國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於特徵點與偏旁資訊之中文字跡真偽辨識演算法

Chinese Handwriting Verification Algorithm

Using Point and Side Information

王冠驊

Guan-Hua Wang

指導教授：丁建均 博士

Advisor: Jian-Jiun Ding, Ph.D.

中華民國108年7月

July, 2019

1. 口試委員會審定書
2. 誌謝

謝謝大家。

1. 中文摘要

字筆跡鑑定的目的在於分辨是該文件是否為當事人所寫，亦或是被別人偽造。字跡辨識在法務鑑定上扮演非常重要的角色，其鑑定的內容十分的廣，例如信用卡、賬單、遺書等等。

要準確的做到筆跡鑑定是一項極需眼力跟細心程度且費時費力的工作。筆跡鑑定的關鍵在於如何從筆跡之間的差異分辨該筆跡是否為同一人所寫。然而，由於同一個人的字跡也存在個體變異，這使得字跡真偽辨識困難許多。

在這篇論文中，我們提出了一套針對中文字的字跡真偽辨識演算法。比起其他語言中的文字，中文字擁有比較複雜的結構。因此，我們所提出的演算法使用了3種不同的特徵集;全局特徵、匹配偏旁特徵以及匹配點特徵。相對於全局特徵，匹配偏旁特徵以及匹配點特徵為局部特徵。局部特徵幫助我們從較小的維度觀察中文文字。我們希望結合這3種特徵集可以更完整的描述整個文字。考量到在做真偽辨識時，不同的特徵對於不同的字有不同的重要性，我們使用*k*-value來篩選合適的特徵。所有篩選出來的特徵被組合成代表該字的特徵向量。支持向量機則會根據組合特徵驗證該手寫文字是否為偽造的文字。實驗結果顯示，我們的演算法在字跡真偽辨識中的準確率達到95.85％，並明顯優於其他現有的辨識方法。此外，實驗結果也顯示我們提出的演算法具有較好的可靠性。

關鍵字:字跡，辨識，中文字，點匹配，偏旁匹配，偏旁，偽造。

1. ABSTRACT

The purpose of handwriting verification is to identify whether the handwritten words were written by a person itself or is forged by others. Handwriting verification plays a very important role in forensics. The content is very extensive, such as credit cards, bills, testaments. Accurate handwriting verification is a work that requires a lot of eyesight and care; overall it is time-consuming and laborious. The key to handwriting verification is how to identify the authenticity of the handwriting from the difference between. However, since there is also an individual variation in the handwriting of the same person, it makes handwriting verification a challenging topic.

In this thesis, we proposed a verification algorithm for the Chinese word. Comparing with words in other languages, Chinese words often have a more complex structure. Therefore, our algorithm adopted three different sets of features, the global features, the matched side feature, the matched point feature. With respect to global features, matched side feature and matched point feature are local features, which help the algorithm to observe the Chinese word in smaller magnitude levels. We hope the combination of the three feature sets is able to completely describe the entire word. Considering that different features may have different importance for the different word, we use *k-*value to select suitable features for each word. All the selected features are combined and the support vector machine with a linear kernel is used as a classifier to verify whether the handwritten word was forged according to the combined feature. Experimental results show that the proposed algorithm reaches 95.85% accuracy in handwriting verification and outperforms several other methods. Moreover, our proposed algorithm shows robustness while dealing with different words.

***Index Terms***—Handwriting, verification, Chinese word, point matching, word side, side matching, forge.

1. CONTENTS

[口試委員會審定書 i](#_Toc14956897)

[誌謝 ii](#_Toc14956898)

[中文摘要 iii](#_Toc14956899)

[ABSTRACT iv](#_Toc14956900)

[CONTENTS vi](#_Toc14956901)

[LIST OF FIGURES ix](#_Toc14956902)

[LIST OF TABLES xi](#_Toc14956903)

[Chapter 1 Introduction 1](#_Toc14956904)

[1.1 Background 1](#_Toc14956905)

[1.2 Thesis Organization 5](#_Toc14956906)

[Chapter 2 Feature Extraction Methods 6](#_Toc14956907)

[2.1 Log-Gabor Filter 6](#_Toc14956908)

[2.2 Local Pattern 9](#_Toc14956909)

[2.2.1 Local Binary Pattern (LBP) 9](#_Toc14956910)

[2.2.2 Local Directional Pattern (LDP) 10](#_Toc14956911)

[2.3 Moment Features 13](#_Toc14956912)

[2.3.1 Spatial and Central Moment 13](#_Toc14956913)

[2.3.2 Hu Moments 15](#_Toc14956914)

[2.3.3 Affine Moments 16](#_Toc14956915)

[2.3.4 Tsirikolias-Mertzios moments 18](#_Toc14956916)

[2.3.5 United Moments 19](#_Toc14956917)

[2.4 Gray Level Co-occurrence Matrix (GLCM) 19](#_Toc14956918)

[2.5 Scale-Invariant Feature Transform (SIFT) 23](#_Toc14956919)

[Chapter 3 Classification Methods 29](#_Toc14956920)

[3.1 Weighted Euclidean Distance 29](#_Toc14956921)

[3.2 K-Nearest Neighbor (KNN) 31](#_Toc14956922)

[3.3 K-Means Clustering 33](#_Toc14956923)

[3.4 Support Vector Machine (SVM) 36](#_Toc14956924)

[Chapter 4 Proposed Algorithm 41](#_Toc14956925)

[4.1 Overview of the Proposed Framework 41](#_Toc14956926)

[4.2 Feature Extraction Stage 42](#_Toc14956927)

[4.2.1 Global features 43](#_Toc14956928)

[4.2.2 Matched Side Feature 49](#_Toc14956929)

[4.2.3 Matched point Feature 53](#_Toc14956930)

[4.3 Feature Selection and Classification 59](#_Toc14956931)

[Chapter 5 Database 61](#_Toc14956932)

[5.1 Introduction of Database 61](#_Toc14956933)

[5.2 Questionnaire Collecting 63](#_Toc14956934)

[5.3 Segmentation 65](#_Toc14956935)

[5.4 Image Noise Reduction 67](#_Toc14956936)

[Chapter 6 Experiment Result 69](#_Toc14956937)

[6.1 Model Evaluation 69](#_Toc14956938)

[6.2 Simulation Result 71](#_Toc14956939)

[Chapter 7 Conclusions and Future Work 80](#_Toc14956940)

[7.1 Conclusions 80](#_Toc14956941)

[7.2 Future Work 81](#_Toc14956942)

[REFERENCE 83](#_Toc14956943)

1. LIST OF FIGURES

[Fig. 2.1 Comparison of Gabor and log-Gabor on both linear and logarithmic spatial frequency scales. (from [5]) 7](#_Toc14956863)

[Fig. 2.2. Three log Gabor filters of different bandwidths. (from [6]) 8](#_Toc14956864)

[Fig. 2.3. An example of the LBP code generation. (from [9]) 10](#_Toc14956865)

[Fig. 2.4 Kirsch edge response masks in eight directions. 11](#_Toc14956866)

[Fig. 2.5 An example of the LDP code generation. (from [9]) 12](#_Toc14956867)

[Fig. 2.6. Stability of LDP vs LBP. (a) Original image (b) Image with Gaussian noise. (from [10]) 13](#_Toc14956868)

[Fig. 2.7. Representation of a two-dimensional digital image. (from [11]) 14](#_Toc14956869)

[Fig. 2.8. The relationships between the pixel of interest and its neighbors specified in both distance and angle. (from [20]) 20](#_Toc14956870)

[Fig. 2.9. An illustration of a gray level co-occurrence matrix with the 0-degree direction. (from[20]) 21](#_Toc14956871)

[Fig. 2.10. Adopt DoG in different octaves in Gaussian Pyramid. (from [22]) 24](#_Toc14956872)

[Fig. 2.11. Local extrema detection. 25](#_Toc14956873)

[Fig. 2.12. An illustration of descriptor generation. (from[22]) 28](#_Toc14956874)

[Fig. 3.1. Illustration of Euclidean distance for n=3 30](#_Toc14956875)

[Fig. 3.2. An example of the *k*-nearest neighbor algorithm (from[26]) 32](#_Toc14956876)

[Fig. 3.3. The validation error of the *k*-nearest neighbor algorithm with a different value of *k* (from[26]) 33](#_Toc14956877)

[Fig. 3.4. An illustration of *k*-means clustering with =5. (from [28]) 36](#_Toc14956878)

[Fig. 3.5. An example of a hyperplane with the largest margin separating two classes. (from[30]) 39](#_Toc14956879)

[Fig. 4.1 The framework of the proposed algorithm 41](#_Toc14956880)

[Fig. 4.2 Example of projection feasture 45](#_Toc14956881)

[Fig. 4.3 Example of stroke after erosion feature 46](#_Toc14956882)

[Fig. 4.4 E xample of orientation feasture 48](#_Toc14956883)

[Fig. 4.5 Example of words with Microsoft JhengHei typeface 49](#_Toc14956884)

[Fig. 4.6. Matched side feature extraction 50](file:///F:\碩士論文\碩士論文4.docx#_Toc14956885)

[Fig. 4.7. Matched point features extraction 55](file:///F:\碩士論文\碩士論文4.docx#_Toc14956886)

[Fig. 4.8**.** Square for direction histogram feature 57](file:///F:\碩士論文\碩士論文4.docx#_Toc14956887)

[Fig. 5.1. A sample of the collected questionnaire sample. 64](#_Toc14956888)

[Fig. 5.2. Segmented words from the known range of x, y coordinate 65](#_Toc14956889)

[Fig. 5.3. (a) Place where has a local maximum response. (b) The obtained location of words. 66](#_Toc14956890)

[Fig. 5.4. (a) Originally captured word image. (b) Refined word image by eliminating the extra stroke pixels 66](#_Toc14956891)

[Fig. 5.5. The relocated word image. 67](#_Toc14956892)

[Fig. 5.6. (a) The raw word image. (b) The word image after noise reduction. 68](#_Toc14956893)

[Fig. 6.1. *k-*fold cross-validation 70](#_Toc14956894)

[Fig. 6.2. Accuracy of words with a different number of strokes 78](#_Toc14956895)

1. LIST OF TABLES

[Table 5.1. The number of stroke and side of each word 62](#_Toc14956820)

[Table 6.1 The confusion matrix 70](#_Toc14956821)

[Table 6.2 Accuracy of words in group1 73](#_Toc14956822)

[Table 6.3 Accuracy of words in group 2 74](#_Toc14956823)

[Table 6.4 Accuracy of words in group 3 75](#_Toc14956824)

[Table 6.5 Accuracy of words in group 4 76](#_Toc14956825)

[Table 6.6 Accuracy of words in group 5 77](#_Toc14956826)

[Table 6.7 The statistical results of the whole database. 78](#_Toc14956827)

# Introduction

## Background

Handwriting verification is usually used in the judicial unit to verify the authenticity of the documents. The content is very extensive, such as credit cards, bills, suicide notes and so on. The purpose of handwriting verification is to identify whether the handwritten words were written by a person itself or is forged by others. In practice, people rely on identification tools and comparison documents for handwriting verification tasks. To accurately verify forged words require a lot of eyesight and patience, which also is time-consuming and laborious. A word can be analyzed from points such as the starting point of stroke, the endpoint of stroke, or features such as the strength of word, the darkness of the ink. The key to handwriting verification is how to identify the authenticity of the handwriting from the difference between. However, since there is also an individual variation in the handwriting of the same person, it makes handwriting verification a challenging topic. Even if we are able to collect a lot of analysis information through accurate measurements, it is still difficult for people to find a clear rule to distinguish the authenticity of the written words.

