Generic Signboard Detection in Image and Video

By

SHEN Hua

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

in

Information Engineering

June 2003

Abstract

Signboard detection is important for such computer vision applications as video surveillance and content based visual information retrieval. Previous researches on this topic focus mainly on application-specific signboards such as car plates and traffic signs. Color segmentation, gradient analysis and Hough transform are widely used. Some papers also use neural networks or symmetry transforms to detect signs. However these methods are often restricted to some specific conditions, e.g., the shape of car plates and color of traffic signs. In this thesis, we will present a system that can detect generic signboards in images. The signs can be any polygons, not necessarily restricted to traffic signs or car plates. They can also be posters or doorplates and so on. The only assumption made is that a signboard has to be formed by straight boundaries.

At first, the image is pre-processed, through edge and corner detection. Then, we use the gradient-based Hough transform to detect straight lines. After that, by finding the intersections of the lines and checking the density

of each segment, we can obtain the candidates as the boundaries of signboards. Finally, the location of the signboards can be accurately detected with the processing of finding closed circuits and deleting redundant segments.

The performance of our generic signboard detection algorithm is promising. It gives correct and accurate results in more than 90% of the experiments, which clearly demonstrates the effectiveness and robustness of the new method.

摘要

標牌識別在諸如攝像監控、基於內容的圖像資訊檢索等等電腦視覺應用方面有著舉足輕重的作用。先前的關於此方向的研究都主要集中在有特定目標的標牌檢測上，比方說汽車牌照，或者是交通指示牌等的檢測。常用的方法有顏色分割、梯度分析和哈夫變換。有些論文也提到用神經元網路或者對稱轉換的方法。但是我們可以看到，這些方法對所要檢測的目標有很多的限制。比方說，規定汽車牌照的形狀，或者交通指示牌的顏色。

在本論文裏面，我們將提出一個可以檢測一般性標牌的系統。這些標牌可以是任意的多邊形，並不局限於汽車牌照和交通指示牌。它可以是一張海報，一個門牌等等。我們對標牌的唯一限制就是，它的邊緣必須是直線。也就是說，它不能有弧線的邊緣。

首先，圖像將會進行預處理。這個過程包括邊緣和轉角檢測。然後，我們利用基於梯度的哈夫變換，將圖像中間的直線檢測出來。通過尋找直線的交叉點、檢查線段上面點數的密度，我們可以得到可能

的組成標牌邊緣的線段。通過線段找到合適的閉合曲線，和刪除多餘線段，標牌的具體位置能夠被準確的檢測出來。

實驗結果說明，本文所提出的一般性標牌檢測演算法，能夠準確檢測出超過百分之九十的實驗圖像中的標牌位置。這清楚表明了新演算法的有效性。

Acknowledgments

I would like to thank my supervisor, Dr. Xiaoou Tang, for the interesting problem he introduced to me and for providing useful technical discussions over my research. I would also like to express my sincere appreciation to Dr. Jianzhuang Liu for his helpful suggestions and support.

Special thanks go to all the members in the Multimedia Lab, who shared with me lots of happiness during my study: Feng Lin, Xiaogang Wang, Bo Luo, Zhifeng Li, Lifeng Sha, Tong Wang, Feng Zhao and Dacheng Tao.

I wish to give my deepest gratitude to my grandparents and parents for their disciplines and continuous encouragement. Thank my dear sisters for their love, care and understanding.

Benrong – I cannot express my greatest thank to you enough! I could not have accomplished anything without your support and love.

To My Fiancé…

Benrong Chen,

Table of Contents

List of Figures

Chapter 1

Introduction

It is estimated that 75% of the information received by a human is visual [1]. When one receives and uses visual information, we refer to this process as perception or understanding. When the job is done by a computer, we call it computer image processing and recognition. The manipulation of images by computer is a relatively recent development in terms of humans' ancient fascination with visual stimuli. In its short history, it has been applied to practically every type of imagery, with varying degrees of success.

Computer vision is the science that develops the theoretical and algorithmic basis by which useful information about the world can be automatically extracted and analyzed from an observed image, image set, or image sequence [2].

1.1 Object Detection

In computer vision, detection refers to anything from identifying a location to identifying and registering components of a particular object class at various levels of detail [3]. One could require a precise outline of the object in the image, or the detection of a certain number of well-defined landmarks on the object, or a deformation from a prototype of the object into the image. The object itself may have different degrees of variability. It may be a rigid 2D object, such as a fixed computer font or a 2D view of a 3D object, or it may be a highly deformable object, such as the left ventricle of the heart. All these are considered object-detection problems, where detection implies identifying some aspects of the particular way the object is present in the image, --namely, some partial description of the object instantiation.

We can find a variety of applications in object detection [4]. For example, one of the goals of Intelligent Transportation Systems (ITS) consists of improving traffic safety. One method is to deploy an on-board driver alert system against approaching warning signs such as stop sign, yield sign, etc. Thus detecting such road signs is useful [6]. Object detection can also be applied in building detection and model construction.

It is a problem of great economic importance, since manual delineation is extremely expensive, and there are a number of applications waiting for affordable 3D building data. So it's important to determine the shape and location of buildings within an aerial image robustly and accurately.

Human beings are constantly sensing the environment and accumulating knowledge about the world. From their world knowledge, they can easily segment an image and build descriptions of the scene. Since most physical objects do not produce images that possess a single measurable characteristic, the problem of object detection is very difficult for computers. However, by noting that objects are usually composed of component parts that may possess a uniform characteristic and that are arranged in a particular way, a multistep subdivision of the problem may be devised.

1.2 Signboard Detection

Why we choose detecting signboards? It is due to the useful information contained in the signboards. There are all kinds of signboards in our daily life. When entering a building, it guides us to the correct room. When driving on the road, it tells us where to turn. So signboard detection

is critical for such computer vision applications as video surveillance and content based visual information retrieval.

Chapter 2

System Overview

The purpose of this thesis is to describe an approach to a problem of great practical concern – detecting signboards in a scene from images of the scene.

2.1 What is the problem?

At the very beginning, let's have a general idea of what assumptions should be made to the signboards.

Actually, we made only one assumption for the signboards. That is the signboards should be in shape of any polygons. Or, we can say, the boundaries of the signboards must be straight lines, not curves, such as the examples shown in Figure 2.1.

This is why we call our work "generic" signboard detection. Of course, if the image's quality can be better, the background noise is small, the signboard is big (statistically more than 5% of the whole image's area),

the performance of the system will be better. But all of above are not necessary assumptions we made for the signboards.

Figure 2.1. Examples of signboards.

2.2 Review of previous work

Object detection is quite an application-oriented process. Unfortunately there are few works focus on the signboard we refer to. But we can reference to other relevant applications. Such similar applications include car plate detection, or traffic sign detection [5]-[19].

Several approaches for the recognition of traffic signs have appeared in the literature. Line detection using Hough Transform and difference of gray value are among the major approaches [12]. However, using Hough Transform alone is very sensitive to deformation of a plate boundary and needs much memory.

Blancard [15] has used shape processing to recognize traffic signs. The use of color for traffic sign recognition has also been investigated by researchers[16]-[19]. In [16] Kehtarnavaz et al. have shown that the combination of color and shape processing provides a reliable approach for recognizing stop signs in the presence of brightness changes. Kang et al. [16][17] further investigated this approach by employing a recognition technique in which color segmentation is followed by an invariant signature descriptor and a neural network classifier. Estable et al. [18] have described a traffic sign recognition approach that performs color segmentation by an artificial neural network and a color connection algorithm. These approaches can get good results. Dong-Su Kim [14] employed symmetry transform and image warping method. These approaches can get fairly good results. But among them, there are some other assumptions made for the object at the beginning. For example the shape or corner of the car license / traffic sign [15][17].

The previous works have given us great help in signboard detection. Although there are shortcomings in Hough transform, it is still an effective approach in detecting lines, which will be used in the thesis. Then the work emphasis on the sign locating process, which overcomes the problems

brought by Hough transform. Moreover, we want to propose a method that can detect generic signboards, not restricted to the assumptions like color or shape.

2.3 System Outline

There are generally four parts in the system we proposed, as shown in Figure 2.2.

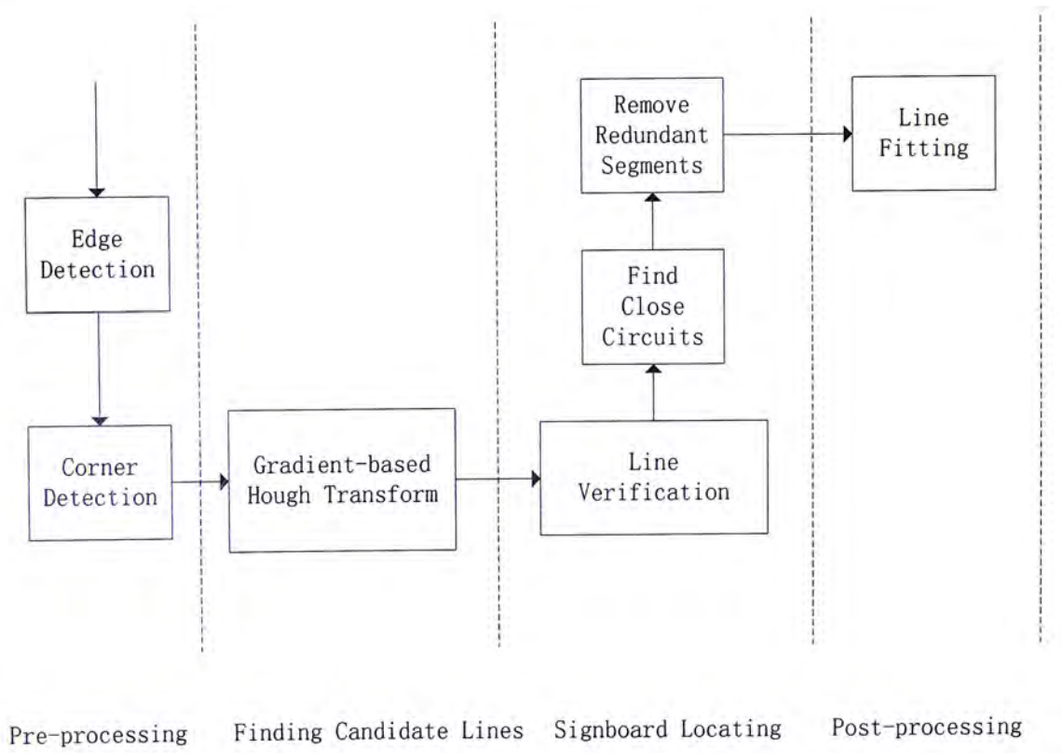


Figure 2.2. System Flowchart.

At first, the image is pre-processed through edge and corner detection. Then, we use gradient-based Hough transform to find the candidate lines that may form the boundaries of the sign.

The main process is the locating of signboards. It is assumed that the signboard's boundaries can form a close circuit. So the system turns to find the close circuits. On the bases of traditional depth-first-search, we propose a new search method that gradually reduces the point's number in the process. Thus the time consumed can be decreased. At the same time, there will be some remaining redundant segments. Then we develop a new method inspired by image coloring to remove those redundant segments.

The last procedure is post-processing, which is consisted of line fitting algorithm. After all these processes, the signboard can be detected accurately. In the following chapters, we'll describe specifically on each step of the system. Conclusions will be made on the experiment results.

Chapter 3

Preprocessing

The images taken outdoors usually have cluttered background. This may cause big errors in the detection step. Thus pre-processing is necessary. The pre-processing is used to erase the messy points roughly, in order for a better performance in the processing steps afterwards.

Unlike car license plates, the signboard's color varies. Thus we cannot take the "color" feature into consideration. This is one of the reasons that make our system more compatible than others.

The pre-processing is consisting of two parts: edge detection & corner detection. Edge detection is used to find the possible points that form the sign's boundary. Then by corner detection, we can delete the corner points appropriately. Thus the irrelevant small segments can be removed naturally.

3.1 Edge Detection

One important feature of the signboards in an image is its boundaries. Thus this property becomes the clue of the detection process. Edge detection is a straight method to detect such boundaries.

Edges are significant local changes of gray level or color in an image. *Edge detection* is an operation that determines if a pixel in an image is an edge point, i.e. a member of an edge.

There are many methods for edge detection. We'll explain three most frequently used edge detection methods here.

3.1.1 Gradient-Based Method

Basically, the idea underlying most edge-detection techniques is the computation of a local derivative operator [27]. From Figure 3.1, we can see that the magnitude of the first derivative can be used to detect the presence of an edge, and the second derivative can be used to determine whether an edge pixel lies on the dark or light side of an edge. Note that the second derivative has a zero crossing at the midpoint of a transition in gray level, shown in Figure 3.1. Thus zero crossings provide a powerful approach for locating edges in an image.

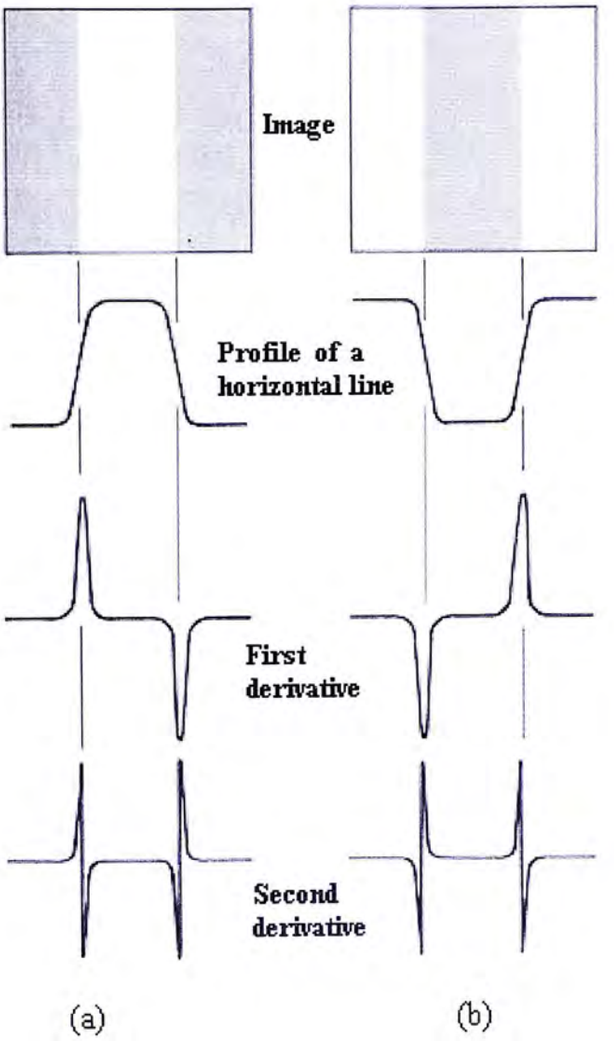


Figure 3.1. First and Second derivative of the image.

