Journal of Engineering Science and Technology
Vol. 16, No. 5 (2021) 3726 – 3736

MIN-COST FLOW NETWORK TO DETECT TEXT LINE ON CERTIFICATE

INDRA RIANTO, EDNAWATI RAINARLI*

Department of Informatic Engineering –
Universitas Komputer Indonesia, Jl. Dipati Ukur 112-114 Bandung
*Corresponding Author: ednawati.rainarli @email.unikom.ac.id

Abstract

This study aims to use the Min-Cost Flow network to obtain text lines in the character detection that process on the certificate. Determining the text line is part of the text detection process before the word recognition process. Detection of the text that appears on the certificate is part of the process for automatically extracting information. The diversity of colour, font types and sizes, and the complexity of the certificate background make the scanner text detection more complex than other optical character detections. This study used the Tesseract tool to get character candidates and added merging characters into a text line using the Min-Cost Flow method. To improve the quality of the image, we used smoothing techniques with integral images. After the thresholding and segmentation process, the result is the input to Tesseract. The Tesseract detects the candidates of character. The test varies the confidence score, the threshold value for the vertical length of the two components, horizontal width, and font size difference. The best results are the confident score of at least 50, the horizontal width of less than 2, the vertical length of less than 0.2, and a difference in size between letters less than 2 with an F-Score of 62%. Even though the F-score is less than 70%, we found that the Min-Cost Flow method can select objects other than text detected by the Tesseract result, such as signatures and logos. Using the Min-Cost Flow, we can combine character candidates into one text line for later processing on text recognition.

Keywords: Candidate components, Certificates, Min-Cost flow, Text detection, Text line.

1. Introduction

Information extraction is a machine process to get knowledge from digital documents automatically. Inputs for information extraction can vary, including letters, bibliographies, or certificates. At present, the certificates are used in print (hardcopy) and files (softcopy) to support several things, such as the speed of delivery, security, and the wholeness of the contained information in the certificate. The certificate contains useful information that generally consists of letterhead, letter number, participant's name, time and place, and endorsement [1]. The information extraction on the certificate is preceded by detecting and recognizing the text on it. The variations of the certificate design make the detection process more difficult. The appearance of components that resemble characters also makes it not easy to detect a letter. At the end of the character detection process, the system will merge characters into text or a single text line (text line). For scanned text documents with the same text colour and horizontal text position, filtering with geometric rules is enough to combine characters into one line [2]. The certificate of the diversity of sizes and types of fonts makes geometric rules not enough to detect text lines. Some studies use the clustering approach or the search for the shortest path to form a text line [3-6].

Research by Tian [6] using the Text Flow method to detect text that appears in natural images. One of the processes used in the Text Flow method is to use the concept of path search that produces the minimum total cost. This method is the Min-Cost Flow Network. The nodes are candidate components of character, while the path with minimal cost is a text line. From this line of text, the one-word line will form. After we obtain the bounding box for the text line, it can proceed with the word recognition process. One of the advantages of using the Min-Cost Flow Network method is that it will produce a globally optimum solution. It means that we will always find a text line with a minimum cost value [7, 8].

Besides, according to Widiastuti [1], the challenge in detecting text on a certificate is a background image and a logo that can cause mistakes in the text detection process. For this reason, in this study, besides using the Min-Cost Flow Network, a smoothing process is also carried out to eliminate noise that may occur during segmentation. This study aims to evaluate the use of the Min-Cost Flow Network method to produce text lines on the certificate.

2. Research Method

We carried out several processes to detect the text on the certificate. Figure 1 provides a sequence illustration of the detection process. The process starts by changing the input image into a grey image, eliminating noise with the smoothing process, and changing the smoothing result into a binary image through segmentation. The output from the segmentation results is a candidate character. Character candidates will enter the text detection process. The detection process uses Tesseract tools. The purpose of the detection process is to select the candidate component that is the text. The trick is to determine the confidence value of each candidate component detected from the results of segmentation. We use the confidence score to filter whether the candidate character is text or not text. The process continues with the extraction of text lines. This process combines the detected component components and uses Min-Cost Flow to construct the text lines. The output of the text line is the bounding box.

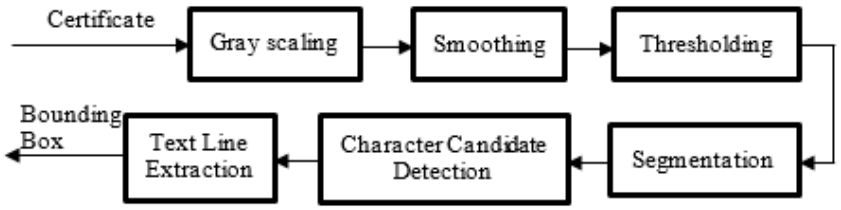

Fig. 1. Block diagram of the certificate detection process.

