Journal of Engineering Science and Technology
Vol. 13, No. 10 (2018) 3331 - 3343
© School of Engineering, Taylor's University

SEGMENTATION OF TOUCHING CHARACTER PRINTED LANNA SCRIPT USING JUNCTION POINT

RUJIPAN KOSARAT*, NUALSAWAT HIRANSAKOLWONG

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

In the northern part of Thailand since 1802, Lanna characters were popular as ancient characters. The segmentation of printed documents in Lanna characters is a challenging problem, such as the partial overlapping of characters and touching characters. This paper focuses on only the touching characters such as touching between consonants and vowels. Segmentation method begins with the horizontal histogram and then vertical histogram for segmentation of text lines and characters, respectively. The results are characters consisted of correct clear characters, partial overlapping characters, and touching characters. The proposed method computes the left edge junction points and right edge junction points. Then find their maximum numbers and find the value of its row to separate consonant and vowel from touching. The trial over the text documents printed in Lanna characters can be processed with an accuracy of 95.81%.

Keywords: Histogram, Line and character Segmentation, Touching character.

1. Introduction

Lanna language, commonly called Northern Thai language or Kam Mueang, is the language most used in Lanna empire, the Northern part of Thailand. Lanna language developed from ancient Mon script similarity with religion scripts of Laos and Burmese. This font of Lanna characters was used in the Lanna Kingdom since 1802. Currently, this language is used in religious that found in the temple of northern Thailand. The Lanna language is a likeness of northern Thai and Chiang Saeng languages. Six million people in north of Thailand and several thousand people in Laos used Lanna for communication, however, although nowadays the Lanna alphabets were inscribed on many places such as the wall in temples, ancient books and stone inscriptions, still there are very few people who know Lanna language. An automatic segmentation of printed Lanna document is very useful for teaching in the classroom and can be applied to different applications such as read automatically and e-learning. Nowadays, there are so many OCR (Optical Character Recognition) available online in many languages, however, it is not available for Lanna language. Therefore, this paper needs to focus the Lanna character segmentation and do the Lanna character recognition in the future work for planning to do OCR available online for Lanna Language in the future.

The Lanna language contains 42 consonants, 14 single vowels and 8 top vowels, 12 transform consonants, 9 forced tone marks, and 10 number characters. The mix between consonants and vowels in Lanna language styles is similar to Thai characters. The Lanna language is consists of consonants around vowels. A vowel can be placed on the top, bottom, left and right of consonants as shown in Fig. 1.

Current font Lanna on palm leaves or published books were mainly written in four levels, as shown in Fig. 2.

Level of words at the present time in Lanna language has four levels. The first level is a tone, which is placed on top of the consonant. The second level is a vowel. The third level is consonants in the main line and the fourth level is a vowel under consonant.

There are two types of overlapping characters; partial overlapping and touching overlapping. Touching and partial overlapping of characters are the first encountered problem like any other languages.

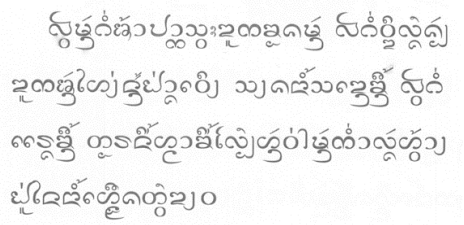


Fig. 1. An example of Lanna language.

Fig. 2. Level of words at the present time in Lanna language.

Some overlapping characters are letters of any two consonants and/or two vowels overlapping each other (in any direction) any levels shown in Fig. 3.

A touching character is a character that contains a consonant and a vowel touching each other between levels shown in Fig. 4. Over most touching character is found a few touching conjunct consonants in Lanna characters as shown in Figs. 5 and 6.