As technology evolves, we attempt to accomplish handwriting verification automatically by computer. There are several benefits to use a computer for automatic handwriting verification. First, we are able to quickly analyze a large amount of handwriting data with the powerful computing power of the computer. It reduces the misidentification due to the variances caused by individuals. Second, the analysis of the digitized handwriting can obtain numerical values for making optimized decision boundary and result in better accuracy. In this paper, we aim to establish an efficient and accurate handwriting verification system through the above advantages.

We often confuse "handwriting verification" with "handwriting identification". Here, we first clarify the definition of these terms. Handwriting verification is generally a two-class problem. It is the problem of confirming the authenticity of the handwriting. To agree or disagree that he is the person he claimed. On the other hand, handwriting identification is the problem of establishing the identity of the person who writes, from a group of people.

Handwriting verification can be divided into two types: online and offline. An online handwritten sample is collected from a digitizing tablet that is capable of recording pen movement during writing. In addition to static images, dynamic information such as stroke order, speed and pressure are captured. Offline handwritten sample, which often is a static image can be obtained by a scanner or digital camera. In this thesis, we focus on offline handwriting verification.

Handwriting verification is a popular topic of image processing. Recent years, machine learning techniques, such as Deep Neural Network (DNN) or Convolutional Neural Network (CNN), are used for image processing and have achieved good results in a number of tasks. In terms of handwriting verification task, [1][2] use autoencoder to extract features from words. In [3][4], they use multi-layer convolutional neural networks to extract features from words and implement end-to-end training. Machine learning techniques seem to do well in handwriting verification. However, they usually require a large amount of data. For the handwriting verification task, the data of imitated handwritten words is not easy to produce and acquire. With a small amount of training data, the machine learning-based method will be affected by overfitting and lower the verification accuracy. To this end, we focus on developing an algorithm based on feature extraction techniques, which is more reliable in a relatively small amount of training data.

In this paper, we focus on verifying Chinese handwriting words. Compared with English words, Chinese words are more complicated. Chinese words have more strokes, and some Chinese words are composed of multiple small components. This makes Chinese words more difficult to analyze them with global features. On the contrary, the complicated structure gives us an opportunity to extract more local features. Therefore, in addition to the global features, our algorithm further adopted two local features, the matched side feature, and the matched point feature. The local features help the algorithm to observe the Chinese word in smaller magnitude levels. We hope the combination of the three feature sets is able to completely describe the entire word. K-value is used to filter out suitable features for each word and SVM with a linear kernel is used as a classifier.

## Thesis Organization

The framework of handwriting verification can be mainly separated into two parts: feature extraction and classification. We will introduce some feature extraction techniques in Chapter 2 and some classification method in Chapter 3. The proposed handwriting verification algorithm based on features extraction will be presented in Chapter 4. In Chapter 5, the collected database and detailed is introduced. The experiment result will be discussed in Chapter 6. Finally, Chapter 7 summarizes this thesis with the direction of future work.

# Feature Extraction Methods

This chapter provides an overview of some common features extraction techniques. Include Log-Gabor filters, local binary pattern, local direction pattern, moment features, gray level co-occurrence matrix, and scale-invariant feature transform.

## Log-Gabor Filter

Field proposed the Log-Gabor function in [5], as an alternative to the Gabor function. Gabor filters are a traditional method for obtaining localized frequency information. Different filters can detect different frequency components in different directions. However, there is a bandwidth limitation in the Gabor filters. The Gabor function cannot construct an arbitrarily wide bandwidth, and it maintains a small DC component in the even symmetric filter. To this end, log-Gabor is proposed as an alternative to the Gabor function. The transfer function of the log-Gabor in the linear frequency scale is defined as:

|  |  |
| --- | --- |
|  | (2.1) |

There are two important characteristics of the log-Gabor function. First, log-Gabor functions always have zero DC components. Second, the transfer function of the log-Gabor function has an extended tail at the high-frequency end, which means it can be constructed with arbitrary bandwidth.

Log-Gabor has Gaussian transfer functions when viewed on the logarithmic frequency scale; it is similar to the spatial-frequency response of visual neurons. Though there is no experimental evidence to support the idea, log-Gabor function fits better than Gabor function in the visual system.

|  |
| --- |
| Fig. . Comparison of Gabor and log-Gabor on both linear and logarithmic spatial frequency scales. (from ) |

Another advantage of log-Gabor is shown in Fig. 2.1. The information in each channel of the log-Gabor function is equally spread while the Gabor function has an excessive representation in low frequencies.

|  |
| --- |
| Fig. .. Three log Gabor filters of different bandwidths. (from ) |

Three log-Gabor filters of different bandwidths are shown in Fig. 2.2. All of them are tuned to the same center frequency. Though there are multiple benefits of the log-Gabor, Studies in [6] shows the constraints of the filters. Since the sharpness of the filters increases when we have wider the bandwidth, a limit of maximum sharpness has to impose. They also illustrated the ability of the log-Gabor function capturing wide spectral information with a compact spatial filter by showing the spatial width of a 1 octave Gabor function is almost the same with a 3 octave log-Gabor function.

## Local Pattern

### Local Binary Pattern (LBP)

The local binary pattern (LBP) was first described in [7] and is used in [8] for signature verification. LBP is defined as a gray level invariant texture measure in a local neighborhood. The LBP feature is created as the following steps: First, divide the examined into cells. For each pixel in a cell, compare the 8 neighboring pixels (on its left-top, left-middle, left-bottom, right-top, etc.) with the center pixel value. If the neighbor is larger than or equal to the center, write 1, otherwise, write 0. Next, an 8–bit code  is generated by concatenating the binary codes of all neighbors in the 3x3 neighborhood. A formal equation of the LBP feature is defined in equation (2.2), note that the LBP feature has been converted to decimal.

|  |  |
| --- | --- |
|  | (2.2) |

where be the unit step function as represented in [9].

shows an example of the LBP operator. In this example, the center pixel in the 3x3 cell is 127, and its neighboring pixel set is. The 8–bit code is generated by comparing the 8 neighbors with the center pixel . Finally the LBP feature, .

|  |
| --- |
| Fig. .. An example of the LBP code generation. (from ) |

### Local Directional Pattern (LDP)

The local directional pattern (LDP) was proposed in [10]. Similar to LBP, LDP represents each pixel in an 8-bit code. The major problem of LBP is its sensitivity to the presence of noise. LDP is calculated based on gradients, which is more stable than gray level thresholds used in LBP. LDP feature is applied in [9] to improve the result of signature verification.

For each pixel, the LBP feature is computed by thresholding its 8 neighbors while the LDP feature is calculated with 8 different orientations () Kirsch masks, centered on its own position. The Kirsch masks of 8 different orientations () are shown below:

|  |
| --- |
| Fig. . Kirsch edge response masks in eight directions. |

With these masks, the edge response value *ml* in 8 directions can be represented as

|  |  |
| --- | --- |
|  | (2.3) |

Each of the response value represents the edge significance in its respective direction.

Since the corner and edge only show high response values in a particular direction, the LDP operator only chooses themost prominent directions from the response value to form the code. The top *k* largest values  are set to 1 while the others are set to 0. The 8–bit codeis formed by concatenating the binary codes. A formal equation of the LDP feature is defined in equation (2.4), note that the LDP feature has been converted to decimal.

|  |  |
| --- | --- |
|  | (2.4) |

where be the unit step function as represented in [9].is the  largest in .

|  |
| --- |
| Fig. . An example of the LDP code generation. (from ) |

An example of the LDP feature is shown in . In this example, is set to 3. First, the 8 direction response values  are computed by the 8 Kirsch masks. The 8–bit code is generated by labeling the top 3 largest value, which are . The 8–bit code isand finally the LDP feature.

Fig. 2.6 illustrated an example to show the stableness of the LDP feature in the presence of noise, comparing to the LBP feature. (b) is generated by adding Gaussian white noise in (a). We compute the LBP feature and the LDP feature. The  bit of the LBP code in (b) shifts from 1 to 0 while the LDP feature remains the same. It demonstrates the LDP feature is more robust to noise and non-monotonic illumination changes because it is based on gradients rather gray-level values.

|  |
| --- |
| (a) (b)  Fig. .. Stability of LDP vs LBP. (a) Original image (b) Image with Gaussian noise. (from ) |

## Moment Features

### Spatial and Central Moment

The moment features have been widely used in the field of image processing and computer vision. An image moment is scalar quantities for capturing a certain particular weighted average of the image pixels' intensities. An introduction of the two-dimension moments is given in [11].

|  |
| --- |
| Fig. .. Representation of a two-dimensional digital image. (from ) |

The spatial moment of order  is defined as

|  |  |
| --- | --- |
|  | (2.5) |

where *N* and *M* are the number of column and number of row, respectively, of image . The central moments of an image are computed as

|  |  |
| --- | --- |
|  | (2.6) |

where and are defined as

|  |  |
| --- | --- |
|  | (2.7) |

The central moments are invariant under the translation of the image.

These moments preserves the information from the original image in some particular points, since the original image can be reconstructed by a finite set of moments. They are useful because they are a small set that can be generated by simple calculations and are highly representative to represent the whole image. In [12], Huang applied the central moments of order up to 3 for writer verification.

The normalized spatial moment and normalized central moment can be derived from the spatial moment and central moment, defined as follows:

|  |  |
| --- | --- |
|  | (2.8) |
|  | (2.9) |

where . The normalized moments are invariant under both translation and scaling of an image.

### Hu Moments

The Hu moments a well-known moment feature, proposed by Hu in [13]. It is also called the Hu moment invariants because they are invariant under scale, translation, and rotation. The 7 moments can be computed from the normalized central moment , as show below:

|  |  |
| --- | --- |
|  | (2.10) |
|  |
|  |
|  |
|  |
|  |
|  |

### Affine Moments

The affine moments were first proposed in . The affine moment invariants are invariant under general affine transformation. The affine transform is a general linear transformation of space coordinates of the image:

|  |  |
| --- | --- |
|  | (2.11) |

The affine transformation can be decomposed into 6 one-parameter transformations:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | *u* = *x* + α. | 2 | *u* = *x* | 3 | u = ω x | (2.12) |
|  | v = y |  | v = y +β |  | v =ω y |
| 4 | *u* = *δ x* + α | 5 | u = x +t y | 6 | *u* = *x* |
|  | v = y |  | v = y |  | v = t’ x + y |

For every function of moments, it would be invariant under the general affine transformations if the function is invariant under the 6 transformations. Flusser than derived 4 invariants from the  and  order moments

|  |  |
| --- | --- |
|  | (2.13) |

### Tsirikolias-Mertzios moments

The Tsirikolias-Mertzios moments are first proposed in . Unlike other moments, Tsirikolias-Mertzios moments are normalized with respect to the standard deviation, instead of the  order moment. These moments are invariant under translation, scaling, and rotation of the image. The general two-dimension form of the modified moments is defined as follows

|  |  |
| --- | --- |
|  | (2.14) |

where  and  are the horizontal and vertical dimensions of the image ,, are the centroid of the coordinates *x* and *y* , respectively, and  are the standard deviation, respectively.  can be defined as

|  |  |
| --- | --- |
|  | (2.15) |

For binary images, the moments can be simplified as

|  |  |
| --- | --- |
|  | (2.16) |

The Hu moments, Affine moments and the Tsirikolias-Mertzios moments are first applied for character recognition in and later applied in [17] for writer verification.

