Institut für Logik,
Komplexität und Deduktionssysteme
Lehrstuhl: Prof. Dr. Waibel

Optical Recognition of Chinese Characters

Diploma Thesis

May 2002

by Andreas Hanemann

advisers: Xilin Chen, Jie Yang

Abstract

The aim of this thesis was to develop a system for the recognition of Chinese characters. The chosen character set (GB2312-80 standard) contains 3,755 characters which cover nearly all of the characters in daily use in China. The system should especially be useful for character images recorded from street signs and consider special problems caused by noises. Starting from a system for clear data, the problems which occur with noisy data were cataloged into several problem classes and special solutions for these problems were developed. For training the character set was printed and captured by a camera. The recognition experiments of samples which were recorded under different lighting conditions and of samples recorded by another camera yielded accuracies of 99.95%. For the recognition of the characters from street signs an accuracy of 85.32% was obtained which can be improved to 85.93% when using a priori probabilities. At the end, the recognizer which uses binary features was successfully combined with another recognizer which is based on a feature extraction using Gabor filtering.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter consists of two parts. The first section describes *Optical Character Recognition (OCR)* in general, its history, problems and common applications. The second section gives some background information about Chinese characters. Their history, simplification, usage in different countries, character classes, and common fonts are presented in this section.

1.1 Optical Character Recognition

The task of OCR is to recognize characters from a given character set which are contained in an input image. In many cases, a person can recognize the characters easily, even when the image contains noises or the characters appear in uncommon fonts. It is obvious that in the human memory the characters have to be stored in an abstract way. But for computers it is more difficult since the way to learn and recognize the characters has to be programmed, trying to approximate the way humans do the recognition.

OCR a is subdiscipline of Document Image Analysis (DIA). The task of DIA is to extract also the metadata from an image like the font, font size, format, italics and boldface.

1.1.1 History of OCR

In this subsection, some background information about the history of Optical Character Recognition is provided [31, 22].

In 1809, the first patents for reading devices for the blind were awarded, but were forgotten afterwards.

The first machine that was able to read characters and transform them into standard telegraph code was invented in Chicago by Emmanuel Goldberg in 1912. At the same time, Edmund Fourier D'Albe was inventing machines to help the blind. This device called "Optophone" was a handheld scanner which emitted tones when moving across a printed page. Each tone should correspond to a character. Even though the device was useable, a long training time and concentration when hearing the tones was required from the users. For this reason, the device never experienced widespread use.

The mentioned and other research efforts showed that the optical reading of printed documents and their transformation into electrical signals, especially for telegraphy, was possible.

In 1931, Goldberg got a patent for a device which used his idea of template matching. He compared an unknown character with a given "search plate" containing a negative image of the reference. If they would match, a light source would be completely blocked from a photographic cell.

After the previous systems did not get widespread public attention, the machine "Gismo", invented by David Shepard in 1951, became known to a broader public. The machine was able to read 23 characters of the alphabet written by a standard typewriter. Shortly afterwards, Shepard founded the Intelligent Machines Research Corporation (IMR) and received a patent for his invention. This was the first attempt to commercialize an OCR system.

At the same time, Jacob Rabinow invented a system that was able to recognize alphanumeric characters at the speed of one character per minute.

In 1954, Reader's Digest in New York State became the first commercial customer of an OCR system when it used an IMR device to convert written documents into punched cards for further processing by a computer. Another important customer for the OCR systems was the US Postal Service which introduced the devices for mail sorting beginning in 1959.

In the 1960s, many companies started to develop OCR equipment that could then also deal with different character fonts and input paper sizes. Recognition Equipment Incorporated based in Dallas, Texas became the leading OCR company. It started with 5 employees in 1961 and employed more than 1,000 seven years later. To make the recognition easier, the American National Standards Institute specified the highly stylised font "USASI-A" for use in OCR which substantially differs from a standard typewriter font.

In the 1970s, there were three main applications for OCR: character recognition of alphanumerics, recognition of barcodes as known from supermarkets, and optical mark readers which scan for the presence or absence of black points at certain locations.

1.1.2 Difficulty of an OCR Task

When characterizing the difficulty of an OCR task, the following criteria have to be taken into account:

printed/hand printed: When characters are printed by a machine, they do not contain the variances which are introduced by printing the characters by hand.

fonts: The more fonts the system shall be able to recognize, the more difficult is the task, especially when the system shall also be able to deal with unknown fonts.

size: When the characters are always in a given size, the system does not have to scale the characters. Otherwise, a scaling procedure has to be used which can lead to distortions.

number of characters: The more characters the system shall be able to recognize, the more difficult is the task. But the number should not be taken as a single indicator, because it is also important how similar the characters are.

noises: The noises which make the task more difficult can be stains/smears or breaks/gaps and a non-constant background.

segmented/unsegmented: If the system has to deal with unsegmented characters the system has to perform an explicit or implicit segmentation which is another source of errors.

1.1.3 Common OCR Problems

Some problems of OCR systems for recognizing printed characters are common and were cataloged in a recent research study from the University of Nevada [29].

imaging defects: These defects which are caused by character printing can be smeared or broken characters, stray marks or light print. Light print, for example, can thin some strokes of a character or the whole character which may cause that the defected parts are treated as noise.

curved base lines: When scanning a page from a book, the binding of the book can cause curved base lines in the scanned image. Many OCR devices have problems with such base lines as they can only deal with rotated straight base lines.

similar symbols: Some characters like l, I, and 1 are sometimes hard to distinguish. To solve this problem, the context of the characters can be used.

punctuation: Since a dot and a comma are quite small, they can be confused with each other or can be treated as noise. Using a language model can help in these cases.

overlapping characters: When italics are used in a text, the segmentation becomes more difficult. One way to deal with this problem is to detect the italics and transform them in a way that they do not overlap anymore.

1.1.4 Application of OCR

In this subsection, some examples for the usage of OCR systems are presented.

postal service: OCR devices are used for the sorting of letters. In this recognition task, the redundancy between the ZIP code and the city name can be used to improve the recognition accuracy. A letter is sorted by hand, if the OCR device rejects the letter.

forms: In forms the information has to be entered at given locations which allows a highly accurate recognition. The automated recognition can be used for tax forms or for checks.

video images: Another application can be to extract character sequences which are contained in video images [2]. The aim is to extract subtitles which can then be used for indexing and searching in a digital video library.

1.2 Chinese Characters

This section contains information about Chinese characters in general, their history, character simplifications, character use in neighboring countries, the six classes of Chinese characters, strokes, and different fonts.

1.2.1 History of Chinese Characters

The origin of the Chinese characters dates back to ca. 2000 BC. The Chinese characters are called *Hanzi* (Han = China, zi = character) and have survived to the present day with several thousands of characters in daily use. The large number of characters is caused by the fact that each of the Hanzi stands for a certain meaning and a sound, typically a syllable.

Development of Different Writing Styles

The earliest characters were inscribed in animal bones and turtle shells and were used to forecast the future. Characters were also found on bronze vessels and tripods and described sacrifices or laws. In the Han dynasty (206 BC - 220 AD) the writing technique changed from using a bamboo pen to an animal hair brush. With the hair brush it was possible to write with a changing thickness which changed the appearance of the characters (Li Shu style).

In ca. 80 AD the writing style changed to the Kai Shu style which until today is the appearance used for handwriting. It is also used for printing since Bi Sheng invented printing with movable (clay) types in 1040 (an invention that was also made by Gutenberg 400 years later).

Character Simplification

In the handwriting of Chinese characters there have been for a long time attempts to simplify the writing, but the printing remained unchanged. From 1956 to 1964 the People's Republic of China (PRC) simplified more than 2000 characters. The actual number of simplifications can be measured in different ways, because sometimes the appearance of a radical (see subsection 1.2.3) was changed in all characters where it is part of.

In December 1977, some PRC's newspapers announced a further change of 550 characters, but this change was withdrawn half a year later. Even though the simplifications for the printing were cancelled, some of the simplifications were used in handwriting and lettering.

In May 1981, the Chinese government divided the characters into frequent used and seldom used characters (code GB 2312-80, "Code of Chinese Graphic Character Set for Information Interchange"). The frequent used character set contains 6,763 characters and the seldom used more than 100,000. The frequent used character set is also divided into two classes, again according to the frequency of use. The first class contains 3,755 characters and is ordered by the pronunciation. It covers ca. 80% - 85% of the characters in daily use. The second class contains 3,008 characters and is ordered by the shape and complexity. Both classes cover more than 99% of the characters in daily use.

An examination of the character frequencies of a 1,808,114 character corpus showed that the characters are quite unevenly distributed. It contained 4,574 different characters. The most frequent character appeared 75,306 times, 425 characters were found only once [42].

Another examination [36] was performed for Chinese characters from Internet newsgroups in 1993/1994 (see figure 1.1). The 1994 corpus contains 162,611,044 characters from 13,053 classes. 160,296,689 (98.58%) of these characters belong to the class 1 of the GB 2312-80 encoding, while 2,368,707 (1.46%) do not belong to the character set.

The modern Chinese dictionaries list between 7,000 and more than 50,000 characters. For these dictionaries, no common indexing system exists.

In Japan 320 characters were simplified after World War II. Singapore changed some printed characters in 1973 and again in 1979, the last time to make them conform which the changes in the PRC. In Taiwan, Hong Kong and Macao as well as in Europe and America the traditional characters are still the standard.

1.2.2 Use of Chinese Characters in Korea, Japan and Vietnam

The neighbor countries of China which had not developed an own writing system before they got into contact with the Chinese characters adopted them in different ways [1].

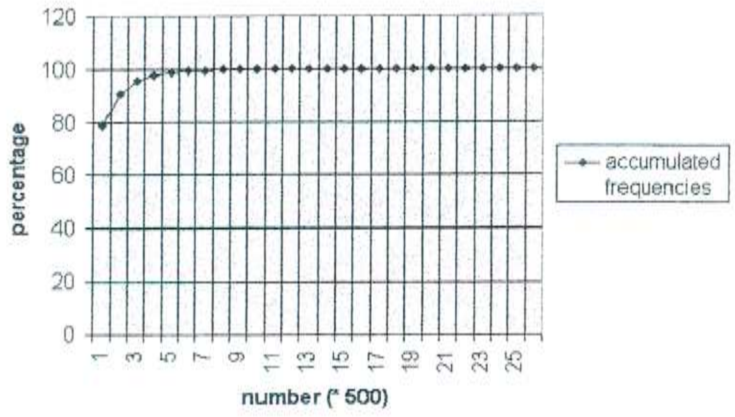


Figure 1.1: Frequencies of Chinese characters

In Korea, the Hanzi were used starting from the 7th century by neglecting their meaning and only taking into account their pronunciation. As there were problems to find characters for special Korean syllables, an alphabetic writing system called *Hangul* was developed in 1443.

Japan received the first Chinese characters in the 3rd century AD. As the Japanese had no own writing system, they used the characters and pronounced them in a way similar to the Chinese pronunciation. Later on, another way of pronouncing the characters was developed. The Japanese developed two character sets, *Hiragana* and *Katakana* which contain 50 characters each. These sets define syllables which can be used to describe the pronunciation of the Chinese characters.

Starting from the 8th century AD, characters called *Chu Nom* which look like Chinese characters, but are not, were developed in Vietnam. They were used for Vietnamese words that were embedded in a Chinese text. In the 17th century European missionaries developed a writing system based on the Latin alphabet which has been widely used since and caused the Chu Nom characters to die out in the middle of the 20th century.

1.2.3 Six Classes of Chinese Characters

The Hanzi can be divided in six classes called *Liu Shu* (six [kinds of] writing).

The first class consists of *pictograms* (Xiangxing Zi) which are simplified pictures of a real world object. Examples of such characters are "sun", "moon", "tree/wood", and "woman".

Abstract concepts are represented in the second and third class of characters. These characters are called *ideograms*. A distinction is made between simple ideograms (second class, Zhi Shi Zi) and compound ideograms (third class, Hui Yi Zi) which contain several characters and form a new meaning from these. Simple ideograms can be the characters for numbers as well as the characters for "top/on/above" and "bottom/below". Examples of compound characters are a character meaning "bright" consisting of the sun and the moon pictograms and a character which contains the characters "woman" and "child" and has the meaning "good, love".

The fourth class (Jia Jie Zi) contains characters for particles and verbs that can not be written as pictograms or ideograms. In these cases, a phonetic borrowing took place, when using characters with similar pronunciation.

Because the fourth class introduced disambiguities, a fifth class called *phono-ideograms* (Xingsheng Zi) evolved. These characters consist of a meaning component which gives a hint to the semantic of the character and a pronunciation component as in the forth class. The components of these characters are called *radicals.*

Over time, this fifth class became most popular for creating new characters. In the time of the Shang dynasty (16.- 11. century BC) 34% of the characters belonged to the fifth category, while 97% of the characters from the Kangxi Zidian dictionary (1716 AD, more than 47,000 characters) are part of the fifth class.

The last class (Zhuanzhu Zi) contains only very few examples and is often neglected.

1.2.4 Strokes

Chinese characters can be as simple as just one horizontal stroke or can have more than 36 strokes. Figure 1.2 shows the stroke distribution of Chinese characters in the database [36].

The 2,965 most frequent characters from that database had an average stroke number of 9.1, while the 1,253 and 733 most frequent had averages of 8.91 and 8.65.

1.2.5 Fonts of Chinese Characters

The variety of fonts used for the Chinese characters is not as large as the variety of fonts used for Western character sets. There are four major kinds of fonts which are explained further on (see figure 1.3 for examples).

Songti: This style is the most common font for body text and headlines. The horizontal strokes carry a triangular ornament in their top right corner and are thinner than vertical strokes. This font is called *Mincho* in Japan.

Fangsongti: This font is used for subtitles and sometimes for body text. It is not so strictly geometrical as the Songti font and the horizontal strokes are slightly slanting upwards.

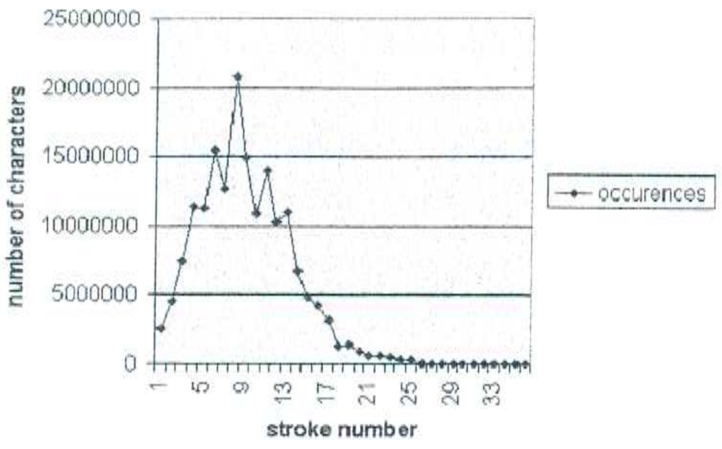


Figure 1.2: Stroke distribution of Chinese characters

Heiti: The Heiti font is mostly used for headlines or specially marked positions of text. The horizontal and vertical strokes are of equal thickness and contain no serifs.

Kaiti: This style is often used in prefaces, subheads, and footnotes. It is also sometimes used for names inside a Songti font body. It is derived from the Kai Shu handwriting and is the most lively of the four fonts.

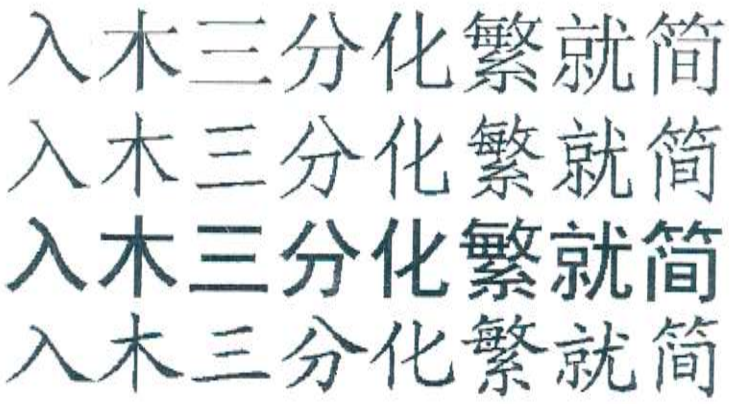


Figure 1.3: Examples of Chinese characters in Songti, Fangsongti, Heiti, and Kaiti font (from top to bottom)

Chapter 2

Preprocessing and Basic Techniques

For the further feature extraction and classification it is important to eliminate some noises and normalize the characters. This chapter explains some common methods to perform these steps which are image to image transformations. It also contains descriptions of other basic techniques in OCR.

2.1 Size Normalization

The aim of size normalization is to scale the size of a given image to a standard size. This section presents several methods to perform the scaling for binary or grayscale images.

2.1.1 Single Character Normalization

If the input image contains a single character, two methods how to scale the image to a given width w_0 and height h_0 are possible. The first is just to scale the image in both directions. This method ignores to relationship between width and height which can lead to strange effects. These effects can especially be seen, when width and height are very different. Figure 2.1 shows that a character that contains only one stroke is transformed to a block.

The second method conserves the relationship between the width and height. A maximum scaling factor is calculated as

$$\min\left(\frac{w_0}{width}, \frac{h_0}{height}\right) \tag{2.1}$$

and used for the scaling. The image then fits at least to one of w_0 and h_0. If it does not fit to i.e. the height, the remaining space is filled up with the background color with the same number of lines at top and bottom. This method eliminates

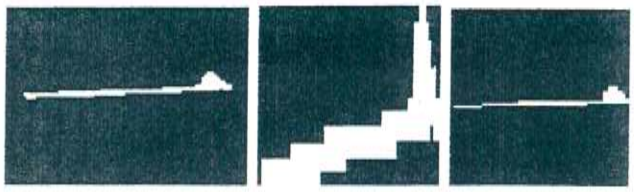


Figure 2.1: Original image (left), ignoring the relationship between width and height (middle), keeping the relationship between width and height (right)

the problem of the first method, but can lead to another difficulty if the input character is stretched in one direction. Since the relationship between the width and the height is conserved the transformed image will stay stretched which does not happen when using the first method (see figure 2.2).

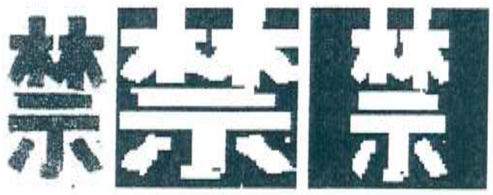


Figure 2.2: Original image (left), ignoring the relationship between width and height (middle), keeping the relationship between width and height (right)

The problems of both methods should be kept in mind when extracting a single character from a character sequence.

2.1.2 Bilinear Interpolation

The *bilinear interpolation* method is a simple method for scaling a grayscale image. Let $s(i,j)$ be the original image, $t(i,j)$ the standardized image.

$$t(i,j) = \sum_{l=0}^{1}\sum_{k=0}^{1} g_{lk}(i,j)s(i_s+l, j_s+k) \tag{2.2}$$

$$i_s^* = \frac{i}{k_x} \tag{2.3}$$

$$j_s^* = \frac{j}{k_y} \tag{2.4}$$

$$i_s = int(i_s^*) \tag{2.5}$$

$$j_s = int(j_s^*) \tag{2.6}$$

k_x and k_y are the stretching factors in both directions. They are used to find a coordinate pair (i_s^*, j_s^*) and its integer values (i_s, j_s) where the transformed point is located in the original image.

$$\begin{aligned} d_i &= i_s^* - i_s && (2.7)\\ d_j &= j_s^* - j_s && (2.8)\\ g_{00}(i,j) &= (1-d_i)(1-d_j) && (2.9)\\ g_{01}(i,j) &= d_i(1-d_j) && (2.10)\\ g_{10}(i,j) &= (1-d_i)d_j && (2.11)\\ g_{11}(i,j) &= d_i d_j && (2.12) \end{aligned}$$

The neighborhood of the point (i_s^*, j_s^*) is used to calculate the new value $t(i,j)$. The four points of the neighborhood are weighted according to their distance to the estimated point (i_s^*, j_s^*).

2.1.3 Cubic Spline Interpolation

Better results for scaling of grayscale images can be achieved by *cubic spline interpolation*. The disadvantage of this method is that it takes more time to perform the transformation.

Let $s(i,j)$ be the original image of size $(M \times N)$ and $t(i,j)$ the standardized image. The scaling factor k_x in row direction is given by

$$C = k_x N \qquad (2.13)$$

To convert one sequence $\underline{s}(i)$ into a sequence $\underline{\tilde{s}}(i)$ the *Hermitic spline functions* are used [30].

$$\tilde{s}(i,\mu) = \underline{\alpha}_\mu \underline{Q} \underline{c}_\mu(i) \qquad i = 1, ..., M; \mu = 1, ..., C \qquad (2.14)$$

with

$$\begin{aligned} \alpha_\mu &= \left(\frac{\mu}{k_x} - [\frac{\mu}{k_x}]\right) && (2.15)\\ \underline{\alpha}_\mu &= [\alpha_\mu^3, \alpha_\mu^2, \alpha_\mu, 1] && (2.16)\\ \underline{Q} &= \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0 \end{pmatrix} && (2.17)\\ \underline{c}_\mu(i) &= [s(i,\mu-1), s(i,\mu), s(i,\mu+1), s(i,\mu+2)]^T && (2.18) \end{aligned}$$

The columns of the intermediate matrix $\tilde{s}$ (dimension $M \times C$)

$$\underline{\tilde{s}}(j) = [\tilde{s}(1,j)...\tilde{s}(M,j)]^T, j = 1, ..., C \qquad (2.19)$$

are transformed in an analogous way to columns of the $(R \times C)$ standardized matrix $t(i,j)$ with scaling factor k_y.

$$\begin{aligned} R &= k_y M & (2.20)\\ \underline{t}(j) &= [t(1,j), ..., t(R,j)]^T, j = 1, ..., C & (2.21)\\ t(\mu, j) &= \underline{\tilde{\alpha}}_\mu \underline{Q} \tilde{c}_\mu(j) & (2.22) \end{aligned}$$

This method can also be used for binary images. After the transformation such an image has to be rebinarized using the middle value between black and white as threshold (see section 2.2).

2.2 Binarization

Binarization is the process to convert a grayscale image into a black/white image. A binary image can be used in the further feature calculation. An advantage of a binary image is that the needed storage space is reduced. A disadvantage of the binarization is that information is lost which can affect the further recognition. The aim of the binarization methods is to lose as few information as possible.

2.2.1 Histogram Method

For the binarization of an image $s(i,j)$ a threshold can be used for separation of the pixels in black or white for the whole image. One method for finding such a threshold value [24] is provided here.

At first, a *histogram* which describes the relative frequencies of the grayscale values is calculated.

$$p_s(g) = n_g/G \qquad g = 0, ..., (G-1) \tag{2.23}$$

with
n_g: number of pixels with graylevel g
G: total number of pixels

The division normalizes the sum of the p_s to 1. Some further definitions are made:

$$\begin{aligned} h_s(g) &= \sum_{k=0}^{g} p_s(k) & (2.24)\\ m_s &= \sum_{k=0}^{G-1} k p_s(k) & (2.25)\\ m_s(g) &= \sum_{k=0}^{g} k p_s(k) & (2.26) \end{aligned}$$

Further on, the maximum weighted square deviation of mean values $m_s(g)$ in dependence of the gray level g is found.

$$f(g^*) = \max_g \left(\frac{h_s(g)}{1 - h_s(g)} [m_s(g) - m_s]^2 \right) \tag{2.27}$$

The binarization threshold θ_s can then be determined as

$$\theta_s = g^* - 1 \tag{2.28}$$

To improve the binarization, a combination $u(i,j)$ of appropriate low pass and high pass filters (s_T, s_H) can be used for the grayscale image.

$$u(i,j) = [\alpha s_T(i,j) + (1-\alpha) s_H(i,j)] \qquad 0 \le \alpha \le 1 \tag{2.29}$$

$$s_T(i,j) = \frac{1}{T}[s(i-1,j) + s(i+1,j) + s(i,j) + s(i,j-1) + s(i,j+1)] \tag{2.30}$$

$$s_H(i,j) = s(i,j) - s_T(i,j) \tag{2.31}$$

This method is used in the presented system as shown in the example in figure 2.3. The chosen threshold value is displayed as a black line in the histogram. Also see other examples in figures 2.5, 2.6, 2.7.

