


國立交通大學

電控工程研究所

博士論文

利用尺度空間二值化與累積梯度投影的方法
應用於車牌字體的擷取與辨識



Extraction and Recognition of License Plate
Characters Using Scale-Space Binarization and
Accumulated Gradient Projection Methods

研究生：葉天德

指導教授：陳永平 教授

中華民國一百年貳月

利用尺度空間二值化與累積梯度投影的方法
應用於車牌字體的擷取與辨識

Extraction and Recognition of License Plate
Characters Using Scale-Space Binarization and
Accumulated Gradient Projection Methods

研究生：葉天德

Student: Tien-Der Yeh

指導教授：陳永平

Advisor: Yon-Ping Chen



A Dissertation

Submitted to Institute of Electrical and Control Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical Engineering

Feb. 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年貳月

利用尺度空間二值化與累積梯度投影的方法 應用於車牌字體的擷取與辨識

學生：葉天德

指導教授：陳永平

國立交通大學電控工程研究所博士班

摘 要

本論文提出一個車牌字體辨識系統，此系統包含三個主要方法。第一個方法稱為尺度空間二值化，可以用來從灰階圖像上擷取字體。此方法結合了穩健的高斯差函數和動態二值化處理，從未知影像中直接擷取出車牌字體。為了使擷取的處理速度加快，本論文也提出優化的方法用以縮短計算時間。第二個方法稱為邊界投票方法，適合用來矯正字體在影像拍攝過程中所導致的幾何型變。此方法一開始假設了許多直線，然後以投票方式找出一條通過最多邊界點的直線當成邊界線。找到的邊界線可以幫助矯正字體的幾何型變，因而藉此改善辨識率。第三個方法稱為累積梯度投影方法，利用累積梯度並且轉換它們成特徵向量來識別獨立字體。這些特徵向量稱為累積梯度投影向量，被實驗證實對雜訊及照度改變是具有穩健性的。

A License Plate Recognition System Using Scale-Space Binarization and Accumulated Gradient Projection Methods

Student: **Tien-Der Yeh**

Advisors: **Dr. Yon-Ping Chen**

Institute of Electrical and Control Engineering
National Chiao Tung University

ABSTRACT

A system consisting of three methods to deal with license plate characters recognition is proposed in this dissertation. The first method, scale-space binarization, is suitable for extracting characters from gray-level images. The method combines the robust Difference-of-Gaussian function and dynamic thresholding technique to extract the license plate characters directly. In order to speed up the extraction process, optimization methods are also disclosed to reduce the computation time. The second method, voting boundary method, is suitable for correcting characters from geometric deformation induced during capture process. It assumes many straight lines candidates and detects the best one passing through most of the edge pixels by voting. The boundary lines can be used for correcting the deformation and improve recognition rate thereby. The third one, accumulated gradient projection method, recognizes isolated characters by accumulating the gradient projection of the characters and converts them into feature vector for comparison. The feature vector is called accumulated gradient projection vector and is proven robust regardless of noise and illumination change in experiments.

Acknowledgement

First of all, I am heartily thankful to my advisor, Prof. Yon-Ping Chen, whose encouragement, guidance and support from the initial to the final stage helped me to learn what I need from the research and made this thesis possible. And the thesis committee, Prof. Li-Chen Fu, Prof. Kuo-Kai Shyu, Prof. Jen-Hui Chuang, Prof. Kai-Tai Song, and Prof. Sheng-Fuu Lin, whose valuable recommendations and comments help to perfect this dissertation.

Next, I'd like to give thank to my parents, who always give me confidence and encourage me to persist in the thesis. Also, a deepest gratitude would be shown to my respected uncle, Mr. Ching-Tsai Peng, who taught me always positive thinking and encouraged me all the time especially when a bottleneck is encountered on thesis writing. Besides, I would give my best thanks and appreciation to my best friends, Erik Lin and Patrick Lam, who supported me during this period and gave me encouragement whenever I need help. Furthermore, this thesis would not be accomplished without my wife's fully support. Her understanding, trust and assistance helped me to persist in this thesis until accomplishment.

Finally, I offer my best regards and blessings to all of those who supported and helped me in any respect during the completion of this dissertation. Because of you, I can focus on the research until completion. Thank you very much.

誌 謝

首先，我要由衷的感謝我的指導教授 陳永平 老師。他多年來從頭到尾的鼓勵、引導、及支持幫助我從研究中學得我所要的及讓這篇論文得以完成。同時要謝謝口試委員 傅立成 老師、徐國鎧 老師、莊仁輝 老師、宋開泰 老師與 林昇甫 老師 對此篇論文的寶貴意見與指正，使本論文更臻完善。

其次，我要謝謝我的父母，他們總是給我信心及鼓勵，讓我可以堅持到最後並完成這篇論文。還有，我也要向我尊敬的伯父 彭清財 先生表達最深的感激，他教我要維持積極的思考，尤其是在寫作論文遇到瓶頸的時候不斷的給我正向的鼓勵。再來我要感激我兩個最好的朋友，Erik Lin 及 Patrick Lam，他們在我寫作論文的這段期間需要幫忙的時候不斷的給予支持及鼓勵。此外，如果沒有內人的全力支持，這篇論文也無法完成。她的體諒、信任及協助讓我可以無後顧之憂一路堅持直到完成這篇論文。

最後，我要對所有在這篇論文寫作期間給予我支持、鼓勵及幫助的人表達最高的敬意與祝福。因為你們的付出，讓我能夠專心於課業直到畢業這一刻，謝謝你們。

Table of Contents

Chinese Abstract	i
English Abstract	ii
Acknowledgement	iii
Chinese Acknowledgement	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Symbols	ix
Chapter 1	Introduction	1
Chapter 2	Review of Related Works	4
2.1	Scale Space Theory and Difference-of-Gaussian Functions	4
2.2	Previous Methods for Image Binarization	5
2.3	License Plate Recognition	6
Chapter 3	The Extraction Method	8
3.1	Profile Localization	9
3.2	Dynamic Threshold Propagation	16
3.3	Thresholding and Connected Component Analysis	19
3.4	Eliminate False Candidates	20
3.5	Implementation for Fast Binarization	21
3.5.1	Optimization in convolution	22
3.5.2	Implement by integers and shifters	22
3.5.3	Use acceleration table for dynamic threshold propagation	24
Chapter 4	The Deformation Correction Method	26
4.1	Useful Properties for Deformation Correction	26
4.2	Voting Boundary Method	28
4.3	The Correction Method	31
Chapter 5	The Recognition Method	33
5.1	Why AGPV	33
5.2	The AGPV Methods	34
5.2.1	Determine Axes	34
5.2.1.1	Build up Orientation Histogram	34
5.2.1.2	Determine the Nature Axes	35
5.2.1.3	Determine the Augmented Axes	37
5.2.2	Calculate AGPVs	39
5.2.2.1	Projection principles	39
5.2.2.2	Gradient projection accumulation	41
5.2.2.3	Normalization	42
5.2.3	Matching and Recognition	44
5.2.3.1	Create standard AGPV database	45
5.2.3.2	Properties used for matching	45
5.2.3.3	Matching of characters	49
Chapter 6	Experimental Results	52
6.1	Scale-Space Binarization Method(Extraction) Test	52
6.1.1	Feasibility Test	52
6.1.2	Reliability Test	53
6.1.2.1	Quantization noise analysis	53
6.1.2.2	Illumination analysis	54
6.1.3	Computation time test	55

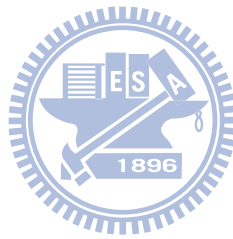
6.2	AGPV(Recognition) Test	59
Chapter 7	Conclusion and Future Work	60
7.1	Conclusion	60
7.2	Future work.....	61
	Bibliographies	62



List of Figures

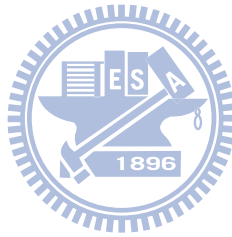
Fig. 1-1	Functional block diagram of the proposed LPR system	3
Fig. 3-1	Functional block diagram of the SSB method	9
Fig. 3-2	An example of the SSB method	10
Fig. 3-3	The procedure to produce DOG Images on different observation scales	12
Fig. 3-4	An ideal unit-step edge (upper graph) and its DOG response (lower graph)	13
Fig. 3-5	Determine boundary pixels	15
Fig. 3-6	Extraction of a perfect sample character image	15
Fig. 3-7	Dynamic threshold propagation	17
Fig. 3-8	An example of dynamic threshold propagation	19
Fig. 3-9	Extraction of an imperfect sample character image	20
Fig. 3-10	Comparison to three Gaussian functions,	23
Fig. 4-1	Typical geometric transformations in LPR systems	27
Fig. 4-2	A character candidate and the bottom pixels	28
Fig. 4-3	Derivation of the voting boundary method	29
Fig. 4-4	Flow chart to vote boundary lines	30
Fig. 4-5	Compensation of geometric deformation	32
Fig. 5-1	The nature axes	37
Fig. 5-2	The nature axes and the augmented axes	38
Fig. 5-3	Gradient projection of COG point and any other pixels	39
Fig. 5-4	Accumulation of gradient projection	41
Fig. 5-5	Normalized AGPV	46
Fig. 5-6	An example comprising different blur index with Fig. 5-5	49
Fig. 6-1	Source image#1 binarization result	56

Fig. 6-2	Source image#2 binarization result	57
Fig. 6-3	Images used in noise analysis	58
Fig. 6-4	Images with different decay curve light sources used in illumination analysis	58



List of Tables

TABLE I	TPR_E BY QUANTIZATION NOISE ANALYSIS	54
TABLE II	TPR_E BY ILLUMINATION ANALYSIS	54
TABLE III	COMPUTATION TIME COMPARISON OF THE THREE METHODS	55
TABLE IV	TPR_R BY QUANTIZATION NOISE ANALYSIS	59
TABLE V	TPR_R BY ILLUMINATION ANALYSIS	59



Symbols

σ	: Gaussian function standard deviation
σ_1	: The first observation scale for Difference-of-Gaussian function
σ_2	: The second observation scale for Difference-of-Gaussian function
$G(\sigma)$: Gaussian function with standard deviation σ
$g_1(x,y)$: The Gaussian function with deviation σ_1
$g_2(x,y)$: The Gaussian function with deviation σ_2
$I(x,y)$: Gray-level source image
$I_1(x,y)$: The first Gaussian blurred image for Difference-of-Gaussian function
$I_2(x,y)$: The second Gaussian blurred image for Difference-of-Gaussian function
$D_1(x,y)$: The first Difference-of-Gaussian image
$DOG(x,y)$: A 2D Difference-of-Gaussian function
$u(x_0)$: Unit step function in parallel to the y-axis with origin shifted to $x=x_0$
Set_1	: The first profile set
Set_2	: The second profile set
Set_3	: Non-profile set
th_f	: Fixed threshold for identifying profile pixels
$effa(x_b,y_b)$: Effective area of (x_b,y_b)
R_{eff}	: Radius of effective area of an edge
Bin_1	: Total number of pixels for the first profile set
Bin_2	: Total number of pixels for the second profile set
(x_b,y_b)	: A boundary pixel
(x_n,y_n)	: A neighboring pixel
$th_d(x,y)$: Dynamic threshold on pixel (x,y)

$ \nabla I_1(x,y) $: Gradient magnitude on pixel (x,y) in the first Gaussian blurred image
(x_b^p, y_b^p)	: The pixel whose dynamic thresholds are assigned in the p -th iteration
Set_b^p	: the set of boundary pixels with dynamic thresholds assigned in the p -th iteration
Set_a^p	: the set of adjacent pixels with dynamic thresholds assigned in the p -th iteration
$(x_a^1(i), y_a^1(i))$: the i -th pixel in Set_a^1
C_E	: Total number of edge pixels of an isolated group
C_B	: Total number of non-edge(body) pixels of an isolated group
C_T	: Total number of pixels of an isolated group
C_P	: The number of edge pixels of an isolated group adjacent to profile sets
W	: Width of a grouped image
H	: Height of a grouped image
U	: Occupancy of a grouped image
S_p	: Profile score of a grouped image
FIT	: Set of boundary pixels falling inside a boundary line
UNDERFIT	: Set of boundary pixels falling below a boundary line
OVERFIT	: Set of boundary pixels falling above a boundary line
$\gamma(x,y)$: Gray-level intensity value of sample pixel (x,y)
$m(x,y)$: Gradient magnitude of sample pixel (x,y)
$\theta(x,y)$: Gradient orientation of sample pixel (x,y)
BIN_{his}	: Total number of bins of an orientation histogram
RES_{his}	: Resolution of an orientation histogram
GE_{his}	: Total gradient energy of an orientation histogram
$H(a)$: The magnitude appeared on angle a of an orientation histogram
p_k	: The k -th peak in an orientation histogram

s_k	: The start angle of the k -th peak in an orientation histogram
e_k	: The end angle of the k -th peak in an orientation histogram
a_{th}	: Threshold for the maximum range of a peak in an orientation histogram
$E(k)$: Energy function for the k -th peak in an orientation histogram
$D(k)$: Outstanding energy function for the k -th peak in an orientation histogram
$\hat{m}(x, y)$: Projected gradient magnitude
$\hat{\ell}(x, y)$: Projected length
$R(x)$: The first gradient projection array
$T(x)$: The second gradient projection array
$\tilde{R}(x)$: The first gradient projection array after smoothing
$\hat{R}(x)$: The smoothed gradient projection array after binarization
$AGPV(i)$: The i -th accumulated gradient projection vector
$EC(V)$: the edge count of an accumulated gradient projection vector V
$C(U, V)$: matching cost function of accumulated gradient projection vectors U and V
UV	: Union vector of two AGPVs
IV	: Intersection vector of two AGPVs
$AA_T(k)$: The k -th axis angle of the test character
$AA(i, j)$: The angle of the j -th axis of the i -th standard character
$NN(i)$: The number of nature AGPVs for i -th standard character
$NA(i)$: The number of augmented AGPVs for i -th standard character
$NV(i)$: The total number of AGPVs for i -th standard character
$V_S(i, j)$: The j -th standard AGPV of the i -th character
$CMC(i)$: Character matching cost between test character and the i -th standard one

Chapter 1 Introduction

The license plate recognition, or LPR in short, has been a popular research topic for several decades [1]-[3],[19]. An LPR system is able to recognize vehicles automatically so that it is useful for many applications such as portal controlling, traffic monitoring, stolen car detection, and etc. Up to now, an LPR system still faces some problems concerning unknown plate size and orientation, various light condition, unexpected image deformation, and limited computation time[3].

Traditional methods for recognition of license plate characters often include several stages. Stage one is detection of possible areas where the license plate may exist. It is a big challenge to detect the plates quickly and robustly since images may contain far more information than just only expected plates. Stage two is segmentation, which divides the detected areas into several regions containing single character candidate. Stage three is normalization; some attributes of the character candidates, e.g., size or orientation, are transformed to certain values for the requirements of recognition stage. Stage four is recognition; the feature vectors extracted from the normalized character candidates can be recognized by technologies such as template matching[16], vector quantization[4], support vector machine(SVM)[15], or neural networks[5][6].

The motivation of this work originates from three limitations of traditional LPR systems. The first limitation is using simple features such as gradient energy to detect possible locations of license plates. Using these simple features may reduce the complexity of computation but may possibly lose some plate candidates because the gradient energy will be suppressed due to camera saturation or underexposure, which often takes place under extreme light conditions such as sunlight, night view, or shadow. The second limitation originates from assuming correct

orientations for both camera and license plates so that high gradient pixels in the image can be expected in the pre-defined direction. In real cases, the license plates may not always keep the same orientations in the captured images. Nevertheless, they can be rotated or slanted due to irregular roads, unfixed camera positions, or abnormal conditions of cars. The third limitation comes from blurred or corrupted characters in license plates, which may fail the LPR process in detection or segmentation stage. The characteristic is dangerous for application because one single unclear character may result in loss of whole license plate. Compare to human nature, people know the position of unclear characters because they see some characters located nearby. Human try different methods, e.g., change head position or walk closer, to read the unclear characters, or even guess it if it is still not distinguishable. This nature is not achievable in a traditional LPR system due to its coarse-to-fine architecture. To retain high detection rate of license plates under these limitations, the method in this work proposes a fine-to-coarse method which firstly finds isolated characters in the captured image. Once some characters on a license plate are found, the entire license plate can be detected around these characters. This method may consume more computation than the traditional coarse-to-fine method. However, it minimizes the probability of missing license plate candidates in the detection stage.

A challenge to do the fine-to-coarse method is recognizing isolated characters. There are few literatures discussing about isolated characters recognition due to several difficulties it has. First, it is difficult to extract orientation of an isolated character. In traditional LPR systems, the orientations of characters can be determined by the baseline [3][8] of multiple characters. However this method is not suitable for isolated characters. Second, the unfixed camera view angle often introduces geometric deformation on the character shapes or stroke directions. It makes the detection and normalization process difficult to be applied. Third, the unknown orientations and shapes exposed under unknown light condition and environment builds a bottleneck for the isolated characters to be correctly detected and recognized.