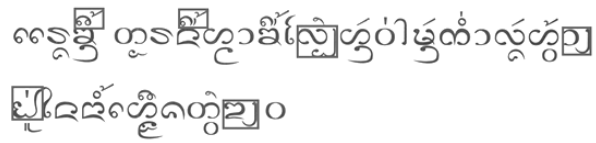


Fig. 3. Some partial overlapping Lanna characters in boundary boxes.

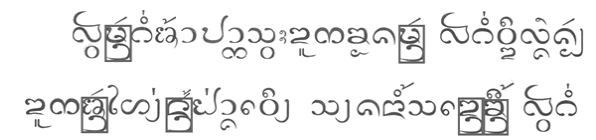


Fig. 4. Examples of touching Lanna characters in boundary boxes.

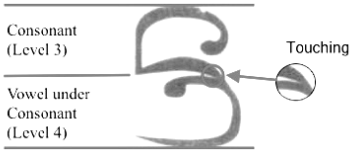


Fig. 5. Touching a consonant and a vowel between levels.

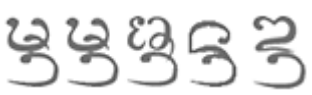


Fig. 6. Examples of few touching characters in Lanna characters.

To many researches, the most problem for segmentation of the characters is the touching characters in several other languages such as English language with a handwritten style [1], Thai language [2], Japanese language [3], Chinese language [4], Telugu language [5], Arabic language [6] and Hindi language [7]. Most of the researches proposed a solution of touching characters for OCR [8-17]. Many languages most problems are caused by the distortion of the image quality such as writing, photography, scanning and quality of ink. Some characters cannot avoid the overlap between a consonant and a vowel.

This paper focuses on touching Lanna characters from a data set of printed text documents in Lanna language. The data set of Lanna language images is created by scanning from various books in Lanna language. This method used the median filter to filter noises from the images. The horizontal and vertical histogram technique was used for segmentation lines and characters. The output is consists of correct clear characters, partial overlapping characters and touching characters. Finally, the methods will segmentation the touching characters by using junction point of edges.

2. Literature Review

Currently, there are various methods for segmentation characters into text documents, but the issue of character discrimination is not yet solvable in many languages. Singh et al. [18] proposed an approach for offline handwritten Devnagari character Recognition using the curvelet transform, the character geometry, and comparing their recognition performances using two different classifiers; the Support Vector Machine (SVM) with Radial Basis Function (RBF), and the k-Nearest Neighbors (k-NN) classifier. This paper reported experimental results of performances by comparing of Curvelets and geometry-based features with using k-NN and SVM with RBF classifier, the percentage accuracy classifier of the Geometry-based feature at 73.4 and 35.9 percent respectively. Likewise, the percentage accuracy classifier of the Curvelet-based feature was at 93.8 and 91.5 percent respectively. Comparison results show that k-NN classifier gives better results than SVM with RBF and the Curvelet-based features are more suitable for Devnagari character recognition.

There are not much research papers in character segmentation of Lanna language. Pravesjit and Thammano [19] with example of previous work with the Lanna languages, proposed a method combined Otsu's method and multi-thresholding method and then used thinning algorithm to extract the skeleton of the touching characters in Lanna handwritten documents. Finally, use the junction points to separate touching characters with an accuracy of 86.67%. The accuracy rate is quite low. At the present time, there are so many OCR available online in many languages. Unfortunately, there is not available for Lanna language. Therefore, the proposed method is not able to find a previous work of segmentation in printed Lanna documents for comparison purposes.

3. Proposed Algorithm

The main program of proposed algorithm begins with the input of Lanna text document images (as shown in Figs. 7 to 11). For each image, the algorithm will do pre-processing, segmentation (Line and character), and then segmentation of touching character as follows:

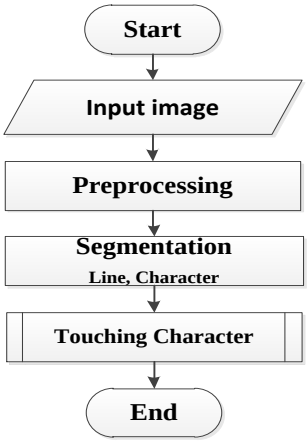


Fig. 7. Main program of the proposed algorithm.