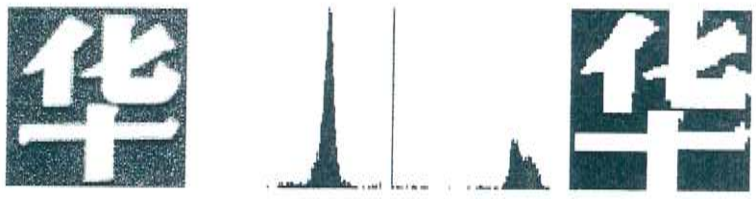


Figure 2.3: Original image (left), histogram (middle), black/white image (right)

2.2.2 Iterative Selection

Another method called *iterative selection* [25] can be used to find a threshold value T in an iterative procedure. At first, the values T_f and T_b which are an estimation for the average of the foreground and background mean values are initialized according to a heuristic. Then T is defined as $\frac{T_f+T_b}{2}$. The heuristic for choosing T_f and T_b can use some knowledge about the image. It is also possible just to define T as the mean of all gray values in the image. Then the iteration starts in which T_f and T_b are redefined as the mean of all pixels above T or below T. Now T is redefined as the mean of T_f and T_b. The iteration stops if the new value for T is the same as the old value. According to Parker this simple method returns good threshold values for many applications and is easy to implement.

2.2.3 Correlation Method

This method [25] calculates the correlations between the original image and images for all possible thresholds. It then returns the threshold which received the maximum correlation. The correlation is calculated as

$$r = \frac{\sum_{i=0}^{N-1}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{N-1}(x_i - \bar{x})^2 \sum_{i=0}^{N-1}(y_i - \bar{y})}} \tag{2.32}$$

with
x: original gray image
$\bar{x}$: mean value of x
y: binarized image
$\bar{y}$: mean value of y

2.2.4 Adaptive Threshold Selection

All three methods which have been presented in the previous subsections use a threshold value for the whole image. But for cases where the background is not the same everywhere this can cause problems. For such cases *adaptive threshold selection* [26] can be used.

Assuming that the original of the given image was as a black/white image, an unknown probability P that a point was initially black can be described as

$$P(x|N(x)) \equiv \text{probability(point x is black|values of neighbors x)} \tag{2.33}$$

For a direction D and a point c the values a and b are defined as

$$a = v(P_c) - v(P_a(D)) \tag{2.34}$$
$$b = v(P_b(D)) - v(P_c) \tag{2.35}$$

where
$v(P_c)$: center value of a point
$v(P_a(D)), v(P_b(D))$: lowest and highest values along direction D
The distance $f(d)$ along a direction is defined as:

$$f(d) = ||a(d)| - |b(d)|| \tag{2.36}$$

For four directions (horizontal, vertical, both diagonal) the direction D which maximizes $f(d)$ is found. The values a and b can be interpretated in the following way. If a and b are small, then the center point is located on a plateau. If a is bigger than b, the intensity function is convex, otherwise the intensity function is concave.

For the implementation a copy $u(p)$ of the image is necessary. The original is denoted as $v(p)$. There are two main procedures.

Procedure A

For all points do:

1. Find that D where $f(D)$ is the maximum.
2. Compute a and b.
3. If a and b are small in absolute value or equal to each other, set $u(p_c) = v(p_c)$.
   Else do:
4. If a is greater than b, then set $u(p_c) = white$.
5. Else if b is greater than a, then set $u(p_c) = black$.
6. For all points set $v(p_c) = u(p_c)$ (or use, instead, the copy as the original for the next iteration).

The procedure A pushes points with a large variation and a high curvature to the goal values (black or white).

Procedure B

For all points do:

1. If a point is already black or white, set $u(p_c) = v(p_c)$.
2. Find the direction D where $f(D)$ is maximum.
3. Compute a and b.
4. If both a and b are small in absolute values or equal to each other, set $u(p_c) = v(p_c)$.
5. If $a + b$ equals $white - black$, then
   (a) If $|a| > |b|$, then set $u(p_c) = black$.
   (b) Else set $u(p_c) = white$.
6. Else
   (a) If $|a|$ is greater than $|b|$, then set $u(p_c) = v(p_a(D))$.
   (b) Else if $|a|$ is less than $|b|$, then set $u(p_c) = v(p_b(D))$.
7. For all points set $v(p_c) = u(p_c)$ (or use, instead, the copy as the original for the next iteration).

Step 5 pushes a point that is between black and white to the nearest neighbor. The procedure B can be used several times to convert every point to black or white. A modified version of the procedure where $v(p_c)$ is used instead of $u(p_c)$ accelerates the domino-effect.

Figure 2.4 shows the effect of procedures A and B on the grayscale distribution along a direction.

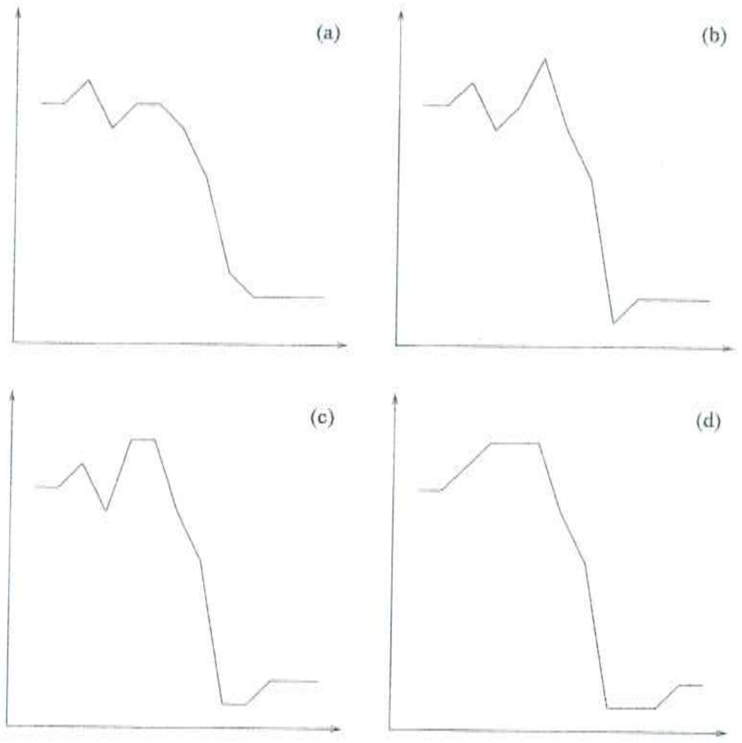


Figure 2.4: Original image (a), after procedure A (b), after procedure B (c), after repeated execution of B (d)

This method has been implemented in the current system and proved to be useful for images with a changing background. If i.e. the background is dark at the top of the image and is light at the bottom, it does not make sense to use a single gray value as a threshold. The examples in figures 2.5, 2.6, 2.7 show the better performance of the adaptive method in comparison with the histogram method.

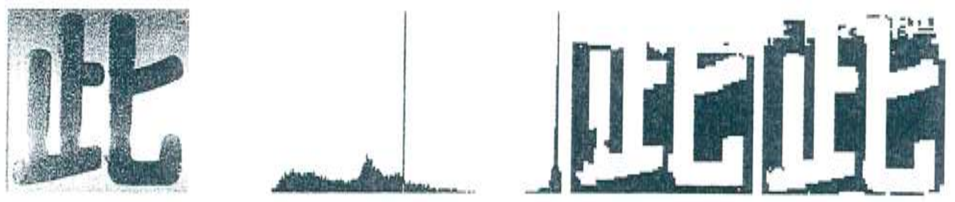


Figure 2.5: Original image, histogram, binary image generated by histogram method, binary image generated by adaptive method (from left to right)

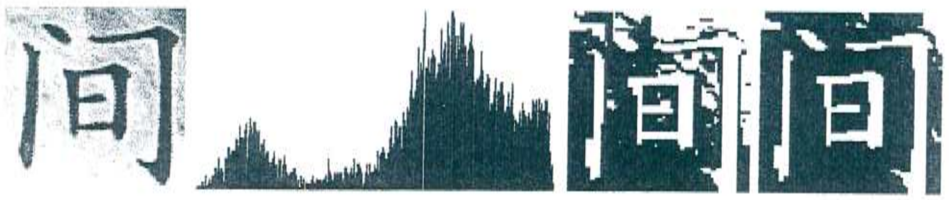


Figure 2.6: Original image, histogram, binary image generated by histogram method, binary image generated by adaptive method (from left to right)

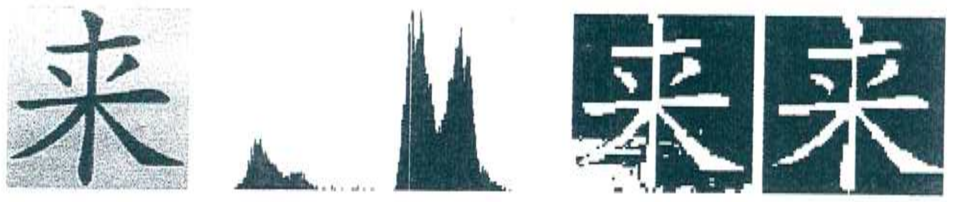


Figure 2.7: Original image, histogram, binary image generated by histogram method, binary image generated by adaptive method (from left to right)

2.3 Smoothing

Smoothing techniques are useful to eliminate disturbance in single pixels and in small areas. The methods can be used to eliminate isolated blobs, fill gaps, and to smooth the edges of lines.

2.3.1 Smoothing Techniques for Gray-level Images

One technique to eliminate random noises in a gray-level image is to average the values of each pixel using its neighbors. The following masks can be used for that.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

The resulting values have to be multiplied by $\frac{1}{9}$, $\frac{1}{10}$ or $\frac{1}{16}$. The last two matrices have the advantage that they do not blur the image as much as the first matrix. The blurring effect can be further reduced by not changing the value of the pixel $f(r,c)$ when is it located at an edge.

$$x(r,c) = \begin{cases} \frac{\sum_{i=r-1}^{r+1} \sum_{j=c-1}^{c+1} (f(i,j)-f(r,c))}{8} & if \left| \frac{\sum_{i=r-1}^{r+1} \sum_{j=c-1}^{c+1} (f(i,j)-f(r,c))}{8} \right| < threshold \\ f(r,c) & otherwise \end{cases} \tag{2.37}$$

Instead of using the mean of the neighbors, it is also possible to use the median. Here the values of the neighboring region are sorted and the middle value is applied to the current pixel.

2.3.2 Erosion and Dilation

The erosion and dilation (see figures 2.8, 2.9) operations can be used for black/white images.

Erosion can be used to eliminate distortions at the edge of the strokes and to break unwanted links. Every black pixel that has contact with a white pixel is removed.

Dilation is an operation to fill gaps. Here every background pixel that has contact with a foreground pixel is changed.

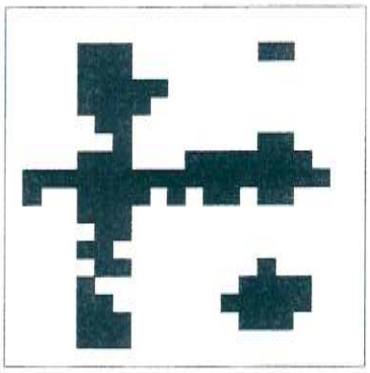


Figure 2.8: Original image before dilation and erosion

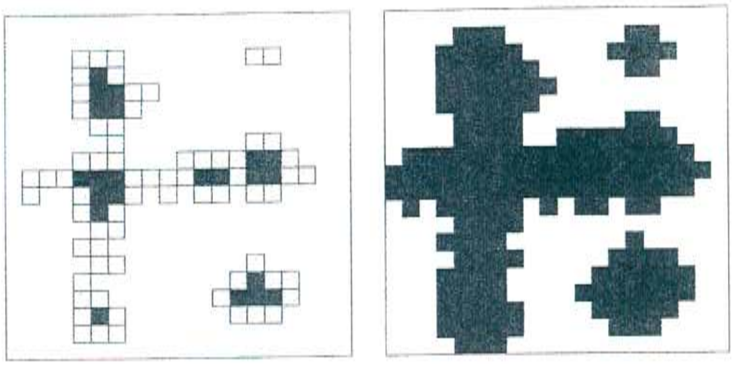


Figure 2.9: Erosion (left) and dilation (right) performed on the original image

The operations opening and closing (see figure 2.10) are composed of an erosion and a dilation. *Opening* performs an erosion first and then a dilation. The effect is that it smoothes contours, breaks narrow isthmuses, eliminates small blobs and sharps peaks or capes. The operation is anti-extensive which means that no black pixels will be added in the operation while some can be removed.

Closing performs the operations in the other order. The effect is that a smoothing of the contour, a fusion of narrow breaks and long thin gulfs, elimination of small holes, and filling of gaps on the contours is performed. This operation is extensive which means that it may add foreground pixels while not adding background pixels.

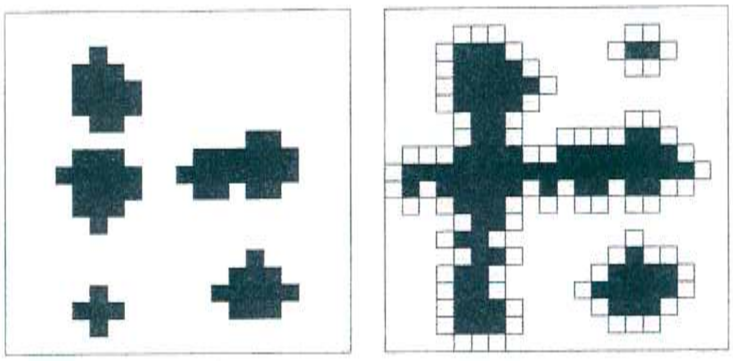


Figure 2.10: Opening (left) and closing (right) performed on the original image

Both opening and closing are idempotent. This means that i. e. if an opening operation has already been performed, the application of another opening operation will have no effect.

2.3.3 Edge Enhancement

For some features it is important to smooth the edges of the characters to make the features more stable. The smoothing of the horizontal and vertical strokes can be done in the following way which demonstrates the improvement of upper edges of horizontal strokes. Let $b(i,j)$ be a binarized image with the values 1 for black pixels and 0 for white pixels. The edge pixels E are defined as

$$E = \{b(i,j) | b(i-1,j) = 0 \wedge b(i+1,j) = 1\} \tag{2.38}$$

In each line i sequences which consist of points belonging to E are detected.

$$P_i(I_1, I_2) = \{b(i,j) | b(i,j) \in E, I_1 \leq j \leq I_2\} \tag{2.39}$$

The sequence P_i can be divided into subsequences consisting of pixels with the same values. Consequently, the neighboring subsequences must have different values. A criterion useful for the detection of defective edges is:

$$(N_S > 3) \vee \left((N_S > 1) \wedge min\left[\frac{W}{W+B}, \frac{B}{W+B}\right] < \theta\right) \tag{2.40}$$

where
N_S: number of subsequences
W: number of white pixels
B: number of black pixels
θ: threshold

The two figures 2.11, 2.12 show the effect of the edge enhancement procedure in the proposed system.

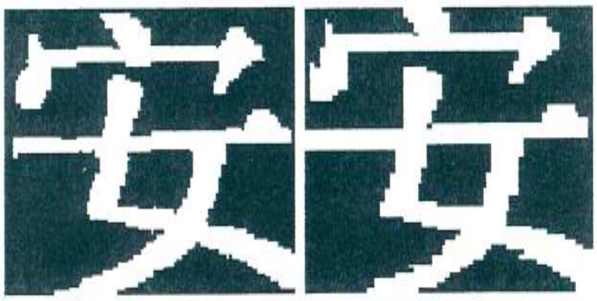


Figure 2.11: Original image (left) and after edge enhancement (right)

2.4 Thinning

The aim of *thinning* is to normalize the character image to a uniform stroke width of one pixel. The thinning can i.e. be used for the feature extraction of the TBC features or as basis of stroke extraction methods.

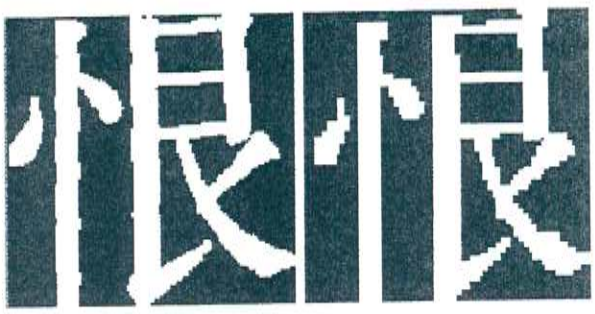


Figure 2.12: Original image (left) and after edge enhancement (right)

From the hundreds of methods for thinning which have been proposed three methods to perform the thinning are presented in this section. The first method is the classical thinning method which uses information about neighbor pixels to perform the thinning. The removal of points is done layer by layer. The second method constructs a line adjacency graph using a horizontal run-length encoding of the input image. The last method constructs a cross section sequence graph for the thinning.

Thinning has some general problems. One common error is that a cross point (4-fork point) is splitted into two branch points (3-fork points). Distortions in junction areas are also common. Sometimes spurious branches can be added in the thinning process.

2.4.1 Hilditch Method

The classical thinning method by Hilditch [12] was aimed at reaching four goals:

thinness: The thinned line's width must be one pixel.

position: The thinned line must lie along the center of the original line.

connectivity: The thinned line must keep the connectivity of the original line.

stability: At each step of the thinning process the thinned line cannot be eroded away from its end points.

For the *Hilditch algorithm* an image with black foreground pixels with the value 1 and white background pixels with the value 0 is given. The region of the black points is denoted as Q. Some conditions which are presented later are checked to mark potentially deletable points with the value -1. The marked region is called R. The *connectivity* N_c^8 is defined as the number of components which are connected to the current pixel.

The conditions for the removal of a point p are the following:

condition 1: Point p belongs to Q.

condition 2: Point p lies on the edge of Q; that is

$$\sum_{k \in S_1} \overline{x}_k \geq 1 \tag{2.41}$$

holds, where x_k are the binary values of the eight neighbor pixels and $S_1 = 1, 3, 5, 7$.

condition 3: Point p has more than one neighbor which means that p is not the tip of a thin line.

$$\sum_{k=1}^{8} x_k \geq 2 \tag{2.42}$$

condition 4: The connectivity is not changed by p's removal. N_c^8 stays 1.

condition 5: The removal of point p does not remove an isolated point. This condition has the aim not to change the number of components in the image.

condition 6: Point p's removal in conjunction with any one of its neighbors that has been removed does not alter the connectivity of Q; that is

$$x_i \neq -1 \vee N_c^8(i) = 1 \qquad i = 3, 5 \tag{2.43}$$

The figure 2.13 shows how the thinning procedure works in general, while the figure 2.14 reveals a specific problem of this procedure. A pattern which looks like a V is converted into a pattern which has the appearance of a Y. This means that an extra branch has been constructed by the algorithm.

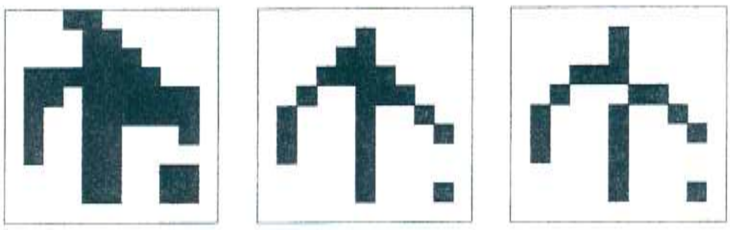


Figure 2.13: Thinning example

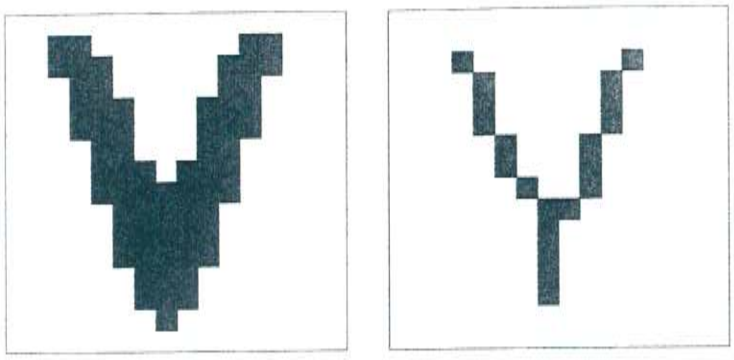


Figure 2.14: VY problem of the Hilditch method

2.4.2 Thinning Using a Line Adjacency Graph

Another method for thinning which was proposed by Pavlidis [27] uses a *line adjacency graph (LAG)*. The binarized image is run-length encoded in the horizontal lines. Assuming that a line always starts with a white pixel the encoding for the line consists of lengths for white and black sequences. The LAG is constructed from the sequences. Figure 2.15 shows the input for the LAG construction, for which figure 2.16 contains the constructed LAG.

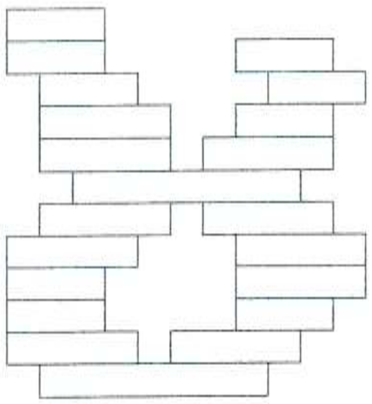


Figure 2.15: Input for the LAG construction

The next figure (figure 2.17) shows a case where a cross point has been splitted into two branch points. The dashed lines show how the LAG should look like.

The proposed method categorizes a black line segment either as a path or as a junction and constructs a compression of the LAG (see figure 2.18).

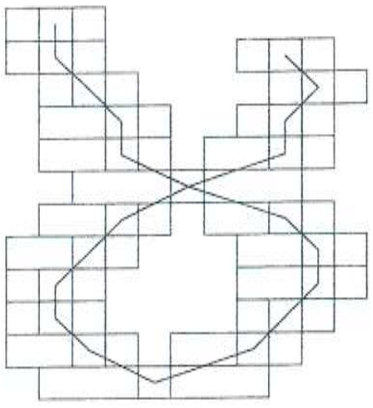


Figure 2.16: Constructed LAG in an ideal case

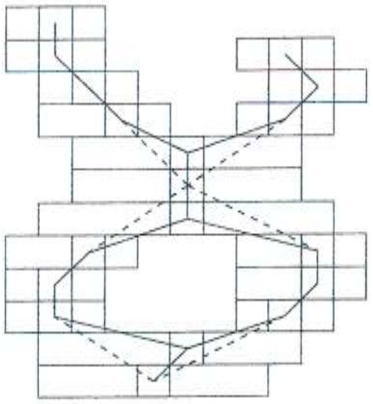


Figure 2.17: Splitting of a cross point into two branch points

According to Mori this method yields better results than the Hilditch method, even though it is not stable against changes in the stroke width.

2.4.3 Thinning Using a Cross Section Sequence Graph

The *cross section sequence graph (CSSG)* was proposed by Suzuki and Mori [34, 21]. It divides the character into regular and singular regions. The regular regions are supposed to be stable and the idea is to reconstruct the singular regions using the regular regions.

The algorithm starts with a smoothing procedure which is different from common smoothing procedures (see section 2.3). The smoothing here is performed along the boundaries of the character and smoothes the angles of a sequence of

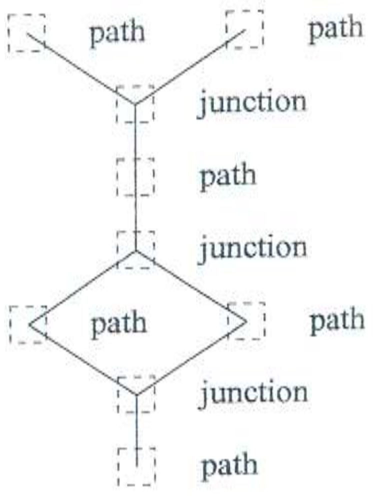


Figure 2.18: Compressed LAG with path and junction nodes

points.

For the thinning procedure several definitions are made:

opposite boundary point: The opposite point b of a boundary point a is found by traversing the graph until reaching the opposite side. Point b is denoted as $opp(a)$.

successor (predecessor) of a boundary point: The points of the contour are labelled $p_1,...,p_n$. The point p_{i+1} is called successor of p_i and the point p_{i-1} is called predecessor of p_i. They are denoted as $p_{i+1} = succ(p_i)$ and $p_{i-1} = pred(p_i)$.

cross section: For an ordered pair (p, q) of boundary points, $C(p, q)$ is defined as a cross section if there exists no white pixel on the path between p and q. A boundary point can be a member of more than one cross section which is important for regions with high curvature. For a cross section $C(p, q)$ $C'(p'q')$ is the next cross section, if $p' \in \{p, succ(p)\}$ and $q' \in \{q, pred(q)\}$ and $C \neq C'$. It is denoted as $C' = next(C)$. In an analogue way the preceding cross section is defined.

cross section sequence: A list $L = C_0, C_1, ..., C_{k-1}$ of cross sections is called a cross section sequence, if for any pair C_i, C_{i+1} $(i = 0, 1, ..., k-2)$ the relationship $C_{i+1} = succ(C_i)$ and the start and end conditions ($next(C_{k-1}) \in \{\phi, C_0\}$ and $pred(C_0) \in \{\phi, C_{k-1}\}$) are satisfied. C_0 is defined as head of L ($C_0 = head(L)$) and C_{k-1} is defined as tail of L ($C_{k-1} = tail(L)$).

singular region: A singular region is defined as a black connected component which is surrounded by heads, tails, and boundary points that are not included in any cross section sequence.

