Optical Character Recognition

Yafang Xue
Department of Biomedical Engineering
University of Michigan

***Abstract*—This paper describes two implementations in optical character recognition using template matching method and feature extraction method followed by support vector machine classification. With proper image preprocessing, the texts are segmented into isolated characters and the correlations between a single character and a given set of templates are computed to find the similarities and then identify the input character. In the second method, features extracted from the segmented characters are used to train the SVM classifiers, which are later, tested by a test set of handwritten digits.**

Keywords—Optical character recognition; template matching; feature extraction; support vector machine.

I. INTRODUCTION

Optical Character Recognition usually abbreviated to OCR involves a computer system designed to translate images of typewritten or handwritten text (usually captured by a scanner) into machine readable and editable text [1]. OCR could be applied to many fields like vehicle license plate recognition, information retrieval, document digitization, and in text-to-speech applications.

Over the years, OCR has attracted a great deal of researches and has developed various successful methods of recognition. In this project, I implement two commonly used methods of OCR to translate images of letters and digits into computer readable texts.

II. PREPROCESSING

A. Data

I have obtained images of 35 fonts from Microsoft Word for the typewritten characters and 120 samples of handwritten digits from 10 people (captured by a scanner). The handwritten texts contain all of the uppercase letters of the alphabet.

B. Binarization

The scanned images of texts require certain pre-processing steps so that they are in suitable forms for character recognition. Since most OCR algorithms require bi-tonal images, we must first convert color or gray images to black and white images, this is called "binarization".

Figure 1 shows the comparison between the original image and the black and white version. After the original RGB image is converted to gray scale, binarization is simply choosing a threshold value.

Fig.1 Original input image and after binarization

C. Morphological Standardization

Initially, the scanned images typically contain varying line thickness for the characters, even within individual letters. To begin with, I use bwmorph(img,'thin',inf) to thin each line in the image. As is shown in figure 2, the lines in the letters are now reduced to the same thickness (1-2 pixels).

Fig.2 Thinned image

Because of the pixels reduction after thinning, now there are some extra "little hairs" in the letters. To cut those unwanted little hairs, I use the Matlab pruning algorithm, bwmorph(img, 'spur') to prune the skeleton of each letter. As we can see in fig 3, the letters are now a lot smoother.

Fig.3 Pruning

In some cases, there would be some smudges or unwanted little dots in the image that may interfere the recognition. To avoid this, I use the function "bwareaopen" to remove all the possible small components in the image. Then, the final step is to thicken the thinned image using Matlab's bwmorph(img,

'dilate',1) so that the lines in the letters do not disappear or fall apart due to the thinning process.

Fig.4 Remove small components

Fig.5 Dilated image

D. Line Detection

This step is necessary to improve the output page layout. By detecting the lines of texts, we are able to determine the order of characters and possibly their layout on the page in later steps. This is done by a horizontal projection of the page. First I assume there is no overlapping between lines of texts, which is the case in most scanned printed images. If the sum of all the pixels within a single row is 0 (no pixels in the line breaks), this row is considered a break between the lines. After we find the break, we could easily find and crop each line matrix.

Fig.6 Divided lines of texts

E. Character Segmentation

Segmentation is the most important step of the pre-processing procedure. Most recognizing methods can only identify single characters. For instance, in feature extraction method, segmentation allows the system to extract features from the segmented letters and then classify them. Here I implement two methods of character segmentation. The first approach is to use the "regionprops" operation to cut up the image into possible pieces of interest. First, I compute the measurements of "Area", "Centroid" and "BoundingBox", if the measurements of the region meet a certain criteria (for instance, if the region contains certain pixels), this region would be extracted from the original image and form a subimage and thus each letter within the image is segmented. Below shows the segmented letters of the "EECS 451".

Fig.7 Segmented letters

Another approach is to use "bwlable" to check the connectivity of the letters and lable the connected components. The following step is to crop out each labled group of pixels by finding the group's minimum and maxium values of its row and column and extracting the letter out. Both methods can successfully segment the characters.

III. CHARACTER RECOGNITION

A. Template Matching

Template matching is a classic Optical Character Recognition technique. It is the process of finding the location of a sub image called a template inside an image. Once a number of corresponding templates are found their centers are used as corresponding points to determine the registration parameters[2]. Template matching involves comparing similarities between a given set of templates and an input image, which is normalized as the same size of the templates and then determining the certain template that produces the highest similarity.