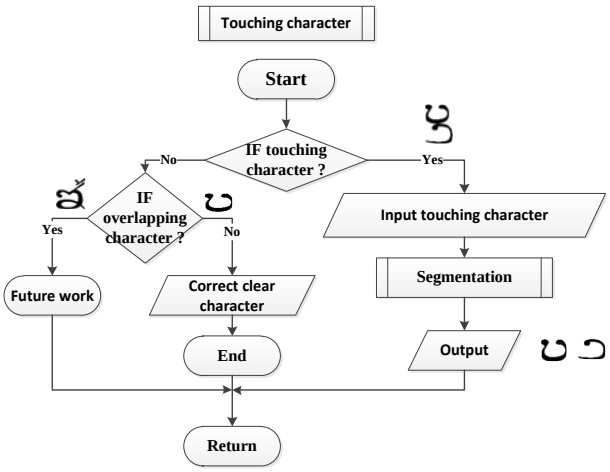


Fig. 8. Touching character subroutine.

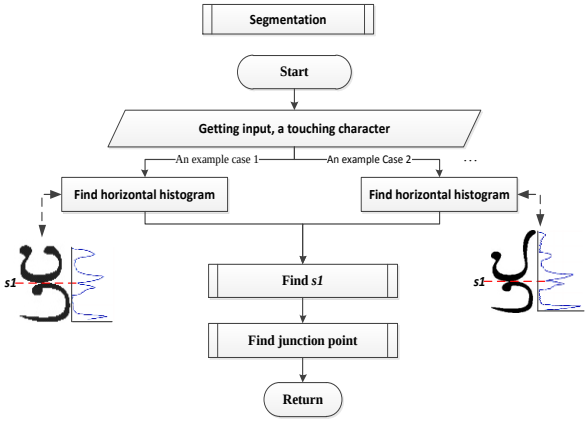


Fig. 9. Segmentation subroutine.

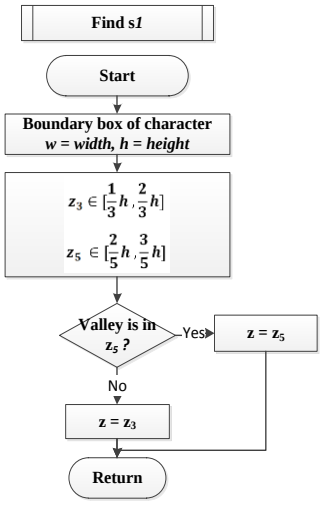


Fig. 10. Find *s*1 subroutine.

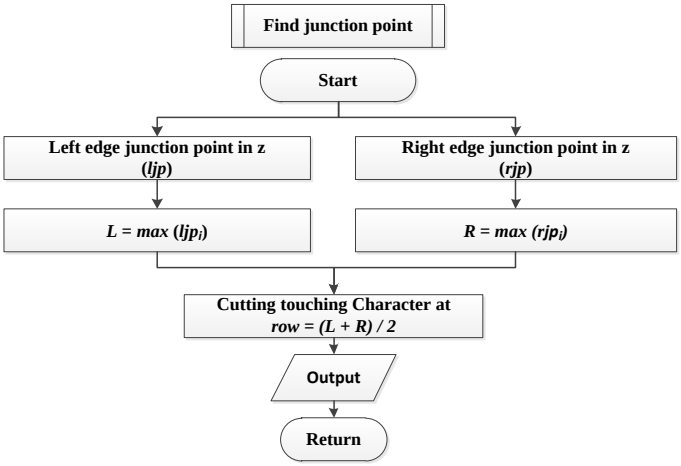


Fig. 11. Find junction point subroutine.

