result of Quadratic Weighted Mean Edge Detection

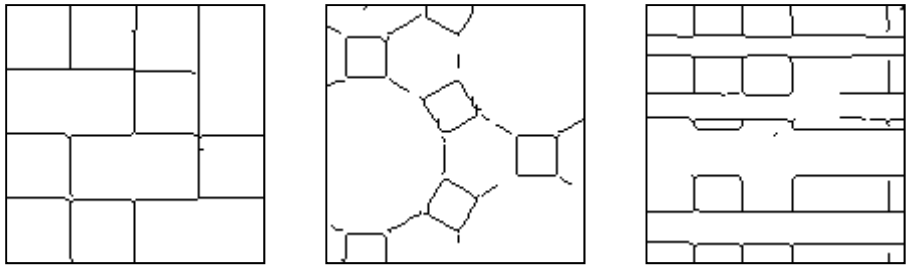
Fig. 5.8 – 5.10 shows the result images with different initial parameters of quadratic weighted mean edge detection through 40 learning epoches. Table 5.2 indicates the exact values of accuracy and parameters. The parameters used in post-processing are $R_{NMS} = 0.8$, window size=5 and *maxNum*=5. The three sets of initial parameters are defined as follows:

A. $\alpha_t = 0.25,\ \alpha_s = 0.75,\ \eta_t = \eta_s = 0.0035,\ \eta_T = 0.035,$ and $T = 20$

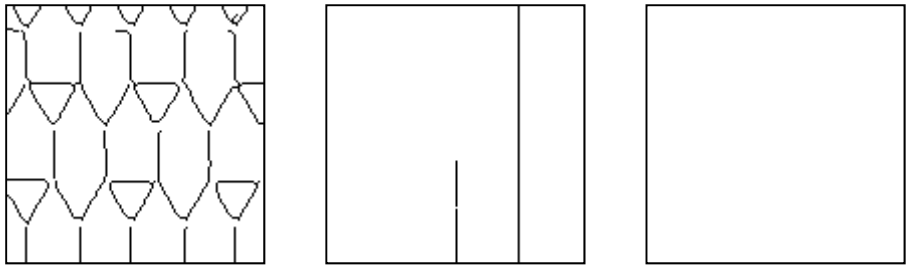


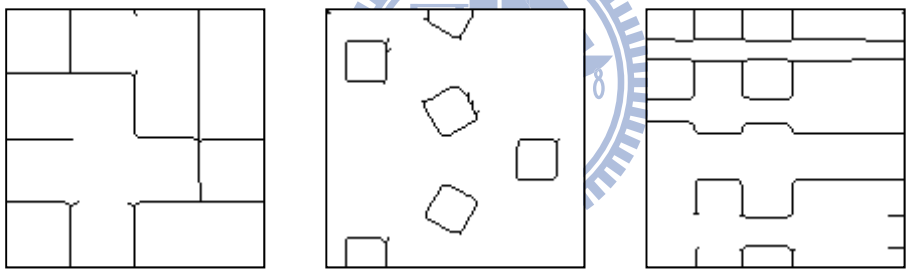
(a)





(b)

