An Efficient Character Segmentation Algorithm for Connected Handwritten Documents

Vishal Rajput[1], N. Jayanthi[2(✉)], and S. Indu[2]

[1] Department of Electronics and Communication Engineering, PDPM-Indian Institute of Information Technology, Design and Manufacturing Jabalpur, Jabalpur 482005, MP, India
vishal.stark42@gmail.com

[2] Department of Electronics and Communication Engineering, Delhi Technological University, Delhi, India
{njayanthi,s.indu}@dce.ac.in

Abstract. This paper proposes an efficient method of character segmentation for handwritten text. The main challenge in character segmentation of hand-written text is the varied size of each letter in different documents, connected alphabets in a word in cursive writing and the presence of ligatures within an open character. Hence, this paper proposes an adaptive vertical pixel count algorithm to solve the problem of over-segmentation due to the presence of open characters such as 'w', 'v' and 'm'. Proposed algorithm works effectively against both the handwritten and standard text. The proposed method is evaluated on IAM and self-created data set.

Keywords: Optical Character Recognition · Potential segmented column

1 Introduction

Since the advent of computers, a whole lot of information has paved its way to the digital format. There are so many digital platforms to save data. Despite the availability of so many digital platforms, paper and pen are still in use. There will always be manuscripts and books which will not be available in the soft copy format. No matter the amount of the digital space, information will still be present in the hard copy format. Since old documents are very fragile and degrades with time so they need a digital storage. Such systems which can interpret handwritten or printed documents needs to build. To solve this problem, researchers use a tool called OCR (Optical Character Recognition). An OCR is a tool which extracts text from images and saves them in a machine-readable format. Today there are types of OCR available in the market but almost each one of them suffers some kind of drawback. Modern day OCR works well on the standard and non-connected text, but they are ineffective for handwritten

S. Sundaram and G. Harit (Eds.): DAR 2018, CCIS 1020, pp. 97–105, 2019.
https://doi.org/10.1007/978-981-13-9361-7_9

and connected text. The primary reason for the failure of most of the OCR is the inability to segment the characters present in a word. Because of different writing styles it is very difficult to find the correct segmentation point. Generally, computer finds it difficult to find the correct segmentation point. To solve this issue proposed algorithm should be intelligent enough to adapt to different writing styles.

2 Related Work

In the past, researchers have used different schemes for the character segmentation. Every technique suffers some or the other drawback. Different errors which happens generally during the segmentation are Bad-Segmentation, Over-Segmentation, and Miss-Segmentation. Figure 1 shows all the different types of errors. Some researchers proposed segmentation based on the average width of a character. This approach works only when the characters are of uniform width and fails when the characters present in a word are of non-uniform width. This technique is useless for normal writing sample because of the difference in each character's width.

Fig. 1. Images showing types of segmentation errors.

In this paper, an improved version of the vertical pixel count algorithm is presented. The Proposed technique was tested on IAM and self-created data set to check the efficiency of the presented algorithm. The proposed method is simple and very effective on connected text but in case of overlapping characters obtained results are not satisfactory. Further investigation will be made to handle overlapping characters by segmenting at a slope instead of vertical segmentation.

Javed et al. [2] proposed a character segmentation algorithm based on white spaces between characters. This algorithm works fine for standard text but fails for connected text. Javed's algorithm is not able to produce any result in presence of a ligature.

Rehman et al. [7] used an implicit technique to divide the image into very fine parts (much smaller than the words). And then every part passes through a neural network which identifies presence of a character in that part. It is a character identification technique whose by-product is character segmentation [6]. This approach is costly in terms of time it takes to segment the character. Its accuracy is dependent on the number of samples used to train the neural network. Another reason for the failure of their technique is that it often confuses 'w' as 'u' & 'i' or 'u' as 'i' & 'i'.

In [10], a graph model describes the possible locations for segmenting neighbouring characters. Then they use an average longest path algorithm to identify the globally optimal segmentation. This approach may fail when the lexicon domain is insufficient.