3.1. Preprocessing

For the preparation of the dataset, the system scans the books of the Lanna language in greyscale, with a resolution of 300 dpi, totally of 112 scanned images. Dataset images may have noise from the scanning process. The system uses a median filter [20] to reduce noises. Text images are converted to binary images by using Otsu's method [21]. A binary image represents 1 (black pixels) as an object and 0 (white pixels) as background.

3.2. Line and character segmentation approach

After the image was passed the pre-processing section, the system used the horizontal and vertical histograms [22] to segment lines and characters, respectively. The main segmentation is according to the following pattern. First, identify a line of text on the page: line segmentation methods based on technical histogram horizontally. Horizontal lines are used to calculate the frequency of black pixels in each row, which represents the horizontal histogram in each row. After that, specify the characters in each line. The private division of the alphabet represents a method based on the vertical histogram technique. The vertical projection method is used to calculate the frequency of black pixels in each column, which represents the vertical histogram of each column.

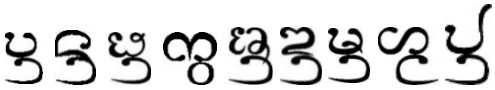
3.3. Character segmentation for finding touching character

From Character segmentation, all of the extracted text characters that cannot separate into levels will be selected as boundary boxes. These boundary boxes may be found three kinds characters, the correct clear characters shown in Fig. 12(a), partial overlapping characters are shown in Fig. 12(b) and touching characters in Fig. 12(c).

The problems for separating characters are partial overlapping characters and touching characters. This paper focuses on touching characters. A touching character is a character that contains a consonant and a vowel touching each other between levels shown in Fig. 13.

(a) The correct clear characters.

(b) Partial overlapping characters.



(c) Touching Lanna characters.

Fig. 12. Some examples for segmentation of Lanna characters.

Fig. 13. A touching character.

3.4. Find *s*1 subroutine

From a touching character, the system process horizontal histogram as in Eq. (1). Then the system process Find s1 subroutine as shown in Fig. 14.

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

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

```
1.  SET v = 0                          // Flag for valley
2.  SET u = 0                          // Flag for increase
3.  SET d = 0                          // Flag for decrease
4.  SET t = 0                          // Flag for turn form up to down
5.  FOR i = s1 TO s2                   // s1 is the start point
                                       // s2 is the ending point
                                       // for zone 3; s1 = 1/3 h, s2 = 2/3 h
                                       // for zone 5; s1 = 2/5 h, s2 = 3/5 h
6.      IF f[i] < f[i+1] THEN
7.          SET u = 1
8.      ELSE
9.          IF d = 1 AND u = 1 AND t = 1 THEN
10.             SET v = 1
11.             EXIT
12.         ELSE
13.             IF f[i] > f[i+1] THEN
14.                 SET d = 1
15.                 IF u = 1 THEN
16.                     SET t = 1
17.                     SET u = 0
18.                 END IF
19.             END IF
20.         END IF
21.     END IF
22. END FOR
23. RETURN v
```

Fig. 14. Pseudo code for finding *s1* subroutine.

3.5. Find junction point subroutine

The system tries to find out the touching point between the consonant and the vowel. The algorithm will divide the touching character into either 5 zones or 3 zones as shown in Fig. 15. Then focus on the middle zone where is a mostly touching zone in a touching character. Find junction point subroutine is as follows:

- Get the valley from finding ***s***1 subroutine in Fig. 14 and finding the middle zone by using Eq. (2) and then checking if the valley is in z_5 then $z = z_5$ otherwise $z = z_3$.

$$z_3 \in [\frac{1}{3}h, \frac{2}{3}h], z_5 \in [\frac{2}{5}h, \frac{3}{5}h] \quad (2)$$

- From the image of the touching character, scan each pixel from left to right line-by-line in middle zone or *z*.
- The system processes Eq. (3) as shown as the red colour points in Fig. 16.