2.1. Data set

Input data are images in the form of certificates scan in jpg or png format. In Fig. 2, some writings appear, including the agency's text, signature, and logo. Variations of text colour, text size, the distance between letters that are not the same are the difficulties that arise when detecting text. Besides, components that resemble letters, low contrast text, and background complexity are also a challenge in making a detection. The inserted image is an image with a size of 1400×1000 pixels.

Fig. 2. The example of certificate.

2.2. Preprocessing

Figure 1 shows that grayscale, smoothing, thresholding, and segmentation are the four initial steps undertaken to prepare data to be processed. We call this stage preprocessing. The final goal of preprocessing is to get a selected candidate character component during the character detection process.

2.2.1. Grayscale

Grayscale is the process of changing an RGB image (Red, Green, Blue) to a grey colour. Gray image processing has a variety of purposes, such as: to improve the quality of the image to be detected or to extract features needed in the detection process. Converting an RGB image for each pixel using equation (1). As for the 8-bit colour depth, the intensity value (I) of grayscale images, the intensity values of red (R), green (G), and blue (B) are in the range of 0 to 255 [9]. The number 0 represents black and 255 is a grey degree value for white.

$$I = (0{,}2989 \times R) + (0{,}5870 \times G) + (0{,}1141 \times B) \quad (1)$$

2.2.2. Smoothing

Smoothing is a process to improve image quality and eliminate noise. The smoothing process uses integral image and mean-filtering proposed by Singh et al. [10]. We create the integral image matrix, $s(x,y)$ for each pixel (x,y) of the grayscale image $I(x,y)$,, with size $M \times N$ using equation (2) [11].

$$s(x,y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i,j),\ with\ x = 0,1,2,..,M-1, y = 0,1,2,..,N-1 \quad (2)$$

By using the recursive nature of integral images, we can process Eq. (2) into three steps. Eqs. (3)-(5) are equations used to replace equation (2). Figure 3 shows an example of the stages of integral image formation. Figure 3(a) is a grayscale image, then processed with Eq. (3) to produce Fig. 3(b). Figure 3(c) is the result of Eq. (4). The last result is Fig. 3(d).

$$s(0,y) = I(0,y) + s(0,y-1), y = 1, 2, .., N-1 \quad (3)$$

$$s(x,0) = I(x,0) + s(x-1,0), x = 1, 2,.., M-1 \quad (4)$$

$$s(x,y) = I(x,y) + s(x,y-1) + s(x-1,y) - s(x-1,y-1),$$
$$x = 1, 2,..,M-1, y=1, 2,.., N-1 \quad (5)$$

$I(i,j)$	0	1	2	3
0	253	253	252	253
1	254	251	252	245
2	247	240	213	224

(d) Original image.

$I(i,j)$	0	1	2	3
0	253	506	758	1011
1				
2				

(c) Result of Eq. (3).

$I(i,j)$	0	1	2	3
0	253	506	758	1011
1	507			
2	754			

(a) Result of Eq. (4).

$I(i,j)$	0	1	2	3
0	253	506	758	1011
1	507	1011	1515	2013
2	754	1498	2215	2937

(b) Result of Eq. (5).

Fig. 3. Illustration of integral image formation.

After the process obtains the integral image, the next step is filtering using the mean filter with size $n_f \times m_f$. There will be 8-neighbor pixels for a filter size 3×3. The equation (6) is a general equation to determine the mean filter $f(x,y)$for each filter $(x,y)$with the filter sizem_f.

$$mean(x,y) = \frac{f(x,y)}{n_f \times n_f},\ x = 0,1,2,\dots,M-1,\ y = 0,1,2,\dots,N-1 \quad (6)$$

where $f(x,y)$ can be calculated by Eq. (7),

$$f(x,y) = [s(x+dx-1, y+dy-1) + s(x-dx, y-dy)] - [s(x-dx, y+dy-1) + s(x+dx-1, y-dy)] \quad (7)$$

with $dx = round\left(\frac{n_f}{2}\right)$and $dy = round\left(\frac{m_f}{2}\right)$. For filter 3×3, the value $dx = dy = 1$ will be generated.