The proposed scheme to extract and recognize license plate characters has procedures as the following. First, in the extraction stage, the scale-space binarization(SSB) method which utilizes the difference-of-Gaussian (DOG) functions [9] is used to extract character candidates. The DOG function has been proven stable against noise, illumination change and 3D view point change [9]-[14]. The binarization method first localizes the character profiles on DOG image and then extracts isolated character candidates by means of dynamic threshold propagation and thresholding. Second, in the deformation correction stage, a voting boundary method is used to detect the linear boundary of character candidates, which can be used for correcting the candidates from some possible deformations. Third, in the recognition stage, the novel accumulated gradient projection vector method(AGPV method) is applied to find out the accumulated gradient projection vectors (AGPVs) of each normalized character candidate, and compare the AGPVs with those of standard letters to find the most similar one as recognition result. Fig. 1-1 shows the functional block of the proposed LPR system. The experimental results show the feasibility of the proposed method and its robustness to several image parameters such as noise, character deformation and illumination change.

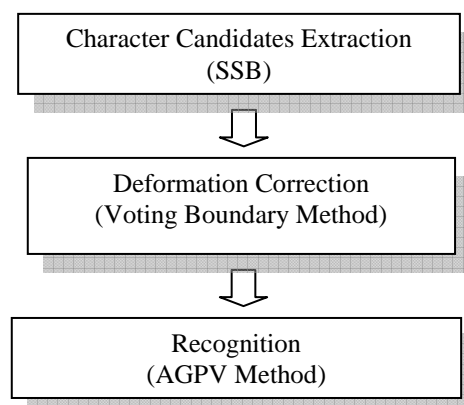


Fig. 1-1 Functional block diagram of the proposed LPR system

Chapter 2 Review of Related Works

This chapter briefly describes three important techniques from which this work is motivated and constructed. First, the methods dealing with recognition of license plate characters are reviewed. Second, the useful scale-space theory and its most popular representation, difference-of-Gaussian functions, are discussed. Finally, the most popular methods doing image binarization are described and compared.

2.1. License Plate Recognition

In traditional LPR systems, there is a detection function in the first step to find possible areas that license plates may appear. The function often requires high speed feature detection and therefore is generally focused on simple features such as gradient energy or Harr-like features[51] in the image. In order to make fast detection, traditional methods often suppose a fixed camera capture angle and allow a small degree of deviation in plate size and orientation. On the detected areas, more specific rules are used to accurately localize the entire license plate and find out the histogram for binarization. Once the plate is binarized, the corresponding baseline becomes an important reference for characters segmentation and normalization. Based on the binarized plate image, the segmentation is often done by projecting the TRUE pixels onto baseline and finding the valley on the projected histogram as segmentation boundaries. For the segmented characters, the statistical features of them are extracted and fed into a statistical classifier such as template matching[16], vector quantization[4], support vector machine(SVM)[15], or neural networks(NN)[5][6], for recognition. The statistical features include some vectors such as CC(contour-crossing count)[46], PBA(peripheral ground area)[47], and CS(character shape), that are common used for recognizing license plate character.

2.2. Scale Space Theory

The concept of scale space [11] starts from the basic observation that real-world objects are composed of different structures at different scales. In other words, real-world objects may appear in different ways depending on the scale of observation. For a computer designed to detect the existence of an object in an image, it is necessary to consider all the possible scales that object may appear in the image in order to capture the interested target in the correct scale.

Earlier works such as [12] and [13] have suggested that Gaussian function is the best choice for scale-space kernel. Also, in [13], the author showed that the difference-of-Gaussian(DOG) function provides a close approximation to the scale-normalized Laplacian of Gaussian, $\sigma^2 \nabla^2 G$, which was proven by detail experiment in [14] that it produces the most stable image features compared to a range of other possible image functions.

There are two additional advantages using Gaussian functions as smoothing kernel. First, its symmetric property makes it practical to decompose the two-dimensional convolution into two independent single dimensional equations. This greatly reduces the computation and shortens processing time in computing different scale images. Second, taking the Fourier transform of a Gaussian function yields another Gaussian function [17]. Consequently, it can be derived that the successive convolution with Gaussian kernel $G(\sigma_2)$ and $G(\sigma_1)$ is equivalent to convolution with $G(\sigma_3)$, where

$$\sigma_3 = \sqrt{\sigma_1^2 + \sigma_2^2} \quad (1)$$

Based on (1) and assumed that a Gaussian point spread function (PSF) is used to approximate the image capturing process[18], it can explain that the blur in input image can be ignored if a sufficiently large observation scale is chosen since $\sigma_3 \sim \sigma_2$ if $\sigma_2 \gg \sigma_1$.

2.3. Image Binarization

The methods for binarization of gray-level images can be divided into two classes: global and local thresholding. Global thresholding methods generally binarize the image with a single threshold. In the contrast, local methods change the threshold dynamically over the image according to local information. The threshold for global methods is often easier to be determined than that of local methods because it focuses on the entire image. However, global methods are easily failed when the dealt image contains noise, variable illumination, or complex background. Local thresholding methods have better adaptability than global ones to deal with illumination change or complex background, however, it is difficult to decide the range of local area for threshold determination and yet still sensitive to noise.

Global thresholding methods often calculate the threshold based on histogram analysis [7], [20]-[21]. Otsu's method [7] proposed from the viewpoint of discrimination analysis is one of the most preferred global techniques by investigators. It directly approaches the feasibility of evaluating the "goodness" of threshold and automatically selects an optimal threshold from the zeroth- and the first-order cumulative moments of the gray-level histogram. In practice, this method does not work well for the images with shadows, inhomogeneous backgrounds, and complex background patterns [22]. It is also discovered in [22] that, a single threshold or some multilevel global thresholds could not result in an accurate binary image.

Local thresholding methods generally find thresholds by statistical measurement in local areas [23]-[26] based on the principle that objects in an image provide high spatial frequency components and illumination consists mainly of lower spatial frequencies [31]. The local intensity gradient (LIG) method in [23] is one of the most popular local thresholding methods which first finds the pixels with high intensity gradient as reference of initial threshold, and then extends the threshold to whole image through region growing method [30]. It uses a

predetermined window size to calculate the regional gradient means, locates low gradient areas in the image based on the regional means, and finds edge pixels by comparing pixel's intensity gradient with the regional means.

In general, local thresholding methods are, considered from real world situations, more accurate than global ones. However, they still suffer from two problems that usually make them unsatisfactory for investigators. First, it is difficult to give a proper size of the "local area" without prior information in the source image. Second, the methods of this class are usually more computationally expensive than the other one; it makes the local methods almost unacceptable for real-time applications.

There are still some hybrid methods to binarize the image by referring to the expected content within the region of interest (ROI). Typical applications performing hybrid binarization such as license plate recognition (LPR) or automatic document analysis, often segment the image into areas and find the areas which are most likely to be ROIs before binarization. Such systems often have faster speed and higher accuracy than general (global or local) thresholding methods but usually require prior information within the ROI for fast detection and binarization. For example, in the LPR system [3], the author uses Haar-like features in the first step to perform fast detection and find out the ROI (license plate candidates), and then perform peak-valley analysis within the ROI for binarization of the license plates candidates. The peak-valley analysis is referring to the histogram acquired in the ROI and assumes some parameters such as number of characters, characters scale and orientation are already known. In document binarization method [28], the input image is firstly segmented into different types ROIs containing different contents such as characters or graphics or images. And specific binarization methods are applied within the ROIs based on the characteristics of the type of contents. In usual, the hybrid methods are not general enough to be applied onto different applications.

Chapter 3 The Extraction Method

The problem of character extraction is similar to that of object localization [38], where the largest bottleneck is almost all relevant factors are unknown in the source image, e.g., the scales of the objects, the condition of illumination, the complexity of the background, and the degree of blur and noise..., etc. As the scale of observation is closely related to the scale of the characters in the image, an incorrect observation scale may incorporate undesirable information and lead to undesirable extraction results [32]. In order to do extraction robustly and efficiently, we propose a scale-space binarization method, or SSB in short, to extract the characters. The extraction is started from the smallest observation scale which has best discrimination for characters sized within a certain range, for example, 16×16 to 64×64 . Smaller sizes characters are discarded because they are most probably caused by noise. For larger sizes characters, a higher observation scale is preferred to minimize the probability of misinterpretation from noise. Note that the extraction on higher observation scales can be performed by utilizing the sub-sampling method to shrink the image size and enlarge the relative observation scale.

The proposed SSB method includes several functional blocks as illustrated in Fig. 3-1. First, a character profile localization block finds inner and outer profile pixels by applying a global threshold on difference-of-Gaussian(DOG) image. The DOG function used to generate the DOG image is proven to have the benefit of enhancing the edges in a digital image while minimizing the impact of noise [39]. Second, a boundary set is formed by collecting pixels neighboring to both inner and outer profile pixels. Third, the thresholds are initiated on boundary set pixels and served as the initial value for dynamic threshold propagation. Fourth, the dynamic thresholds are propagated from boundary to the remaining pixels in the image. Fifth, thresholding function compares the dynamic threshold with smoothed gray-level intensity

to binarize the image. Sixth, connected component analysis is applied to connect pixels into character candidates, and measure their preliminary features such as width, height and occupancy for the next stage. Finally, the character candidates are eliminated if their preliminary features fall beyond reasonable ranges or the profile scores are lower than general characters. An example on the simulation results of the SSB method is given in Fig. 3-2 for easy understanding. In the next sections we'll step by step explain the behavior of each functional block in detail.

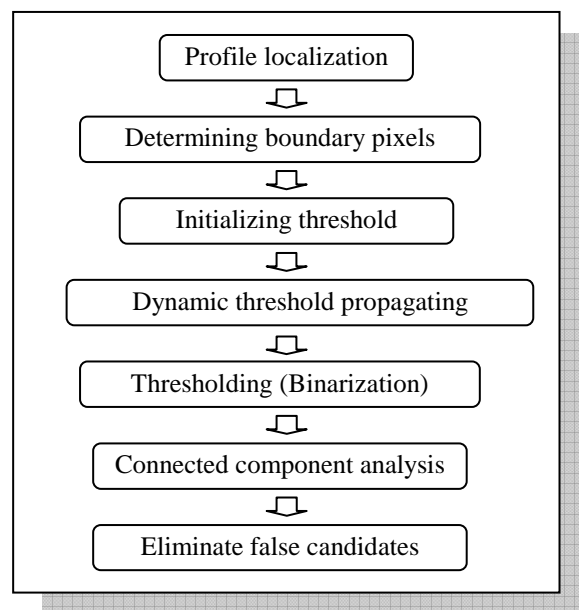


Fig. 3-1 Functional block diagram of the SSB method

3.1. Profile Localization

Profile localization, similar to edge detection, is often applied in the first stage of an image recognition process to locate pixels as the basis of segmentation or matching. Many operators can be found in literatures to detect edges or corners in an image, e.g., Sobel operator[40], Harris detector[41], or Canny detector[42]. Most of them use gradient based detection and suffer from the difficulties in noise rejection and threshold determination. The extraction method in this work utilizes the DOG functions so that it minimizes the impact of noise and

makes robust extraction without prior filtering.

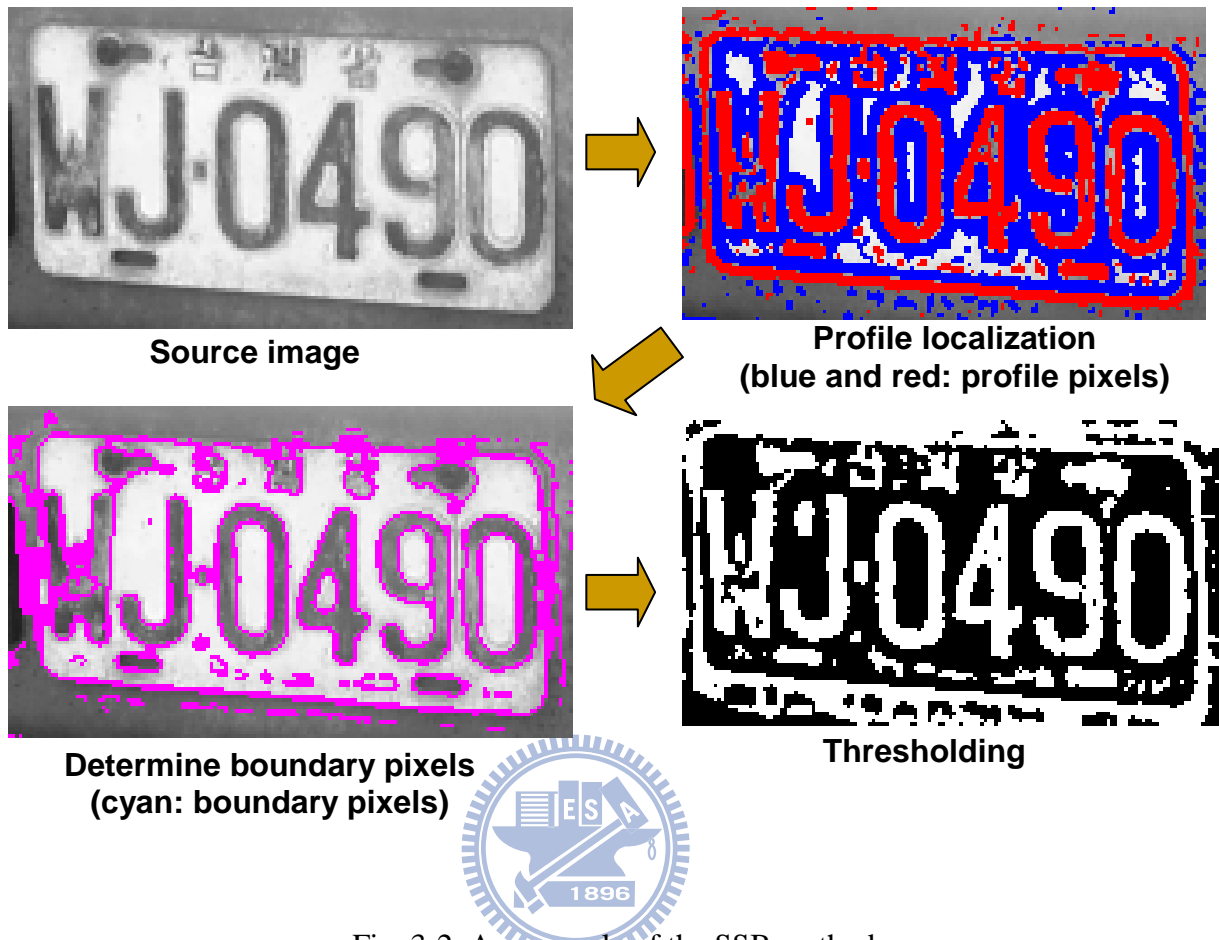


Fig. 3-2 An example of the SSB method

The profile localization consists of several steps as in the following procedures. At first, the gray-level input image, $I(x,y)$, is respectively convolved with two Gaussian functions, $g_1(x,y)$ with deviation σ_1 and, $g_2(x,y)$ with deviation σ_2 to get two Gaussian images, $I_1(x,y)$ and $I_2(x,y)$. And the difference of the two Gaussian images, $D_1(x,y) = I_1(x,y) - I_2(x,y)$, is called the DOG image.

The two standard deviations, σ_1 and σ_2 , of the two Gaussian functions are respectively called the first and the second observation scale. A smaller observation scale observes more details in an area but is more sensitive to noise. On the contrary, a larger observation scale is more stable against noise but may lose significant details of the interested characters or mix the interested characters with adjacent objects so that the characters become difficult to be extracted. In the

experiments we set the two scales $\sigma_1=1$ and $\sigma_2=\sqrt{2}$ for the profile extraction, which is proven by experiments a better choice for processing 16×16 to 64×64 character sizes in general 256-step (8-bit) gray-level images.

In order to deal with larger scale characters with minimum computation time, an efficient method in Fig. 3-3 is applied by sub-sampling the second blurred image $I_2(x,y)$ by every two pixels on each row and column to form a smaller image $I_2'(x,y)$. Then based on $I_2'(x,y)$ calculates the Gaussian filtered image $I_3(x,y)$ and their DOG image $D_2(x,y)$, and applies the same procedure again to localize the profile pixels. As a result, the observation scale w.r.t. $D_2(x,y)$ is double to that w.r.t. $D_1(x,y)$.