The thinning itself consists of five steps.

boundary extraction: The boundary of the character is extracted using a non-sequential boundary processing technique [20].

cross section sequence construction: Perpendicular cross sections are found, each taken almost perpendicular to the direction of the line. Let $cs(a, b)$ $(a, b \in P)$ be the logical function to find a cross section (see figure 2.19).

$$cs(a,b) = \begin{cases} 1 & \text{if } ((a = opp(b)) \wedge (D(opp(a), b) < Th1)) \vee \\ & \quad ((b = opp(a)) \wedge (D(opp(b), a) < Th1)) \\ 0 & \text{otherwise} \end{cases} \tag{2.44}$$

where
$D(p, q)$: number of the boundary points between p and q (without p and q)
$Th1$: threshold parameter, usually set to 2

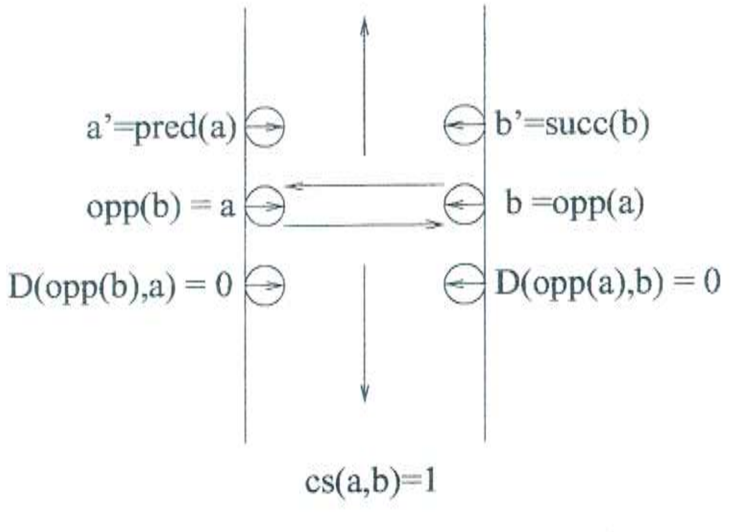


Figure 2.19: Cross section construction

If a pair (a, b) with $cs(a, b) = 1$ is found, the cross section $C(a, b)$ is constructed.

A label for boundary points is introduced to indicate to which cross section a point p belongs.

$$LBL(p) = \begin{cases} \geq 0 & \text{if it belongs to the LBL(p)-th cross section} \\ < 0 & \text{if it belongs to no cross section} \end{cases} \tag{2.45}$$

For all boundary points the initial value is set to -1 and then the found cross sections are used to change the values of the member points. Afterwards, some rules are defined for the concatenation of cross sections to cross section sequences.

cross section sequence extension: Since the character image usually contains noises at the edges, this step is performed to extend the cross sections into the noisy areas [34].

singular region extraction: Let $L_k(k = k_1, ..., k_n, n > 0)$ be the cross section sequences that are connected to a singular region S. The vector tangent V_k is constructed for each of the L_k representing their directions. W_k is defined as the vector $(x - x_k, y - y_k)$, where (x, y) is a point in the singular region and (x_k, y_k) are the coordinates of the center of the head/tail of L_k. The value F for each point (x, y) is defined as

$$F = \frac{1}{n} \sum_{k=k_1}^{k_n} \frac{(V_k, W_k)}{|V_k||W_k|} = \frac{1}{n} \sum_{k=k_1}^{k_n} \cos \phi_k \tag{2.46}$$

The singular point is obtained as maximum of all points.

skeletonization: After the other steps have been performed, the extraction of the skeleton is quite simple as the singular regions have been examined before.

For each cross section the center points are added to the skeleton. For a singular region the number of connections to regular regions are calculated. If the number is zero, the singular region is assumed to be noise. Otherwise, the singular point of the region is used as a vertex and is connected to the regular regions (see figure 2.20).

The method is supposed to be faster than the standard method and to retrieve graphs with fewer distortions [21].

2.5 Contour

The contour describes the boundary of a given character.

A simple method to define the boundary is that every character pixel which shares an edge with the background is a contour pixel.

Lee and Wu [18] use a more complex definition. Here, n_i is the i-th neighbor of a pixel in chain encoding (see below).

interior point: If $(n_1 \wedge n_3 \wedge n_5 \wedge n_7) = 1$ and $(n_2 \vee n_4 \vee n_6 \vee n_8) = 1$, then this pixel is defined as an interior point.

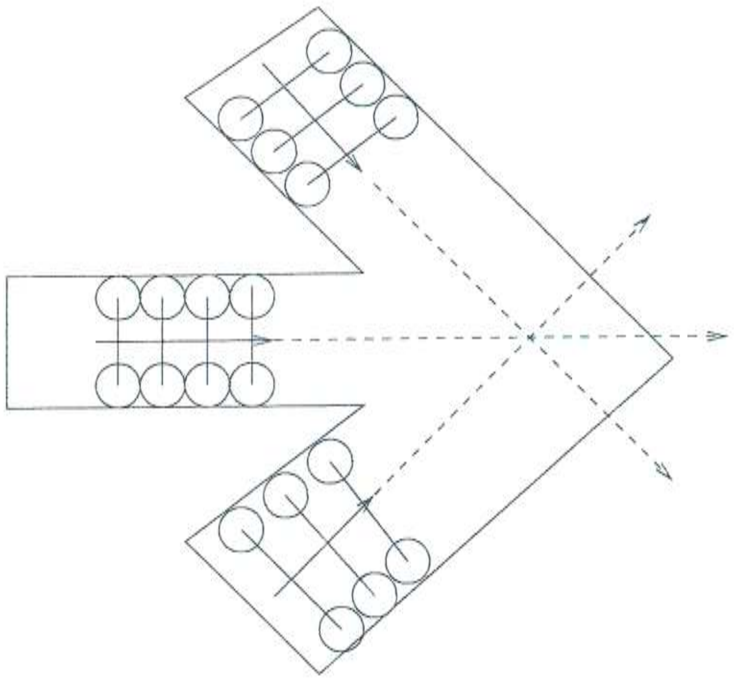


Figure 2.20: Skeleton construction for a singular region

noise point: When $(n_{i+1} \wedge n_{i+2} \wedge ... \wedge n_{((i+m)mod8)+1})$ is 1 $(0 \leq i \leq 7, m \geq 1)$, then $(n_{i+1}, n_{i+2}, ..., n_{((i+m)mod8)+1})$ is defined as a set of consecutive points. If the neighborhood satisfies one of the following two criteria, then the pixel is defined as a noise point.

1. There are not any consecutive points as shown in the first two examples of the noise points in figure 2.21.
2. The number of sets of consecutive points equals two and there does not exist a simple cycle as shown in the last example of a noise point in figure 2.21.

contour point: A contour point is defined indirectly as neither an interior point nor a noise point.

The *chain code* [5] which is also called *Freeman code* after its inventor assigns the numbers 1 to 8 to different directions as shown in the following matrix.

$$\begin{pmatrix} 8 & 1 & 2 \\ 7 & & 3 \\ 6 & 5 & 4 \end{pmatrix} \tag{2.47}$$

The definition of the directions is used for the TBC and WDH features.

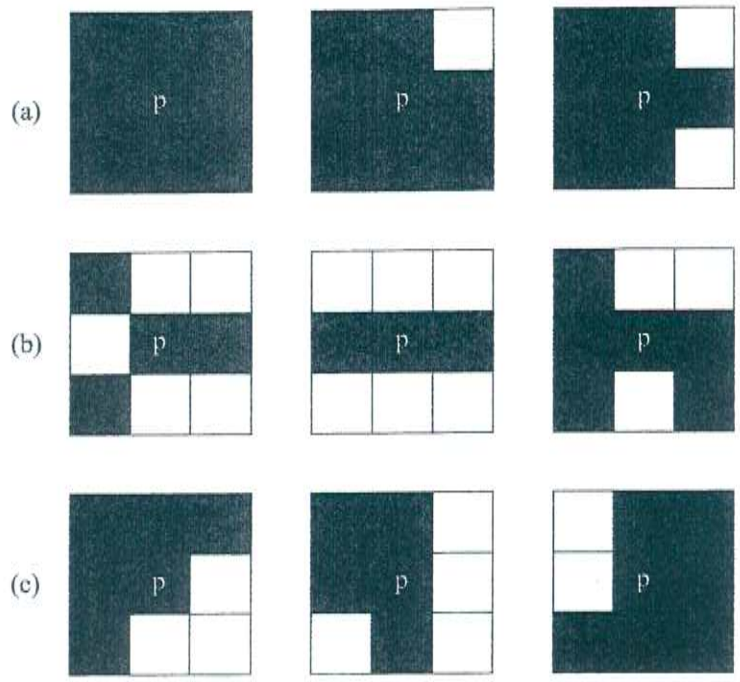


Figure 2.21: Examples of interior (a), noise (b) and contour points (c)

2.6 Hough Transformation

The *Hough transformation* [25] can be used to detect lines in a given image. Every pixel in the input defines a family of lines passing through that pixel.

$$Y = mX + b \tag{2.48}$$

This equation can be transformed to

$$b = -Xm + Y \tag{2.49}$$

which can be interpreted as the definition of a line in the (m, b) space. Using this transformation for every point which is part of the foreground the image is transformed to the (m, b) space. Points in the (m, b) space where lines intersect correspond to pixels which are located on the same line in the original image. If a point in the (m, b) space contains many intersections, it represents a line in the original image.

The basic idea of the Hough transformation is to detect those points. But the proposed transformation does not work properly for vertical lines, because these lines have an infinite slope. This problem can be solved using the normal form for a straight line.

$$x \cos \omega + y \sin \omega = r \tag{2.50}$$

where

$$\sin\omega = \frac{-1}{\sqrt{m^2+1}} \tag{2.51}$$

$$\cos\omega = \frac{m}{\sqrt{m^2+1}} \tag{2.52}$$

$$r = \frac{-b}{\sqrt{m^2+1}} \tag{2.53}$$

r is the distance from the coordinate center, ω is the angle between the line and the x-axis. Using the Hough transformation the image is transformed to the *Hough space* (r, ω). In an implementation the Hough space has to be quantized. The r-axis which ranges von -90 to 90 degrees can i.e. be split into 1-degree increments. The maximum distance r_{max} for a 64×64 image is $64\sqrt{2}$. As r can be positive or negative and a pixel is defined for every distance, $264\sqrt{2} \approx 181$ values are needed. While the proposed method can be used to find strokes in a character image, the exact stroke length has to be determined from the original image.

2.7 Stroke Extraction

Since the Chinese characters are composed of strokes, a lot of research has been done to develop methods to extract the strokes from a given character image. Some of these methods are based on thinning (see section 2.4), but thinning leads to some deformations of the skeleton. These distortions make the retrieval of strokes more difficult.

In this section, two advanced methods for stroke extraction are presented.

2.7.1 Stroke Extraction Based on Contour Information

This algorithm which was proposed by Lee and Wu [18] consists of five main steps.

contour extraction: The contour (see section 2.5) of a binarized input image is extracted and each point of the contour is encoded with an index for its neighbors. The contour is labelled clockwise.

corner detection: The corner points of the image are found using a heuristic.

CSSG construction: For this step, a modification of the cross section sequence graph (CSSG, see subsection 2.4.3) is used. Contour segments are used to find the cross section sequences.

structure extraction: A region connection graph is constructed from the CSSG.

stroke extraction: In this step, some regions are merged into strokes by a heuristic using Bezier curves.

Lee and Wu tested the algorithm on 200 printed and 64 hand printed characters. The strokes of 195 printed and 62 hand printed characters could be extracted in the correct way.

2.7.2 Stroke Extraction Based on Run Length Encoding

Another stroke extraction method was presented in [4].

For the algorithm the following definitions are made ($p = p[i, j]$ is a point in the binarized input image).

directional run: set of black pixels along a specified direction

run length: number of black points along a run

RUN-H(p): run length of the horizontal run that contains p.

RUN-V(p): run length of the vertical run that contains p.

RUN-45(p): run length of the 45° run that contains p.

RUN-135(p): run length of the 135° run that contains p.

The algorithm is divided into three phases.

horizontal scan: The aim of this phase is to find skeletons of vertical or slanted strokes by a horizontal scan.

step 1: A set A which contains the left boundary of the character is obtained.

$$A = \{p[i,j] | p[i,j] = black \wedge p[i,j-1] = white \wedge p[i,j+1] = black\} \tag{2.54}$$

step 2: For all elements of A perform the following steps:

step 2.1: Find RUN-H(p), RUN-45(p), RUN-135(p). Let q, r, s be the center points of the runs and find RUN-V(p), RUN-45(s) and RUN-135(r).

step 2.2: If RUN-H(p) lies between some threshold thd and $\frac{thd}{2}$ or RUN-45(s) $> 2thd$ or RUN-135(r) $> 2thd$, the further examinations are done to detect the type of stroke.

1. If RUN-V(q) $> 2.5thd$ and RUN-V(q) > 2.5 RUN-H(p), then q is located on a vertical stroke. q is then registered as a member of the skeleton. If this condition is not met, p may be located on a slanted stroke and the following tests are done. Otherwise, the process continues in step 2.1 with the next point p.

2. The next step is to examine whether p is located on a right-up and left-down slanted stroke (RL-stroke). If this would be the case, then the length of the 45° runs would be larger than the length of the 135° runs (see figure 2.22). If the length of the 135° run is also large, then the point is supposed to lie on a junction and is not used for the stroke extraction. Three conditions are tested to determine if p is located on a RL-stroke:
   (a) RUN-135(p) lies between $\frac{1}{2}thd$ and $\frac{3}{2}thd$
   (b) RUN-45(s) > RUN-135(p)
   (c) RUN-45(s) > thd

   If the conditions are satisfied, then s is stored as member of the skeleton. Otherwise, the examinations continue.
   If p lies on a short vertical stroke, one of the two following conditions has to be satisfied.
   (a) RUN-135(p) lies between $\frac{1}{2}thd$ and $\frac{3}{2}thd$ and RUN-V(q) > RUN-H(p)
   (b) RUN-V(q) > 2 RUN-H(p)

   If this test is positive, then q is registered as part of the skeleton and the search is stopped.
3. The next step is to check if p is located on a left-up and right-down slanted stroke (LR-stroke, see figure 2.22). If all three of the following conditions are met, then p is located on a LR-stroke and r is put into the skeleton.
   (a) RUN-45(p) lies between $\frac{1}{2}thd$ and $\frac{3}{2}thd$
   (b) RUN-135(r) > RUN-45(p)
   (c) RUN-135(r) > thd

   The last possibility is that p lies on a short vertical stroke which is the case if either of the following two conditions is satisfied.
   (a) RUN-45(p) lies between $\frac{1}{2}thd$ and $\frac{3}{2}thd$ and RUN-V(p) > RUN-H(p)
   (b) RUN-V(p) > 2 RUN-H(p)

   In this case q is saved as part of the skeleton and the next point p is examined starting in step 2.1.

vertical scan: This phase has the aim to extract horizontal and slanted strokes by using a vertical scan.

step 1: A set B which describes the lower boundary is defined as

$$B = \{p[i,j] | p[i,j] = black \wedge p[i+1,j] = white \wedge p[i-1,j] = black\} \tag{2.55}$$

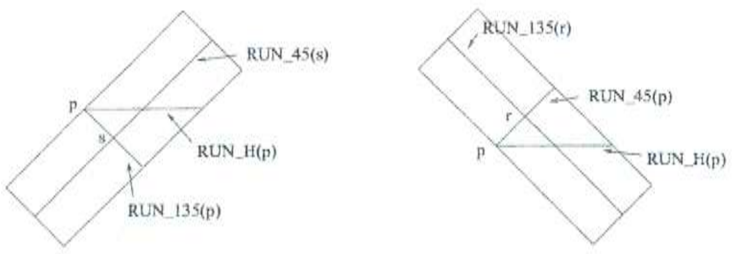


Figure 2.22: Two kinds of strokes in the horizontal scanning: In a RL-stroke RUN-45(s) is larger than RUN-135(p) (left), while RUN-135(r) is larger than RUN-45(p) in a LR-stroke (right)

step 2: For all $p \in B$ perform the following steps:

step 2.1: Find RUN-V(p), RUN-45(p) and RUN-135(p). Let t, u, v be the center points of these runs. Find RUN-H(t), RUN-45(v) and RUN-135(u).

step 2.2: If RUN-V(p) lies between thd and $\frac{thd}{2}$, then the following examinations are performed to decide the type of stroke that contains p.

1. If all of the following three conditions are met, p is located on a RL-stroke and v is a member of the skeleton (see figure 2.23). In this case the next point p is tested in step 2.1.
   (a) RUN-135(p) lies between $\frac{1}{2}thd$ and thd.
   (b) RUN-45(v) > RUN-135(p)
   (c) RUN-H(t) < 2 thd
2. Similar to case 1, p may be located on a LR-stroke (see figure 2.23). The conditions are:
   (a) RUN-45(p) lies between $\frac{1}{2}thd$ and thd
   (b) RUN-135(u) > RUN-45(p)
   (c) RUN-H(t) < 2 thd

   If the conditions are satisfied, u is a part of the skeleton and the examination of the next point p begins.
3. Since p is not located on a slanted stroke, it may be part of a horizontal stroke. To test this, RUN-H(t), RUN-45(t) and RUN-135(t) are found. Then the following four conditions are tested.
   (a) RUN-H(t) > RUN-V(p)
   (b) RUN-H(t) > thd
   (c) RUN-45(t) < thd

(d) RUN-135(t) $< thd$
If all conditions are satisfied, t is put into the skeleton.

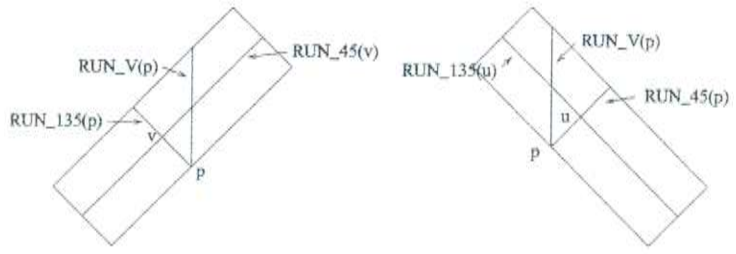


Figure 2.23: Two kinds of strokes in the vertical scanning: RUN-45(v) is larger than RUN-135(p) in a RL-stroke (left), while RUN-135(u) is larger than RUN-45(p) in a LR-stroke

skeleton correction: This phase is used to check the output of the first two phases. It removes redundant points and fills small gaps.

Points that do not have any neighbors are deleted. If the distance of the end between two line segments is less than three, a connection between the two line segments is added.

Using the extracted line segments, it is possible to merge them into strokes. For details of this merging see [4].

The method was tested on 2500 Hei font characters and extracted the strokes in the correct way in 94.2% of the cases.

The method is not suitable if the characters have a large variation in stroke width (i.e. Kai font and Ming font). Different thickness of horizontal and vertical strokes can be overcome by using two different thresholds.

2.8 Blurring

Blurring can be used for systems which perform an analogue pattern matching. It smoothes the character and makes the character more shift-invariant.

The general formulation of a blurring operation is:

$$\Psi_0[g(x'); x, \sigma] = \int_{-\infty}^{\infty} \psi(g(x'); x, x', \sigma) dx' \tag{2.56}$$

with
Ψ_0: functional with transforms a function to a real value
σ: blurring parameter

Often ψ is a Gauss function which proved to be useful as a blurring function.

In an experiment performed by Yasuda [21] the Gauss function is approximated by the following matrix.

$$\begin{pmatrix} 2 & 4 & 6 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 6 & 12 & 15 & 12 & 6 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 6 & 4 & 2 \end{pmatrix} \tag{2.57}$$

2.9 Non-linear Normalization

Non-linear normalization can be used to restore characters that are distorted in a way that cannot be corrected by a linear normalization. This is often the case for handwritten characters.

One normalization method for Japanese handwriting recognition was presented by Kimura et al[16].

1. A smoothing operation with a 2×2 mean filter is applied to the input (binary) image of dimension $I \times J$. The output is again a binary image.

2. The density of lines $p(i,j)$ [43] is calculated at each point (x,y) and is projected by

$$h_x(x_i) = \sum_{j=1}^{J} p(i,j) + \alpha \tag{2.58}$$

$$h_y(y_i) = \sum_{i=1}^{I} p(i,j) + \alpha \tag{2.59}$$

   to obtain the horizontal and vertical projections h_x and h_y. Histogram normalization is applied to both histograms. The factor α determines the level of nonlinearity. The larger the value of α, the less the nonlinearity.

3. Mapping functions which equalize both projections are applied to the smoothed binary image.

4. Another smoothing operation is applied. This time a 3×3 mean filter is used.

2.10 Color Images

For color images methods to convert them to gray-level images are presented in this section.

The color images often use the Red-Green-Blue format (RGB-format). This 24-bit format contains 8-bits for the intensity of each of the three colors. A simple method to get a gray value from this format is to average the values of red, green and blue (see figure 2.24). The North American standard NTSC (see figure 2.25) uses the formula:

$$\text{gray} = 0.299 \text{ red } + 0.587 \text{ green } + 0.114 \text{ blue} \tag{2.60}$$

In Europe (PAL, see figure 2.26), another formula is used:

$$\text{gray} = 0.222 \text{ red } + 0.707 \text{ green } + 0.071 \text{ blue} \tag{2.61}$$

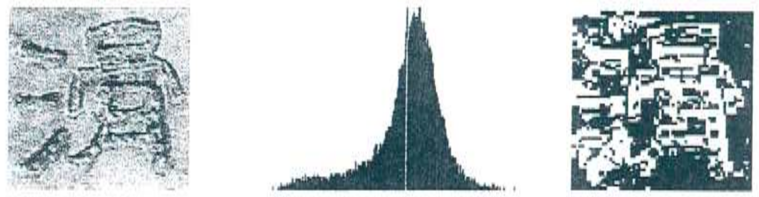


Figure 2.24: Grayscale image generated by using the average of the three colors (left), histogram (middle) and binary image (right)

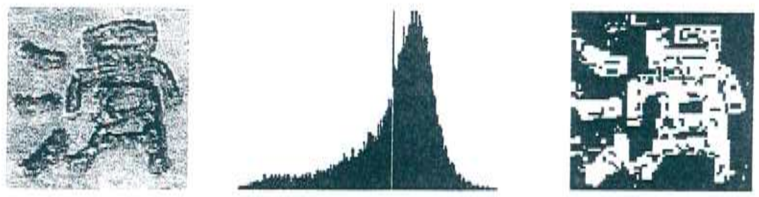


Figure 2.25: Grayscale image generated by using the NTSC formula (left), histogram (middle) and binary image (right)

The proposed system uses the formula from PAL for the recognition of the characters from the street signs. Using this formula helps to avoid cases where the foreground and background are mapped onto the same gray value.

2.11 Character Segmentation

For character recognition of a given text or sign the contents have to be segmented into single characters. This is difficult for the Chinese characters, because some

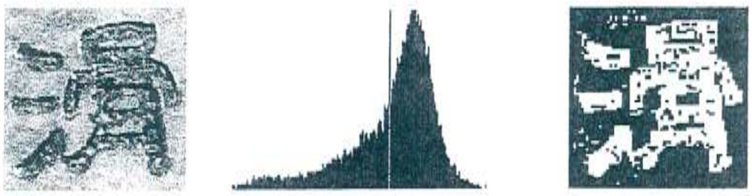


Figure 2.26: Grayscale image generated by using the PAL formula (left), histogram (middle) and binary image (right)

characters are composed of other characters (see section 1.2.3). Especially if there is a vertical gap between the components and if these components are also valid characters, the segmentation often has to include contextual information.

An approach for character segmentation which can also deal with different distances of the characters is the following:

At first, a given image containing several lines of text has to be binarized. A horizontal projection profile is computed as

$$P_r(i) = \sum_{j=1}^{C} B[i,j] \qquad i = 1, ..., R \tag{2.62}$$

with
R: number of rows in the image
C: number of columns in the image
B: binarized image
$P_r(i)$: projection of the i's row

The projection profile is then normalized to the values of (0...255).

$$P_r^*(i) = \frac{255\,(P_r(i) - \min\{P_r(k)|k = 1, ..., C\})}{\max\{P_r(k)|k = 1, ..., C\} - \min\{P_r(k)|k = 1, ..., C\}} \tag{2.63}$$

This normalized projection profile P_r^* can be segmented with a given threshold which should be greater than 0 to avoid that single pixels disturb the segmentation. In the TECHIS system [33] a value of 5 was proved to be useful.