Choudhary et al. [1] have used a pixel counting approach for the character segmentation. If words are well separated, proposed method takes the average of all PSC's to solve the problem of over-segmentation. And in case of open characters like 'w', 'u' or 'm' they use the distance between two PSC's, if it is less than 7 following PSC's merges with the previous PSC. A significant problem in their proposed approach is to decide the threshold value i.e. 7 (in their case). Threshold value will change if there is large variation in characters width in a single word. Even with consistent character width same threshold may not work when same word is in large font. Choudhary's technique performs very bad in cases like 'm' & 'n'.

Yamada et al. [12] used multiple segmentation determined by contour analysis for Cursive handwritten word recognition and Saba et al. [9] considered a neural network for cursive character recognition which gives the confidence for the similarity of a given character with all the characters in the database. In [11], a probabilistic model based on Word model recognizer is used to segment the handwritten words. Lee et al. [3] used a totally different technique in which the character segmentation regions are determined by using projection profiles and topographic features extracted from the gray-scale images. A multi-stage graph search algorithm finds a nonlinear character segmentation path in each character which is the basis for the segmentation point.

3 Problem Statement

There are various challenges in the character segmentation of handwritten words due to the varied nature of individual's hand-writing. Most significant problem in character segmentation is the presence of a connected component in a given word. Cursive nature of words adds another layer of difficulty in the segmentation process. Overlapping characters also create lots of problem in segmentation as it increases the chance of a point wrongly identified as a segmentation point. Most of the characters can be written in multiple ways thus a holistic solution is required which is able to identify the correct segmentation point even after all the feature differences in the same character. Proposed method solves the problem of connected components by using the modified vertical pixel count.

4 Proposed Methodology

The proposed method uses local maxima and minima to convert vertical pixel count graph into a binary graph which is the basis for finding the segmentation points in a word. The binarized graph is processed adaptively according to the width of peaks present in it to identify the wrong segmentation points and finally potential segmentation points are marked i.e. positions where a word is segmented.

Characters which doesn't have any type of full or partial loop in them like m, n, u, w, are hard to segment. These characters are often over segmented because of multiple local minima's in vertical pixel count graph as shown in Fig. 1. To avoid the over-segmentation problem, peaks in vertical pixel count graph are combined in an adaptive manner to find the correct PSC (Potentially Segmented Column). In Fig. 2(a), three peaks of character are combined together to find the correct segmentation point. Same is the case in Fig. 2(b), where peaks are combined to solve over segmentation in character 'n'.

In the proposed method vertical pixel count of a word is used as the basis for finding the PSC's. The proposed text segmentation algorithm is summarized by Algorithm 1 and explained in detail in the further subsection.

Algorithm 1

1. Image pre-processing for noise removal and contrast enhancement.
2. Formation of vertical pixel count graph.
3. Binarizing the vertical pixel count graph.
4. Assignment of Flag value (either 0 or 1) to each peak according to their width.
5. Merging the peaks to eliminate the wrong segmentation point based on the Flag value of peaks.
6. Centre of each trough of the processed graph is used as the point of segmentation.
7. If a character is segmented at a position where PSC cuts the text more than once, ignore that PSC.

4.1 Formation of Binarized Graph

Given word image is read vertically from top to down following the same along the column (width) of the image. Vertical pixel count graph for the sample word "guys" is shown in Fig. 3(b). It is clear from the sample word graph shown in Fig. 3(b) that the vertical pixel graphs is not uniform in nature. To make further processing easy, obtained vertical pixel count graph is converted into binary

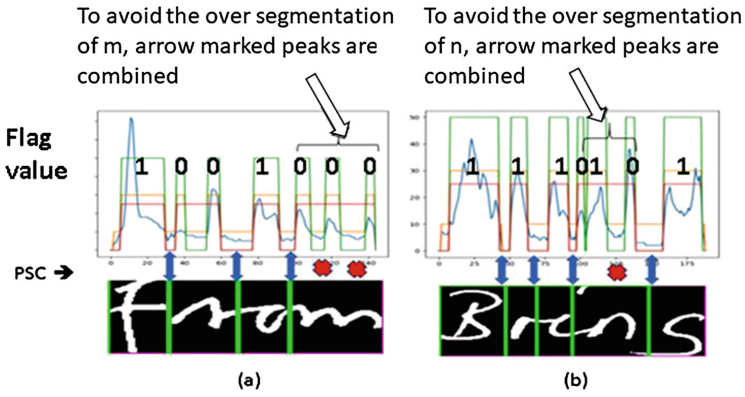

Fig. 2. How peaks are combined in vertical pixel count graph to solve the problem of over segmentation.