### United Moments

The United Moments are first proposed in [18]. They focus on avoiding the affection from the geometric varieties of shape. They attempt to find the invariant feature values in all kind of instances. They derived 8 formulas from the Hu moments. The 8 formulas are called united moment invariants, which are invariant in translation, scaling and rotation and can be applied for both region and boundary in discrete and continuous condition. The united moments, , are defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (2.17) |
|  |  |
|  |  |
|  |  |

where are the Hu moments, introduced in . The united moments are applied in for handwriting digit recognition.

## Gray Level Co-occurrence Matrix (GLCM)

The gray level co-occurrence matrix (GLCM) was first presented by Haralick in [20]. Since the texture in the image is formed by the repeated occurrence of the grayscale distribution in the spatial position, there is a certain relationship between pixels in the image space. GLCM is a method for describing textures by studying the relationship between the pixels in a grayscale image. For an image, we first quantized the grayscale into  levels,  is the quantized gray tones. Let  and  be the width and height of the image,  and be the horizontal spatial domain and the vertical spatial domain respectively. The image can be mapped from  by calculating the frequencies of gray tone in the adjacent cells for each pixel.

|  |
| --- |
| Fig. .. The relationships between the pixel of interest and its neighbors specified in both distance and angle. (from ) |

As shown in Fig. 2.8, the feature can be generated along 4 directions, 0-degree, 45-degree, 90-degree, and 135-degree. It captures the neighboring gray-tone relation in the 4 directions. The more formal expression is as follows

|  |  |
| --- | --- |
|  | (2.18) |

where  indicates the number of elements in the set. Each function *P* will generate a GLCM.

|  |
| --- |
| Fig. .. An illustration of a gray level co-occurrence matrix with the 0-degree direction. (from) |

The right matrix in Fig. 2.9 shows the GLCM with the 0-degree direction, while the matrix on the left is the original image. The GLCM (1, 1) value is 1 indicates that the original image only contains one pair of pixels, whose gray level is (1, 1). The same can be said, the GLCM (1, 2) value is 2 indicates that the original image only contains two pair of pixels, whose gray level is (1, 2).

Several classical advance features can be further extracted. Here, we introduce 4 advanced features, texture homogeneity, texture contrast, texture contrast, and texture correlation. These features are used in for off-line signature verification.

1. Texture contrast

|  |  |
| --- | --- |
|  | (2.19) |

Texture contrast measures the local variations in the gray-level co-occurrence matrix. This feature will increase when there is a high variation of intensity because the value  will be diverted away from the diagonal.

1. Texture correlation

|  |  |
| --- | --- |
|  | (2.20) |

 are the standard deviations, and  are the means, respectively, of . Texture correlation measures the joint probability occurrence of the specified pixel pairs.

1. Texture homogeneity

|  |  |
| --- | --- |
|  | (2.21) |

Texture homogeneity measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. If the image is homogeneous, the value  will be extremely high in a certain area. Thus, the sum of the squares should be high.

1. Texture entropy

|  |  |
| --- | --- |
|  | (2.22) |

A homogeneous image reveals high first-order entropy, while a non-homogeneous image has lower entropy.

## Scale-Invariant Feature Transform (SIFT)

The scale-invariant feature transform (SIFT) was first proposed by Lowe in [22] and has been applied in many fields. It is widely used in fields of computer vision such as object recognition, image stitching, 3D modeling, gesture recognition, video tracking. SIFT is invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion. There are mainly four steps involved in the SIFT algorithm.

1. ***Scale-space extrema detection:***

To assure the SIFT feature is invariant to scale, Difference of Gaussian (DoG) is adapted for different octaves of the image in Gaussian Pyramid. The difference of Gaussian is acquired as the difference between Gaussian blurring of an image with two different σ.

|  |
| --- |
| Fig. .. Adopt DoG in different octaves in Gaussian Pyramid. (from ) |

Let be a variable-scale Gaussian

|  |  |
| --- | --- |
|  | (2.23) |

The difference of Gaussian with σ and σ can be computed as

|  |  |
| --- | --- |
|  | (2.24) |

Once the DoG is obtained, the local extrema are detected over scale and space. An example is shown in Fig. 2.11. The pixel denoted as **×** in the image is compared with its 8 neighbors as well as 9 pixels in the previous scale and 9 pixels in next scales. The pixel will be preserved as a potential keypoint if it is a local extrema.

|  |
| --- |
| Fig. .. Local extrema detection. |

1. ***Keypoint localization:***

This step refines the potential keypoints to get a more accurate result. First, Taylor expansion is used to remove keypoints, where its intensity is less than a threshold value. The Taylor expansion is written as:

|  |  |
| --- | --- |
|  | (2.25) |

Next, we set the derivative of  to zero to obtain the location of the extrema

|  |  |
| --- | --- |
|  | (2.26) |

The remove a low-intensity keypoint, we calculate the offset of the extrema:

|  |  |
| --- | --- |
|  | (2.27) |

If , we remove the potential keypoint.

In addition, the DOG also has a higher response for edges and needs to be removed. The concept to detect edges is similar to the Harris corner detector. We know from the Harris corner detector that for edges, one eigenvalue is larger than the other. The details of the process are illustrated below,

First, a Hessian matrix (H) is used to compute the principal curvature.

|  |  |
| --- | --- |
|  | (2.28) |

Let *α* be the eigenvalue with a larger magnitude of *H*, and *β* be the smaller one.

|  |  |
| --- | --- |
| α + β | (2.29) |

Let

|  |  |
| --- | --- |
|  | (2.30) |

The value only relies on the ratio of the eigenvalues, and , rather than their exact values. Therefore, to check whether the principal curvatures is below some threshold, , we only need to check

|  |  |
| --- | --- |
|  | (2.31) |

If the value is greater than , we discard the keypoint.

1. ***Orientation assignment:***

In this step, the keypoint descriptor achieves invariance to image rotation by representing relative to a consistent orientation based on local image properties. The gradient magnitude, , and the orientation, , of the image  are computed as

|  |  |
| --- | --- |
|  | (2.32) |

An orientation histogram is created. The orientation histogram has 36 bins, each bin represents 10 degrees, so it covers the 360-degree range of orientations. The contribution to the histogram is weighted by the gradient magnitude and Gaussian-weighted circular to reduce the response, which is farther away from the center point. The highest peak of the histogram and any peak that is above 80% of the highest peak are taken to compute the main orientation.

1. ***Keypoint descriptors:***

Finally, keypoint descriptor is created in this step. For each keypoint, a block of size 16x16 around the keypoint is taken and then divided into 16 sub-blocks of size 4x4. An 8-bin orientation histogram is created for each sub-block. The keypoint descriptor is generated from the 8-bin orientation histogram, each has  dimensions.

|  |
| --- |
| Fig. .. An illustration of descriptor generation. (from) |

Fig. 2.12. simply illustration the generation of the descriptor. The left shows the gradient magnitude and orientation at each image sample point in a region around the keypoint location. These samples are then accumulated into the orientation histograms summarizing the contents over 4x4 subregions, as shown on the right. Studies in [23][24][25] applied the SIFT descriptors for writer identification.

# Classification Methods

This chapter provides an overview of some common classification methods. Include weighted Euclidean distance, *k*-nearest neighbor, *k*-means clustering, and the support vector machine.

## Weighted Euclidean Distance

The similarity between the feature vectors of a data point can be used to determine which class the data point belongs to. The Euclidean distance provides a simple way to compute the feature distance. The Euclidean distance between points *p* and *q* is the length of the line segment connecting them. Let and be two points in Euclidean n-space, the distance (d) from *p* to *q* is given by

|  |  |
| --- | --- |
|  | (3.1) |

|  |
| --- |
| https://upload.wikimedia.org/wikipedia/commons/thumb/1/10/Euclidean_distance_3d_2_cropped.png/1024px-Euclidean_distance_3d_2_cropped.png  Fig. .. Illustration of Euclidean distance for n=3 |

However, if each element varies in a wide range, the Euclidean distance can be easily dominated by certain elements. Thus, the Euclidean distance can be standardized with standard deviations. The standardized Euclidean is given by

|  |  |
| --- | --- |
|  | (3.2) |

where is the standard deviation of the *i*th variable. We can consider the value of the standard deviation as a given weight; the standardized Euclidean distance can be further generalized into the weighted Euclidean distance.

|  |  |
| --- | --- |
|  | (3.3) |

The weighted Euclidean distance is more flexible since the weights can be determined by any other factors that are considered important. The classification result can be decided by comparing the weighted Euclidean distance with a set threshold.

## K-Nearest Neighbor (KNN)

The k-nearest neighbor is one of the most basic classification algorithms for supervised learning. A brief introduction of the *k*-nearest neighbor algorithm is in [26]. For a target, we first find its *k* nearest neighbors among all data points in the feature space. A commonly used distance metric for continuous variables is Euclidean distance. The target is assigned to the class by making a majority vote by the labels of the k nearest neighbors. A simple example is illustrated in Fig. 3.2.

|  |
| --- |
| scenario2  Fig. .. An example of the *k*-nearest neighbor algorithm (from) |

Two classes of data points in the feature space are noted as red circles and green squares. We intend to classify the blue star to one of the classes. We set, which means a majority vote will be taken by the 3 nearest neighbors. In this case, the result became very obvious as all three votes from the closest neighbor went to the red circle.

|  |
| --- |
| training error_1  Fig. .. The validation error of the *k*-nearest neighbor algorithm with a different value of *k* (from) |

It is critical to decide the factor *k*. At , the algorithm seems to be overfitting the training data. The error rate continues to decrease and reaches minima in a particular value of *k*. After a minimal point, it then increases with increasing *k*. The criteria for selecting the value *k* is not explicit, the best way is to test all the *k* values to obtain the optimal one.

## K-Means Clustering

The standard algorithm of *k*-means clustering was first proposed by Stuart Lloyd in 1957, though it wasn't published as a journal article until 1982 [27]. K-means clustering is an unsupervised learning algorithm used for classification. It is widely used in the topic of computer vision including image segmentation and texture analysis. The user-defined value *k* determines the number of final clusters. For example, if *k* equals 3, the algorithm will cluster the data into 3 groups. Given, the process of clustering can be separated into 3 steps:

1. ***Initialization***

 centroids are created randomly.