For a given threshold c, a segment of text is beginning at position i_b and ending at position i_e, if the following conditions are met:

$$P_r^*(i_b - 1) \; < c \leq \; P_r^*(i_b) \tag{2.64}$$
$$P_r^*(i_e) \; \geq c > \; P_r^*(i_e + 1) \tag{2.65}$$

With these begin and end positions single text lines can be extracted from the image.

$$TL(i,j) = \{B[i,j]|i_b \leq i \leq i_e; j = 1, ..., C\} \tag{2.66}$$

In analogy to this segmentation into text lines, the text lines themselves can be separated into characters by the application of a vertical profile.

$$P_c(j) = \sum_{i=i_b}^{i=i_e} B[i,j] \qquad j = 1, ..., C \tag{2.67}$$

This profile is also normalized and segmented with the threshold 1.

Some characters that contain vertical gaps cause problems in the segmentation. 55 of the 3755 characters contain gaps in the standard font that are more than four pixels wide (character size 64×64).

Another method for the segmentation of a character line was proposed by Y. Kobayashi, K. Yamada, and J. Tsukumo [17]. The method was used especially for handwritten characters and it also performs a recognition in the segmentation process.

At first, a presegmentation step is performed in which the character line is divided into small segments which can be smaller than a character. In the next step, candidate characters are composed by adding some segments. These candidate characters are recognized by single character recognition, producing several hypotheses for each character. To each hypothesis a confidence value is assigned. Dynamic programming is used to find a character sequence which fits to the character line.

2.12 Linear Discriminant Analysis

The aim of *Linear Discriminant Analysis* [3] is to find a matrix which application maximizes the distance between classes and minimizes the distance in the classes. It can be applied after the features have been extracted to increase the discrimination.

The aim can be formalized as the maximization of

$$\frac{|\tilde{S}_b|}{|\tilde{S}_w|} = \frac{|W^t S_b W|}{|W^t S_w W|} \tag{2.68}$$

with
S_b: between-class scatter matrix
S_w: within-class scatter matrix
W: optimal transformation for the maximization

Let χ_i be the set of examples for class i, the following definitions can be made:

$$m_i = \frac{1}{n_i} \sum_{x \in \chi_i} x \qquad \text{average for class } i \tag{2.69}$$

$$m = \frac{1}{n} \sum_i n_i m_i \qquad \text{average of all classes} \tag{2.70}$$

$$\begin{aligned} S_i &= \sum_{x\in\chi_i}(x-m_i)(x-m_i)^t \qquad \text{deviation for class } i && (2.71)\\ S_w &= \sum_i S_i && (2.72)\\ S_b &= \sum_i n_i(m_i-m)(m_i-m)^t && (2.73) \end{aligned}$$

Further definitions for finding W are:

$$\begin{aligned} y &= W^t x && (2.74)\\ \psi_i &\equiv \left\{y_i|(x_i\in\chi_i, y_i = W^t x_i)\right\} && (2.75)\\ \tilde{m}_i &= \frac{1}{n_i}\sum_{y\in\psi_i} y && (2.76)\\ \tilde{m} &= \frac{1}{n}\sum_i n_i m_i && (2.77)\\ \tilde{S}_w &= \sum_i\sum_{y\in\psi_i}(y-\tilde{m}_i)(y-\tilde{m}_i)^t && (2.78)\\ \tilde{S}_b &= \sum_i n_i(\tilde{m}_i-\tilde{m})(\tilde{m}_i-\tilde{m})^t && (2.79) \end{aligned}$$

And it follows that

$$\begin{aligned} \tilde{S}_w &= W^t S_w W && (2.80)\\ \tilde{S}_b &= W^t S_b W && (2.81) \end{aligned}$$

Taking the determinant of the scatter matrix is equivalent to finding the product of the eigenvalues. This product corresponds to the product of the variances.

$$S_b w_i = \lambda_i S_w w_i \qquad (2.82)$$

The between-class variance in dimension i is proportional to the eigenvalue λ_i. If the within-class variance is constant, the between-class variance of a dimension i determines the discriminativity of that dimension. This can be used to improve the recognition speed if only the dimensions with high discriminativity are used.

The variances in each dimension i have to be normalized

$$w_i^t S_w w_i \qquad (2.83)$$

and each dimension can be scaled using $\hat{w}$ instead of w, where

$$\hat{w}_i = \frac{w_i}{\sqrt{w_i^t S_w w_i}} \qquad (2.84)$$

2.13 Gabor Functions

Gabor filters can be used to transform an input image. The image is transformed in several directions and frequencies.

The convolution masks of 2D-Gabor functions have the general form

$$h(x,y) = g(x',y')e^{2\pi i(Ux+Vy)} \tag{2.85}$$

with
$h(x,y)$: complex sinusoidal function
$(x',y') = (x\cos\phi + y\sin\phi, -x\sin\phi + y\cos\phi)$: rotated coordinates
$g(x,y) = \frac{1}{2\pi\lambda\sigma^2}e^{-\frac{(\frac{x}{\lambda})^2+y^2}{2\sigma^2}}$
λ: parameter
σ: scaling parameter
ϕ: main direction
U, V: central frequency

The space frequency response of the Gabor function is:

$$H(u,v) = e^{-2\pi^2\sigma^2[(u'-U')^2\sigma^2+(v'-V')^2]} \tag{2.86}$$

with
$H(u,v)$: Gaussian function
$(u',v') = (u\cos\phi + v\sin\phi, -u\sin\phi + v\cos\phi)$
(U',V'): rotation of the central frequency (U,V)
ϕ: angle between main axis and u-axis
$\frac{1}{\lambda}$: parameter
$F = \sqrt{U^2+V^2}$: radial central frequency
$\psi = \arctan(\frac{V}{U})$: orientation

Figure 2.27 shows the impulse responses of four Gabor filters.

2.14 Central Projection

Since the structure of Chinese characters is quite complex, a projection method to transform an input character into a convex pattern was proposed [35].

The transformation uses the following steps:

Coordinate transformation: The input character is shifted in a way that its center of gravity is located in (0,0).

The function $f_k(x,y)$ of the k-th character can be considered as a mass distribution on a plane. The set of foreground points is denoted as D

$$f_k(x,y) = \begin{cases} 1 & \text{if } (x,y) \in D \\ 0 & \text{otherwise} \end{cases} \tag{2.87}$$

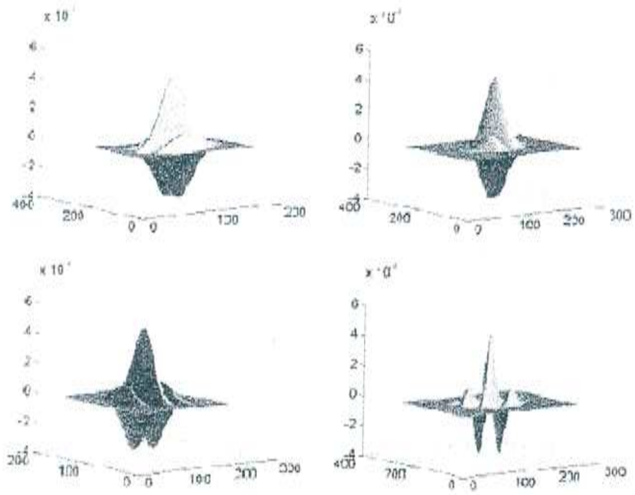


Figure 2.27: Impulse responses of four Gabor filters

For a uniform mass distribution the center value of the region has the coordinates (x_0, y_0). A scaling factor L is defined as:

$$L = \max_{N \in D} |N(x,y) - m(x_0,y_0)| \tag{2.88}$$

with
$|N(x,y) - m(x_0,y_0)|$: Euclidian distance between the points N and m on the plane

The character coordinates are transformed to polar coordinates in the standard way.

$$x = \gamma \cos\phi; y = \gamma \sin\phi; \gamma \in [0,\infty), \phi \in [0, 2\pi] \tag{2.89}$$

Analysis of the information distribution: The following integral which is equal to the mass distribution along the rays with the center $m(x_0, y_0)$ is computed.

$$f_k(\phi) = \int_0^L f_k(\gamma\cos\phi, \gamma\sin\phi)d\gamma \tag{2.90}$$

The function $f_k(\phi)$ can be viewed as a one dimensional pattern. Since the computer can only compute an approximation of the integral, the following sum is computed.

$$f_k(\phi) = \sum_{\gamma=0}^{L} f_k(\gamma\cos\phi, \gamma\sin\phi) \tag{2.91}$$

The frequency of black pixels at the location (γ, ϕ) for all characters is computed as

$$\overline{F}_{\gamma\phi} = \sum_{k=1}^{K} f_k(\gamma \cos\phi, \gamma \sin\phi). \tag{2.92}$$

This is used to compute the probability of the occurrence of a pixel at location (γ, ϕ) and for the information entropy.

$$\overline{P}_{\gamma\phi} = \frac{\sum_{k=1}^{K} f_k(\gamma \cos\phi, \gamma \sin\phi)}{\sum_{k=1}^{K} \sum_{\gamma=0}^{L} \sum_{\phi=0}^{2\pi} f_k(\gamma \cos\phi, \gamma \sin\phi)} \tag{2.93}$$

$$\overline{H} = -\sum_{\gamma=0}^{L} \sum_{\phi=0}^{2\pi} \overline{P}_{\gamma\phi} \log_2 \overline{P}_{\gamma\phi} \tag{2.94}$$

Analysis of the information content: In this step a regrouping of the pixels in a given image is done without increasing or decreasing their number. According to the topological structure of relative positions of the black pixels, they are projected along the rays from the outside to the inside to form a new solid polygon.

The information contained in every pixel of the solid pattern is represented by

$$\overline{I}_{\gamma\phi} = -log_2 \overline{P}_{\gamma\phi} \tag{2.95}$$

Using this technique each character can be transformed to such a contour which can facilitate the contour analysis.

Chapter 3

Feature Extraction

The aim of feature extraction is to define features for the classification. The features should have characteristic values for a certain class and different values for other classes to make a discrimination of the classes possible.

Global methods for counting along certain directions or positions of points and their distances from a reference point are less sensitive against noise and minor local distortions. Structural features, for example edges and lines, outlines or centerlines of characters, describe the topology of a character more accurately. They are less sensitive to deformations of the character than global features.

At first, some features which can be calculated easily, but have limited discrimination ability are presented. They can be used for the rough classification which reduces the candidate set. Features which can be used for the main classification are introduced later on. Their definitions, advantages, and disadvantages are discussed. The last section describes useful ways to combine feature sets.

3.1 Projection Profiles

A character can be projected onto the horizontal or the vertical axis. The result is called a *projection profile* (see figure 3.1). It is also possible to project the characters in the two diagonal directions. For simple characters these profiles can be used also for the fine classification, but for the complex characters the projection profiles get more and more flat. Therefore the projection profiles should be used in the preclassification.

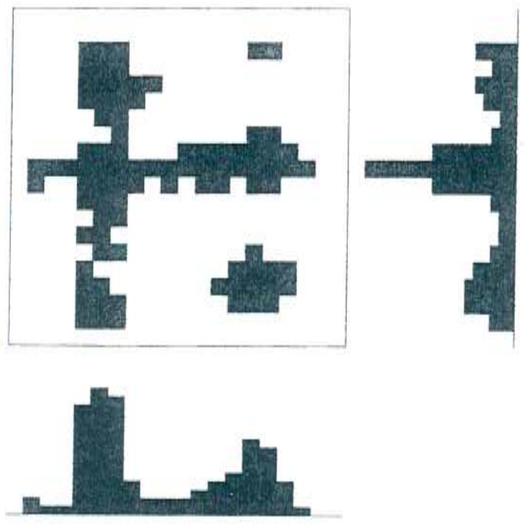


Figure 3.1: Horizontal and vertical projection profiles

3.2 Average Stroke Width

For the *average stroke width* each position in the binarized input image is convolved with the following three masks.

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & \\ & 1 & 1 \end{pmatrix} \begin{pmatrix} & 1 & 1 \\ 1 & 1 & \end{pmatrix}$$

If there is a hit with one of them, then the number of hits h is increased by one. The average stroke width is estimated as $\frac{b}{(b-h)}$, where b is the total number of black pixels.

3.3 Total Stroke Length

A feature useful for preclassification is the *total stroke length (TSL)* which is an approximation of the length of all strokes in the character. It is calculated as

$$TSL = \frac{b}{W} = \frac{b}{\frac{b}{(b-h)}} = b - h \tag{3.1}$$

with
b: total number of black pixels
W: average stroke width (see section 3.2)
h: number of hits with the three masks

The TSL is an indicator for the complexity of an input character.

3.4 Characteristic Loci

The *characteristic loci* were developed by Glucksman [7]. For every background point a stroke detection line is spanned in top, bottom, left, and right direction. The number of crossings of black lines are counted for every direction. If the number is bigger than two, the number is set to two.

Because the features are not stable if one relies on values for some points, a feature histogram is used. Using linear discrimination functions Gluckman's features obtained good results for printed letters from the Roman alphabet. Further developments of the method are able to segment a picture into several regions where the points in one region have the same characteristic loci.

3.5 TBC Features

The *TBC features* [21], where T stands for terminal, B for branch and C for crossing, use special points of a character for the recognition. The connectivity of a point is described as the number of components which are connected to a point. Connectivities from 0 to 4 are possible for a two-dimensional image on a grid plane. After thinning the connectivities indicate the following meaning:

0: The point is an isolated point.

1: The point is a terminal point.

2: The point lies inside a curve.

3: The point is a branch point.

4: The point is a crossing point.

The terminal, branch and crossing points can be used in different ways as features for the recognition system. The simplest way is to take the number of points. Since this is not sufficient to distinguish many characters, the direction from which a terminal point is accessed can also be stored. This can be done by using the chain code (see section 2.5). For further discrimination, the positions of the points can be used. An advantage of these features is that they are shift invariant.

3.6 Direction Features

The *direction features* [9, 10, 37, 38] create four planes for strokes that run in horizontal, vertical, or one of the two diagonal directions.

For a given preprocessed image the strokes are extracted and according to their direction put into one of the direction planes. The last three planes are rotated in a way that all strokes run in the vertical direction. The implementation in the proposed system uses the edge enhancement procedure (see section 2.3.3) to make the features more stable. Then the contour of the strokes is calculated in all planes. Afterwards a blurring step with a Gaussian function is performed. A grid consisting of 16×4 regions is defined for each plane and the number of pixels are summed in each region. 256 features are extracted for all four planes. The feature extraction procedure is shown in figure 3.2.

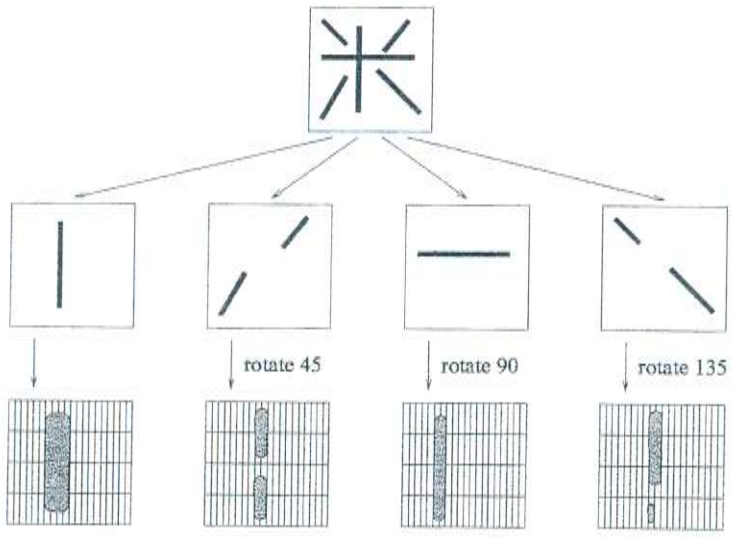


Figure 3.2: Extraction of the direction features: Stroke distribution and feature extraction after rotation and blurring

A simpler feature set with only 64 values can be calculated by adding four neighboring feature values of the large feature set. This small feature set can be useful for preclassification.

3.7 ET Features

The *ET1 features* describe the periphery of a character. For their calculation the character image is divided into eight horizontal and eight vertical stripes. The 32 features for a 64×64 binary image $b(i,j)$ are defined as follows:

$$ET1(0,n) = \sum_{i=1}^{8} \min\{j | b(8n+i,j) = 1\} \tag{3.2}$$

$$ET1(1,n) = \sum_{i=1}^{8} [64 - \max\{j | b(8n+i,j) = 1\}] \tag{3.3}$$

$$ET1(2,n) = \sum_{j=1}^{8} \min\{i|b(i,8n+j)=1\} \tag{3.4}$$

$$ET1(3,n) = \sum_{j=1}^{8} [64 - \max\{i|b(i,8n+j)=1\}] \qquad n=0,...,7 \tag{3.5}$$

Figure 3.3 shows the ET1 features for a sample character.

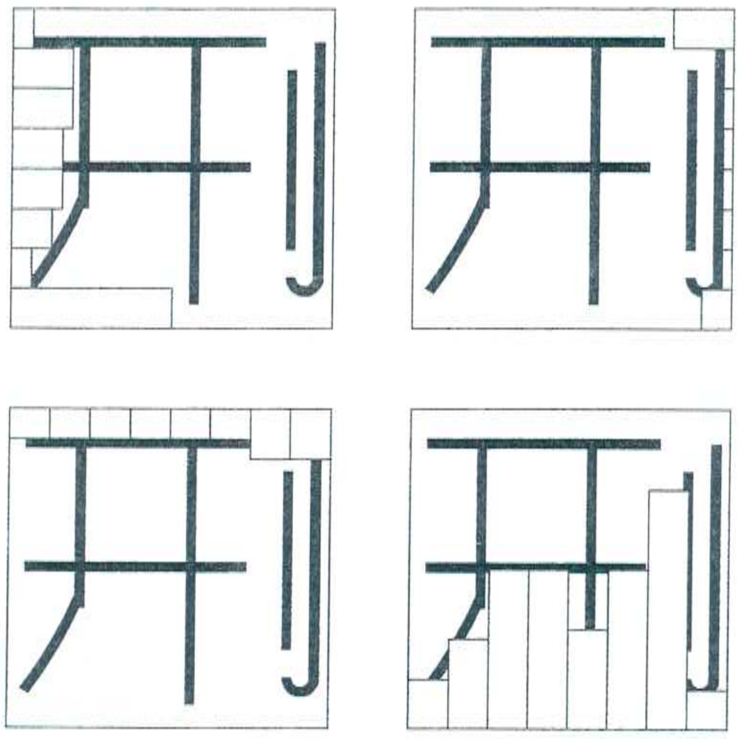


Figure 3.3: ET1 features for a sample character

While the ET1 features only use the periphery of a character, the *ET2 features* (see figure 3.4) describe the inner structure of the character by calculating the second position where a transition from white to black occurs.

$$N_J(i,j^*) = \sum_{j=1}^{j^*} \sigma[b(i,j)-b(i,j-1)] \qquad \text{changes in vertical direction} \tag{3.6}$$

$$N_J(i^*,j) = \sum_{i=1}^{i^*} \sigma[b(i,j)-b(i-1,j)] \qquad \text{changes in horizontal direction} \tag{3.7}$$

$$\sigma[x] = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \qquad \text{step function} \tag{3.8}$$

$$ET2(0,n) = \sum_{i=1}^{8} \min\{j^* | N_J(i,j^*-1) < 2; 2 = N_J(i,j^*)\} \tag{3.9}$$

$$ET2(1,n) = \sum_{i=1}^{8} (64 - \max\{j^* | N_J(i,64) - N_J(i,j^*) = 1; N_J(i,j^*-1) < N_J(i,j^*)\}) \tag{3.10}$$

$$ET2(2,n) = \sum_{j=1}^{8} \min\{i^* | N_J(i^*-1,j) < 2; 2 = N_J(i^*,j)\} \tag{3.11}$$

$$ET2(3,n) = \sum_{j=1}^{8} (64 - \max\{i^* | N_J(64,j) - N_J(i^*,j) = 1; N_J(i^*-1,j) < N_J(i^*,j)\}) \quad n = 0,\ldots,7 \tag{3.12}$$

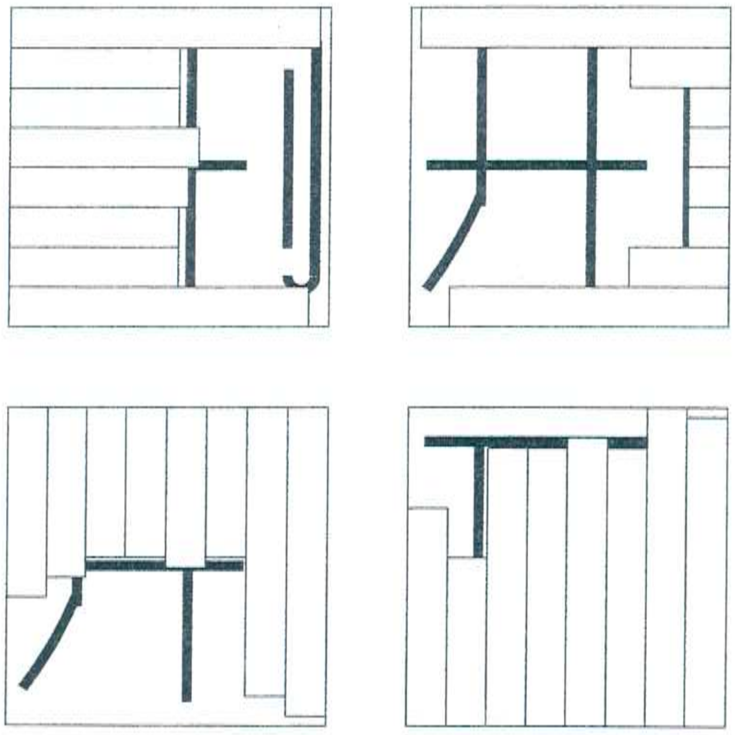


Figure 3.4: ET2 features for a sample character

Experiments done by Suchenwirth et al. reported recognition accuracies of 96.35% for ET1 and 98.90% for ET2.

3.8 Local Direction Contributivity

The *local direction contributivity (LDC)* features [8] calculate for every black pixel the run length in horizontal, vertical, and the two diagonal directions. These values called d_0, d_1, d_2, d_3 can determine the direction of the stroke on which the pixel is located. They are normalized by the following computation:

$$d_k^*(i,j) = \frac{d_k(i,j)}{\sqrt{\sum_{l=0}^{3} d_l^2(i,j)}} \qquad k = 0, ..., 3 \qquad (3.13)$$

After that the picture is divided into $n \times n$ regions and the values of each direction are summed in every region, resulting in $4n^2$ features.

Figure 3.5 shows an example where the directions for a certain point have been calculated.

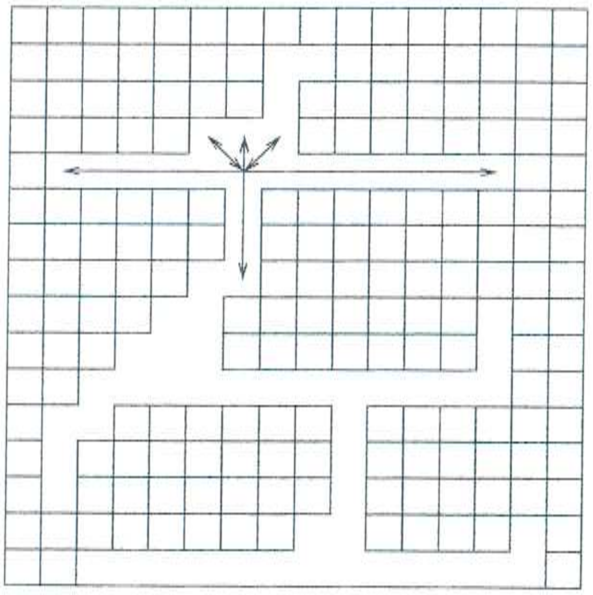


Figure 3.5: Calculation of run length for the LDC features

The LDC features are well suited for hand printed characters, because these have a uniform stroke width. When the stroke width differs, this causes problems and makes the recognition of printed characters quite font-dependent.

3.9 Stroke Proportion

The *Stroke Proportion (SP4)* features were derived from the LDC features to make them more independent from the stroke width.

