CHAPTER 6
ONLINE HANDWRITTEN CHINESE CHARACTER RECOGNITION

The main experiment in this study involves recognizing a handwriting version of a Chinese character from its database or library. A new recognition system is developed which includes database CL2009, feature extraction using *X*-graph and *Y*-graph transformation and a similarity measure, R_p^2 for fine classification. The performance of the proposed recognition algorithm using R_p^2 is studied. The construction of the database CL2009 as well as the *X*-graph and *Y*-graph feature extraction processes will be described in Section 6.2 to Section 6.4. Section 6.5 discussed some distance measures used in the current Chinese character recognition system. This is followed by some experimental results from CL2009 and discussions in Section 6.6. Section 6.7 performed additional experiment on HCH-GB1 dataset to verify the results of CL2009. Concluding remarks are detailed in Section 6.8.

6.1 Handwritten Chinese Character Recognition

The problem of handwritten Chinese character recognition (HCCR) has received considerable attention from the research community due to its wide-ranging applications, which include text entry for form filling, message composition in mobile, computer-aided education, personal digital assistants (PDA) and handwritten document retrieval. However, HCCR is different from the handwriting recognition of other languages. It poses a special challenge due to its complex structure, large shape variation, large character set and many instances of highly similar characters in Chinese words. Even for the same character, there is a large variation in its graphical pattern because of the

variability of writing styles of different writers. The area of handwriting recognition can be divided into two different categories: offline and online approaches. This study will concentrate on the latter approach.

Feature extraction (Michalak and Kwasnicka, 2006; Swiniarski, 2001) and classification (Miquelez *et al*., 2004; Fujarewicz and Wiench, 2003) are crucial to a recognition system. However, defining a feature vector for handwritten Chinese character recognition is not trivial. High recognition rate (accuracy and precision) can be achieved by investigating the use of different features. Feature extraction schemes can be classified into three categories: structural, statistical and hybrid statistical-structural (Liu *et al*., 2004). Structural approaches such as the Attributed Relational Graph (ARG) (Liu *et al*., 1996) and Fuzzy Attributed Relational Graph (FARG) (Zheng *et al*., 1997) are amongst the earliest and most widely used methods. The increase of character complexity, for example, the increase of stroke number will increase the size of the feature vector and subsequently influence recognition rates. Numerical measurements that describe structure can be used to develop statistical approaches due to its computational efficiency. An alternative to the stroke number, the number of occurrences for stroke directions of each character can be used to describe structure. Among the statistical based feature extraction methods, the most frequently used method is the Direction Feature Densities (DFD) (Kawamura *et al*., 1992; Kimura *et al*., 1997). However, this direction feature cannot tolerate well with character shape deformation and some of them are stroke number dependent. The structural approach and statistical ideas have been combined, for example, Hidden Markov Model (HMM) (Shimodaira *et al*., 2003; Takahashi *et al*., 1997), as an efficient way for temporal modeling. Nonetheless, the problems of huge time consumption (due to complex computation) and large storage space (due to learning process) are two major problems of this method.

Many distance measures for online character recognition have been developed which include Modified Quadratic Discriminant Function (MQDF) (Kimura *et al*., 1987), Support Vector Machine (SVM) (Gao *et al*., 2002), Neural Network (NN) (Romero *et al*., 1995), Compound Mahalanobis Function (CMF) (Suzuki *et al*., 1997) and City Block Distance with Deviation (CBDD) (Kato *et al*., 1999). These distance measures achieve high recognition rate in HCCR. Applications for embedded systems employ simple distance measures such as Minimum Distance (MD) classifier (Gonzalez and Woods, 1993) for limited storage consideration (Liu *et al*., 2005), but may face problems with character shape deformation. These methods require normalization which uses methods that may incur costs in terms of time spent.

This study proposes a novel HCCR system using features extracted from the *X*-graph and *Y*-graph. The sequence of points $(x_t, y_t), 1 \le t \le N$ where $t, N \in \mathbb{Z}^+$ obtained from the trajectory of handwritten Chinese character is transformed into two separated graphs, firstly the graph of *x*-coordinate versus time sequence (called *X*-graph), and secondly the graph of *y*-coordinate versus time sequence (called *Y*-graph). The *X*-graph (or *Y*-graph) may be treated as a discrete signal, $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$. The Haar wavelet transform was applied to handle dimensionality problem. The coefficient of determination $\left(R_p^2\right)$ for the 2-dimensional unreplicated linear functional relationship model is proposed as a similarity measure and is used for estimating recognition rates.

6.2 Database

HCL2000, ETL9B and CASIA are the commonly used databases or libraries in Chinese character recognition, with some detail given in Table 6.1. The number of classes is defined as the number of different characters, whereas a sample is defined as the number (or subset) of reproductions by different writers for each character (or class).

The HCL2000 database has been used in Long and Jin (2008), Liu and Ding (2005), The ETL9B database in Dong *et al*. (2005), Gao and Liu (2008) and CASIA database in Gao and Liu (2008) respectively. The existing databases (Table 6.1) store many samples of different writing styles for each character, in order to cope with the problem of handwriting variation of different writers. This technique encounters with the problem of large storage space. This study considers situations with limited storage space only. In particular, only one sample of each character is available in the database.

A new database with only a single sample for each character is created, namely CL2009. This database is based on Jun Da's modern Chinese character frequency list (Dan, 2004), in which he collected from a large corpus of Chinese texts obtained from online sources. 3000 frequently used simplified Chinese characters are selected and reproduced individually by an expert (someone with at least 10 years experience in writing *songti*) thus creating the characters in our database. The writing style *songti* was considered due to its wide usage and similar to most of the handwritten Chinese characters. A subset of CL2009 is given in Figure 6.1.

Table 6.1: Three commonly used databases for Chinese character recognition and a new created database for this research.

