Journal of Engineering Science and Technology
Vol. 15, No. 4 (2020) 2339 - 2354

SEGMENTATION OF OVERLAPPING CHARACTERS IN LANNA USING MIXED ALGORITHM

RUJIPAN KOSARAT[1]*, NUALSAWAT HIRANSAKOLWONG[1]

[1]Department of Computer Science, Faculty of Science,
King Mongkut's Institute of Technology Ladkrabang,
Ladkrabang, Bangkok 10520, Thailand
*Corresponding Author: 60605013@kmitl.ac.th

Abstract

Lanna language is a popular language in the northern part of Thailand. The segmentation of printed Lanna characters is a challenging problem. Normally, the segmentation method of characters begins with using horizontal and vertical histograms into line segmentation and character segmentation. Then, when written the output will be the correct clear characters, overlapping characters and touching characters. Frequently, there are some problems with the overlapping characters and the touching characters. This paper focuses on only the overlapping characters, overlapping between alphabets and vowels. The methods will segment the overlapping characters by histogram techniques, beginning with splitting, rotating and merging the characters. The experiments used the printed Lanna characters in many books as a data set. The results achieved an accuracy rate of 96.72%.

Keywords: Character segmentation, Merge character, Overlapping character, Rotate character, Split character.

1. Introduction

Lanna language was a popular language in the Kingdom of Lanna, more than 500 years ago in the northern part of Thailand. The Lanna language is similar to the language used in Myanmar, called Mon. Customarily, this language will appear on the Moraine Tripitaka, fair allegory prayer inscriptions on palm leaves, and chronicles of the legendary history of astrology. There are many books written with the Lanna language (both alphabets and vowels). In the northern part of Thailand, there are classes where Lanna language is taught in many schools, especially for monks. Printed Lanna documents are widely used. The automatic segmentation of printed Lanna characters is very helpful for teaching in class and can be applied for many applications, such as automatic book reading, and e-Learning for Lanna teaching classes.

The Lanna language contains 42 alphabet letters, 8 top vowels, and 14 single vowels, 9 forced tone marks, 12 transforming alphabet letters, and 10 number characters. The mix between alphabet and vowels in Lanna language have styles similar to Thai characters. The Lanna language consists of alphabet letters around vowels. Vowels can be placed on the top, bottom, left and right of alphabet letters. Nowadays, Lanna characters on the palm leaves, Lanna literature, or published books are mostly written in three levels, as shown in Fig. 1.

Fig. 1. A word in the former times of Lanna language.

The character segmentation is the first step to create an OCR system. It is an operation that seeks to decompose an image of a sequence of characters into sub-images of individual symbols. It is one of the decision processes in a system for optical character recognition (OCR). Text-lines and character segmentation is still the challenge in OCR development for handwritten documents, and text documents, because of various distortions of document layout, unusual spacing between lines and characters, and so on. At the present time, many methods have been developed for segmentation of characters in text documents, but the problem of character segmentation is not fully deciphered in many languages.

Today, there are many OCR (Optical Character Recognition) in the online applications, but it cannot be used for the Lanna language. Therefore, this article tries to focus on the segmentation of Lanna characters and to recognize Lanna characters for their development to have Lanna language in OCR applications available online in the future.

Therefore, this paper focuses on the images of printed Lanna documents. The algorithm begins with filtering noises onto the images using the salt and pepper method. Then, follow up with the horizontal and vertical histograms for segmentation of lines and characters, correspondingly. The output will be the correct clear characters, the overlapping characters, and the touching characters. Previous research work solved the touching character issues. Therefore, this paper will focus on solving

the overlapping characters in printed Lanna documents. The techniques began with using the split character subroutine to separate overlapping characters into the top part and bottom part, and then separate the top part into the left and right of the top part. This technique may rotate the top part of the character. After which the method will merge the top part and bottom part at the connection point. The algorithm will be explained in more detail in Section 3.

2. Related Work

Over the past ten years, text detection and character segmentation has received a lot of attention from the research society [1-4], while segmentation of characters from complex backgrounds was also interesting in education circles. Khushbu and Isha [5] proposed a fast Otsu algorithm based on improved histogram to reduce the high computation complexity of 1D and 2D Otsu algorithms, but the variance of the object and the background intensity will give the incorrect threshold. Kiruba et al. [6] proposed using Otsu's algorithm for background elimination to separate text, and a simple histogram-based approach to segment line and characters on Tamil palm script characters. After separating characters between lines and word segmentation, some characters could not be separated, such as on the overlapping characters and touching characters.