The matching formula I implement here to detect similarities between the patterns of 2 signals is the cross-correlation method we learned in class except it is implemented in 2D instead of 1D. I use the matlab function "corr2" to compute the correlation coefficients from each comparison between the tested image and the template. In the formula below, if A_{mn} is the input image, B_{mn} is one of the templates. The matching function r will return a value indicates how well A_{mn} matches B_{mn}. If one of the correlation coefficients is the highest, the input image is identified as this letter or digit.

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \bar{A}\right)\left(B_{mn} - \bar{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \bar{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \bar{B}\right)^2\right)}}$$

Following line detection and character segmentation, the matching process starts to read the input signal from the first line of the texts to the bottom line, from left to right, which guarantees the order of each letter and their output layout of each line. Then the final step is to write the words in a text file. For an input image in Figure 1, the system could read the image into texts as shown in the figure below.

Fig.8 Output texts

B. Drawbacks of the Template Matching Method

In Template Matching method, the recognition is based on measuring the similarities between the structure of the input image and a given set of templates. Inherently, this method is sensitive to template mismatch when the input characters are not exactly the same font as the templates. In the example below, errors occur when the font of the input image (Ariel) is different from the template. The system misreads letter I,Q and R to 0,O and P, respectively. Because of the slightly change in the structure of the input characters, the highest matches of certain characters are not found in their true corresponding templates of letters or digits. The recognition accuracy of this method is highly affected by the font of the input characters.

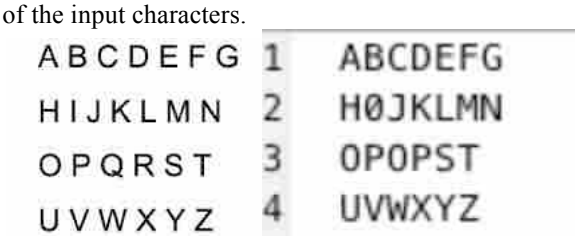

Fig.9 Template Mismatch

IV. FEATURE EXTRACTION AND SVM CLASSIIFIER

Another technique I implement is digits recognition by feature extraction and Support Vector Machines (SVM) classification.

A. Dataset

The training dataset consists of 1020 synthetic and handwritten images of digits 0-9 and the test set is composed of 120 samples of handwritten digits from 0 to 9. Before feature extraction, the images undergo similar image preprocessing steps as mentioned above: first the input image is binarized to black and white, and then single characters are segmented, and finally the image is resized to a 16 by 16 pixels scale and ready to feed in the feature extraction procedure.

B. HOG Feature Extraction

In feature extraction method, the extracted features are used to train the classifier and later identify the character. Therefore it is crucial to determine which features can best represent the characters and are optimal for classification. First let's consider the most straightforward case, the raw pixels values of a single character. The input image is a 16 by 16 grayscale image with each pixel value range from 0 to 255. The simplest feature is to use the 16×16 pixels as a feature vector to train the classifier. However, we can predict intuitively that the feature of raw pixels values is not the most representative classification feature since it cannot provide much information concerning the structure and the shape of a character. Thus the gradient histogram feature is introduced.

I experiment with features constructed through histograms of oriented gradients using the Matlab function "extractHOGFeatures". Each pixel in the image is assigned an orientation and magnitude based on the local gradient and histograms are constructed by aggregating the pixel responses within cells of various sizes [3]. I construct with cell size parameter of 2×2, 4×4, and 8×8 and visualize the result to see which cell size contains the right amount of structure and shape information of a character.