A 2-D DOG function used to extract the characters can be expressed as,

$$DOG(x, y) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{x^2-y^2}{2\sigma_1^2}} - \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{x^2-y^2}{2\sigma_2^2}}. \quad (2)$$

Consider a case that an unit step edge $u(x_0)$ exists in parallel to the y-axis($x=x_0$), the position of peak response on convolving the unit step edge with a DOG function can be obtained by solving the differential equation,

$$\frac{\partial}{\partial x} [DOG(x, y) \otimes u(x_0)] = 0. \quad (3)$$

Transforming into frequency domain and then taking inverse transform, the solution of (3) yields equivalent to that of the equation

$$DOG(x - x_0, 0) = 0. \quad (4)$$

Solving (4) to get the positions of the two peak responses at

$$x = x_0 \pm \sqrt{\frac{2\sigma_1^2\sigma_2^2}{\sigma_2^2 - \sigma_1^2} \cdot \ln \frac{\sigma_2}{\sigma_1}}. \quad (5)$$

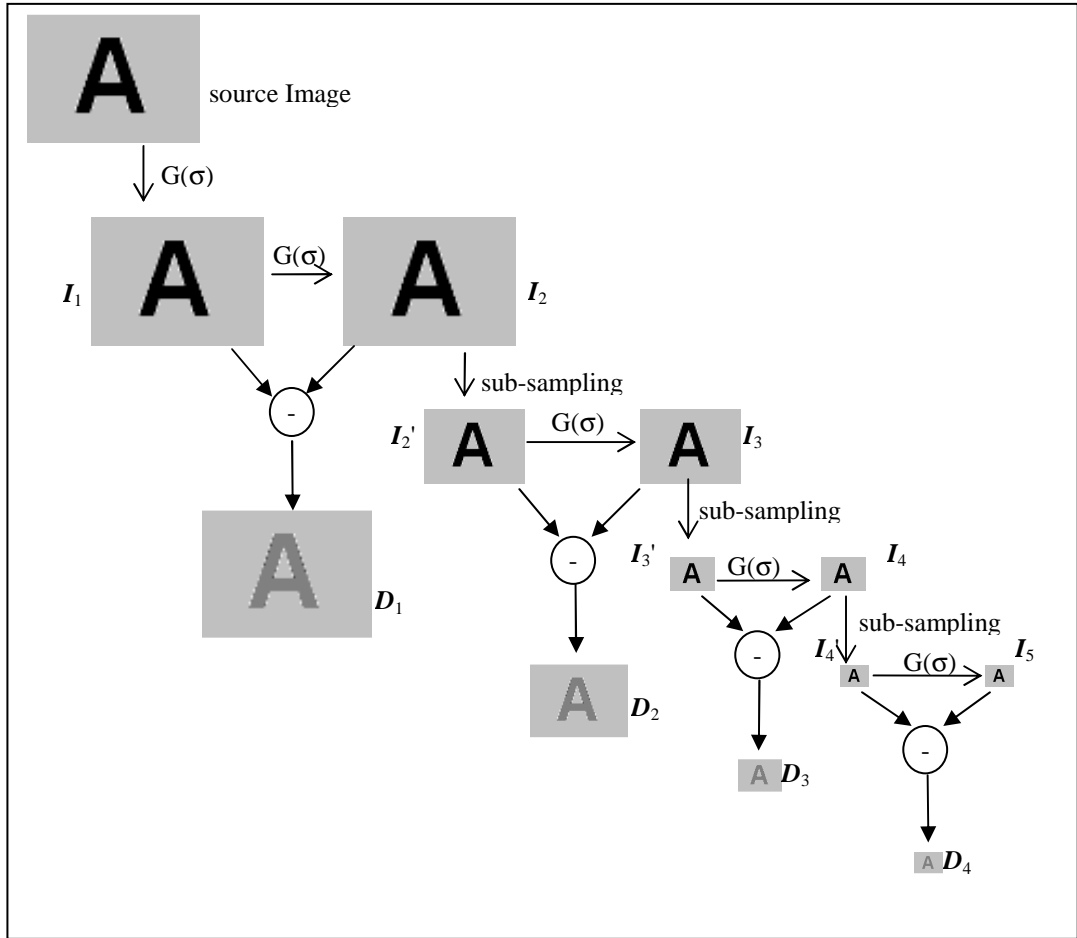


Fig. 3-3 The procedure to produce DOG Images on different observation scales

A plot by equation (5) in Fig. 3-4 on $x_0=0$ reveals that convolution of a unit step edge with the DOG function generates two odd-symmetrical peaks beside the unit step edge, i.e., positive peak A and negative peak B. The most valuable characteristic of the DOG function is that these peaks are quite stable even if the testing image consists of small undesirable artifacts such as noise, out-of-focus or variable illumination. Based on this result, the DOG image is divided into three sets by a fixed global threshold th_f and its complementary $-th_f$. The first set, Set_1 , is composed of the pixels of $D_1(x,y) \geq th_f$, the second set, Set_2 , is composed of the pixels of $D_1(x,y) \leq -th_f$; and the third set, Set_3 , is the superset of the remainder containing the pixels of $th_f > D_1(x,y) > -th_f$.

Set_1 and Set_2 are both called profile sets and have the following representation for easily

identification according to the way they appear. For characters having lower gray-level intensity (deeper color) than its nearby background, Set_1 is also called the inner profile set because it spreads interior characters' boundaries. Similarly, Set_2 is also called the outer profile set for the location it appears. The two profile sets are respectively drawn in Fig. 3-5 in blue and yellow colors.

The global threshold th_f is used for determining whether a change of intensity is caused by noise or a real edge. Smaller threshold collects more pixels into Set_1 and Set_2 , and takes more computation time to deal with noise before extracting the characters. It is worth notify that the lowest threshold for DOG function can be set to $th_f=0$. Although setting threshold to zero introduces much information generated by noise, it can still retain correct extraction results because that the energy of noise in the DOG response is automatically suppressed when it appears near an edge. As a result, it is recommended to set a small threshold, e.g., $th_f=1$, for all the input images because it ensures reliable results can be persisted with reasonable computation time regardless of the condition of the input image. Different from some other gradient operators which would possibly lose some character candidates if a smaller threshold is given, the only drawback for giving a smaller threshold in DOG function is higher computation time consumption. From various simulation results we can tell that a wide range of threshold on DOG images can still provide reliable results on localizing the profile pixels.

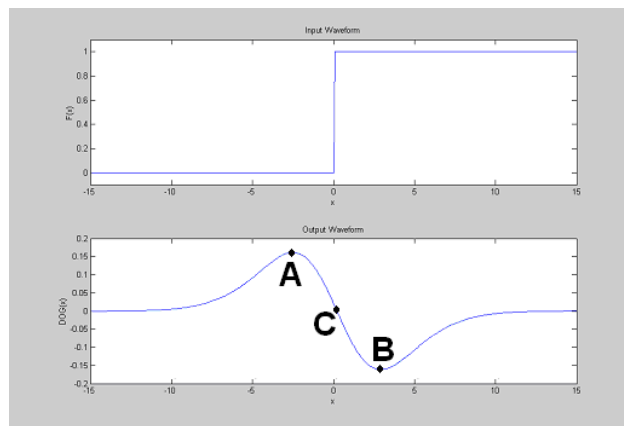


Fig. 3-4 An ideal unit-step edge (upper graph) and its DOG response (lower graph)

When a near-perfect input image like Fig. 3-6(a) is given for binarization, the first step is to find the corresponding two profile sets from the DOG image as in Fig. 3-6(b). It is worth to note that the pixels of the inner profile set often appear in a connected group, which is called the inner profile groups or simply profile groups. As in Fig. 3-6(c), the smallest rectangle covering the entire profile group is called the bonding rectangle of the profile group. Note that a profile group often represents the profile of an isolated character in normal case. However, it might happen that a character is broken into two or more profile groups due to special geometric distribution or noise or special lighting condition. The broken profile groups will be linked up by the connected component analysis later on to reveal the original characters.

According to (5), a constant R_{eff} is defined to represent the radius of the effective area of an edge (intensity change), and

$$R_{\text{eff}} = \text{ceil} \left(\sqrt{\frac{2\sigma_1^2 \sigma_2^2}{\sigma_2^2 - \sigma_1^2} \cdot \ln \frac{\sigma_2}{\sigma_1}} \right), \quad (6)$$

where the function $\text{ceil}(x)$ rounds x towards positive infinity. Note that the R_{eff} is the horizontal distance of AC or BC in Fig. 3-4, or equivalently the radius of the circle of effective range in Fig. 3-5. In addition to the profile sets, a boundary set Set_B is formed to represent the boundary of character candidates. A pixel p_b is collected into Set_B if it satisfies the following two conditions,

1. Except the zero-crossing pixels, i.e., the position **C** in Fig. 3-4, or the non-profile pixels in Fig. 3-5, the pixels inside the effective area of p_b belong to either inner or outer profile sets.
2. The total number of pixels belongs to inner profile set and the total number of pixels belongs to outer profile set inside the effective area of p_b are the same.

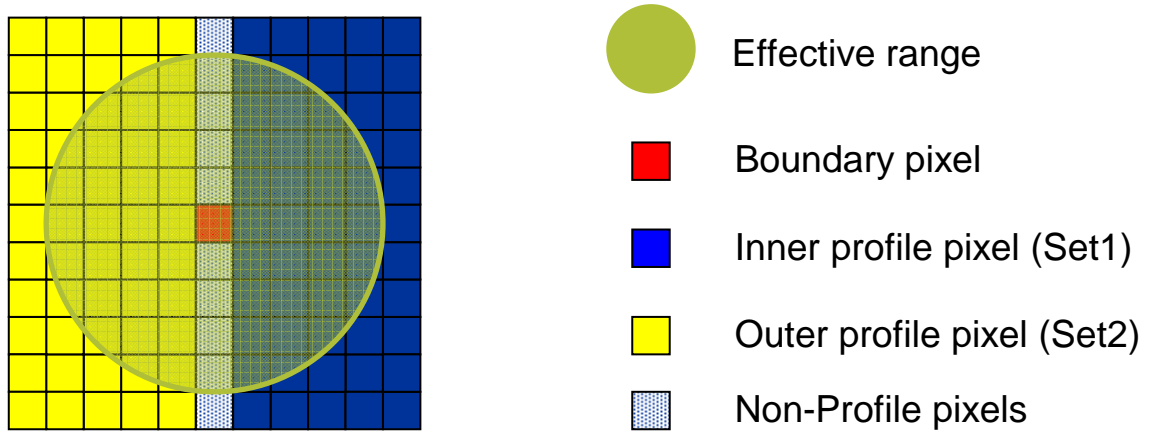


Fig. 3-5 Determine boundary pixels

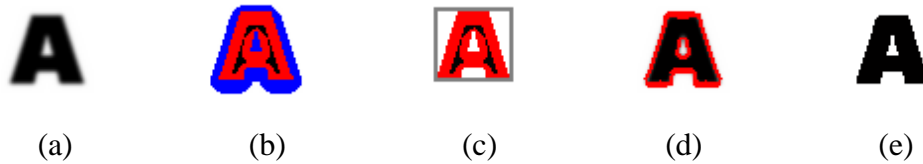


Fig. 3-6 (a) A perfect sample character image. (b) The DOG responses: positive response in red and negative response in blue. (c) The inner profile set in red and the bonding rectangle in gray. (d) Boundary set. (e) Extraction result.

In implementation, consider to discrete pixel coordinate and error tolerance, the pixels of Set_1 and Set_2 inside the effective area of p_b are accumulated into Bin_1 and Bin_2 respectively, and p_b is collected into Set_B if it satisfies the following equations:

$$\begin{cases} Bin_1 + Bin_2 \geq \text{round}((R_{eff} - 1)^2 * \pi) \\ |Bin_1 - Bin_2| < R_{eff} * 2 \end{cases} \quad (7)$$

The pixels of Set_B make up the boundaries of character candidates as in Fig. 3-6(d) and become the base of threshold propagating in the next step. Note that the character can be extracted as in Fig. 3-6(e) after dynamic threshold propagation and binarization.

3.2. Dynamic Threshold Propagation

In order to solve the global-thresholding problems such as noise, variable illumination, and complex background, and local-thresholding difficulties such as pre-determining local area size, and reducing computational complexity, a novel method using dynamic threshold propagation is proposed in this work.

Before the propagation process, each pixel in the image is assigned a dynamic threshold initialized to zero. As the process starts, the dynamic threshold on a boundary set pixel is assigned by looking for the best threshold in its neighboring area. Based on the values assigned to boundary set pixels, the dynamic thresholds are sequentially propagated to the remaining pixels through neighboring pixels. As a result, the thresholds detected around boundary pixels are able to spread out to the entire image so that the interested characters can be figured out by comparing gray-level intensity with the dynamic threshold pixel-by-pixel.

The first step of dynamic threshold propagation starts from the boundary pixels. For each boundary pixel (x_b, y_b) , the gradient magnitude of the i -th neighboring pixel $(x_n(i), y_n(i))$ inside the effective area, $effa(x_b, y_b)$, is calculated. The pixel having maximum gradient magnitude inside the effective area is selected as the reference pixel (x_{n_ref}, y_{n_ref}) . In other words, $(x_{n_ref}, y_{n_ref}) \in effa(x_b, y_b)$ and $|\nabla I_1(x_{n_ref}, y_{n_ref})| = \max(|\nabla I_1(x_n(i), y_n(i))|)$, where $|\nabla I_1(x_n(i), y_n(i))|$ is the gradient magnitude of the i -th neighboring pixel calculated by Sobel operators as in [23]. After that, the dynamic threshold of the boundary pixel, denoted as $th_d(x_b, y_b)$, is assigned by the first Gaussian gray-level of the reference pixel (x_{n_ref}, y_{n_ref}) , i.e.,

$$th_d(x_b, y_b) = I_1(x_{n_ref}, y_{n_ref}). \quad (8)$$

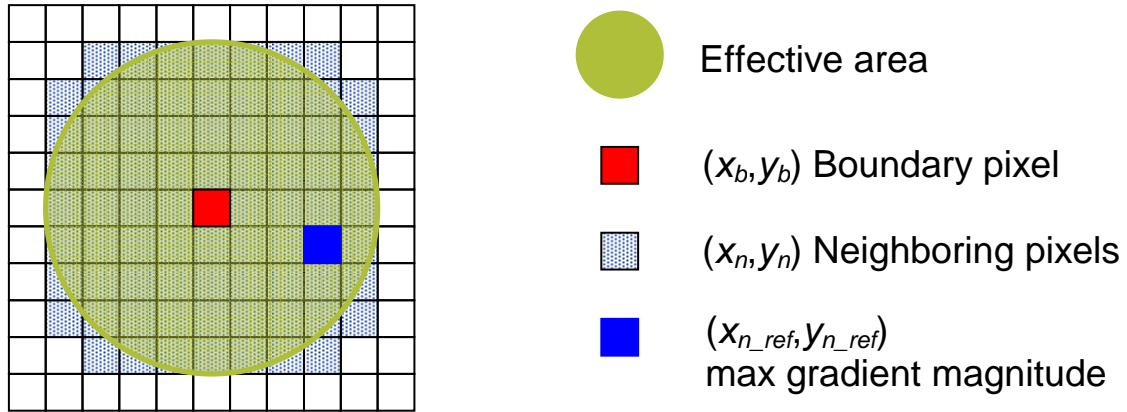


Fig. 3-7 Dynamic threshold propagation

The above definition can be viewed in graphical representation as in Fig. 3-7, where the red pixel is one of boundary set pixels and the blue pixel is the one having maximum gradient magnitude inside the effective area. The reason for referring to the pixel of maximum gradient magnitude is based on the discovery that the pixels having maximum gradient magnitude often appear in the mid point of edges. It is worth to note that the calculation of gradient magnitude is referring to the first Gaussian image $I_1(x,y)$ instead of source image $I(x,y)$ and the second Gaussian image $I_2(x,y)$ because of the following two reasons: First, the condition of noise in the source image $I(x,y)$ is unknown; the gradient referring to noisy pixels is not meaningful and may mislead the decision in finding correct threshold. Second, the second Gaussian image gives too much smoothness on boundary so that it often makes the boundary distorted after thresholding. As a result, the first Gaussian image is the best choice for gradient magnitude comparison and dynamic thresholding.

Once the dynamic thresholds of all boundary pixels have been assigned, they are iteratively propagated to the other pixels through neighboring pixels. For easily explanation, a pixel whose dynamic threshold has been assigned is called an assigned pixel.

The dynamic threshold propagation is processed by iterations. Let Set_b^p denote the set of pixels (x_b^p, y_b^p) whose dynamic thresholds are assigned in the p -th iteration and Set_a^p denote the

set of the pixels (x_a^p, y_a^p) adjacent to Set_b^p in the same iteration. Note that Set_b^1 stands for the boundary set containing assigned pixels (x_b^1, y_b^1) and $Set_b^k = Set_a^{k-1}$ for $k > 1$. In the first iteration, the dynamic thresholds on boundary pixels (x_b^1, y_b^1) are propagated to its adjacent pixels (x_a^1, y_a^1) . Let $(x_a^1(h), y_a^1(h))$ be the h -th pixel in Set_a^1 simultaneously adjacent to m boundary pixels, denoted as $(x_b^1(i), y_b^1(i))$, $i=1$ to m , m can be any number from 1 to 8. The dynamic threshold of $(x_a^1(h), y_a^1(h))$ is assigned by averaging the dynamic thresholds of all the adjacent boundary pixels, i.e.,

$$th_d(x_a^1(h), y_a^1(h)) = \frac{1}{m} \sum_{i=1}^m th_d(x_b^1(i), y_b^1(i)). \quad (9)$$

The first iteration ends and the next iteration starts right after all the pixels adjacent to the boundary pixels have been processed. In a general representation, the relationship for the q -th iteration is,

$$th_d(x_a^q(j), y_a^q(j)) = \frac{1}{m_j^q} \sum_{i=1}^{m_j^q} th_d(x_b^q(i), y_b^q(i)), \quad (10)$$

where $(x_a^q(j), y_a^q(j))$ is the j -th pixel in Set_a^q and m_j^q is the total number of assigned pixels adjacent to $(x_a^q(j), y_a^q(j))$. The propagation process will not finish until all the pixels become assigned.

An example of the propagation can be seen in Fig. 3-8. The red pixels in Fig. 3-8(a) are the boundary pixels with the dynamic threshold initialized according to Eq. (8). The orange pixels in Fig. 3-8(b) and the yellow pixels in Fig. 3-8(c) are respectively the pixels after first and second iteration of dynamic threshold propagation.

