一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
BioX2015-50,PRMU2015-173 (2016-03)

Text-Line and Character Segmentation for Off-line Recognition of Handwritten Japanese Text

Kha Cong Nguyen
Department of Computer and Information Sciences
Tokyo University of Agriculture and Technology
Tokyo, Japan
Email: congkhanguyen@gmail.com

Nakagawa Masaki
Department of Computer and Information Sciences
Tokyo University of Agriculture and Technology
Tokyo, Japan
Email: nakagawa@cc.tuat.ac.jp

***Abstract*—Text-line and character segmentation is one of the most important steps to develop an Optical Character Recognition (OCR) system for handwritten text. Due to distortions of handwriting, i.e., variations of character size, irregular gaps between characters and touching of characters, etc., however, the step is still a difficult task. This paper proposes a method for segmenting Japanese text-lines and characters from off-line handwritten text pages. The proposed method includes: text-line segmentation using a morphological method, zone projection and character separation for each segmented text-line by vertical projection, Stroke Width Transform (SWT), bridge finding and Voronoi diagrams. The method segments curved, touching and skew text-lines and complicated touching characters.**

***Index Terms*—Text-line; touching character; segmentation; projection; Stroke Width Transform; bridge finding; Voronoi.**

I. Introduction

Text-line and character segmentation is still the big challenge in OCR development for handwritten documents because of various distortions of document layout, irregular spacing between lines and characters and so on. For Japanese OCR, the step is more difficult, resulting from various categories of characters in documents such as numerals, punctuation marks, Kana (phonetic characters) and Kanji (Chinese characters) as shown in Figure 1. The diversity makes the general separation method for all characters extremely difficult and requires heavy consideration on each type of characters.

The character segmentation is the first step to create an OCR system. It begins with text-line segmentation, and subsequently separates characters in each segmented text-line. OCR recognizes separated characters and the recognition of whole text is made by the best-path search such as the Viterbi algorithm which considers segmentation likelihood, character recognition and linguistic context.

For text-line segmentation, current methods are classified into five groups: methods making use of projection profiles, those based on the Hough transform, smearing methods, component grouping methods and other methods [1]. According to the ICDAR segmentation contest 2013 [2], the most accurate method for text-line segmentation is the INMC method which is a component grouping method, based on an algorithm of energy minimization using fitting errors and distances between text-lines. The method yields the segmentation result of more than 98% for text pages in the ICDAR database, but it is complex and it takes time for the learning process to estimate a cost function that imposes the constrains on the distances between text-lines and the curvilinearity of each text-line. For our patterns as shown in Figure 2, the combination of the method using projection profiles and the smearing method seems adequate and takes less time. Therefore, we smear text pages by the closing morphology and extract text-lines from zone projection profiles.

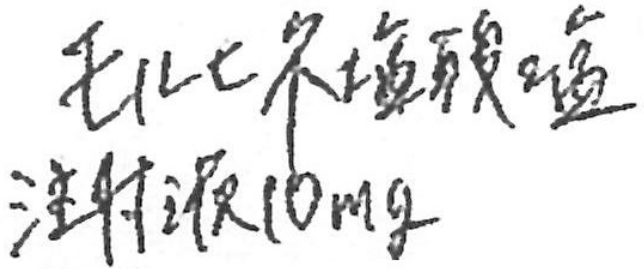


Fig. 1: Japanese handwritten text including Kana, Kanji, numerals and alphabet characters.

For character extraction from each segmented text-line, until now, there are many methods proposed. Zhao et al. apply two stages of coarse segmentation and fine segmentation. The first stage employs background skeleton and vertical projection while the second stage follows fuzzy decision rules [3]. Xu et al. proposes a learning based filter for segmenting single-touching Chinese handwriting [4]. Nonetheless, both of them are very complicated to implement. In order to segment characters, we propose a method, including two stages as [3]. Coarse segmentation uses vertical projection and Stroke Width Transform (SWT) [5]. SWT is proposed for detecting text in natural scene images, but it is used for coarse segmentation with significant effect. Fine segmentation employs bridge finding and Voronoi diagrams. We take the simplest approach, but our proposed method provides the good result not only for single touching characters but also for complicated touching characters.