Bharathi and Chandrasekar [7] proposed an overlapping bounding box approach for segmenting the conjunct consonants along with an algorithm for identifying the correct touching location in Telugu script, which achieved high accuracy rates, but failures were found to be on some characters which had projections into the bottom zones, and second characters projecting into the middle zone. Some of the rejected letters also include the alternative forms of the conjunct consonants. Yibin et al. [8] proposed using the method of dividing the tree-indexed areas using forward and backward search and proposed to divide the group of exposed characters. However, this method was sensitive to the groups of narrow characters.

There are many research papers that focus on segmentation of the characters in touching and overlapping characters in several other languages, such as Daldali and Souhar [9], who proposed using the seam carving approach to text line segmentation for an Arabic text. Shuchi and Vivek [10] proposed a fragmentation of Handwritten Touching Characters in Devanagari Script. However, this method failed for certain characters because of variations in handwriting.

Giuseppe et al. [11] proposed a novel method based on fuzzy logic segmenting of touching characters in the case of Latin printed and handwritten characters. The system performances are illustrated and supported by numerical examples showing that this approach could achieve a reasonably good overall accuracy in segmenting characters. Tapan et al. [12], proposed a method using regularized logistic regression and neural network for handwritten English character recognition. There are constraints as to the difficulty of obtaining ground truth data. This approach was tested on a small dataset.

Kraisak and Phornsiri [13] proposed to exploit an approximate string matching (ASM) technique to resolve the ambiguous problems of words when OCR cannot recognize some Thai characters. Soumendu and Sreeparna [14] proposed an approach for Japanese Character Recognition. The method is formulated in two steps: coarse segmentation and fine segmentation. Using this process to employ a

recognition model by combining the linguistic context and geometric contexts to recognize over segmented characters and consider all possible touching positions.

Nguyen et al. [15] presented a methodology for Chinese word segmentation based on a conditional random field. Supriana and Albadr [16] proposed an approach using the Arabic Optical Character Recognition (AOCR) system. Unfortunately, overall performance of the system only reached 48.3% accuracy. Divakar et al. [17] proposed for optical character recognition for Hindi language using a neural network.

Vikas and Vijay [18] proposed an approach for character segmentation for Telugu image documents by using multiple histogram projections using morphological operators to extract features of the image. Sakkayaphop and Arit [19] using an example of previous work undertaken on Lanna languages, proposed a method combined with Otsu's method and multi-thresholding method, and then used thinning algorithms to extract the skeleton of the touching characters in Lanna handwritten documents. The accuracy rate was relatively low and focused on the touching characters.

Finally, separating the characters from each other, recognition is the last step in making OCR. Khaled [20] proposed an approach by using a deep neural network for the handwritten Arabic character recognition problems that uses convolutional neural network (CNN) models with regularization parameters.

In previous work [21], the proposed methods for segmentation of touching characters in printed Lanna script used junction point. The method was computed for the left edge junction points and right edge junction points. Afterward find their maximum numbers and find the value of its row to separate consonant and vowels from touching. The results gained a high accuracy rate for separating the touching characters at 95.81%, while the limitation of the previous work could not segmentation of the overlapping characters.

Therefore, this new research focused on solving the overlapping characters in printed Lanna documents, for more detail in Sections 3 to 5.

3. Proposed Approach

The main program of the proposed approach is shown in Fig. 2, and subroutines as shown in Figs. 3-5.

The data set is created from scanning a variety of books written in Lanna language with 300 dpi, of 112 images. The proposed approach used is as follows: read image, pre-processing, line segmentation, character segmentation, and then make a decision from the output characters. If the output character is a touching character, this is solved using our previous research work. If the output character is a clear character, it is a correct segment character and nothing is done to this character. If the output character is an overlapping character, the technique will do in subroutines of overlapping character segmentation. All of the proposed approaches are shown in Figs. 2-5.

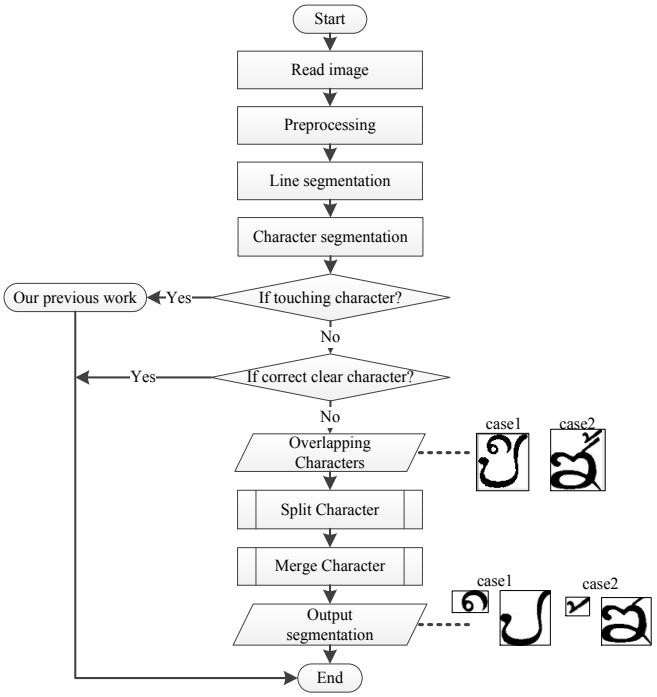

Fig. 2. Main program of the proposed algorithm.