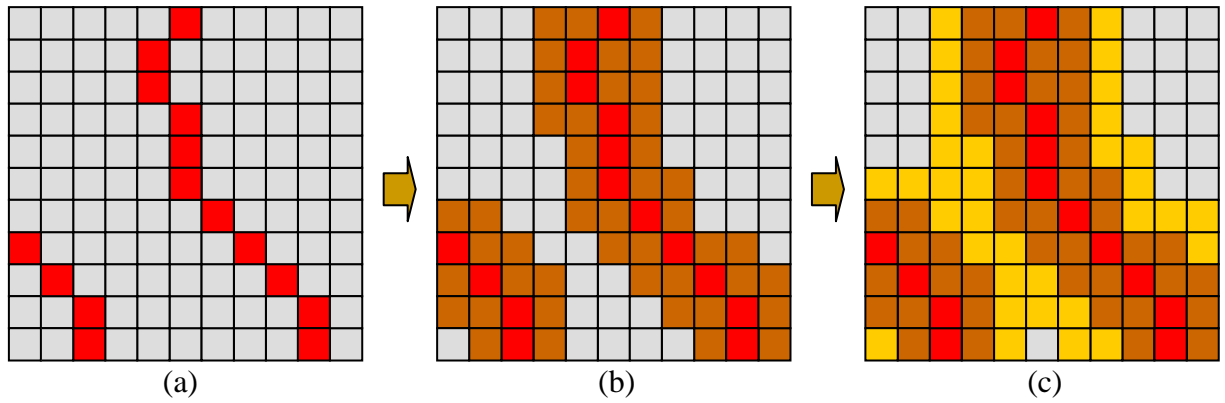


Fig. 3-8 An example of dynamic threshold propagation (a) Boundary pixels (b) Assigned pixels after first iteration (c) Assigned pixels after second iteration

3.3. Thresholding and Connected Component Analysis

Based on the propagated dynamic thresholds, each pixel is converted into binary form (TRUE or FALSE) by comparing its Gaussian smoothed gray-level intensity to the own dynamic threshold. Then the connected component analysis (CCA) is applied to connect the TRUE pixels into groups named isolated groups.

During the CCA process, the TRUE pixels of each isolated group are divided into two classes and accumulated into two counters, respectively. The first class is edge pixels, which is adjacent to at least one FALSE pixel after the CCA and is accumulated into counter C_E . The second class is body pixels, which is the complementary to edge pixels, i.e., all the eight adjacent pixels are TRUE, and is accumulated into counter C_B . The total number of pixels in an isolated group is denoted C_T , $C_T = C_E + C_B$. For edge pixels, another counter C_P is allocated to accumulate the number of pixels adjacent to Set_1 or Set_2 pixels in order to give profile score to the isolated group.

In normal cases as shown in Fig. 3-6, the profile pixels of an isolated character can be connected into an individual profile group, and the whole character should belong to an individual isolated group, too. However, it might happen that the profile group is broken into

segments due to noise or irregular light condition as the example in Fig. 3-9(a)-(d). In this case the broken profiles can still be extracted into an connected group as in Fig. 3-9(e) after thresholding and the connected component analysis.

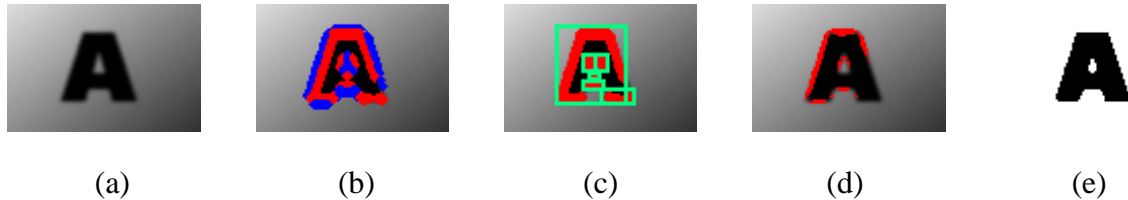


Fig. 3-9 (a) An imperfect sample character image. (b)The DOG responses: positive response in red and negative response in blue. (c) The inner profile set in red and the bonding rectangle in cyan. (d) Boundary set.(e) Extraction result.

3.4. Eliminate False Candidates

There are two stages elimination to filter out the false character candidates in order to minimize the computational consumption in later stages.

The first stage elimination is based on the geometric features captured by the CCA process. After the CCA, each isolated group has own preliminary features measured by its bonding rectangle, i.e., group width W , group height H , group occupancy U (pixel count w.r.t. the bonding rectangle area), $U=C_T/(W \times H)$. The groups having abnormal preliminary features are possibly caused by non-character objects such as noise or background or variable illumination and are eliminated immediately. For example, large ratio of W to H may represent a long edge or a thin line in the image; small W and H may be caused by noise or a spot; large U may stand for a solid object or shadow..., etc. General characters have a typical value for occupancy ranged in $0.3 \leq U \leq 0.8$.

The second elimination is based on a quantity that measures from the “goodness” of the

profiles of each isolated group, namely, the profile score S_p . In ideal cases, the edge pixels found by the CCA should be adjacent to Set_1 or Set_2 pixels. i.e., $C_p = C_E$. However, in real world images, it is often not the case and most are $C_p < C_E$. Therefore, the profile score defined by $S_p = C_p/C_E$ is calculated for each isolated group to evaluate how much goodness it is from the ideal case. In our experiments, the isolated groups having $S_p < 0.8$ is eliminated. The remaining isolated groups form the character candidates and can be used for recognition or other purposes hereafter.

3.5. Implementation for Fast SSB

Besides a stable and accurate performance, the computational complexity of a binarization algorithm is also important in evaluating the performance. The demand for low computational complexity methods is especially strong in a real-time embedded system. In such systems they require low computational complexity methods for not only speeding up the response to external events but also reducing the power consumption. Although the computational complexity of the method presented here is higher than a global thresholding method, a good implementation can still make it computed efficiently and executed as fast as a global method. Of course, it is expected to compete with the most local thresholding methods both in accuracy and speed.

The problems to be discussed here is similar to the optimization in implementation. For the proposed method, the optimization can be considered from several aspects,

1. Simplify the convolution with Gaussian filter.
2. Use integers instead of floating points.
3. Use shifter to replace multiplier or divider.
4. Use acceleration table for dynamic threshold propagation.

3.5.1. Optimization in convolution

The convolution with Gaussian kernel takes much computation time because it is directly proportional to the size of the input image and the Gaussian kernel. Let W denote the width and H denote the height of the input image, and give a Gaussian kernel sized $n \times n$. To convolve the input image with the Gaussian kernel, it needs $H \times W \times n^2$ multiplications and $H \times W \times (n^2 - 1)$ additions. Due to the symmetrical properties of a Gaussian function, the 2D convolution can be decomposed into horizontal and vertical direction. For each direction, $n \times 1$ dimensional Gaussian function is used so that $H \times W \times n$ multiplications and $H \times W \times (n - 1)$ additions is required. This simplifies the complexity from $O(n^2)$ to $O(n)$.

3.5.2. Implement by integers and shifters

In computer systems, integer manipulation is always faster than floating points. Especially, many computer systems still have no hardware floating point processor and allow only manipulations by integers. On the other hand, multiplications or divisions often take longer computation time than simple manipulations such as addition, subtraction, or shifter; it would be preferred if they can be replaced by shifter for speeding up the computation and making the algorithm more practical on various grade computer systems. Consider to implement by integer and shifter in the program, we decide to select the Gaussian kernel as $G(x) = G(y) = [1 \ 4 \ 8 \ 4 \ 1]$. Fig. 3-10 gives a comparison to the three normalized Gaussian functions: selected Gaussian kernel in Gau3, ideal continuous Gaussian function ($\sigma=1$) in Gau1, and ideal discrete Gaussian function in Gau2. It shows that the selected Gaussian kernel is close to the ideal discrete Gaussian function.

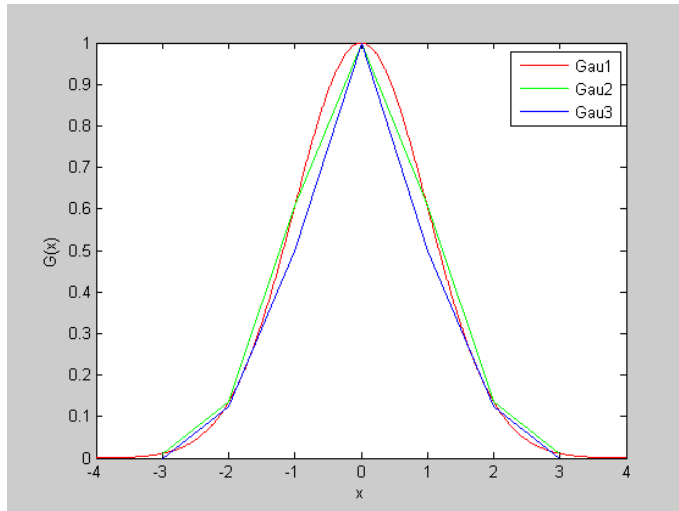


Fig. 3-10 Comparison to three Gaussian functions, Gau1: ideal continuous function, Gau2: ideal discrete function, Gau3: selected kernel

The advantages of using the selected Gaussian kernel are, first, the coefficients are integers; second, all the coefficients are 2's multiples so that the multiplications can be replaced by shifters. Based on the selected kernel, the convolution for the first Gaussian image $I_1(x,y)$ ($\sigma_1=1$) can be written as

$$I_1(x,y) = (I(x,y) \otimes G(x)) \otimes G(y) \quad (11)$$

This can be achieved in program 1,

```

=====
                          Program 1
////////////////////////////////////
T [x][y]=(I [x-2][y]+I [x+2][y]) + ((I [x-1][y] + I [x+1][y])<<2) + (I [x][y]<<3);
I1[x][y]=( T [x][y-2]+ T [x][y+2]) + ((T [x][y-1] + T [x][y+1])<<2) + (T [x][y]<<3);
=====

```

Where $T [x][y]$ is an intermediate array, $I [x][y]$ is the gray-level intensity on $I(x,y)$ and $I_1[x][y]$ is the Gaussian smoothed gray-level intensity on $I_1(x,y)$. The “<<” operator denotes the left shifter. According to (1), the second Gaussian image $I_2(x,y)$ can be obtained by convolving $I_1(x,y)$ with the same Gaussian kernel, i.e.,

$$I_2(x,y) = (I_1(x,y) \otimes G(x)) \otimes G(y). \quad (12)$$

The same program1 can be used by substituting $I_1[x][y]$ with $I_2[x][y]$ and $I[x][y]$ with $I_1[x][y]$. Note that the equivalent scale for $I_2(x,y)$ is $\sigma_2 = \sqrt{2}$ based on equation (1). Finally, for computing the DOG image, the two Gaussian images must be normalized to the same level. Therefore, the summation of the Gaussian kernel must be eliminated. The equation is written as

$$D(x, y) = I_2(x, y) - I_1(x, y) \times \sum_x G(x) \times \sum_y G(y). \quad (13)$$

Since the $\sum_x G(x) = \sum_y G(y) = 18 \times 18 = 324$ and $324 = 256 + 64 + 4 = 2^8 + 2^6 + 2^2$. As a result, the

program to find the DOG image is implemented as:

```

=====
                          Program2
////////////////////////////////////
D[x][y]= I2[x][y]- (I1[x][y]<<8) - (I1[x][y]<<6) - (I1[x][y]<<2);
=====

```

It is important to check if the value in each step manipulation exceeds the full range of integers of a computer system and trim some least significant bits (LSBs) from the operands if necessary. For a 8-bit gray-level input image, the maximum value of $I_1(x,y)$ is 324×256 which becomes 18-bit signed integers. And the maximum value of $I_2(x,y)$ is $324 \times 324 \times 256$ which is extended to 26-bit. For a 16-bit computer system implemented by the proposed method, the program to calculate $I_1(x,y)$ and $I_2(x,y)$ can be changed to program 3 to avoid integers overflow.

```

=====
                          Program3
////////////////////////////////////
T [x][y]=((I [x-2][y]+I [x+2][y]) + ((I [x-1][y] + I [x+1][y])<<2) + (I [x][y]<<3))>>2;
I1[x][y]=(( T [x][y-2]+ T [x][y+2]) + ((T [x][y-1] + T [x][y+1])<<2) + (T [x][y]<<3)) >> 4;

Y [x][y]=((I1 [x-2][y]+I1 [x+2][y]) + ((I1 [x-1][y] + I1 [x+1][y])<<2) + (I1 [x][y]<<3)) >> 4;
I2[x][y]= ( Y [x][y-2]+ Y [x][y+2]) + ((Y [x][y-1] + Y [x][y+1])<<2) + (Y [x][y]<<3) ;

D[x][y]= I2[x][y]- ((I1[x][y]<<4) + (I1[x][y]<<2) + (I1[x][y]>>2));
=====

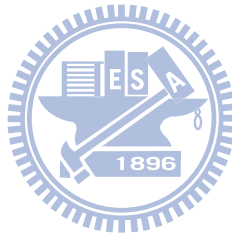
```

3.5.3. Use acceleration table for dynamic threshold propagation

The dynamic threshold propagation often runs over ten iterations for a typical input image

sized 640×480. It is very time-consuming if a whole-image scan is performed on each iteration. Therefore, an acceleration table is used to reduce the time required for propagation.

The acceleration table is composed of two first-in-first-out (FIFO) memories, namely FIFO A and FIFO B. When the propagation is started from boundary set pixels, the coordinates of the adjacent pixels that belong to non-boundary pixels, i.e., pixels of Set_a^1 , are sequentially stored into FIFO A. After a whole-image scan, all the boundary pixels are visited and the locations for the adjacent non-boundary pixels are saved. Then, in the second iteration, the propagation starts from the pixels saved in FIFOA, they become Set_b^2 pixels for this iteration. Again, the coordinates of the pixels adjacent to Set_b^2 form Set_a^2 and are stored into FIFO B. The process repeats the same flow and toggles FIFO A and FIFO B by iteration. As a result, only one full image scan is required for the first time and the computation is greatly reduced by the way.



Chapter 4 The Deformation Correction Method

In general cases, the license plate characters are often involved with certain degree of deformation when they are projected into two-dimensional images. The deformation in turns of mathematics could be composed of any transformation such as rotation, scaling, affine transform or mixed transformations..., etc. It is difficult to recognize these characters without correcting the deformation beforehand. In this chapter a novel method is discussed to correct the extracted characters in the proposed license plate recognition system.

4.1. Useful Properties for Deformation Correction

The extracted character candidates are not suitable for recognition directly because they probably undergo some geometric transformations such as rotation, affine deformations or mixed deformation...due to abnormal camera location or capture angle. The method in this section tries to eliminate the geometric transformations of character candidates and transform them into normal orientation for stable recognition. Fig. 4-1 shows some typical transformations from normal plate image in Fig. 4-1(a) such as rotation in Fig. 4-1(b), affine deformation in Fig. 4-1(c) and mixed deformation in Fig. 4-1(d). Due to the difficulties in finding invariant reference points, we utilize two useful properties for license plate characters to eliminate the undergone geometric transformations. The properties may not be sufficient to make perfect recovery from the deformation; however they can be used to detect the deformation and correct it in certain degrees to improve the successful rate in recognition.

The first property used for correcting geometric deformation of character candidates is the baseline. The baseline is an invisible line above which all the characters on a license plate are aligned. For various geometric deformations such as Fig. 4-1(b)-(d), the baseline can be used to correct a part of them, e.g., Fig. 4-1(b). However, for some other deformations, e.g., Fig.

4-1(c)-(d), it needs more information in addition to baseline to correct them for recognition. In order to correct from these complex deformations, a second property is adopted by referring to the horizontal boundary lines of each candidate. Unlike the baseline belonging to a group of character candidates, the horizontal boundary lines are the left and right boundaries belonging to a single character candidate which can be used to normalize the slant angle of each character candidate so that it can be changed to a state suitable for feature extraction and recognition.

Before locating the baseline, the character candidates are grouped by their sizes and positions. The rules of license plates [48] with an acceptable tolerance are used to check if the character candidates belong to the same license plate. The candidates obeying the rules will be grouped and considered as a single license plate. For each group of character candidates, a baseline is expected to exist below and can be found by the following methods.



Fig. 4-1 Typical geometric transformations in LPR systems

4.2. Voting Boundary Method

The voting boundary method is suitable to find boundary lines of a group of pixels in an image. It works by assuming many straight line candidates and detecting the best one passing through most of the edge pixels by voting. The method is in some respects similar to Hough transform[49] and has the same advantage with it in robust detection. However, it simplifies the computation from Hough transform by replacing the complex trigonometric functions with simple additions and subtractions.

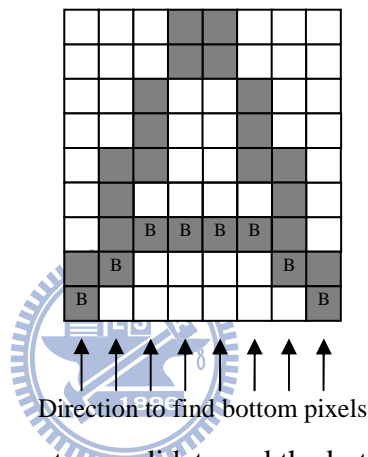


Fig. 4-2 A character candidate and the bottom pixels

Before the voting boundary method, it is required to find the edge pixels in four directions, respectively top, bottom, left and right boundary pixels. The edge pixels are the most outside pixels of an image group. For example, the bottom pixels are defined as the set of pixels that first appear when searching from bottom to top on each vertical pixel line. Fig. 4-2 shows an example on how to find the bottom pixels, where the gray pixels are grouped by connected component analysis in the extraction stage and the pixels marked as 'B' are the bottom pixels found according to the definition above. The principle for computing the voting boundary method starts from similar triangles. Let's see Fig. 4-3 for example, in the similar triangle pair $\triangle ABC$ and $\triangle ADE$, it is known that $a \times d = (a + b) \times c$. Let the line \overline{NG} be one of the bottom boundary lines of the pixel groups inside rectangle $MNOP$ and the black circles are the

corresponding bottom pixels. The distances from the bottom edge, \overline{NO} to each bottom pixel are stored in an array BP , where the array has w elements $BP[x]$, $x=1$ to w . If $BP[x]$ is on the line \overline{NG} , then it satisfies

$$x \times g = (w) \times BP[x]. \quad (14)$$

Consider to include error tolerance and rearrange the equation, the $BP[x]$ is on line \overline{NG} if it satisfies

$$\begin{cases} x \times (g/w) < BP[x] + r \\ x \times (g/w) \geq BP[x] - r \end{cases} \quad (15)$$

where the variable r represents thickness of the boundary line and can be adjusted according to different applications.