1. ***Form clusters***

For every data point in the dataset, label each point with the nearest centroid. This means that if the data point is closer to the centroid of the cluster than any other centroid, the data point is considered to be in a particular cluster. Eventually, the data points are clustered into *k* clusters*.*

1. ***Recompute centroids***

The centroids of the *k* clusters are recalculated by taking the mean of all data points, which belongs to that cluster. If all the centroids remain, the clustering is done; otherwise, iterates Step 2 and Step 3.

We can notice that the *k*-means clustering is a heuristic algorithm, which aims to reduce the total intra-cluster distance. The objective function is given by:

|  |  |
| --- | --- |
|  | (3.4) |

where  is the centroid of the cluster.

Though, the algorithm is robust and easy to implement. However, the value  is hard to set since the ground truth of the amount of cluster is sometimes unknown. Choosing an inappropriate *k* value can lead to poor cluster results. Additionally, the convergence to a local minimum may produce counterintuitive results. Fig. 3.4 illustrate the *k*-means clustering with =5 in each iteration.

|  |  |  |
| --- | --- | --- |
| iteration 1 | iteration 2 | iteration 3 |
| iteration 4 | iteration 5 | iteration 6 |
| iteration 7 | iteration 8 | iteration 9 |
| iteration 10 | iteration 11 |  |
| Fig. .. An illustration of *k*-means clustering with =5. (from [28]) | | |

## Support Vector Machine (SVM)

The support vector machine was first proposed by Cortes and Vapnik in [29]. SVM is a supervised learning method that is widely used for classification and regression analysis. This method has outstanding performance in solving high dimensional, nonlinear by mapping their inputs into high-dimensional feature spaces. In addition, it is also good at solving problems in the small dataset while the algorithm tries to maximize the gap between different classes. A derivation is given in [30].

Let be the data points in *d* dimension space with class

|  |  |
| --- | --- |
|  | (3.5) |

The aims are to find the "maximum-margin hyperplane" that separates the group of points for which = -1 from the group of points for which = 1, so that the gap between the hyperplane and the nearest point (support vectors) from either group is maximized. A hyperplane can be written as a set of points *x* satisfying

|  |  |
| --- | --- |
|  | (3.6) |

where *w* is the normal vector to the hyperplane. If the data points are linearly separable (hard-margin), these hyperplanes can be described by the equations

|  |  |
| --- | --- |
|  | (3.7) |

These constraints state that each data point must lie on the correct side of the margin. This can be rewritten as

|  |  |
| --- | --- |
|  | (3.8) |

So the SVM optimized the hyperplane to have a maximal margin by maximizing the distance , which is the distance between the nearest points from the two classes. That is, to solve

|  |  |
| --- | --- |
|  | (3.9) |

To extend SVM to cases in which the data are not linearly separable, we add slack variables () to allow points to fall into the margin (soft margin).

|  |  |
| --- | --- |
|  | (3.10) |

We add the sum of slack variables to the cost function, which makes limits the violations during the optimization process.

|  |  |
| --- | --- |
|  | (3.11) |

where *C* controls the degree of influence of the sum of . A larger *C* means that the tolerance for noise is not large. The optimize problem can be converted to Lagrange function

|  |  |
| --- | --- |
|  | (3.12) |

According to Karush-Kuhn and Tucker condition, If we want to get the optimal parameters, we differentiate the Lagrange function from this parameter.

|  |  |
| --- | --- |
|  | (3.13) |

The optimization problem change into

|  |  |
| --- | --- |
|  | (3.14) |

Finally, we get the solution

|  |  |
| --- | --- |
|  | (3.15) |

where are the support vectors. A visualize example of the SVM classifier is shown in Fig. 3.5.

|  |
| --- |
| https://miro.medium.com/max/875/1*aqNgiEu0ZBeB-ojoWyryZA.png  Fig. .. An example of a hyperplane with the largest margin separating two classes. (from) |

In 1992, Boser suggested a way to use nonlinear kernels to maximum-margin hyperplanes [31]. Rather than using dot product in the linear SVM, the non-linear SVM uses a nonlinear kernel function. This allows the algorithm to map the data points into high dimensional transformed space. Although the classifier is a linear hyperplane in the transformed space, it may accomplish nonlinear segmentation in the original input space. Here, we introduce 3 common kernels:

1. Linear:
2. Polynomial:
3. Gaussian radial basis function:
4. Hyperbolic tangent:

where are parameters of the kernels.

# Proposed Algorithm

## Overview of the Proposed Framework

|  |
| --- |
| framwork2.png    Fig. . The framework of the proposed algorithm |

We proposed a handwriting verification algorithm for Chinese words, based on feature extraction techniques. The proposed algorithm framework, as shown in , consists of 3 stages: feature extraction stages, feature selection stage and classification stage. For each genuine or forge image, they are first passed to the feature extraction stage to extract word features. Different features are extracted in the feature extraction stage; include global features, matched side feature and matched point feature. Matched side feature and matched point feature are local features, which help the algorithm to observe the Chinese word in smaller magnitude levels. Together with the global features, these three features can more fully describe the whole picture of the whole word. All the extracted features are passed to the feature selection stage. In this stage, we use *k*-value to select features, which are important for this word. Finally, all the selected features are passed into the classification stage. We use SVM with the linear kernel as a classifier. The linear kernel prevents the model from overfitting within a relatively small dataset.

In the following of this section, details of feature extraction would be given in Section . Section 4.3 contains an illustration of how *k*-value helps in feature selection and how the support vector machine is utilized for classification.

## Feature Extraction Stage

Three different feature sets are extracted in this stage, include several global features, matched point feature, and matched side feature.

### Global features

1. ***Moment feature***

For a two-dimensional image, raw image moments are defined as:

|  |  |
| --- | --- |
|  | (4.1) |

where is the size of the raw image and is the value of the pixel which is 0 or 1 for a binary image. The moment features for this paper *mij* are defined as:

|  |  |
| --- | --- |
|  | (4.2) |

where and are the central moments,

|  |  |
| --- | --- |
| . | (4.3) |

In our proposed algorithm, we adopted 7 moments: *m20, m11, m02, m30, m21, m12,* and *m03*.

|  |  |
| --- | --- |
|  | (4.4) |
|  |
|  |
|  |
|  |
|  |
|  |

1. ***Projection feature***

We divide the binary word image into *k* blocks of size along the x-direction and k blocks of size along the y-direction. The projection feature is the total number of stroke pixel in each block. The feature can be formulated as:

|  |  |
| --- | --- |
|  | (4.5) |

where is the size of the image *I* and is the value of the pixel which is 0 or 1 for a binary image. An illustration of the projection feature is shown in Fig. 4.2.

|  |
| --- |
| projection.png  Fig. . Example of projection feature |

1. ***Stroke after erosion feature***

For a binary image *A*, the erosion process of the binary image *A* by a structuring element *B* is defined by:

|  |  |
| --- | --- |
|  | (4.6) |

where denotes the translation of *A* by –b. In our case, the structuring element *B* is a 3 by 3 square. The stroke after erosion feature indicates the thickness of the strokes in a word. The strokes of a word become thinner every time after the erosion process and eventually disappear. The stroke after erosion feature is the number of the remaining stroke pixel after each erosion process. We repeat the erosion process for 5 times. An illustration of the strokes after erosion feature is shown in Fig. 4.3.

|  |
| --- |
| strokeaftererosion.png  Fig. . Example of stroke after erosion feature |

1. ***Orientation feature***

The orientation feature represents the obliqueness of a word. Let {|=1} be the coordinates of stroke pixels in Cartesian coordinate, where is the value of the pixel in a binary image *I*. The centroid of the image *I* is written as

|  |  |
| --- | --- |
|  | (4.7) |

where *P* is the number of .

Let, and . We apply eigendecomposition to *A*. *A* can be factorized as:

|  |  |
| --- | --- |
| , | (4.8) |

where *Q* is a 2 by 2 square matrix, whose *i*th column *qi* is an eigenvector of *A* and *Λ* is the diagonal matrix whose diagonal elements are the corresponding eigenvalues. The obliqueness of a word is described by the angle between the eigenvectors and the *y*-axis. We pick the eigenvector which has a smaller angle and its corresponding eigenvalue as the orientation feature. An illustration of the orientation feature is shown in Fig. 4.4.

|  |
| --- |
| orientation.png  Fig. . Example of orientation feature |

1. ***Intensity feature***

The intensity of handwriting represents the force exerted during writing. To extract the intensity feature, we first convert the image from RGB color space to *YCbCr* color space, written as follow

|  |  |
| --- | --- |
|  | (4.9) |

where *Y* in *YCbCr* color space represents [luminance](https://en.wikipedia.org/wiki/Luminance_(relative)) component and *Cb* and *Cr* are the blue-difference and red-difference [chroma](https://en.wikipedia.org/wiki/Chrominance) components. We use *Y* to represent the intensity of the pixel. The standard deviation and the mean value of *Y* are used as the intensity feature.

### Matched Side Feature

Word side feature is a unique feature in Chinese words. Chinese words can be composed of multiple word side. For example, the word ‘台’ can be decomposed into 2 word side, ‘ㄙ’ and ‘口’. We consider the word side as a smaller word and can be used to extract helpful features. In our experiment, a word side is defined as a minimum connected component in a word. The number of word side may vary between different handwriting images although they are writing the same word because of the writing habits of a person. If a person's handwriting is scribbled, the number of word side may decrease due to the connection between word sides.