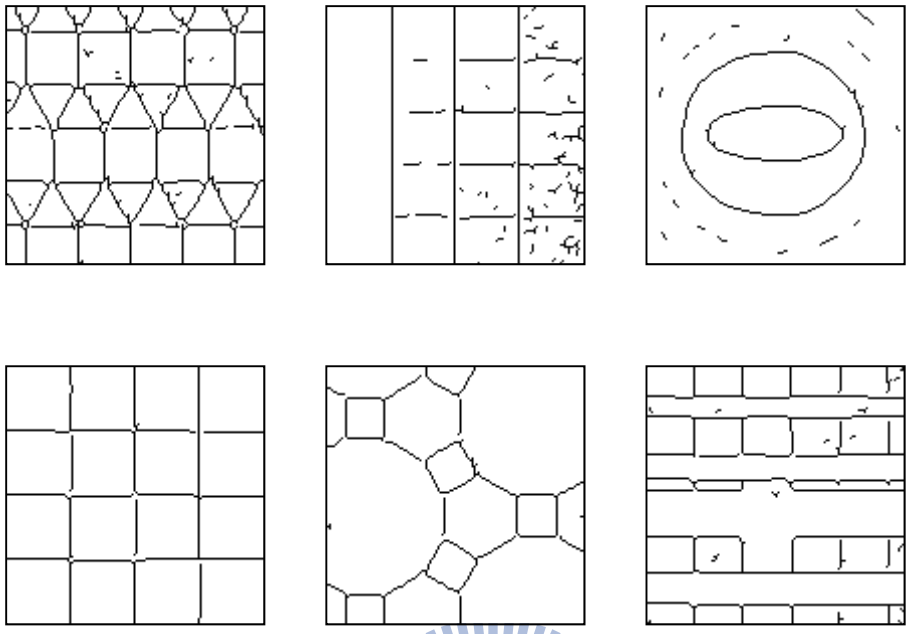




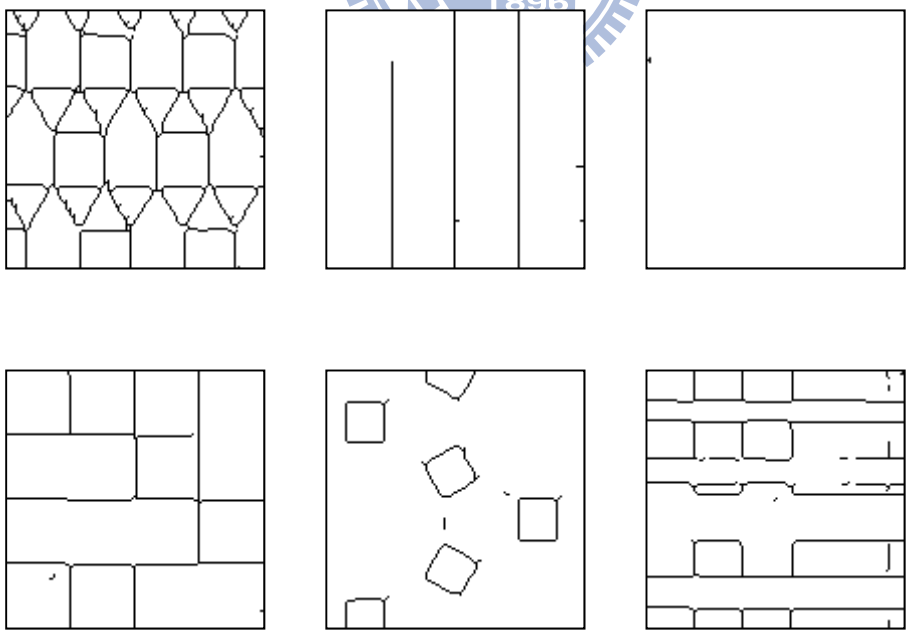
(c)

Fig. 5.8 The result images which use different r in the learning algorithm of quadratic weighted mean edge detection with initial values $\alpha_t = 0.25$, $\alpha_s = 0.75$, $T = 20$, (a) Images obtained by computing accuracy based on $r = 1$ with learned parameters $\alpha_t = 0.0298$, $\alpha_s = 0.9691$, T=22.7966, (b) Images obtained by computing accuracy based on $r = 2$ with learned parameters $\alpha_t = 0.2433$, $\alpha_s = 0.7541$, T=26.7424, (c) Images obtained by computing accuracy based on $r = 4$ with learned parameters $\alpha_t = 0.4132$, $\alpha_s = 0.6014$, T=28.1916.