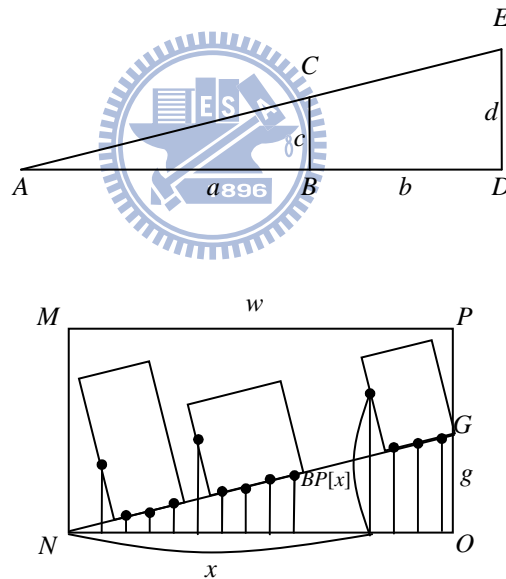


Fig. 4-3 Derivation of the voting boundary method

Each boundary pixel is voted into one of the following three sets according to the inequality pairs: the first set **FIT** if a boundary pixel satisfies the both inequality, the second set **UNDERFIT** if a boundary pixel falls in the range $x \times (g/w) \geq BP[x] + r$, and the third set **OVERFIT** if a boundary pixel meets the condition $x \times (g/w) < BP[x] - r$. Let the pixels voted to set **FIT** be p_i , $i=1$ to n . The coordinates of p_i are respectively (x_i, y_i) and $x_1 < x_2 < x_3 < \dots < x_n$. On

each boundary line candidate, the distance of start pixel p_1 on coordinate (x_1, y_1) , and end pixel p_n on coordinate (x_n, y_n) , are measured as $d = \sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}$ and treated as the length of the boundary line.

The process to vote boundary lines is drawn in Fig. 4-4, where it can be seen that the computation is very simple because of continuity of the x -axis. Only one division representing the angle between the boundary line and the x -axis is required at the beginning and few additions or subtractions are required afterward. After the voting process, the line gets the most votes in set **FIT** is assigned to be the true boundary line of the pixel group. Note that the set **UNDERFIT** and **OVERFIT** can be referenced to delete improper character candidates if any one of them is abnormally large.

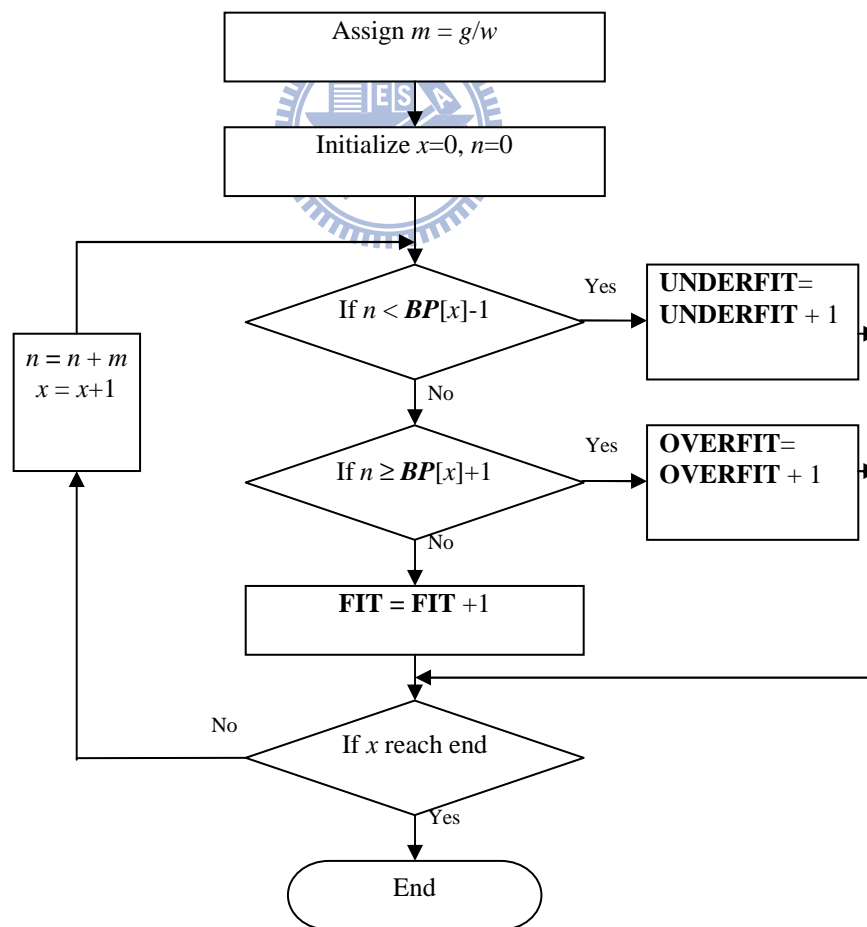


Fig. 4-4 Flow chart to vote boundary lines

4.3. The Correction Method

The method used to find the baseline is first locating the bottom pixels of each character candidates, and then use voting boundary method to find a line that most bottom pixels pass through. After finding the bottom pixels of each character candidate, the voting boundary method is applied to detect the baseline passing through most of the bottom pixels.

Once the baseline is detected, the next step is to correct rotation angles of character candidates. As discussed above that the characters on a license plate are aligned above the baseline. If the baseline found by the voting process is rotated, it stands for that all the character candidates on it are rotated, too. Therefore, the rotation angle of the character candidates can be recovered to normal position according to the detected baseline. During the recovery process, each character candidate is rotated and the related preliminary features such as width, height and occupancy are re-measured for the feature extraction in next stage.

For each recovered single character candidate, the voting boundary method used to find baseline of multiple characters is applied again to find the horizontal boundary lines of each single character candidate. While something different from the former, the conditions for detecting horizontal boundary lines are adjusted for different characteristics of single characters. After the voting boundary process, the true boundary line is selected according to the following two rules: First, the number of votes to set **UNDERFIT** must be zero. It stands for that all the edge pixels must lay inside the boundary lines. Second, instead of referring to the number of votes in set **FIT**, the length of boundary line is referred as the key factor to select true boundary line. The length of a boundary line is defined as the length from the first edge pixel to the last one in set **FIT**. The boundary line candidate of longest length is selected as the true boundary line if its length is longer than a pre-defined threshold. For some characters containing curvature boundaries, the thickness r in (15) can be adjusted to retain accurate results. A typical choice for

32×32 size characters is $r=2$.

Based on the left and right boundary lines, each candidate is adjusted to balance the left and right boundary. Fig. 4-5 shows an example on how to adjust a deformed character based on the detected boundary lines; Fig. 4-5(a) is the source character and Fig. 4-5(b) is the character after adjustment, say, adjusted character. Rectangle ABCD and A'B'C'D' are respectively the rectangular borders of the source character and adjusted character. w and w' are the widths of the characters before and after adjustment. The character height, h , is unchanged after the adjustment. Node1 to node4 are left edge pixels and node5 to node8 are right edge pixels. Node1 and node4 are respectively the start pixel and end pixel of the left boundary line. Node5 and node7 are of the right boundary line. Our target is to arrange the left and right boundary lines symmetrically, i.e., any two pixels having the same y-coordinate on left and right boundary line have the same distance to the outer rectangular left and right borders. Once the deformation is corrected, the characters candidates are then passed to next stage for recognition.

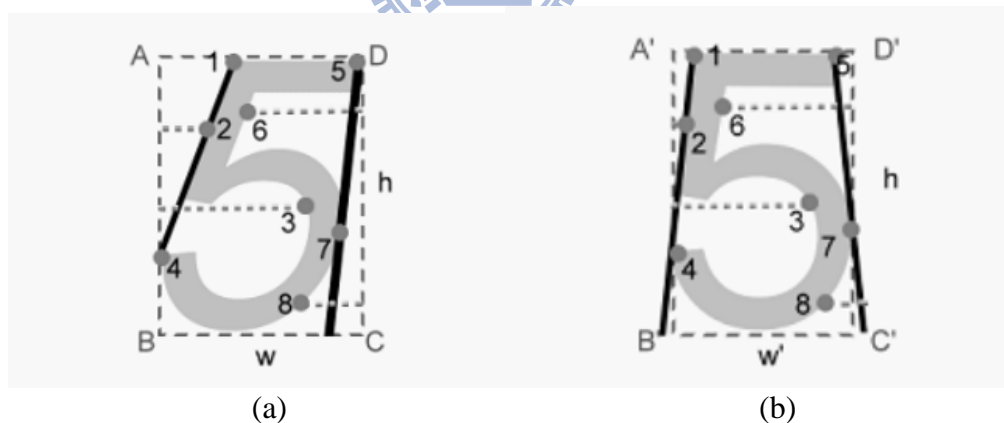


Fig. 4-5 Compensation of geometric deformation

Chapter 5 The Recognition Method

After deformation correction, a novel method named accumulated gradient projection vector method, or AGPV method in short, is applied to recognize the extracted character candidates.

5.1. Why AGPV

When dealing with detection or recognition of characters, edge/line is a basic component that could never be ignored. Straight edges have simple representation and stable characteristic that make them easier detected than any other attributes in an image.

There are numerous methods of edge detection can be found in literatures[16]-[20], among which Hough transform [16] is well-known for its stable and reliable performance. However, Hough transform is also famous for the expensive cost on computation and memory consumption. Although some methods [21][22] are proposed to improve the speed and reduce memory consumption of Hough transform, sometimes it is still insufficient in consideration of accuracy for some applications. In our study, Hough transform provides an important concept to us that stable performance can be achieved by means of accumulation.

In this work we propose a novel accumulated gradient projection method for detection of edges. The new method adopts the same concept as Hough transform to accumulate the pixels of similar attributes in order to achieve stable and reliable result. Besides, two more concepts are included to guarantee the reliability of the method. First, the new method projects the pixels of similar gradient orientations onto an axis which is chosen parallel to the majority of these gradient orientations. In general cases the gradient orientations of edge pixels are perpendicular to the direction of the edge. The projection method achieves the best accuracy of measuring since the edges are measured from their perpendicular direction. Second, instead of referring to pixel intensity which might be sensitive by illumination change, the new method accumulates

the gradient magnitudes which are relatively more stable against illumination change. Besides, the result is also stable against noise because it refers to the majority of accumulation and minimizes the effect of random distributed noise.

5.2. The AGPV Methods

There are four stages to recognize a character using the AGPV method. First, determine the axes; including the nature axes and augmented axes. Second, calculate the AGPVs based on these axes. Third, normalize the AGPVs for comparing with standard ones. Fourth, match with standard AGPVs to validate the recognition result. The procedure will be explained in detail in the following sections.

5.2.1. Determine Axes

When discussing about the AGPV method, it is important to introduce an essential property, axes, in advance. An axis of a character is a specific orientation on which the gradients of grouped pixels are projected and accumulated to form the desired feature vector. An axis is represented by a line that has the specific orientation and passes through the center of gravity point of the grouped pixels. The axes of a character can be separated into two different classes named nature axes and augmented axes. The two classes are different in characteristics and usages and will be described below.

5.2.1.1. Build up Orientation Histogram

The first step of the AGPV method is to build up the corresponding orientation histograms of the character candidates. The orientation histograms are formed from gradient orientations of grouped pixels. Let $\gamma(x,y)$ be the intensity value of sample pixel (x,y) of an image group I , the gradients on x-axis and y-axis are respectively,

$$\begin{aligned} \nabla X(x,y) &= \gamma(x+1,y-1) - \gamma(x-1,y-1) + 2 \times (\gamma(x+1,y) - \gamma(x-1,y)) + \gamma(x+1,y+1) - \gamma(x-1,y+1) \\ \nabla Y(x,y) &= \gamma(x-1,y+1) - \gamma(x-1,y-1) + 2 \times (\gamma(x,y+1) - \gamma(x,y-1)) + \gamma(x+1,y+1) - \gamma(x+1,y-1), \end{aligned} \quad (16)$$

the gradient magnitude, $m(x,y)$, and orientation, $\theta(x,y)$, of this pixel is computed by

$$\begin{aligned} m(x, y) &= \sqrt{(\nabla X(x, y))^2 + (\nabla Y(x, y))^2} \\ \theta(x, y) &= \tan^{-1}(\nabla Y(x, y) / \nabla X(x, y)) \end{aligned} \quad (17)$$

By assigning a number BIN_{his} in the orientation histogram, the gradients are accumulated into BIN_{his} bins and the angle resolution is $RES_{his} = (360/BIN_{his})$. The BIN_{his} is chosen as 64 in the experiments and the angle resolution RES_{his} is therefore 5.625 degrees. Each sample added to the histogram is weighted by its gradient magnitude and accumulated into the two nearest bins by linear interpolation. Besides the histogram accumulation, the gradient of each sample is accumulated into a variable GE_{his} which stands for the total gradient energy of the histogram.

5.2.1.2. Determine the Nature Axes

The nature axes is essential for the AGPV method; the word “nature” is used because the axes always exist “naturally” regardless of most environment and camera factors that degrade the recognition rate. The nature axes have several good properties helpful for the recognition. First, they have high gradient energy on specific orientation and therefore are highly detectable in the input image. Second, the angle differences among the nature axes are invariant to image scaling and rotation. It means, they can be used as references to correct the unknown rotation and scaling factors on the input image. Third, the directions of nature axes are robust within a range of focus and illumination differences. Fourth, although some factors, such as different camera view angle, may cause character deformation and change the angle relationship among the nature axes, the detected nature axes are still useful to filter out the dissimilar ones and narrow down the range of recognition results.

The nature axes are determined by performing peak-valley analysis on the orientation histogram. A peak on the orientation histogram represents a specific orientation in the character candidate. Let function $H(a)$ denote the histogram magnitude appeared on angle a ; the k -th peak,

p_k , of the orientation histogram is located by seeking the angles satisfying

$$H(p_k) > H(p_k - 1) \text{ and } H(p_k) > H(p_k + 1)$$

Beside the center of the peak, the two boundaries named start angle s_k and end angle e_k , within an angle difference to p_k less than a_{th} are found by the following equations,

$$s_k = a, H(a) \leq H(b), \forall b \in (p_k - a_{th}, p_k) \quad (18)$$

$$e_k = a, H(a) \leq H(b), \forall b \in (p_k, p_k + a_{th}) \quad (19)$$

The threshold a_{th} is used to guarantee the boundaries of a peak stay nearby of its center and is defined to be 22.5 degrees in the experiment. The reason to choose ± 22.5 degrees threshold is because it segments a 360-degree circle into 8 orientations; which is similar to human eyes since we often see a circle in 8 octants

Once the start angle and end angle of a peak is determined, an energy function standing for the gradient energy of the k -th peak is defined as $E(k) = \sum_{a=s_k}^{e_k} H(a)$. In addition, an outstanding energy function $D(k)$ is also defined for each peak,

$$D(k) = E(k) - \frac{(H(s_k) + H(e_k)) \times (e_k - s_k)}{2} \quad (20)$$

The outstanding energy neglects the energy contributed by neighboring peaks and is more meaningful than $E(k)$ to represent the distinctiveness of a peak. Peaks with small outstanding energy are not considered as nature axes because that they do not stand out from the neighboring peaks and may not be detectable in new images.

In the experiments, there are different strategies to threshold the outstanding energy for training and recognition. In training phase, we select one perfect image for each character; it is called standard character image and is assigned to be the standard of the recognition. The most

important task in this phase is finding stable peaks in the standard character image. Therefore, a higher threshold $GE_{his}/32$ is applied and a peak has outstanding energy higher than the threshold is considered as a nature axis of the standard character image. In recognition phase, the histogram may have many unexpected factors such as noise, focus error, variable illumination..., so that the task is changed to find one or more matched candidates for further recognition. Therefore, a lower threshold $GE_{his}/64$ is used to filter out the dissimilar ones by the outstanding energy. After threshold check, the peaks whose outstanding energy higher than the threshold is called nature peaks of the character image and the corresponding angles are called the nature axes. Typical license plate character images (alphabet and numerical) can be found having two to six nature axes by the procedures above.

Fig. 5-1 is an example to show the nature axes. Fig. 5-1(a) is the source image, where intensity is ranged from 0(black) to 255(white). Fig. 5-1(b) is the corresponding orientation histogram which are accumulated from the pixels intensity in Fig. 5-1(a). Fig. 5-1(c) overlays the source image with the detected nature axes shown by red arrows. We can see six peaks in the histogram, marked as A,B,C,D,E and F respectively, which correspond to the six red arrows in Fig. 5-1(c).