The remaining of the paper is organized as follows: section II describes a text-line segmentation method, section III presents a character segmentation method, Section IV is experiment and discussion and section V draws conclusion.

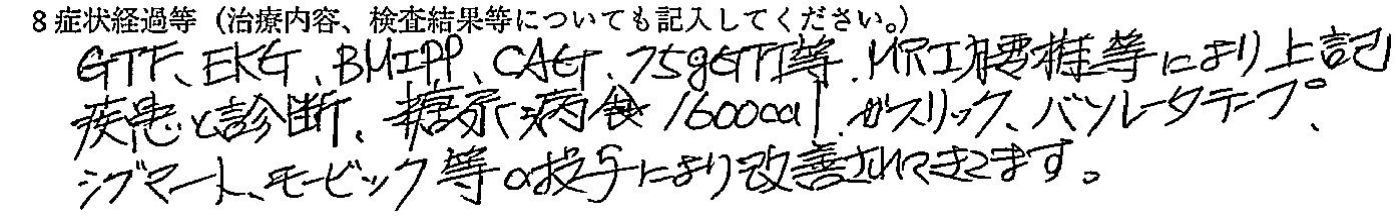


Fig. 2: Curved, skew and touching text-lines.

II. TEXT-LINE SEGMENTATION

Humans tend to write in curved and skew lines, especially when we write on sheets without lines. Text-lines are also touching in some components as shown in Figure 2. As the result, the text-line segmentation based on global projection profiles is unsuccessful.

In order to separate text-lines, we first binarize a scanned document by the simple image statistics (SIS) method which uses the weight for each considered pixel. It is faster than the most popular binarization method by Otsu [6]. Secondly, we apply the thinning algorithm [7] to mitigate touching between text-lines. In order to highlight the difference between text-line regions and space regions, we employ the closing morphology, which is a combination of the erosion morphology and the dilation morphology:

$$Close(I(x,y), elem) = dilate(erode(I(x,y), elem)) \quad (1)$$

where $elem$ is a matrix that has the number of columns larger than the number of rows. In our system, they are chosen 17 and 3, respectively. The result is shown in Figure 3 in which text-line regions are enhanced while touching regions are almost unaffected. Subsequently, we divide the text page into n stripes with the width of stripes depending on the density of texts. The width w_i of stripe s_i is computed as follows:

$$w_1 = w_i = ...w_n = \begin{cases} 0.05 * W \ if \ \frac{\sum_{W,H} I(x,y)}{W*H} \geq 0.2 \\ 0.1 * W \ if \ 0.2 > \frac{\sum_{W,H} I(x,y)}{W*H} > 0.1 \\ 0.25 * W \ if \ \frac{\sum_{W,H} I(x,y)}{W*H} \leq 0.1 \end{cases} \quad (2)$$

where W, H is the width and the heigh of the document image $I, I(x,y)$ is the value of a foreground pixel (x,y). According to the density, the number of stripes varies from 4 to 20 with the same width for each s_i.

The horizontal projection profile $yProj_i$ is computed for each stripe and it is smoothed by the simple moving average filter with the window of length 7:

$$yProj_i[y] = \frac{1}{7} \sum_{y=y-3}^{y=y+3} yProj_i[y] \quad (3)$$

The result after smoothing the projection profile for each stripe is shown in Figure 4 where spurious peaks and valleys are removed. This makes the separating decision by a threshold more effective. The threshold T_i for each stripe s_i is calculated:

$$T_i = \frac{1}{\alpha * H} \sum_{0 \leq y \ < H} yProj_i[y] \quad (4)$$

where α is a constant, in our system we set up $\alpha = 10$.