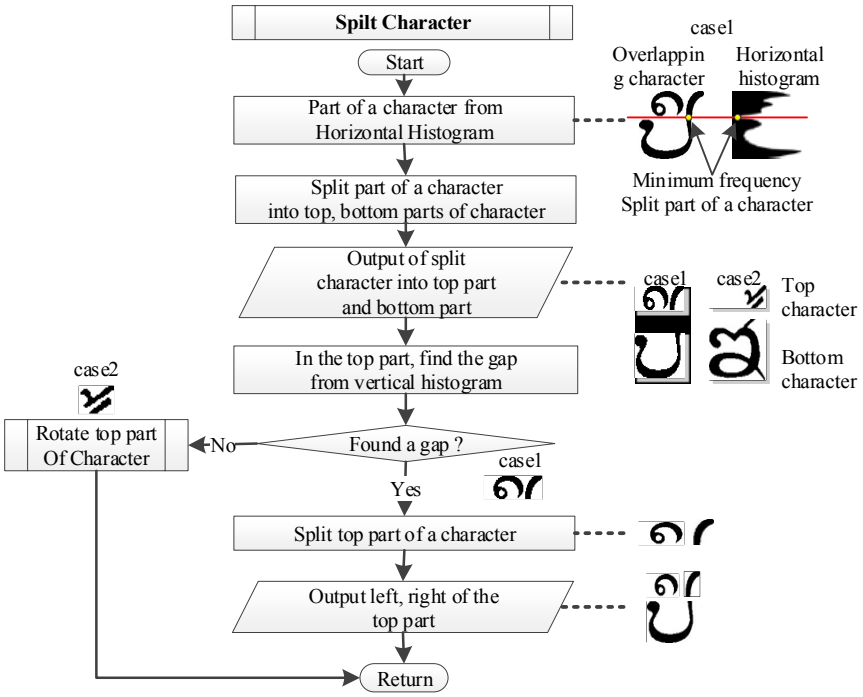

Fig. 3. Split character subroutine.

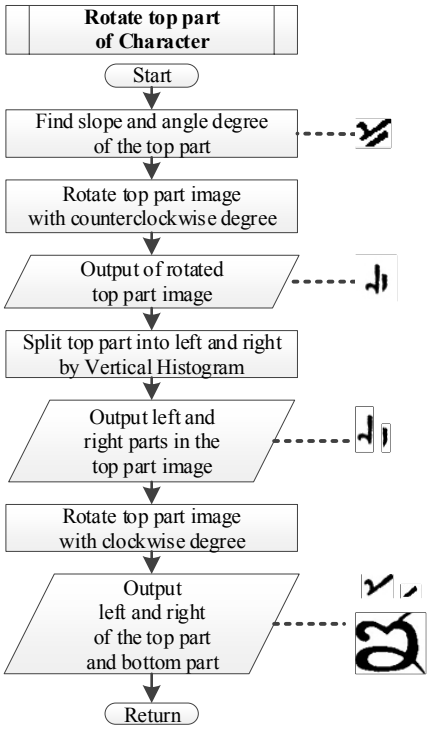

Fig. 4. Rotate character subroutine.

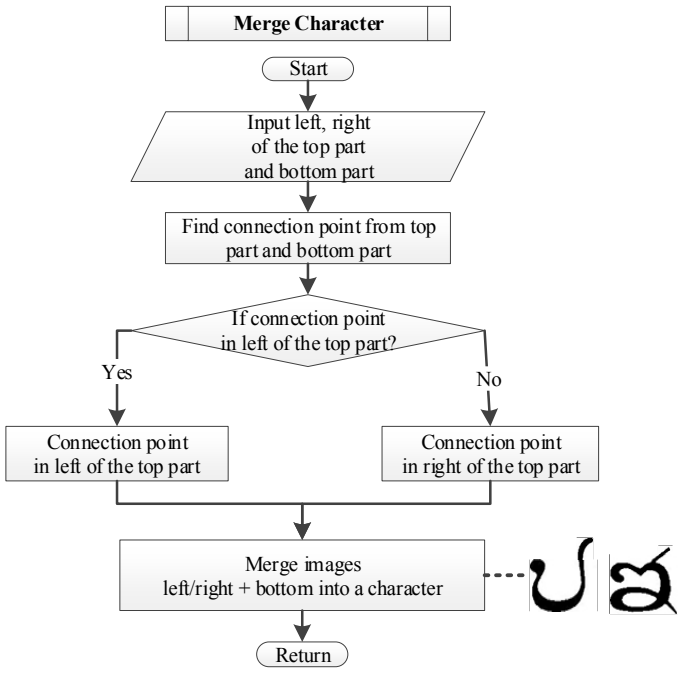

Fig. 5. Merge character subroutine.