Database	Origin	Number of classes	Number of samples
HCL2000	Collected by Beijing University of Posts and Telecommunications for China 863 project.	3755	1000
ETL9B	Collected by Electro-Technical Laboratory (ETL) of Japan	2965	200
CASIA	Collected by Institute of Automation of Chinese Academy of Sciences	3755	300
CL2009	Based on Jun Da's modern Chinese character frequency list (Dan, 2004)	3000	1

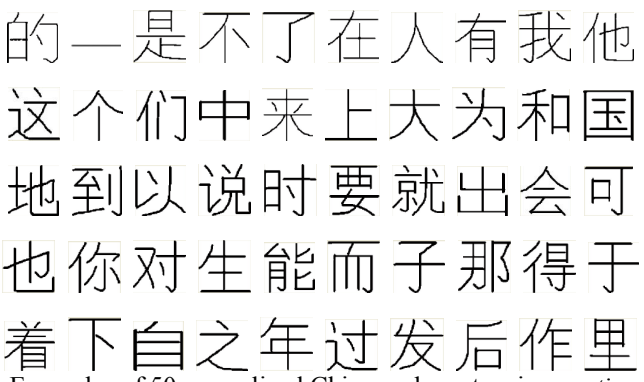


Figure 6.1: Examples of 50 normalized Chinese characters in *songti* written style.

6.3 The Experiment

The recognition system was developed in a Dell Vostro 1400 N-Series notebook of Intel® Core™2 Duo Processor T5470 and 1GB (2×512 MB) 667MHz Dual Channel DDR2 SDRAM (see http://www.dell.com.my). The programming language used is MATLAB and the implementations utilized only one CPU core.

Two writers reproduce each of the 3000 characters in CL2009 only once. Each writer was given one week to complete the reproduction process under similar conditions. The first writer (A) has more than 15 years of experience in writing Chinese character, whereas the second writer (B) has 6 years of experience. The Wacom Intuos®3 pen tablet (Figure 6.2, also see http://www.wacom.com.au/intuos3) was used by each writer for the reproduction process. For each character, 128 points are used to represent each stroke. Thus, a *w*-strokes character, for example, will have a total of $128\times w$ points. This fix number of points for each stroke ensures that the scale invariant property of R_P^2 given in Section 4.4.6 hold.

Figure 6.2: Wacom Intuos®3 tablet (A3 wide) and grip pen.

Samples of 20 characters from the writer A and writer B illustrate the size, slant and position variation, as well as deformation of character as given in Figure 6.3 and Figure 6.4 respectively.

Figure 6.3: A sample of 20 Chinese characters written by writer A.

Figure 6.4: A sample of 20 Chinese characters written by writer B.

Once a character is written, it is studied without further pre-processing or it is cropped and normalized before analysis. This is followed by extracting the feature vector from the *X*-graph and *Y*-graph, which include using the Haar wavelet transformation for reducing dimension. The derived feature vector is subjected to a two-

stage classification procedures; firstly is the rough classification and secondly is the fine classification.

The proposed recognition system (Figure 6.5) is compared to eight other systems using the city block distance with deviation (CBDD), minimum distance (MD), compound mahalanobis function (CMF), modified quadratic discriminant function (MQDF), MSSIM, RMSE, R_S^2 and R_F^2 (see Section 6.5). The experiment was carried out for both pre-processing normalized and non-normalized characters.

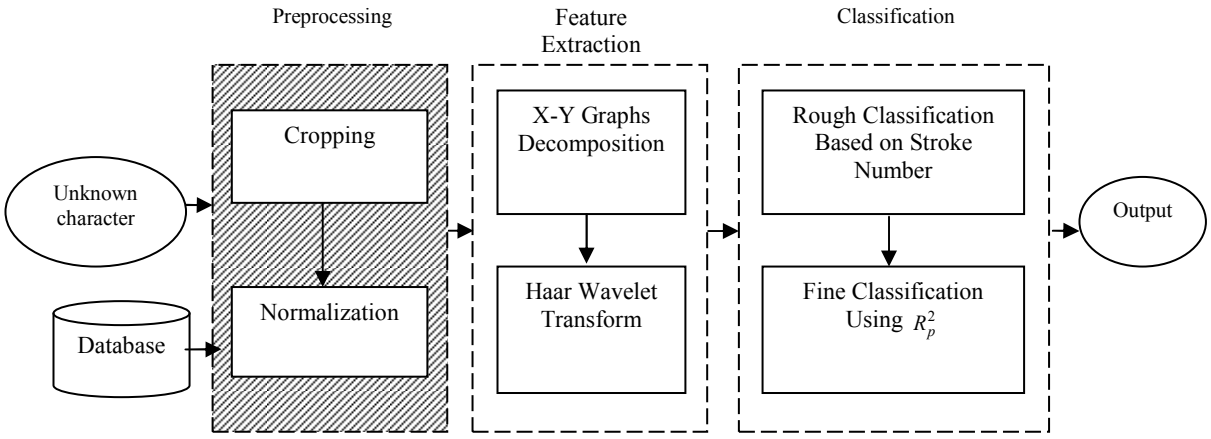


Figure 6.5: The complete recognition system with preprocessing. The shaded area implies that pre-processing is optional.

6.3.1. Cropping

Given the sequence of points for an input character, the maximum x and y coordinates, and also their corresponding minimum were determined. Then, a subset of the original 256×256 image, labeled as the subarea, $\left(y_{\min}:y_{\max},x_{\min}:x_{\max}\right)$ (from row $y_{\min}$ to row $y_{\max}$ and column $x_{\min}$ to column $x_{\max}$) was cropped (Figure 6.6).

6.3.2. Normalization

The sequence of points $\left[x_t,y_t\right]$, which ranges within the cropped subarea is normalized to the size of 128×128, as shown below.

$$x_t^* = 127\left(\frac{x_t - x_{\min}}{x_{\max} - x_{\min}}\right) + 1 \tag{6.1}$$

$$y_t^* = 127\left(\frac{y_t - y_{\min}}{y_{\max} - y_{\min}}\right) + 1 \tag{6.2}$$

An illustration of normalization is given in Figure 6.6 and Figure 6.7. The Linear normalization (LN) method (Saeed, 2000; Deepu *et al*., 2004) is preferred in our experiment over non-linear normalization (NLN) (Casey, 1970, Liu *et al*.; 2003, Liu and Marukawa, 2004; Horiuchi *et al*., 1997; Liu and Marukawa, 2005).