The above and below borders (valleys) in each stripe are decided:

$$AB_j = \ y \ - \ 2 \ \ If \ (yProj_i[y] > T_i \cap \ yProj_i[y-1] \leq T_i) \quad (5)$$

$$BB_j \ = \ \ y+2 \ If \ \ (yProj_i[y] > T_i \cap \ \ yProj_i[y+1] \leq T_i) \quad (6)$$

Borders will be combined together if the distance between them is less than the average distance D:

$$D = \frac{1}{N} \sum_{1 \leq j \leq N} |AB_j - BB_j| \quad (7)$$

where N is the number of pairs (above border and below border). Finally, we connect AB and BB in a stripe i to the nearest AB and BB in the stripe $i+1$ and obtain text-lines as shown in Figure 5 where AB and BB are displayed by red and green lines, respectively. Some borders are not detected, so we interpolate new borders from borders of the previous and next stripe.

III. CHARACTER SEGMENTATION

After getting segmented text-lines, we continue to extract characters. In this section, we describe the process to separate characters in each text-line.

First, we use the SWT algorithm [5] to calculate the average width of all strokes in text pages. The original purpose of SWT was to detect texts in natural scenes with many fonts and languages. We employ SWT for coarse segmentation. SWT converts an image to an array containing likely stroke width for each pixel by using the Canny edge detector. Therefore, the average stroke width AW is computed:

$$AW = \frac{\sum_{W,H} I^{'}(x,y)}{\sum_{W,H} I(x,y)} \quad (8)$$

where $I^{'}(x,y)$ is the value at a pixel (x,y) after applying SWT for the original binary image $I(x,y)$.

Secondly, we apply the closing morphology as described in section II with the element matrix that has the row number larger than the column number (a 3x15 matrix) for each

Fig. 3: Text page after applying closing morphology.

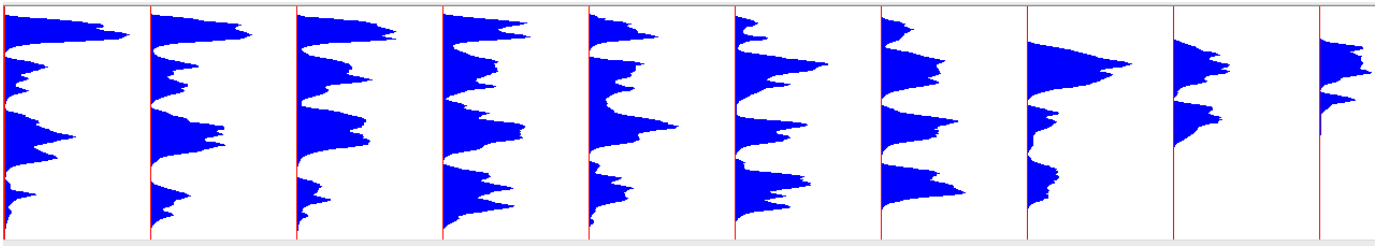


Fig. 4: Stripe projection profile.

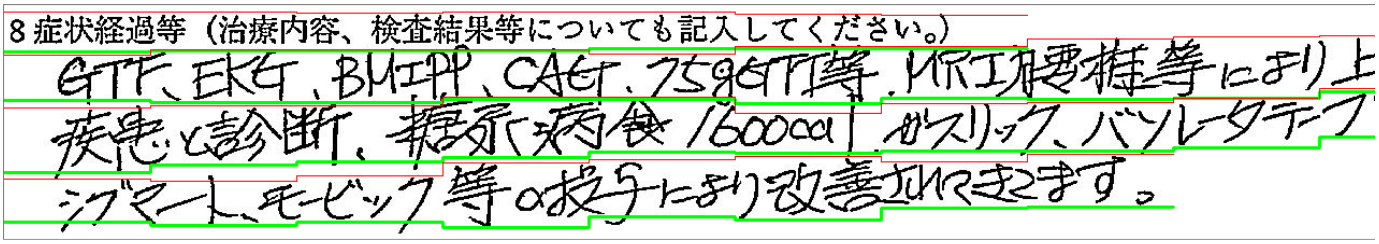
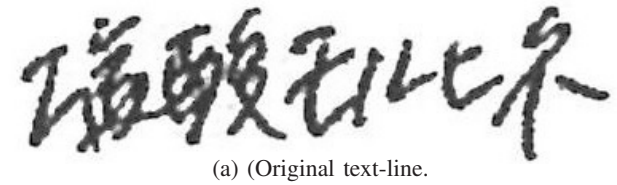