3.1. Preprocessing

The system reads images and then does the pre-processing as per the following steps: -

i. The scanned images may have some noises from the scanning machine. The system is applied using the salt and pepper method to reduce noises.
ii. The text images are converted into binary images by thresholding with Otsu's method. The original image contains 1 representing object (black pixels) and 0 representing as background (white pixels).

3.2. Horizontal and vertical histograms

After the data set images are processed with the pre-processing method as discussed in Section 3.1, the system applies various techniques for segmentation to gain document lines and characters. The process of segmentation is as per the process outlined below: -

3.2.1. Line segmentation

The horizontal projection for finding lines is used to compute the frequency of all black pixels on each row, horizontal histogram in each row as shown in Fig. 6. The steps for line segmentation algorithm are as follows: -

i. Count the frequency of black pixels in each row and show its histogram.
ii. Mark the bounding box for Lanna text lines after using the horizontal histogram.
iii. Extract each Lanna text line from its histogram (zero frequency means an empty line).

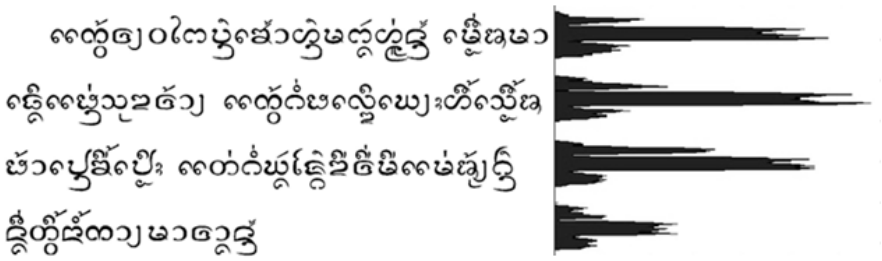

Fig. 6. An example of text lines from the horizontal histogram.

3.2.2. Character segmentation

The vertical projection for finding character methods is used to compute the frequency of all black pixels on each column, the vertical histogram of each column is as shown in Fig. 7. The steps for character segmentation algorithm are as follows: -

i. Create the vertical histogram from each extracted text line.
ii. Count the frequency of black pixels in each column.
iii. Mark the bounding box for text characters in Lanna language after using a vertical histogram.
iv. Extract each text character from its histogram (zero frequency means no character or blank).

Fig. 7. An example of text characters from vertical histogram.

From Character segmentation, all of the extracted text characters that cannot be separated into levels will be selected as boundary boxes. These boundary boxes may be found in three kinds of characters, the correct clear characters as shown in Fig. 8(a), the touching characters as shown in Fig. 8(b) and the overlapping characters as shown in Fig. 8(c).

(a) The correct clear characters. **(b) Touching characters.**

(c) Overlapping characters.

Fig. 8. Some examples for segmentation of Lanna characters.

The correct clear character is a successful output for character segmentation. A touching character is a character that contains either the two alphabets or vowels touching each other between levels. This problem had been solved in our previous work. An overlapping character is a character that contains two alphabets and/or vowels stacked (in any orientation except of horizontal orientation) between levels. This paper will focus on the overlapping character segmentation.

3.3. Overlapping character segmentation

The steps for overlapping character segmentation algorithm are as follows: split the character, rotate the top part of character and merge character.

3.3.1. Split character

The horizontal projection of overlapping characters is used to compute the frequency of all black pixels in each row as shown in Fig. 9. The steps for splitting parts of characters are as follows: -

i. Create the horizontal histogram from each extracted overlapping character.
ii. Count the frequency of black pixels in each row.
iii. Find the frequency in each row of the overlapping characters after using the horizontal histogram.

iv. Split parts of the overlapping character at a point that has the minimum frequency of the horizontal histogram.

v. Save images obtained from separating the overlapping characters in each separate file, top part of a character and bottom part of character.

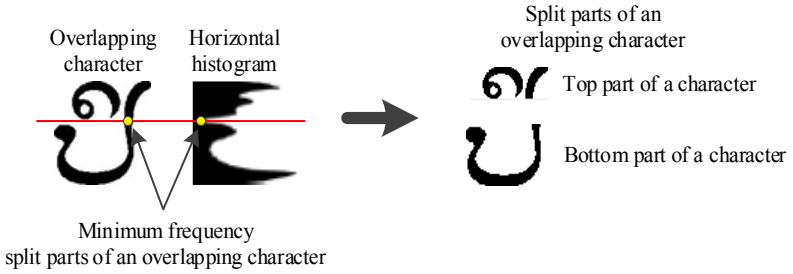

Fig. 9. An example of split parts in a character.