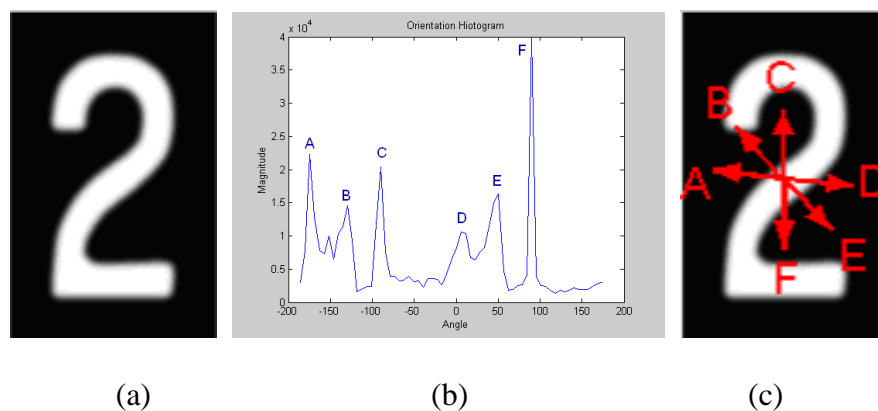


Fig. 5-1 (a) Input image (b) Orientation histogram (c) The nature axes

5.2.1.3. Determine the Augmented Axes

Augmented axes are defined, as augmentations to nature axes, to be the directions on which the generated feature vectors, AGPVs, are unique or special to represent the source character. Unlike the nature axes possessing strong gradient energy on specific orientation, augmented axes do not have this property so that they may not be observed from orientation histogram.

Some characters have only few (one or two) apparent nature axes such as the example in Fig. 5-2. Therefore, it is necessary to generate enough AGPVs on augmented axes for reliable recognition. The experiments tell us that it needs at least four AGPVs in order to recognize a character in a high successful rate. The four AGPVs can be any one from nature axes AGPVs or augmented axes AGPVs. More augmented axes can be declared to refine the recognition result if four AGPVs are not enough to distinguish a character from similar characters. From the experiment results we know that good recognition rate can be achieved for license plate characters by at most six AGPVs.

The augmented axes can be defined by character shapes or by fixed directions. In our experiments, there are only four fixed directions, as the four arrows in Fig. 5-2(c), defined as augmented axes for the total 36 characters. It is not meaningful to declare an augmented axis on a character if it already exists in the nature axes. Therefore, if any one of the four directions already exists in the nature axes, it will not be declared any more in the augmented axes.

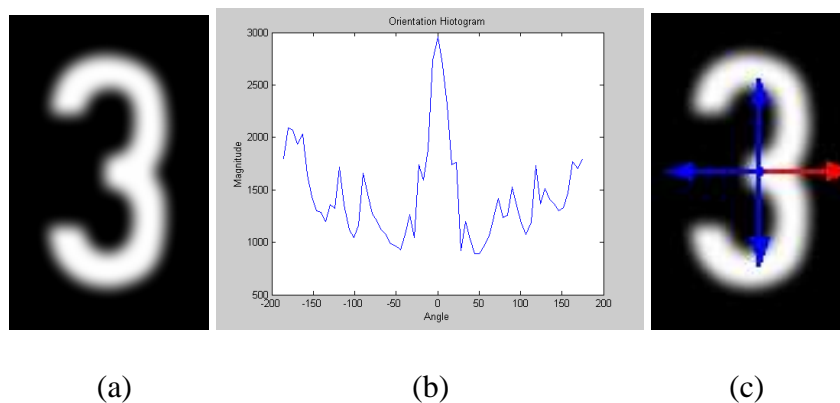


Fig. 5-2 (a) A character that has only one nature axis. (b) Orientation histogram. (c) The nature axes in red arrow and three augmented axes in blue arrows.

5.2.2. Calculate AGPVs

Once the axes of a character are determined, the next step is to calculate the accumulated gradient projection vectors (AGPVs) based on these axes. On each axis of corresponding peak p_k , the gradient magnitudes of pixels whose gradient orientations fall inside the range $s_k < \theta(x, y) < e_k$ are projected and accumulated. The axis could be any one in the nature axes or augmented axes.

5.2.2.1. Projection principles

The projection axis, η_ϕ , is chosen from either nature axes or augmented axes with positive direction ϕ . Fig. 5-3 figures out the projection of sample pixel (x, y) and the center of gravity (COG) point of an object.

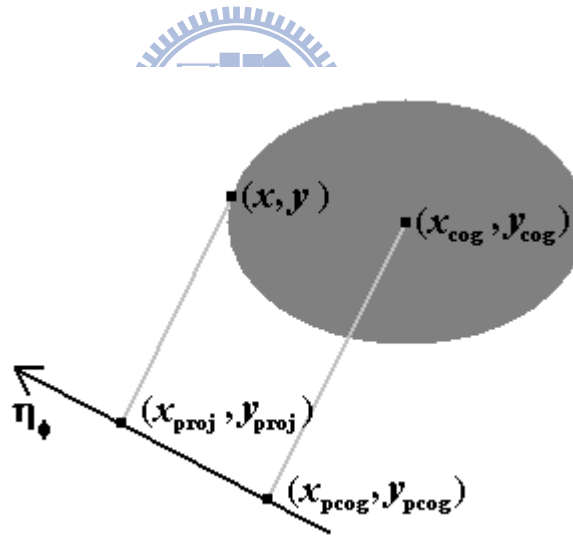


Fig. 5-3 Gradient projection of COG point and any other pixels

Let the (x_{cog}, y_{cog}) be the **COG** point of the input image, i.e.,

$$\begin{bmatrix} x_{cog} \\ y_{cog} \end{bmatrix} = \frac{1}{N} \times \begin{bmatrix} \sum_{i=1}^N (x_i) \\ \sum_{i=1}^N (y_i) \end{bmatrix}, \quad (21)$$

where (x_i, y_i) is the i -th pixel and N is the total number of pixels of a character candidate. Let the function $A(x,y)$ denote the angle between pixel (x,y) and the x-axis, i.e.,

$$A(x, y) = a \tan\left(\frac{y}{x}\right). \quad (22)$$

The process of projecting a character onto axis η_ϕ can be decomposed into three operations. First, rotate the character by angle $\Delta\theta = (A(x_{cog}, y_{cog}) - \phi)$. Second, scale the rotated pixels by a projection factor $\cos(\Delta\theta)$. And third, translate the axis origin to the desired coordinate. Apply the process on the **COG** point, the coordinate of **COG** point after rotation is

$$\begin{bmatrix} x_{rcog} \\ y_{rcog} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \cdot \begin{bmatrix} x_{cog} \\ y_{cog} \end{bmatrix}. \quad (23)$$

Scaling by a projection factor $\cos(\Delta\theta)$, it becomes

$$\begin{bmatrix} x_{pcog} \\ y_{pcog} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & 0 \\ 0 & \cos(\Delta\theta) \end{bmatrix} \cdot \begin{bmatrix} x_{rcog} \\ y_{rcog} \end{bmatrix}. \quad (24)$$

Finally, combine (23) and (24) and further translate the origin of axis η_ϕ to $(x_{\eta ori}, y_{\eta ori})$, the final coordinate (x_{proj}, y_{proj}) of projecting any sample pixel (x,y) onto axis η_ϕ is computed by

$$\begin{bmatrix} x_{proj} \\ y_{proj} \end{bmatrix} = \begin{bmatrix} \cos^2(\Delta\theta) & -\sin(\Delta\theta)\cos(\Delta\theta) \\ \sin(\Delta\theta)\cos(\Delta\theta) & \cos^2(\Delta\theta) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{pcog} \\ y_{pcog} \end{bmatrix} + \begin{bmatrix} x_{\eta ori} \\ y_{\eta ori} \end{bmatrix}. \quad (25)$$

Note that the origin of axis η_ϕ , $(x_{\eta ori}, y_{\eta ori})$, is chosen to be the **COG** point in the experiments, i.e., $(x_{\eta ori}, y_{\eta ori}) = (x_{cog}, y_{cog})$, because it concentrates the projected pixels around the origin (x_{cog}, y_{cog}) and minimizes the axis length to accumulate the projected samples.

5.2.2.2. Gradient projection accumulation

In this section, the pre-computed gradient orientation and magnitude will be projected onto specific axes then summed up. Only sample pixels of similar gradient orientations are projected onto the same axis. As the example in Fig. 5-4, an object O is projected onto axis η of angle 0-degree. In this case, only the sample pixels of gradient orientations $\theta(x,y)$ near 0-degree will be projected onto η and then accumulated.

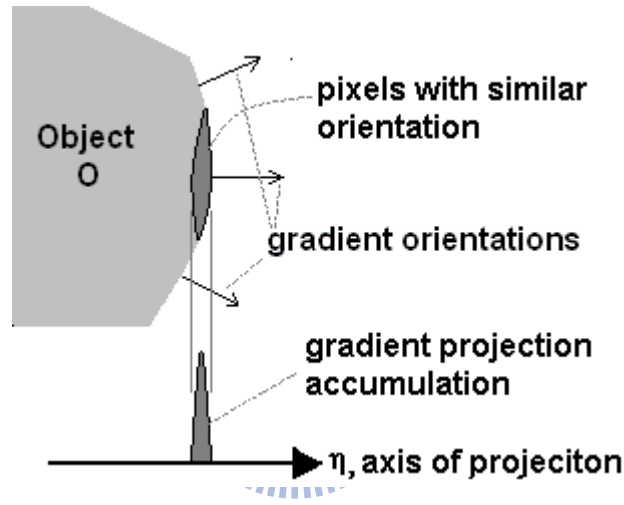


Fig. 5-4 Accumulation of gradient projection

According to axes types, there are two different cases to select sample pixels of similar orientations. For nature axis corresponding to k -th peak p_k , the sample pixels with orientation $\theta(x,y)$ ranged inside the boundaries of the p_k , i.e., $s_k < \theta(x,y) < e_k$, are projected and accumulated. For augmented axis with angle ϕ , the sample pixels with gradient orientations $\theta(x,y)$ ranged by $\theta(x,y) \geq \phi - 22.5$ and $\theta(x,y) \leq \phi + 22.5$ will be projected and accumulated. From (17) and (25), the projected gradient magnitude, $\hat{m}(x,y)$, and the projected distance, $\hat{l}(x,y)$ of sample pixel (x,y) onto axis η_ϕ are respectively

$$\hat{m}(x,y) = m(x,y) \times \cos(\theta(x,y) - \phi), \quad (26)$$

and

$$\hat{\ell}(x, y) = \sqrt{(x_{proj} - x_{pcog})^2 + (y_{proj} - y_{pcog})^2}. \quad (27)$$

To accumulate the gradient projections, an empty array $R(x)$ is created with length equals to the diagonal of the input image. Since the indexes of an array must be integers, linear interpolation is used to accumulate the gradient projections into the two nearest indexes of the array. In mathematical representations, let $b = \text{floor}(\hat{\ell}(x, y))$ and $u = b + 1$, where $\text{floor}(z)$ rounds z to the nearest integers towards minus infinity. For each sample pixel (x, y) on input image I , do the following accumulations,

$$R(b) = R(b) + \hat{m}(x, y) \times (u - \hat{\ell}(x, y)); \quad R(u) = R(u) + \hat{m}(x, y) \times (\hat{\ell}(x, y) - b). \quad (28)$$

Besides $R(x)$, a second array, $T(x)$, is also created to collect overall information required for normalization. There are two differences between $R(x)$ and $T(x)$. First, unlike $R(x)$ targeting on only the sample pixels of similar orientation, $T(x)$ targets on all the sample pixels of a character and accumulates their gradient magnitudes. Second, $R(x)$ accumulates the projected gradient magnitude $\hat{m}(x, y)$, while $T(x)$ accumulates the original gradient magnitude $m(x, y)$. Referring to eq.(28),

$$T(b) = T(b) + m(x, y) \times (u - \hat{\ell}(x, y)); \quad T(u) = T(u) + m(x, y) \times (\hat{\ell}(x, y) - b). \quad (29)$$

The purpose of $T(x)$ is to collect the overall gradient information of the interested character candidate for normalizing array $R(x)$ into desired AGPV.

5.2.2.3. Normalization

The last step to find out the AGPV of an axis is to normalize the gradient projection

accumulation array $R(x)$ into a fixed-length vector. With the fixed length, the AGPVs have standard dimensionality and can be compared with standard AGPVs easily. Before the normalization, the length of AGPV, L_{AGPV} , has to be determined. Depends on the complexity of recognition targets, different length of AGPV may be selected to describe the distribution of projected gradients. In our experiments, the L_{AGPV} is chosen as 32. A smaller L_{AGPV} lowers the resolution and degrades the recognition rate while a larger L_{AGPV} slows down system performance and makes no significant difference on recognition rate. It is worth to note that, one AGPV formed upon an axis is independent from the other AGPVs formed upon different axes. This is important to make the AGPVs independent from one another regardless of the source character and axes.

In order to avoid the impact of isolated sample pixels which are mostly caused by noise, the array $R(x)$ is filtered by a Gaussian filter $G(x)$:

$$\tilde{R}(x) = R(x) * G(x), \quad (30)$$



where the operator $*$ stands for convolution operation. The variance of the $G(x)$ is chosen as $\sigma = (Len_R)/128$ in the experiments, where Len_R is the length of $R(x)$. It is found that this choice benefits in both resolution and noise rejection. Similarly, the array $T(x)$ is also filtered by the same Gaussian filter to eliminate the effect of noise. After Gaussian filtering, the array $T(x)$ is analyzed to find effective range, the range in which the data is effective to represent a character. The effective range starts from index X_s and ends in index X_e , defined as

$$X_s = \{x_s, T(x_s) \geq th_T; T(x) < th_T, \forall x < x_s\}, \quad (31)$$

and

$$X_e = \{x_e, T(x_e) \geq th_T; T(x) < th_T, \forall x > x_e\}, \quad (32)$$

where the threshold th_T is used to discard noise and is chosen as $th_T = \text{Max}(T(x))/32$ in the experiment. The effective range of $R(x)$ is assigned to be the same as the effective range of $T(x)$, from X_s to X_e .

As mentioned previously, the gradient projection accumulation results in a large sum along a straight edge. This is a good property if the interested character is composed of straight edges. However, some characters may consist of not only straight edges but also some curves and corners which only contribute small energy on array $R(x)$. In order to balance the contribution of different types of edges and avoid the disturbance from noise, a threshold th_R is used to adjust the content of array $R(x)$ before normalization,

$$\hat{R}(x) = \begin{cases} 0, & \text{if } \tilde{R}(x) < th_R \\ 255, & \text{if } \tilde{R}(x) \geq th_R \end{cases}, \quad (33)$$

After finding the effective range and adjusting the content of array $R(x)$, the accumulated gradient projection vector (AGPV) is defined to resample from $\hat{R}(x)$,

$$AGPV(i) = \hat{R} \left(\text{round} \left(\left(\frac{i}{32} \right) \times (X_e - X_s) + X_s \right) \right). \quad (34)$$

Fig. 5-5 gives an example of the gradient accumulation array $T(x)$, gradient projection accumulation array $R(x)$ and normalized AGPV. The example uses the same test image as Fig. 5-1 and displays only one of the nature axes, axis E. Similar to the method of finding the peaks of orientation histogram, the k -th effective peaks, ep_k , on $R(x)$ is defined as $R(ep_k) > R(ep_k - 1)$ and $R(ep_k) > R(ep_k + 1)$. It can be observed that four effective peaks exist in Fig. 5-5(c) and each of them represents an edge projected onto axis E in Fig. 5-1(c).

5.2.3. Matching and Recognition

This section describes how to apply the AGPV method to recognize the extracted character

candidates, or say, test characters, in three aspects. First, the standard AGPV database is collected to be the standard templates for matching with test characters. Second, three properties used in the matching process are discussed. Third, the methods to match the AGPV of a test character with standard AGPVs.

5.2.3.1. Create standard AGPV database

A standard database is created by collecting all the AGPVs extracted from characters of standard parameters: standard size, standard aspect ratio, no noise, no blur, and neither rotation nor deformation. The extracted AGPVs are called standard AGPVs and stored by two categories: the one calculated on nature axes is called the standard nature AGPVs and the other calculated on augmented axes is called the standard augmented AGPVs. Let the number of total standard characters be N , $N=36$ (0~9 and A~Z) for license plate characters in this paper. Denote the number of standard nature AGPVs for i -th standard character as $NN(i)$, the number of standard augmented AGPVs as $NA(i)$, and the total number of AGPVs as $NV(i)$, where $NV(i)=NN(i)+NA(i)$. The j -th standard AGPV of the i -th character is denoted as $V_S(i,j)$, where $j=1$ to $NV(i)$. Note that $V_S(i,j)$ are standard nature AGPVs for $j \leq NN(i)$ while $V_S(i,j)$ are standard augmented AGPVs otherwise.

5.2.3.2. Properties used for matching

Unlike general vectors matching problem directly referring to the RMS error of two vectors, the matching of AGPVs refers to special properties which are derived from their physical meanings. There are three properties useful for similarity measuring between two AGPVs.