Specifically for pixels in row and column 0, row N-1 and column M-1, in this study, a pixel search technique is proposed in the filtering process that has no value

by adding an artificial function (x,y) in Eq. (6), so that it becomes like in Eq. (8).The value of function $a(x,y)$ can be calculated by Eq. (9).

$$mean(x,y) = \frac{f(x,y)}{n_f \times n_f} + a(x,y), x = 0,1,2,\dots,M-1,\ \ y = 0,1,2,\dots,N-1 \quad (8)$$

$$a(x,y) = \begin{cases} \frac{nw(x,y)}{n_f \times m_f}, & if(x - nr < 0\ or\ x + nr \geq M) \\ & or\ (y - nc < 0\ or\ y + nc \geq N) \\ 0, & otherwise \end{cases} \quad (9)$$

with $nr = trunc(\frac{wx}{2})\ dan\ nc = trunc(\frac{wy}{2})$.

where the function$nw(x,y)$can be calculated with the following Eq. (10),

$$nw(x,y) = \begin{cases} 255\left[rp(x) \times m_f + cp(y) \times \left(n_f - rp(x)\right)\right], & \begin{matrix} if\ (x - nr < 0\ and\ y - nc < 0) \\ or(x - nr < 0\ and\ y + nc \geq N) \\ or(x + nr \geq M\ and\ y - nc < 0) \\ or(x + nr \geq M\ and\ y + nc \geq N) \end{matrix} \\ 255 \times rp(x) \times m_f, & \begin{matrix} if\ (x - nr < 0\ or\ x + nr \geq M) \\ and\ (y > 0\ and\ y < N) \end{matrix} \\ 255 \times cp(x) \times n_f, & \begin{matrix} if\ (y - nc < 0\ or\ y + nc \geq N) \\ and\ (x > 0\ and\ x < M) \end{matrix} \end{cases} \quad (10)$$

Variable x is row padding, the number of rows in the mask outside the dimensions of the image is calculated based on the outer distance. Variable y is column padding, the number of columns in the mask outside the dimensions of the image is calculated based on the outer distance. The functions $rp(x)$ and $cp(y)$ are given in Eqs. (11) and (12).

$$rp(x) = \begin{cases} |x - nr|, & if\ (x - nr < 0) \\ x + nr - M + 1, & if\ (x + nr \geq M) \end{cases} \quad (11)$$

$$cp(y) = \begin{cases} |y - nc|, & if(y - nc < 0) \\ y + nc - N + 1, & if(y + nc \geq N) \end{cases} \quad (12)$$

2.2.3. Thresholding

The thresholding process uses the binary thresholding method, namely grayscale images processing into binary images (black and white). In this study, the grayscale image used is the image of the smoothing result. We use the mean value of the maximum grey intensity value, which is 128. The use of 128 is not mandatory. The research [2] is using different threshold values. Moallem [12] uses Particle Swarm Optimization to determine adaptive threshold values. Equation (13) shows the following algorithm. The process is carried out for each pixel, starting from (0,0) until all the image smoothing pixels.

$$if\ (gray > 128)\ then\ val = 0\ else\ val = 0 \quad (13)$$

2.2.4. Segmentation

The segmentation process aims to separate objects, in this case, separate candidate characters with a background. The segmentation method is Connected Component Labelling (CCL) [13]. The output of this process is the objects that appear in an image. The process will validate the objects a character candidate or not. The segmentation begins with the inclusion of a binary image matrix resulting in using binary thresholding. The system will count each coordinate to find the intensity value of one (white). If the system found the value of one (1), then the system will store the

coordinates (x, y)into a variable and change the value of the coordinates of the matrix $b(x, y)$to (0). Variables are used to hold the number one (1) in the form of a matrix named $pt(i, j)$, where i is an index of objects found with an initial value of zero (0). The process continues to find a matrix with a value of one (1) at the neighbor level with a filter 5×5 dimension. In the next, the system will count the matrix $pt(i, j)$from beginning to end to be the focal point of binary matrix data search. Following is the process at the segmentation stage using the binary matrix $b(x, y)$.

i. Define supporting variables, $pt(i, j)$ is a matrix to store object coordinates, $br(i, j)$is a matrix to store the boundaries of the coordinate object. Variable c is a 1D array to store temporary x and y coordinate values.
ii. The search starts from coordinates (0,0) to find a matrix worth one (1).
iii. Save these coordinates in the matrix $pt(i, j)$, then change the matrix value worth one (1) with a value of zero (0).
iv. Save the coordinate boundary of the object in the matrix $br(0, j)$. Because the variable does not currently have a value, the *x*-coordinate value will fill the upper and lower limits, and the *y*-coordinate will fill the left and right boundaries.
v. Next, find the pixel value of the neighboring matrix with a central coordinate
vi. (3.3) with a 5×5 dimension mask.
vii. Save the pixel coordinates of one (1) into the matrix $pt(i, j)$, then initialize the values in the matrix coordinates of one (1) with a value of zero (0).
viii. Use coordinates with a value of one in the neighbor matrix, initialize the x - coordinate in the C_0 variable, and *y*-coordinate in the C_1variable. Next, the system compares the values in the matrix $br(0, j)$from the previous data with coordinates of one value in the neighboring matrix. This process obtains the outer boundary value for an object.
ix. Repeat the process in steps number 5 to 7 using the next coordinate in the matrix $pt(0, j)$until there are no more coordinates that can be processed.