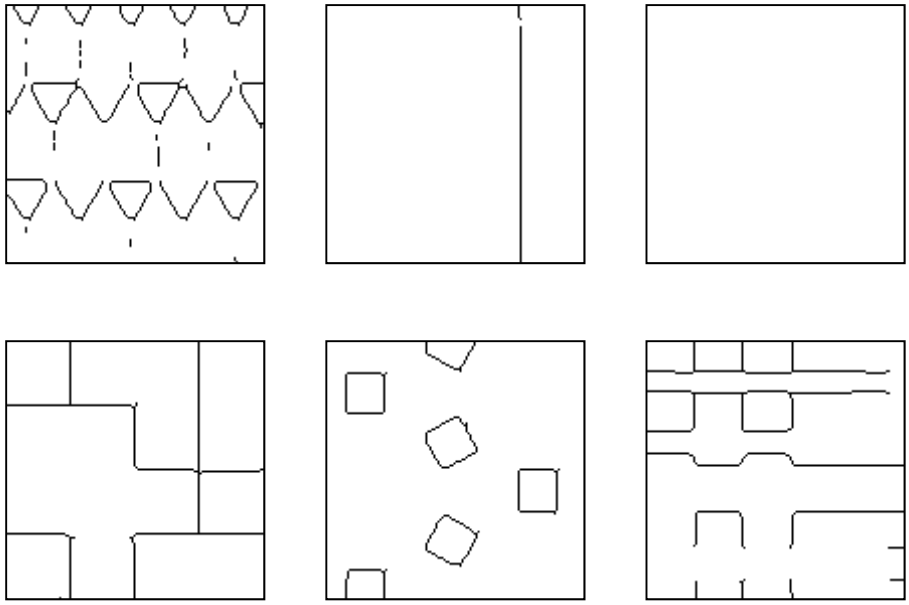
B. $\alpha_t = 0.2,\ \alpha_s = 0.7,\ \eta_t = \eta_s = 0.0035,\ \eta_T = 0.035,$ and $T = 20$



(a)



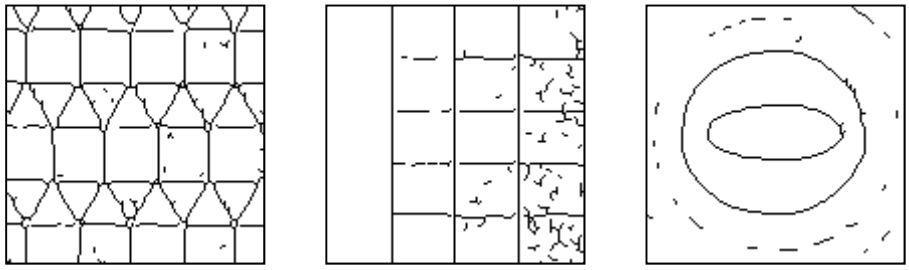
(b)

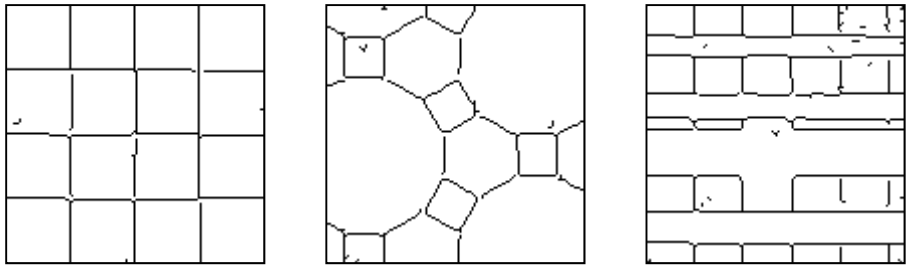


(c)

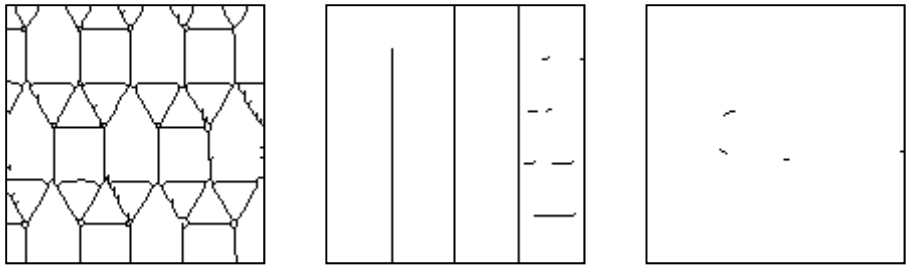
Fig. 5.9 The result images which use different r in the learning algorithm of quadratic weighted mean edge detection with initial values $\alpha_t = 0.2$, $\alpha_s = 0.7$, $T = 20$. (a) Images obtained by computing accuracy based on $r = 1$ with learned parameters $\alpha_t = 0$, $\alpha_s = 1$, T=22.6019, (b) Images obtained by computing accuracy based on $r = 2$ with learned parameters $\alpha_t = 0.2977$, $\alpha_s = 0.7066$, T=25.6479, (c) Images obtained by computing accuracy based on $r = 4$ with learned parameters $\alpha_t = 0.4260$, $\alpha_s = 0.5829$, T=28.1265.