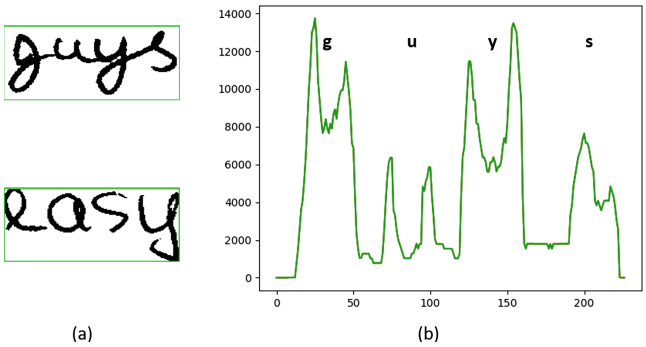

Fig. 3. (a) Sample word"guys" (self-created dataset) and "easy" (from IAM dataset) to show the sample images. (b) Pixel count of word "guys".

graph. The Obtained vertical pixel count graph is converted into a binary graph P_y based on basic thresholding operation mentioned in Eq. 1 and the binarized graph is shown in Fig. 4(a).

$$P_y = \begin{cases} 1, \text{ if } pixel\ count > 10 \\ 0, \text{otherwise} \end{cases} \tag{1}$$

4.2 Assignment of Flag Values to the Peak

To identify the over/wrongly segmented characters, peak's width in the binarized graph needs to be checked. Some peaks are very short in width as compared to the rest of the peaks. Peaks which are having width lesser than the average width indicates an over/wrongly segmented word. Now, to mark the characters which are over/wrongly segmented, the peak of that particular character in the binarized graph P_y is flagged with value '0'. For marking the peaks, comparison of each peak width in the binarized graph P_y is done with the average peak width W_{avg} to assign the flag values to be either 0 or 1 based on the Eqs. 2 and 3. Binarized flagged graph is shown in Fig. 4(a).

$$W_{avg} = \frac{\sum_{i=1}^{n} W_i}{n}, \tag{2}$$

$$F_{ip} = \begin{cases} 0, \text{ if } W_i < k \cdot W_{avg} \\ 1, \text{ otherwise,} \end{cases} \tag{3}$$

where F_{ip} is the flag used for marking of i_th peak, k is a constant whose value lies between 0.6 & 0.9 and W_i is the width of i_th peak.

The optimum value of $k = 0.7$ is obtained by experimenting multiple times. Peaks width is compared to $0.7W_{avg}$ because single character width often varies quite largely with the average width.

Identification of over/wrongly segmented characters is done after marking the peaks and based on the width of the peak. Combining the peak flagged as '0' with the adjacent peaks solves the problem of over-segmentation. This can be understood by seeing the binarized graph of word "guys" shown in Fig. 4(a), where the character 'u' is marked with two peaks instead of one peak, to solve this problem following two peaks are merged into one as shown in Fig. 4(b), solving the over-segmentation problem. All the '0' flagged peaks needs to be merged with other adjacent peaks to eliminate the case of over-segmentation. Merged peaks can be seen in Fig. 4(b) with reassigned flag values. The middle point of each trough is marked as an SC (segmentation column), i.e. point of segmentation. The process of merging of peaks is based on the following ways:

- **CASE 1:** If the successive peak/peaks are marked as flag 0 initially, merge them to form a single peak and reassign it as flag 1, removing the over-segmentation.
- **CASE 2:** If peaks adjacent to flag 0 are flag 1 then the distance from both the adjacent peaks is calculated and flag 0 peak is merged to the peak which has a shorter distance from it.