2.3. Character candidate detection

We use Tesseract to select character candidates. The output of this process is the confidence value of each character. The system will use this value to select objects as characters or not characters. Figure 4 is an example of the Tesseract output. We can find a more detailed discussion about the Tesseract in Smith's paper [14].

Fig. 4. The segmentation and character detection result.

2.4. Text line detection

The next process is combining the candidate characters found in Fig. 4 into text lines. The method used is the Text Flow method proposed by Tian [6], which begins by forming a network of detected character components. The system will process the character candidates detected by the segmentation process with the Tesseract before entering the Min-Cost Flow Network stage. If the Tesseract result states that a candidate character is a component character, then the system will mark the component character with green boxes and mark the component character with red boxes, as shown in Fig. 4. Furthermore, forming a line of text only processes the green box to construct a flow network. Min-cost flow is one of the problems in optimization. The purpose of finding solutions on the min-cost flow network is to determine the shortest path that produces the minimum total cost. The nodes are character components, while the connecting lines are possibilities of text lines. The optimal lines are representations of formed text lines. To solve the min-cost flow case, we adopted the technique proposed by Tian [6].

Before forming the text line, first, check the input image. Input image checking aims to find out whether the image has more than one line of candidate text or not. The system evaluates the segmented characters based on the y-axis. Figure 5 is an example of line checking.

x/y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
4	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0
5	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0
6	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0
7	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0
8	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0
9	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0
10	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0
11	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0

Fig. 5. Candidate character candidates I as A (red box) and candidate character candidates N as B (yellow box).

From Fig. 5, suppose that A is a red square, B is a yellow square. The y coordinate of A is $A.y = 3$, the height of A is $A.h = 9$. For B, the y coordinates and character height are $B.y = 3$ and $B.h = 9$, respectively. Figure 6 is an algorithm for checking the position of characters from the input image. If this function produces true, then the two objects entered the function will be considered as one line. Otherwise, the system will consider that component as a different line.

We assume that all lines of text start from left to right. The system will sort all candidate characters to get their horizontal coordinates. Obtained candidate's characters get from the previous result of segmentation. Figure 7 are designing a min-cost flow network. Variables W_A, W_B are the widths of component A and B. H_A, and H_B are the height of components A and B. Variable $H(A,B)$ is the horizontal distance between components A and B. Variable $V(A,B)$is the vertical distance between components A and B. In the example in Fig. 7, the value of W$W_A = 5, W_B = 9, H_A = 9, H_B = 9, H(A,B) = 1$. Because of the location of both components in a parallel position, then the value of $V(A,B) = 0$.

Several limits of each candidate character A to be associated with the next candidate character B. The goal is to reduce errors in calculating the distance between characters A and B. Table 1 shows all conditions [6]. After checking and confirming the rules, we can interpret that the two characters A and B, are neighbors. The parameters T_H, T_V and T_S are the respective threshold values for vertical, horizontal, and letter size distances.

Input :A, B
Output :True or False
Function :
initialize $boxA$, $boxB$
if $(A.y>B.y)$ then

$$boxA = B, boxB = A$$

else

$$boxA = A, boxB = B$$

if $(boxB.y \geq boxA.y \; and \; boxB.y \leq (boxA.y + boxA.h))or$
$((boxB.y + boxB.h) \geq box\ A.y \; and \; ((boxB.y + boxB.h) \leq (boxA.y + boxA.h))$ then
    return True
Else
    return False

Fig. 6. The algorithm for checking the position of two components.

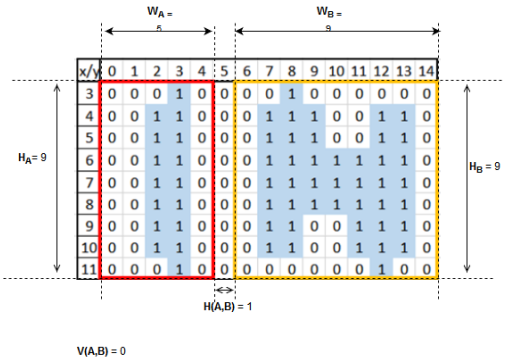

Fig. 7. