IN4393 Computer Vision Final Project Report: Identification and Verification of Offline Handwritten Signature

Xiaoxu Gao 4504348

July 7, 2016

1. PROBLEM STATEMENT

A handwritten signature is a depiction that a person writes on documents as a proof of identification. The main function of a signature is to prove authentication for a document, especially in financial transactions. Hence, it's essential to develop a signature identification and verification system to help people identify their identification or detect fake signatures more effectively. There are 2 types of system: online system and offline system. Online system uses dynamic features, such as the motion of the stylus, pen pressure, locations, time and etc. While offline system uses static features, such as 2-D scanned images of handwritten signatures. In this project, offline system is used due to its simplified recording process and lower implementation cost.

To make the system more practical and valuable, both identification model and verification model are implemented. The aim of identification model is to identify the writer of a questioned signature given writing examples of several known writers. While the aim of verification model is to tell whether two questioned signatures are from the same writer or different writers, normally it's used in detecting forged signatures.

In total, this project implements an approach for signature identification and verification in an offline environment based on GSC (Gradient, Structural and Concavity) features and KNN Classifier. The detailed algorithm will be explained later.

2. ALGORITHM DESCRIPTION

2.1 GSC Algorithm

GSC (Gradient, Structural, Concavity) recognition algorithm has been used successfully in several document reading applications. It takes a quasi-multiresolution approach to feature generation. Several distinct feature types are applied at different scales in the image, which measure the image characteristics at local, intermediate and large scales.

Gradient Features

The gradient features are computed by convolving two 3*3 Sobel operators on the binary image. Sobel operators can approximate the x and y derivatives in the image. The gradient is a vector with magnitude and direction, but only direction is used in the computation of a feature vector. The range of direction is from 0 to 360, which can be split into 12 non-overlapping regions. The gradient direction of each pixel is put in the corresponding region and then using histogram. A threshold is applied to the histogram and the corresponding feature is set to "1" if its count exceeds the threshold. Therefore, the gradient feature is a 12-bit binary vector.

Structural Features

The structural features capture certain patterns in the gradient map, such as "mini-strokes" of the image. A set of 12 rules is applied to each pixel. Each rule checks a particular pattern of eight neighboring pixels. The rules used in this project is listed in Table 1. If any rule is satisfied in the image, its corresponding bit in the feature vector is set to "1". In the end, the structural feature is also a 12-bit binary vector.

Concavity Features

The concavity feature is a 8-bit binary vector which consists of three parts. 1) Density Feature: This single-bit feature is computed by counting the number of image pixels that fall into each image grid. If the amount exceeds the threshold, the binary is set to "1". 2) Large-Stroke Feature: This feature analyzes large horizontal and vertical strokes in the image. The presence of strokes are determined by testing for stroke lengths above a threshold. One bit for horizontal strokes and the other bit for vertical strokes. 3) Concavity Feature: An operator which is like a star is used. The operator shoots rays in four directions. A ray can hit an image pixel or the image border. The class of each pixel is determined by applying rules to each pixel.

Table 1. Structural feature definitions

Rule	Description	Neighbour 1 (Range)	Neighbour 2 (Range)
1	Horizontal line 1	N 0 (2,3,4)	N 4 (2,3,4)
2	Horizontal line 2	N 0 (8,9,10)	N 4 (8,9,10)
3	Vertical line 1	N 2 (5,6,7)	N 6 (5,6,7)
4	Vertical line 2	N 2 (1,0,11)	N 6 (1,0,11)
5	Diagonal rising 1	N 5 (4,5,6)	N 1 (4,5,6)
6	Diagonal rising 2	N 5 (10,11,0)	N 1 (10,11,0)
7	Diagonal falling 1	N 3 (1,2,3)	N 7 (1,2,3)
8	Diagonal falling 2	N 3 (7,8,9)	N 7 (7,8,9)
9	Corner 1	N 2 (5,6,7)	N 0 (2,3,4)
10	Corner 2	N 6 (5,6,7)	N 0 (1,0,11)
11	Corner 3	N 4 (8,9,10)	N 2 (1,0,11)
12	Corner 4	N 6 (1,0,11)	N 4 (2,3,4)

2.2 KNN Classifier

KNN is a powerful classifier in machine learning problem. It computes a classification function by using the labeled training points as nodes in high dimensional space (512 dimensions in this project). An object is classified by a majority vote of its neighbors. In this project, I used existing KNN classifier in PRTools, which is a powerful MATLAB package for machine learning problems.

3. IMPLEMENTATION

In this project, I implemented pre-processing part, GSC Algorithm and similarity measurement by myself. The only library I used is PRTools. All codes are original.