6.3.3. *X*-Graph and *Y*-Graph

The *X*-graph is defined as $\{t, x_t\}$, $1 \le t \le N = 128 \times w$ where x_t is the value of the *x*-coordinate of a point on the character at position (space) *t*. The *Y*-graph $\{t, y_t\}$ is similarly defined. The subscript *t* in $\{t, x_t\}$ and $\{t, y_t\}$ are depend on stroke direction, stroke order and stroke number, thus preventing the possibility of different pattern for the same character. As an example shown in Figure 6.8, the values in the *X*-graph drop while in the *Y*-graph the values rise, when the first stroke is written; whereas in the case of the second (horizontal) stroke (this order is also fixed in Chinese character), the values in the *X*-graph rise while the corresponding values for the *Y*-graph remain unchanged. Note that the origin is defined at the top-left corner. The process was repeated until the seventh stroke. Consequently, the feature vectors obtained are $[x_1, \ldots x_N]^T$ and $[y_1, \ldots y_N]^T$, $N = 7 \times 128$ representing the *X*-graph and *Y*-graph respectively.

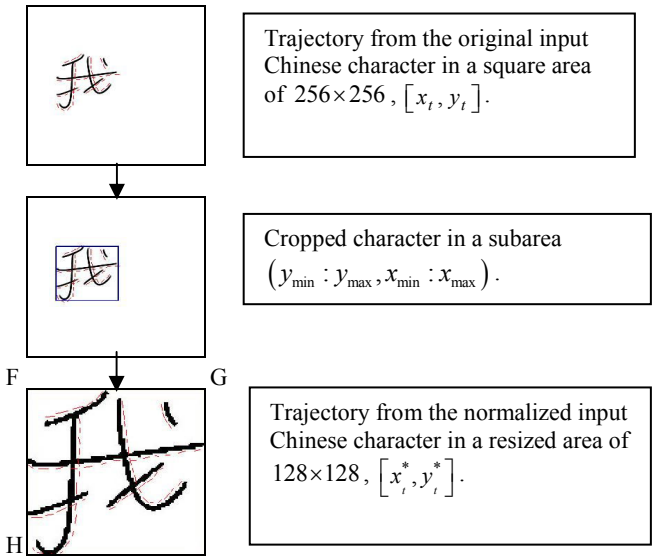


Figure 6.6: Diagram of the whole preprocessing procedure for the Chinese character '我' (means I or me). Point F is defined as the origin, FG is the *x*-axis and FH is the *y*-axis. The *y*-axis is defined such that moving from F to H implies increasing *y* values.

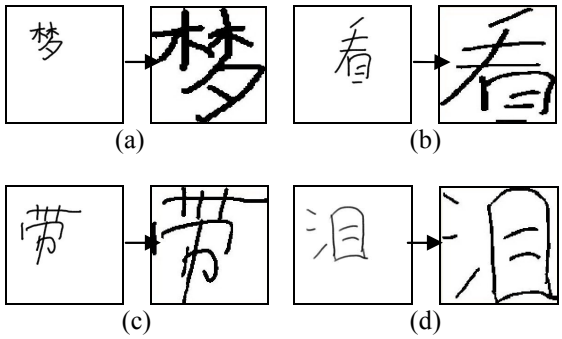


Figure 6.7: Examples of non-normalized (left) and normalized (right) Chinese character: (a) '梦' (means dream), (b) '看' (means see or look), (c) '带' (means bring) and (d) '泪' (means tear).

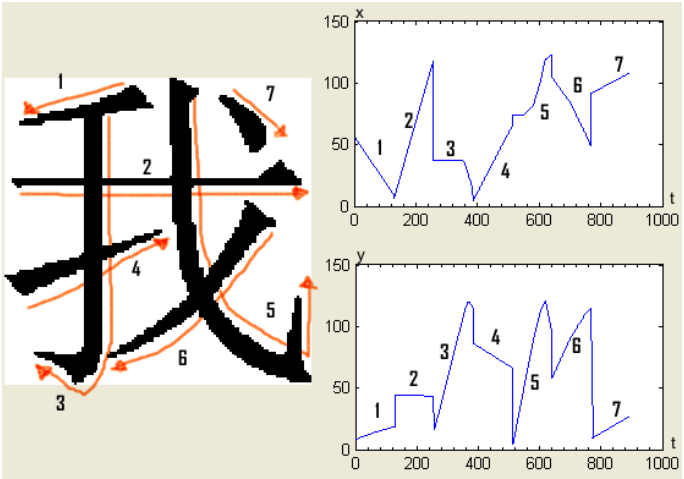


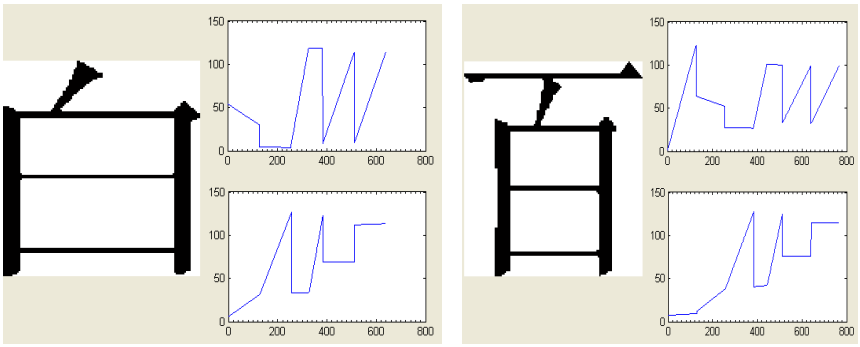
Figure 6.8: *X*-graph (above) and *Y*-graph (below) of Chinese character '我' (means I or me).

6.3.4. Properties of the *X*-Graph and *Y*-Graph

The drive to use the *X*-graph and *Y*-graph lies in its properties. Only one character can be represented by a pair of *X*-graph and *Y*-graph. Different writing styles will still yield the same pair of *X*-graph and *Y*-graph. Finally, it is simple to use the *X*-graph and *Y*-graph. Henceforth the properties of uniqueness, invariance and simplicity are illustrated by examples in the following sub-sections.

(i) Uniqueness

For similarly shaped characters, Figure 6.9, the *X*-graph and *Y*-graph are in different shapes. Hence, both the *X*-graph and *Y*-graph can be considered as useful features for discrimination.