From Fig. 9, the overlapping character is split into 2 parts: the top part and the bottom part. The top part is in the upper zone, either a vowels or tone masks, mostly attached to the consonant. The bottom part is in the middle zone, a consonant.

The vertical projection of the top part is used to compute the frequency of all black pixels in each column as shown in Fig. 10. The steps for finding a gap in the top part are as follows: -

i. Create the vertical histogram from the top part of the overlapping character.

ii. Count the frequency of black pixels in each column.

iii. Split the left part and the right part at the gap of the vertical histogram.

iv. Save images of the left part and the right part in the separate files.

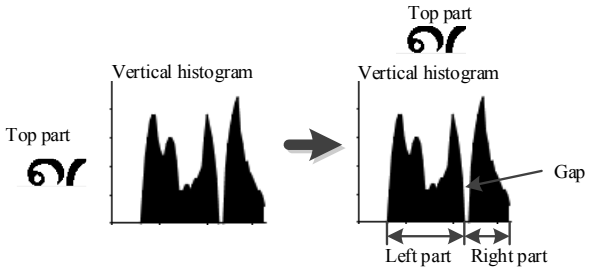

Fig. 10. Find the gap between 2 parts by using vertical histogram.

On the other hand, finding the gap between 2 parts can be described as follows. From the top part, the system processes vertical histograms as in Eq. (1). And then the system finds the gap between the left part and the right part as shown in Fig. 11.

$$f_i = \sum_{j=1}^{h} b_j \quad (1)$$

where f_i is the frequency of black pixels in column i^{th}, h is a height of the top part, b_j is a {1 (black pixels), 0 (white pixels)} in the row j^{th}, $i \in [1, \text{width}]$, and $j \in [1, \text{height}]$.

```
1.  SET start = 0                       // start is the starting point
2.  SET stop = 0                        // stop is the ending point
3.  SET count = 0                       // count = 1(rotate), or count = 2(split)
4.  SET round = 0                       // round can be only 0 or 1
5.  SET w = img.width                   // width of the image
6.  WHILE round < 2
7.     FOR i = start TO w               // i represents for the column number
8.        IF f[i] > 0 THEN              // f[i] represents for frequency in column i^th
9.           SET start = i              // this gap in the second round
10.          BREAK
11.       END IF
12.    END FOR
13.    FOR i = start TO w
14.       IF f[i] == 0 THEN
15.          SET stop = i
16.          BREAK
17.       END IF
18.    FOR i = start TO w
19.       IF start == stop OR stop = 0 THEN
20.          count = 1                  // rotate
21.       ELSE
22.          count = 2                  // split at the gap
23.       END IF
24.     END FOR
25.     start = stop
26. END WHILE
27. RETURN count
```

Fig. 11. Finding the gap between the left part and right part.

After checking the gap between the left part and the right part by using the vertical histogram, if a gap is found then the program will separate the top part into the left part and right part at the gap point, as shown in Fig. 12, and then proceed to section 3.3.3. If the character cannot be separated, the program will continue in section 3.3.2.

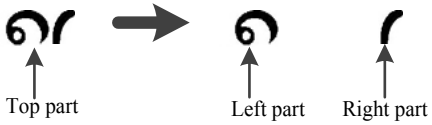

Fig. 12. Split top part into the left part and right part at the gap.

3.3.2. Rotate top part of character

Some top parts cannot be separated by using the vertical histogram because of not the gap in the top part, as shown in Fig. 13.

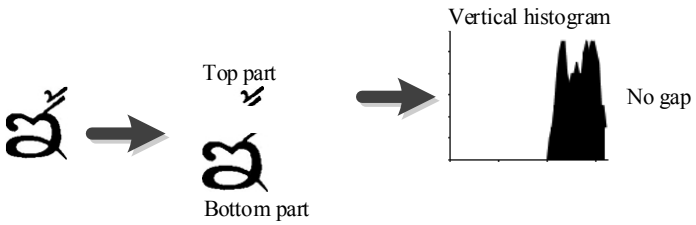

Fig. 13. A top part cannot be separated without a gap.

Therefore, the top part should be rotated to find a gap, by using the vertical histogram in the rotated top part, and then separated into the left part and right part at the gap.

In rotation on the x-axis, an image is rotated with counterclockwise movement in angle θ, using the coordinates (x, y) associated with the same axis. Using the rotation principle, the relationship between the source coordinates (x, y), and the destination coordinates (x', y') as shown in Fig. 14.