The first derivative is obtained by using the magnitude of the gradient at that point. The gradient of an image $x(r,s)$ of continuous spatial coordinates r and s, is

$$G[x(r,s)] = \begin{bmatrix} G_r \\ G_s \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x(r,s)}{\partial r} \\ \dfrac{\partial x(r,s)}{\partial s} \end{bmatrix} \tag{3.1}$$

Hence,

$$\text{Magnitude of } G[x(r,s)] = \sqrt{G_r^2 + G_s^2} \tag{3.2}$$

$$\text{Direction of } G[x(r,s)] = \tan^{-1}\left(\frac{G_s}{G_r}\right) \tag{3.3}$$

In discrete image coordinates, it can be achieved by convoluting the image's pixels with some kind of masks as Figure 3.2.

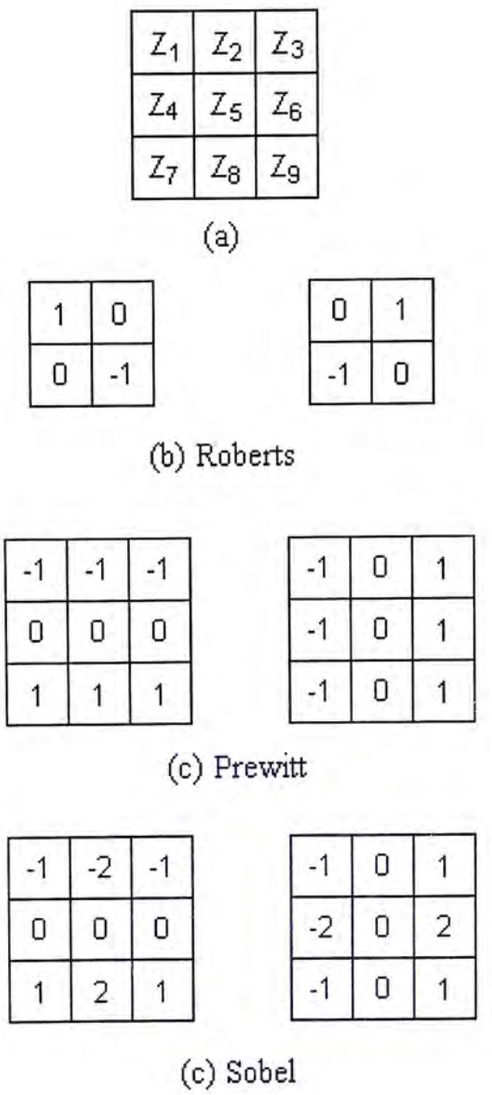


Figure 3.2. Edge detection operators.

Figure 3.2 shows three convolution masks to compute the derivative at point labeled Z5. Let's consider Sobel operator, which is the most frequently used operator, for example. It contains two operators. They are convolved with a digital image $x(m,n)$ to produce $S_h(m,n)$ & $S_v(m,n)$

whose magnitudes represent the amount of changes in horizontal and vertical directions respectively.

The edge magnitude $E_m(m,n) = \sqrt{S_h^2 + S_v^2}$, (3.4)

direction $E_d(m,n) = \tan^{-1}\left[\frac{S_h}{S_v}\right]$. (3.5)

A point (m,n) is said to be an edge point if $E_m(m,n)$ is larger than a threshold.

The algorithm is shared among all other operators in Figure 3.2. The performances of the masks vary [28]. The Roberts operator responds best on sharp transitions in low-noise images. The Prewitt and Sobel operators, being three by three, handle more gradual transitions and noisier images better.

3.1.2 Laplacian of Gaussian

The edge detection operators in Figure 3.2 use only the first derivative. A more elaborate approach is to impose an additional requirement that an edge point should also be a local maximum in edge magnitude, i.e., the 2nd derivative of gray level should be zero [28].

For an image $x(r,s)$ of continuous spatial coordinates r and s, the 2D equivalent of the 2nd derivative is the Laplacian operator, which is

$$\nabla^2 x = \frac{\partial^2 x}{\partial r^2} + \frac{\partial^2 x}{\partial s^2} \qquad (3.6)$$

For an image $x(m,n)$ of discrete spatial coordinates m and n, the Laplacian operator may be approximated by a discrete space LSI system of impulse response:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \qquad (3.7)$$

In many cases, images contain noise, which may result in spurious edges. Applying a smoothing operation to the image before edge detection can reduce the noise level and so the spurious edges. A commonly used smoothing operator is Gaussian filter. Thus the Laplacian of Gaussian method is stated as following:

- Use a Gaussian filter to smooth out noises first and then
- Use the Laplacian operator to detect the locations of the zero crossings of the 2nd derivative
- Identify the detected locations as edge points if the magnitudes of the first derivative are larger than a threshold.

3.1.3 Canny edge detection

The Canny method finds edges by looking for local maxima of the gradient. The method uses two thresholds, to detect strong and weak edges,

and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be "fooled" by noise, and more likely to detect true weak edges. Thus it is the edge detection method we use in the system.

The Canny edge detector in discrete spatial coordinates is

- Use a Gaussian filter to smooth out noises,
- Compute $G(x(m,n))$, the gradient of $x(m,n)$

$$G_m(x(m,n)) = \{x(m+1,n) - x(m,n) + x(m+1,n+1) - x(m,n+1)\}/2 \quad (3.8)$$

$$G_n(x(m,n)) = \{x(m,n+1) - x(m,n) + x(m+1,n+1) - x(m+1,n)\}/2 \quad (3.9)$$

$$\text{Magnitude of } G[x(m,n)] = M(m,n) = \sqrt{G_m^2 + G_n^2} \quad (3.10)$$

$$\text{Direction of } G[x(m,n)] = \theta(m,n) = \tan^{-1}\left(\frac{G_n}{G_m}\right) \quad (3.11)$$

- Apply nonmaxima suppression to $M(m,n)$ to form $N(m,n)$
  - Find $\xi(m,n) = \xi(\theta(m,n)) \in [0,3]$
  - Compare $M(m,n)$ with its two neighbors along the direction given by $\xi(m,n)$. If $M(m,n)$ is not greater than both neighbors, then $N(m,n)$ is set to zero, otherwise $M(m,n)$.
- Apply double thresholding to $N(m,n)$ to form the edge map.
  - Apply two thresholds t_1 & $t_2 \approx 2t_1$ to form $T_1(m,n)$ & $T_2(m,n)$.
  - Use $T_1(m,n)$ to link edges in $T_2(m,n)$.

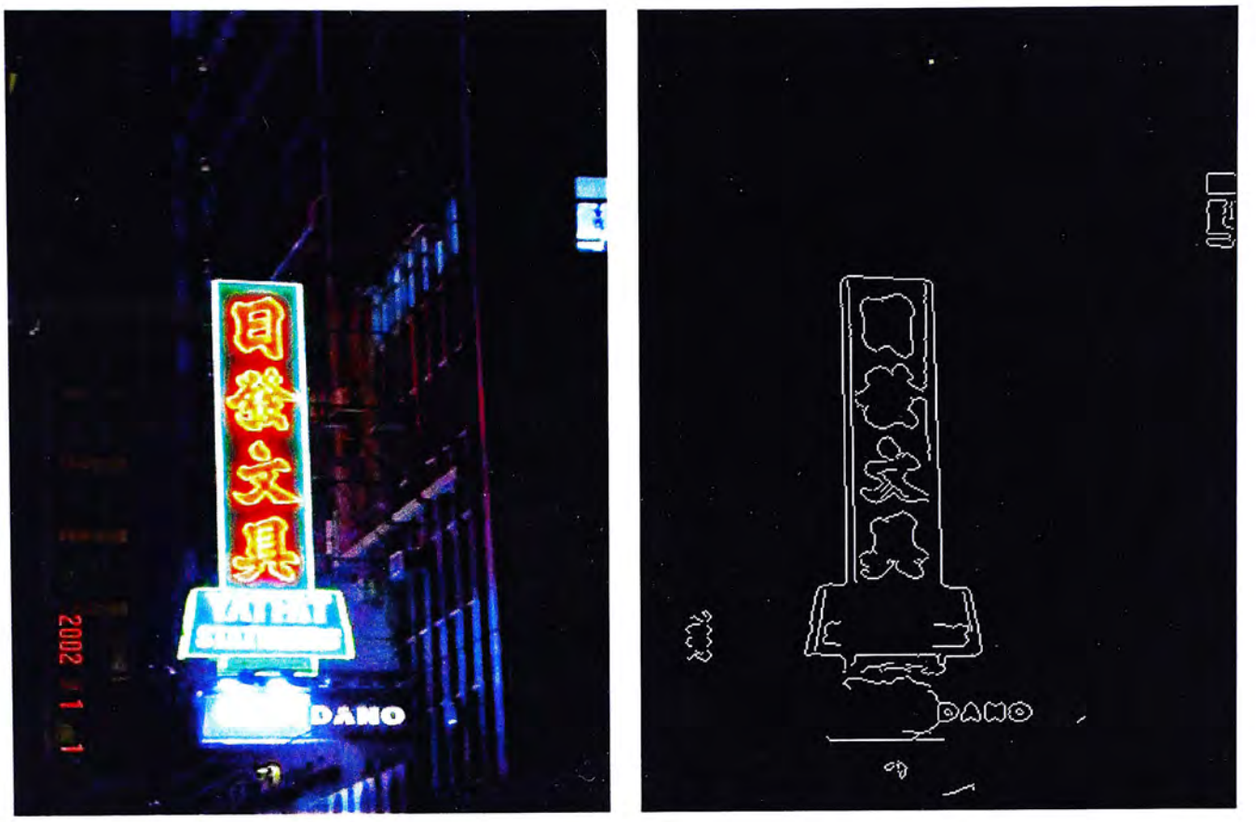


Figure 3.3. Image before and after Canny edge detection.

Figure 3.4. Image before and after Canny edge detection.

(with too many short segment remain)

Figure 3.3 shows an example of Canny edge detection. We can see that the boundary of the signboard is detected. But for different images, the result will sometimes not be so satisfying. The leaves beside the signboard in Figure 3.4 cause a large amount of edges. Thus the system needs the corner detection process afterwards to remove them.

3.2 Corner Detection

As we can see from Figure 3.4, some images cannot get good results after Canny edge detection. It is mainly because of the messy background. In here, we apply a corner detection method.

The approach comes from the fact that the signboards have relatively straight long boundaries. But the noise of the background may be short discontinuous segments. So if the corners of both sign and noise are detected [29] and removed, the noise will be shorter, but no obvious effects happen to the sign's boundaries. Thus we can denoise the whole image so that to make it easier for the processing afterwards.

How do we detect corner features? How do we detect corner features? Consider the spatial image gradient, $[E_x, E_y]^\Gamma$ (the subscripts indicate partial differentiation, e.g., $E_x = \frac{\partial E}{\partial x}$). A generic image point, p, a neighborhood Q of p, and a matrix, C, defined as

$$C = \begin{bmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{bmatrix} \qquad (3.12)$$

Where the sums are taken over the neighborhood Q. This matrix characterizes the *structure* of the gray levels. How?

The key to the answer is in the eigenvalues of C and their geometric interpretation. Notice that C is symmetric, and can therefore be diagonalized by a rotation of the coordinate axes; thus, with no loss of generality, we can think of C as a diagonal matrix:

$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \qquad (3.13)$$

The two eigenvalues, λ_1 and λ_2, are both nonnegative. Let us assume $\lambda_1 \geq \lambda_2$. The geometric interpretation of λ_1 and λ_2 can be understood through a few particular cases. First, consider a perfectly uniform Q: the image gradient vanishes everywhere, C becomes the null matrix, and we have $\lambda_1 = \lambda_2 = 0$. Second, assume that Q contains an ideal black and white step edge: we have $\lambda_2 = 0, \lambda_1 > 0$, and the eigenvector associated with λ_1 is parallel to the image gradient. Note that C is rank deficient in both cases, with rank 0 and 1 respectively. Third, assume that Q contains the corner of a black square against a white background: as there are two principal directions in Q, we expect $\lambda_1 \geq \lambda_2 > 0$, and the larger the

eigenvalues, the stronger (higher contrast) their corresponding image lines are. At this point, you have caught on with the fact that the eigenvectors encode edge directions, the eigenvalues edge strength. A corner is identified by two strong edges; therefore as $\lambda_1 \geq \lambda_2$, a corner is a location where the smaller eigenvalue, λ_2, is large enough.

We now summarize the procedure for locating the corners:

The input is formed by an image, I, and two parameters: the threshold on λ_2, τ, and the linear size of a square window (neighborhood), say 2N+1 pixels.

- Compute the image gradient over the entire image;
- For each image point p,
  - Form the matrix C (in eq. 3.11) of over a $(2N+1)\times(2N+1)$ neighborhood Q of p;
  - Compute λ_2, the smaller eigenvalue of C;
  - If $\lambda_2 > \tau$, save the coordinates of p into a list, L.
- Sort L in decreasing order of λ_2.
- Scanning the sorted list top to bottom: for each current point, p, delete all points appearing further on the list which belong to the neighborhood of p.

The output is a list of feature points for which $\lambda_2 > \tau$ and whose neighborhoods do not overlap.

After the corners are detected, the short segments become shorter, while the long boundaries have no change. Then remove the segment shorter than a threshold; the image can be much clear, shown in Figure 3.5. Thus the main processing can be applied to this image.