Fig. 5: Result of text-line segmentation.

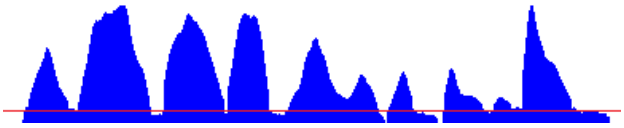
text-line. Thirdly, we compute a vertical projection profile $xProj_j$ for each text-line j and separate a text-line at points with the projection value of zero. Then we calculate the average width of split segments in the text-line. Fourthly, we further divide oversized segments at points where projection values are less than AW. To avoid mis-segmentation, we just consider projection values in the range between peaks of the projection profile within an oversized segment and we do not separate oversized segments if the separation creates very small components.

This coarse segmentation separates non-touching characters and single-touching characters effectively. Without SWT, we do not know exactly the common stroke width of characters to set up the AW threshold because the common stroke width is dependent on each text page. The coarse segmentation by the projection profile with AW is shown in Figure 6. In the example, zero point segmentation is for segments 4 and 5 firstly. Because the segment (1,2,3) is oversized, it is split into three smaller segments 1, 2 and 3 with the AW threshold. Segment 3 is not separated into two small segments because the segmentation creates a very small component.

With oversized segments unseparated by the threshold AW (complicated touching cases), we proceed to separate them finely by the fine segmentation as described below and shown in Figure 7:



(a) (Original text-line.



(b) Vertical projection and AW threshold.

(c) Character segmentation by threshold AW.

Fig. 6: Result of coarse character segmentation.

1) From the vertical projection profile of each oversized segment, we find local maximal peaks (X-axis) in the projection profile. Then we find central points (Y-axis) of connected components (CCs) on the vertical line at

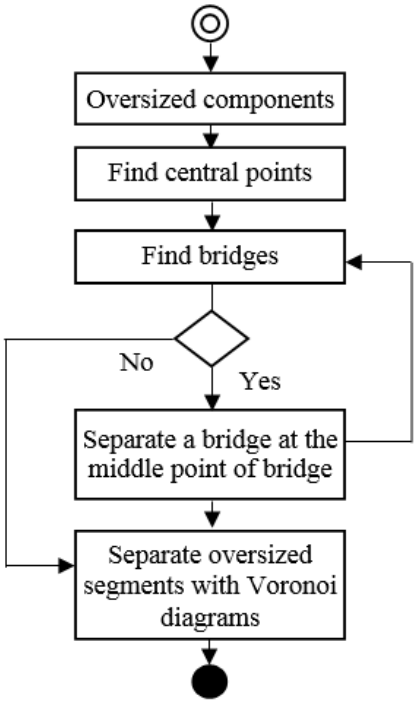


Fig. 7: Flow of fine segmentation.

each peak position. These points represent parts of CCs that need to separate.

2) Between two parts of CCs, we find a bridge, which is the shortest path between two central points of them. The algorithm to find the shortest path is A*, the heuristic searching algorithm, which achieves better time performance than traditional algorithms like Dijkstra.