C. $\alpha_t = 0.3$, $\alpha_s = 0.65$, $\eta_t = \eta_s = 0.0035$, $\eta_T = 0.035$, and $T = 50$

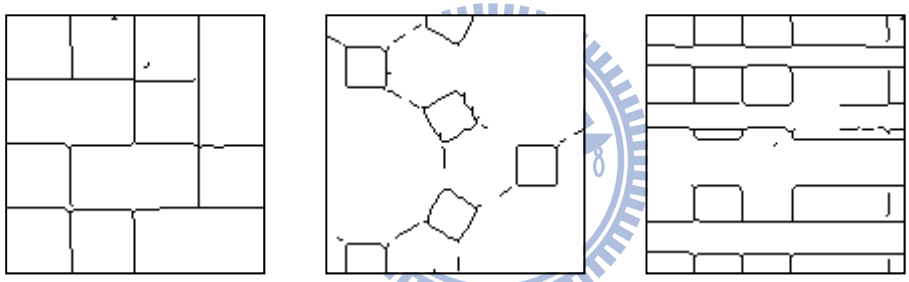




(a)



(b)



(c)

Fig. 5.10 The result images which use different r in the learning algorithm of quadratic weighted mean edge detection with initial values $\alpha_t = 0.3$, $\alpha_s = 0.65$, $T = 50$. (a) Images obtained by computing accuracy based on $r = 1$ with learned parameters $\alpha_t = 0$, $\alpha_s = 1$, T=22.3590, (b) Images obtained by computing accuracy based on $r = 2$ with learned parameters $\alpha_t = 0$, $\alpha_s = 1$, T=38.2036, (c) Images obtained by computing accuracy based on $r = 4$ with learned parameters $\alpha_t = 0.3045$, $\alpha_s = 0.6870$, T=39.2558.

TABLE II

THE BEST OPERATING PARAMETERS AND AVERAGE ACCURACY OF QUADRATIC WEIGHTED MEAN EDGE DETECTION

Initial Values	Parameter r Used in the Learning Algorithm	The Best Learned Parameters	Accuracy $\bar{a}_{cc}^{r=1}$ of Edge map Images Obtain
$\alpha_t = 0.25$ $\alpha_s = 0.75$ $T = 20$	$r = 1$	$\alpha_t = 0.0298$, $\alpha_s = 0.9691$, $T = 22.7966$	84.77%
	$r = 2$	$\alpha_t = 0.2433$, $\alpha_s = 0.7541$, $T = 26.7424$	76.30%
	$r = 4$	$\alpha_t = 0.4132$, $\alpha_s = 0.6014$, $T = 28.1916$	67.30%
$\alpha_t = 0.2$ $\alpha_s = 0.7$ $T = 20$	$r = 1$	$\alpha_t = 0$, $\alpha_s = 1$, $T = 22.6019$	85.42%
	$r = 2$	$\alpha_t = 0.2977$, $\alpha_s = 0.7066$, $T = 25.6479$	73.31%
	$r = 4$	$\alpha_t = 0.4260$, $\alpha_s = 0.5829$, $T = 28.1265$	64.80%
$\alpha_t = 0.3$ $\alpha_s = 0.65$ $T = 50$	$r = 1$	$\alpha_t = 0$, $\alpha_s = 1$, $T = 22.3590$	85.29%
	$r = 2$	$\alpha_t = 0$, $\alpha_s = 1$, $T = 38.2036$	75.17%
	$r = 4$	$\alpha_t = 0.3045$, $\alpha_s = 0.6870$, $T = 39.2558$	68.26%

From the experiment results, we have observed that our algorithm is stable due to the similarity in the average accuracies obtained with different initial parameters. Besides, if we choose larger r used in learning procedure to compute the average accuracy, the sensitivity of edge map images obtained will be decreased, i.e., the edge map sensitivity will be reduced because of larger r.