In order to find the feature of the corresponding side between different input words, we apply side matching for sides in each input word with sides in the standard word. After that, we extract the matched point feature according to the matching result. In our experiment, the standard word is printed in Microsoft JhengHei [17], which is a typeface included in Windows Vista and Microsoft Office 2007 for Chinese words. Microsoft JhengHei typeface is similar to the typeface of the handwritten word. Therefore, we use it as the typeface for the standard word.

|  |
| --- |
| 正 黑 體  Fig. . Example of words with Microsoft JhengHei typeface |

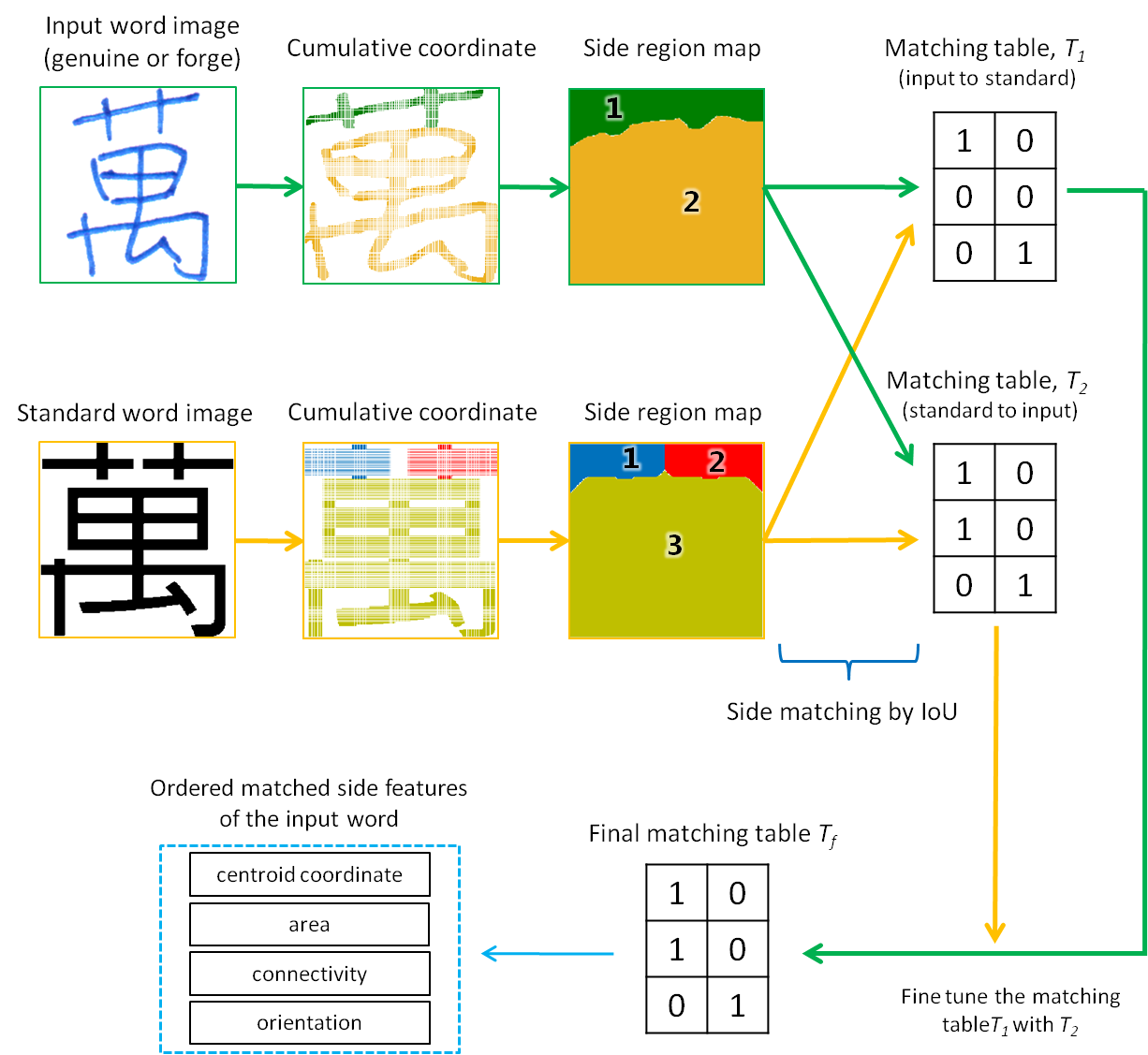


Fig. .. Matched side feature extraction

The overall extracting process of the matched side feature is shown in . The details of the implementation are as followed: First, we convert the stroke pixels from Cartesian coordinate to cumulative coordinate. Let {|=1} be the coordinates of stroke pixels in Cartesian coordinate. The cumulative coordinate is defined as follow:

|  |  |
| --- | --- |
|  | (4.10) |

The cumulative coordinate is invariant under translation and magnification. The value of this coordinate ranges from 0 to 1, but we re-adjust the range to match the original input image then restore it to an image. Side region map is built according to the restored image by labeling every pixel with the nearest side. The map will be presented in a combination of multiple side regions, where a side region is a region that pixels have the same label.

Next, we build two matching tables, *T1* and *T2*, to record the side matching result between the sides of the input word and the standard word. *T1* records the matching result for the sides of the input word, while *T2* records the matching result for the sides of the standard word. Each row of the matching table represents a word side of the standard word, and each column represents a word side of the input word. *T*(*i*, *j*) represents the matching result of the *i*th side of the standard word and the *j*th side of the input word. The values are initialized with 0 and changes to 1 if the two sides are matched. Note that the order of side of the standard word is fixed through the whole process. Intersection over union (IoU) is used to determine the matching result. IoU is an evaluation metric for object detection, which is defined as:

|  |  |
| --- | --- |
|  | (4.11) |

Every side will be matched with the side that has the highest IoU score between the side regions. The final matching table, *Tf*, is accomplished by fine-tuning *T1* with *T2*. For each row in *T1,* if a row does not contain any 1, which means no side from the standard word is matched by the side presenting by the row, we replace the row with the corresponding row in *T2.* The final matching table shows the final matching result of all word sides.

Finally, we extract the matched side features from the side in the input word according to the matching result. The extracting order follows the order of the sides in standard word. If there is more than 1 side that matches the same side in the standard word, we unite all matched word sides. If a word side matches more than 1 side in the standard word, we separate the word side according to the boundary of the side region map of the standard word. The matched side feature includes 4 different features: centroid coordinate feature, area feature, connective feature, and orientation feature.

1. ***Centroid Coordinate Feature***

The centroid coordinate feature is the centroid coordinates of each word side. Let be the coordinate of the pixels in a word side. The centroid coordinate can be calculated by equation (4.7).

1. ***Area Feature***

The area feature is the size of each word side. It is defined as the percentage of the number of pixel of the word side among the total number of stroke pixels in the input word.

1. ***Connective Feature***

The connective feature can be computed from the final matching table, *Tf*. We normalize each column in *Tf* so that the element in each column sum to 1. Now, the non-zero element in each row should be the connective feature. The connective feature shows the connecting status among all word sides of the input word. If the connective feature is *a*, and , that means the word side in the standard word is split into *a* side in the input word. If the connective feature is *b*, and , it means sides in the standard word have united into 1 side in the input word.

1. ***Orientation Feature***

The orientation feature represents the obliqueness of a word side. The extracting method is the same as the one in 4.2.1.D after we change to the coordinates of pixels in the word side.

### Matched point Feature

The matched point feature is a local feature, extracted from certain specific points from a word. For example, in the word ‘一’, the matched point feature would be extracted from the left end and the right end of the word. Same as the matched side feature, in order to find the feature from the corresponding point between different input words, we applied point matching for feature points in each input word with points in the standard word. After that, we extract the matched point feature in a fixed order of point according to the matching result.

From observations on Chinese characters, we first discovered that end-points and turning points on strokes usually contain most of the key features for verification. Since most amateur forgers ignore these details, end-points and turning points are places where they are most likely to reveal their personal handwriting characteristics.

The overall extracting process of the matched point features is shown in Fig. 4.7. The details of the implementation are as follows: First, we applied the thinning process to the input word to obtain the skeleton of the word. On the skeleton, we find all feature points (shown as red points in ) include: branch point, end point, and bending point. The angle of each point is formed by itself and 2 adjacent points of distance *d*. The bending point is the point which has a local minimum angle. The value, d should be adjusted according to the size of the image of the input word. In our experiment, we set d to 1/20 of the input image size. The standard word is also applied with the same process to obtain the feature points. As mention before, for each word, the feature points in the standard word should be fixed in a certain order through the point matching process.

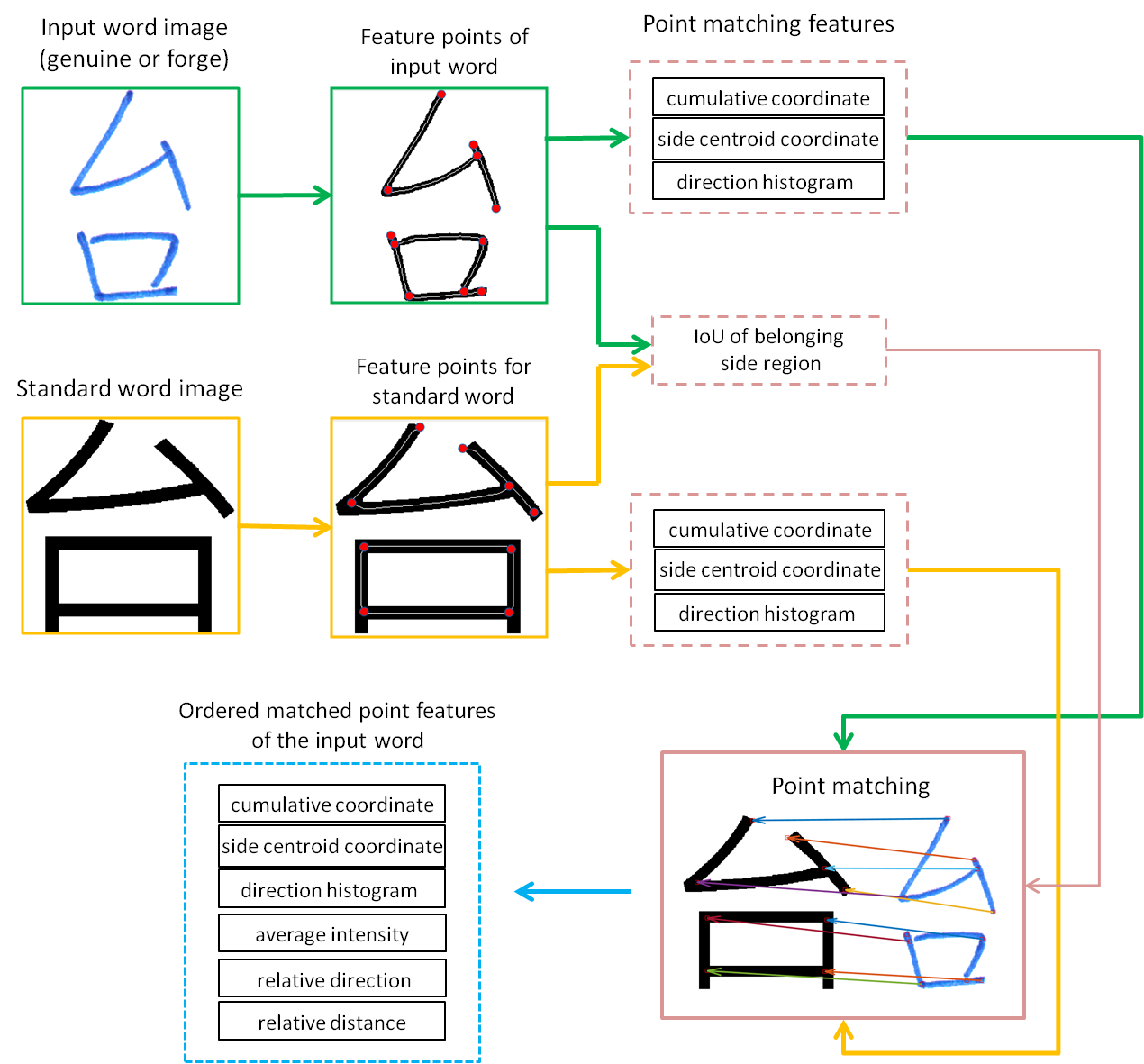


Fig. .. Matched point features extraction

Next, the features for point matching, which we called point matching features are extracted from the feature points. Point matching features include 4 different features: cumulative coordinate feature, direction histogram feature, side centroid coordinate feature, and IoU of belonging side region.