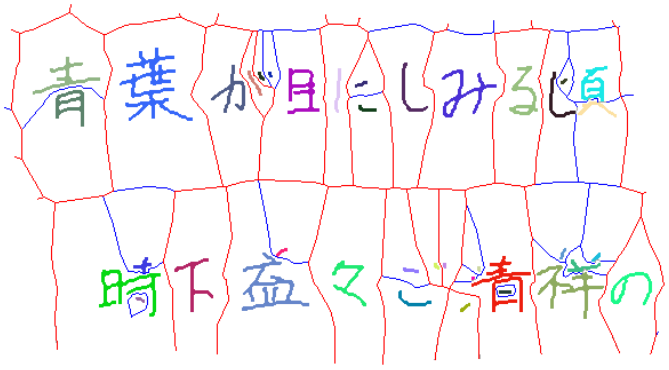
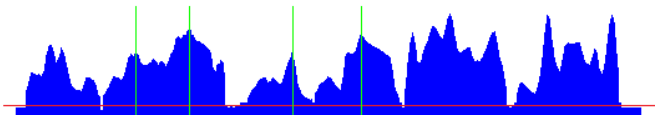


Fig. 8: Voronoi diagram.

3) If there is a bridge, we remove the middle point of the bridge and go back to the step 2 to find other bridges. This step splits CCs into smaller CCs.
4) If there is no existing bridge, the average size of CCs after removing bridges in the text line is estimated. For example, in Figure 6(b), there are five split segments, but there are seven CCs. Any segment with the size more than the average size is separated by Voronoi diagrams [8]. The Voronoi diagram shows natural borders between CCs, called Voronoi edges as show in Figure 8. A segment curve between two characters is constituted by several Voronoi edges. We perform some improvements such as removing Voronoi edges between a small component and a very big component, choosing edges extending sharply to top or bottom of text lines rather than those extending horizontally when finding segment curves, and discarding segment curves which start and end at the same bottom or top side of text lines.

(a) Original text-line.



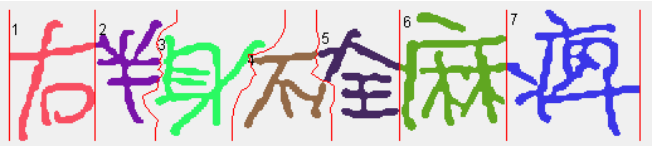
(b) Vertical projection, AW threshold and vertical lines at peak positions.

(c) First round to find bridge.

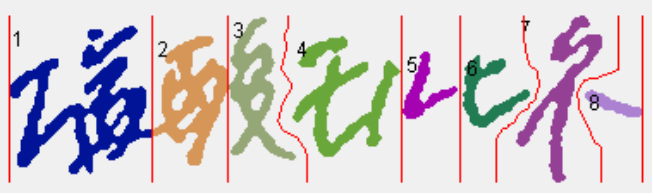
(d) Second round to find bridge.

Fig. 9: Finding and separating bridges between two parts of connected components.

Figure 9 shows an example to find and separate bridges between touching characters within an oversized segment. Figure 9(a) shows an original text-line and Figure 9(b) presents a vertical projection profile with the AW threshold (red line) and vertical lines (green lines) at peak positions. The oversized segment 2 in Figure 9(c) is composed of two CCs. From the average size of characters and peaks of the projection profile, we can assume each CC is composed of two touching characters. For each CC, two central points are selected on vertical lines at peak positions (black points). We have to examine several central points on each vertical line to find bridges. If we only select a central point at the middle or the bottom of the vertical line, there is no bridge for the right part of the second CC in the segment 2. Therefore, we have to examine several points to make sure that there is no undetected bridge. After finding a bridge, we separate it (blue circles), and find another until there is no remaining bridge between two touching parts of the CC as shown in Figure 9(d).



(a)



(b)

Fig. 10: Result after applying fine segmentation.

The result after applying the fine segmentation is shown in Figure 10. Because we just remove the middle point on a bridge of components, however, some strokes of one character is split into two sub-stokes as shown in Figure 10(a) (characters 4, 5). On the other hand, character 5 is not separated into upper and lower components because of our improvement for Voronoi diagrams, that is segment curves starting from and ending at the same top or bottom of a text-line are discarded. Although the application of Voronoi diagrams is to segment oversized components by Voronoi edges between untouched CCs, it also creates over-segments such as stroke 8 in Figure 10(b).