The quadratic mean weighting form can strengthen the weighted mean difference between the center pixel and its neighbors, thus it is sensitive to minor difference between pixels. Namely, not only edge pixels but also noise pixels will be prone to be detected. We also test Canny method to the same filtered noisy images of Fig. 5.2, and the edge images obtained and their average accuracy are shown in Fig. 5.11 and Table III, respectively. From Table III, Canny can only achieve the average accuracy of 76.91%, which is worse than our linear and quadratic method.

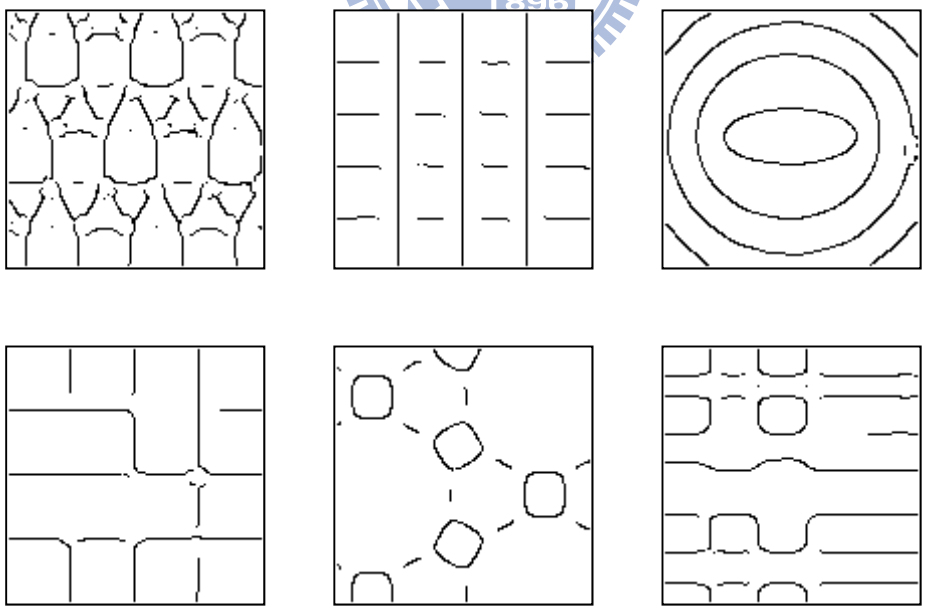


Fig. 5.11 The results of Canny edge detector for six filtered synthetic noisy images of Fig. 5.2. ($T_{low} = 0.1,\ T_{high} = 0.12$)

TABLE III

THE RESULTS OF BEST OPERATING PARAMETERS TRIALS AND AVERAGE ACCURACIES

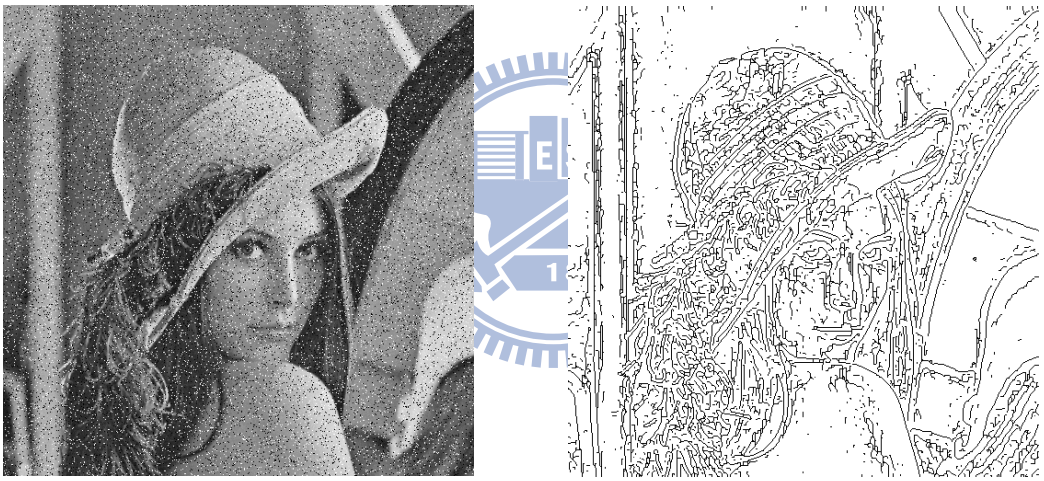
Methods	The Best Learned Parameters	Average Accuracy
The Linear Weighted Mean Edge Detection learned with $r=1$	$\alpha_t = 0.2805,\ \alpha_s = 0.7203,$ $T = 0.6620$	$\bar{a}_{cc}^{r=1} = 90.44\%$
		$\bar{a}_{cc}^{r=4} = 95.89\%$
The Linear Weighted Mean Edge Detection learned with $r=4$	$\alpha_t = 0.3637,\ \alpha_s = 0.6346,$ $T = 2.19$	$\bar{a}_{cc}^{r=1} = 73.18\%$
		$\bar{a}_{cc}^{r=4} = 89.19\%$
The Quadratic Weighted Mean Edge Detection learned with $r=1$	$\alpha_t = 0,\ \alpha_s = 1,$ $T = 22.6019$	$\bar{a}_{cc}^{r=1} = 85.42\%$
		$\bar{a}_{cc}^{r=4} = 93.79\%$
The Quadratic Weighted Mean Edge Detection learned with $r=4$	$\alpha_t = 0.3045,\ \alpha_s = 0.6871,$ $T = 39.2358$	$\bar{a}_{cc}^{r=1} = 68.26\%$
		$\bar{a}_{cc}^{r=4} = 87.24\%$
Canny Method	$T_{low} = 0.1,\ T_{high} = 0.12$	$\bar{a}_{cc}^{r=1} = 78.66\%$
		$\bar{a}_{cc}^{r=4} = 90.31\%$