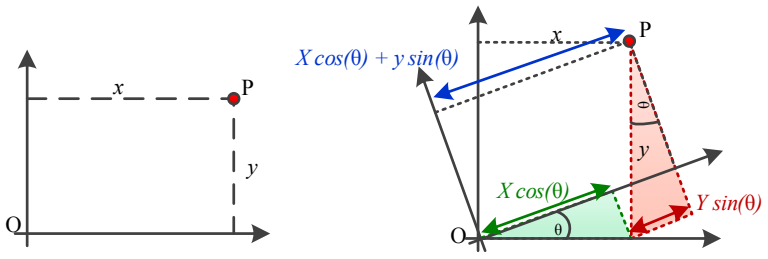

Fig. 14. Rotation of image.

In two dimensions, the standard rotation matrix is represented in Eq. (2) and the rotation column vectors are represented in the Eq. (3). The new coordinates (x', y') of the point (x, y) after rotation are computed from Eq. (4) and Eq. (5).

$$\boldsymbol{R}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\cos\theta} & \boldsymbol{-\sin\theta} \\ \boldsymbol{\sin\theta} & \boldsymbol{\cos\theta} \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\cos\theta} & \boldsymbol{-\sin\theta} \\ \boldsymbol{\sin\theta} & \boldsymbol{\cos\theta} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \tag{3}$$

$$\boldsymbol{x' = x\cos\theta - y\sin\theta} \tag{4}$$

$$\boldsymbol{y' = x\sin\theta + y\cos\theta} \tag{5}$$

The direction of vector rotation is counterclockwise if θ is positive (e.g. 90°), and clockwise if θ is negative (e.g. −90°). Thus, the clockwise rotation matrix is shown in Eq. (6).

$$\boldsymbol{R}(\boldsymbol{-\theta}) = \begin{bmatrix} \boldsymbol{\cos\theta} & \boldsymbol{\sin\theta} \\ \boldsymbol{-\sin\theta} & \boldsymbol{\cos\theta} \end{bmatrix} \tag{6}$$

Finding an angle degree to rotate the top part is used as Eqs. (7-8). Rotate the image of the top part with counterclockwise (90- θ) movement as shown in Fig. 15.

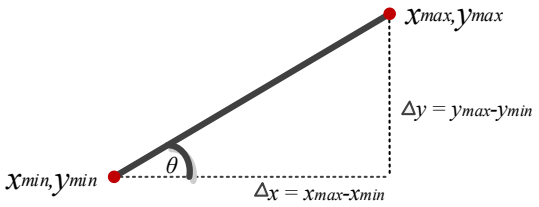

Fig. 15. Find an angle degree from a slope of the line.

$$\boldsymbol{Slope} = \frac{\boldsymbol{y_{max} - y_{min}}}{\boldsymbol{x_{max} - x_{min}}} \quad (7)$$

$$\boldsymbol{\theta} = \mathbf{tan^{-1}}(\boldsymbol{Slope}) \quad (8)$$

From Fig. 16, the image of the top part is rotated counterclockwise by turning to a 56.70° degree that is computed from the slope of the top part character. Then the vertical histogram has a gap. Consequently, the gap can be separated into two parts: left part and right part. After the top part is split into the left part and right part, then it can be rotated clockwise with -56.70° degree, as shown in Fig. 17.

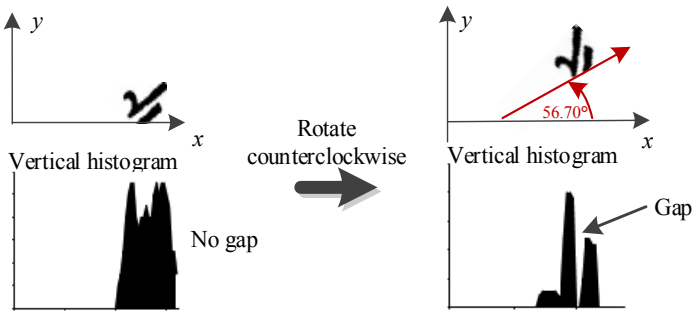

Fig. 16. Rotation at counterclockwise 56.70° degree of the top part.

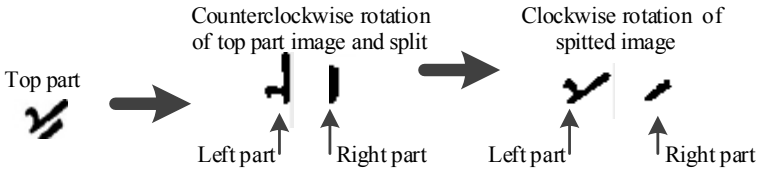

Fig. 17. An example of segmentation of the top part.

3.3.3. Merge character

From section 3.3.1 and 3.3.2, the overlapping character is separated into the left part, right part, and bottom part. After this is done, the top part (left part and right part) and the bottom part will be merged by finding a connection point to be the original character. The connection point is a black pixel in the last row of the top part, and the first row of the bottom part, as shown in Fig.18.