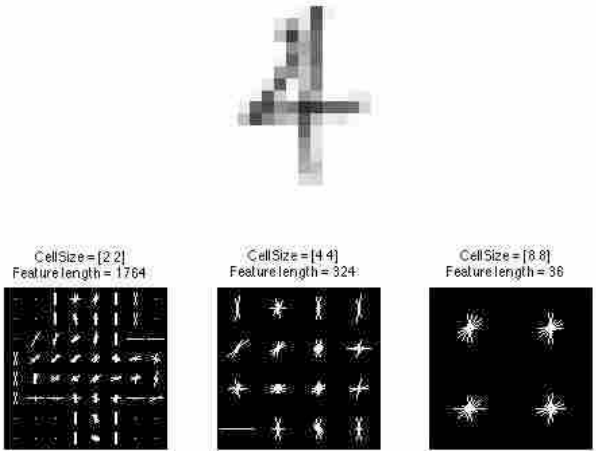

Fig.10 Comparison of different cell sizes

Figure 8 shows the visualization of 3 cell size parameters. First, cell size 8×8 does not contain much shape information. Cell size 2×2 seems to be the most representative figure among the three cell sizes. However, since HOG works by decomposing the input image into square cells of size "CellSize", computing the histogram of oriented gradient in each cell and then renormalizing the cells by looking into adjacent blocks [4]. A decrease in the cell size means a significantly increase in the dimensionality of the HOG feature vector. So for compromising purposes, I choose 4×4 as the cell size since it limits the dimensions in HOG feature vectors and also contains enough shape information to identify a character.

C. *Support Vector Machine Classifier*

After choosing the optimal cell size, we start to train the SVM classifiers using the extracted HOG features and raw pixels values features. A support vector machine is a classifier defined by a separating hyperplane. Given labeled training data, the SVM algorithm outputs an optimal hyperplane that categorizes other test data [5].The Matlab function implemented for training the SVM classifiers is "svmtrain". Since Matlab only supports 2-class SVM classifier, a commonly used multiclass SVM classification method called "one–to-all" is implemented here to classify digits 0-9. The idea of one-to-all is that we train the SVM classifier of each digit, for classifier SVM(0), the samples recognized as this class(0) is considered positive whereas samples of all the other classes (1-9) are negative. By looping through all the classifiers SVM(0-9), a digit could be classified into one of these 10 classes.

Then we test the SVM classifiers trained above. The procedure is similar: first extract HOG and raw pixels values features of the test set, and then classify the test images according to their features by the SVM classifiers using Matlab function svmclassify.

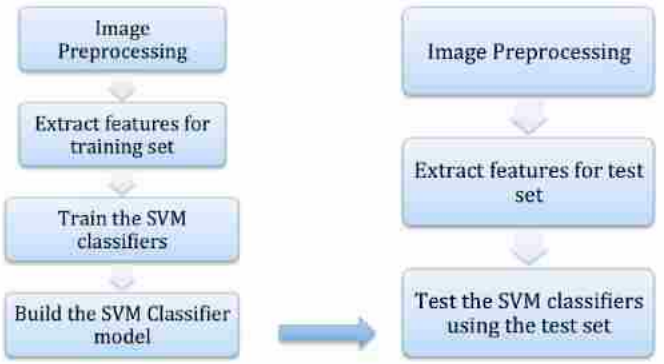

Fig.11 Digits Classification Process

D. *Results*

The rows of the tables contain the results of each SVM classifier for raw pixel values feature and HOG feature. An ideal classification system should present a diagonal line of 12 (numbers of test images of each digit) in the matrix, with 0 elsewhere. As we can see in the tables below there are false positives in both cases. As predicted, the SVM classifiers trained by the feature of raw pixels values do not perform satisfactorily. They only have a relatively high classification accuracy on digit "0", "1", and "4".

Digits	0	1	2	3	4	5	6	7	8	9
SVM(0)	7	0	0	0	1	1	1	2	0	3
SVM(1)	0	9	4	1	0	0	1	1	0	0
SVM(2)	0	0	3	1	0	1	0	0	0	0
SVM(3)	0	0	0	1	0	2	0	0	0	1
SVM(4)	5	1	1	1	7	0	5	2	0	0
SVM(5)	0	0	0	1	0	0	0	0	0	0
SVM(6)	0	0	0	2	1	1	1	0	0	0
SVM(7)	0	0	0	0	1	0	1	3	2	1
SVM(8)	0	0	0	0	0	0	0	0	1	0
SVM(9)	0	0	0	1	1	1	0	0	0	3

Table 1. Raw Pixels Values

The SVM classifiers trained by HOG features perform much better than the raw pixels values classifiers. Digit "1", "2" and "4" have the highest classification accuracy. The most difficult digits to read are "9" and "5", and SVM(6) has the highest false positive rate.