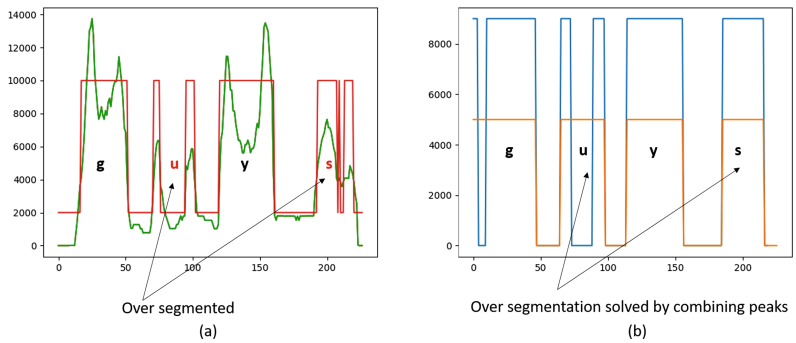

Fig. 4. (a) Pixel count converted into binary graph (b) Smaller peaks merged into one peak (Ti & Pi is width of ith trough & peak respectively).

4.3 Elimination of False PSC's

If the obtained PSC intersects the text more than once (Fig. 5(a) 'o' is over-segmented), that PSC is discarded. To find the number of intersections made by a PSC, adjacent text pixels are traced along the PSC from top to bottom. If the intersection count increases to 2 then the given PSC is discarded. Wrongly segmented characters shown in Fig. 5(a) and (c) is corrected by the elimination of false PSC as shown in Fig. 5(b) and (d) respectively.

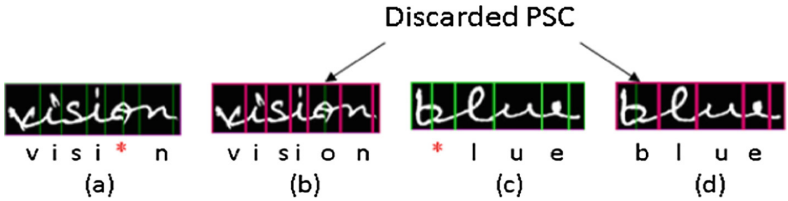

Fig. 5. (a) and (c): Wrong segmented results (b) and (d): Corrected over segmentation results.

5 Result and Discussion

In this paper self-created database consisting of 5 writers is used to test on the given algorithm along with the standard IAM dataset [5], out of which 300 non-overlapping words are chosen randomly. Segmentation results of the proposed algorithm are shown in Fig. 6. The proposed technique overcomes the problem of wrong segmentation due to the varied width of the same character as shown in Fig. 7. For the qualitative comparison of the proposed technique, the work of Amjad et al. [8] and Choudhary et al. [1] are used on the following words 'several', 'common', 'accomplish' and 'percentage' and the results are shown in Fig. 8. Method presented in [8] over segments the character like 'm' and 'n' whereas [1] miss-segments and over-segments the word like 'm', 'n', 'i' etc.

Also, it is quite difficult to compare the segmentation results with different researchers because the dataset used by everyone is different. Results achieved varies too much because some researchers assumed the absence of noise, some researchers collected the handwriting samples from a different number of writers and so on. However, for the quantitative comparison of the proposed technique, the methods presented by Salvi et al. [10] and Marti et al. [4] are considered because they have also used IAM dataset for the segmentation. The quantitative result comparison is shown in Table 1. Proposed technique fails to segment the characters if the characters are overlapping on each other, samples of wrongly segmented words with overlapping characters are shown in Fig. 9.

Table 1. Segmentation results of different methods on IAM dataset

Method	No. of words (IAM dataset)	Correctly segmented words	Wrongly segmented words	Percentage of correctly segmented words	Percentage of wrongly segmented words
Proposed method	300	255	45	85%	15%
Salvi et.al [10]	300	195	105	65%	35%
Marti et.al [4]	300	220	80	73.45%	26.55%

Fig. 6. Segmentation results from IAM and self-created dataset.

Fig. 7. Same word of different character width is segmented properly.

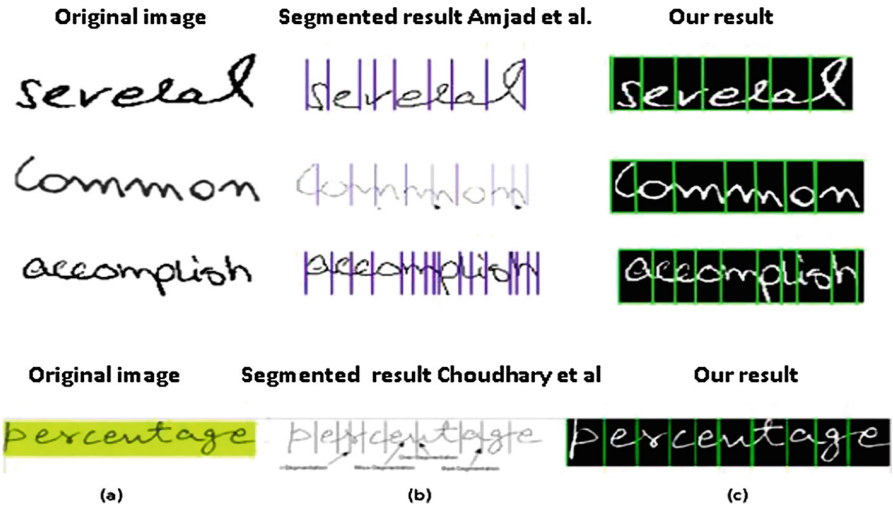

Fig. 8. (a) Original image, (b) wrong segmented images, (c) proposed method segmentation result.

Fig. 9. Wrongly segmented samples.

6 Conclusion

In this paper, an improved version of the vertical pixel count algorithm is presented. The Proposed technique was tested on IAM and self-created data set to check the efficiency of the presented algorithm. The proposed method is simple and very effective on connected text but in case of overlapping characters obtained results are not satisfactory. Further investigation will be made to handle overlapping characters by segmenting at a slope instead of vertical segmentation.

References

1. Choudhary, A., Rishi, R., Ahlawat, S.: A new character segmentation approach for off-line cursive handwritten words. Proc. Comput. Sci. **17**, 88–95 (2013)
2. Javed, M., Nagabhushan, P., Chaudhuri, B.B.: A direct approach for word and character segmentation in run-length compressed documents with an application to word spotting. In: 2015 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 216–220. IEEE (2015)
3. Lee, S.W., Lee, D.J., Park, H.S.: A new methodology for gray-scale character segmentation and recognition. IEEE Trans. Pattern Anal. Mach. Intell. **18**(10), 1045–1050 (1996)
4. Marti, U.V., Bunke, H.: Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In: Proceedings of Sixth International Conference on Document Analysis and Recognition, pp. 159–163. IEEE (2001)
5. Marti, U.V., Bunke, H.: The IAM-database: an English sentence database for offline handwriting recognition. Int. J. Doc. Anal. Recogn. **5**(1), 39–46 (2002)
6. Oliveira, L.S., Britto, A., Sabourin, R.: A synthetic database to assess segmentation algorithms. In: Eighth International Conference on Document Analysis and Recognition (ICDAR 2005), pp. 207–211. IEEE (2005)
7. Rehman, A., Mohamad, D., Sulong, G.: Implicit vs explicit based script segmentation and recognition: a performance comparison on benchmark database. Int. J. Open Prob. Comput. Math. **2**(3), 352–364 (2009)
8. Rehman, A., Saba, T.: Performance analysis of character segmentation approach for cursive script recognition on benchmark database. Digi. Signal Process. **21**(3), 486–490 (2011)
9. Saba, T., Rehman, A., Sulong, G.: Cursive script segmentation with neural confidence. Int. J. Innov. Comput. Inf. Contr. (IJICIC) **7**(7), 1–10 (2011)
10. Salvi, D., Zhou, J., Waggoner, J., Wang, S.: Handwritten text segmentation using average longest path algorithm. In: 2013 IEEE Workshop on Applications of Computer Vision (WACV), pp. 505–512. IEEE (2013)
11. Tulyakov, S., Govindaraju, V.: Probabilistic model for segmentation based word recognition with lexicon. In: Proceedings of Sixth International Conference on Document Analysis and Recognition. pp. 164–167. IEEE (2001)
12. Yamada, H., Nakano, Y.: Cursive handwritten word recognition using multiple segmentation determined by contour analysis. IEICE Trans. Inf. Syst. **79**(5), 464–470 (1996)