These features consist of 16 horizontal and 16 vertical features. For the horizontal features the image is divided into four horizontal stripes. For each of these stripes four direction features are calculated which sum all points that lie on a stroke in the corresponding direction. To determine the stroke direction for a point the run length is calculated for each direction as for the LDC features. The features are then divided by the number of all points in the current stripe. For the vertical direction analogue calculations are performed after the image has been divided into four vertical stripes.

The formulas for these features are:

$$f_1(n) = \sum_{i=16n+1}^{16n+16} \sum_{j=1}^{64} b(i,j) \tag{3.14}$$

$$f_2(m,n) = \sum_{i=16n+1}^{16n+16} \sum_{j=1}^{64} \left\{ b(i,j) | \text{for each } (i,j) \text{ with } d_m(i,j) = \max_k d_k(i,j) \right\} \tag{3.15}$$

$$H - SP4(m,n) = \frac{f_2(m,n)}{f_1(m,n)} \tag{3.16}$$

$$f_3(n) = \sum_{j=16n+1}^{16n+16} \sum_{i=1}^{64} b(i,j) \tag{3.17}$$

$$f_4(m,n) = \sum_{j=16+1}^{16n+16} \sum_{i=1}^{64} \left\{ b(i,j) | \text{for each } (i,j) \text{ with } d_m(i,j) = \max_k d_k(i,j) \right\} \tag{3.18}$$

$$V - SP4(m,n) = \frac{f_4(m,n)}{f_3(n)} \qquad n = 0, ..., 3; m = 0, ..., 3 \tag{3.19}$$

As the fourth direction is dependent from the other direction values, it can be neglected, because it contains no additional information. The feature space can therefore be reduced to 24 values.

The stroke directions for each point of a sample character are displayed in figure 3.6.

3.10 Black Jump Distribution in Balanced Subvectors

The *Black Jump Distribution in Balanced Subvectors (BJD-BS)* features were developed in the TECHIS project [33]. Using these features for the recognition of the class 1 of the GB 2312-80 standard a recognition rate of 98.3% was obtained in that project.

The binarized image $b(i,j)$ (dimension $m \times m$) is arranged in a vector $\underline{v}_\mu(k)$, $k =$

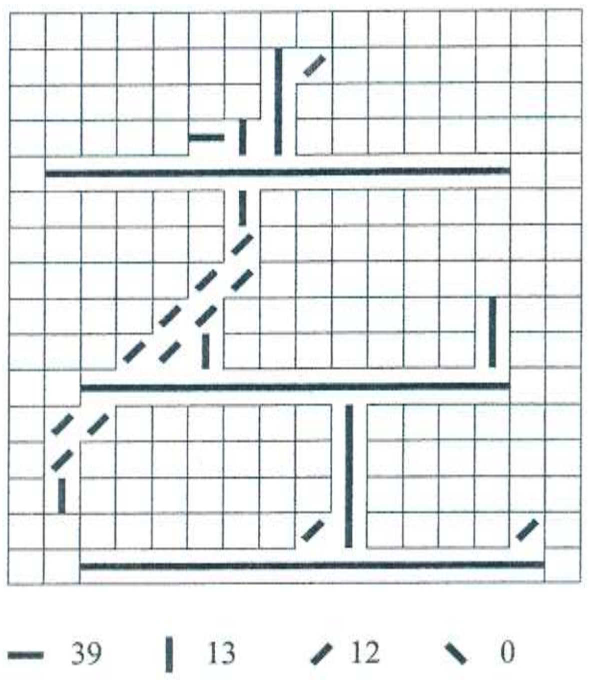


Figure 3.6: Stroke directions for each character point for the SP4 feature calculation

$1, \ldots, m^2$ in four different ways.

$$b(i,j) = \underline{v}_\mu(\underline{k}_\mu[i,j]), \mu = 1, \ldots 4 \tag{3.20}$$

The first arrangement is column-wise.

$$\underline{k}_1[i,j] = (j-1)m + i \tag{3.21}$$

The second arrangement is row-wise as shown in figure 3.7.

$$\underline{k}_2[i,j] = (i-1)m + j \tag{3.22}$$

The third arrangement is build going through the image in 225° diagonal direction (see figure 3.7)

$$\underline{k}_3[i,j] = \begin{cases} \frac{1}{2}(i+j-1)(i+j-1)+i & \text{if } i+j \leq m+1 \\ \frac{1}{2}m(m+1)+m-j+ & \\ \frac{1}{2}(3m+1-i-j)(i+j-2-m) & \text{otherwise} \end{cases} \tag{3.23}$$

The last arrangement is the other diagonal direction. i is here replaced by $u = m - i + 1$.

$$\underline{k}_4[i,j] = \begin{cases} \frac{1}{2}(u+j-1)(u+j-1)+u & \text{if } u+j \leq m+1 \\ \frac{1}{2}m(m+1)+m-j+ & \\ \frac{1}{2}(3m+1-u-j)(u+j-2-m) & \text{otherwise} \end{cases} \tag{3.24}$$

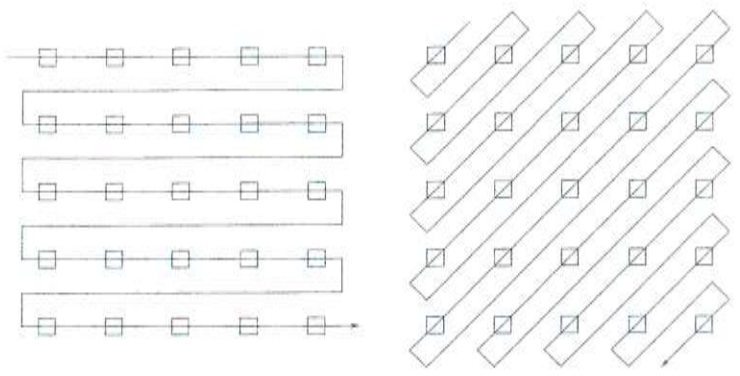


Figure 3.7: Horizontal and diagonal vector construction for the BJD-BS features

In the previous formulas the counting of i and j ranges from 1 to m. These arrangements can be done without actual copying of the values.

The number k of subvectors (k=8 is the TECHIS project) determines the number of black pixels in each subvector $\underline{t}_{\mu i}$. The number is called q.

$$q = int\left[\frac{m_{00}(\underline{v})}{k}\right] \tag{3.25}$$

Afterwards $\underline{v}_\mu$ consists of subvectors $\underline{t}_{\mu i}$.

$$\underline{v}_\mu = [\underline{t}_{\mu 1}, ..., \underline{t}_{\mu n}] \tag{3.26}$$

The feature values $\underline{x}_\mu(j)$ of the subvectors $\underline{t}_{\mu i}$ (j=1,...,k;μ=1,...,4) indicate the number changes from white to black in the subvector relative to the number of such changes in the whole vector $\underline{v}_\mu$. N_J is defined as for the ET2 features (see section 3.7).

$$\underline{x}_\mu(j) = \frac{N_J(\underline{t}_{\mu j})}{N_J(\underline{v}_\mu)} \tag{3.27}$$

The feature values therefore can have only values from the interval [0,1] and their sum is less or equal 1.

3.11 WDH Features

The *weighted direction code histogram (WDH)* features [16] consist of 392 values which are extracted from the contour of a character using the following steps (see figure 3.8):

1. For each pixel in the contour the chain code (see section 2.5) is found and the vector sum with a neighbor in the contour is calculated resulting in 16 direction values.

2. The picture is divided into 13 × 13 regions of equal size. For each region a direction code histogram is calculated.

3. The resolution is reduced from 13 × 13 to 7 × 7 by down sampling every two horizontal and every two vertical blocks with a 5 × 5 Gaussian filter. Similarly, the directional resolution is reduced from 16 to 8 by downsampling with a weight factor $[121]^T$ to produce 392 features (7 horizontal, 7 vertical and 8 direction).

4. Every feature is replaced by its square root. The distribution will then be Gaussian-like [40, 6], which is important for the further application of a LDA.

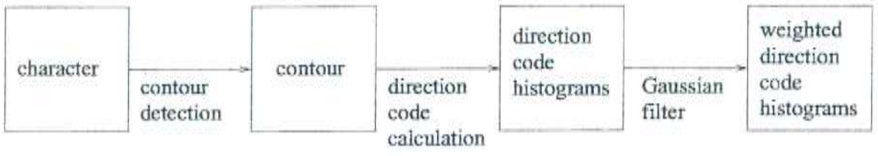

character
contour detection
contour
direction code calculation
direction code histograms
Gaussian filter
weighted direction code histograms

Figure 3.8: Steps of the WDH features calculation

3.12 Feature Selection and Weighting Using Genetic Algorithms

In this chapter, many features which can be used in the classification were introduced. This section deals with selecting the most appropriate features for a recognition task.

A method to select features by using genetic algorithms was introduced by Kim [15]. A binary string containing a bit for each feature is defined. A bit is set to 1 if the feature is used and 0 otherwise. Using the standard mutation and crossover operators known from the simple genetic algorithm, new binary strings are generated. Their "fitness values" are determined by the recognition results using the selected features. These values are used to select the strings which are discarded.

Another proposed method from that paper can be used to determine the importance of the features. The genetic algorithm is again used and each time a feature is used a counter is increased. At the end, the counter values for the features are normalized and applied as a weight factor. Experiments show that the first method can be used to reduce the feature space without losing too much recognition accuracy. The weighting of the features improves the accuracy.

A drawback of the proposed methods is that the evaluation of the fitness value for every new bit string can be quite time consuming.

Chapter 4

Character Classification

The task of *character classification* is to classify an unknown input character as one of the known character classes.

The character classification can be done in a flat or hierarchical way. In the flat classification, a score that the input character belongs to a certain class is calculated for every class. In the hierarchical classification a tree structure has been contracted to group the character classes. The height of the tree determines the number of classification steps that are needed until a final result is returned. The hierarchical classification can save computation time, but can lead to a decline in the classification accuracy.

The assigning of an input character to a character subset is called *preclassification*. The subsets of characters can either be fixed or variable. The variable sets need more computation time, but can be more useful. They can consist of an ordered list of the characters with the lowest distances to the input character. For a variable character set it is possible to use a fixed number of characters or all characters that have a distance under a certain threshold. Using the fixed number of characters limits the computation time, but may exclude the real character, if the number is too small. The subset with a variable size depends on an appropriate threshold. If the threshold cannot be selected properly, the subset gets either very large which increases the computation time or gets too small which can lead to the exclusion of the correct character. When using subsets with variable length, upper and lower bounds for the subset size should therefore be used.

A character can either be recognized correctly or incorrectly by the system. If a confidence measure is defined, the system can reject an input character which should be either too noisy or does not belong to the system's character set. This can result in different cases:

correct input, correct recognition: This should be the case.

correct input, incorrect recognition: Worst case which should occur very seldom.

correct input, rejection: This is not what the system should do, but it is not as bad as an incorrect recognition.

incorrect input, rejection: This should be the case for an incorrect input.

incorrect input, not rejected: This is also a bad case which should be avoided.

One of the simplest and oldest methods for comparing characters is *pattern matching*. In this case, it means to take the black/white images of two characters and compare them pixel by pixel. But this method is quite time consuming for all the characters and is not robust against noise inside the images. Therefore the comparison of feature vectors is used.

4.1 Classification Functions

For the classification of an unknown character the feature vector x has to be compared with the feature vectors y_i of the known character classes. The comparison requires the definition a similarity relation between the feature vectors. One common distance calculation is the l_p-norm:

$$||x - y_i||_p = \left[\sum_{j=1}^{N} |x(j) - y_i(j)|^p\right]^{\frac{1}{p}} \qquad 1 \le p < \infty \tag{4.1}$$

The most common values for p are 1 (cityblock distance) and 2 (Euclidian distance). Using the cityblock distance is faster, because no multiplications are needed.

It is also possible to calculate an angle between the vectors.

$$\cos(\alpha) = \frac{xy_i}{||x||||y_i||} \tag{4.2}$$

The Bayes classifier is defined as

$$\tilde{c} = \arg\max_c p(c|X) = \arg\max_c \frac{p(X|c)p(c)}{p(X)} = \arg\max_c p(X|c)p(c) \tag{4.3}$$

where
$\tilde{c}$: selected class with the maximum probability of all classes c
X: input, here the feature vector
$p(c|X)$: probability of the class c under the input X(a posteriori probability)
$p(X|c)$: probability of the input X given the class c
$p(c)$: probability of the class c (a priori probability)
$p(X)$: probability of the input X; This value is independent from c and can therefore be omitted.

A modified version of the Bayes classifier was used by Kimura et al.[16] for the classification of Japanese handwriting. It is defined as

$$g(X) = (N + N_0 + 1)ln\Big\{1 + \frac{1}{N_0\sigma^2}\{||X - M||^2 \quad (4.4)$$

$$-\sum_{i=1}^{k} \frac{\lambda_i}{\lambda_i + \frac{N_0}{N}\sigma^2}[\Phi_i^T(X - M)]^2\}\Big\} \quad (4.5)$$

$$+\sum_{i=1}^{k} ln\left(\lambda_i + \frac{N_0}{N}\sigma^2\right) \quad (4.6)$$

where
X: feature vector of size n
N: number of the design sample
N_0: correction variable to increase the recognition performance. Its value is determined by experiments.

The required computation time and storage is $O(kn)$.

The subspace method which was also used by Kimura et al. uses the following distance measure.

$$g(X) = \sum_{i=1}^{k}(\Phi_i^T X)^2 \quad (4.7)$$

Here Φ_i is the eigenvector of an autocorrelation matrix. In this method the feature vector X is normalized to the norm 1. Different categories share the same subspaces which is a drawback of the classifier. If $k = n$ then all categories share the same feature space and the classifier returns the result 1 for all categories.

4.2 Plausibility Checks

Some plausibility checks can be used for the candidate list in the final classification.

The distance between the first and the second candidate can be used. If the distance is large, the system can return the first character without performing other examinations. In the classification process a cut can be applied which means that the classification is stopped, if a character has a very small distance to the input.

The character frequency which can be estimated from a large text corpus can be used to weight the results in the character list. A simple technique is to return the most frequent character of the character list.

4.3 Combining Different Feature Sets

To improve the recognition rate, not only the distance calculated from one set of features can be used, but also a combination of the distances calculated from several feature sets. With two feature sets a weighting factor can be found in experiments. In these experiments several factors are tested and the one with the best results is selected [39]. For more than two feature sets this method has to test much more combinations of weighting factors which creates a demand for an automated approach. In the TECHIS system (see section 6.2) a solution for this was to weight the features in the following way:

$$\tilde{D}_k = \sum_{j=1}^{m} \frac{D_k^j - \min\{D_i^j | i = 1, ..., N\}}{\max\{D_i^j | i = 1, ..., N\} - \min\{D_i^j | i = 1, ..., N\}} \qquad k = 1, ..., N \quad (4.8)$$

with
N: number of candidates
m: number of feature sets
D_i^j: distance of the i-th candidate for the j-th feature set

This is called *relative distance*. An *absolute distance* measure for rejecting characters is defined as

$$\hat{D}_k = \sum_{j=1}^{m} \frac{|x(j) - x_k(j)|}{E\{|x(j) - x^*(j)|\}} \hat{=} \sum_{j=1}^{m} \frac{D_k^j}{D^{*j}} \qquad k = 1, ..., N \quad (4.9)$$

with
D^{*j}: expected value of the distance between a character sample and its reference for the j's feature set. Its value is estimated in the training.

A rule for classifying in the TECHIS system was defined as: If the absolute distance of the candidate with the minimal relative distance is less than a threshold, the character category of the candidate is assigned to the unknown sample, otherwise the character is rejected.

4.4 Bagging

Bagging methods [23] can be used to combine several classifiers. A simple method for combining N sub-classifiers is to take the character that is the result of most sub-classifiers as result of the combined classifier.

If the sub-classifiers generate probabilities for each character, a more complex combination method is possible. For each class α the output value can be defined as

$$P_\alpha(x) = f\left(P_{\alpha_1}(x), ..., P_{\alpha_N}(x)\right) \quad (4.10)$$

P_{α_i} is the probability generated by the i's sub-classifier and f a combination function. For f the average or the maximum can be used.

4.5 Constraint Graph Approach

The *constraint graph* approach aims at decomposing an input character into its components. Huang et al. [14] used a three-level representation hierarchy for a

Chinese character, consisting of the character level, the component level, and the stroke level. (see figure 4.1). The component level contains approximately 400 basic components. In the stroke level 20 strokes were defined as essential.

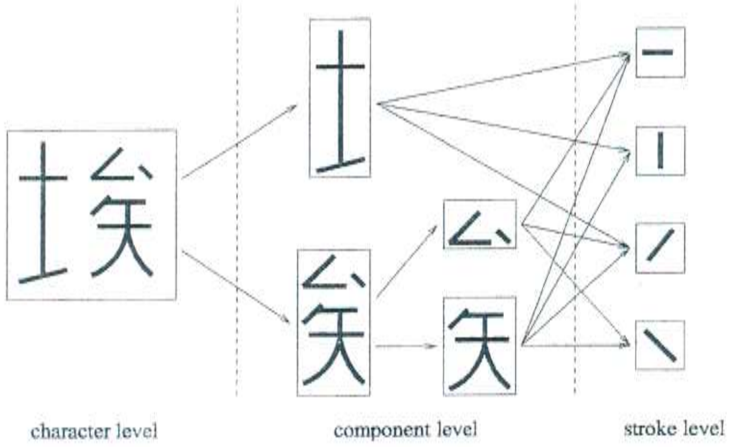


Figure 4.1: Decomposing a character

The classification is done by representing each character as a set of characteristic points. For every couple of points the existence or non-existence of a connection in the character is noted. The input character is matched with the representation of the samples by finding a mapping of the representation and calculating a distance. Using some heuristics the finding of such mappings is accelerated. For different fonts and input sizes the system obtained recognition accuracies between 95% and 99%.

Chapter 5

Postprocessing

The *postprocessing* can be used to apply knowledge about the used language. The application of such knowledge can be very useful to get only valid output.

The *perplexity* of a recognition task is defined as the average number of possible successors for a character. The higher the perplexity the more difficult is the recognition task.

5.1 Linguistic Context

Linguistic context can be used for weighting of different character sequences. The probability of a character sequence is defined as

$$p(w_1, ..., w_i) = p(w_1)p(w_2|w_1)...p(w_i|w_1, ..., w_{i-1})$$

The values $p(w_i|w_1, ..., w_{i-1})$ are difficult to estimate, they can be approximated as $p(w_i|w_{i-1})$ (bigrams) and $p(w_i|w_{i-2}w_{i-1})$ (trigrams). For the beginning of a character sequence the a priori probability $p(w_1)$ can be used.

The values of the bigrams and trigrams have to be estimated from a large text corpus.

5.2 Rectification Method

Hsieh et al. [13] proposed a postprocessing method for Chinese characters to correct some misrecognized characters in a character sequence.

The rectification scheme receives as input the results of the classification. These are the candidate sets for the character images. The process itself has three phases.

In the *candidate-word-classification* phase the candidate sets are separated into valid sets and possibly invalid sets. It is checked whether the top ranking character from the current character set can form a valid character pair with a character from the neighboring character set. If such a valid pair can be found, in most cases both of its constituent characters are correct. These characters are not changed anymore. The character sets where none of the characters is part of a valid pair are regarded as possibly invalid.

The *candidate-character-examination* phase is used to find the highest ranking character in each possibly invalid character set by using the *line-segment cancellation* method. This method reconstructs missing line segments using geometrical features. If the highest ranking character can form a legal word with one of its neighbors, then both are finalized. Otherwise, for each possibly invalid top ranking character new characters which are similar to the character are introduced into the character set.

In the final *additional-character-inspection* phase the cancellation method is reapplied on the characters in each additional set to determine the final most likely character. Using this additional characters, a correct character can be found even when it was not part of the input character set.

Chapter 6

Other Recognition Systems

This chapter presents other recognition systems that can be used for Chinese character recognition. The first system uses Gabor functions to perform the recognition task and was developed for character recognition of street signs. The TECHIS and BBN Byblos systems are designed for the recognition of Chinese characters which are printed in a newspaper or magazine. These systems can detect the same characters as the system developed in this diploma thesis. The radical extraction system decomposes an input character into the constituent strokes. The next system is designed for the recognition of Japanese handwriting, but its methods are also interesting for the current task. The last system is different from the other systems. Its purpose is to detect the font which is used in an input image. The input image has to contain several characters.

6.1 System Based on Gabor Analysis

This system was developed by Jing Zhang at the same time as the proposed system. The aim of the system was to perform the recognition of characters from street signs using Gabor functions.

6.1.1 Preprocessing

In contrast to many other recognition systems the input image is not converted into a binary image which avoids an information loss in this step. Instead a transformation is done to avoid problems with different light conditions.

6.1.2 Feature Extraction

In the feature extraction a Gabor function is defined as

$$G(x, y, k, \phi) \;=\; G_1(x, y)[\cos(R) - \exp^{-\frac{\sigma^2}{2}}] + iG_1(x, y)\sin(R)$$

$$\begin{aligned} G_1(x,y) &= \frac{k^2}{\sigma^2} exp^{-\frac{k^2(x^2+y^2)}{2\sigma^2}} \\ R &= kx\cos\phi + ky\sin\phi \\ k &= \frac{2\pi}{\tau} \end{aligned}$$

where
σ: deviation of Gaussian envelope
τ, ϕ: wavelength and orientation of Gabor function

For a given pixel (x_1, y_1) with the intensity $I(x_1, y_1)$ its Gabor feature can be regarded as the convolution

$$J(x_1, y_1, k, \phi) = \int I(x_1 - x, y_1 - y) G(x, y, k, \phi) dx dy$$

For m frequencies and n orientations, a vector of mn complex coefficients for each position is obtained. Here, the character is divided into a 7×7 grid, resulting in $49mn$ features.

6.1.3 Classification

After the feature extraction an LDA matrix is used to transform the features and to reduce the feature space. The classification is performed by using a KNN method.

6.2 TECHIS

The TECHIS project [33] was performed in 1989 by Suchenwirth et al.

The database for the system consists of reference patterns in Songti font, one for each character recorded by a camera. A second version of each character was created by using a smoothing algorithm (see section 2.3). The test data were 15 pages of the Chinese magazine "Xiandaihua" with a total number of 19,241 Chinese chars (19,231 character are part of class 1, 1,544 different characters are contained).

Best results were found for the BJD-BS features in one experiment and for a combination of ET1, ET2, SP4, and 4-SDF features in another. The 4-SDF features are similar to the BJD-BS features without balancing the subvectors. The result for the BJD-BS features was 98.21% in an average recognition time of 2.85 seconds. A further test in which the mean and the standard deviation for some frequent characters were calculated showed the stability of the BJD-BS features. In the test of the feature combination (ET1, ET2, SP4, 4-SDF) on the same data a recognition rate of 99.92% was obtained (7 substitutions and 9 rejections occurred). The recognition time here was 3.2 seconds on a more powerful computer.

6.3 BBN Byblos

The BBN Byblos OCR system [19] uses a Hidden Markov Model [28] approach for the character recognition. The system was mainly designed for the OCR of English characters, but also proved to be useful for Chinese OCR.

6.3.1 Feature Extraction

A given text line is segmented into a sequence of overlapping narrow vertical frames. The frames themselves are divided into 20 overlapping cells. For each frame the following features are calculated: the intensity (percentage of black pixels in a cell), the vertical and horizontal derivatives of the intensity, and the local slope and correlation across a window of a 2-cell square [32]. Then a LDA (see section 2.12) is performed to reduce the feature space from 80 to 15.

6.3.2 Continuous-density HMMs for Characters

The characters are modelled with 7-state or 14-state left-right HMMs. For each state, the modelling allows that the system stays in the same state or jumps one or two states. For each character, the probability density is modelled using a mixture of 128 Gaussian densities.

6.3.3 Using the System for Chinese OCR

The system was trained and tested first on computer generated data that were printed and scanned. These included the 3,755 characters of GB2312-80 class 1 and 115 Latin and punctuation characters in the 4 main fonts (see section 1.2.5). Using a 14-state HMM for each character, an average error rate of 0.5% was observed. Then the system was tested on a part of a Chinese newspaper (People's Daily) which contained 2,600 unique characters in mainly a single font. Using another part of the newspaper data and the computer-generated data for training, a recognition result on 98.8% was obtained.

6.4 Hierarchical System Using Radical Extraction

A hierarchical system for the recognition of Chinese characters is presented in [41]. Its idea is to recognize an input character by decomposing it into its radicals.