3.1 Pre-processing

In the pre-processing part, there are basically three stages. The first stage is **binarization**. I used MATLAB graythresh function which is based on Otsu's method to define the threshold of the image and then convert an intensity image to a binary image. The second stage is **rotation normalization**. The signature curve is rotated until the axis of least inertia coincided with the horizontal axis. The signature curve is represented in the following equation:

$$C = \left\{ X(i) = \begin{bmatrix} u(i) \\ v(i) \end{bmatrix}, i = 1...N \right\}$$

$u(i)$ is x-coordinate of the ith pixel in the signature curve, $v(i)$ is y-coordinate of the ith pixel in the signature curve. Then I calculated the center of the signature and the second order moments of the signature. The orientation of the axis of least inertia was given by the orientation of the least eigen vector of the following matrix:

$$I = \begin{pmatrix} \bar{u}^2 & \bar{uv} \\ \bar{uv} & \bar{v}^2 \end{pmatrix}$$

The last stage is subdivision. After removing unnecessary black space, I subdivided the image into 4*4 grids. The aim of this step is to do analysis and get feature vectors for each grid, which can get more information than directly using the original image. The results of pre-processing can be seen from Figure 1 to Figure 3.

3.2 GSC Feature Extraction

In the GSC feature extraction part, I get a 32-bit binary vector for each grid. The detailed steps can be seen in the last section or in the codes. Hence, the final feature vector for the image has 512 bits. Figure 4 and Figure 5 are the results of gradient maps. Next, 12 structural rules are applied to each gradient map. Let's take the 7th grid for an example. Figure 6 is the result of applying 12 structural rules. We can see that each rule extracts different structural pattern as shown in the Table 1. After that, I get 16 8-bit concavity feature vectors as well.

3.3 Evaluation

I used two different evaluation strategies for identification model and verification model. For identification model, I used **KNN classifier** and **10-fold cross validation**. For verification model, I used the following equation as the method to do **similarity measurement**.

$$S(X,Y) = \frac{s_{11}s_{00} - s_{10}s_{01}}{((s_{10}+s_{11})(s_{01}+s_{00})(s_{11}+s_{01})(s_{00}+s_{10}))^{1/2}}$$

4. EXPERIMENTS AND RESULTS

4.1 Identification model

I used SigComp2011 dataset. These data are made publicly available for the ICDAR'11 signature verification competition. It has Dutch dataset and Chinese dataset. Each dataset has 10 reference writers and each writer has 24 real signatures and 4 skilled forgeries.

4.1.1 Experiment

In the experiment of identification model, I only used 240*2 real signatures. 90% of data is used as training data, 10% of data is used as test data and then did 10-fold cross validation based on KNN (n=1) classifier.

4.1.2 Result

The result is shown as the following table.

Table 2. the result of identification system

	AUC
Dutch dataset	99.17%
Chinese dataset	97.5%

4.2 Verification model

4.2.1 Experiment

In the experiment of verification mode, I used the same dataset. However, besides 240*2 real signature, I also used 40*2 forgeries to calculate both winthin and between writers distances. Normally, to judge if the questioned signature is real or forged, we select another real signature as reference and calculate the similarity distance between them.

4.2.2 Result

The statistics of within and between writers are shown as Table 3. Therefore, if the distance between two signatures drops in the range of same-writer distance, the questioned signature is real. Otherwise, it's forged. In this project, I used $mean_{same} - var_{same}^{0.5}$ as threshold.

Table 3. results of verification model

	same writer distance		different writers distance	
	mean	variance	mean	variance
Dutch dataset	0.7360	3.9627e-04	0.5783	0.0057
Chinese dataset	0.7454	6.5510e-04	0.6471	0.0017

Table 4. the result of verification model

	AUC
Dutch dataset	87.79%
Chinese dataset	87.39%

5. APPENDIX

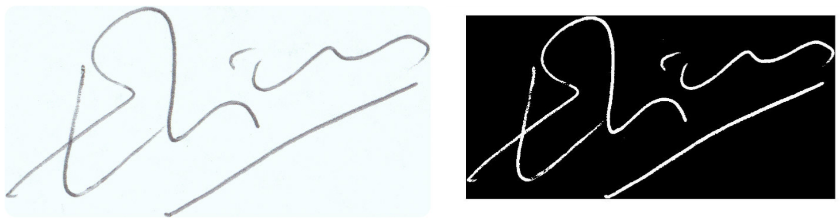
Figure 1. original image (left) and binary image (right)

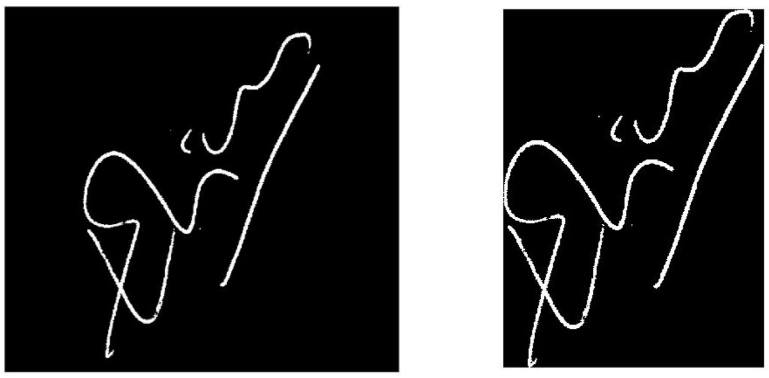
Figure 2. rotation normalization (left) and after cropping (right)

