Chinese Character Recognition

J.F. van Wezel

University of Groningen

July 13, 2017

1 Introduction

Character recognition is an important part in the fields of computer vision and artificial intelligence. For a machine to recognize sentences, words ,and characters in a given image is useful for multiple goals. For example to digitize handwritten texts. digitized texts have a number of advantages over physical texts. An important advantage is that digital texts are more easily distributed and reproduced. It is also easier to search in digital texts. But probably the greatest advantage is storage. It is easier and takes less room to save and backup a few bytes on a disk than it is to physically backup archive the same texts. As is often the case in the fields computer vision and artificial intelligence, compression is the end goal.

An other practical application of character recognition is text translation. Text recognition for the use of translation is useful for example tourists who use their smart-phone to take a picture of a sign and have that sign translated in their native language by an translation app. Or even better, have their text translated and inserted into the camera feed directly. This form of augmented reality is still in its infantsy but being able to recognize and digitize texts on walls, signs ,and posters for all kinds of purposes is not that hard to imagine.

Usage of character recognition can also be imagined in the retail industry. Here it can be used for label recognition on certain products, currency ,and identification purposes for restricted product groups (like liquor and tobacco).

The automotive industry uses character recognition in their autonomous cars. They read speed limits and other road signs to form a state of awareness of their surroundings They abstract what the next course of action should be based partly on that gathered state. A normal road sign is not a character but a computer does not know the difference between 'human' characters and a specific image. The important part is that the goal is to find meaning in an image by 'reading' parts of that image. Sign recognition and character recognition are therefor closely related. They differ in that characters tend to come in a sequence forming sentences. Where road signs are mostly isolated.

Staying on the road character recognition is used for numberplate recognition. It is used by law enforcers to find stolen or unregister vehicles but also as an automated form of speed control. Here a mounted or portable camera is used to take pictures of cars with their number plates and the software recognizes the plates and the characters on them to identify the vehicle.

bruggetje?

A character recognition system can be roughly divided into three main components: segmentation, feature extraction, and classification.

Segmentation is where an algorithm tries to find the location of the character or sentences in a given image. Before it can start this task the image often is preprocessed. During preprocessing, an image can be binerized and filtered to reduct small noise and other unwanted large components that might be present in the image. Preprocessing can also include rotating, shearing or other morphological operations.

Feature extraction is where the features of the segmented character images are found and measured. For example if an image is black and white a feature could be the ratio between black and white pixels. An other example of a feature could be the amount of horizontal edges in the image by using an edge detector. The type of features that should be extracted depends heavily on the type of dataset. A western handwritten scroll houses different characters and features than Egyptian hieroglyphs.

Classification is where the, until then unknown, characters get assigned a label. This task can be done by a number of algorithms from the simple K-nearest neighbors to complicated multilayer convolutional neural networks. Which algorithm yields the best results depends on the dataset, the segmentation, and the feature extraction.

This paper will focus on the building of pragmatic preprocessing and segmentation system for a Chinese character dataset. There has been done work in the past on Chinese character segmentation by others. From this work we know there are multiple ways of segmenting the Chinese characters. Examples of methods used in the past are: vertical and horizontal projection based, recognition based, skeleton based, and connected component based.

For the created system a combination of various prepossessing techniques, horizontal projection, and connected components was used to segment the dataset. This results in a pragmatic segmentation system that was designed for the given dataset but should also be useful for other Chinese character datasets. The segmented images are used for feature extraction in his paper by Lennart and classification of the data by Leon .

This work will use labeled data from the given dataset to validate the resulting segmented images and compare these results with results from other methods . The use of already used methods that were shown to have worked in related work will probably result in a reliable segmentation system.

Lennart achter-naam en paper

achternaam en paper

what other methods

ref

This paper will continue by explaining a bit more about the dataset. Then the used methods and used parameters will be further explained. After, the type of experiment and how the segmented images were validated will be explained in the Experiment section. From the experiment the results will be shown and discussed. This paper will end with a conclusion on the work done.