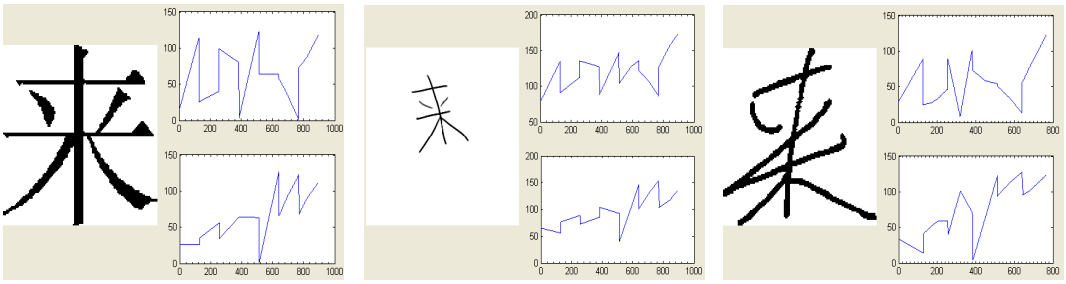


(a) (b)

Figure 6.9: *X*-graphs (above) and *Y*-graphs (below) plotted for two similar characters: (a) '白' (means white) and (b) '百' (means hundred).

(ii) Invariance to different writing styles

Different writing styles is the main problem faced in developing a recognition system. The character '来', obtained from the database with its corresponding *X*-graph and *Y*-graph is given in Figure 6.10(a). The same character written by two different writers are given in Figure 6.10(b) and Figure 6.10(c). Figure 6.10(b) and Figure 6.10(c) show clear differences in writing styles, yet their *X*-graph and *Y*-graph are similar in appearance. Similar remarks can be made about the other characters in CL2009.



(a) (b) (c)

Figure 6.10: *X*-graph (above) and *Y*-graph (below) plotted for character '来' (means come): (a) Regular character in database, (b) Characters '来' written by writer A and (c) Character '来' written by writer B.

(iii) Simplicity

Transforming the character coordinates into two separated *X*-graph and *Y*-graph, and obtaining the feature vectors from the corresponding graphs are the only tasks that is required. This simple approach should boost the efficiency and speed of the recognition system.

6.4 Haar Wavelet Transform

Wavelet transformation (Shioyama *et al*., 1998; Huang and Huang, 2001) was used to reduce the dimension of the feature vector. Although there are many different wavelet families, such as Haar, Daubechies, Coiflet, Symmlet and Mallat, only the Haar wavelet transformation will be considered due to its simplicity. Haar wavelet reduces the size of the feature vector by creating two new sequences of points $\mathbf{a}_j = [a_{xj}, a_{yj}]$ and $\mathbf{d}_j = [d_{xj}, d_{yj}]$, $1 \le j \le D$, $2^5 \le D < 2^6$, which are known as the approximation and detailed coefficients. Only the approximation vector $\mathbf{a}_j$ will be used, in particular,

$$a_{xj} = \frac{x^*_{2j-1} + x^*_{2j}}{\sqrt{2}} \tag{6.3}$$

represents the *X*-graph and

$$a_{yj} = \frac{y^*_{2j-1} + y^*_{2j}}{\sqrt{2}} \tag{6.4}$$

represents the *Y*-graph. The new extracted feature $\mathbf{a}_j$ is then used for classification.

Let $\mathbf{b}_j = [b_{xj}, b_{yj}]$ represents the approximation vector of the *j*-th character in the database obtained in the same manner as the vector $\mathbf{a}_j$. The notations in Equation 6.3 and Equation 6.4 follow Ritter and Wilson (2001).

6.5 Classification

Compared to numerals and alphabets, the number of Chinese character set is extremely large. Hence, in order to speed up the recognition system, the classification process is separated into two stages: rough classification and fine classification.

6.5.1. Rough Classification

The number of strokes in each character, *w*, for all characters in CL2009 is determined as follows.

$$w = \frac{N}{128} \tag{6.5}$$

where *N* is the number of points for all strokes (see Section 6.3). Characters with the same number of strokes are gathered into one group. The rough classification takes into account the stroke number only. Defining character by stroke number may create fewer classes of characters that need to be considered and may speed up the recognition system. Results are given in Table 6.7.

6.5.2 Fine Classification

After the rough classification stage, the process of fine classification is applied sequentially. In fine classification, the number of strokes for the unknown feature vector $\mathbf{a}_j$ is first counted. Suppose $\mathbf{a}_j$ has nine strokes, then it is compared to all the 363 members (see Table 6.7) of the nine-stroke class using a similarity measure. The smallest distance or largest similarity value between $\mathbf{a}_k$ and $\mathbf{b}_j$ (j=1,2,…,363) will mean that the unknown $\mathbf{a}_k$ has been indentified or recognized as $\mathbf{b}_j$. The distance or similarity measures considered are CBDD, MD, MQDF, CMF, RMSE, MSSIM, R_S^2, R_F^2 and $R_{p=2}^2 = \frac{SS_R}{S_{bb}} = \frac{\hat{\beta} S_{ab}}{S_{bb}}$. The similarity measures CBDD, MD, MQDF and CMF are defined as follows:

(i) City block distance with deviation (CBDD)

Define the D-dimensional handwritten character as $\mathbf{a} = \left(a_1, a_2, \ldots, a_D\right)^T$ and defined the M character classes in the database as $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_M$, where $\mathbf{b}_i = \left(b_{i1}, b_{i2}, \ldots, b_{iD}\right)^T$ for $i = 1, 2, \ldots, M$. The city block distance with deviation (CBDD) measure (Kato *et al*., 1999) is defined as

$$d_i^{CBDD}(\mathbf{a}) = \sum_{j=1}^{D} \max\left\{0, \left|a_j - b_{ij}\right| - \theta \bullet s_{ij}\right\} \text{ for a class } \mathbf{b}_i \qquad (6.6)$$

where s_{ij} denotes the standard deviation of jth element, and θ is a constant. Kato *et al*., (1999) claims that variations of handwritten characters are being taken into account in the city block distance measure.

(ii) Minimum distance (MD)

One way to determine the class membership of an unknown input character **a** is to assign it to the character class of its closest prototype. To determine the closeness, Euclidean distance is used:

$$D_i(\mathbf{a}) = \|\mathbf{a} - \mathbf{b}_i\| \text{ for a class } \mathbf{b}_i,\ i = 1, 2, \ldots, M \tag{6.7}$$

where $\|\mathbf{h}\| = \left(\mathbf{h}^T\mathbf{h}\right)^{1/2}$ is the Euclidean norm. Without loss of generality, it is equivalent to evaluating the functions

$$d_i^{MD}(\mathbf{a}) = \mathbf{a}^T\mathbf{b}_i - \frac{1}{2}\mathbf{b}_i{}^T\mathbf{b}_i \tag{6.8}$$

Equation 6.8 is, therefore, the MD discriminant function as mentioned in (Gonzalez and Woods, 1993). The MD measure is also known as 1-nearest neighbor (NN) rule.