5.3 Simulation Results of Natural Images

For natural image, it is impossible to determine a ground truth edge map, so we can not compute the average accuracy. Therefore, we try our best parameters shown in Table III to test the natural image "Lena" with 10% impulse noise and Gaussian noise ($\mu = 0,\ \sigma = 4$). We obtain the edge image shown in Figs. 5.12(b)−(f), while the

Canny method produces the edge image shown in Fig. 5.12(g). From these images, we found that our method show the details more clearly in the edge maps due to the large average ratio of non-edge points to edge points in the ground truth images.

In Fig. 5.12(b) and (d), even though more edges can be detected, the parameter sets obtained by computing accuracy with $r = 1$ seem too sensitive to obtain suitable edge images for the natural image "Lena." Then, we recommend the parameter sets obtained by computing accuracy with $r = 4$ to get proper edge maps for the "Lena" image.



(a) (b)

(c) (d)

(e)

(f)

Fig. 5.12 (a) The image "Lena" with 10% impulse noise and Gaussian noise ($\mu = 0,\ \sigma = 4$), (b) Edge image obtained by linear weighted mean aggregation with parameter set ($\alpha_t = 0.2805,\ \alpha_s = 0.7203,\ T = 0.6620$ and $r = 1$). (c) Edge image obtained by linear weighted mean aggregation with parameter set $\alpha_t = 0.3637,\ \alpha_s = 0.6346,\ T = 2.19$ and $r = 4$. (d) Edge images obtained by quadratic weighted mean aggregation with parameter set $\alpha_t = 0, \alpha_s = 1,\ T = 22.6019$ and $r = 1$. (e) Edge images obtained by quadratic weighted mean aggregation with parameter set $\alpha_t = 0.3045,\ \alpha_s = 0.6871,\ T = 39.2358$ and $r = 4$. (f) The edge map of Canny method ($T_{low} = 0.1,\ T_{high} = 0.12$).

Chapter 6 Conclusion

In this thesis, we apply the linear/quadratic weighted mean aggregation and interval-valued fuzzy relations to detect edges of images. In each 3×3 sliding window, we use the upper and lower constructors to calculate the weighted mean aggregations of the central pixel and its eight neighbor pixels to construct the interval-valued fuzzy relation and its associated W-fuzzy relation. The W-fuzzy relation, constructed by computing the difference between the upper and lower aggregations, indicates the degree of intensity variation between the center pixel and its neighborhood. And thus it represents an edge if it is larger than a threshold. Moreover, we derive the learning formulas of the weighting parameters of the mean to reduce the edge detection error and utilize pocket algorithm to obtain the final optimal parameter set for training images. Also, the parameter r of accuracy calculation is introduced to control the sensitivity of parameter set to edge image and the post-processing techniques is applied to enhance the continuity of edge. Finally, our design model is applied to edge detection of the synthetic and natural images. In comparison with the edge images obtained by Canny edge detector, our edge map can better highlights the object details and shows the contours strongly.