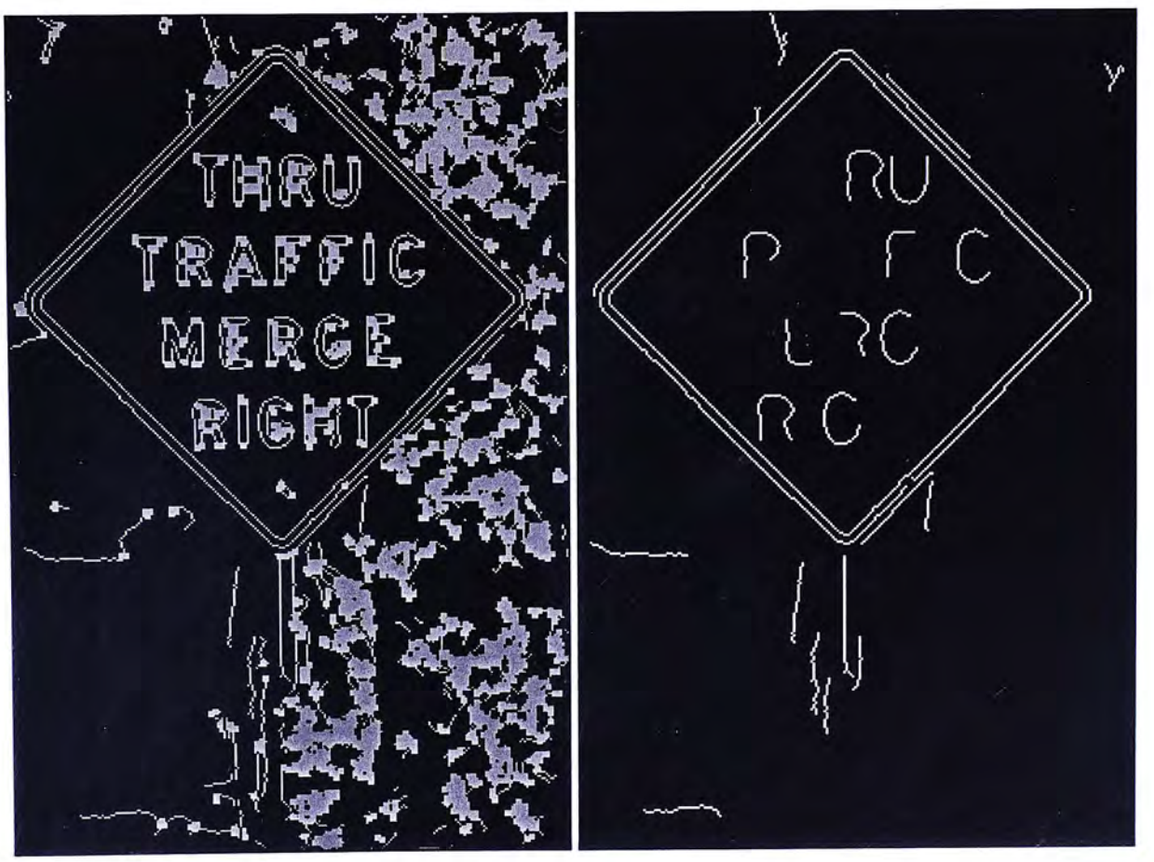


Figure 3.5. Cutting short segments after corner detection.

Left: Corners detected; Right: Edges after deleting short segments.

Chapter 4

Finding Candidate Lines

4.1 Hough Transform

4.1.1 What is Hough Transform

The *Hough transform* [30] is a standard tool in image analysis that allows recognition of global patterns in an image space by recognition of local patterns (ideally a point) in a transformed parameter space. In particular, it is used in our system to detect straight lines.

The basic idea of this technique is to find straight lines that can be parameterized in a suitable parameter space.

4.1.2 Parameter Space

We can analytically describe a line segment in a number of forms. However, a convenient equation for describing a set of lines uses the following parameter form:

$$d = x\cos\theta + y\sin\theta \qquad (4.1)$$

where d is the length of a normal from the origin to this line and θ is the angle with the normal. (See Figure 4.1) For any point on this line, d and θ are constant.

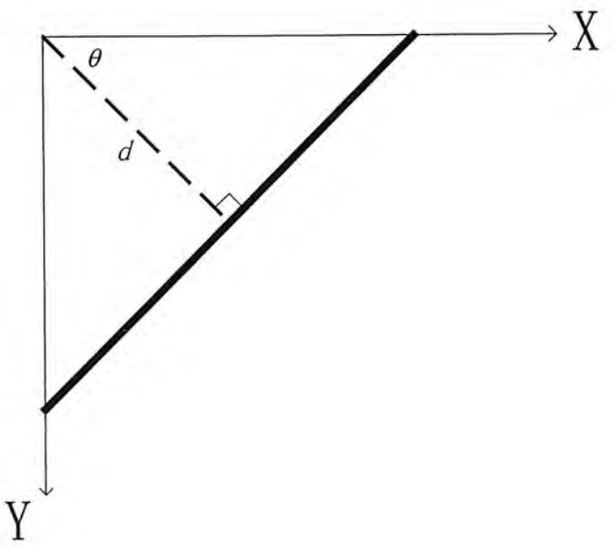


Figure 4.1. Parameters used in the Hough transform.

The edge points after the preprocessing are the input of the Hough transform. The coordinates of the points are known. Therefore they serve as constants in the parametric line equation, while d and θ are unknown variables. If we plot the possible (d,θ) values defined by each point's coordinates in xy image space map to curves (i.e., sinusoids) in the polar Hough parameter space. This point-to-curve transformation is the *Hough transformation*. When viewed in Hough parameter space, points that are

collinear in the xy space become readily apparent as they yield curves, which intersect at a common (d,θ) point. See in Figure 4.2.

4.1.3 Accumulator Array

In order to describe the parameter space, the Hough transform algorithm requires an *accumulator array* whose dimension corresponds to the number of unknown parameters (2 parameters for lines) in the equation of the family of curves being sought [2]. The transform is implemented by quantizing the Hough parameter space into finite intervals or *accumulator cells.*

The Hough transform runs like this: for each edge point (x,y), calculate the distance $d = x\cos\theta + y\sin\theta$, for every possible value from 0 to 360 of θ. Thus a curve of (d,θ) is made by edge point (x,y). Peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image.

In order to illustrate the Hough transform in detail, we begin with an example shown in Figure 4.2 [31]. (a) is the original image, and (b) is the input of Hough transform. (c) shows the curves in parameter space, or accumulator array. The more curves overlapped, the more lighter the color is. By seeking for these peaks in the accumulator array, we can get the most

possible lines' parameters (d,θ). (d) in Figure 4.2 shows the result of Hough transform.

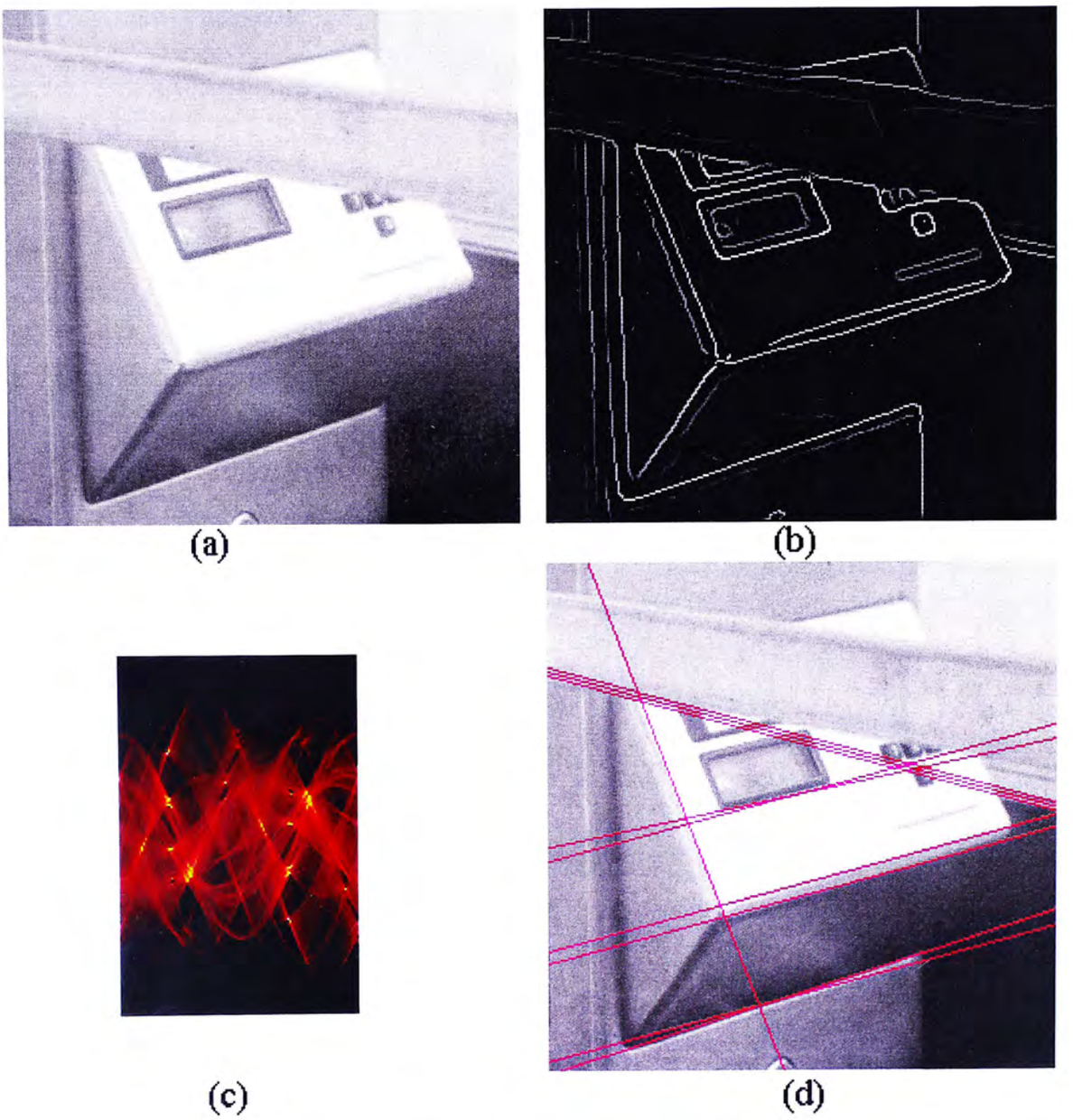


Figure 4.2. Illustration of Hough Transform.

4.2 Gradient-based Hough Transform

The main problem with the classic Hough transform comes from the huge computational workload. Because for every point in the edge map, we

need to calculate 360 (d,θ) pairs for them, if quantization interval is 1 degree for angle θ.

The basic idea of the gradient-based Hough transform [32] comes from the idea that, point is most likely belong to the line that has the same gradient as the point. If we can calculate the gradient of the point, then we assume that the line passing through that point must have the same gradient. Then the parameter θ is unique. So there is only one parameter pair for that point. Although this algorithm is sometimes not so accurate due to the gradient error, it is still used because the great amount of time it saved. Because in classic Hough transform, the parameter θ is changing from 0 to 360 degrees. Although in this algorithm, there is only one value for θ. The gradient error can be overcome in the post processing steps.

4.2.1 Direction of Gradient

The relationship between parameter θ and the direction of gradient is illustrated in Figure 4.3. r represents the vertical gradient. c represents the horizontal gradient. Thus we have

$$tg(\theta) = \frac{r}{c}. \qquad (4.2)$$

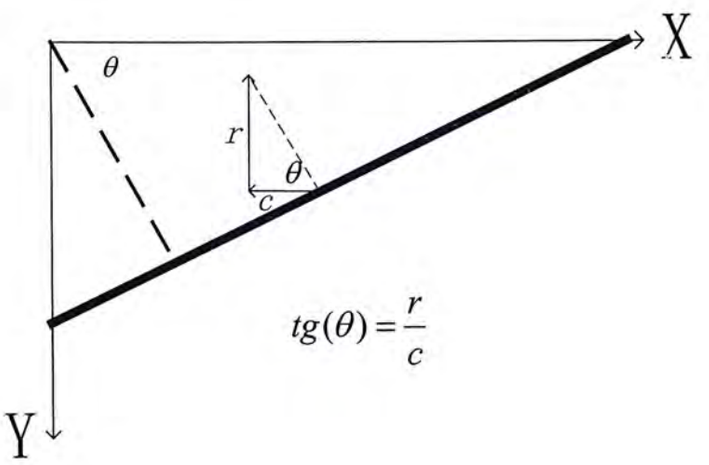


Figure 4.3. Direction of gradient.

The horizontal and vertical gradient is calculated using Sobel operator in Figure 4.4.

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

(a) Vertical Gradient

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

(b) Horizontal Gradient

Figure 4.4. Sobel operator for direction of gradient.

Let's see a simple example in Figure 4.5 (a) shows a straight line with $\theta = 71.5^{\circ}$. The rectangular part is enlarged into (b). They are convolved with the Sobel operator respectively; r and c are their result. Then using (4.2), we can thus get the gradient's direction angle for each point.

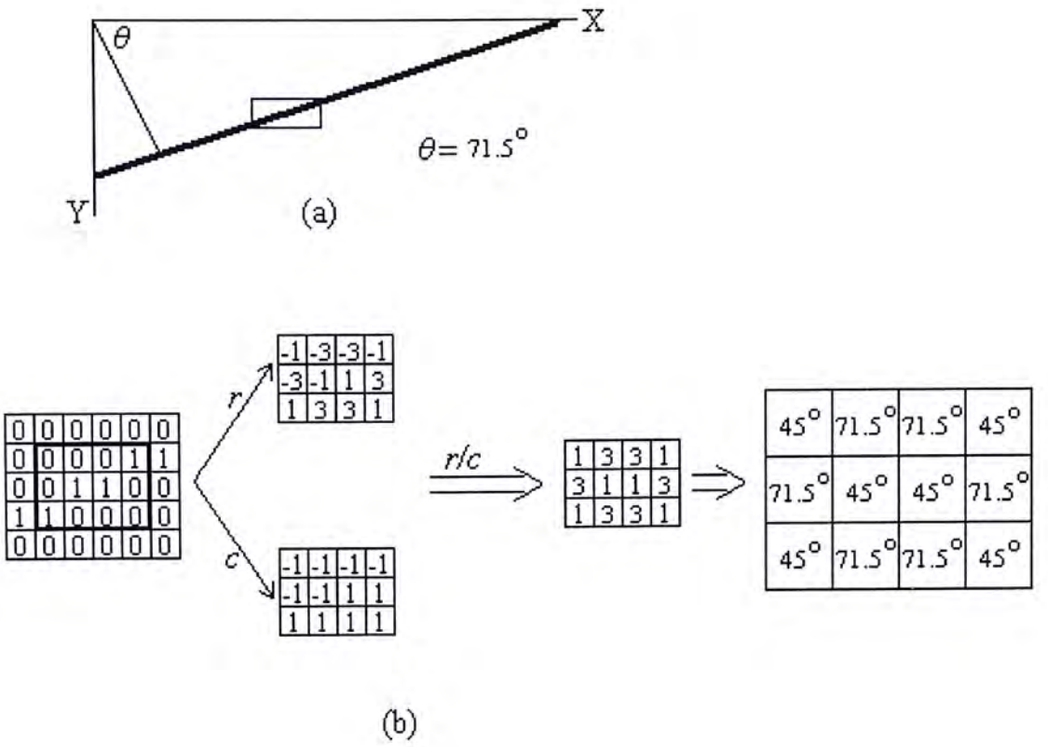


Figure 4.5. An example of gradient calculation.