After character segmentation, we employ our latest OCR [9] to recognize isolated characters. Our OCR can achieve the recognition result of more than 97% regarding to the Kuchibue and Nakayosi database [10]. Because of over segmentation, many characters are split into small components such as characters (2,3), (7,8) in Figure 10(b). Therefore, we combine the recognition result by OCR with the linguistic context [11] to obtain the better result. By character classification with OCR, each candidate character pattern is associated with a number of candidate classes (30 candidates) with confident scores. The combination of all candidate patterns is presented by a candidate lattice diagram as shown in Figure 11. Each path of the diagram is evaluated by combining the score of the candidate character with its score in the context of a linguistic dictionary. For each character, the linguistic dictionary includes its frequency of occurrence with two previous candidates on the path (tri-gram). The Viterbi algorithm is used to find the path with the highest score. This path will become the final recognition result of the text-line.

IV. Experiment and Discussion

For text-line segmentation, we use the HIT-MW database including 853 images [14] to test our method. The detected text-line R and its ground- truth text-line G is defined MatchScore as follows:

$$MatchScore\,(G,R) = \frac{|G \cap R|}{|G \cup R|} \tag{9}$$

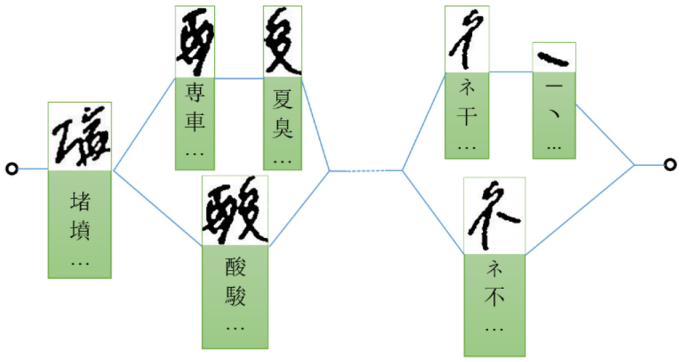


Fig. 11: Candidate lattice diagram.

This method is also used in the ICDAR contest [2] as well as the IMNC method. A line is correctly detected when:

$$MatchScore\,(G,R) > 0.95 \tag{10}$$

In addition, a detection rate (DR) is defined as:

$$DR = \frac{The\ number\ of\ detected\ lines}{The\ number\ of\ ground\_truth\ lines} \tag{11}$$

TABLE I: Text-line segmentation rate.

	DR (%)	Execution Time (min.)	# of lines
INMC method	99.52	14 - 28	8,653
Our method	99.42	10.24	8,653
Our method	98.60	0.23	144

The text-line segmentation result by our method is almost the same as the INMC method, but the execution time is less than INMC. INMC is very sensitive to short text-lines and touching text-lines, so we also prepare a small dataset of 34 handwritten Japanese text pages of 144 text-lines with many touching text-lines and short text-lines, which rarely exist in the HIT-MW database. The rate of our method does not reduce significantly.

For character segmentation, because we do not have a large database of touching characters, we test it on the above 34 text pages after text-line segmentation is performed. Within the 34 text pages, there are 112 touching character pairs.

TABLE II: Character segmentation rate.

	Correct segmentation rate (%)	# of touching pairs
Coarse segmentation	78.57	112
Fine segmentation	83.93	

There remain following problems:

1) Although we have achieved the state-of-the-art performance by a rather simple method, both of text-line segmentation and character segmentation remain to be improved. Some components of text-lines and characters are cut out or become parts of others as shown in Figure

5 and characters 4, 5 in Figure 10(a), respectively. In the future, we have to apply sophisticated algorithms like Fuzzy decision [12] or bi-variate Gaussian decision [13] to find refined points on bridges as well as segmentation points between text-lines to reduce the stroke loss.