(iii) Modified quadratic discriminant function (MQDF)

Modified quadratic discriminant function (MQDF) is originally proposed by Kimura *et al*. (1987). It is a modified version of the ordinary QDF, in which it reduces the complexity of QDF by replacing the minor eigenvalues of covariance matrix of each class with a constant. MQDF is defined as

$$d_i^{MQDF}(\mathbf{a}) = \sum_{j=1}^{K}\frac{1}{\lambda_{ij}}\left[\phi_{ij}^T(\mathbf{a}-\mathbf{u}_i)\right]^2 + \sum_{j=K+1}^{D}\frac{1}{\delta_i}\left[\phi_{ij}^T\left(\mathbf{a}-\mathbf{u}_i\right)\right]^2 + \sum_{j=1}^{K}\log\lambda_{ij} + (D-K)\log\delta_i$$

$$= \frac{1}{\delta_i}\left(\|\mathbf{a}-\mathbf{u}_i\|^2 - \sum_{j=1}^{K}(1-\frac{\delta_i}{\lambda_{ij}})\left[\phi_{ij}^T(\mathbf{a}-\mathbf{u}_i)\right]^2\right) + \sum_{j=1}^{K}\log\lambda_{ij} + (D-K)\log\delta_i \tag{6.9}$$

where $\mathbf{u}_i$ is the mean of ith class for $i = 1, 2, \ldots, M$, λ_{ij} denotes the eigenvalues (in descending order) of Σ_i (covariance matrix of ith class) for $j = 1, \ldots, D$, ϕ_{ij} are the ordered eigenvectors, δ_i is constant replacing the minor eigenvalues and K denotes the number of dominant eigenvectors.

(iv) Compound Mahalanobis function (CMF)

Compound Mahalanobis Function (CMF) (Suzuki *et al*., 1997) is a discriminant function which improves the discriminant performance of the ordinary Mahalanobis Distance (MD), by projecting the difference class-mean feature vectors of two similar classes onto a certain subspace such that the two similar classes can be clearly differentiated. Let $\mathbf{b}_1$ and $\mathbf{b}_2$ be the two similar classes. We denote the eigenvalues of the covariance matrix of class $\mathbf{b}_1$ as $\lambda_1, \lambda_2, \cdots \lambda_D$ where $\lambda_j \geq \lambda_{j+1}, j = 1, 2, \cdots, D-1$ and the eigenvectors which correspond to λ_j as ϕ_j. The difference between class $\mathbf{b}_1$ and $\mathbf{b}_2$ is considered to be clearly demonstrated in a subspace constructed by the eigenvectors $\phi_{K+1}, \phi_{K+2}, \cdots, \phi_D$ and hence, the CMF for class $\mathbf{b}_1$ is defined as below.

$$\mathrm{CMF}_1(\mathbf{a},\mathbf{u}) = \sum_{j=1}^{p} \frac{\left\{\phi_j^T(\mathbf{a}-\mathbf{u})\right\}^2}{\lambda_j + Q} + \frac{1}{Q}\left\{\left\|\mathbf{a}-\mathbf{u}\right\|^2 - \sum_{j=1}^{p}\left\{\phi_j^T(\mathbf{a}-\mathbf{u})\right\}^2\right\}$$

$$+\mu\left[\sum_{j=K+1}^{p} \frac{\left\{\phi_j^T\boldsymbol{\delta}_1\right\}^2}{\lambda_j + Q} + \frac{1}{Q}\left\{\left\|\boldsymbol{\delta}_1\right\|^2 - \sum_{j=1}^{p}\left\{\phi_j^T\boldsymbol{\delta}_1\right\}^2\right\}\right] \qquad (6.10)$$

where $\mathbf{u}$ is a class-mean vector of class $\mathbf{b}_1$, Q is a bias, μ is the weighting parameter and $\boldsymbol{\delta}_1$ is a projective vectors for $\mathbf{b}_1$ as shown in the following.

$$\boldsymbol{\delta}_1 = \left\{\psi^T(\mathbf{a}-\mathbf{u})\right\}\psi$$

$$\psi = \frac{(\mathbf{u}-\mathbf{v}) - \sum_{j=1}^{K}\left\{\phi_j^T(\mathbf{u}-\mathbf{v})\right\}\phi_j}{\sqrt{\left\|\mathbf{u}-\mathbf{v}\right\|^2 - \sum_{j=1}^{K}\left\{\phi_j^T(\mathbf{u}-\mathbf{v})\right\}^2}} \qquad (6.11)$$

where $\mathbf{v}$ is a mean vector of $\mathbf{b}_2$ and ψ is a unit vector obtained by projecting the difference class-mean vectors $(\mathbf{u}-\mathbf{v})$ onto a subspace constructed by $\phi_{K+1}, \phi_{K+2}, \cdots, \phi_D$ and normalizing the length to 1. In the same way, we can calculate $\mathrm{CMF}_2(\mathbf{a},\mathbf{v})$ for

class $\mathbf{b}_2$. Finally, the two classes are differentiated by comparing $\mathrm{CMF}_1(\mathbf{a},\mathbf{u})$ and $\mathrm{CMF}_2(\mathbf{a},\mathbf{v})$.

6.6 Results and Discussions

Some parameters used in the distance measures need to be defined before performing the experiments. Selected parameter values are as follows;

(I) CBDD: $\theta = 1.2$, as stated in (Kato *et al*., 1999).

(II) MQDF: $K = 1$ since there is only one sample for each character in database and hence it results in only one dominant eigenvalue.

$\delta_i = 1.3641$, in which it is made class-independent and equals to the average of all eigenvalues of all classes. Note that as stated in (Long and Jin, 2008), the performance of classifier is superior when setting the constant class-independent rather than class-dependent.

(III) CMF: $p, K = 1$ (refer to Section 6.6(II)).

$Q = 1.3641$ which is the average of all eigenvalues of all classes.

$\mu = 2.8$, as stated in (Suzuki *et al*., 1997).

6.6.1 Recognition Rate (Accuracy) and Precision

The recognition rate or accuracy is defined as follows

$$Recognition\ Rate\ =\ \frac{Number\ of\ Test\ Samples\ with\ Correct\ Matching}{Total\ Number\ of\ Test\ Samples} \times 100\% \tag{6.12}$$

Every character recognized correctly is given the weight of 1.