1. ***Cumulative Coordinate Feature***

The cumulative coordinate feature is the cumulative coordinate of the feature point. The cumulative coordinate is converted from the [Cartesian](https://en.wikipedia.org/wiki/Cartesian_coordinates) coordinate by equation (4.10). Cumulative coordinates represent the cumulative percentage of the coordinate relative to the overall stroke pixel coordinate. This coordinate is invariant under translation and magnification.

1. ***Direction Histogram Feature***

The direction histogram feature record the distribution of strokes around the feature point. We place a square of size 3*n* by 3*n*, which the center of the square is the feature point. We further divide the square into 9 smaller grids with a size of *n* pixels, as shown in Fig. 4.8. Exclude the one in the middle; the direction histogram feature is composed of the number of pixel in all other 8 grids and over the total number of the pixels in the 8 grids.

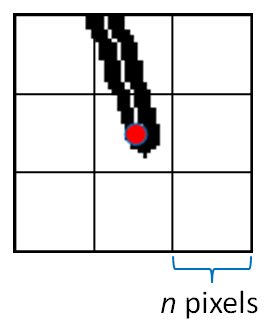


Fig. .**.** Square for direction histogram feature

1. ***Side Centroid Coordinate Feature***

The side centroid coordinate feature records the centroid coordinate of the word side which the feature point belongs to. The centroid coordinate of the word side defined in 4.2.2.A

1. ***IoU of the belonging side region***

The IoU of belonging side region tells the chance of 2 feature points belongs to the same word side. It is computed by the IoU score between the side region of the belonging word side. As a reminder, the side region is a region that contains all pixels share the same nearest side, and the definition of IoU is shown in equation (4.11).

With all 4 point matching features, we use the weighted Euclidean distance, introduced Chapter 3.1 to measure the distance between feature point. The feature distance is formulated as follow

|  |  |
| --- | --- |
|  | (4.12) |

where *a1, a2, a3, a4*, are the weights. We conduct the point matching by selecting the nearest feature point.

After matching, we further extract 3 different features from the matched points: average intensity feature, relative direction feature, relative distance feature, and combine them with the direction histogram feature, the direction histogram feature, and the side centroid coordinate feature to be the matched point feature.

1. *Average Intensity Feature*

The average intensity feature is computed by averaging the intensity of the stroke pixels in a square of size *q* by *q*, which the center of the square is the feature point. The intensity of a pixel can be defined as the *Y* in equation (4.9).

1. *Relative Direction Feature*

The relative direction feature tells the relative direction between 2 feature points, defined as follow:

|  |  |
| --- | --- |
|  | (4.13) |
| where if[*x*, *y*]≠[0, 0], . | |

1. *Relative Distance Feature*

The relative distance feature shows the distance between 2 feature points. The distance between the coordinates of the 2 feature points is defined by the Euclidean distance, defined in equation .

## Feature Selection and Classification

Considering that the extracted features may have different importance for a different word, we use k-value to select the features that are important for each word. The formula of k-value is written as:

|  |  |
| --- | --- |
|  | (4.14) |

where and are the means of genuine words and forged words, and and are the standard deviations of genuine words and forged words respectively. We select the features with k-value greater than 0.2 for handwriting verification.

Support vector machine (SVM) is a well known supervised learning method for classification and [regression](https://www.sciencedirect.com/topics/computer-science/regression) analysis. A more precise definition would state that [SVM](https://www.sciencedirect.com/topics/computer-science/support-vector-machine) builds a [hyperplane](https://www.sciencedirect.com/topics/computer-science/hyperplanes) to classify all inputs in a high-dimensional space. As illustrated in Section , the goal of SVM is to maximize the margin between the hyperplane and the support vectors, which are the closest values to the classification margin. For this reason, the SVM is maintaining high accuracy when there is a relatively small amount of data.

In 2003, Lin developed a simple, easy-to-use, fast and efficient SVM package . Here we also adopt LIBSVM to perform verification. The most common SVM is the [linear SV](https://www.sciencedirect.com/topics/computer-science/linear-classifiers)M model. The model uses the linear kernel  to establish the hyperplane. Compare to other SVM kernels, the linear kernel is simpler and therefore it does not easily overfit. Due to the fact that the collected database is relatively small, we choose the linear kernel to build our model.

# Database

In our study, we a Chinese handwriting database for writer verification. The database should include a set of genuine words and a set of corresponding forge words. However, most of the available public databases were built for writer identification. The content in these collected handwritten databases is not based on forgery. Although the databases for signature verification are closer to our request, seldom are in written in Chinese. Therefore, we have collected a database on our own. The details of this database would be presented in this chapter.

## Introduction of Database

This database includes 44 Chinese words. The 44 Chinese words were chosen based on two criteria: First, these words need to contain words with complex structure and simple structure. Second, these words should be widely used. In the end, we select 44 words, consist of 5 groups of words:

1) ‘台’, ‘大’, ‘電’, ‘信’, ‘工’, ‘程’, ‘學’, ‘研’, ‘究’, ‘所’.

2) ‘個’, ‘十’, ‘百’, ‘千’, ‘萬’, ‘億’, ‘兆’, ‘京’.

3) ‘東’, ‘南’, ‘西’, ‘北’, ‘春’, ‘夏’, ‘秋’, ‘冬’.

4) ‘子’, ‘丑’, ‘寅’, ‘卯’, ‘甲’, ‘乙’, ‘丙’, ‘丁’.

5) ‘你’, ‘我’, ‘的’, ‘是’, ‘在’, ‘要’, ‘到’, ‘來’, ‘不’, ‘有’.

The first group is the name of the “Graduate Institute of Communication Engineering”, which is the name of our department. The second group is about unit numbers, such as unit(個), ten(十), hundred(百), thousand(千), ten thousand(萬). The third group is the four directions; east(東), south(南), west(西), north(北), and the four-season; spring(春), summer(夏), autumn(秋), winter(冬). The fourth group is a combination of the first four characters in Earthly Branches and the first four characters in the Heavenly Stems. The fifth group is selected from the 40 most commonly used Chinese characters. Table 5.1 shows the number of stroke and side of each word. The numbers are calculated from the standard word we used for side matching and point matching in Chapter 4.2.2 and 4.2.3.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 5.1. The number of stroke and side of each word | | | | | | | | | | |
| **Group 1** | **台** | **大** | **電** | **信** | **工** | **程** | **學** | **研** | **究** | **所** |
| # stroke | 5 | 3 | 13 | 9 | 3 | 12 | 16 | 9 | 7 | 8 |
| # side | 2 | 1 | 6 | 5 | 1 | 3 | 4 | 2 | 2 | 2 |
| **Group2** | **個** | **十** | **百** | **千** | **萬** | **億** | **京** | **兆** |  |  |
| # stroke | 10 | 2 | 6 | 3 | 13 | 15 | 8 | 6 |  |  |
| # side | 3 | 1 | 1 | 1 | 3 | 2 | 3 | 4 |  |  |
| **Group3** | **東** | **西** | **南** | **北** | **春** | **夏** | **秋** | **冬** |  |  |
| # stroke | 8 | 6 | 9 | 5 | 9 | 10 | 9 | 5 |  |  |
| # side | 1 | 1 | 2 | 2 | 1 | 1 | 3 | 3 |  |  |
| **Group4** | **甲** | **乙** | **丙** | **丁** | **子** | **丑** | **寅** | **卯** |  |  |
| # stroke | 5 | 1 | 5 | 2 | 3 | 4 | 11 | 5 |  |  |
| # side | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 2 |  |  |
| **Group5** | **你** | **我** | **的** | **是** | **在** | **要** | **到** | **來** | **不** | **有** |
| # stroke | 7 | 7 | 8 | 9 | 6 | 9 | 8 | 8 | 4 | 6 |
| # side | 4 | 1 | 3 | 2 | 1 | 2 | 3 | 3 | 1 | 1 |

This database collects genuine and forges handwritings from 37 volunteers. The first volunteer will write the 44 words in its writing style, each for 40 times. The other 36 people imitate the 44 words written by the first volunteer, each for 1 time. Eventually, the database contains 44 words, each has 40 genuine words and 36 forge words. Since our verification system is character-wise, there are 40+36 = 76 images for each word in the experiment.

## Questionnaire Collecting

We used an anonymous questionnaire to collect word data from volunteers. Subjects will practice 2 times before imitating each word. During practice, we will provide correction lines, , to make it easier for the subject to find tips for imitation. In the formal imitation, we only give the subject a blank area to be close to the actual situation. All the samples were scanned by Brother MFC-8860DN at the resolution of dpi and the type set as 24-bit color to be a raw collected data. A collected questionnaire sample is shown in Fig. 5.1.

|  |
| --- |
| C:\Users\Chien_Yu\Desktop\new_handwriting_data_1\image015.bmp  Fig. .. A sample of the collected questionnaire sample. |

## Segmentation

To separate all the words from the collected questionnaire sample are challenging. Although the vertical coordinate of the written words is limited in a certain range, the horizontal coordinate may vary in a wide range. Some people might write words sparsely, while others will write words closely. In order to find the actual location of each word, we adopt a flat kernel to do the convolution with the collected questionnaire sample. The overall process is shown as follows

1. Since the vertical coordinate of the written is fixed, we first segment the words from the known range of x, y coordinate.

|  |
| --- |
| Fig. .. Segmented words from the known range of x, y coordinate |

1. We estimate the size of the words. We have two assumptions, first, each written word is distributed in a square area. Second, the size of words written by the same person is similar. Thus, the size of a word can be represented by a block. *n* is simulated by the , where y is the coordinate of all the stroke pixels in the segmented image.
2. A flat filter of size is adapted to do the convolution with the segmented image. We assume the place where has a local maximum response is the location of a written word. We refine the result by averaging the coordinates which the distance between is less than 0.6*n*. Now, we obtain the location of each word.

|  |
| --- |
| (a) |
| (b) |
| Fig. .. (a) Place where has a local maximum response. (b) The obtained location of words. |

1. For each obtained location, we capture a square region around it. The size of the square region is which is much bigger than the estimated word size. It ensures the captured window contains a complete word image. The captured window will be refined by eliminating the stroke pixels from the edge of the word image until it meets a blank gap. The width of the blank gap is set to 0.2*n*.

|  |  |
| --- | --- |
| (a) | (b) |
| Fig. .. (a) Originally captured word image. (b) Refined word image by eliminating the extra stroke pixels | |

1. Let x, y be the stroke pixel coordinates in the word image after step 4. We once more capture a square region, centered by . The size of the square region is , where . This makes the word centered in a square image and the .

|  |
| --- |
| C:\Users\Chien_Yu\Desktop\new_handwriting_data_2_test\23\froge_image\042.bmp  Fig. .. The relocated word image. |