We can see from Figure 4.5(b) that the 4-neighborhoods of the edge points can represent the angle accurately (Their gradients are 71.5°. Errors will occur at the edge points and their 8-neighbors. But we don't need to worry about it. These points cannot make up a peak in the accumulator at all. They don't even belong to the line decided by their coordinates and the wrong gradient angle.

4.2.2 Accumulator Array

In our application, parameter θ means the angle between the normal and the x-axis. The edge points are all in the area with $x > 0, y > 0$. Thus

the range for θ is $[0,180)\,\&\,(270,360)$. The range for d is (0, dmax), where dmax=length(diagonal of the image).

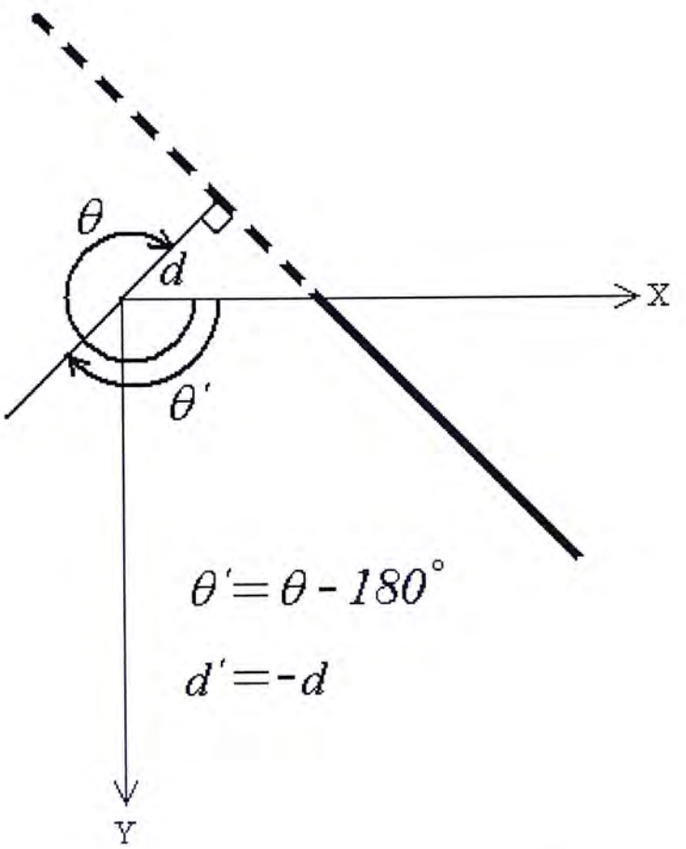


Figure 4.6. Ranges of the parameters in Hough transform.

But as we calculate $\theta = tg^{-1}\left(\frac{r}{c}\right)$, a negative value will be resulted when θ is larger than 180. Therefore the real value of θ should be

$$\theta = tg^{-1}\left(\frac{r}{c}\right) + 360^\circ \quad \text{if} \quad \theta = tg^{-1}\left(\frac{r}{c}\right) < 0 \qquad (4.3)$$

For convenience, we define the line with $\theta \in (270,360)$ to $(90,180)$. See in Figure 4.6. From the definition before, θ, d are shown in the figure. But we use θ' instead of θ, which

$$\theta = \theta - 180^\circ \quad \text{if} \quad \theta > 180^\circ \qquad (4.4)$$

Therefore the range for θ is [0,180). Then the distance is

$$d = 0 - d \quad \text{if } \theta > 180, \qquad (4.5)$$

as illustrated in Figure 4.6. Then the range for d is (-dmax, dmax).

As the algorithm runs, each point (x, y) is transformed into a parameter (d, θ) point. Quantize such parameters $d \& \theta$ into two arrays: $qd \& qt$. These two arrays have the same size with the image. They record the quantized $d \& \theta$ for each point. The range of θ is [0,180) , totally 180 degrees. Suppose the quantization interval is 5°. Then there will be 180/5=36 angles for one point to calculate the distance *d*. But if we use gradient information, we only need to calculate once. So it saved 35/36=97.22% time.

When the parameters (qt, qd) are calculated, the corresponding accumulator cell $A(qt, qd)$ is incremented by one.

4.2.3 Peaks in the accumulator array

As we have discussed before, the straight lines will create a peak in the accumulator. By seeking for such peaks, we can get the parameters of the lines.

But because of the quantization error of the parameters, the lines will not be very accurate. So once the peak $A(qt, qd)$ is found, we search back

to the points that belong to this line. Usually the neighborhoods are correctly represented by the parameters. So these points' neighborhoods are also considered as the line's points. Also the accumulator cell $A(qt,qd)$'s neighborhoods are set to be zero. Thus it won't result in many closed nearly parallel lines, as shown in Figure 4.2. But what is the appropriate value of (qt,qd) we assigned to this line?

$$qt = \frac{\sum qt(x,y)}{n}, qd = \frac{\sum qd(x,y)}{n} \qquad (4.6)$$

where (x,y) are the coordinates of the line's points and their neighborhoods. n is the number of these points. By such method, the errors can be reduced. And the parameters (qt,qd) are adjusted.

We now summarize the procedure for the gradient-based Hough transform as follows:

- Compute r and c using Sobel operator in Figure 4.4.
- Compute parameter θ:
  - For points with $c = 0 \,\&\, r \neq 0 \Rightarrow \theta = 90^\circ$,
  - For points with $c \neq 0 \Rightarrow \theta = tg^{-1}\left(\frac{r}{c}\right)$.
  - If $\theta < 0 \Rightarrow \theta = \theta + 360^\circ$. If $\theta = 180^\circ \Rightarrow \theta = 0^\circ$
- Compute parameter d:

$$d = x\cos\theta + y\sin\theta$$

- Adjust $d \,\&\, \theta$:

  For points with $\theta > 180^\circ \Rightarrow \theta = \theta - 180^\circ \,\&\, d = 0 - d$.

- Quantize $d \,\&\, \theta$ into two arrays: $qd \,\&\, qt$. These two arrays have the same size with the image. They recorded the quantized $d \,\&\, \theta$ for each point.

- Create an accumulator A. For each point with $r^2 + c^2 > 0$, and has the parameter $(qd, q\theta)$, increase the accumulator cell with 1:

$$A(qd, q\theta) = A(qd, q\theta) + 1$$

- Search for the peaks in the accumulator A. The coordinates of these peaks represent the quantized parameters for the straight lines.

4.2.4 Performance of Gradient-based Hough Transform

The result of the gradient-based Hough transform is shown in Figure 4.7 & Figure 4.8.

Figure 4.7. Result of gradient based Hough transform (1).

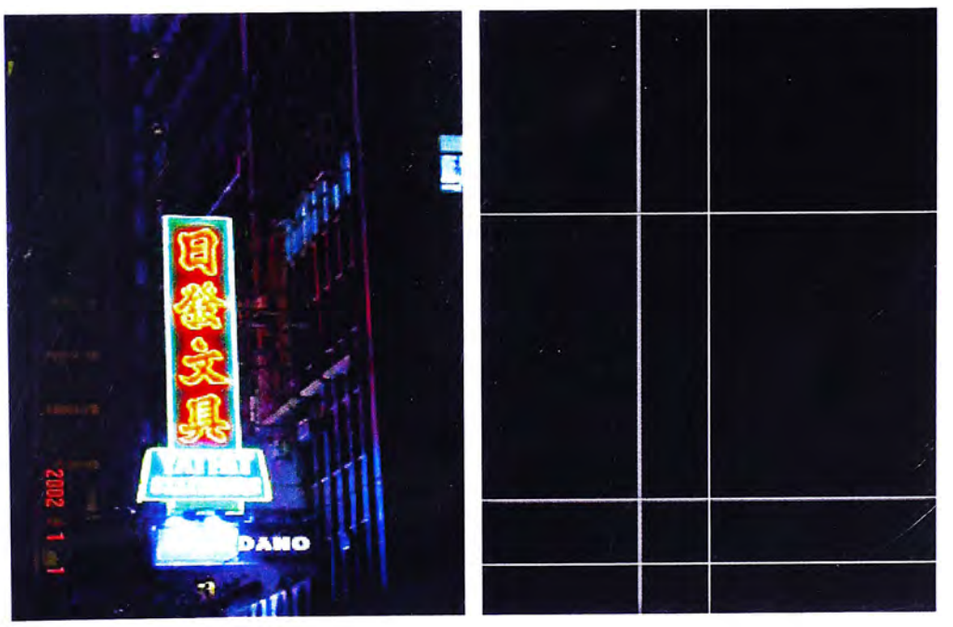


Figure 4.8 Result of gradient based Hough transform (2).

Compare Figure 4.7 with Figure 4.2(d), we can see that the gradient based Hough transform can get as good performance as classic Hough

transform. In addition, due to the peak-seeking method we used in 4.2.3, the straight lines are more accurate and unique. The most attractive property of gradient-based Hough transform is the great reduction of time and memory consuming comparing with classic Hough transform.

Chapter 5

Signboards Locating

5.1 Line Verification

In the previous chapter, the lines of the signboards are detected using gradient-based Hough transform. Here we have one question now: which segments of the lines are the sign's boundaries?

5.1.1 Line Segmentation

After Hough transform, what we get are the most possible lines in the image. In order to find out the two ends of the boundary segments, we can use the equations of the lines to get their intersections.

The input are the parameters (d,θ) found by the gradient based Hough transform. Therefore the equation of the line with the parameters (d,θ) is:

$$\begin{aligned} tg\theta \cdot x + y &= \frac{d}{\cos\theta} \qquad (\text{if } \theta \neq 90^\circ) \\ x &= d \qquad\qquad (\text{if } \theta = 90^\circ) \end{aligned} \tag{5.1}$$

Three parameters (a_i, b_i, c_i) are recorded for these lines:

$$\begin{cases} a_i = tg\theta \\ b_i = 1 \quad (\text{if } \theta \neq 90^\circ) \\ c_i = \dfrac{d}{\cos\theta} \end{cases} \qquad \begin{cases} a_i = 1 \\ b_i = 0 \quad (\text{if } \theta = 90^\circ) \\ c_i = d \end{cases} \tag{5.2}$$

Also there are four lines we should not ignore: four borders of the image. Their equations are: $y = 1, y = y_{\max}, x = 1, x = x_{\max}$. $(x_{\max}, y_{\max})$ represents the size of the image. Then the four sets of parameters should be added into the series:

$$\begin{cases} a = 0 \\ b = 1, \\ c = 1 \end{cases} \begin{cases} a = 0 \\ b = 1 \\ c = y_{\max} \end{cases}, \begin{cases} a = 1 \\ b = 0, \\ c = 1 \end{cases} \begin{cases} a = 1 \\ b = 0 \\ c = x_{\max} \end{cases} \tag{5.3}$$

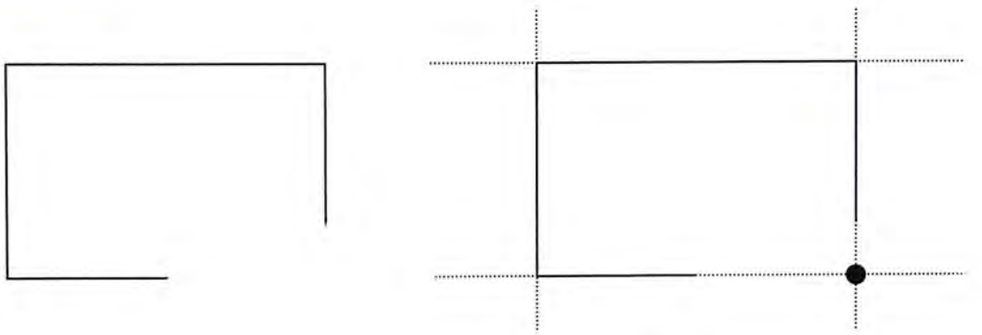
After all the parameters (d, θ) are transformed into (a_i, b_i, c_i), and the four borders are added, we can get the intersections of each two lines:

Suppose the two lines are of the parameters (a_1, b_1, c_1) and (a_2, b_2, c_2), then their intersections are:

$$x = \frac{c_1 b_2 - c_2 b_1}{a_1 b_2 - a_2 b_1}, y = \frac{c_2 a_1 - c_1 a_2}{a_1 b_2 - a_2 b_1} \tag{5.4}$$

By this means, the lines are divided into segments caused by the intersections of each other. This step is important and useful especially when the polygon's corners are covered by other objects. In this case, there is no point on the corner of the boundary segments. Thus make the boundary incomplete, as shown in Figure 5.1. But if we find the

intersections of the boundaries (shown as the emphasized point in Figure 5.1(b)), the segments will be extended automatically, and then complete the whole boundary.



(a) (b)

Figure 5.1. Line segmentation when the corner is covered.

5.1.2 Density Checking

In 5.1.1, the lines are divided into segments. If the segment is really the boundary, there must many points with the same gradient and distance lying on that line. So if we check the point's density on the segment, then the segments with the density over a threshold (set to be 0.4 in the experiment) are kept, and other segments are removed. See Figure 5.2. (a) is the output of gradient-based Hough transform. They are the straight lines in the image. Figure 5.2(b) shows the points that contribute in the accumulator cells in (a). And also these points are the bases of density checking process.