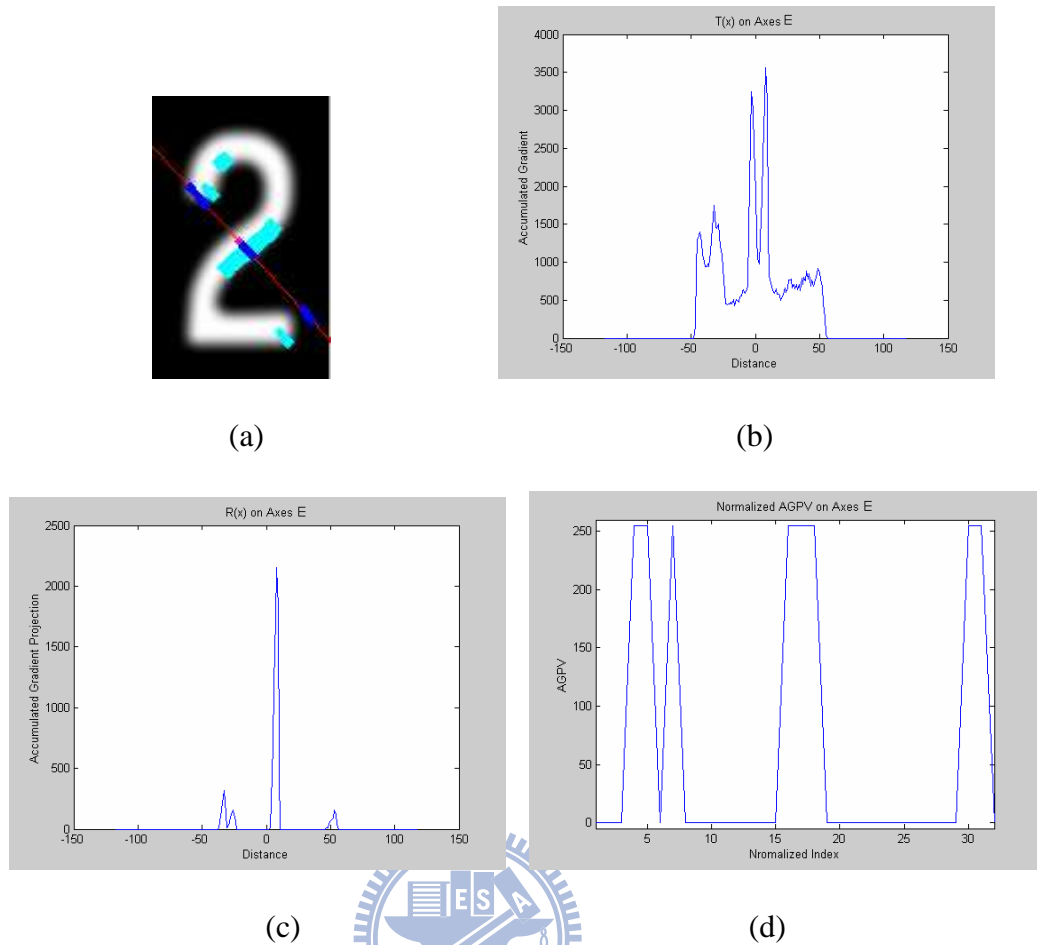


Fig. 5-5 (a) Gradient projection on axis D. (pink: COG point; red: axis D; cyan: selected sample pixels; blue: projected samples) (b) The gradient accumulation array $T(x)$ with distance to the COG point. (c) The gradient projection array $R(x)$. (d) Normalized AGPV.

The first property used for similarity measuring between two AGPVs is that each peak in an AGPV represents an edge on the source character. The number of peaks, or say the edge count, is useful to represent the difference between two AGPVs. For example, there are four peaks on the extracted AGPV in Fig. 5-5(d) and each of them represents an edge on the axis. The edge count is invariant no matter how the character exists in the input image. In this paper, a function $EC(V)$ is defined to calculate the edge count of an AGPV V by the following algorithm,

===== Algorithm 1: Count the number of edges in an AGPV =====

$ec=0$;

```

for  $i=1$  to (size( $V$ )-1)
    if(  $V(i)==0$  and  $V(i+1) >0$ )
         $ec=ec+1$ ;
    end
end
 $EC(V)=ec$ ;
===== end of formula 1 =====

```

The second property used for similarity measuring between two AGPVs is that although the edge count in an AGPV is invariant for the same character, the position of the edges could be varied if the character is deformed. This is the major reason to explain why the RMS error is not suitable to measure the similarity between two AGPVs. In order to compare AGPVs under the cases of character deformation, a matching cost function $C(U, V)$ is calculated to measure the similarity between AGPV U and AGPV V , expressed as,

$$C(U, V) = |EC(U) - EC(V)| + |EC(UV) - EC(V)| + |EC(IV) - EC(V)|, \quad (35)$$

where $UV = U \cup V$ is the union vector of AGPV U and AGPV V while $IV = U \cap V$ is the intersection vector of them. UV and IV are calculated by the following formulas:

```

===== Formula 2: calculate union vectors of two AGPVs=====
for  $i=1$  to 32
    if( $V(i)>0$  or  $U(i) >0$ )
         $UV(i)=1$ ;
    else
         $UV(i)=0$ ;
    end
end
===== end of formula 2=====

```

==== Formula 3: calculate intersection vectors of two AGPVs =====

for $i=1$ to 32

if($V(i)>0$ and $U(i) >0$)

$IV(i)=1;$

else

$IV(i)=0;$

end

end

===== end of formula 3=====

The third property used for similarity measuring between two AGPVs is that the angular relationships of nature axes on the test character are similar to those on the corresponding standard character. In the experiment, a threshold $th_A = \pi/32$ is used to check if the AGPVs of the test character match the angular relationship of nature axes of a standard character. Let $AA_T(k)$ be the k -th axis angle of the test character, the function $AA(i,j)$ denote the angle of the j -th axis of the i -th standard character, $0 \leq AA(i,j) < 2\pi$, for $i=1$ to 36, $j=1$ to $NV(i)$. If the m -th and n -th axis of the test character are respectively corresponding to the g -th and h -th axis of the i -th standard character, then

$$|(AA_T(m) - AA_T(n)) - (AA(i,g) - AA(i,h))| \leq th_A. \quad (36)$$

A typical example can be seen by comparing Fig. 5-5 and Fig. 5-6 that the characters in Fig. 5-5(a) and Fig. 5-6(a) are the same but differs in blur index. Let the extracted AGPV in Fig. 5-5(d) be U and the one in Fig. 5-6(c) be V . From algorithm 1, the edge count of the associated vectors are respectively, $EC(U)=4$, $EC(V)=3$, $EC(UV)=3$, $EC(IV)=4$. From the definition of matching cost in (35),

$$C(U,V) = |EC(U) - EC(V)| + |EC(UV) - EC(V)| + |EC(IV) - EC(V)|$$

$$= |4 - 3| + |3 - 3| + |4 - 3| = 2$$

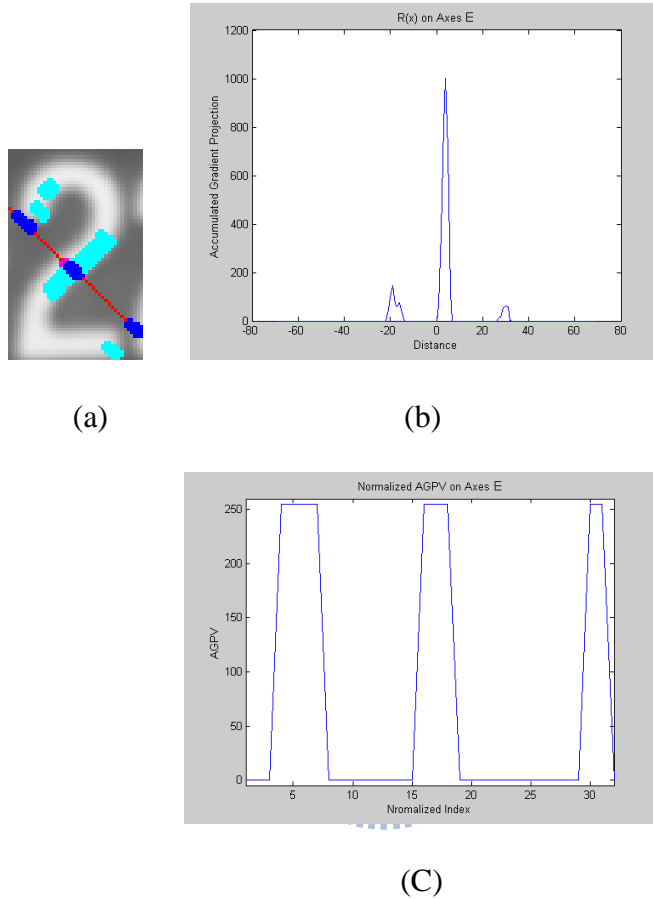


Fig. 5-6 An example comprising different blur index with Fig. 5-5 (a) Gradient projection on axis D. (pink: COG point; red: axis D; cyan: selected sample pixels; blue: projected samples) (b) The gradient projection array $R(x)$. (c) Normalized AGPV.

5.2.3.3. Matching of characters

In order to recognize the test character, the AGPVs of the test character is stage-by-stage compared with the standard AGPVs in the database. Moreover, a candidates list is created by including all the standard characters at the beginning and remove the standard characters those have high matching cost to the test character on each stage. Until the end of the last stage, the

candidate in the list consisting of the lowest total matching cost is considered as the recognition result.

Stage 1: Find the fundamental matching pair. Calculate the cost function between the test character and the j -th AGPV of the i -th standard character.

$$C_1(k, j) = C(\mathbf{V}_T(k), \mathbf{V}_S(i, j)) \quad (37)$$

Find a pair of axes whose matching cost is the minimum. Let k_T and j_S be the pair of axes respectively on the test character and i -th standard character

$$pair(k_T, j_S) = \underset{k, j}{arg \min}(C_1(k, j)); \quad (38)$$

If $C(k_T, j_S)$ is less than a threshold th_F , the i -th standard character is kept in the candidates list and the pair (k_T, j_S) is served as the fundamental pair of the candidate.

Stage 2: Find the other matching pairs between the standard AGPVs and the test character: Based on the fundamental pair, the axes angles of the test character are compared with those of the standard character. Let the number of nature AGPVs detected on the test character be NN_T . For the i -th standard character, create an empty array $mp(j)=0$, $1 \leq j \leq NV(i)$, to denote the matching pair with the test character. Taking use of eq(36), calculate

$$\begin{aligned} |(AA_T(k) - AA_T(k_T)) - (AA(i, j) - AA(i, j_S))| \leq th_A \quad ; \quad \forall k \in [1, NN_T], k \neq k_T \quad ; \\ \forall j \in [1, NN(i)], j \neq j_S \end{aligned} \quad (39)$$

the k -th test AGPV satisfies (39) is called the j -th matching pair of the standard character, denoted as $mp(j)=k$. Note that there might be more than one test AGPVs satisfying (39). In this case only the one of lowest matching cost is recognized as the j -th matching axis and the others are ignored.

Stage 3: Calculate total matching cost of standard nature AGPVs: Define a character

matching cost function $CMC(i)$ to measure the similarity between test character and the i -th standard character by summing up the matching costs of all the matching pairs,

$$CMC(i) = \sum_{j=1, mp(j)>0}^{NN(i)} MC(\mathbf{V}_T(mp(j)), \mathbf{V}_S(i, j)) \quad (40)$$

Stage 4: Calculate the matching costs of augmented AGPVs: At the first step, find the axis angle AX on the test character corresponding to the j -th standard augmented axis as

$$AX = (AA(i, j) - AA(i, j_s)) + AA_T(k_T) \quad (41)$$

If there is one AGPV of the test character, say, the k -th nature AGPV satisfying (39), i.e., $|(AA_T(k) - AX)| \leq th_A$, then the k -th nature AGPV is mapped to the j -th augmented axis and $mp(j)=k$. Otherwise, the AGPV corresponding to the j -th standard augmented axis must be calculated based on the axis angle AX . After that, the matching costs of the augmented AGPVs are accumulated into the character matching cost function as,

$$CMC(i) = \sum_{j=NN(i)+1}^{NV(i)} MC(\mathbf{V}_T(mp(j)), \mathbf{V}_S(i, j)) \quad (42)$$

Stage 5: Recognition: Due to the different number of AGPVs for different standard character, the character matching cost function is normalized by the total number of standard AGPVs, i.e.,

$$CMC(i) = CMC(i) / NV(i) \quad (43)$$

Finally, the test character is recognized as the h -th standard character of the lowest matching cost if the character matching cost $CMC(h) < th_R$.

Chapter 6 Experimental Results

The experiments are designed in two aspects to respectively test the feasibility and performance of the two novel methods proposed in this work. The first aspect is focused on the extraction function, where the proposed scale space binarization method is compared with two popular binarization methods on several properties. The second aspect is the recognition function, where the proposed AGPV method is compared with traditional method, too, to show the performance.

6.1. Scale-Space Binarization Method(Extraction) Test

The experiments to test the scale-space binarization method are divided into three parts. The first one is feasibility test which is held in order to prove the feasibility of proposed method under various environments and light conditions. Two well-known binarization methods, Otsu's method [1] (global thresholding) and the local intensity gradient method (LIG) [5], (local thresholding) are compared with the proposed method.

The second experiment is reliability test. We respectively add different levels of noise and illumination into the test images and re-measure the extraction result. Similarly, the results of using Otsu's method and the LIG method are compared as well.

The final experiment on the binarization method is the computation time test. We record the computation time for the three different methods on a Pentium-M machine running 1.5GHz and compare their performance.

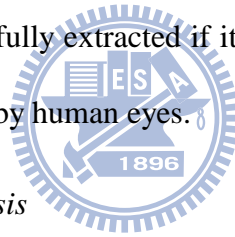
6.1.1. Feasibility Test

The test images are captured from various environments containing license plates captured from different orientation and distances. Totally 54 images are converted into 8-bit gray-scale images and resampled to 640×480 pixels. Two sample images and their simulation results are

shown in Fig. 6-1 and Fig. 6-2. Two popular gray-level image binarization methods, Otsu's method and LIG, are compared with the proposed method. From the simulation results, we can see that the proposed SSB method perform better binarization results than the two prior methods. It is worth to note that, the Otsu's method is convenient in implementation but often failed in the images containing complex background; the LIG method performs nicely around edges but failed to identify the interior of characters.

6.1.2. Reliability Test

Different levels of noise and illumination are added into the test images and the true positive rate (TPR_E), i.e., the rate that the true characters are extracted successfully in the test image, is measured. The subscript E is for extraction, used for distinguishing from the true positive rate of recognition TPR_R where the true characters are recognized successfully in the test image. A character is considered as successfully extracted if it is isolated from external objects and the grouped pixels can be recognized by human eyes.



6.1.2.1. Quantization noise analysis

The first factor affects TPR_E is quantization noise. We respectively add 6 levels (0.8%, 1.6%, 3.2%, 6.4%, 12.8%, 25.6%) noise into each pixel, where the 0.8% noise level is equivalent to add 1 or -1 randomly into each 8-bit gray-level pixel; 1.6 % is equivalent to add 2 or -2 into each pixel... 25.6% is equivalent to add 32 or -32 into each pixel. The images after adding 12.8 % and 25.6% noise are shown in Fig. 6-3. The simulation results are shown in Table I, where the TPR_E is ranged from 0 to 100; $TPR_E=50$ represents half(27) license plates on the 54 images are successfully extracted.

From the simulation result in Table I, it is obvious that the proposed method (scale-space binarization, SSB) performs better than the other two methods when the input image is corrupted with noise. In addition, a conclusion can be derived from the simulation that the TPR_E

of binarization is closely related with characters sizes. The bigger the character size is, the higher the TPR_E is.

TABLE I
 TPR_E BY QUANTIZATION NOISE ANALYSIS

Noise level	0.8%	1.6%	3.2%	6.4%	12.8%	25.6%
SSB	94	94	94	92	90	82
Otsu's	83	81	75	62	53	32
LIG	89	87	85	78	63	47

6.1.2.2. Illumination analysis

Illumination is another important factor to binarization of images. In order to test the robustness of the proposed method to illumination change, four directional light sources L_1 to L_4 are added in the test images to imitate the responses under different illumination. The gray-level intensity of the three test images are multiplied by the following four directional light sources.

$$\begin{aligned}
 L_1(x, y, k) &= ((x + y + 10) / (H + W + 10))^k \\
 L_2(x, y, k) &= (((W - x) + y + 10) / (H + W + 10))^k \\
 L_3(x, y, k) &= ((x + (H - y) + 10) / (H + W + 10))^k \\
 L_4(x, y, k) &= (((W - x) + (H - y) + 10) / (H + W + 10))^k
 \end{aligned} \tag{44}$$

where the W and H are respectively the width and height of the test image and k is the decay curve of the directional light sources. Fig. 6-4 shows the number one image exposed under $L_1(x, y, k)$ with 3 different decay curves $k=1, 2, 4$. The simulation is executed by changing decay curve k and considered as successful if the characters exposed under the four directional lights can be extracted and recognized by human eyes.

TABLE II.
 TPR_E BY ILLUMINATION ANALYSIS

Illumination decay curve	$K=1$	$K=2$	$K=3$	$K=4$
SSB	93	93	86	74
Otsu's	32	5	0	0
LIG	89	87	54	38

It is evident in Table II that local thresholding methods(SSB, LIG) are better than global one(Otsu's), and among them the SSB method is better than LIG under different illumination.

6.1.3. Computation time test

The three binarization methods are implemented by C-language and executed in a Pentium-M machine running 1.5GHz. The computation times required for binarizing the test images are measured and the averages of them and the corresponding frame rates are recorded in TABLE III. Note that the computation time measured doesn't include the connected component analysis because the comparison is focused on binarization only.

TABLE III
COMPUTATION TIME COMPARISON OF THE THREE METHODS

	Averaged time	Averaged frame rate
SSB	62ms	16.13
Otsu's	33ms	30.0
LIG	89ms	11.2



From table III, we can see the SSB performs faster than the LIG (local thresholding) while still slower than the Otsu's method (global thresholding). Actually many binarization methods require pre-processings such as smoothing filtering in order to ensure the noise are minimized before binarization. We can see this from the earlier results of quantization noise analysis. The SSB method here already incorporates two stages pre-smoothing by Gaussian filter. Therefore, the time differences required for SSB and Otsu's method may be smaller than that in table III when the preprocessings are taken into account.