$$ljp_i = \sum_{i=1}^{?} b_j \quad (3)$$

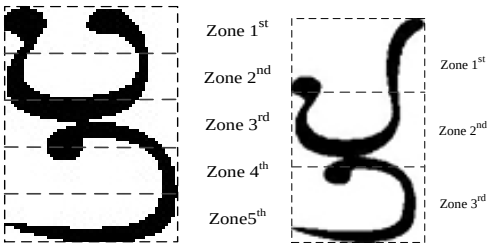


Fig. 15. Some examples for dividing either 5 zones or 3 zones.

where ljp_i is the frequency of white pixels in row i^{th}, w is a width of the touching character, b_j is a {1 (black pixels), 0 (white pixels)} in the column j^{th}, $i \in z$, and $j \in \{[1,?] \mid b_j = 0$ until found $b_? = 1$, ? is not greater than $w\}$.

- In the parallel process from step 3, the system processes Eq. (4) as shown as the blue colour points in Fig. 16.

$$rjp_i = \sum_{i=1}^{?} b_j \quad (4)$$

where rjp_i is the frequency of white pixels in row i^{th}, w is a width of the touching character, b_j is a {1 (black pixels), 0 (white pixels)} in the column j^{th}, $i \in$ z, and $j \in \{[1,?] \mid b_j = 0$ until found $b_? = 1$, ? is not greater than $w\}$. Note: 1 is equal to the width and moves from right to the left.

- Find maximum using $L = i$, where i is the maximum of ljp_i and $R = i$, where i is the maximum of rjp_i) as shown in Table 1.
- The system finds the junction point using $row = (L+R)/2$.

Table 1 shows an example of *ljp* and *rjp* points with a number each row in the middle zone (*z*) of the touching character. This example of touching character has height with 75 pixels and width with 53 pixels. The algorithm will find the maximum number of ljp_i and find the maximum number of rjp_i. The maximum number of ljp_i is equal to 22 in row number 37. Therefore, $L = 37$.

The maximum number of rjp_i is equal to 27 in row number 36. Therefore, $R = 3$. The result of the touching row by using junction point is in row 36th (row = 36.5) from row = $(L+R)/2$. Figure 17 is a representation of the touching character and its isolation result at row 36th.

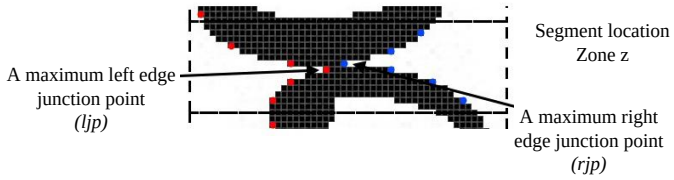


Fig. 16. *ljp* in the red color points and *rjp* in the blue color points.

Table 1. An example of *ljp* and *rjp* points in middle zone (*z*).

Rows	Junction point value	
	Left edge	Right edge
	ljp	***rjp***
31	4	17
32	5	17
33	6	17
34	8	19
35	11	23
36	16	**27**
37	**22**	19
38	19	15
39	16	12
40	15	10
41	14	9
42	13	7
43	13	6
44	13	3
45	13	3

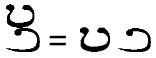


Fig. 17. An example of touching character and its isolation results.

4. Experimental Result

This dataset was created by scanning Lanna textbooks into a gray image by setting a 300 dpi resolution of 112 images. There are 7,724 characters, including 597 touching characters. This paper focuses only on touching characters. Dataset was processed using the proposed method and the results are shown in Tables 2 to 4.

Table 2 shows some examples of touching characters, their maximum of left edge junction points and right edge junction points, rows of left edge junction point and rows of right edge junction point, and the correction of segmentation character, respectively.