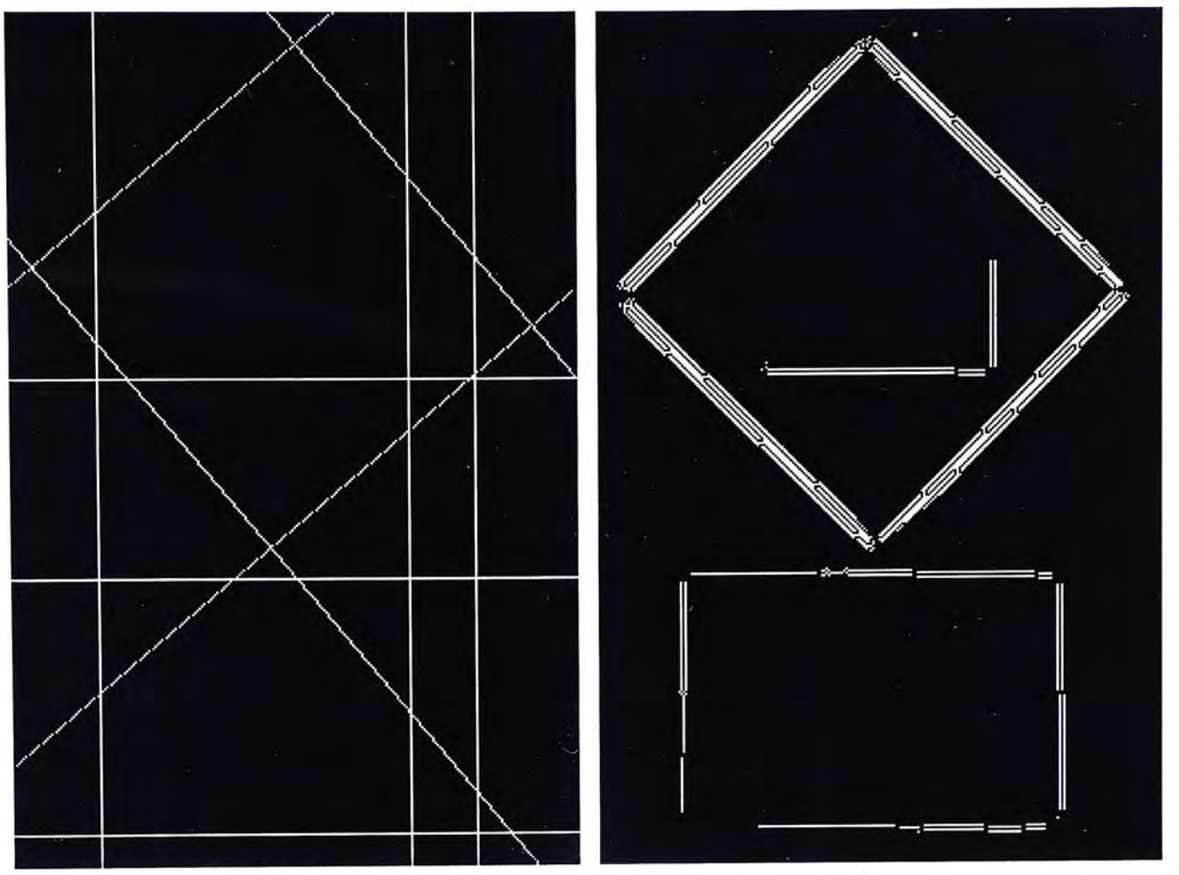


Figure 5.2. Lines and the points belonging to them.

Checking density process is to project the points on the segment. And then check such a segment's density.

Given a segment with parameters (a,b,c). Firstly, plot the segment; see the example in Figure 5.3(a). It is a segment from Figure 5.2. Then and extend its width as shown in Figure 5.3(b).

Secondly, Find the overlapping points in both the segment and the points contribute to the accumulator cell. In the example, it means to find the points being white both in Figure 5.3(b) and Figure 5.2(b).

If the point's density is larger than a threshold (set to be 0.4 in the experiment), then the segment is kept. Otherwise, the segment will be removed. Figure 5.4(b) shows such a result after density checking.

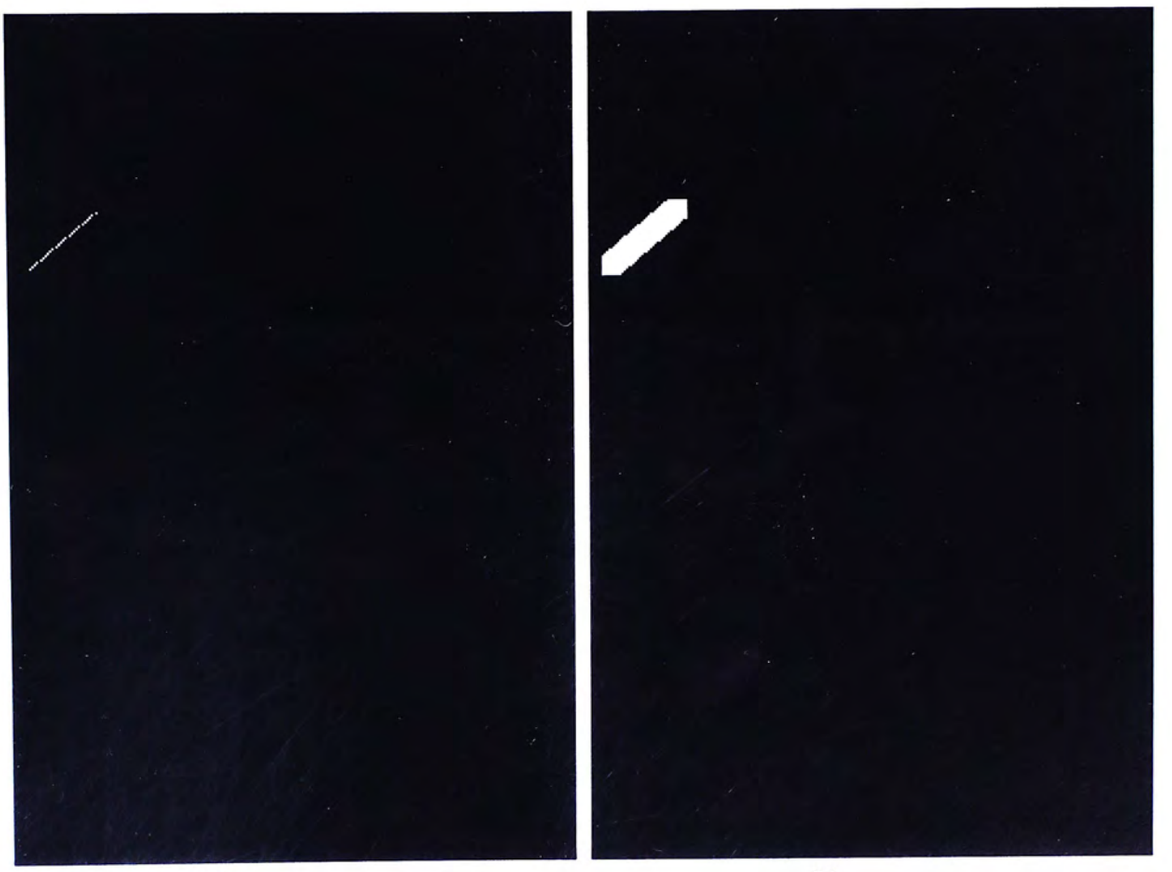


Figure 5.3. Expand the segment's width.

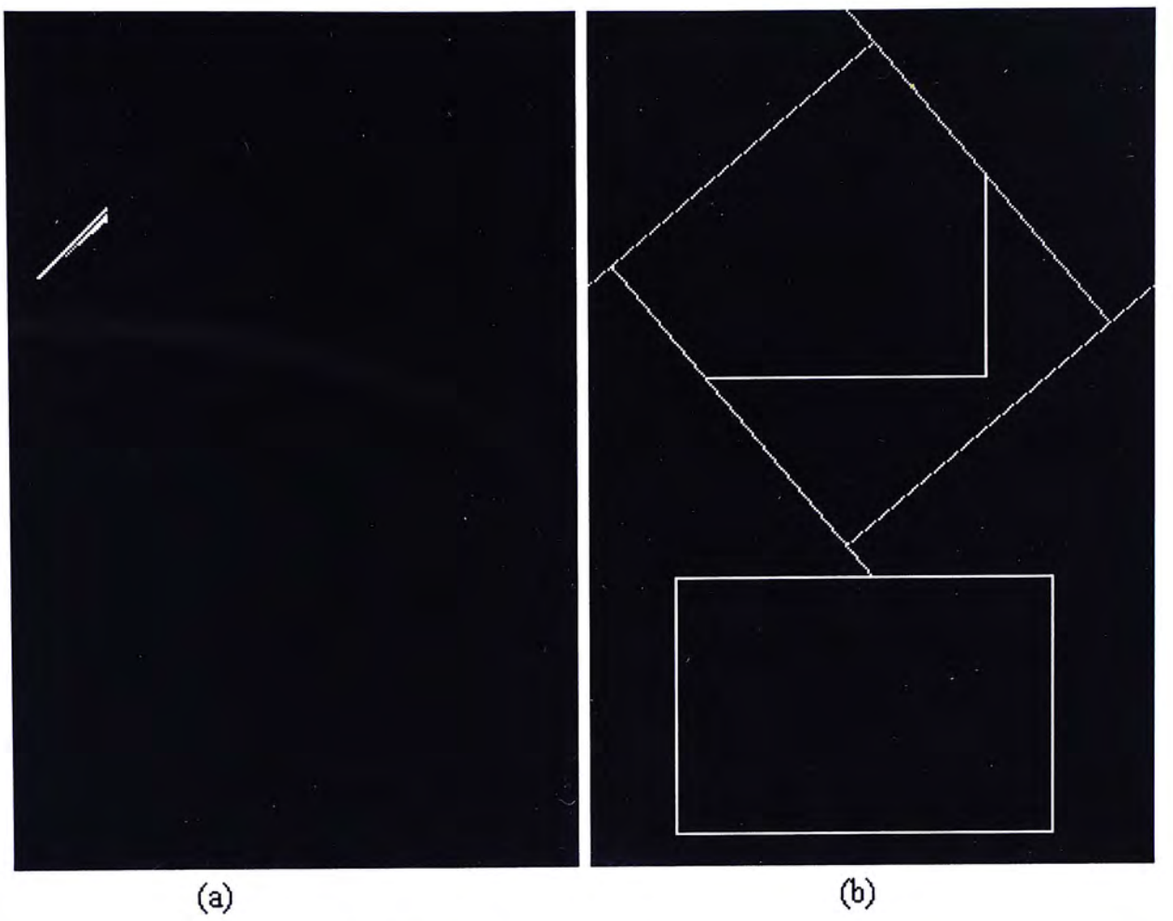


Figure 5.4. Result of density checking.

5.2 Finding Close Circuits

Till now, the most possible segments that formed the signboards are found. It is supposed that if they can form some close circuits, then the circuits are the signboards we detect. Some of the segments may not be connected due to the noise, but in fact they are connected. So we must first merge some intersections that are close enough. Then the task ahead is to find the close circuits.

There are many methods to find close circuits. Mainly are using depth-first-searching method [33]. It is a classical and effective method. If we want to find out every possible closed circuit, depth-first search should be used for every points, that may seem too tedious and memory consuming. In our system, the time and memory consuming is a key factor that we're considering about.

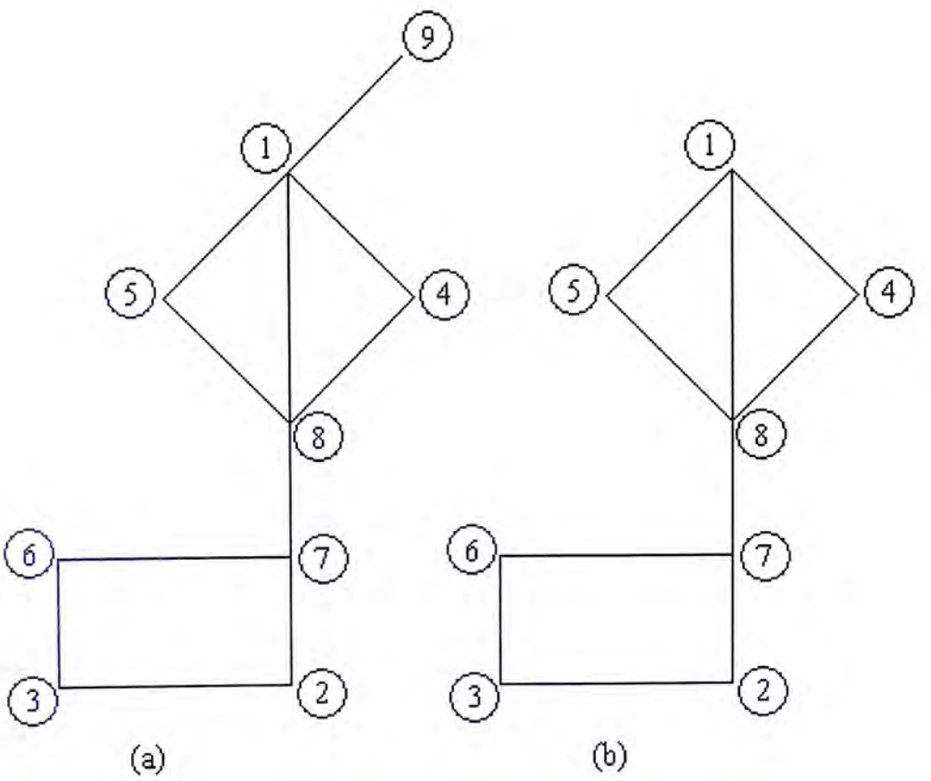


Figure 5.5. Illustration of finding close circuits (1).