In the future, we intend to find out the best ratio of non-edge points to edge points, which strongly affects the selection of parameter set in the learning procedure, to obtain the optimal parameter set generating suitable edge map for most natural images.

Reference

[1] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol 8, no. 6, pp679－698, 1986.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Ed., Prentice Hall, New Jersey, 2002.

[3] R. Medina-Carnicer, F.J. Madrid-Cuevas, “Unimodal thresholding for edge detection”, *Pattern Recognition*, vol. 41, pp. 2337－2346, 2008.

[4] H. D. Cheng, Y.H. Chen, X.H Jiang, “Thresholding using two-dimensional histogram and fuzzy entropy principle,” *IEEE Transactions on Image Processing*, vol. 9, no. 4, 2000, 732－735.

[5] L. A. Zadeh, *Fuzzy sets, Inform. and Control 8* (1965) 338－353.

[6] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning－ I,” *Inform. Sci.* 8 (1975) 199－49.

[7] H. R. Tizhoosh, “Fuzzy Image Processing,” *Springer*, Heidelberg, Germany,1998

[8] N. R. Pal, J.C. Bezdek, “Measures of fuzziness: a review and several new classes,” in R.R. Yager, L.A. Zadeh(Eds.), *Fuzzy Sets, Neural Networks and Soft Computing*, Van Nostrand Reihold, New York, 1994, pp. 194－212.

[9] L. K. Huang, M.J. Wang, “Image thresholding by minimizing the measure of fuzziness,” *Pattern Recognition*, 28 (1995), 41-51.

[10] H. R. Tizhoosh, “On thresholding and potentials of fuzzy techniques,” *R. Kruse, J. Dassow*(Eds.), *Informatik’98*, Heidelberg, Springer, Berlin, 1998, pp. 97－16.

[11] S. K. Pal, A. Ghosh, “Index of area coverage of fuzzy image subsets and object extraction,” *Pattern Recognition Lett.* 7 (1988) 77－86.

[12] J. M. Mendel, R.I. John, “Type-2 fuzzy sets made simple,” *IEEE Trans. Fuzzy Systems* 10 (2002) 117－127.

[13] E. Barrenechea, H. Bustince, and C. Lopez-Molina, “Construction of interval-value fuzzy relations with application to the generation of fuzzy edge images,” *IEEE Trans. Fuzzy System.*, vol 19, no. 5, pp. 819−830, October 2011.

[14] H. Bustince, E. Barrenechea, M. Pagola, and J. Fernandez, “Interval-valued fuzzy sets constructed from matrices: Application to edge detection.” *Fuzzy Sets Syst.*, vol. 160, pp. 1819−1840, 2009.

[15] J. Barzilai and J.M. Borwein, “Two point step size gradient methods,” *IMA J. Numer. Anal.*, vol. 8, pp. 141−148, 1988.

[16] E. P. Klement, R. Mesiar, and E. Pap, “Triangular Norms. Dordrecht,” *The Netherlands: Kluwer*, 2000.

[17] J. H. Hong, S. Campbell, and P. Yeh, “Optical pattern classifier with Perceptron learning,” *Applied Optics*, vol. 29, no. 20, pp. 3019−3025, 1990.

[18] M. E. Yuksel and E. Besdok, “A simple neuro-fuzzy impulse detector for efficient blur reduction of impulse noise removal operators for digital images,” *IEEE Trans. Fuzzy Syst.*, vol.12, no. 6, pp. 854−865, Dec. 2004.

[19] Rosenblatt, Frank (1957), “The Perceptron--a perceiving and recognizing automaton.” Report 85-460-1, *Cornell Aeronautical Laboratory*.

[20] Liou, D.-R.; Liou, J.-W.; Liou, C.-Y. (2013). “Learning Behaviors of Perceptron”. ISBN 978-1-477554-73-9. *iConcept Press*.

[22] Lei Huang, Genxun Wan, Changping Liu, “An Improved Parallel Thinning Algorithm,” *IEEE, Document Analysis and Recognition*, pp. 780−783, 2003.