Table 6.2(a) and Table 6.2(b) show that R_p^2, R_F^2, R_S^2 and MSSIM perform consistently (not more than 5% differences of recognition rates) for both with pre-

processed and without pre-processed characters as well as for both writers (except MSSIM). Although R_F^2 and R_S^2 achieve slightly lower recognition rate than R_p^2, their performance is better than MSSIM in the case of without pre-processing. On the other hand, recognition rates using these similarity measures are generally higher than recognition rates using the distance measures namely CBDD, MD, MQDF, CMF and RMSE. The recognition rates dropped to 80% or less for these distance measures when the input characters do not undergo pre-processing.

When similar recognition rates are achieved for different levels of character complexity (variation in number of strokes), the recognition system is said to be precise. Three categories of character complexity are defined; namely (i) character of low complexity with less than 6 strokes, (ii) character of medium complexity with 6 to 12 strokes, and (iii) very complex character with more than 12 strokes. Writer A with greater experience shows higher precision rates regardless of pre-processing or otherwise when using R_p^2, R_F^2 and R_S^2 (Table 6.3a). Writer B is less precise than Writer A with respect to these similarity measures, however low precision is seen when MSSIM, RMSE, CBDD, MD, MQDF and CMF are used (Table 6.3a and Table 6.3b).

Table 6.2: Experimental results for different writers: (a) with pre-processing and (b) without pre-processing. Each writer writes all 3000 different Chinese characters.

(a)

Distance measures		CBDD	MD	MQDF	CMF	R_P^2	R_F^2	R_S^2	MSSIM	RMSE
Recognition rate (%) with pre-processing	**Writer A**	96.6	98.2	98.2	97.6	98.0	97.6	98.0	98.4	98.4
	Writer B	93.1	94.1	94.1	94.1	96.1	96.1	97.0	94.1	95.1

(b)

Distance measures		CBDD	MD	MQDF	CMF	R_P^2	R_F^2	R_S^2	MSSIM	RMSE
Recognition rate (%) without pre-processing	**Writer A**	70.0	79.6	81.6	75.0	98.2	98.2	98.2	87.6	80.4
	Writer B	55.9	66.7	66.7	54.9	97.4	96.1	97.1	85.3	65.9

Table 6.3: Experimental results for different number of strokes: (a) writer A and (b) writer B. Number of strokes is grouped into three categories: less than 6 strokes, between 6 to 12 strokes, and more than 12 strokes.

(a) Writer A

Distance measures		**CBDD**	**MD**	**MQDF**	**CMF**	R_P^2	R_F^2	R_S^2	**MSSIM**	**RMSE**
Recognition rate (%) with pre-processing	**Low complexity**	95.3	96.6	97.3	97.3	94.6	93.9	95.9	96.6	97.3
	Medium complexity	97.0	98.8	98.5	97.6	99.4	99.1	99.4	99.1	98.8
	High complexity	100	100	100	100	100	100	100	100	100
Recognition rate (%) without pre-processing	**Low complexity**	58.1	67.6	73.6	64.9	96.6	97.3	98.0	79.0	68.9
	Medium complexity	73.8	83.7	84.0	78.0	98.8	98.5	98.8	90.7	84.3
	High complexity	95.0	100	100	100	100	100	100	100	100

(b) Writer B

Distance measures		**CBDD**	**MD**	**MQDF**	**CMF**	R_P^2	R_F^2	R_S^2	**MSSIM**	**RMSE**
Recognition rate (%) with pre-processing	**Low complexity**	86.1	86.1	86.1	86.1	88.9	88.9	91.7	88.9	88.9
	Medium complexity	96.9	98.5	98.5	98.5	100	100	100	96.9	98.5
	High complexity	100	100	100	100	100	100	100	100	100
Recognition rate (%) without pre-processing	**Low complexity**	58.3	69.4	63.9	52.8	94.4	91.7	94.4	80.6	66.7
	Medium complexity	53.8	64.6	67.7	55.4	98.5	98.5	98.5	87.7	64.6
	High complexity	100	100	100	100	100	100	100	100	100

Characters in the high complexity category are most likely to be recognized accurately for both with and without pre-processing cases as well as for both writers. This is probably because the larger number of strokes makes recognition easier.

6.6.2 Processing Time

The speed of the recognition system is one of the important factors for an efficient recognition system, in which reduction of processing time is important. The

processing time are listed in Table 6.4 for feature extraction, rough classification and fine classification. Applying parallel processing or dynamic link library (DLL) further speeds up the recognition system.

Table 6.4: Result of processing time by components.

	Average processing time (millisecond (ms) per character)
Feature extraction	0.913
Rough classification	1.176
Fine classification	0.957

Details of the processing time analysis for the feature extraction and fine classification are explained in the following sub-sections.

(1) Comparison between feature extraction methods

Compared to other feature extraction methods such as Attributed Relational Graph (ARG), whole character-based Hidden Markov Model (HMM) and Directional Feature Densities (DFD), *X*-graph and *Y*-graph transformation may be regarded as the simplest method in term of computation. This is illustrated in Table 6.5.

(2) Rough classification

The main purpose of implementing rough classification is to speed up the recognition system. By having rough classification, R_p^2 only needs to deal with the small subset of characters, instead of the whole database in fine classification.

Table 6.6 illustrates an example of calculating processing time with and without rough classifications. Suppose that we want to recognize the 7-stroke Chinese character '我'. If rough classification is implemented the character '我' needs only 309 comparisons (instead of 3000 characters) with corresponding 7-stroke characters. Thus,

R_p^2 values need to be calculated for the 309 characters only. The following table shows the approximate speedup value.

Table 6.5: Processing steps for four different feature extraction methods.