The main idea of the proposed algorithm is trying to reduce the number of vertex in the image. Therefore the complexity will be smaller. We'll explain it using the example shown in Figure 5.5

At first, assign a number randomly to each of the point. Delete the points with only one segment connected with them. For example, the point 9 in Figure 5.5 (a) is such a point. There is only one segment '19' connected with point 9. "Delete" here means delete the point and all the segments that end at this point. The image after "deleting" is shown in Figure 5.5(b). Repeat such a check until every point is connected with more than 1 segment. If no point remained, the algorithm terminates.

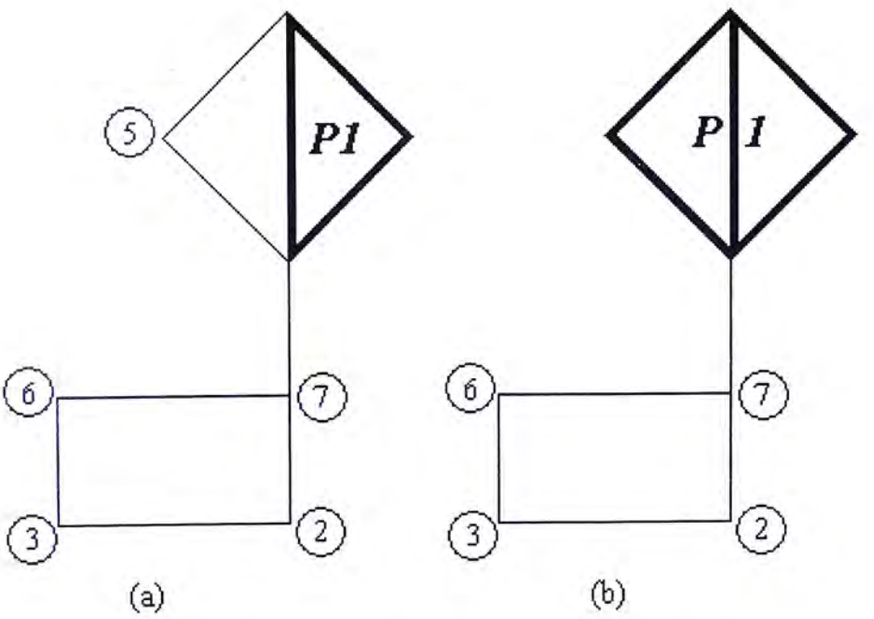


Figure 5.6. Illustration of finding close circuits (2).

Then begin with the first point remained. Find the first loop starts and ends at it. Number the loop as loop No.1. And record the loop's points and segments. In the example of Figure 5.5(b), the first point is point No. 1. It

can get a result using the DFS in Figure 5.6(a). The thick lines represent the first close circuit. Put the loop's points (1,4,8) into a point set P1. Thus the two sets are

$$P1 = \{1,4,8\} \qquad (5.5)$$

Regard the point-set P1 as only one point. This point has the same priority as other points. Then the points remained are: P1, 2, 3, 5, 6 and 7. The only difference between P1 and other points is that, there may exist two segments connecting one point and P1, while there must be only 1 segment connecting two ordinary points. For example there are 2 segments connecting 5 and P1. But this property will not affect the processing afterwards.

Then, try to find another closed circuit. In the example, such a closed circuit is shown in thick lines of Figure 5.6(b). The path is: P1-5-P1. Because some of its points comes from the point set P1, then point 5 is also added into point set P1. Thus

$$P1 = \{1,4,5,8\} \qquad (5.7)$$

as shown in Figure 5.6(b)

Repeat finding the closed circuits. In the example the next closed circuit is shown in dotted lines of Figure 5.7(a). In this close circuit, every

vertex is an ordinary point (not in point set). So classify the points 2, 3, 6 and 7 into point set P2.

$$P2 = \{2,3,6,7\} \qquad (5.8)$$

Now there are only two points in the image: P1 & P2. And no close circuit exists. We'll begin check the connections between point sets. In the example, there is a segment connecting point sets P1 and P2. In this case, these two sets are merged into one point: P1, shown in Figure 5.7(b).

Continue such a checking until the point sets are all isolated. Then the finding process terminates. The results are the point sets. In the example, the result is point set P1.

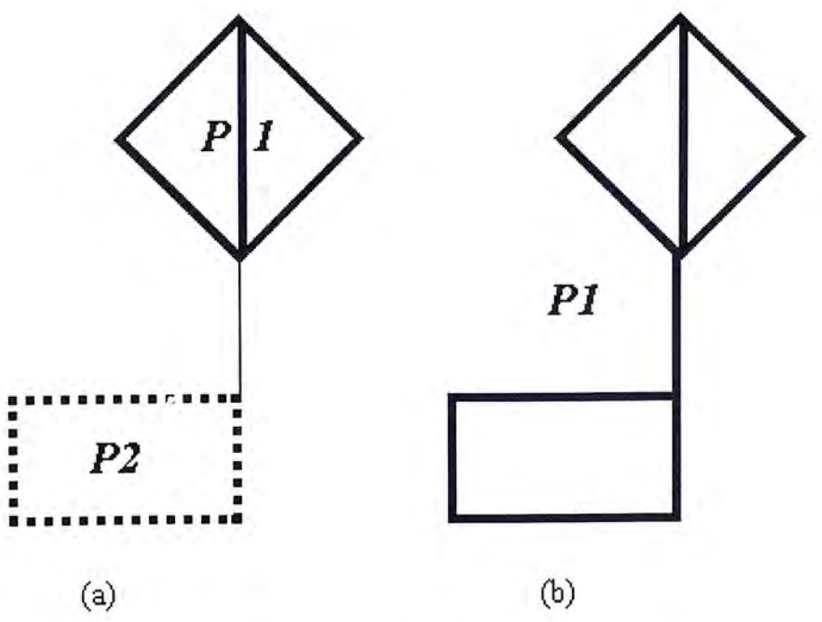


Figure 5.7. Illustration of finding close circuits (3).

The signboard's boundaries are detected. Although there are some redundant segments, but at least, we did not miss any of the true segments. Removing redundant segments are processed afterwards.

So let's summarize the finding signboards algorithm as follows:

- Number all the vertexes in the image.
- Delete the points with only one segment connecting with them.

  Double check until no such points exist.
- Find a close circuit.
  - If the vertexes of the circuit are all ordinary points:

    Create a point set P_i to record the vertexes. Shrink them into only one point P_i.
  - If one of the vertexes is point set P_i

    Merge all the points in the circuit into point set P_i.
  - If more than one of the vertexes are point set, for example, $P_i, P_j, ...P_n$ (assume $i < j < ... < n$):

    Merge all the points in the circuit into point set P_i, and

    Merge all the points in $P_j, ...P_n$ into P_i

    Delete point set $P_j, ...P_n$.
- Repeat finding close circuits until no close circuits exist.

- Check the connections between each of the point sets. If there is a segment connecting P_i & P_j (assume $i < j$), then merge P_i & P_j into P_i. And delete point set P_j.
- Repeat checking until all the point sets are isolated.
- The output is the point sets remained.

This method is more efficient than classic DFS. It reduces the point's number gradually. Although it may add some redundant segments in the result, but at least, it won't miss any close circuit at all.

(a) (b)

Figure 5.8. Signboards found in the image.

Figure 5.8 shows the result of this step. The signboards can be detected correctly. But at the same time, there are some redundant segments still remain, as (b) in Figure 5.8, or (b) in Figure 5.7. If such redundant segments exist, we need to remove them appropriately, which will be described in the next session.

5.3 Remove Redundant Segments

Although the closed loops can be detected, there exist some redundant segments that we have discussed before. The problem can be generally classified into three kinds, shown in Figure 5.9. The middle line in (a), the smaller rectangle in (b), and the connecting segment in (c), are redundant segments that appeared in our experiments. Since there are no algorithms specified for this goal, we designed an algorithm to achieve it.

The algorithm is originally comes from the idea of coloring. It tries to color the background pixels, which also surround the outer closest circuit.

Then the line pixels with these colored pixels around are the pixels we want.

Let's specify the algorithm using the example of Figure 5.9(a).

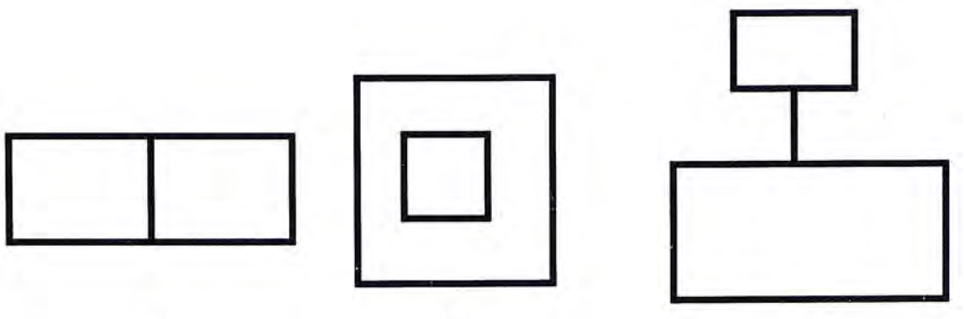


Figure 5.9. Three types of redundant segments.

At the beginning, it is defined that '1' is for the line pixels, which are black in Figure 5.9, and '0' is for the other pixels (white)."Colored" means the value of the point is 2. Of course, there is no point with value 2 at the beginning, because the points are either 0 or 1. C in Figure 5.10 shows the values of Figure 5.9(a).

$$C = \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&[0]&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

Figure 5.10. Remove Redundant Segments (1).

The searching order is from left to right, and up to down direction.

First, expand the lines. It means change the 8-neighbours of the line points from 0 to 1. Set the new matrix as D, shown in Figure 5.11. Why is that? Because we only concern about the values of the line points' 8-neighborhoods. So the neighbor points should be identified at first.

$$D=\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&[1]&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&1&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

Figure 5.11. Remove Redundant Segments (2).

Among the line's neighborhood points, try to find the first uncolored background pixel in C. That means $C(x,y)=0$ and $D(x,y)=1$. Because the searching order is from left to right, and up to down direction, then the first such pixel must be a left upper most background pixel. Color it into 2. In the example, this pixel is (4,4), which is bracketed in Figure 5.10 and Figure 5.11.

Color the background pixels in the same area (outer or inner background) with point (4,4). It means, scan the pixels on its right side and on the same row with it until we find a pixel which cannot satisfy the following conditions: $C(x,y)=0 \,\&\, D(x,y)=1$. The coloring direction is shown in Figure 5.12. Then color all these pixels into 2. Thus D becomes the matrix shown in Figure 5.13.

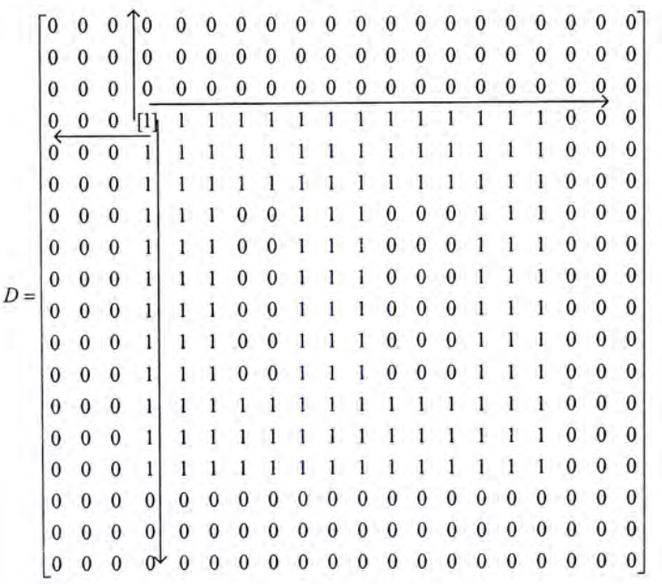


Figure 5.12. Remove Redundant Segments (3).

Then find the next non-line and non-colored pixel (x,y). In addition, it should satisfy that at least one of its 4-neighbours is colored. Thus we can make sure that such pixel is also in the same outer or inner background area as the previous colored pixels. Such pixel is (5,17) which is bracketed shown in Figure 5.13.

$$
D=\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&2&2&2&2&2&2&2&2&2&2&2&2&2&2&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&[1]&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&1&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}
$$

Figure 5.13. Remove Redundant Segments (4).

$$
D=\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&2&2&2&2&2&2&2&2&2&2&2&2&2&2&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&2&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&2&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&2&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&2&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&2&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&2&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&2&0&0&0\\
0&0&0&2&1&1&0&0&1&1&1&0&0&0&1&1&2&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&2&0&0&0\\
0&0&0&2&1&1&1&1&1&1&1&1&1&1&1&1&2&0&0&0\\
0&0&0&2&2&2&2&2&2&2&2&2&2&2&2&2&2&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}
$$

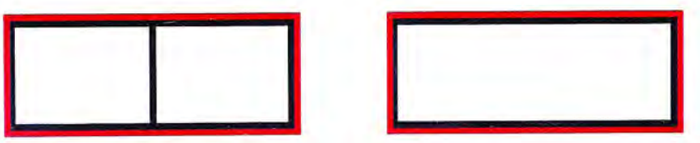
Figure 5.14. Remove Redundant Segments (5).

The same coloring process is applied to this point (5,17). Then D is colored as shown in Figure 5.14.

Now there is no point is neighbored with the colored points and satisfied the condition $C(x,y)=0 \,\&\, D(x,y)=1$. The coloring process terminates here.

The redundant points can be identified now. Observe the line points, which are 1 in C (Figure 5.10). The colored image is shown in Figure 5.15. Red parts are the colored pixels. We can tell that the pixels around the segments we want are both red and white (white for background pixels). But the pixels around the redundant segments are all white. Thus we can identify the redundant pixels very easily.

The result of coloring process for Figure 5.9(b) and (c) is also shown in Figure 5.16(a).



(a) (b)

Figure 5.15. Result of coloring process.

This method can identify the redundant lines cleverly. The neighbors of the inner segments (Figure 5.9 a & b) are all 0. The neighbors of the connecting segments (Figure 5.9c) are 2. And the neighbors of the pixels on the lines that we really want are 1 and 2. Also, this method is very fast, because only the neighbors of the line points are considered. Thus we can save time effectively.