2 Method

TODO: Small introduction to this section

TODO: the skew of the chars is not done by this algorithm

2.1 Segmentation

As discussed in the introduction (1) the Chinese characters were supplied in greyscale strokes of characters in the western orientation (from left to right).

In order to segment the characters from an image a combination of methods were used. First the image is binarized. Binarization is done by applying a Gaussian filter on the image to remove some of the pixel noise. Then Otsu's [1] method of finding a threshold is used. The image is then binarized using this threshold.

TODO: Add references for Otsu and gausian filtering

After binarization the image is rotated. This is done because most images in the dataset are titled slightly. This is probably due to some variations in the orientation of the paper during the scanning process. Rotation is done by resizing the image in the vertical direction to half its size. The idea behind this is that the characters will be squeezed together and form a horizontal line this line can then be used to find the optimal rotation. this is done by taking the horizontal densities. The maximum densities are saved for the different rotations (between $+1°$ and $-1°$). The rotation with the largest maximum density is where the line is likely to be the most horizontal. This is taken as the optimal rotation.

After the characters are rotated the vertical location of the characters are located by taking the horizontal densities. The densities are smoothed by using a averaging filter with ten of the neighboring densities. Then the densities are thresholded. The threshold is determined by taking $0.1 \times imageWidth$. From the resulting vertical areas the largest area is taken. This is based on the assumption that characters will produce the most wide peak in horizontal densities. For example many of the images hold a horizontal line above or beneath the characters. This line will produce a narrow but high peak when the horizontal densities are taken. The characters will in this case produce a lower and wider peak. After the determination of the vertical location of the characters the rest of the image is removed leaving only a horizontal stroke of characters.

TODO: Add image of the densities

We are left with a isolated stroke of characters. In order to find the characters in this stroke the connected components are taken. This will give us a list of all the connected components in the image. If multiple components are above each other we assume that these components are part of the same character. These components are merged. Some components are not directly on top of each-other. It is determent how much the components overlap vertically. If the overlapping width is 0.4 times the width of the width of the smaller of the two components, they are merged.

After merging vertically we are left with a list of components that might be outliers in terms of width in comparison with the mean width of the characters in the supplied labeled dataset. In order to recognize outliers the mean and standard deviation was calculated from the characters in the labeled dataset. A component is deemed a small out-lier if:

$$C_w < \mu_w + 2\sigma_w \tag{1}$$

and big if:

$$C_w > \mu_w + 2\sigma_w \tag{2}$$

Here C_w is the component's width, μ_w is the mean width ,and σ_w is the standard deviation in width.

Small components that are characters on their own are often preceded and followed by more white space than components that form a character with the component left or right from it. The small characters are inspected on this feature and it is determined if this small component is actually a character. Some small components are just noise. If a component is too small, has too much black pixels or too much white pixels, it is recognized as noise and it is removed from the list of components.

TODO: At the moment these determinations are made by chosen thresholds. But these thresholds can be extracted from the labeled dataset

All that is left in the list of small components are expected to be components of characters. The components are merged with their left or right neighbor based on the width the newly merged component will have. Small components merge with the component (left or right) that together will have the smallest width. If however both neighboring components are deemed as big by the outliers detection, no merging will take place. The component is measured after merging. If the merged components is still small, the algorithm will try to merge it again with its left or right neighbor.

TODO: Sometimes 2 characters are seen as one because of some noise above the characters or because they are both small. This can be detected by taking the components that are deemed big and see if they can be separated in two or more components if there is a lot of white space in between them or by looking at the vertical densities.

2.2 Feature Extraction

2.3 Classification

3 Dataset

This section will shed light on the used Chinese character dataset. It will show some metrics, examples of the noise encountered, and other problems that needed to be solved to achieve the segmented images.