Fig. 6-1 (a) Source image#1. (b) Converted binary image; blue rectangles are isolated groups (c) Results after elimination, blue rectangles are character candidates. (d) Binarization result using Otsu's method. (e) Binarization result by LIG.

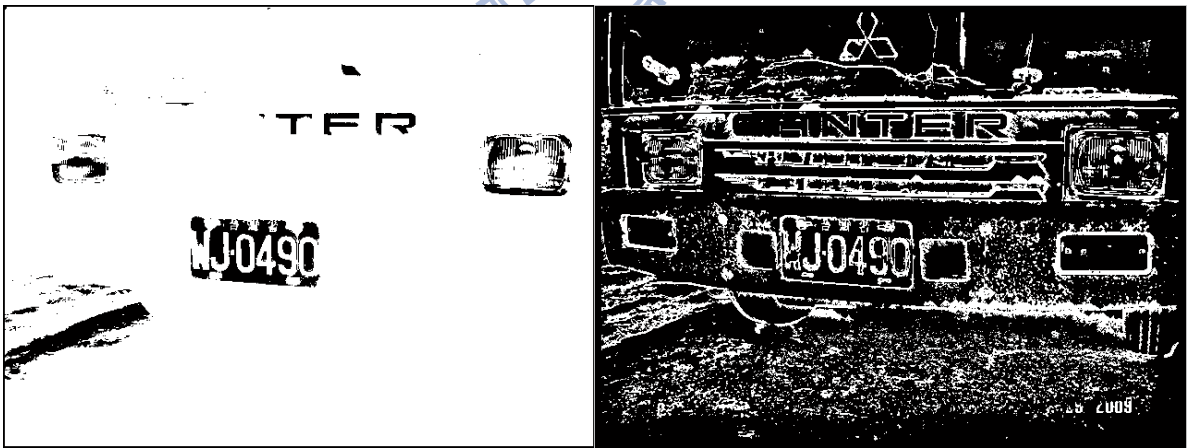
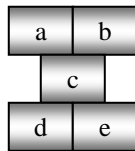


Fig. 6-2 (a) Source image#2. (b) Converted binary image; blue rectangles are isolated groups. (c) Results after elimination, blue rectangles are character candidates. (d) Binarization result using Otsu's method. (e) Binarization result by LIG.



Fig. 6-3 Images used in noise analysis(left: 12.8%, right:25.6% quantization noise)



a	b
c	d

Fig. 6-4 Images with different decay curve light sources used in illumination analysis, (a) $k=1$, (b) $k=2$, (c) $k=3$ (d) $k=4$

6.2. AGPV(Recognition) Test

Although the AGPV method is robust due to its accumulation property, it becomes a limitation that the size of the characters must be big enough for stable recognition. Therefore, the characters smaller 64×64 after the extraction are up-sampled twice respectively on horizontal and vertical axes by interpolation. After that, we choose some characters from the test images to be the standard characters and calculate the standard AGPVs.

Two factors, noise and illumination as that for extraction test is again used to test the reliability of the AGPV method. The results are measured by the true positive rate on recognition (TPR_R). Total 264 characters extracted from the extraction stage are treated as the test characters in this measuring. $TPR_R=50$ represents that 132 characters out of the 264 ones are recognized successfully. The simulation result is listed in Table IV - Table V.



TABLE IV

TPR_R BY QUANTIZATION NOISE ANALYSIS

Noise level	0	0.8%	1.6%	3.2%	6.4%	12.8%	25.6%
TPR_R	93	92	90	86	81	73	61

TABLE V

TPR_R BY ILLUMINATION ANALYSIS

Illumination decay curve	$K=0$	$K=1$	$K=2$	$K=3$	$K=4$
TPR_R	93	92	87	82	65

Chapter 7 Conclusion and Future Work

7.1. Conclusion

This dissertation is devoted to present an approach comprising three novel methods for recognition of license plate characters. The technologies related to the license plate recognition are first reviewed in Chapter 2 and then, the three methods respectively in charge of extraction, normalization and recognition of license plate characters are discussed in Chapter 3 to Chapter 5. After that, the experimental results are shown in Chapter 6 to demonstrate the feasibility and the performance of the presented approach.

The first method named scale space binarization method (SSB) is used in the extraction stage, intending to extract the characters quickly and reliably in the source image. The method utilizes Difference-of-Gaussian function to localize the profiles of the interested characters and dynamic thresholding to binarize the license plates. Then the characters are extracted from the binary image by connected component analysis and the false candidates are eliminated from both geometrical properties and profile scores. Optimization methods are also disclosed for implementation and experimental results are provided to show the robustness and performance of the proposed method in comparison with the two most-used methods, Otsu's method and LIG method. Compared with these methods, the SSB method is obviously robust from noise and illumination change.

The second method, voting boundary method, is used for correcting the geometric deformation of characters which often acts as the major reason for recognition rate degradation in license plate recognition systems. The voting boundary method is helpful to estimate the boundary lines used for correcting the deformation of characters.

The third method, AGPV method, is designed in the recognition stage to recognize isolated

characters on license plates. The feature vectors AGPVs calculated from Gaussian-filtered images are independent from rotation and scaling and suitable for characters recognition. The experimental results demonstrate the success of the proposed method and its robustness to noise and illumination change.

7.2. Future work

Although the methods in this work already include complete functions to recognize license plates from gray-level images, there still exist some issues worthy of future studies.

First, although the SSB method can be designed to do full scale search, the computation for a full scale search is still too heavy by pure software for real-time applications. A hardware accelerator can be studied to speed up the extraction process. Besides, the SSB method may not be able to extract character candidates accurately if the resolution of the image is low. Some methods to improve the successful extraction rate in low resolution must be researched.

Second, although the voting boundary method is helpful to correct characters from deformation, it is sometimes inaccurate to correct characters of non-linear edges like “S” or “D” or “Q” from certain deformations due to their curvature edges. Besides, it cannot work properly if the license plate includes dirty smudges around characters or the resolution is low.

Third, the AGPV method requires manual decisions such as selecting standard characters by human eyes. It is better to be improved by some systematic procedures to do automatically training from various input images. In addition, currently the AGPV method may degrade recognition rate seriously if the geometric deformation of characters is not fully corrected in the normalization stage. How to improve the recognition rate when the test characters undergo certain degree of geometric deformation is also an important topic for future studies. Moreover, the computational complexity of the AGPV method is heavy and still needs improvement in the future.

Bibliographies

- [1] Takashi Naito, Toshihiko Tsukada, Keiichi Yamada, Kazuhiro Kozuka, and Shin Yamamoto, "Robust License-Plate Recognition Method for Passing Vehicles under Outside Environment," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 6, 2000.
- [2] S. Kim, D. Kim, Y. Ryu, and G. Kim, "A Robust License Plate Extraction Method Under Complex Image Conditions," in Proc. 16th International Conference on Pattern Recognition (ICPR'02), Quebec City, Canada, vol. 3, pp. 216-219, Aug. 2002.
- [3] S. Z. Wang and H. J. Lee, "A Cascade Framework for a Real-Time Statistical Plate Recognition System," *Transactions on Information Forensics and Security, IEEE*, vol. 2, no.2, pp. 267 - 282, DOI: 10.1109/TIFS.2007.897251, June 2007.
- [4] Zunino, R. and Rovetta, S., "Vector Quantization for License-Plate Location and Image Coding." *Transactions on Industrial Electronics, IEEE*, vol. 47, no. 1, pp. 159 - 167, Feb 2000, DOI: 10.1109/41.824138.
- [5] Kwan, H.K., "Multilayer Recurrent Neural Networks [Character Recognition Application Example]." The 2002 45th Midwest Symposium on, vol. 3, pp. 97-100, 4-7 Aug. 2002, ISBN: 0-7803-7523-8, INSPEC: 7736581.
- [6] Wu-Jun Li, Chong-Jun Wang, Dian-Xiang Xu, and Shi-Fu Chen., "Illumination Invariant Face Recognition Based on Neural Network Ensemble." *ICTAI*, pp. 486 - 490, 15-17 Nov. 2004, DOI: 10.1109/ICTAI.2004.71
- [7] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms." *Transactions on Systems, Man and Cybernetics, IEEE*, vol. 9, no. 1, pp. 62-66, Jan. 1979, ISSN: 0018-9472, DOI: 10.1109/TSMC.1979.4310076
- [8] Atallah AL-Shatnawi and Khairuddin Omar., "Methods of Arabic Language Baseline Detection – The State of Art," *IJCSNS*, vol. 8, no. 10, Oct 2008.

- [9] D. G. Lowe, "Distinctive Image Features from Scale-invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.
- [10] D. G. Lowe, "Object Recognition from Local Scale-invariant Features," International Conference on Computer Vision, Corfu, Greece (September 1999), pp. 1150-1157.
- [11] Witkin, A. P., "Scale-space Filtering," International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, pp. 1019-1022, 1983.
- [12] Koenderink, J. J., "The Structure of Images," *Biological Cybernetics*, 50:363-396, 1984.
- [13] Lindeberg, T. "Scale Space Theory: A Basic Tool for Analyzing Structures at Different Scales." *Journal of Applied Statistics*, vol. 21, no. 2, pp. 224-270, 1994.
- [14] Mikolajczyk, K. "Detection of Local Features Invariant to Affine Transformations." Ph.D thesis, Institut National Polytechnique de Grenoble, France, 2002.
- [15] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *J. Data Mining Knowl. Disc.*, vol. 2, no. 2, pp.121-167, 1998.
- [16] Lixin Fan, "What A Single Template Can Do in Recognition," Fourth International Conference on Image and Graphics, pp. 586-591, 2007.
- [17] Smith, Julius O. Spectral Audio Signal Processing, October 2008 Draft, <http://ccrma.stanford.edu/~jos/sasp/>, online book, accessed <20100528>.
- [18] M. Irani and S. Peleg, "Motion Analysis for Image Enhancement: Resolution, Occlusion and Transparency," *Journal of Visual Communications and Image Representation*, Dec. 1993, vol. 4, pp. 324-335,
- [19] C. N. Anagnostopoulos, et al., "A license plate recognition algorithm for Intelligent Transportation System applications, " *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 377-392, 2006

- [20] J. N. Kapur, P. K. Sahoo and A. K. C. Wong, "A New Method for Gray-level Picture Thresholding Using the Entropy of the Histogram," *Computer Vision, Graphics, and Image Processing*, vol. 23, no. 3, pp. 273-285., 1985.
- [21] S. U. Lee, S. Y. Chung, and R. H. Park, "A Comparative Performance Study of Several Global Thresholding Techniques Segmentation," *Computer Vision, Graphics, and Image Processing*, vol. 52, no. 2, pp. 171-190, 1990.
- [22] P. K. Sahoo, S. Soltani and A.K.C. Wong, "A Survey of Thresholding Technique," *Computer Vision, Graphics, and Image Processing*, vol. 41, no. 2, 1988.
- [23] J. R. Parker, "Gray Level Thresholding in Badly Illuminated Images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 8, pp. 813-819, 1991.
- [24] F. Deravi and S. K. Pal, "Gray Level Thresholding Using Second Order Statistics," *Pattern Recognition Lett.*, vol. 1, no. 5-6, pp. 417-422, 1983.
- [25] J. Kittler and J. Illingworth, "Threshold Selection Based on a Simple Image Statistic," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 2, pp. 125-147, 1985.
- [26] Y. Yang and H. Yan, "An Adaptive Logical Method for Binarization of Degraded Document Images," *Pattern Recognition*, vol. 33, pp. 787-807, 2000.
- [27] Y. P. Chen and T. D. Yeh, "A Method for Extraction and Recognition of Isolated License Plate Characters," *IJCSIS*, vol. 5, no. 1, pp. 1-10, 2009.
- [28] Sauvola J. and Pietikainen M., "Adaptive Document Image Binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225-236, 2000.
- [29] Farrahi Moghaddam, R. and Cheriet, M., "A Multi-Scale Framework for Adaptive Binarization of Degraded Document Images," *Pattern Recognition*, vol. 43, no. 6, 2010.
- [30] R. Haralick, "Image segmentation survey," in *Fundamentals of Computer Vision*, O. D. Faugeras, Ed. London: Cambridge University Press, 1983.

- [31] T. G. Stockham, "Image processing in the context of a visual model," *Proc. IEEE*, vol. 60, no. 7, pp. 828-842, 1972.
- [32] Witkin, A. P., "Scale-space filtering," International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, pp. 1019-1022, 1983.
- [33] Koenderink, J. J. "The Structure of Images," *Biological Cybernetics*, vol. 50, pp. 363-396, 1984.
- [34] Lindeberg, T. "Scale Space Theory: A Basic Tool for Analyzing Structures at Different Scales." *Journal of Applied Statistics*, vol. 21, no. 2, pp. 224-270, 1984.
- [35] Mikolajczyk, K. "Detection of Local Features Invariant to Affine Transformations." Ph.D thesis, Institut National Polytechnique de Grenoble, France, 2002.
- [36] H. J. Lee and B. Chen, "Recognition of Handwritten Chinese Characters via Short Line Segments," *Pattern Recognition*, vol. 25, no. 5, pp. 543-552, 1992
- [37] M. Irani and S. Peleg, "Motion Analysis for Image Enhancement: Resolution, Occlusion and Transparency," *Journal of Visual Communications and Image Representation*, vol. 4, pp. 324-335, 1993.
- [38] Marek Brej and Milan Sonka, "Object Localization and Border Detection Criteria. Design in Edge-Based Image Segmentation: Automated Learning from Examples," *IEEE Transactions on Medical Imaging*, vol. 19, no. 10, Oct. 2000.
- [39] Y. P. Chen and T. D. Yeh, "Isolated Characters Extraction Using Difference-of-Gaussian Function," National Computer Symposium: Workshop on ICM, pp. 274-282, Taipei Taiwan, Nov. 27-28, 2009
- [40] Gonzalez and R. Woods, Digital Image Processing, Addison Wesley, pp. 414 - 428, 1992.
- [41] C. Harris and M. Stephens. "A Combined Corner and Edge Detector," Proceedings of the 4th Alvey Vision Conference, pp. 147-151, 1988.

- [42] John Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, Nov. 1986, doi:10.1109/TPAMI.1986.4767851.
- [43] S. Nomura, et al., "A Novel Adaptive Morphological Approach for Degraded Character Image Segmentation," *J. Pattern Recognit.*, vol. 38, pp. 1961-1975, Jan. 2005.
- [44] Xiaou Tang, Feng Lin and Jianzhuang Liu, "Video-Based Handwritten Chinese Character Recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 1, Jan. 2005.
- [45] Liana M. Lorigo, Venu Govindaraju, "Offline Arabic Handwriting Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, May 2006.
- [46] L. Tu., et al., "Recognition of Handprinted Chinese Characters by Feature Matching." in Int. Conf on Computer Processing of Chinese and Oriental Languages, pp. 154-157, 1991.
- [47] Y. H. Tseng, C. C. Kuo and H. J. Lee, "Speeding up Chinese Character Recognition and Its Application on Automatic Document Reading," *Pattern Recognition*, vol.31, no. 11, pp. 1589-1600, 1998.
- [48] Z. H. Yang et.al, "A Study of Algorithms for Handheld License Plate Recognition System," National Computer Symposium: Workshop on ICM, pp. 304-315, Taipei Taiwan, Nov. 27-28, 2009.
- [49] R. O. Duda, R. E. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures," *CACM*, vol. 15, no. 1, pp. 11-15, January 1972.
- [50] X. Pan, X. Ye, and S. Zhang, "A Hybrid Method for Robust Car Plate Character Recognition," *J. Eng Appl. Artif. Intell.*, vol.18, no. 8, pp. 963-972, 2005.
- [51] M. Oren et al., "Pedestrian Detection Using Wavelet Templates," in Proc. IEEE Int. Conf. Comp. Vision and Pattern Recognition, pp. 193-199, 1997.

- [52] P. Viola and M. Jones, "Robust Real-Time Object Detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137-154, 2004.
- [53] B. Enyedi, et al, "Strategies for Fast License Plate Number Localization," in IEEE Int. Symp. Electron. Marine, Zadar, Croatia, pp. 579-584, Jun. 2004.
- [54] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," in Proc. IEEE Int. Conf. Image Processing, New York, vol. 1, pp. 900-903, Sep. 2002.
- [55] H. A. Hegt, R. Haye, and N. A. Khan, "A High Performance License Plate Recognition System," in Proc. IEEE Int. Conf. Systems, Man, Cybern., San Diego, CA, vol.5, pp. 4357-4362, Oct. 1998.
- [56] S.-Z. Wang and H.-J. Lee, "Detection and Recognition of License Plate Characters with Different Appearances," in Proc. IEEE Int. Conf. Intelligent Transportation Systems, Shanghai, China, vol.2, pp.979-984, Oct, 2003.
- [57] Y. Amit, D. Geman, and X. Fan, "A Coarse-to-fine Strategy for Multiclass Shape Detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 12, pp. 1606-1621, Dec 2004.
- [58] S. L. Chang, et al., "Automatic License Plate Recognition," *IEEE Trans. Intell. Transport. Syst.*, vol. 5, no. 1, pp. 42-53, Mar. 2004.
- [59] D. U. Cho and Y. H. Cho, "Implementation of Preprocessing Independent of Environment and Recognition of Car Number Plate Using Histogram and Template Matching," *J. Korean Comm. Sci.*, vol. 23, no. 1, pp. 94-100, 1998.
- [60] S. Draghici, "A Neural Network Based Artificial Vision System for License Plate Recognition," *Int. J. Neural Syst.*, vol. 8, pp. 113-126, Feb. 1997.