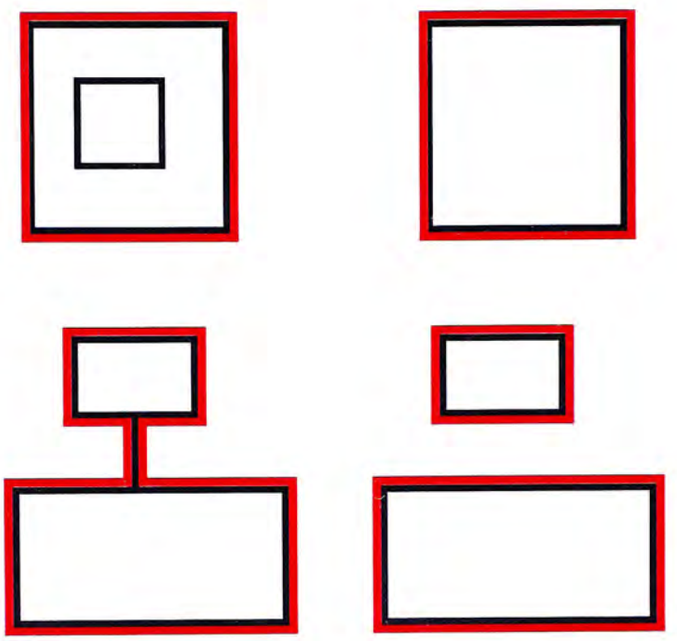


Figure 5.16. More results of coloring process.

Chapter 6

Post processing

The post processing is mainly adjusting the lines detected from the previous steps. After all the steps, the signboards are detected. But some times the borders are not so accurate (Figure 6.1). In order for a better performance, we apply such a post processing to adjust the position of the lines.

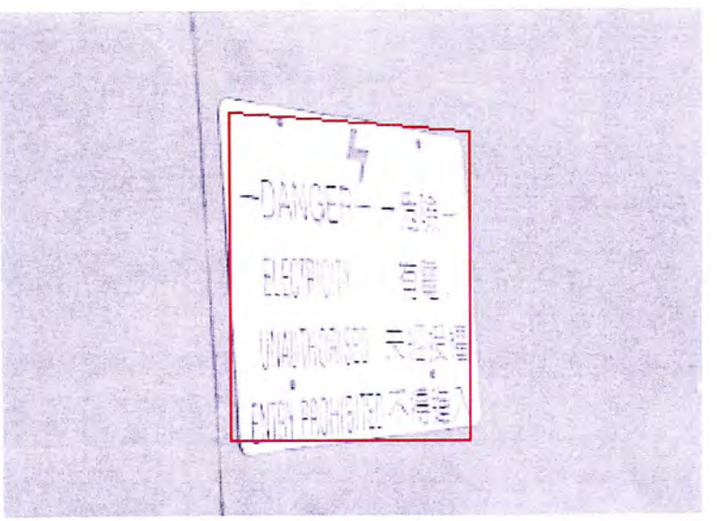


Figure 6.1. Signboards detected without post processing.

At first, let's find the factors that cause such errors.

1. Gradient error

2. Quantization error

3. Errors caused by merging close points

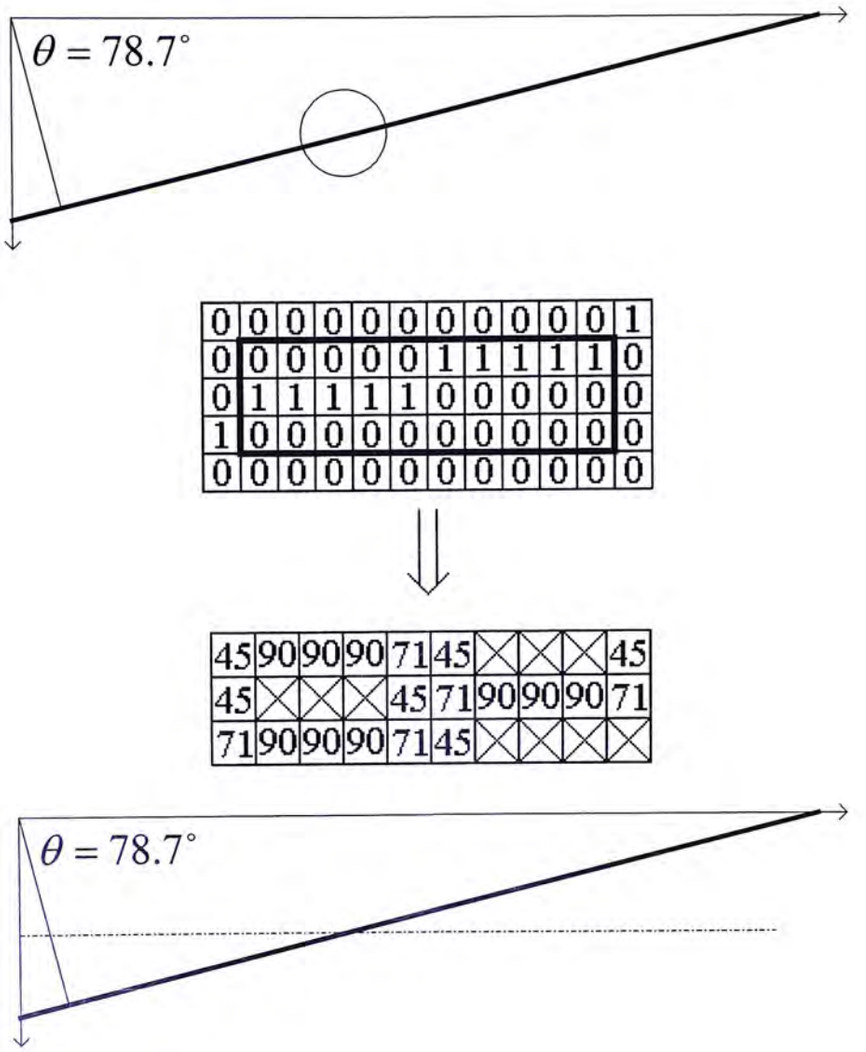


Figure 6.2. Gradient error.

In section 4.2.1 we introduced the calculation of gradient. That's is the angle θ between the line's normal and x-axis. The gradient is the result of convolution with the operator shown in Figure 4.4. The operator's size is 3×3. So it cannot reflect the gradient's changes outside that range. Let's look at an example shown in Figure 6.2.

The angle θ should be 78.7° for the line in Figure 6.2. But after the calculation of gradient discussed in chapter 4, the result is quite different. There are many angles: $45^\circ, 71^\circ$ and 90°. Since 90° appeared most frequently, it may probably contribute a peak in the accumulator array. Then the line's gradient will at last be resulted in 90°. The error is $90^\circ - 78.7^\circ = 11.3^\circ$. Then the result is the dotted line in Figure 6.2.

One of the reasons is due to the quantization error. In the gradient based Hough transform, the angle θ and distance d are quantized to the parameters qt and qd.

The other reason is due to the merging of close points. This is at the beginning of finding close circuit, discussed in 5.2 . The signboards are detected in the condition that the boundaries can make a close circuit. But because of the noise, some part of the boundaries cannot be detected by Hough transform. The merged point is the middle of two close points. Thus the error of the point's locations is caused.

So we use the least-square error method to adjust these segments. What should be the input of this process? We use the points in the edge map. The coordinates (x_i, y_i) of the edge points are known.

Then find the points that corresponding the segments we found. Assume the accurate line should be

$$y = kx + b \qquad (6.1)$$

k and b are unknown.

The error of the line to each edge point is:

$$R_i = kx + b - y \qquad (6.2)$$

The square error of the line to the whole points is:

$$\varphi(k,b) = \sum R_i^2 = \sum (kx_i + b - y_i)^2 \qquad (6.3)$$

If we want the square error to be as small as it can, then $\varphi(k,b)$ should satisfy

$$\begin{aligned} \frac{\partial \varphi}{\partial k} = 0 \Rightarrow \sum 2x_i(kx_i + b - y_i) = 0 \\ \frac{\partial \varphi}{\partial b} = 0 \Rightarrow \sum 2(kx_i + b - y_i) = 0 \end{aligned} \qquad (6.4)$$

From equation (6.4), the values of k and b can be calculated. Then the line decided by these two parameters is the line that most appropriately fit the edge points.

Result of post processing is shown in Figure 6.3. Compare it with Figure 6.1, the signboard's boundaries are much more accurate.

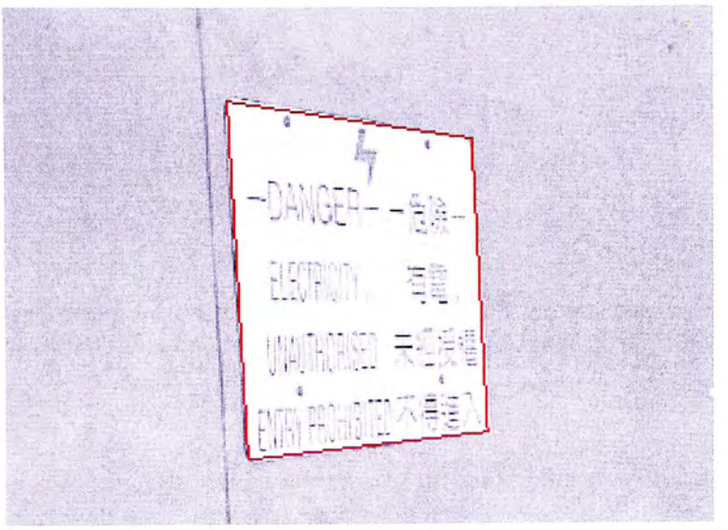


Figure 6.3. Result of post processing.

Chapter 7

Experiments and Conclusion

In the previous chapters, we have introduced our system to detect signboards in an image. The system consists of four parts: pre-processing, gradient-based Hough transform, signboards locating, and post-processing. This chapter presents the experimental results, discusses some problems in the system, and gives the conclusion finally.

7.1 Experimental Results

In the experiment, we have tested the system on 104 images. Figure 7.1, Figure 7.2, Figure 7.3 and Figure 7.4 show some of the results.

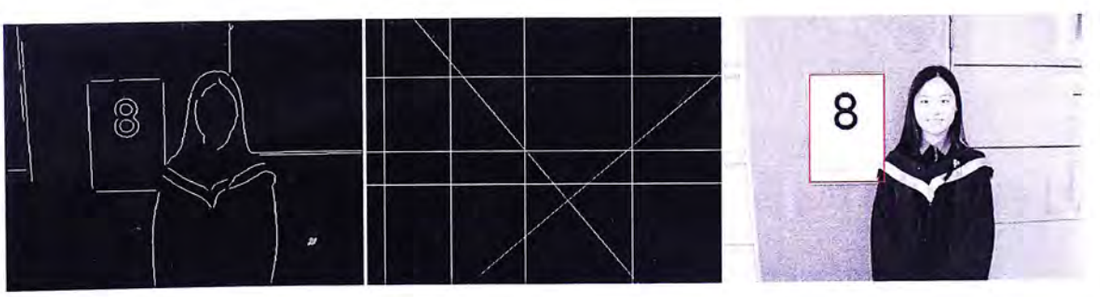


Figure 7.1. Experimental Result.

Left: Result of pre-processing;

Middle: Result of gradient-based Hough transform;

Right: Detected signboard superimposed on the original image.

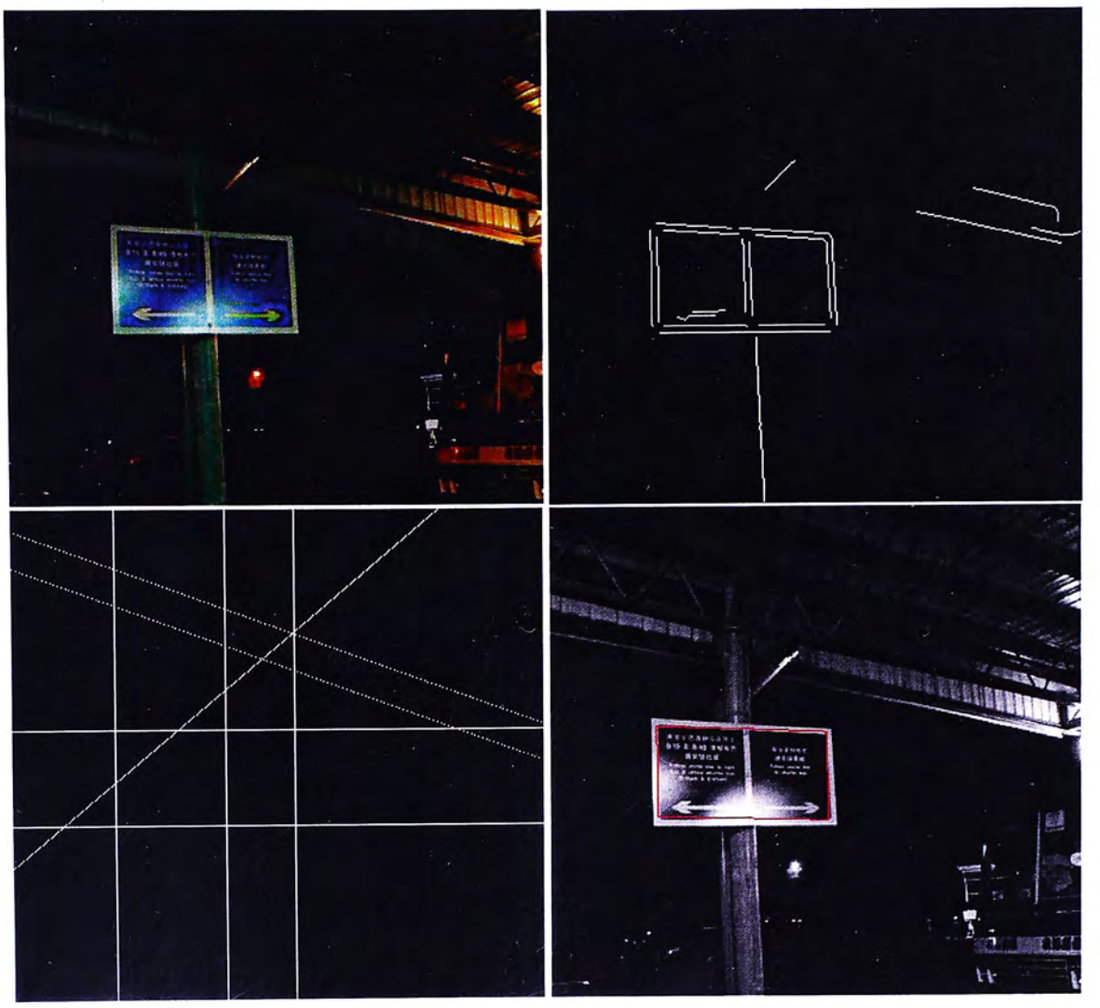


Figure 7.2. Experimental Result.