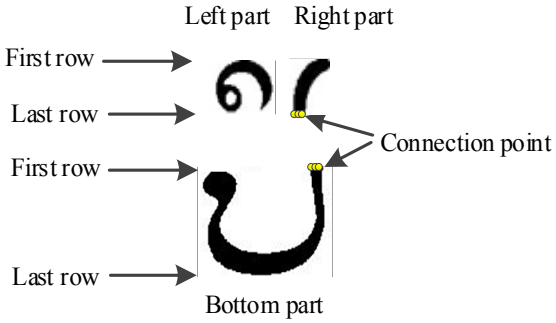

Fig. 18. An example split parts with a connection point.

The connection point can be found only by searching for the column position of black pixels in the last row from the top part that matches with the column position of black pixels in the first row from the bottom part. As a final point, the connection between the top part and bottom part are checked successfully. The characters will be merged at the connection point as shown in Fig. 19.

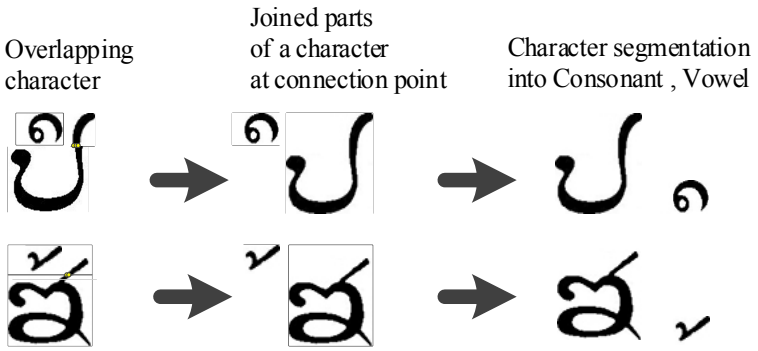

Fig. 19. Examples for segmentation of overlapping characters.

4. Experimental results

The data set is created from scanning the various categories of Lanna textbooks into greyscale images by setting resolution with 300 dpi of 112 images. There is a total of 7,724 characters including 396 overlapping characters. The data set was processed with the proposed approach, and the output results are shown in Tables 1-4.

The overlapping characters are separated into 2 cases. The character in case 1 can be separated from the top part character with the gap without rotation. The character in case 2 can be separated from the top part character with rotation.

Table 1. The results of the overlapping characters by using the proposed method (some examples of case 1).

Character	Top part	Bottom part	Left part	Right part	Merge character

Table 2. The results of the overlapping characters by using the proposed method (some examples of case 2).

Character	Top part	Bottom part	Rotate (degree)	Result rotate	Left part	Right part	Merge character
			56.70°				
			53.57°				

Table 3. The results of the overlapping characters.

The overlapping Characters	Character segmentation by the proposed algorithm	
	Consonants	Vowels/Tone mark
ᩁ	ᩁ	ᩍ
ᩅᩥ	ᩅ	ᩥ
ᨾᩩ	ᨾ	ᩩ
ᩈᩤ	ᩈ	ᩤ
ᨠ᩠	ᨠ	᩠
ᩃ᩠ᩃ	ᩃ	᩠
ᩁ᩠ᩁ	ᩁ	᩠

Table 4 shows the summary results of the database set using the proposed approach. The total document consists of 112 pages, with 7,724 characters, and 396 of overlapping characters. The experimental results segment correctly with a total of 383 correct segments with the overlapping characters from 396, with an accuracy rate 96.72%.

Table 4. The summary results of overlapping characters.

Total documents (pages)	112
Total characters	7,724
Total overlapping characters	396
Total correct segments (overlapping characters)	383
Accuracy rate (%)	96.72

There is an overlapping character ᨦ that cannot be separated correctly. From the original character, ᨦ after running the split character subroutine the output results are the top part ᨧ, and bottom part ᨦ. The gap from top part cannot find the vertical histograms, even though using a rotation in the top part of character. The input and output from Fig. 4 (rotate character subroutine) are the same as ᨧ. The solution to solve this problem is a recursive function of the split character subroutine, again by using ᨧ as an input. Then the results of segmentation will be correct. This will gain more of an accuracy rate.

5. Conclusions

The experimental results of this research show that the proposed approach can segment the overlapping characters from the various categories of Lanna textbook images with an accuracy rate of 96.72%. This high accuracy rate can confirm that the proposed method is suitable for solving the segmentation of overlapping characters. Some incorrect segmentation may come from the error of scanning the image, having less frequency characters in the data set, or the complicated characters. For addressing complicated characters, researchers can do a recursive function of the split character subroutine as outlined in Section 4. This will gain a higher accuracy rate.