Starting with an input of straight line strokes the system consists of three modules (see figure 6.1). The first module which produces a radical list is structured into three layers. Layer 1 tries to determine in which way the character

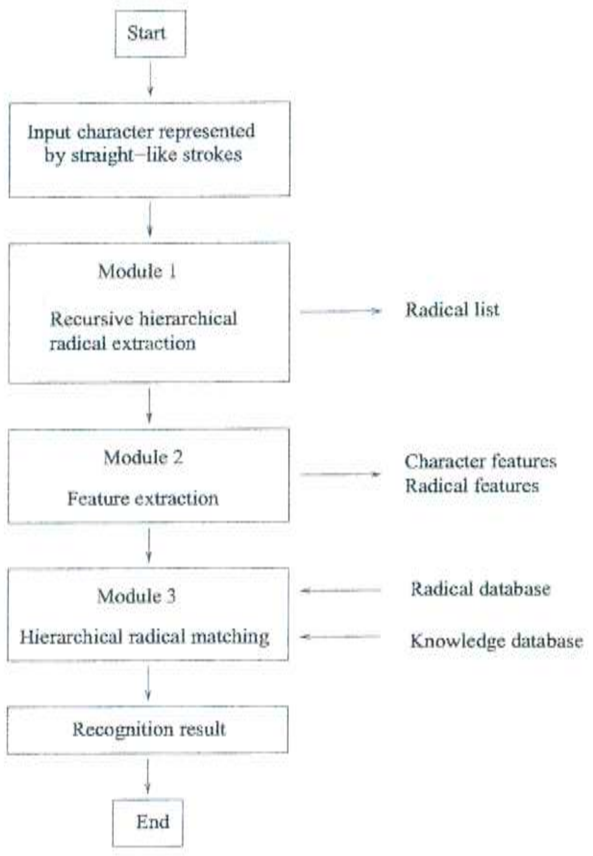


Figure 6.1: Diagram of radical-based OCR system

is composed of radicals. There are ten possibilities how a character can be composed of the radicals (see figure 6.2). Specific tests are performed for each of the ten possibilities until one is chosen. In layer 2 lines to cut the character into parts are determined and in layer 3 the strokes are clustered into radicals by using a k-means clustering algorithm.

The feature extraction module (module 2) normalizes the representation of the character to a 150×150 grid. Two sets of features are used for the further description of each character: The given stroke decomposition and the radicals extracted in module 1.

Module 3 performs the character matching in a hierarchical way. At first, the radicals are matched with a radical database. Then, a knowledge database is used to detect the whole character. If this is not successful, a matching of the character's strokes is performed.

Experiments for radical extraction were performed on 1,856 characters. The successful rate of radical extraction was 92.5%.

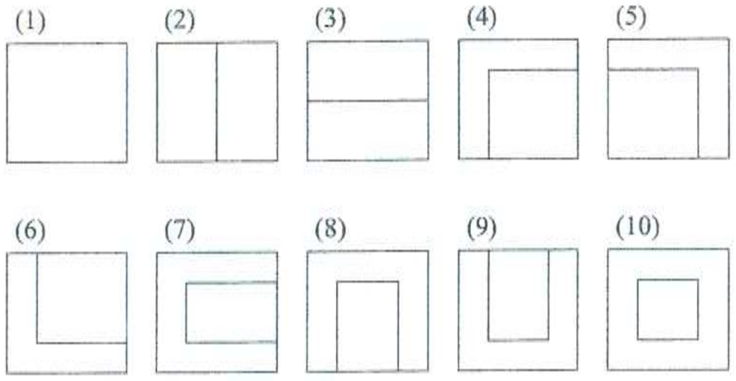


Figure 6.2: Ten patterns of Chinese characters

6.5 System Using Direction Features

A system for off-line and on-line recognition for Japanese Kanji was proposed by Hamanaka et al.[9, 10, 37, 38].

In the system the input data are first normalized to a given size, resampled, and smoothed. Then they are transformed into a bitmap pattern. This pattern is transformed into four patterns for different directions by using a non-linear transformation [11]. These four patterns are resolved into a 256-element feature pattern and blurred. For preclassification a 64-element feature pattern is constructed by using the average of four elements.

The classification is done in three steps: candidate preselection, rough classification and fine classification. The candidate preselection uses the number of strokes in combination with the estimated handwriting quality. In the rough classification, the extracted hypotheses are compared with the input using the 64-element feature patterns. The 264-element feature patterns are then applied in the fine classification.

The database for this system contained data from 51 writers. The character set was defined as the characters from the Japanese Industry Standard JIS (2,965 Kanji) and several other characters (71 Hiragana, 71 Katakana, 62 alphanumerics and 28 other symbols).

The newest version of the system defined three handwriting qualities and received recognition accuracies of 94.3%, 90.5%, and 76.6% for the different qualities.

6.6 System for Font Recognition

A system for the detection of Chinese fonts was proposed by Zhu et al. [44].

6.6.1 Preprocessing

The preprocessing of an image which may contain characters of different sizes on several lines of text is performed in four steps:

- Using horizontal and vertical projection profiles (see section 2.11) the locations and sizes of the characters are determined.
- The characters are scaled to the same size.
- Using again the projection profiles the spaces between the characters and also the distances of lines are normalized.
- Text padding is used to fill empty spaces in the image.

6.6.2 Feature Extraction Using Multi-channel Gabor Filtering

Gabor filters (see section 2.13) that are used for feature extraction are defined as follows:

$$\begin{aligned} h_e(x,y) &= g(x,y)\cos\left[2\pi f(x\cos\phi + y\sin\phi)\right] \\ h_o(x,y) &= g(x,y)\sin\left[2\pi f(x\cos\phi + y\sin\phi)\right] \end{aligned}$$

where

$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{\left[-\frac{x^2+y^2}{2\sigma^2}\right]}$$

The spatial frequency responses are:

$$\begin{aligned} H_e(u,v) &= \frac{H_1(u,v) + H_2(u,v)}{2} \\ H_o(u,v) &= \frac{H_1(u,v) - H_2(u,v)}{2i} \end{aligned}$$

where

$$\begin{aligned} H_1(u,v) &= e^{-2\pi^2\sigma^2\left|(u-f\cos\phi)^2+(v-f\sin\phi)^2\right|} \\ H_2(u,v) &= e^{-2\pi^2\sigma^2\left|(u+f\cos\phi)^2+(v-f\sin\phi)^2\right|} \end{aligned}$$

f,ϕ and σ are spatial frequency, orientation and space constant of the Gabor envelope.

Four values for ϕ are used: $0^\circ, 45^\circ, 90^\circ, 135^\circ$. The central frequencies are chosen that they have the distance 1 octave from each other. The system receives input images of size 128 × 128 and uses the spatial frequencies of 4, 8, 16, and 32 for each of the four orientations. Therefore, the total number of Gabor channels is 16. The spatial constants σ of the channels are chosen to be inverse proportional to the central frequencies of the channels.

The mean values and standard deviations of the channel output images are chosen to represent texture features. Thus, a total of 32 features is extracted.

6.6.3 Font Recognition

For the classification the weighted Euclidian distance is used.

$$WED(k) = \sum_{i=1}^{N} \frac{(f_i - f_i^{(k)})^2}{(\delta_i^{(k)})^2}$$

where
f_i: i-th feature of the unknown font
$f_i^{(k)}$: i-th feature of font k
$\sigma_i^{(k)}$: standard deviation of the i-th feature of font k
N: total number of features

For six different fonts and four printing styles an accuracy of 98.6% was obtained.

Chapter 7

Architecture of Recognition System

This chapter presents the aim of the proposed system which was developed in this diploma thesis. It also contains the preprocessing, feature extraction and classification of the proposed system as shown in figure 7.1. The chapter describes the used databases and the results of the performed experiments. A result interpretation and comparison with other recognition systems is presented afterwards. Finally, a screenshot of the system's demo interface is presented.

7.1 Aim of the Proposed System

The aim in the development of the proposed system was to build a system that can recognize Chinese characters from the most frequent character set (3,755 characters). The system should especially be useful for characters which were captured from street signs in China.

The character recognition can be used as part of a system which shall help tourists to understand the meaning of Chinese street signs. A tourist can use a camera to capture a street sign and the recorded image is used as the system's input. The location of the characters is detected and the characters are segmented. In the next step a single character recognizer like the proposed system can be used to recognize the characters. In the last step, the extracted characters are translated into the tourist's language.

7.2 Preprocessing

In the preprocessing the character image is first scaled to a size of 64×64 pixels using the cubic spline interpolation (see subsection 2.1.3).

The character image is then converted from a grayscale image to a binary image using the first of the presented methods in section 2.2 which uses a threshold

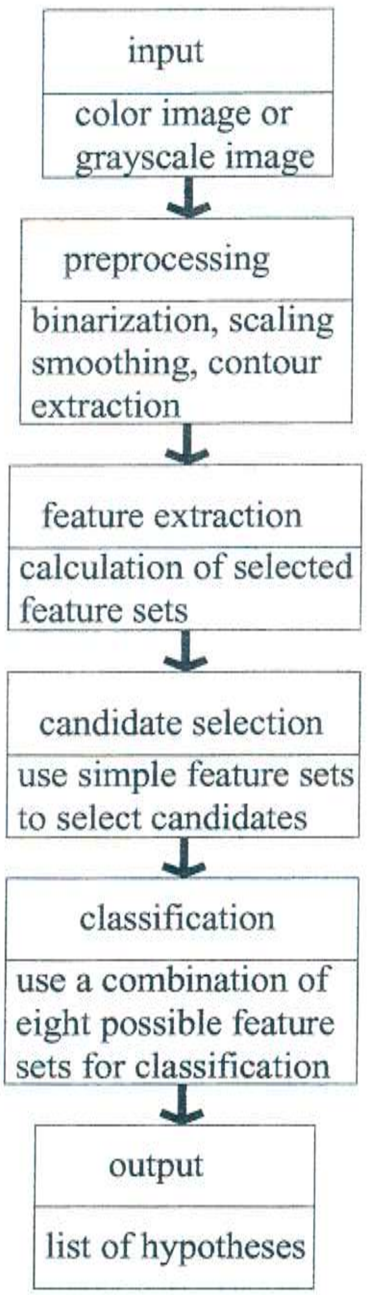

input
color image or grayscale image
preprocessing
binarization, scaling smoothing, contour extraction
feature extraction
calculation of selected feature sets
candidate selection
use simple feature sets to select candidates
classification
use a combination of eight possible feature sets for classification
output
list of hypotheses

Figure 7.1: System overview

calculated from the histogram. Alternatively, the last method for binarization from that section which uses an iterative process to convert the image can be used in cases with a nonconstant background.

After the binarization it is checked whether the image has to be inverted or not. This check is performed by using the average value of the edge pixels and by comparing the value with the average of the whole image. This inversion method is much more stable than using the average values of the graylevel image.

In the next step isolated points which are expected to be noise are removed from the image.

The scaling of the character to a given character size (64×64 pixels) is performed in two steps. At first, lines which only contain background information are removed approaching from top, bottom, left, and right. Then the image is scaled using the bi-linear interpolation (see subsection 2.1.2) and rebinarized. The bi-linear interpolation is used here, because it seems to be more appropriate for binary images than the cubic spline interpolation. Afterwards a black edge is added to avoid special cases in the feature calculation.

7.3 Feature Extraction

For the main classification up to 8 feature sets can be used.

The TBC features (see section 3.5) contain 81 feature values and are calculated after a thinning procedure has been performed. The Hilditch method (see section 2.4) is applied here.

The direction features (see section 3.6) contain 256 values. The edge enhancement procedure (see subsection 2.3.3) is used to make the features more stable against distortions.

Each of the ET1 and ET2 feature sets (see section 3.7) consists of 32 values. No special preprocessing is used for them.

The LDC features and the derived SP4 features (see sections 3.8 and 3.9) use the edge enhancement procedure to stabilize the character edges. They contain 64 and 24 values.

The BJD-BS features (see section 3.10) are extracted after the contour of the image has been calculated. This feature set has 32 values.

The WDH features (see section 3.11) use the edge enhancement and the image contour. These features consist of 256 values.

7.4 Classification

For the main classification the cityblock, Euclidian or angle distance can be used. For each used feature set the distances between the input features and the features for every example are calculated.

There are two methods to combine the feature sets.

bagging: From the recognition results for each feature set, a certain number c (i.e. 10) with the least distances is chosen and ordered by the distances. A score p_i is assigned to each example using the inverse of the individual distance d_i. The values are normed to sum up to one using the factor n.

$$p_i = \frac{n}{d_i} \tag{7.1}$$

$$n = \sum_{i=1}^{c} \frac{1}{d_i} \tag{7.2}$$

The scores of each of the feature sets are then combined using a bagging algorithm (see section 4.4). The scores calculated from the feature sets are summed and then again normed that their sum is 1. The values from the different feature sets can be weighted using individual weights for each feature set.

distance combination: The second method does not combine the results of the feature sets, but adds the distances of the feature sets. This addition is done by using the formula for the relative distance (see section 4.3). For the new distances scores for a certain number of characters are then assigned as in the previous method.

After the application of one of the above methods, the a priori probabilities of a character can be used to change the scores using formula (7.3).

$$p_i^* = p_i apriori(i)^{factor} \tag{7.3}$$

The factor is used to balance the influence of the a priori probabilities. For the recognition of the characters from the street sign a factor of 0.1 proved to be useful.

7.5 Acceleration Techniques

The TSL feature and the projection features (see sections 3.3 and 3.1) can be used for the preclassification of the characters which can especially be useful if many of the character sets are used. Another feature set which can be used for preclassification is a direction feature vector with 64 components. Each component is calculated as the sum of four neighboring values in the 256 feature vector.

The preclassification features are calculated for the input character and compared with the features for all classes. Afterwards a certain number of characters (i.e. 500) with the least distances is chosen. To save time the cityblock distance is used for the distance calculation.

7.6 Database

For the recognition system the first class of characters from the GB2312-80 standard was chosen, because these characters represent the most common Chinese characters and nearly all of the characters which are contained in the collected signs. For the system three examples of each character in the standard font were recorded. In the second recording the light conditions were changed and in the third another camera was used. The first example of each character was used for training while the other examples were used for testing. From the training example seven other variants were generated by rotation (-5,-2.5,+2.5,+5 degrees) and the application of closing, dilation and double application of dilation. The rotated variants were generated to make the features, especially the direction features, more stable against rotated input. The dilation operations were chosen to thicken the thin horizontal strokes from the standard font.

The figures 7.2, 7.3, 7.4 show examples for the three recordings after binarization.

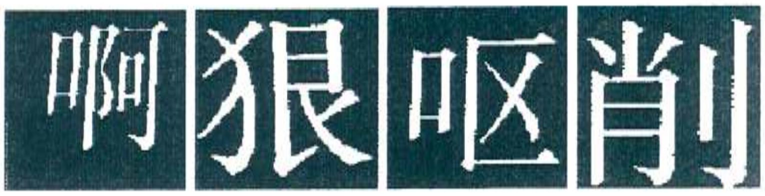


Figure 7.2: Examples from the first recording

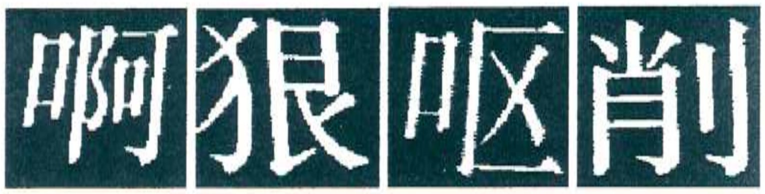


Figure 7.3: Examples from the second recording

The other database consists of color images which were extracted from street signs. The characters from this database were sorted into different problem areas.

no special problems: These characters (see figure 7.5) have no or only light distortions. (6218 characters)

background as mirror: The character background reflects something as shown in the examples in figure 7.6. (267 characters)

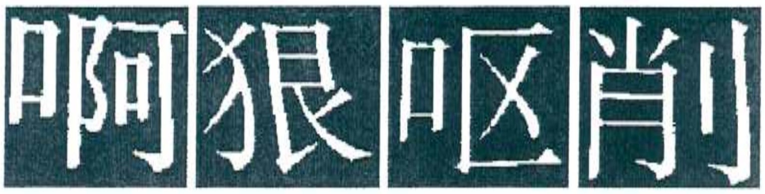


Figure 7.4: Examples from the third recording

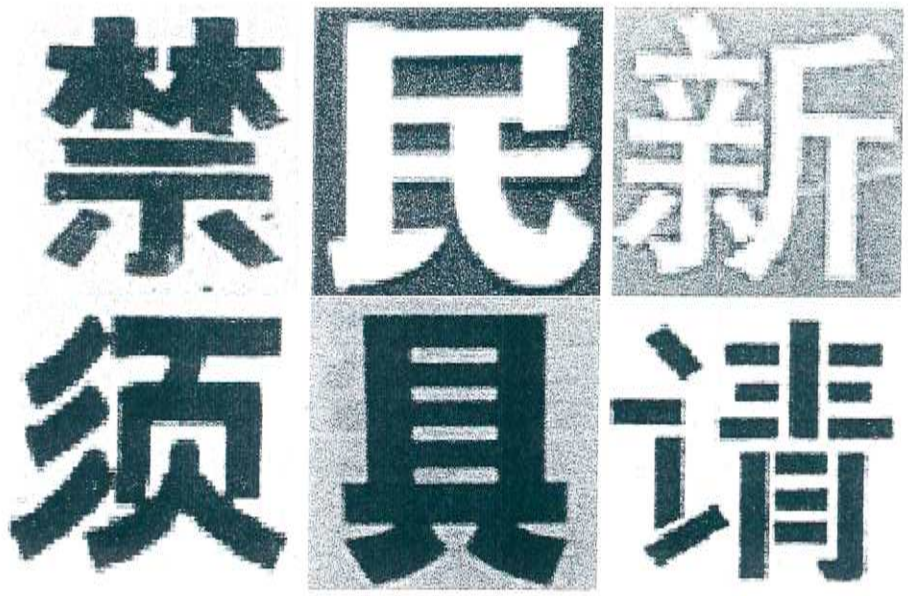


Figure 7.5: Examples of pictures with no special problems

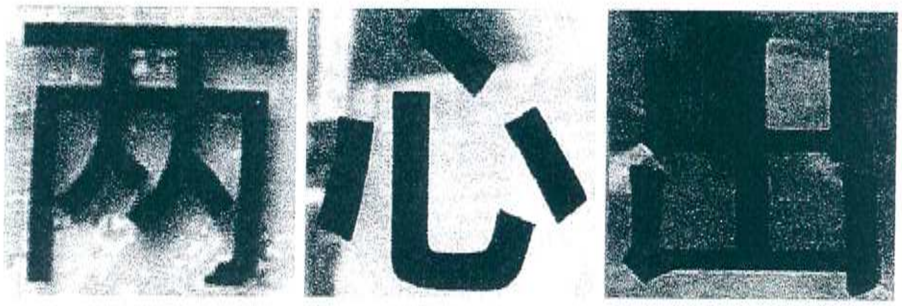


Figure 7.6: Examples with a reflection of the background

part missing: A part of the character is missing. (6 characters, see figure 7.7)

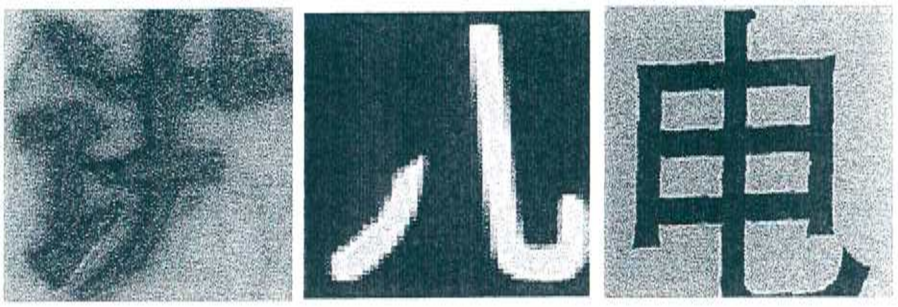


Figure 7.7: Examples with a missing part

part destroyed: Another object hides a part of the character. (59 characters, see figure 7.8)

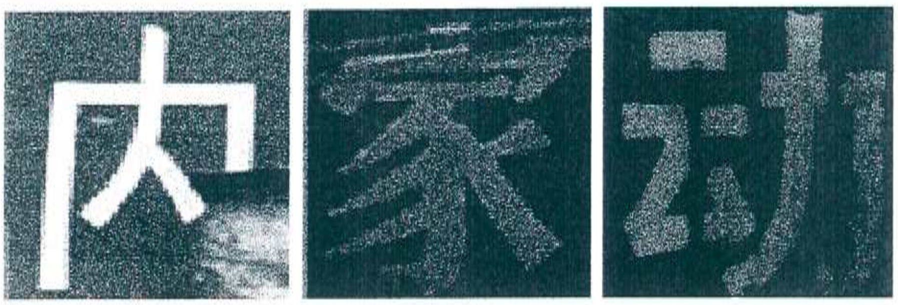


Figure 7.8: Examples with a destroyed part

rotation: The character is rotated as shown in the examples in figure 7.9. (146 characters)

small: The character is small and therefore some distortions appear after the scaling. (141 characters, see figure 7.10)

unsharp: The character image is unsharp. (64 characters, see figure 7.11)

same grayvalue: The background and foreground colors are converted into the same grayvalue which makes the distinction of the character from the background difficult. (72 characters, see figure 7.12)

light effects: A light shines onto the character and causes a nonconstant background. (360 characters, see figure 7.13)

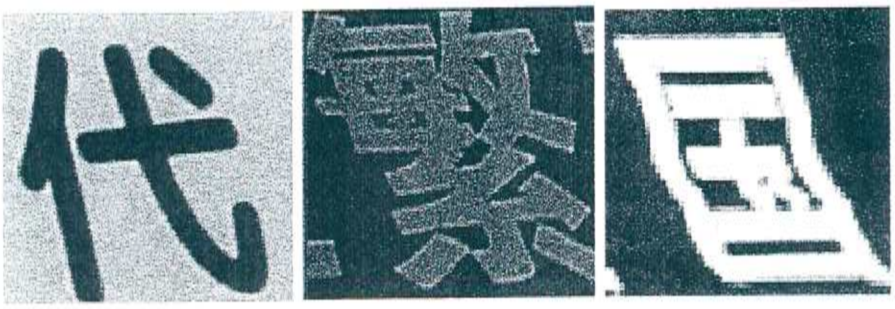


Figure 7.9: Rotated examples

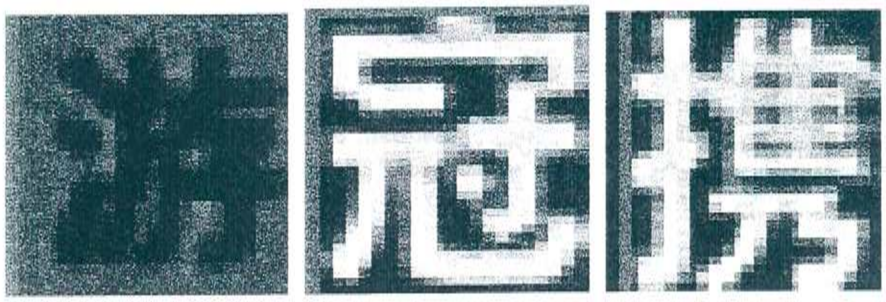


Figure 7.10: Examples of originally small characters

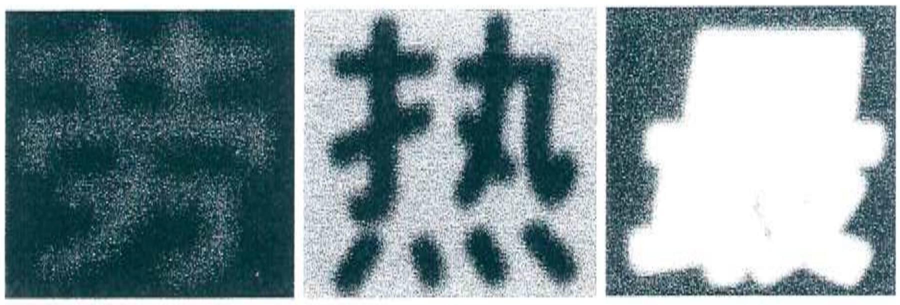


Figure 7.11: Examples of unsharp character images

noises: The image contains other kinds of noises, i.e. dirt. (344 characters, see figure 7.14)

uncommon font: The character is displayed in a special font. (527 characters, see figure 7.15)

Figure 7.12: Examples with the color conversion problem

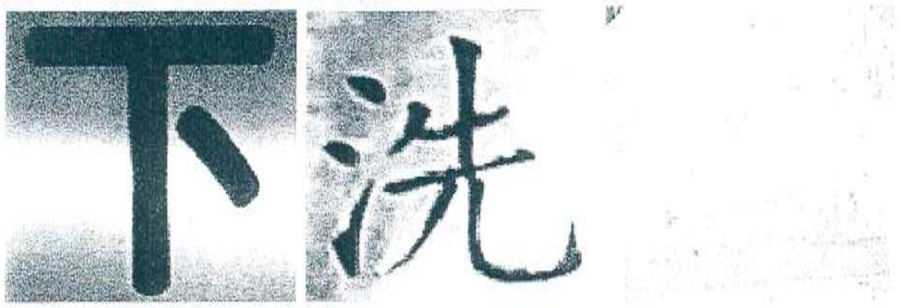


Figure 7.13: Examples where a light shines onto the character image

23 of a total of 8171 characters did not belong to the chosen character set (GB 2312-80, class 1).