Left top: Original Image; Right top: Result of pre-processing;

Left bottom: Result of gradient-based Hough transform;

Right bottom: Detected signboard superimposed on the original image..

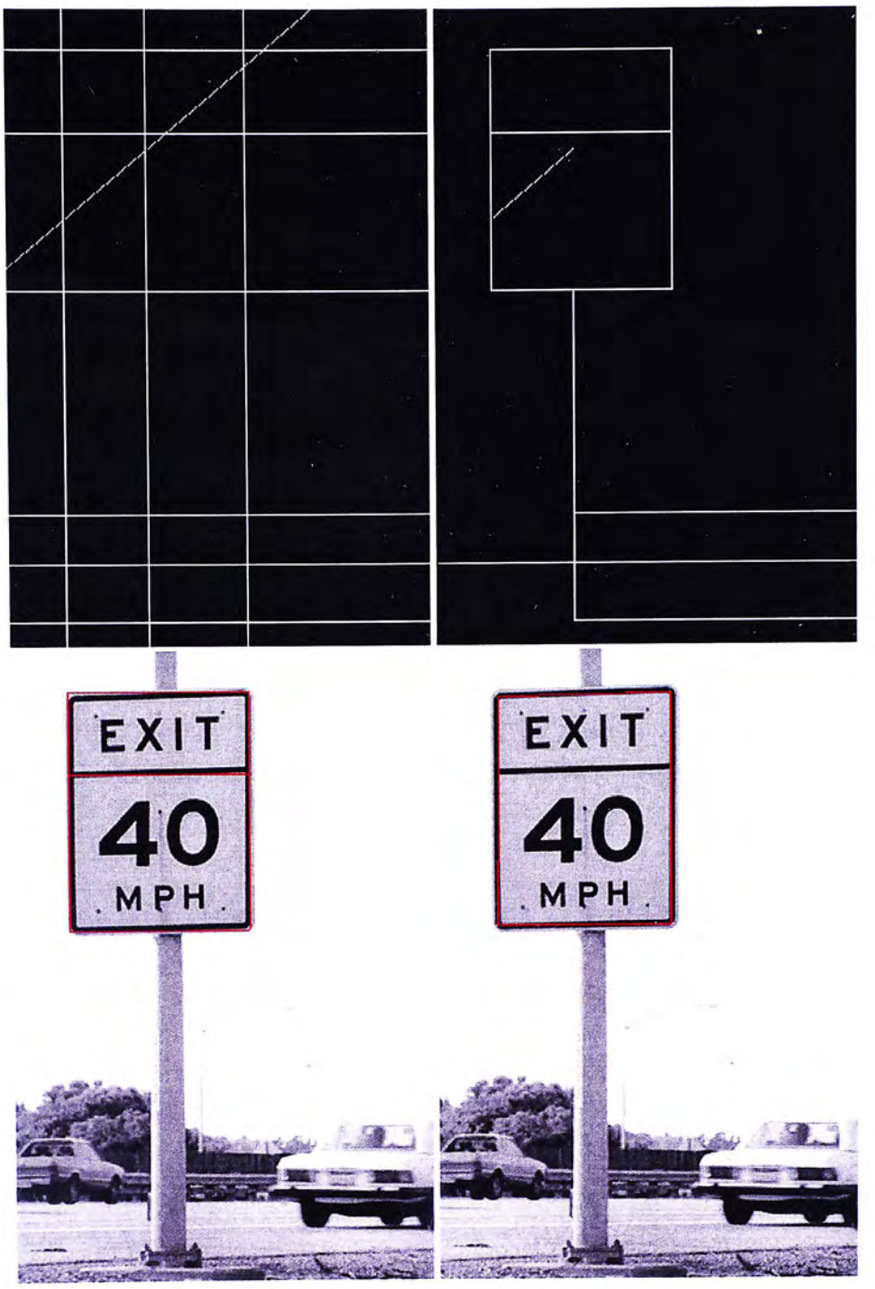


Figure 7.3. Experimental Result.

Left top: Result of gradient-based Hough transform;

Right top: Result after density checking;

Left bottom: Result of finding close circuit;

Right bottom: Detected signboard superimposed on the original image..

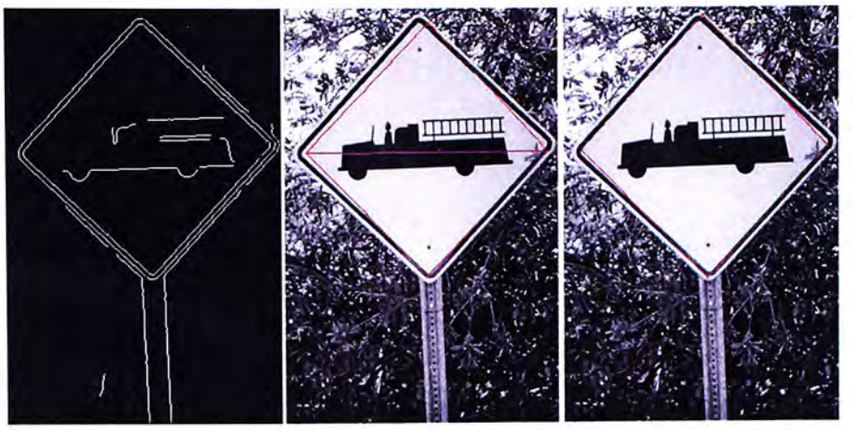


Figure 7.4. Experimental Result.

Left: Result after pre-processing;

Middle: Result of density checking;

Right: Detected signboard superimposed on the original image..

Table 7.1. Experiment Data.

Total images	Success	Fail	Success Rate
104	97	7	93.3%

The experimental data are listed in Table 7.1. From the experimental results and data, we see that the system performs very well on more than 90 percent of the images. The average time consumed by the algorithm using Matlab is about 5 seconds per image of 300×400 pixels, on a 1.8G Pentium IV PC. Below we will first discuss the reasons that cause the system failure on some images, and then present some special cases.

1. The boundaries cannot be extracted in the edge detection due to the poor image quality. Especially when the other part of the image has

great gradient changes, but the boundary doesn't have such a gradient change. The system is sensitive to edges. If it failed to find the correct edges, it will fail at the very beginning.

2. Gradient error can cause the system fail to detect the signboards. It is a key factor in gradient-based Hough transform. If the gradient error is too big, the line's parameters (d,θ) cannot contribute a peak in the accumulator. Thus we cannot extract the signboard's boundaries.

3. The boundaries of the signboard are not complete. For example the signboard is behind of a large object. Of course, if the object is not very large, the system can make the boundaries complete automatically. But since such a threshold is used (a threshold to determine how close the two points are considered as they're connected), sometimes the boundary cannot be found correctly.

In the following, some of the advantages of the system are listed.

1. The system is robust to noise.

In the pre-processing step, noise is removed by detecting corners in the edge map. Noise is mainly caused by the background.

Figure 7.5. Result when the noise is not ignorable.

2. A big problem of using Hough transform is that it's quite time-consuming. But our method can overcome it successfully. It is due to the following methods we use. First, in pre-processing, the corner detection is applied to delete the small segments that are not belonging to the boundaries of the sign. Second, we use the gradient information of the edge points. Third, we quantized d and θ appropriately.

3. In addition, the algorithm can also detect the sign that is even covered partly, shown in Figure 7.6. Of course, the covered part should not

be too larger. Otherwise, the algorithm will deem it that there doesn't exist such a segment.

Figure 7.6. Result when the signboard is partly covered.

7.2 Conclusion

In the work, we have developed a system to detect generic signboards in images. The signs can be any polygons, not limited to rectangles or triangles. In this method, we first apply an edge detection and corner detection in pre-processing step. Then the gradient-based Hough transform is used to find straight segments. After that, signboards are detected by checking the point density of each segment, finding a closed circuit and removing redundant lines. Precise signboard positions are obtained further by line fitting. Our method can also detect signboards when their boundaries are overlapped or the background is very complicated. Promising results have been obtained. The experiments show that the system gives a correct rate of 93.3%.

For future work, we may apply more information on the image. For example, hue in images and temporal information in videos. Because we ignored such information in the system, which may make the method more generalized. But adding more information will probably improve the system's result.

Bibliography

[1] Ernest L. Hall, "Computer Image Processing and Recognition", Academic Press, Inc.

[2] Robert M. Haralick, Linda G. Shapiro., "Computer and robot vision", Addison-Wesley Publishing Company.

[3] Amit, Yali, "2D object detection and recognition: models, algorithms, and networks".

[4] "Proceedings of the Second IEEE Workshop on Applications of Computer Vision", Sarasota, Florida, 1994.

[5] Eun Ryung Lee, Pyeoung Kee Kim, and Hang Joon Kim, "Automatic Recognition of A Car License Plate Using Color Image Processing".

[6] N. Kehtarnavaz and A. Ahmad, "Traffic Sign Recognition in Noisy Outdoor Scenes".

[7] A. Gruen, E.P. Baltsavias, O. Henricsson, "Automatic extraction of man-made objects from aerial and space images (II)" , Birkhauser Verlag, 1997.

[8] Lucas Paletta, "Detection of Traffic Signs Using Posterior Classifier

Combination", Pattern Recognition, 2002. Proceedings. 16th International Conference on , Volume: 2 , 11-15 Aug. 2002 Page(s): 705 -708 vol.2.

[9] Arturo de la Escalera , Miguel Angel Salichs, "Road Traffice Sign Detection and Classification", *IEEE transactions on Industrial Electronics*, Vol. 44, No. 6., December 1997.

[10] Sunghoon Kim, Daechul Kim, Younbok Ryu, Gyeonghwan Kim, "A Robust License-Plate Extraction Method under Complex Image Conditions", Pattern Recognition, 2002. Proceedings. 16th International Conference on , Volume: 3 , 11-15 Aug. 2002 Page(s): 216 -219 vol.3.

[11] Paolo Comelli, Palo Ferragina, Mario Notturno Granieri, and Flavio Stabile, "Optical Recognition of Motor Vehicle License Plates", *IEEE transactions on Vehicular Technology*, Vol. 44, No. 4, November 1995.

[12] Ming G. He, Alan L. Harvey, Thurai Vinay, "Hough Transform In Car Number Plate Skew Detection", ISSPA, 1996.

[13] Yuji Aoyagi and Toshiyuki Asakura, "A Study on Traffic Sign Recognition in Scene Inage Using Genetic Algorithms and Neural Networks."

[14] Dong-Su Kim, Sung-Il Chien, "Automatic Car License Plate Extraction Using Modified Generalized Symmetry Transform and Image Warping", *ISIE 2001*, Pusan, KOREA.

[15] M. Blancard, "Road sign recognition: a study of vision based decision making for road environment recognition", Proceedings of Vision-Based Vehicle Guidance Roundtable, pp.1-9, 1990.

[16] N. Kehtarnavaz, N. Griswold, D. Kang, "Stop-sign recognition based on color and shape processing", Machine Vision and Applications, vol. 6, pp. 206-208, 1993.

[17] D. Kang, N. Griswold, N. Kehtarnavaz, "An invariant traffic sign recognition system based on sequential color processing and geometrical tranformation", Proceedings of IEEE Southwest Symposium on Image Analysis, pp. 88-93, 1994.

[18] S. Estable, J. Schick, F. Stein, R. Janssen, R. Ott, W. Ritter, Y-J. Zheng, "A real-time traffice sign recognition system," Proceedings of Intelligent Vehicles Symposium, pp. 213-218, 1994.

[19] L. Pacheco, J. Batlle, X. Cufi, "A new approach to real time traffic sign recognition based on color information," Proceedings of Intelligent Vehicles Symposium, pp. 339-344, 1994.

[20] Heikki Kalviainen, Petri Hirvonen, Lei Xu and Erkki Oja, "Probabilistic and Non-probabilistic Hough Transforms: Overview and Comparisons", *Image and Vision Computing* Volume 13 Number 4 May 1995.

[21] C. Galambos, J. Kittler and J. Matas, "Gradient Based Progressive Probabilistic Hough Transform", IEEE Proc. – Vis. Image Signal Process., Vol. 148, No. 3, June 2001.

[22] J. Song, M. Cai, M.R.Lyu, S. Cai, "A New Approach for Line Recognition in Large-size Images Using Hough Transform", *ICPR 2002*.

[23] Emanuele Trucco, Alessandro Verri, *Introductory Techniques For 3-D Computer Vision*, p82-85, Prentice Hall.

[24] Hough, P. V. C., "*A Method and Means for Recognizing Complex Patterns*," U.S. Patent No. 3,069,654, 1962.

[25] Robert J. Schalkoff, "Digital Image Processing and Computer Vision".

[26] Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing", P416-423, P197-201.

[27] Kenneth R. Castleman "Digital Image Processing", Prentice-Hall

International, Inc. P465-466.

[28] Milan Sonka, Vaclav Hlavac, Roger Boyle, "Image Processing, Analysis, and Machine Vision", Second Edition.

[29] Emanuele Trucco, Alessandro Verri, *Introductory Techniques For 3-D Computer Vision*, p82-85, Prentice Hall.

[30] P.V.C. Hough, Machine Analysis of Bubble Chamber Pictures, International Conference on High Energy Accelerators and Instrumentation, CERN, 1959.

[31] http://www.cs.unc.edu/~lee/class/comp290_75/HoughTransform/HoughTransform.html

[32] Robert M. Haralick, Linda G. Shapiro, Computer And Robot Vision, Addision-Wesley Publishing Company, 1992, P578-584.

[33] William Ford, William Topp, *Data Structures With C++*, p748-751, Prentice Hall, 1996.

[34] Angela Tam, Hua Shen, Jianzhuang Liu, Xiaoou Tang, "Quadrilateral Signboard Detection and Text Extraction", International Conference on Imaging Science, Systems, and Technology 2003.