Nomenclatures

f_i	Frequency of black pixel
b_j	Pixels of the row (1 is black pixels), 0 is white pixels)
j	A column of the top part
h	A height of the top part
R	Rotation image
x	The coordinates of the point x-axis
x'	The coordinates of the point after rotation x-axis
y	The coordinates of the point y-axis
y'	The coordinates of the point after rotation y-axis
x_{max}	Maximum point on the x-axis
x_{min}	Minimum point on the x-axis
y_{max}	Maximum point on the y-axis
y_{min}	Minimum point on the y-axis
Slope	Detection slope of line in image

Greek Symbols

θ	Angle of slope (degree)

Abbreviations

LANNA	Lanna language

References

1. Harpreet, M.; and Chetan, M. (2017). A review on the text segmentation techniques. *International Journal of Engineering and Computer Science*, 6(3), 20567-29571.
2. Namrata, D. (2015). Segmentation methods for handwritten character recognition. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(4), 155-164.
3. Souhar, A.; Boulid, Y.; Ameur, EIB.; and Ouagague, M. (2017). Segmentation of Arabic handwritten documents into text lines using watershed transform. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(6), 96-102.
4. Alok, K.; Madhuri, Y.; Tushar, P.; and Bhupendra, K. (2013). A survey on touching character segmentation. *International Journal of Engineering and Advanced Technology (IJEAT)*, 2(3), 569-574.
5. Khushbu; and Isha, V. (2017). Otsu image segmentation algorithm: A review. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(6), 11945-11948.
6. Kiruba B.; Nivethitha A.; and Vimaladevi M. (2017). Segmentation of handwritten Tamil character from palm script using histogram approach. *International Journal of Informative & Futuristic Research*, 4(5), 6418-6424.
7. Bharathi, J.; and Chandrasekar, P.R. (2013). Segmentation of Telegu touching conjunct consonants using overlapping bounding boxes. *International Journal of Soft Computing and Engineering (IJSCE),* 5(6), 538-546.

8. Yibin, T.; Xufei, L.; Yan, Z.; Meng, L.; and Mengying, X. (2017). Segmentation of touching characters via tree-indexed demarcation using forward and backward searches. *Advances in Mechanical Engineering*, 9(10), 1-11.
9. Daldali M.; and Souhar A. (2018). Handwritten Arabic documents segmentation into text lines using seam carving. *International Journal of Interactive Multimedia and Artificial Intelligence*, 5(5), 1-8.
10. Shuchi, K.; and Vivek, V, (2014). Fragmentation of handwritten touching character in Devanagari script. *International Journal of Information Technology, Modeling and Computing (IJITMC)*, 2(1), 11-21.
11. Giuseppe, A.F.; Nadir, M.; and Rosaria, R. (2016). A fuzzy approach for segmentation of touching characters. https://arxiv.org/.
12. Tapan, K.H.; Rajdeep, S.; and Ankit, K. (2017). Handwritten English character recognition using logistic regression and neural network. *International Journal of Science and Research (IJSR)*, 5(6), 750-754.
13. Kraisak K.; and Phornsiri P. (2018). Optical Character Recognition (OCR) enhancement using an approximate string-matching technique. *Engineering and Applied Science Research*, 45(4), 282-289.
14. Soumendu, D.; and Sreeparna, B. (2015). An algorithm for Japanese character recognition. *International Journal of Image, Graphics and Signal Processing*, 1.
15. Nguyen K.C.; Nguyen C.T.; and Nagakawa M. (2017). A segmentation method of single and multiple touching characters in offline handwritten Japanese text recognition. *The Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems*, 100(12), 2962-2972.
16. Supriana I..; and Albadr, N. (2013). Arabic character recognition system development. *The 4th International Conference on Electrical Engineering and Informatics (ICEEI 2013*). Selangor, Malaysia, 11, 334-341.
17. Divakar, Y.; Sonia, S.C.; and Jorge, M. (2013). Optical character recognition for Hindi language using a neural-network approach. *The Journal of Information Processing Systems (JIPS)*, 9(1), 117-140.
18. Vikas, J.D.; and Vijay, H.M. (2011). Devanagari document segmentation using histogram approach. *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, 1(3), 58-53.
19. Sakkayaphop, P.; and Arit, T. (2012). Segmentation of historical Lanna handwritten manuscripts, *IEEE International Conference Intelligent Systems*, Sofia, Bulgaria.
20. Khaled, S.Y. (2017). Arabic handwritten character recognition based on deep convolutional neural networks. *Jordanian Journal of Computers and Information Technology (JJCIT),* 3(3), 186-200.
21. Rujipan, K.; and Nualsawat, H. (2018). Segmentation of touching character printed Lanna script using junction point. *Journal of Engineering Science and Technology* (*JESTEC*), 13(10), 3331-3343.