1. Resize the image to any image size you desired by the interpolation method. We resize the word image to .

## Image Noise Reduction

When writing or scanning, it may cause stains inadvertently due to human factors. These stains are noise that is not relevant to our experiment, thus needs to be removed. These noises can be removed from two criteria. First: Generally, the areas of these noises are quite small. Therefore, we calculate the area of each minimum connection unit. If the area is under a threshold, we determine that the minimum connection unit is a noise point. Second, the intensity of noise is often much lower than the stroke pixels. Hence, we remove the pixels, which the intensity is lower than a threshold. The intensity of pixels can be represented by the *Y* value in the *YCbCr* color space, written as follows,

|  |  |
| --- | --- |
|  | (5.1) |

where *Y* in *YCbCr* color space represents [luminance](https://en.wikipedia.org/wiki/Luminance_(relative)) component and *Cb* and *Cr* are the blue-difference and red-difference [chroma](https://en.wikipedia.org/wiki/Chrominance) components. With the two noise removal criteria, the noise in the word image can be effectively reduced. An example is shown in Fig. 5.6. It illustrates the effect of noise reduction by comparing the word image before and after the process.

|  |  |
| --- | --- |
| (a) | (b) |
| Fig. .. (a) The raw word image. (b) The word image after noise reduction. | |

# Experiment Result

## Model Evaluation

The dataset we used is introduced in Chapter 5. The dataset includes 44 words. For each word, there are 40 genuine words and 36 forge words, totally 40+36 = 76 images for each word in our experiment. Since the dataset we used is relatively small, we use *k*-fold cross-validation to obtain a reliable performance. An illustration is shown in Fig. 6.1. *K*-fold cross-validation randomly divides the data into *k* sets, then treat one set as validation data and the remaining *k*-1 sets as training data, and repeat until each set is used as validation data. The cross-validation process will repeat *k* times. We average the *k* evaluating score from the *k* cross-validation as the final score. The value *k* is set to 5 for our experiment, which means 80% of data are used for training and the rest 20% are used for testing in each iteration.

|  |
| --- |
| kfold.png  Fig. .. *k-*fold cross-validation |

We use 4 evaluation metrics in our experiment: precision, recall, f1score, accuracy. The definitions of these measurements are as follows

|  |  |  |
| --- | --- | --- |
| Table 6.1 The confusion matrix | | |
| Ground truth  Prediction | genuine | forge |
| genuine | True positives(TP) | False positives(FP) |
| forge | False negatives(FN) | True negatives(TN) |

|  |  |
| --- | --- |
|  | (6.1) |
|  | (6.2) |
|  | (6.3) |
|  | (.4) |

Table 6.1 shows a confusion matrix for binary classification with four outcomes: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). TP is the number of genuine data that are been correctly predicted, while FN is the number of genuine data that are been predicted wrong. FP is the number of forge data that are been predicted wrong, while TN is the number of forge data that are been correctly predicted. Precision is the percentage of images predicted to be genuine that were actually genuine. Recall is the percentage of genuine images that were predicted to be genuine. F1 score, which considers both the precision and the recall, is the harmonic average of precision and recall.

## Simulation Result

We compare our proposed algorithm with several related other methods including using moment + projection features, point-matching position features[12], log-Gabor + advanced moments + GLCM[17], the LBP[8], and the CNN based model[3]. We also show the performance of the different combination of three features, global features(G), matched side feature(MF), matched point feature(MP), which are included in our proposed method. Table 6.2 ~ Table 6.6 record the accuracy of each word under different methods, while Table 6.7 records the accuracy, precision, recall, and F1score of the whole dataset. The best performance among all methods is colored in green.

As shown in Table 6.2 ~ Table 6.6, the method we proposed has the best accuracy among 20 of the 44 words and 34 words have accuracy higher than 95%. As shown in Table 6.7, our proposed method reaches 96.27% in precision, 95.86% in recall, 96.06 % in f1score and 95.85% in accuracy, which are the best in individual among all methods. Our proposed method outperformed the 2nd best method by 2% in accuracy. The standard deviation of accuracy shows the stableness of the method. Our proposed method has the lowest standard deviation among all methods, which means it is more robust while dealing with different words.

Furthermore, we found that the matched point feature to be very helpful for Chinese handwriting verification. The combination of the matched point feature with the global features improved the accuracy by 7.3% from only using the global features. The accuracy of our proposed method decreases by 7.6% without the matched point feature.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CNN | M+P | Huang[12] | Lee[17] | Yilmaz[8] | G | G+MS | G+MP | Proposed  (G+MP+MS) | Group 1 | Table 6.2 Accuracy of words in group1 |
| 0.867 | 0.803 | 0.855 | 0.855 | 0.961 | 0.875 | 0.875 | 0.988 | 0.988 | 台 |
| 0.733 | 0.909 | 0.833 | 0.805 | 0.936 | 0.925 | 0.875 | 0.925 | 0.938 | 大 |
| 1.000 | 0.868 | 0.789 | 1.000 | 0.947 | 0.900 | 0.900 | 0.975 | 0.963 | 電 |
| 0.938 | 0.845 | 0.988 | 0.909 | 0.936 | 0.863 | 0.938 | 0.963 | 0.988 | 信 |
| 0.867 | 0.884 | 0.741 | 0.843 | 0.961 | 0.838 | 0.750 | 0.938 | 0.963 | 工 |
| 1.000 | 0.779 | 0.922 | 0.909 | 0.922 | 0.888 | 0.838 | 0.950 | 0.950 | 程 |
| 0.933 | 0.961 | 0.987 | 0.948 | 0.987 | 0.912 | 0.948 | 0.963 | 0.961 | 學 |
| 0.800 | 0.842 | 0.882 | 0.842 | 0.882 | 0.875 | 0.863 | 0.975 | 0.963 | 研 |
| 0.867 | 0.868 | 0.921 | 0.868 | 0.987 | 0.925 | 0.900 | 0.938 | 0.938 | 究 |
| 1.000 | 0.947 | 0.855 | 0.947 | 0.921 | 0.850 | 0.888 | 0.975 | 1.000 | 所 |
| 0.900 | 0.871 | 0.877 | 0.893 | 0.944 | 0.885 | 0.877 | 0.959 | 0.965 | Avg |
| 0.090 | 0.597 | 0.079 | 0.060 | 0.032 | 0.030 | 0.056 | 0.021 | 0.021 | std |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CNN[3] | M+P | Huang[12] | Lee[17] | Yilmaz[8] | G | G+MS | G+MP | Proposed  (G+MP+MS) | Group 2 | Table 6.3 Accuracy of words in group 2 |
| 0.933 | 0.961 | 0.934 | 0.882 | 0.974 | 0.963 | 0.938 | 0.950 | 0.963 | 個 |
| 0.867 | 0.750 | 0.908 | 0.776 | 0.921 | 0.850 | 0.800 | 0.950 | 0.913 | 十 |
| 0.933 | 0.934 | 0.908 | 0.921 | 0.908 | 0.913 | 0.913 | 0.950 | 0.950 | 百 |
| 1.000 | 0.816 | 0.961 | 0.855 | 0.947 | 0.863 | 0.900 | 0.925 | 0.950 | 千 |
| 1.000 | 0.803 | 0.974 | 0.882 | 0.921 | 0.850 | 0.788 | 0.938 | 0.925 | 萬 |
| 0.933 | 0.816 | 0.868 | 0.921 | 0.947 | 0.900 | 0.975 | 0.975 | 0.975 | 億 |
| 0.867 | 0.800 | 0.828 | 0.893 | 0.842 | 0.763 | 0.813 | 0.925 | 0.963 | 京 |
| 1.000 | 0.855 | 0.947 | 0.908 | 0.947 | 0.888 | 0.863 | 0.950 | 0.925 | 兆 |
| 0.950 | 0.842 | 0.916 | 0.880 | 0.926 | 0.873 | 0.873 | 0.945 | 0.945 | Avg |
| 0.047 | 0.072 | 0.049 | 0.047 | 0.040 | 0.058 | 0.069 | 0.016 | 0.022 | std |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CNN[3] | M+P | Huang[12] | Lee[17] | Yilmaz[8] | G | G+MS | G+MP | Proposed  (G+MP+MS) | Group 3 | Table 6.4 Accuracy of words in group 3 |
| 0.867 | 0.895 | 0.895 | 0.947 | 0.934 | 0.863 | 0.838 | 0.863 | 0.888 | 東 |
| 0.933 | 0.882 | 0.895 | 0.855 | 0.895 | 0.881 | 0.948 | 0.948 | 0.948 | 西 |
| 0.733 | 0.829 | 0.908 | 0.868 | 0.895 | 0.814 | 0.907 | 0.933 | 0.973 | 南 |
| 1.000 | 0.895 | 0.908 | 0.947 | 0.947 | 0.948 | 0.935 | 0.987 | 1.000 | 北 |
| 0.467 | 0.868 | 0.842 | 0.842 | 0.961 | 0.922 | 0.922 | 0.922 | 0.960 | 春 |
| 1.000 | 0.933 | 0.920 | 0.920 | 0.933 | 0.975 | 0.935 | 0.973 | 0.987 | 夏 |
| 0.933 | 0.895 | 0.934 | 0.908 | 0.974 | 0.875 | 0.838 | 0.975 | 0.975 | 秋 |
| 0.800 | 0.908 | 0.921 | 0.974 | 0.961 | 0.888 | 0.900 | 0.938 | 0.963 | 冬 |
| 0.842 | 0.888 | 0.903 | 0.908 | 0.937 | 0.896 | 0.903 | 0.942 | 0.962 | Avg |
| 0.178 | 0.031 | 0.028 | 0.048 | 0.030 | 0.051 | 0.043 | 0.040 | 0.034 | std |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CNN[3] | M+P | Huang[12] | Lee[17] | Yilmaz[8] | G | G+MS | G+MP | Proposed  (G+MP+MS) | Group 4 | Table 6.5 Accuracy of words in group 4 |
| 0.867 | 0.816 | 0.829 | 0.842 | 0.961 | 0.838 | 0.838 | 0.938 | 0.950 | 甲 |
| 0.933 | 0.792 | 0.805 | 0.832 | 0.870 | 0.805 | 0.767 | 0.832 | 0.869 | 乙 |
| 1.000 | 0.855 | 0.882 | 0.895 | 0.974 | 0.863 | 0.825 | 0.950 | 0.963 | 丙 |
| 0.933 | 0.855 | 0.789 | 0.842 | 0.895 | 0.913 | 0.913 | 0.913 | 0.963 | 丁 |
| 0.867 | 0.803 | 0.776 | 0.882 | 0.908 | 0.788 | 0.800 | 0.900 | 0.938 | 子 |
| 1.000 | 0.842 | 0.895 | 0.855 | 0.921 | 0.800 | 0.700 | 0.975 | 0.963 | 丑 |
| 1.000 | 0.855 | 0.947 | 0.921 | 0.974 | 0.850 | 0.963 | 0.963 | 0.963 | 寅 |
| 1.000 | 0.842 | 0.947 | 0.895 | 0.934 | 0.888 | 0.900 | 1.000 | 0.988 | 卯 |
| 0.950 | 0.833 | 0.859 | 0.870 | 0.929 | 0.838 | 0.841 | 0.934 | 0.939 | Avg |
| 0.059 | 0.025 | 0.069 | 0.032 | 0.038 | 0.051 | 0.083 | 0.051 | 0.063 | Std |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CNN[3] | M+P | Huang[12] | Lee[17] | Yilmaz[8] | G | G+MS | G+MP | Proposed  (G+MP+MS) | Group 5 | Table 6.6 Accuracy of words in group 5 |
| 0.933 | 0.921 | 0.882 | 0.921 | 0.974 | 0.875 | 0.950 | 0.988 | 0.988 | 你 |
| 0.933 | 0.829 | 0.908 | 0.961 | 0.961 | 0.863 | 0.888 | 0.963 | 0.963 | 我 |
| 0.933 | 0.829 | 0.895 | 0.974 | 0.961 | 0.938 | 0.925 | 1.000 | 0.988 | 的 |
| 1.000 | 0.829 | 0.895 | 0.868 | 0.961 | 0.813 | 0.863 | 0.950 | 0.988 | 是 |
| 0.933 | 0.842 | 0.829 | 0.829 | 0.974 | 0.888 | 0.888 | 0.975 | 0.988 | 在 |
| 1.000 | 0.895 | 0.855 | 0.974 | 0.921 | 0.913 | 0.938 | 0.900 | 0.950 | 要 |
| 0.867 | 0.855 | 0.921 | 0.974 | 0.895 | 0.875 | 0.900 | 0.975 | 0.975 | 到 |
| 0.933 | 0.829 | 0.921 | 0.921 | 0.882 | 0.913 | 0.888 | 0.950 | 0.975 | 來 |
| 0.800 | 0.895 | 0.882 | 0.816 | 0.947 | 0.863 | 0.863 | 0.913 | 0.900 | 不 |
| 0.867 | 0.882 | 0.961 | 0.934 | 0.961 | 0.875 | 0.938 | 0.950 | 0.963 | 有 |
| 0.926 | 0.861 | 0.895 | 0.917 | 0.943 | 0.881 | 0.904 | 0.956 | 0.968 | Avg |
| 0.062 | 0.034 | 0.036 | 0.059 | 0.032 | 0.035 | 0.032 | 0.031 | 0.027 | Std |