Another set of character was extracted by Jing Zhang from the recorded images.

7.7 Results

This section uses several tables to display the results for the performed experiments. The interpretation of the experiments can be found in the next section. The abbreviations "bag" and "dis" are used to distinguish between the usage of the bagging algorithm or the distance combination for the recognition with multiple feature sets.

Tables 7.1, 7.2 show the results for different light conditions.

The next tables (tables 7.3, 7.4) show the results for the characters which were recorded using the other camera. The first camera is from LG, model LVC-M100NM and the second is from Sun, Model IK-M28SA.

The results in the following tables (tables 7.5, 7.6, 7.7) were gained for the

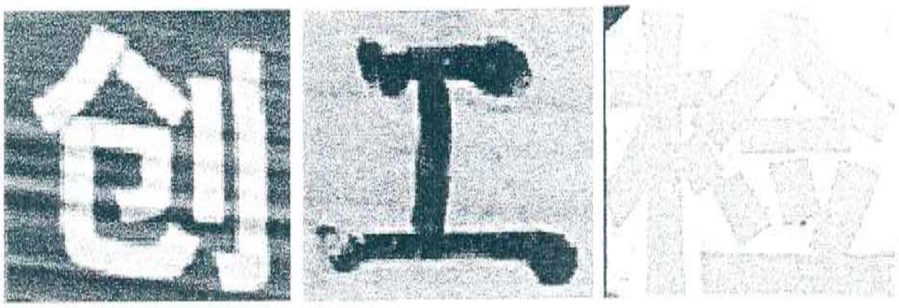


Figure 7.14: Examples of noisy characters

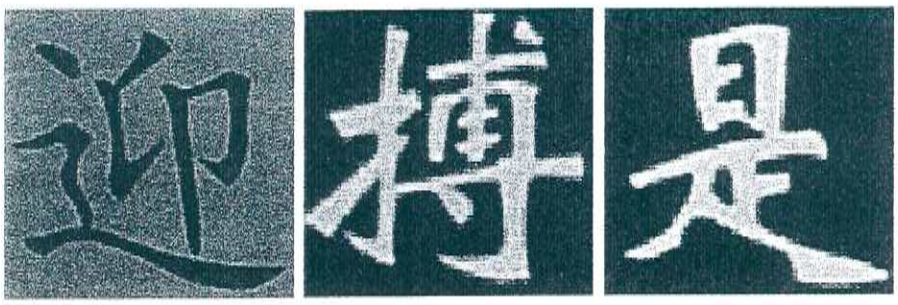


Figure 7.15: Examples of uncommon fonts

Table 7.1: Results for different light conditions using the Euclidian distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	51.50	61.49	67.11	70.04	72.38	74.22	75.69	77.04	78.35	79.17
Direction	99.65	99.84	99.95	99.95	99.97	99.97	99.97	100.0	100.0	100.0
ET1	90.60	97.07	96.46	97.02	97.44	97.92	98.11	98.35	98.48	98.64
ET2	98.45	99.25	99.52	99.63	99.68	99.71	99.73	99.79	99.84	99.89
LDC	95.47	97.82	98.32	98.80	99.07	99.23	99.28	99.31	99.39	99.44
SP4	84.10	89.77	91.82	93.05	94.11	94.67	95.13	95.42	95.71	96.22
BJD	83.22	90.81	93.40	94.88	95.71	96.32	96.80	97.07	97.28	97.44
WDH	92.68	95.93	96.86	97.42	97.71	97.95	98.16	98.22	98.40	98.45
Dir, ET2, WDH (bag)	99.73	99.97	99.97	99.97	100.0	100.0	100.0	100.0	100.0	100.0
all (bag)	99.57	99.97	99.97	99.97	100.0	100.0	100.0	100.0	100.0	100.0
Dir, ET2, WDH (dis)	99.76	99.95	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
all (dis)	99.71	99.97	99.97	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 7.2: Results for different light conditions using the angle distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	52.94	63.54	68.78	71.74	74.03	76.19	77.42	78.88	80.13	81.12
Direction	99.73	99.84	99.89	99.89	99.89	99.92	99.92	99.95	99.97	99.97
ET1	90.31	95.07	96.27	96.88	97.26	97.58	97.90	98.14	98.38	98.56
ET2	98.40	99.04	99.44	99.55	99.68	99.71	99.79	99.82	99.84	99.84
LDC	95.50	97.95	98.51	98.85	99.01	99.25	99.28	99.39	99.44	99.49
SP4	82.74	89.21	91.45	92.65	93.52	94.17	94.73	95.13	95.34	95.74
BJD	83.25	90.89	93.29	94.86	95.74	96.30	96.86	97.07	97.31	97.50
WDH	93.26	96.54	97.63	98.32	98.64	98.74	98.82	98.93	99.01	99.07
Dir, ET2, WDH (bag)	99.68	99.92	99.97	100.0	100.0	100.0	100.0	100.0	100.0	100.0
all (bag)	98.93	99.89	99.95	99.97	100.0	100.0	100.0	100.0	100.0	100.0
Dir, ET2, WDH (dis)	99.95	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
all (dis)	99.76	99.95	99.95	99.97	99.97	99.97	99.97	99.97	99.97	99.97

Table 7.3: Results for the other camera using the Euclidian distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	47.64	57.92	62.93	66.42	68.81	70.65	72.20	73.66	74.70	75.82
Direction	99.25	99.87	99.89	99.89	99.89	99.92	99.95	99.95	99.95	99.95
ET1	83.28	89.53	91.50	92.84	93.74	94.41	95.05	95.34	95.71	96.01
ET2	96.01	98.48	99.01	99.12	99.28	99.44	99.49	99.49	99.55	99.57
LDC	93.18	96.51	97.55	98.03	98.59	98.80	99.01	99.20	99.28	99.36
SP4	83.14	90.36	92.52	93.90	94.81	95.39	95.82	96.40	96.70	96.88
BJD	77.07	86.95	90.52	92.38	93.37	94.03	94.59	95.02	95.53	95.85
WDH	85.19	91.11	93.21	94.35	95.07	95.61	96.06	96.25	96.43	96.43
Dir, ET2, WDH (bag)	98.85	99.73	99.92	99.92	99.92	99.92	99.95	99.95	99.97	99.97
Dir, ET2, WDH (dis)	99.76	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
all (dis)	99.84	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 7.4: Results for the other camera using the angle distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	49.91	60.48	66.10	69.72	72.57	74.33	76.09	77.52	78.54	79.44
Direction	99.39	99.81	99.84	99.84	99.87	99.87	99.89	99.89	99.89	99.89
ET1	82.88	89.35	91.48	92.92	93.69	94.19	94.67	95.10	95.53	95.66
ET2	95.74	98.32	98.93	99.09	99.31	99.36	99.41	99.52	99.57	99.65
LDC	93.34	96.51	97.76	98.30	98.70	98.93	99.15	99.28	99.39	99.49
SP4	81.54	89.29	91.85	93.02	94.11	94.83	95.26	95.82	96.01	96.54
BJD	77.17	86.98	90.49	92.49	93.42	94.03	94.70	95.10	95.61	95.98
WDH	87.62	93.08	94.75	95.87	96.35	96.64	97.10	97.50	97.68	97.87
Dir, ET2, WDH (bag)	98.70	99.57	99.73	99.81	99.87	99.87	99.89	99.89	99.92	99.92
Dir, ET2, WDH (dis)	99.95	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
all (dis)	99.84	99.97	99.97	99.97	99.97	99.97	99.97	99.97	99.97	99.97

Table 7.5: Results for characters with no special problems using the Euclidian distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	20.59	27.55	32.55	35.98	38.47	40.73	42.83	43.83	45.18	46.49
Direction	76.67	84.86	88.22	90.02	91.14	92.06	92.49	92.87	93.24	93.50
ET1	30.07	38.11	42.23	45.23	47.28	49.24	50.48	51.80	52.72	53.70
ET2	58.74	67.96	72.25	75.47	77.55	79.02	80.60	81.62	82.32	83.07
LDC	54.39	64.50	68.94	72.48	75.27	76.87	78.23	79.36	80.17	80.85
SP4	27.12	34.81	39.05	42.51	45.40	47.64	49.57	50.87	51.95	53.07
BJD	41.67	52.44	58.35	62.54	65.03	67.29	69.00	70.50	71.81	72.93
WDH	77.55	85.85	89.06	90.83	91.78	92.59	93.17	93.73	94.11	94.47
Dir, ET2, WDH (bag)	82.66	89.36	91.62	92.99	93.85	94.44	95.07	95.48	95.81	95.96
all (bag)	82.18	89.79	92.13	93.53	94.26	94.61	94.95	95.38	95.69	95.97
Dir, ET2, WDH (dis)	85.32	91.31	93.32	94.33	94.72	95.10	95.45	95.78	96.01	96.21
Dir, ET1, ET2, LDC, BJD, WDH (dis)	86.14	91.80	93.50	94.49	94.90	95.30	95.58	95.89	96.16	96.29
all (dis)	85.19	91.19	92.76	93.78	94.28	94.79	95.05	95.32	95.61	95.81
Dir, ET2, WDH (dis*)	85.60	91.13	93.07	94.21	94.77	95.17	95.40	95.69	95.96	96.21
all (dis*)	86.54	91.55	93.32	94.19	94.99	95.36	95.69	95.96	96.02	96.17

characters from the street signs with no special problems. The abbreviation (dis*) means that the distance combination was used and the distances from each feature set were weighted by the recognition accuracy of each feature set for the recognition of the data extracted by Jing Zhang.

The next three tables (tables 7.8, 7.9, 7.10) show the results for the data that were extracted by Jing Zhang. The abbreviation (dis*) stands for a special weighting of the distances according to the feature accuracies for the recognition of the characters without special problems.

For the characters which are quite small it was tested whether the scaling with a cubic spline interpolation yields better results than scaling with the bilinear interpolation. As features the combination of direction, ET2 and WDH was used (see tables 7.11, 7.12).

The next experiment was performed to compare the adaptive thresholding technique to the histogram method. The data where a spotlight shines onto the character image were used to perform this experiment, because these data cause many problems for the histogram method. Again, the direction, ET2 and WDH features were used in this experiment (see tables 7.13, 7.14).

Using the adaptive threshold for the data extracted by Jing Zhang (jing) and for the characters without special problems (normal) the following results (see tables 7.15, 7.16) were obtained. Here the distance combination of the direction, ET2 and WDH features were used.

When converting the color images to grayscale images, there is sometimes a

Table 7.6: Results for characters with no special problems using the angle distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	17.06	22.75	26.62	29.79	31.90	33.52	34.94	36.19	37.08	38.21
Direction	71.20	80.07	83.22	85.17	86.61	87.41	88.22	88.96	89.39	89.95
ET1	29.54	37.17	40.91	44.00	46.65	48.33	49.69	51.04	51.90	52.47
ET2	56.48	66.22	70.67	73.54	75.78	77.33	78.80	79.79	80.81	81.56
LDC	54.35	64.53	69.10	72.39	75.32	76.95	78.14	79.25	80.07	80.68
SP4	25.87	33.95	38.17	41.70	44.14	46.42	48.02	49.46	50.58	51.95
BJD	41.82	52.67	58.56	62.75	65.28	67.47	69.23	70.69	71.94	73.18
WDH	71.71	80.98	84.74	86.44	87.78	88.78	89.59	90.30	90.73	91.27
Dir, ET2, WDH (bag)	80.09	86.84	89.38	90.80	91.93	92.81	93.26	93.85	94.16	94.49
all (bag)	76.34	85.35	88.72	90.53	91.75	92.69	93.17	93.65	94.09	94.47
Dir, WDH (dis)	77.14	84.91	88.16	89.64	90.58	91.42	91.97	92.49	92.79	93.07
Dir, ET2, WDH (dis)	83.44	89.71	91.59	92.76	93.47	93.98	94.24	94.54	94.77	94.95
Dir, ET1, ET2, LDC, BJD, WDH (dis)	86.11	91.52	93.06	93.96	94.56	94.99	95.36	95.60	95.78	95.86
all (dis)	81.95	88.04	90.68	91.72	92.20	92.79	93.15	93.39	93.65	93.93

Table 7.7: Results for characters with no special problems using the cityblock distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	20.31	27.85	32.81	35.96	38.16	39.97	41.70	43.32	44.67	45.93
Direction	74.25	83.26	87.08	89.56	90.88	91.70	92.43	92.87	93.25	93.48
ET1	33.78	42.35	46.57	49.90	52.21	54.11	55.74	57.04	58.13	59.11
ET2	61.94	71.54	75.92	78.52	80.30	82.02	83.27	84.28	84.84	85.53
LDC	57.64	67.83	72.78	76.01	78.49	80.30	81.54	82.76	83.65	84.53
SP4	27.65	35.47	39.95	43.35	45.94	48.04	49.90	51.65	53.02	54.24
BJD	38.77	49.98	56.47	60.29	63.39	65.89	67.83	69.50	71.08	72.15
WDH	75.60	85.02	88.50	90.25	91.42	92.30	92.86	93.45	93.95	94.26
Dir, WDH (dis)	82.30	89.51	92.28	93.47	94.26	94.85	95.58	95.78	96.11	96.30
Dir, ET2, WDH (dis)	85.30	91.64	93.55	94.62	95.15	95.50	95.84	96.06	96.26	96.50
Dir, ET2, WDH (dis*)	85.32	91.62	93.48	94.54	95.07	95.53	95.94	96.11	96.29	96.40
all (dis*)	86.77	92.20	93.96	95.02	95.53	95.99	96.24	96.40	96.55	96.68

Table 7.8: Results for the data extracted by Jing Zhang using the Euclidian distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	20.82	28.59	32.02	36.36	39.79	41.63	43.61	44.80	45.59	47.04
Direction	84.45	91.17	93.28	94.99	96.57	97.10	97.23	97.36	97.36	98.02
ET1	35.44	42.82	47.83	51.52	53.89	55.99	57.58	59.03	60.34	61.26
ET2	60.47	71.14	77.07	80.11	82.61	83.27	84.72	86.17	87.09	88.01
LDC	58.76	67.33	71.81	74.18	76.68	77.60	78.92	79.97	80.76	81.69
SP4	22.66	30.96	35.44	38.47	40.84	43.74	45.32	47.43	48.48	50.20
BJD	45.45	56.39	61.26	66.53	68.91	71.28	72.99	74.84	75.63	76.55
WDH	82.08	90.25	92.89	94.60	95.26	95.65	96.18	96.71	96.84	96.97
Dir, WDH (bag)	87.35	93.94	96.18	97.50	98.29	98.55	98.81	99.08	99.08	99.08
Dir, ET2, WDH (bag)	88.14	93.68	94.86	96.18	96.84	97.50	98.02	98.16	98.55	98.68
Dir, ET2, LDC, WDH (bag)	85.64	91.57	95.13	96.18	97.10	97.36	97.63	98.02	98.42	98.55
all (bag)	84.06	92.09	95.65	96.71	97.36	97.63	98.29	98.29	98.55	98.55
Dir, WDH (dis)	88.41	94.33	96.44	97.63	98.02	98.29	98.55	98.68	98.95	99.34
Dir, ET2, WDH (dis)	90.25	94.86	97.10	98.02	98.42	98.68	98.81	99.08	99.08	99.21
Dir, ET2, WDH (dis*)	89.20	95.39	97.63	99.08	99.34	99.34	99.34	99.34	99.47	99.47
Dir, ET1, ET2, LDC, BJD, WDH (dis*)	89.86	95.39	97.50	98.68	98.95	98.95	98.95	98.95	98.95	98.95
all (dis*)	90.12	95.52	97.50	98.68	98.81	98.95	98.95	98.95	98.95	98.95

Table 7.9: Results for the data extracted by Jing Zhang using the angle distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	18.71	25.43	29.91	33.33	35.57	37.81	40.18	41.50	42.82	43.48
Direction	81.16	88.14	90.51	92.09	93.02	93.94	94.33	94.86	95.65	95.92
ET1	34.12	40.97	46.25	50.46	52.44	54.68	55.86	57.58	59.16	60.48
ET2	60.08	70.62	75.49	79.31	81.82	84.06	85.11	85.77	86.43	86.96
LDC	59.03	67.19	71.81	74.70	76.68	77.34	78.52	79.71	80.50	81.69
SP4	21.48	28.46	33.20	36.23	38.74	41.24	43.21	45.85	47.04	48.35
BJD	45.59	56.52	61.53	66.53	68.91	71.54	73.52	75.10	76.02	76.81
WDH	75.49	87.62	90.12	91.44	92.75	94.20	94.86	95.78	96.18	96.18
Dir, WDH (bag)	82.87	92.09	94.33	95.52	96.31	96.71	97.23	97.63	97.89	98.02
Dir, ET2, WDH (bag)	86.43	93.41	94.73	95.26	96.44	96.44	97.36	97.89	98.16	98.29
Dir, ET2, LDC, WDH (bag)	84.32	91.04	93.41	94.86	95.65	96.97	97.36	97.36	97.63	97.76
all (bag)	81.42	89.20	92.36	94.33	95.92	96.84	97.76	97.89	98.02	98.02
Dir, WDH (dis)	84.85	92.09	94.99	95.78	96.57	96.71	97.36	97.76	97.76	98.02
Dir, ET2, WDH (dis)	88.93	94.86	96.44	96.84	97.50	98.02	98.16	98.16	98.29	98.42
Dir, ET2, LDC, WDH (dis)	88.41	93.41	95.78	97.23	97.50	97.76	97.76	97.89	97.89	98.02
Dir, ET1, ET2, WDH (dis)	86.83	93.02	94.99	96.05	97.63	97.63	97.76	97.89	98.02	98.42
ET1, ET2, SP4, BJD (dis)	78.40	86.96	89.06	91.30	92.62	93.68	94.20	94.73	95.13	95.39
Dir, ET1, ET2, LDC, BJD, WDH (dis)	89.06	93.94	96.31	97.10	97.89	97.89	98.29	98.42	98.42	98.42
Dir, ET1, ET2, BJD, WDH (dis)	88.67	93.68	96.44	97.10	97.89	98.02	98.29	98.42	98.42	98.42
all (dis)	86.39	93.02	95.39	96.44	96.97	97.50	97.50	97.76	97.89	97.89

Table 7.10: Results for the data extracted by Jing Zhang using the cityblock distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	21.08	28.19	32.28	35.97	39.00	41.37	42.42	43.74	45.45	46.11
Direction	82.21	91.17	93.28	95.26	96.97	97.23	97.50	97.50	98.29	98.42
ET1	38.74	48.89	53.49	57.71	59.16	60.74	62.58	63.90	65.09	66.40
ET2	65.22	75.63	79.18	82.61	84.58	86.82	88.14	89.20	89.72	90.25
LDC	60.87	70.62	77.47	79.58	81.29	82.35	83.40	83.93	85.24	86.17
SP4	23.06	31.36	35.84	39.39	42.69	45.06	46.38	48.22	49.54	51.25
BJD	41.63	54.15	60.61	64.43	66.27	69.30	71.15	73.39	74.84	76.02
WDH	79.97	90.51	93.94	94.47	94.86	95.78	96.31	96.57	96.97	97.23
Dir, WDH (bag)	88.01	95.26	97.36	98.42	99.21	99.34	99.47	99.47	99.47	99.74
Dir, ET2, WDH (bag)	88.27	94.86	96.31	97.76	98.02	98.16	98.68	99.08	99.21	99.34
Dir, ET2, LDC, WDH (bag)	87.88	93.54	95.13	96.97	97.63	98.29	98.42	98.68	99.08	99.08
all (bag)	84.45	92.09	95.39	96.18	97.23	97.50	97.63	98.42	98.81	98.81
Dir, WDH (dis)	88.01	94.60	97.36	98.16	98.55	99.08	99.21	99.34	99.34	99.34
Dir, ET2, WDH (dis)	90.91	96.18	97.63	98.68	99.08	99.34	99.47	99.60	99.60	99.87
Dir, ET2, LDC, WDH (dis)	90.25	95.52	97.63	98.16	98.55	98.95	99.08	99.34	99.47	99.60
Dir, ET1, ET2, WDH (dis)	90.51	95.26	96.84	97.36	98.29	98.55	98.81	98.81	98.95	98.95
ET1, ET2, SP4, BJD (dis)	80.11	87.75	90.25	92.36	93.28	94.60	95.52	96.05	96.18	96.71
Dir, ET1, ET2, LDC, BJD, WDH (dis)	91.57	95.78	97.10	98.16	98.42	98.68	98.81	98.81	98.81	98.81
Dir, ET1, ET2, BJD, WDH (dis)	90.78	96.18	97.10	98.02	98.55	98.68	98.81	98.81	98.81	98.81
all (dis)	90.91	95.92	97.36	98.02	98.42	98.55	98.68	98.68	98.68	98.68
Dir, ET2, WDH (dis*)	88.67	96.44	97.89	98.68	99.34	99.60	99.60	99.60	99.74	99.87
Dir, ET1, ET2, LDC, BJD, WDH(dis*)	90.25	96.05	98.02	98.55	98.68	98.95	99.08	99.08	99.21	99.21
all (dis*)	90.51	96.57	97.76	98.55	98.81	99.08	99.21	99.21	99.21	99.21

Table 7.11: Results for the small characters using the Euclidian distance

interpolation	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
bilinear	55.32	62.41	65.25	68.79	72.34	73.76	75.89	76.60	77.30	78.01
cubic spline	72.34	78.01	80.14	81.56	83.69	83.69	84.40	85.82	85.82	87.23

Table 7.12: Results for the small characters using the angle distance

interpolation	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
bilinear	43.97	64.61	61.70	64.54	65.25	65.25	67.38	67.38	69.50	71.63
cubic spline	70.21	75.89	79.43	81.56	82.98	84.40	84.40	84.40	84.40	85.82

Table 7.13: Results for the spotlight characters using the Euclidian distance

binarization	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
histogram	30.28	41.11	45.83	50.83	54.72	57.50	60.83	62.22	63.33	63.89
adaptive	39.44	47.22	53.06	58.89	61.67	63.06	65.00	65.83	66.39	67.78

Table 7.14: Results for the spotlight characters using the angle distance

binarization	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
histogram	25.00	31.67	35.28	38.61	43.61	44.72	46.67	47.50	48.89	49.72
adaptive	35.56	41.94	45.83	49.44	51.67	53.89	56.39	56.94	58.06	58.06

Table 7.15: Results for the adaptive thresholding using the Euclidian distance

data set	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
jing	89.06	95.78	98.02	98.81	99.08	99.21	99.21	99.21	99.21	99.47
normal	85.75	91.21	93.19	94.19	94.64	95.03	95.32	95.56	95.88	96.06

Table 7.16: Results for the adaptive thresholding using the angle distance

data set	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
jing	88.67	94.99	97.36	98.02	98.29	98.68	98.68	98.68	98.68	98.81
normal	83.69	89.18	91.13	92.36	93.09	93.63	94.00	94.34	94.64	94.79

Table 7.17: Results for several color conversion methods using the Euclidian distance

conversion	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
average	18.06	25.00	25.00	29.17	31.94	31.94	31.94	31.94	31.94	33.33
NTSC	16.67	18.06	25.00	29.17	31.94	31.94	34.72	36.11	37.50	37.50
PAL	25.00	29.17	31.94	36.11	37.50	37.50	37.50	37.50	37.50	38.89

Table 7.18: Results for several color conversion methods using the angle distance

conversion	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
average	18.06	19.44	20.83	22.22	23.61	23.61	25.00	26.39	26.39	26.39
NTSC	13.89	15.28	20.83	22.22	22.22	23.61	23.61	23.61	23.61	23.61
PAL	23.61	26.39	26.39	26.39	26.39	26.39	29.17	29.17	30.56	31.94

problem that different colors are mapped onto the same gray value. Therefore it was tested which conversion method from color to grayscale is best suited to avoid such problems (see section 2.10). The test was performed on the characters with such a problem using the combination of Direction, ET2 and WDH features. The results are shown in tables 7.17, 7.18.

To test the stability of the features for noises, a test was performed for the noisy characters. See tables 7.19, 7.20 for results.