Table 3 shows the total results of the database sets used in the proposed approach. All documents contain 112 pages, 7,724 characters and 597 touching characters. When the system used $z = z_5$ in all cases, the results of the experiments could find 554 accurate from 597 with accuracy of 92.79%. The incorrect segmentation of a touching character comes from that its junction point is not always in zone $z = z_5$. The junction point of a touching character is mostly in zone $z = z_3$. If the system always divides in 3 zones, this will take time to process. Therefore, the purpose algorithm compromise by adjusting either 5 zones or 3 zones instead.

Table 2. Examples for the correct segmentation of touching characters.

Touching character	Maximum edge junction point value		Row of maximum edge junction point		Junction point row = *(L+R)/2*	Correction of segmentation character	
	ljp_i	rjp_i	*L*	*R*		Consonant	Vowel
	22	27	36	37	36.5		
	22	9	36	37	36.5		
	33	23	36	37	36.5		
	50	7	37	41	39		
	31	18	39	36	37.5		
	25	27	36	37	36.5		
	39	19	33	34	33.5		
	44	19	37	37	37		o
	40	27	34	35	34.5		
	23	27	63	62	62.5		

Table 3. The results of touching characters ($z = z_5$ in all cases).

Total documents (pages)	112
Total characters	7,724
Total touching characters	597
Total correct segments ($z = z_5$ in all cases)	554
Percent of accuracy segmentation (%)	92.79%

Table 4. The results of touching characters (either $z = z_5$ or $z = z_3$).

Total documents (pages)	112
Total characters	7,724
Total touching characters	597
Total correct segments	572
Percent of accuracy segmentation (%)	95.81%

5. Conclusions

From the experimental results, the proposed method can be summarized remarkable conclusions as follows:-

- From the observation, normally the junction point of touching character stays in the middle zone. It should be divided the zone into either 3 zones or 5 zones.

Well, if it is divided into 3 zones, the checking rows will be more than checking rows in the 5 zones. However, when using 3 zones, it may take more time than using 5 zones. The performance of dividing into 3 zones will cover more of the junction point of touching characters than the performance of dividing into 5 zones.

- For finding the junction point, the system computes only in the middle zone instead of the whole height of each touching character. This saves its processing time about 4/5 of the height of each touching character (in case dividing into 5 zones) or 2/3 of the height of each touching character (in case dividing into 3 zones). It can imply that the proposed algorithm can save processing time from 66.67 to 80% for each touching character.
- Firstly, the proposed approach asks whether there is a valley in the middle of 5 zones instead of running only 3 zones. This is for the maximum of saving processing time and gain more performance.

There are so many OCR available online with many languages, without Lanna language. This paper can segment touching character of printed Lanna script using junction point with high accuracy rate. Future work will focus on the segmentation of overlapping character in printed Lanna script and plan to do the recognition of Lanna characters for preparing to be an OCR available online for Lanna Language in the future.

Nomenclatures

fi	Frequency of black pixel
jp	Junction point
L	Row of left edge junction point
ljp	Left edge junction point
R	Row of right edge junction point
rjp	Right edge junction point

Abbreviations

LANNA	Lanna language

References

1. Sharma, N.; Patnaik, T.; and Kumar, B. (2013). Recognition for handwritten english letters: A review. *International Journal of Engineering and Innovative Technology*, 2(7), 318-321.
2. Phokharatkul, P.; and Kimpan, C. (2013). Handwritten Thai character recognition using fourier descriptors and genetic neural networks. *International Journal of Computational Intelligence*, 18(3), 270-293.
3. Soumendu, D.; and Banerjee, S. (2015). An algorithm for Japanese character recognition. *International Journal, Image Graphics and Signal Processing*, 7(1), 9-15.
4. Guo-hong, L.; and Peng-fei, Shi. (2004). An approach to offline handwritten Chinese character recognition based on segment evaluation of adaptive duration. *Journal of Zhejiang University Science*, 5(11), 1392-1397.