2) There is no effective method to estimate the exact number of characters in a text-line as well as choosing peaks of touching characters. That leads to find central points wrongly.
3) Finding a bridge with central points is not always possible because characters often include some isolated components. The current solution, finding a bridge on some central points, is just a temporary solution.

V. Conclusion

In this paper, we have proposed a method to segment text-lines and characters in handwritten Japanese text pages. The text-line segmentation is based on the closing morphology and the zone projection profile. The character segmentation for each text-line uses the Stroke Width Transform and a bridge-finding algorithm. The text-line segmentation can separate curved, skew and touching text-lines that cannot do by the global projection profile. The character segmentation is effective even in case of complicated touching characters. Nonetheless, there remain some difficult problems such as finding accurate segmentation points between touching text-lines and between touching characters, which causes missing parts of text-lines and characters, the inaccurately estimated number of characters and determination of peaks to find central points and the determination of central points that we can find bridges.

Acknowledgment

This work is being partially supported by the Grant-in Aid for Scientific Research (B)-224300095 and (S)-25220401.

References

[1] Fernndez-Mota, David, Josep Llads, and Alicia Forns. "A graph-based approach for segmenting touching lines in historical handwritten documents." International Journal on Document Analysis and Recognition (IJDAR) 17.3 (2014): 293-312.

[2] Stamatopoulos, Nikolaos, et al. "Icdar 2013 handwriting segmentation contest."Document Analysis and Recognition (ICDAR), 2013 12th International Conference on. IEEE, 2013.

[3] Zhao, Shuyan, et al. "Handwritten Chinese character segmentation using a two-stage approach." Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on. IEEE, 2001.

[4] Xu, Liang, et al. "An over-segmentation method for single-touching Chinese handwriting with learning-based filtering." International Journal on Document Analysis and Recognition (IJDAR) 17.1 (2014): 91-104.

[5] Epshtein, Boris, Eyal Ofek, and Yonatan Wexler. "Detecting text in natural scenes with stroke width transform." Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010.

[6] Kittler, Josef, John Illingworth, and J. Fglein. "Threshold selection based on a simple image statistic." Computer vision, graphics, and image processing 30.2 (1985): 125-147.

[7] Zhang, T. Y., and Ching Y. Suen. "A fast parallel algorithm for thinning digital patterns." Communications of the ACM 27.3 (1984): 236-239.

[8] Van Phan, Truyen, Bilan Zhu, and Masaki Nakagawa. "Development of Nom character segmentation for collecting patterns from historical document pages."Proceedings of the 2011 Workshop on Historical Document Imaging and Processing. ACM, 2011.

[9] Van Phan, Truyen, et al. "Effects of Line Densities on Nonlinear Normalization for Online Handwritten Japanese Character Recognition." Document Analysis and Recognition (ICDAR), 2011 International Conference on. IEEE, 2011.

[10] Nakagawa, Masaki, and Kaoru Matsumoto. "Collection of on-line handwritten Japanese character pattern databases and their analyses." Document Analysis and Recognition 7.1 (2004): 69-81.

[11] Zhu, Bilan, et al. "A robust model for on-line handwritten Japanese text recognition." International Journal on Document Analysis and Recognition (IJDAR) 13.2 (2010): 121-131.

[12] Zhao, Shuyan, et al. "Handwritten Chinese character segmentation using a two-stage approach." Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on. IEEE, 2001.

[13] Arivazhagan, Manivannan, Harish Srinivasan, and Sargur Srihari. "A statistical approach to line segmentation in handwritten documents." Electronic Imaging 2007. International Society for Optics and Photonics, 2007.

[14] T. Su, T. Zhang, and D. Guan, Corpus-based HIT-MW database for ofine recognition of general-purpose Chinese handwritten text, Int. J. Doc. Anal. Recognit., vol. 10, no. 1, pp. 2738, Jun. 2007.