The tables 7.21, 7.22 show the results when applying the Linear Discriminant Analysis (LDA) for several features for the data extracted by Jing Zhang.

The following experiments had the aim to test the application of the Simple Genetic Algorithm for the feature selection and weighting as proposed in section 3.12. To save time in the fitness evaluation process, the cityblock distance was used and a preclassification with a direction feature vector with 64 entries was used. Table 7.23 shows the results for the feature selection inside a feature set, while table 7.24 shows the effect of the weighting. For the experiments the data from Jing Zhang were used for training and the characters with no special problems were used for testing.

Tables 7.25, 7.26 show the results when using a priori probabilities in the recognition process. The accuracies relate to the recognition of the images with no special problems with a combination of the direction, ET2 and WDH features.

In the next experiment (see tables 7.27, 7.28) the data extracted by Jing Zhang were added to the training set and a test was performed onto the data with no special problems using the direction, ET2 and WDH features.

The next series of experiments had the aim to test the use of preclassification to accelerate the recognition process. The table 7.29 shows the results when using the TSL feature in preclassification. The left column shows the number

Table 7.19: Results for the noisy characters using the Euclidian distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	7.85	11.92	14.53	16.28	18.02	19.48	20.93	22.67	22.97	23.55
Direction	48.55	57.56	62.79	65.70	67.44	68.60	70.64	72.67	73.26	73.84
ET1	12.50	17.44	21.22	25.00	27.03	28.20	29.94	30.52	30.81	31.98
ET2	25.00	30.52	33.72	35.47	36.63	38.37	39.53	41.57	43.31	43.90
LDC	27.33	32.27	35.17	38.08	39.24	41.28	43.32	45.06	46.80	47.09
SP4	14.53	17.73	21.51	22.96	26.16	26.74	29.36	30.23	31.98	33.14
BJD	16.57	21.80	26.45	28.49	29.36	30.81	31.69	32.85	34.01	34.59
WDH	49.71	59.88	62.79	64.83	66.86	68.60	69.19	70.06	70.64	71.22
Dir, WDH (bag)	54.94	64.83	69.77	71.80	74.42	75.29	76.74	77.03	78.78	79.07
Dir, ET2, WDH (bag)	47.09	57.85	63.66	68.31	72.09	75.00	76.16	77.33	78.78	78.78
Dir, WDH (dis)	55.81	64.24	67.73	71.51	73.84	75.87	76.45	77.03	78.20	78.49
Dir, ET2, WDH (dis)	53.78	66.57	69.48	70.64	71.22	74.13	75.29	76.45	77.62	78.49

Table 7.20: Results for the noisy characters using the angle distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
TBC	5.52	6.69	9.30	9.88	10.17	10.47	11.92	12.50	13.37	14.24
Direction	39.83	46.51	52.33	55.23	58.14	61.05	62.50	63.37	63.66	64.53
ET1	11.92	17.73	19.48	22.97	27.03	28.78	29.94	30.23	31.10	31.98
ET2	22.38	29.36	33.72	35.17	36.92	38.37	39.24	40.41	41.57	43.02
LDC	27.03	31.69	35.17	37.79	39.53	41.28	43.02	44.19	46.22	46.80
SP4	13.37	17.15	20.93	22.38	24.42	24.71	27.03	28.49	30.23	30.81
BJD	17.15	22.38	27.33	28.78	29.36	30.81	31.40	32.85	34.30	34.30
WDH	41.28	51.16	55.52	58.43	59.59	61.63	63.37	64.24	65.12	65.41
Dir, WDH (bag)	47.97	54.94	57.56	60.47	61.63	62.21	63.95	64.24	65.12	66.86
Dir, ET2, WDH (bag)	44.48	54.65	59.30	62.79	65.12	65.99	67.15	69.48	69.48	70.06
Dir, WDH (dis)	46.80	55.23	59.30	63.08	64.83	66.28	67.15	68.02	69.19	69.77
Dir, ET2, WDH (dis)	53.20	59.88	63.08	64.83	66.86	68.02	68.90	69.77	70.35	70.93

Table 7.21: Results for the application of LDA using the Euclidian distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
Direction	14.10	17.13	19.76	20.95	22.53	24.11	25.03	25.82	26.48	27.27
WDH	54.68	68.12	72.07	75.10	77.60	79.45	80.24	81.55	82.48	83.27

Table 7.22: Results for the application of LDA using the angle distance

feature	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
Direction	16.47	20.03	23.19	26.09	27.01	28.06	28.99	30.04	30.43	31.36
WDH	51.25	62.85	68.64	72.07	74.18	75.36	76.81	78.52	79.58	80.63

Table 7.23: Results for using feature selection for several feature sets

features	chosen	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
Direction	219	73.57	82.76	86.54	88.55	89.74	90.70	91.47	91.93	92.35	92.71
ET1	30	38.98	49.39	55.10	58.88	61.84	64.37	66.02	67.80	69.23	70.31
ET2	31	63.89	74.02	78.36	81.38	83.14	84.38	85.55	86.23	87.00	87.66
LDC	56	57.56	69.37	74.61	77.75	80.44	82.37	83.49	84.51	85.30	86.19
WDH	227	75.93	85.07	88.19	89.54	90.58	91.44	92.13	92.61	93.10	93.39

Table 7.24: Results for using feature weighting for several feature sets

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
Direction	74.27	83.88	87.36	89.38	90.63	91.52	91.95	92.51	92.82	93.24
ET1	38.90	49.14	55.13	59.01	61.79	64.27	66.12	67.72	69.02	70.36
ET2	63.64	73.49	78.39	81.05	82.89	84.25	85.35	86.28	86.85	87.63
LDC	58.33	69.53	74.91	78.32	80.72	82.38	83.85	84.72	85.62	86.41
WDH	76.18	85.12	88.32	89.66	90.56	91.52	92.15	92.73	93.17	93.53

Table 7.25: Results for different weighting factors using the Euclidian distance

factor	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
0	85.32	91.31	93.32	94.33	94.72	95.10	95.45	95.78	96.01	96.21
0.05	85.93	91.80	94.00	94.84	95.46	95.83	96.04	96.32	96.52	96.62
0.07	85.93	91.98	94.08	94.92	95.46	95.81	96.11	96.30	96.44	96.60
0.1	85.40	91.83	93.96	94.90	95.33	95.78	95.94	96.17	96.37	96.54

Table 7.26: Results for different weighting factors using the angle distance

factor	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
0	83.44	89.71	91.59	92.76	93.47	93.98	94.24	94.54	94.77	94.95
0.05	84.38	90.73	92.76	93.75	94.33	94.75	95.10	95.40	95.50	95.66
0.07	84.39	90.60	92.87	94.03	94.49	94.89	95.18	95.43	95.55	95.69
0.1	84.00	90.71	92.73	93.71	94.49	94.94	95.17	95.36	95.50	95.73

Table 7.27: Results for training with video images using the Euclidian distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
Dir, ET2, WDH (dis)	89.05	93.68	94.90	95.58	95.97	96.26	96.37	96.52	96.65	96.77

Table 7.28: Results for training with video images using the angle distance

features	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
Dir, ET2, WDH (dis)	87.96	92.36	93.70	94.39	94.97	95.30	95.43	95.63	95.81	95.97

of characters that were selected for the fine classification, while the next column shows the average recognition time (seconds, centiseconds). Here the data from Jing Zhang, the direction, ET2 and WDH features and the cityblock distance were used. The used computer has an Intel Pentium III processor with a processor speed of 867 MHz and 512 RAM.

Table 7.30 shows the results when using the projection profiles for preclassification.

Table 7.31 shows the results under the same conditions when using the direction feature vector with 64 entries for preclassification.

The next experiment had the aim to combine the recognizer developed by Jing Zhang and the proposed recognizer. For the combination the distances were combined according the distance addition procedure. Experiments were performed to determine a special weighting factor (0: proposed system, 1: Jing Zhang's system). For the proposed system the direction, ET2 and WDH features and the cityblock distance were used. Table 7.32 shows the results for the data extracted by Jing Zhang.

Table 7.29: Results for several numbers in preclassification with the TSL feature

number	time	1-best	2-best	3-best	4-best	5-best	6-best	7-best	8-best	9-best	10-best
3755	1.34	90.91	96.18	97.63	98.68	99.08	99.34	99.47	99.60	99.60	99.87
2000	0.95	83.93	88.93	90.51	90.91	91.30	91.83	92.09	92.23	92.36	92.36
1000	0.65	67.98	71.01	71.54	71.81	72.07	72.20	72.20	72.33	72.33	72.46
500	0.49	46.90	48.48	48.88	48.88	49.14	49.14	49.14	49.28	49.28	49.54

7.8 Result Interpretation

Table 7.8 shows the results for the use of different feature sets and lists some problems of the features. The results show that the direction features yield the best results for the first three data sets, while the WDH features have the best result for the characters with no special problems. The WDH features seem to be quite robust, since they do not show such a performance degradation than other feature sets. Even though the ET1 and ET2 features are quite simple, both feature sets, especially the ET2 features, yield quite good results for all tasks and can be used as addition to the direction and WDH features. The LDC features also proved to be useful, even though their performance is weaker than the one of the ET2 features. The performance of the SP4 and BJD-BS features was quite disappointing. For each of the feature sets it was claimed that the features should be quite robust against noises, but the performance degraded very much when using the sets for the data from the street signs. The TBC features do not have the discriminativity for the recognition task, because there are characters which have the same terminal, branch and crossing points. But these features can be used in a feature combination.

feature	simple1	simple2	jing	sign	problems
TBC	52.94	49.91	21.08	20.59	broken strokes
Direction	99.73	99.39	84.45	76.67	rotation, edge noises
ET1	90.60	83.28	38.74	33.78	position deviations, broken strokes
ET2	98.45	96.01	65.22	61.94	blurred strokes
LDC	95.50	93.34	60.87	57.64	font dependent (stroke width)
SP4	84.10	83.14	23.06	27.65	broken strokes
BJD-BS	83.25	77.17	45.59	41.82	edge noises
WDH	93.26	87.62	82.08	77.55	edge noises
combination	99.95	99.95	91.57	85.32	

Table 7.33: Result overview

As a whole the experiments show that the combination of the distances yields better results than the combination of the recognition results. But for the combination of several feature sets it is difficult to find an optimal weighting of the feature sets as shown in the experiments with a special distance weighting and the combination with the recognizer from Jing Zhang.

The comparison experiments demonstrate that some more sophisticated methods yield better results than the simpler methods. The cubic spline interpolation should be used for the scaling of grayscale images instead of the bilinear interpolation. The experiments also show that the adaptive threshold selection is better than the histogram method for images with a non-constant background.

For images where these problems do not occur, it makes no big difference which method is being used. For the color conversion the PAL formula seems to be appropriate to avoid the mapping of different colors to a single grayscale value.

The usage of the LDA was not successful for the features. This could be caused by their discrete nature and that the values of the examples do not resemble a Gaussian distribution. The genetic algorithms can instead improve the accuracy a bit. The feature selection experiments show that some features disturb the recognition.

The acceleration experiments show that the direction features with 64 values are best suited for the preclassification. Even though the TSL feature and projection features consist of fewer values (1 and 32 respectively) and the distance calculation is faster for them, the higher accuracy of the direction features allows to select much fewer characters for the fine classification.

7.9 Comparison with Other Systems

Since every recognition system uses its own database, the difficulties of the recognition tasks may be different. The TECHIS and BBN Byblos system are designed for the same character set as the proposed system (GB2312-80 encoding, class 1), but those systems are just designed for newspaper recognition, while the proposed system should also be able to deal with distortions of recorded signs (light effects).

The system was compared with the system developed by Jing Zhang (see section 6.1). On the data which were extracted by her, my system obtained an accuracy of 91.57% while she obtained an accuracy of 94.26%. The weaker performance of my recognition system can be explained by the fact that the characters were extracted without leaving a bit of space at the edges. This causes problems when trying to decide after the binarization, whether an inversion shall be performed or not. Because the other system does not convert the data to a binary image, these mistakes do not happen there. For the characters without special problems the system from Jing Zhang had an accuracy of 91.01%, while my system obtained 85.3% without using the frequency data.

The other problem for my system was that she uses data from several fonts for the training.

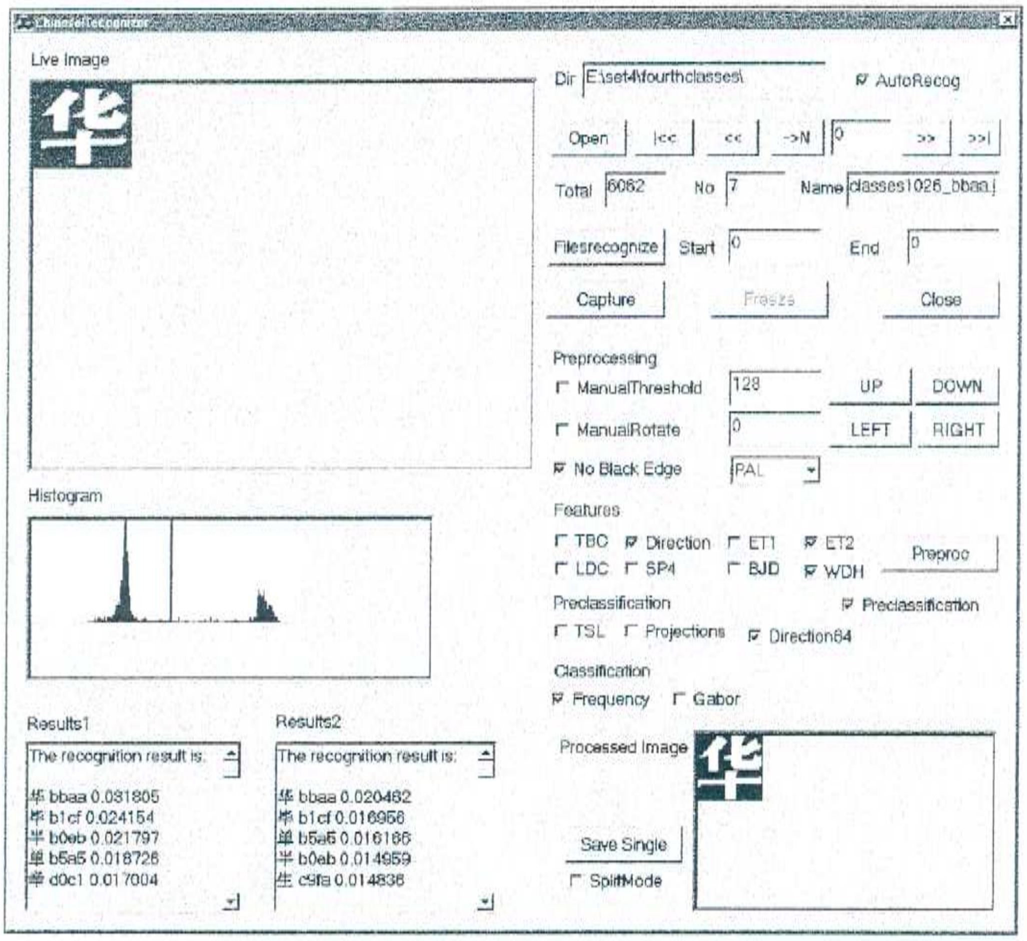
The last experiment shows that a combination of both systems yields better results than each system itself. This can be explained by the differences of the recognition approaches: The proposed system converts the input to a binary image and does not apply an LDA on the features. Jing Zhang's system uses another kind of preprocessing and uses a grayscale input for the feature extraction. Her system performs an LDA on the features.

7.10 System Interface

Figure 7.16 shows the user interface of the proposed system. The window in the upper left side is used to display the system input. This input can either be loaded from a file or from a camera. If the user would like to load an image from a file, the "open" button can be used to load the names of all files with the same ending (.jpg or .bmp) from a directory. The buttons "$| <<$","$<<$","$>>$","$>> |$" and "$- > n$" can be used to select a file from the directory. For recognizing a sequence of images automatically the "files recognize" button can be pressed. The start and end numbers of the sequence have to be entered into the fields "start" and "end". If the user would like to use input from a camera, the input can be started by using the "capture" button. The function of the "freeze" button is to freeze the current image for the recognition. Afterwards the user can select a region from the image by drawing a rectangle in the input window and the recognition of that region will be performed.

The right side of the interface contains several checkboxes to influence the recognition process. Some manual operations like rotation and threshold selection have been used when testing the system. The button "no black edge" is used to select which scaling method is used (see section 2.1.1). Below, the user can select which feature sets should be used for the classification and whether preclassification shall be performed. For the preclassification the feature sets can also be selected. The checkbox "frequency" indicates whether data of the character frequencies shall influence the recognition result.

The window "histogram" on the left shows the histogram of the character and a black line from top to bottom indicates the selected threshold value. Below two windows show the results for different testing methods (i.e. using the Euclidian distance on the left and the angle distance on the right). The window "processed image" shows the image after preprocessing.



Dir E:\set4\fourthclasses\ [✓ AutoRecog]

Total 6062 No 7 Name classes1026_bbaa.j

Start 0 End 0

Preprocessing

ManualThreshold 128

ManualRotate 0

[✓ No Black Edge] PAL

Features

TBC [✓ Direction] ET1 [✓ ET2]

LDC SP4 BJD [✓ WDH]

Preclassification [✓ Preclassification]

TSL Projections [✓ Direction64]

Classification

[✓ Frequency] Gabor

SplitMode

Results1

The recognition result is:

华 bbaa 0.031805
毕 b1cf 0.024154
半 b0eb 0.021797
单 b5a5 0.018726
华 d0c1 0.017004

Results2

The recognition result is:

华 bbaa 0.020482
毕 b1cf 0.016956
单 b5a5 0.016166
半 b0eb 0.014959
生 c9fa 0.014836

Figure 7.16: System interface

Chapter 8

Future Work

For the further improvement of the system some approaches which were presented at several locations could be implemented.

training with more fonts: In most of the experiments the training data contain only characters written in the standard font. For further improvement examples from the other standard fonts could be used.

sequence recognition: While the proposed system can only deal with single characters, it could be extended for sequence recognition. Methods for the character segmentation and the postprocessing were presented in this thesis (see section 2.11 and chapter 5). The sequence recognition could also use a font recognition method like in the system proposed in section 6.6.

genetic algorithms: Genetic algorithms could be used for feature selection or weighting as described in section 3.12. While these methods have been used only inside a feature set, they could also be used for weighting several feature sets.

[11] M. Hamanaka, K. Yamada, and J. Tsukumo. *Handprinted Kanji Character Recognition Using Normalization-Cooperated Feature Extraction.* Proc. 3rd IWFHR, pp. 343-348 (1993).

[12] C. J. Hilditch. *Linear Skeletons from Square Cupboards.* in B. Meltzer and D. Michie, eds., Machine Intelligence IV, Edinburgh: University Press, pp. 403-420, 1969.

[13] S.-L. Hsieh and T.-M. Parng. *A new Scheme for Rectifying Recognition Results of Printed Chinese Characters.* Pattern Recognition 34 (2001), pp. 2121-2132.

[14] X. Huang, J. Gu, and Y. Wu. *A Constraint Approach to Multifont Chinese Character Recognition,* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 15, Issue 8, August 1993.

[15] G. Kim and S. Kim. *Feature Selection Using Genetic Algorithms for Handwritten Character Recognition.* In: L. Schomaker and L. Vuurpijl (Eds.), Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, September 11-13, 2000, Amsterdam.

[16] F. Kimura, T. Wakabayashi, S. Tsuruoka, and Y. Miyake. *Improvement of Handwritten Japanese Character Recognition using Weighted Direction Code Histogram.* Pattern Recognition, vol. 30, no. 8, p. 1329-37, August 1997.

[17] Y. Kobayashi, K. Yamada, and J. Tsukumo. *A Segmentation Method for Hand-Written Japanese Character Lines Based on Transitional Information.* Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings, pp. 487-491., 11th IAPR, 1992.

[18] C. Lee and B. Wu. *A Chinese-Character-Stroke-Extraction Algorithm Based on Contour Information.* Pattern Recognition, Vol. 31, No. 6, pp. 651-663, 1998.

[19] Z. Lu, R. Schwartz, P. Natarajan, I. Bazzi, and J. Makhoul. *Advances in the BBN BYBLOS OCR System.* International Conference on Document Analysis and Recognition (ICDAR), 1999.

[20] W. Lunscher and M. Beddoes. *Fast Binary-Image Boundary Extraction.* Comp. Vision Graphics Image Proc., vol. 38, pp. 229-257, 1987.

[21] S. Mori, H. Nishida, and H. Yamada. *Optical Character Recognition.* Wiley Interscience Publication, New York, 1999.

[22] G. Nagy. *Twenty Years of Document Image Analysis in PAMI.* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 38-62, Jan. 2000.

[23] H. Nishimura, M. Kobayashi, M. Maruyama, and Y. Nakano. *Off-line Character Recognition Using HMM by Multiple Directional Feature Extraction and Voting with Bagging Algorithm.* Proceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR), 1999.

[24] N. Otsu. *A Threshold Selection Method from Gray-Level Histogram.* In: IEEE Trans. SMC-9 (Jan. 1979), pp. 62-66.

[25] J. R. Parker. *Practical Computer Vision Using C.* John Wiley and Sons, 1993.

[26] T. Pavlidis and G. Wolberg. *An algorithm for the segmentation of bilevel images.* Proc. IEEE Comp. Soc. Conf. Comp. Vision and Pattern Recognition, pp. 570-575, June 22-26, 1986.

[27] T. Pavlidis. *A vectorizer and feature extractor for document recognition.* Comp. Vision, Graphics, Image Proc. , vol. 35, pp. 111-127, 1986.

[28] L. R. Rabiner. *A Tutorial on Hidden Markov Models and selected applications in speech recognition.* Proceedings of the IEEE, Volume: 77, Issue: 2, Feb. 1989, Page(s): 257-286.

[29] S. Rice, G. Nagy, and G. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier.* Kluwer, Boston, 1999.

[30] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, New York, 2nd ed., 1982.

[31] H. Schantz. *The History of OCR, Optical Character Recognition.* Recognition Users Association, Manchester Chester, Vt, 1982. ACM SIG Proceedings.

[32] R. Schwartz, C. LaPre, J. Makhoul, C. Raphael, and Y. Zhao. *Language-Independent OCR Using a Continuous Speech Recognition System.* Proc. Int. Conf. on Pattern Recognition, Vienna, Austria, pp. 99-103, August 1996.

[33] R. Suchenwirth, J. Guo, I. Hartmann, G. Hincha, M. Krause, and Z. Zhang. *Optical Recognition of Chinese Characters.* Vieweg, Braunschweig/Wiesbaden, 1989.

[34] T. Suzuki and S. Mori. *Structural Description of Line Images by the Cross Section Sequence Graph.* Int. J. Pattern Recognition and Artificial Intelligence 7(5), 1055-1076 (1993).

[35] Y. Tao and Y. Tang. *The Feature Extraction of Chinese Characters Based on Contour Information.* Proceedings of the Fifth International Conference on Document Analysis and Recognition (ICDAR), 1999.

[36] Internet site *http://www.geocities.com/hao510/charfreq/*

[37] J. Tsukumo. *Handprinted Kanji Character Recognition based on Flexible Template Matching.* Proc. 11th Int. Conf. Pattern Recognition, pp. 483-486, 1992.

[38] J. Tsukumo and H. Tanaka. *Classification of Handprinted Chinese Characters Using Non-linear Normalization and Correlation Methods.* Proc. 9th Int. Conf. Pattern Recognition, pp. 168-171, 1988.

[39] M. Umeda. *A multi-font printed Chinese character reader.* In: Trans. IECEJ J64-D.8 (1984) 908-915.

[40] T. Wakabayashi, S. Tsuruoka, F. Kimura, and Y. Miyake. *On the Size and Variable Transformation of Feature Vector for Handwritten Character Recognition.* Trans. IEICE Japan J76-D-II, 2495-2503 (1993). (in Japanese)

[41] A.-B. Wang and K.-C. Fan. *Optical Recognition of Handwritten Chinese Characters by Hierarchical Radical Matching Method.* Pattern Recognition 34(2001), pp. 15-35.

[42] Xiandai Hunyu Pinlu Cidian. *Frequency Dictionary of the Modern Chinese Language.* Languages Institute Press, 1986.

[43] H. Yamada, K. Yamamoto, and T. Saito. *A Nonlinear Normalization Method for Handprinted Kanji Character Recognition - Line Density Equalization.* Pattern Recognition 23(9), 1023-1029 (1990).

[44] Y. Zhu, T. Tan, and Y. Wang. *Font Recognition Based on Global Texture Analysis.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 23, Issue 10, Oct. 2001. pp. 1192-1200.