Table 6.7 The statistical results of the whole database.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **Precision** | **Recall** | **F1score** | **Accuracy** | **Accuracy std** |
| Proposed  (G+MP+MS) | 0.9627 | 0.9586 | 0.9606 | 0.9585 | 0.028 |
| G+MP | 0.9523 | 0.9468 | 0.9485 | 0.9467 | 0.040 |
| G+MS | 0.8942 | 0.8815 | 0.8877 | 0.8816 | 0.059 |
| G | 0.8910 | 0.8763 | 0.8835 | 0.8763 | 0.045 |
| Yilmaz[8] | 0.9382 | 0.9354 | 0.9368 | 0.9367 | 0.034 |
| Lee[17] | 0.8956 | 0.8964 | 0.8960 | 0.8945 | 0.052 |
| Huang[12] | 0.8896 | 0.8906 | 0.8901 | 0.8896 | 0.057 |
| M+P | 0.8593 | 0.8585 | 0.8589 | 0.8594 | 0.049 |
| CNN[3] | 0.9381 | 0.9034 | 0.9204 | 0.9168 | 0.103 |

|  |
| --- |
| Fig. .. Accuracy of words with a different number of strokes |

Fig. 6.2 shows the accuracy of our proposed method versus the different number of word strokes. The lowest accuracy appears at the word with the number of strokes equals to 1. Accuracy rises wh**e**n the number of stroke increases. Generally, the more the number of strokes is, the higher the accuracy rate is. The reason is that it is hard to extract sufficient feature for handwriting verification from words with fewer stroke. Besides, the fewer the stroke is, the easier to forge the word since the forgers only need to pay attention to a small number of strokes. On the other hand, the words with a larger number of strokes provide more opportunity to extract features, and also, the complex structure makes them difficult to be forged.

# Conclusions and Future Work

## Conclusions

We proposed an algorithm for the word-wise Chinese handwriting verification based on feature extraction technique. The algorithm extracts global features such as moment features, stroke after erosion features, projection features, orientation features, and intensity features. Moreover, our algorithm takes advantage of the complicated structure of Chinese words to extract various local features, such as matched side feature and matched point feature. The matched side feature extracted features from the word side, which is a smaller part of a Chinese word. The matched point feature extract feature from feature points such as branch points, end point, and bending point. These local features help the algorithm to observe the Chinese word in different magnitude levels. Together with the global features, these three features can more fully describe the whole picture of the whole word. *K*-value is used for feature selection and SVM with a linear kernel is used as the classifier. The experiment result shows our algorithm reaches 95.85% accuracy in handwriting verification and outperforms multiple existing methods. Our proposed algorithm also shows robustness while dealing with different words. The matched point feature is found to be very helpful for Chinese handwriting verification. It increased the accuracy by 7.3% in our algorithm. Besides, we found that fewer stroke words are harder to verify for our algorithm. Our algorithm achieves better performance when the word has more strokes.

## Future Work

Although our proposed method achieves high accuracy, there are several aspects that need to be improved. First, our proposed method requires a lot of time to extract all the word features. We can extract some more sophisticated and representative features from the matched points or matched side and reduce the total number of features while still maintaining high accuracy.

Second, the point matching and the side matching process are sensitive to the quality of words. If a person's handwriting is scribbled, the detection of the word side and the feature point will be misguided. Eventually leads to a poor feature quality. So a better approach is needed to accurately detect candidate feature points and precisely segment the connected word side from all the disturbance.

Third, our verification system is a character-wise system, which means it requires a number of genuine and forges word pair for the same word. It would be hard to be used in practical applications since such data is hard to require in normal life. The problem can be overcome by changing the system to person-wise system. The training data would be a whole handwritten script or arbitrary handwritten words instead of some pre-selected words. After training, the system should be able to verify the genuineness of any words written by anyone.

1. REFERENCE
2. N. A. Murshed, F. Bortolozzi, and R. Sabourin, "Binary image compression using identity mapping backpropagation neural network", Electronic Imaging '97. International Society for Optics and Photonics, 1997, pp. 29-35.
3. N. A. Murshed, R. Sabourin, and F. Bortolozzi, "A cognitive approach to off-line signature verification", International Journal of Pattern Recognition and Artificial Intelligence, vol. 11, no. 05, 1997, pp. 801-825.
4. H. Khalajzadeh, M. Mansouri, and M. Teshnehlab, "Persian Signature Verification using Convolutional Neural Networks", International Journal of Engineering Research and Technology, vol. 1, 2012.
5. L. Hafemann, R. Sabourin, and L. Oliveira, "Writer-independent feature learning for offline signature verification using convolutional neural networks", Neural Networks The 2016 International Joint Conference on, 2016.
6. D. Field, "Relations between the statistics of natural images and the response properties of cortical cells," Journal of the Optical Society of America A, 1987, pp. 2379-2394
7. What Are Log-Gabor Filters and Why Are They Good? <http://www.csse.uwa.edu.au/~pk/research/matlabfns/PhaseCongruency/Docs/convexpl.html>
8. T. Ojala, M. Pietikäinen, and D. Harwood. “A comparative study of texture measures with classification based on featured distributions.” Pattern Recognition 29 , 1996, pp.51-59.
9. M. B. Yilmaz, B. Yanikoglu, C. Tirkaz, and A. Kholmatov, "Offline signature verification using classifier combination of HOG and LBP features," 2011 International Joint Conference on Biometrics (IJCB), Washington, DC, 2011, pp. 1-7.
10. M. A. Ferrer, A. Morales, and J. F. Vargas, "Off-line signature verification using local patterns," CONATEL 2011, Catolica San Pablo, Arequipa, 2011, pp. 1-6.
11. T. Jabid, M. H. Kabir and O. Chae, "Facial expression recognition using Local Directional Pattern (LDP)," 2010 IEEE International Conference on Image Processing, Hong Kong, 2010, pp. 1605-1608.
12. G. Cash, M. Hatamian. "Optical character recognition by the method of moments."Computer Vision, Graphics, and Image Processing, 1987, pp. 291-310.
13. J. Y. Huang and J. J. Ding, “Automatic handwriting verification algorithm in Chinese words using point-matching position features,” Computer Vision, Graphics, and Image Processing, Tainan, Taiwan, Aug. 2018.
14. M. K. Hu. "Visual pattern recognition by moment invariants." Information Theory, IRE Transactions on 8.2, 1962. pp. 179-187.
15. J. Flusser, T. Suk. "Pattern recognition by affine moment invariants." Pattern recognition 26.1, 1993. pp. 167-174.
16. K. D. Tsirikolias, and B. G. Mertzios. "Statistical pattern recognition using efficient two-dimensional moments with applications to character recognition."Pattern Recognition 26.6, 1993, pp. 877-882.
17. Y. C. Chim, A. A. Kassim, and Y. Ibrahim. "Character recognition using statistical moments." Image and vision Computing 17.3, 1999, pp. 299-307.
18. P. X. Lee, J. J. Ding, T. C. Wang, and Y. C. Lee. "Automatic Handwriting verification Algorithm for Chinese Words Semi-Global Features and Adaptive Classifier."International Conference on Multimedia & Expo Workshops (ICMEW), 2018, pp. 1–6.
19. Y. Sun, W. Liu, Y. Wang, "United moment invariants for shape discrimination, "IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003, pp. 88-93.
20. G. F. Fitriana, "Handwriting Digit Recognition using United Moment Invariant feature extraction and Self Organizing Maps," 2014 Third ICT International Student Project Conference (ICT-ISPC), 2014, pp. 43-46.
21. R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-3, no. 6, pp. 610-621, Nov. 1973.
22. J. F. Vargas, M. A. Ferrer, C. M. Travieso, and J. B. Alonso, "Off-line signature verification based on grey level information using texture features." Pattern Recognition, vol 44-2, 2011, pp. 375-385.
23. D. G. Lowe, "Distinctive image features from scale-invariant keypoints."International journal of computer vision, 2004, pp. 91-110.
24. J. Woodard, M. Lancaster, A. Kundu, D. Ruiz, and J. Ryan, "Writer recognition of Arabic text by generative local features," 2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS), Washington, DC, 2010, pp. 1-7.
25. M. Sreeraj and S. M. Idicula, "The effect of SIFT features as content descriptors in the context of automatic writer identification in Malayalam language," 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems, St. Petersburg, 2012, pp. 613-617.
26. X. Wu, Y. Tang, and W. Bu, "Offline Text-Independent Writer Identification Based on Scale Invariant Feature Transform," in IEEE Transactions on Information Forensics and Security, vol. 9, no. 3,2014, pp. 526-536.
27. Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm. <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>
28. S. Lloyd, "Least squares quantization in PCM," in IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137, March 1982.
29. <http://ai.stanford.edu>
30. C. Cortes, and V. Vladimir. "Support-vector networks." Machine learning 20.3, 1995, pp. 273-297.

1. <https://medium.com/@chih.sheng.huang821/機器學習-支撐向量機-support-vector-machine-svm-詳細推導-c320098a3d2e>
2. B. E. Boser, I. M. Guyon, and V. N. Vapnik. "A training algorithm for optimal margin classifiers." In D. Haussler, editor, 5th Annual ACM Workshop on COLT, 1992, pp. 144–152.