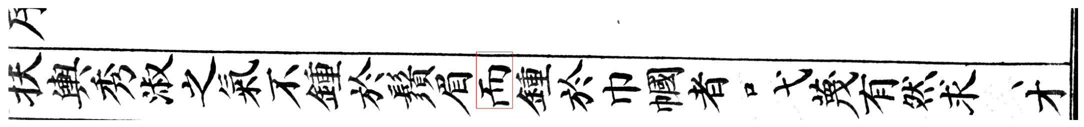


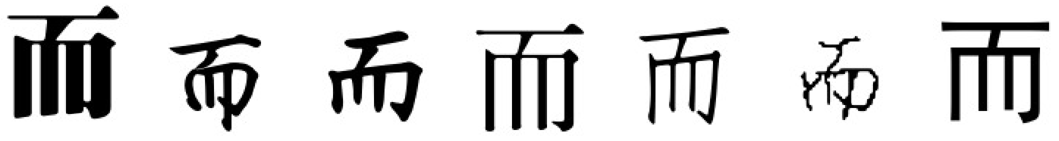
Figure 1: Example of a typical image in the dataset

Table 1: Dataset metrics

Size	Number of labels	Number of unique labels	Number of unique characters	Number of fonts
6000	27026	589	6500	7

The metrics of the used dataset for segmentation

Table 1 shows some metrics from the dataset. The size (6000) is smaller than the number of labels (27026). The size is based on the number of images the dataset holds. The images in the dataset consist of unsegmented lines of Chinese characters. An example of a typical image is shown in figure 1. The red box in the image shows the location of an a labeled character. The locations of these labeled characters are given in xml files along with the line images.



(a) ChaoMing (b) CuoKai (c) HeiTi (d) MingTi (e) TeMing (f) XinSong (g) YanKai

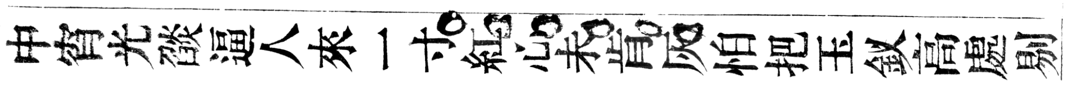
Figure 2: The Chinese printed fonts present in the dataset.

The fonts used in the dataset are shown in figure 2. There are seven fonts in total, which show a similarity with the characters in the images in the dataset. It seems assumable from this observation that the characters were printed instead of handwritten.

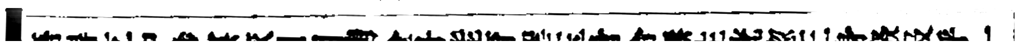
The images in the dataset are rows of characters but as shown in figure 1 contain sometimes an extra row with one or more characters in it. Sometimes there is whitespace on one or all sides of the image and sometimes the whole image is noise. The character rows are more or less given but the exact location of the the character row needs still to be found in the image.

The Chinese language is generally written from top to bottom. The orientation of the images is however in the western orientation of the language, from left to right. Because the whole dataset was in the western orientation the dataset will be handled as such. The horizontal line present in the image is to separate the character rows. This line is not present in all images. The image also shows that the rotation of the image is not straight, it is rotated by a few degrees. The characters them self however do not seem to show a curve. This is also not true for all images, some images do show a slight curvature. The rotation is probably a result of the scanning process. This is mostly done by hand and prone to error. The line could be abused to find the correct rotation of the image. But as stated the line is not present in all images and a different approach would be needed for the images without the line.

Figure 1 also shows a 'fat' vertical line on the far right side of the image. This line is also present in most images but not all. Sometimes this line shows up on the far left side of the image or on both sides. This line is not a character and needs to be handled as noise. but it can also be used to find the starting and end point of a character row.



(a) Partial noise



(b) Whole image noise

Figure 3: Examples of noise found in the dataset

The image shown in figure 3a illustrates noise on the center characters often found in this dataset. This noise might be the result of water damage or perforation of the sheet where the characters were originally on. This noise could be resolved with morphological operations and using a dataset specific filter. The dataset also contains some images that are by them self noise. Figure 3b shows an image that is complete noise. This specific image seems to be located to low during the scanning process. This resulted in only half of the original characters showing in the dataset.

4 Experiment

5 Results

6 Discussion

7 Conclusion

References

[1] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.