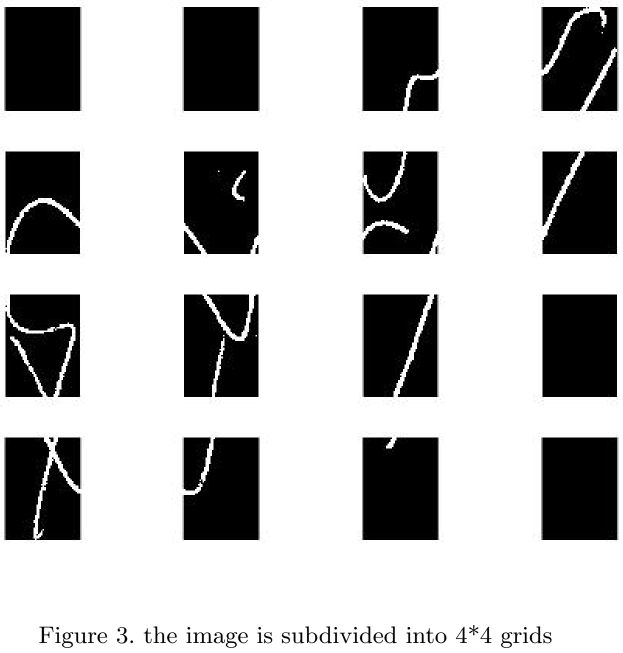
Figure 3. the image is subdivided into 4*4 grids

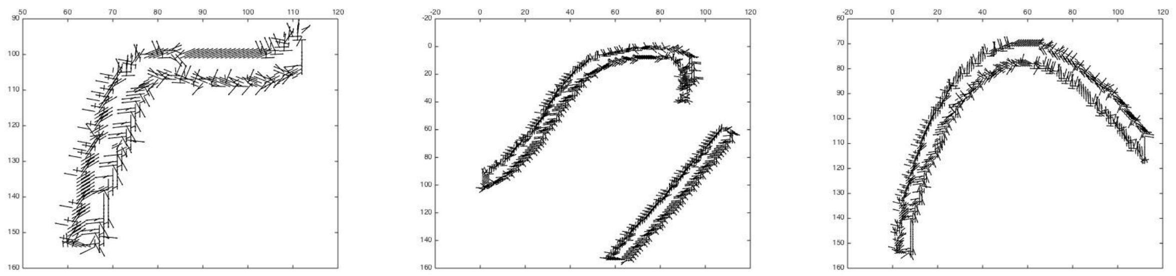

Figure 4. gradient maps of three grids (3,4,5)

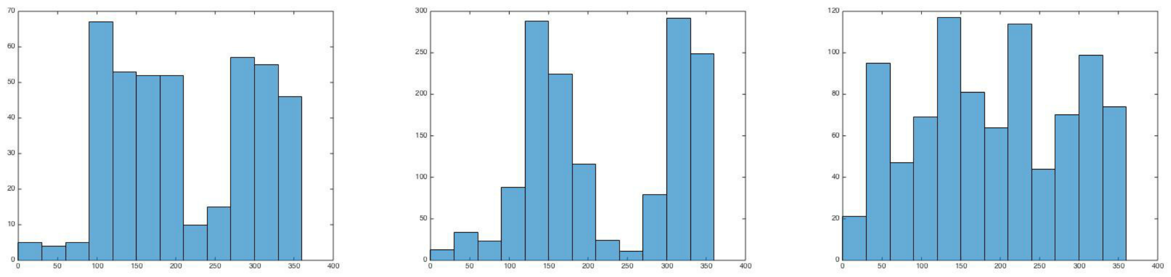

Figure 5. histograms of three gradient maps (3,4,5)

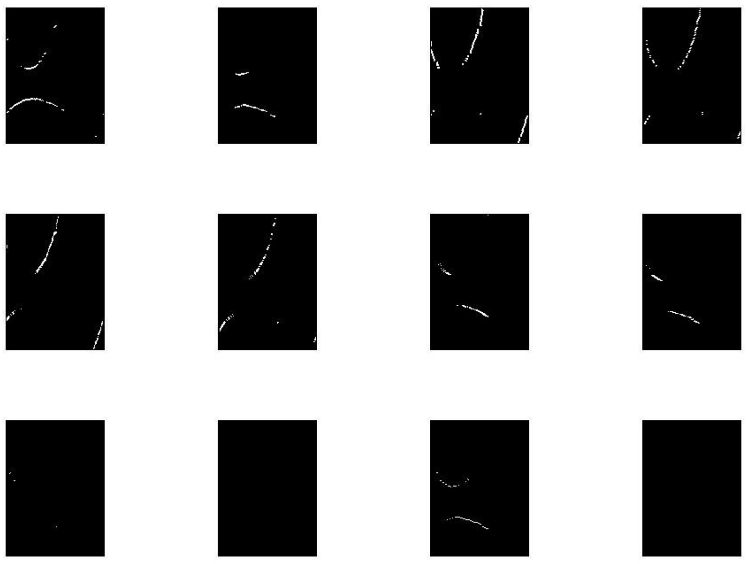

Figure 6. 12 structural rules applied to the 7th grid