Digits	0	1	2	3	4	5	6	7	8	9
SVM(0)	6	3	0	0	0	0	0	0	0	0
SVM(1)	0	10	2	0	0	0	0	0	0	1
SVM(2)	0	0	8	0	0	0	0	0	0	0
SVM(3)	0	0	0	7	0	0	0	1	1	1
SVM(4)	0	0	0	0	9	0	2	0	0	1
SVM(5)	0	0	0	0	0	4	0	0	0	1
SVM(6)	6	0	1	4	0	7	6	0	0	0
SVM(7)	0	2	1	0	0	0	0	5	1	0
SVM(8)	2	0	0	0	0	1	3	0	5	0
SVM(9)	0	0	0	0	1	0	0	1	2	2

Table 2. HOG features

V. Discussion

In this project, I implement two methods of optical character recognition. The template matching method is one of the classic methods in OCR. The theory is easy to understand and implement. But it has an obvious drawback of template mismatch. The recognition accuracy of this method is highly dependent on the similarities between the input image and the templates.

In the second implement, the classifiers created by feature extraction seem to have a relatively low accuracy in recognition with the highest accuracy being 83.3% (SVM(1) of HOG features). The main reason of this might be the training set we select here is not big and representative enough and there is a rather big gap in the similarities between the training set and the test set. A larger dataset might improve the performance of the classifiers. Although with the relatively poor classification accuracy of the two sets of classifiers, the advantage of the HOG features extraction over raw pixels values is still notable.

VI. Other In-class DSP tools

I'm really interested in the convolution matrixes aka kernels we learned in class so I have tried some kernels on my image of letters.

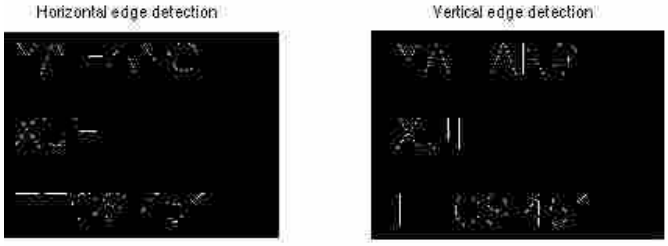

Fig.12 Horizontal and Vertical edge detection

These two filters can perform local differencing operations by convolving with matrixes [1,1; -1,-1] and [1,-1; 1,-1]. When there is a large jump in neighboring pixel values, the output will be large. The identification of horizontal and vertical edges is clearly shown through the letter E in the above image. Another matrix I implement is [-1,-1,-1;-1,8,-1;-1,-1,-1]. Also by detecting the large jump in neighboring pixel values, the output gives us the outlines of the letters.

Fig.13 Edge detection

VII. Conclusion

Optical Character Recognition is an interesting project to implement and learn. There are various approaches of identifying a character, but they always start with an image preprocessing procedure. Proper preprocessing steps like binarization, morphological standardization, and segmentation are crucial for future steps of recognition. Template matching is one of the oldest approaches in OCR field. The simple, yet powerful theory is to compare the input image with a stored template and identify the character according to its highest match in similarities. The obvious drawback of this method is that the recognition accuracy depends highly on the similarities between the input image and the stored templates. Another approach is by feature extraction with the following SVM classification. The histogram of oriented gradient feature is more representative and valuable than the raw pixels values feature in training an OCR system. Though the accuracy of this approach is not ideal due to the small dataset and the disparity between the training set and test set, we could still see the feasibility of this method. If there are future opportunities in implementing OCR, I would like to try more classifiers and explore more on the machine learning aspect like neural network.

References

[1] Kumar, R., & Singh, A. (2010). Detection and segmentation of lines and words in Gurmukhi handwritten text. doi:10.1109/IADCC.2010.5422927

[2] Patel, Krunal M. and Amrut N. Patel "Approaches for Multi-Font/Size Character Recognition:A Review ." Quest International Multidisciplinary Research Journal 1.2 (2012) : 189 – 193.

[3] Maji, Subhransu, Author Malik, Jitendra, Author. (n.d.). *Fast and Accurate Digit Classification*. EECS Department, University of California.

[4] VLFeat - Documentation > C API. (n.d.). Retrieved from http://www.vlfeat.org/api/hog.html

[5] Introduction to Support Vector Machines — OpenCV 2.4.9.0 documentation. (n.d.). Retrieved from http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html