5. Anupama, N.; Rupa, C.; and Reddy, E.S. (2013). Character segmentation for Telugu image document using multiple histogram projections. *Global Journal of Computer Science and Technology Graphics & Vision*, 13(5), 11-15.
6. Al-Thaani, A.T.; and Al-Haj, S. (2010). Recognition of on-line Arabic handwritten characters using structural features. *Journal of Pattern Recognition Research*, 5(1), 23-37.
7. Jain, R.; Frinken, V.; Jawahar, C.V.; and Manmatha, R. (2011). BLSTM neural network based word retrieval for Hindi documents. *Proceedings of the International Conference on Document Analysis and Recognition*. Beijing, China, 83-87.
8. Gomathi, R.S.; Uma, D.R.S.; and Mohanavel, S. (2012). Segmentation of touching, overlapping, skewed and short handwritten text lines. *International Journal of Computer Applications*, 49(19), 24-27.
9. Bharathi, J.; and Reddy, P.C. (2013). Segmentation of Telugu touching conjunct consonants using overlapping bounding boxes. *International Journal on Computer Science and Engineering* (*IJCSE*), 5(6), 538-546.
10. Kapoor, S.; and Verma, V. (2014). Fragmentation of handwritten touching characters in Devanagari script. *International Journal of Information Technology, Modeling and Computing* (*IJITMC*), 2(1), 11-21.
11. Bansal, V.; and Sinha, R.M.K. (2014). Segmentation of touching and fused Devanagari characters. *Pattern Recognition*, 35(4), 875-893.
12. Das, M.S.; Reddy, C.R.K.; Govardhan, A.; and Saikrishna, G. (2010). Segmentation of overlapping text lines, characters in printed Telugu text document images. *International Journal of Engineering Science and Technology*, 2(11), 6606-6610.
13. Roy, P.P.; Pal, U.; Llados, J.; and Delalandre, M. (2009). Multi-oriented and multi-sized touching character segmentation using dynamic programming. *Proceedings of the International Conference on Document Analysis and Recognition*. Barcelona, Spain, 11-15.
14. Srivastav, A.; and Sahu, N. (2016). Segmentation of Devanagari handwritten characters. *International Journal of Computer Applications*, 142(14), 15-18.
15. Shah, A.N.; and Gaikwad, A.S. (2016). A review-recognition of license number plate using character segmentation and OCR with template matching. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(2), 159-162.
16. Man, H.; and Marwaha, C. (2017). A review on the text segmentation techniques. *International Journal of Engineering and Computer Science*, 6(3), 20567-20571.
17. Souhar, A.; Boulid, Y.; Ameur, E.; and Ouagague. (2017). Segmentation of Arabic handwritten documents into text lines using watershed transform. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4(6), 96-102.
18. Singh, B.; Mittal, A.; and Ghosh, D. (2011). An evaluation of different feature extractors and classifiers for offline handwritten Devnagari character recognition. *Journal of Pattern Recognition Research*, 6(2), 269-277.

19. Pravesjit, S.; and Thammano, A. (2012). Segmentation of historical Lanna handwritten manuscripts. *Proceedings of the* 6th *IEEE International Conference on Intelligent Systems*. Sofia, Bulgaria, 332-337.
20. Shrestha, S. (2014). Image denosing using new adaptive based median filter. *Signal & Image Processing: An International Journal* (*SIPIJ*), 5(4), 13 pages.
21. Jing, G.; Rajan, D.; and Siong, C.E. (2005). Motion detection with adaptive background and dynamic thresholds. *Proceedings of the Fifth International Conference on Information, Communications and Signal Processing*. Bangkok, Thailand, 41-45.
22. Dongre, V.J.; and Mankar, V.H. (2011). Devanagari document segmentation using histogram approach. *International Journal of Computer Science, Engineering and Information Technology* (*IJCSEIT*), 1(3), 46-53.