Methods	Processing Steps
Attributed Relational Graph (ARG) (Liu *et al*., 1996**)**	Step 1: Perform strokes identification Step 2: Fit the strokes with straight lines Step 3: Determine geometric centres for each stroke Step 4: Construct complete ARG with nodes and arcs Step 5: Convert the ARG to generalized relation matrix
Whole Character-Based Hidden Markov Model (HMM) (Takahashi *et al*., 1997**)**	Step 1: Estimates parameters, such as state transition probabilities, output emission probabilities and initial state probability through learning process, in which it is time-consuming. Step 2: Determine HMM of the character
Directional Feature Densities (DFD) (Kawamura *et al*., 1992**)**	Step 1: Define vectors for each consecutive points on the strokes Step 2: Compute directional feature vectors Step 3: Define vector for square areas Step 4: Perform dimension condensation
***X-Y* Graphs Transformation with Haar Wavelet**	Step 1: Define feature vectors from *X-Y* graphs Step 2: Implement Haar wavelet transform

Table 6.6: Example of processing time with and without rough classifications.

Processing Time for Classification Stage Per Input Character (Seconds)	
With Rough classification	Without Rough Classification
2.379ms(Rough classification) + 1.340ms(Fine classification) = 3.719ms	(1.340ms/309characters) * 3000 characters = 13.010ms. *The character '我' is being compared to 3000 characters in database.

Therefore, the approximate speedup value for recognizing the 7-stroke character '我' is 13.010ms - 3.719ms = 9.291ms or with a time reduction of 71%.

(3) Comparison between similarity and distance measures

Figure 6.11 shows the average processing time for the recognition of without pre-processed characters with varying stroke numbers by using different distance or

similarity measures; namely R_p^2 (R2P), MD, CBDD, MQDF, CMF, R_F^2 (R2F), R_S^2 (R2S), MSSIM and RMSE. Note that R_F^2 and R_S^2 required longer processing times than R_p^2 because their calculations were based on pixel by pixel. Characters with stroke number between 6 and 12 consume longer processing time for all distance and similarity measures. The frequencies of characters in this stroke number range are highest suggesting that more characters are chosen for fine classification (see Table 6.7). Consequently, the processing time taken for fine classification will be greater.

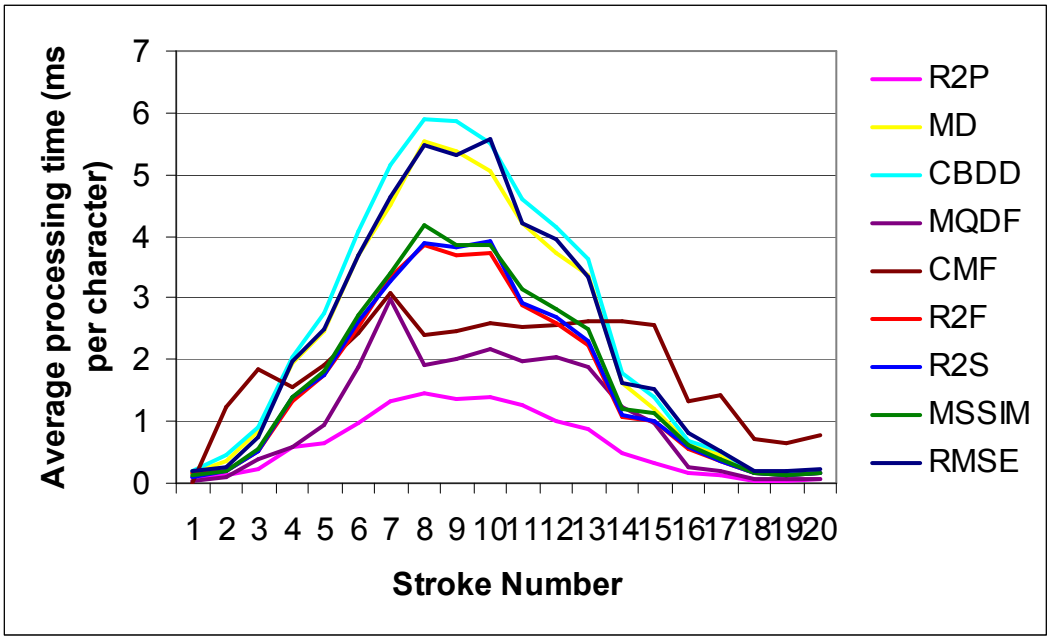


Figure 6.11: Processing time for recognizing characters with varying stroke numbers by using R_p^2, MD, CBDD, MQDF, CMF, R_F^2, R_S^2, MSSIM and RMSE.

It is also seen in Figure 6.11 that the recognition system with the proposed R_p^2 results in the least processing time, followed by MQDF, CMF, R_F^2, R_S^2, MSSIM, RMSE, MD and CBDD for both with and without pre-processing approaches. Due to the algorithm simplicity of R_p^2, the processing time for without pre-processing procedures can be reduced up to 40.69%, 58.27%, 60.20%, 60.92%, 63.16%, 73.02%, 73.04% and

75.31%, respectively (refer to Table 6.8). This reduced in time rate can be calculated by using

$$\frac{\text{Time required by other distance measure} - \text{time required by } R_p^2}{\text{Time required by other distance measure}} \times 100\%$$

Table 6.7: Distribution of the number of strokes.

Number of stroke, v	**Amount of characters with v-stroke(s)**
1	1
2	19
3	51
4	116
5	147
6	240
7	309
8	367
9	363
10	336
11	277
12	262
13	186
14	111
15	91
16	48
17	30
18	12
19	10
20	12
> 20	12
Total	**3000**

Table 6.8: Reduced time rates in comparing the algorithm of R_p^2 with CBDD, MD, MQDF and CMF, R_F^2, R_S^2, MSSIM and RMSE.

Classifiers		R_F^2	R_S^2	**MSSIM**	**RMSE**	**MD**	**CBDD**	**MQDF**	**CMF**
Reduced time rate (%) using R_p^2	*With normalization*	59.34	60.07	62.32	72.25	72.28	74.57	40.36	58.09
	Without normalization	60.20	60.92	63.16	73.02	73.04	75.31	40.69	58.27

6.6.3 Feature Size and Storage Space

A comparison of the size of features with other feature extraction schemes such as (i) Attributed Relational Graph (ARG) (ii) whole character-based Hidden Markov

Model (HMM) and (iii) Directional Feature Densities (DFD) is illustrated in Table 6.9, where the feature derived from the *X*-graph and *Y*-graph has the lowest dimension.

Table 6.9: Feature sizes for four different feature extraction methods.

Methods	Feature Size (Dimension)
Attributed Relational Graph (ARG) (Liu *et al*., 1996)	$(stroke\ number)^2$, increase of stroke number will increase the feature size massively. For characters with stroke number between 6 and 12 which is the typical number of strokes in Chinese characters, dimension = $[6^2, 12^2] = [36, 144]$.
Whole Character-Based Hidden Markov Model (HMM) (Takahashi *et al*., 1997**)**	Sum of the size of parameters $\{a_{ij}\}, \{b_{ik}^1\}, \{b_{il}^2\}$ and $\{\pi_i\}$, which amounts to $2(N-1)+N(L+2)+N$, where N is the number of states and L is the number of quantized directions. In Takahashi *et al*. (1997), it results in the feature size of 187 with $N=9$ and $L=16$.
Directional Feature Densities (DFD) (Kawamura *et al*., 1992**)**	$8\times8\times4=256$
***X*-graph and *Y*-graph transformation with Haar Wavelet**	Between $2^5\times2=64$ (inclusive) and $2^6\times2=128$ (exclusive)

Besides, the more complex the algorithm is, which involves heavy computation, the more memory is required for storage and execution. For instance, the MQDF needs to store the parameters such as mean vectors, eigenvalues and eigenvectors of the covariance matrix for each class. As stated in Long and Jin (2008), suppose that *D*, *K* and *M* are the parameter size of the mean vectors, eigenvalues and the number of class respectively, about 293 MB storage space is required under a system setup of *D*=512, *K*=40 and *M*=3755. On the other hand, only mean vectors are needed to be stored for R_p^2 and this occupies about 53.5 MB for *M*=3000. Larger memory size will cause the increase of computational and production costs.

6.7 Verification of the Experimental Results

In order to verify the experimental results based on the new created database (CL2009) with 3000 characters and 1000 testing samples obtained in this chapter, additional experiment on HCH-GB1 dataset is conducted. HCH-GB1 dataset is one of

the 11 datasets in SCUT-COUCH2009 database which is developed by Jin *et al*. (2010). This database is built to facilitate the research of unconstrained online Chinese handwriting recognition and is publicly available for usage in research community. In HCH-GB1 dataset, there are 3755 characters written by 188 writers and this makes up a total number of 705940 for the character samples. All these samples were collected using PDAs (Personal Digit Assistant) and smart phones with touch screens.

In the verification experiment, 20% of the HCH-GB1 dataset is used as the new testing samples while CL2009 remains as the database of the recognition system. Since this research only concerns about the matter of recognition system without normalization, experiment regarding the case with normalization is not carried out here. The recognition rate without normalization of the five similarity measures (i.e. CBDD, MD, MQDF, CMF and R_P^2) in this additional experiment are shown in Table 6.10. Dataset with stroke number restriction is used.

Table 6.10: Recognition rates without normalization based on HCH-GB1 dataset.

Classifiers	CBDD	MD	MQDF	CMF	R_p^2
Recognition rate (%) without normalization	8.9	11.1	22.1	19.2	74.2

As stated in Jin *et al*. (2010), the benchmark recognition rate of HCH-GB1 is 95.27% and MQDF is the benchmark recognizer. The algorithms of the above MQDF and of this benchmark MQDF are the same. The only difference is that they are implemented in the platform with different conditions. The benchmark MQDF is treated with normalization while the above MQDF is treated without normalization. Notice that this normalization process is described in Section 6.3.

On the other hand, most of the testing samples from the HCH-GB1 dataset are of incorrect stroke number and stroke order. However, the proposed recognition system is stroke number and stroke order dependent. This explains why R_P^2 achieved a lower

recognition rate of 74.2% when HCH-GB1 dataset was used. In this verification experiment, although R_P^2 obtains a lower recognition rate if compared to the results in Table 6.2(b) and Table 6.3 which give the minimum recognition rate of 97.1% and 94.4% respectively, R_P^2 still achieves the highest recognition rate among the similarity measures. For CBDD, MD, MQDF and CMF, even worse recognition rates of 22.1% and below are obtained from Table 6.10 because they fail to recognize the cursive handwritings with severe shape deformation which occupy a large portion in HCH-GB1 dataset. This indicates that they perform more badly and are inappropriate in recognizing characters with incorrect stroke number and stroke order if compared to R_P^2.

Since there is no effect on the results of processing time (in Table 6.4) and storage space (in Table 6.9) by using different testing samples, verification of these two results are unnecessary.

6.8 Summary

A novel online HCCR system was proposed in this chapter and was found to be efficient and simple to apply. This is made possible by the uniqueness and invariance of the *X*-graph and *Y*-graph which involved only simple computations. The R_p^2 gave among the highest recognition rates and precision for both experienced writer A and less experienced writer B as well as for characters in a wide range of complexity (number of strokes). The R_p^2 is also outperformed other similarity measures when the pre-processing stage is removed from the recognition system. A remarkable achievement of using R_p^2 is to save 40.36% to 75.31% of processing time of the recognition system.

Central to the development of the proposed recognition system is the used of R_p^2, whose properties of boundedness $\left(0 \le R_P^2 \le 1\right)$, reflexivity ($R_P^2 = 1$ when the two characters are the same), position invariant (translation invariant in Section 4.4.5), and character size invariant (scale invariant in Section 4.4.6 with the fixed 128 points for each stroke in Section 6.3) motivates its use as a similarity measure. The error terms in the MULFR model handled the slant variations in both *x*-direction and *y*-direction as shown in Figure 6.12. The experiments showed R_p^2 being robust to size, slant and position variation. In general, the HCCR system with similarity measure R_p^2 has better performance and is to be preferred over the use of CBDD, MQDF, MD, CMF, MSSIM, RMSE, R_F^2 and R_S^2 when accuracy and precision, and processing time are considered.

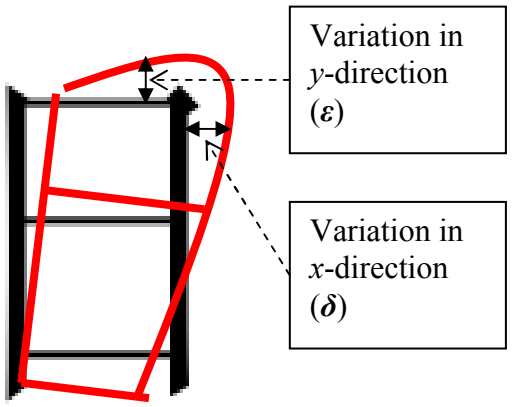


Figure 6.12: Slant variations in both x-direction and y-direction. Character in black color is the regular writing and the character in red color is slant.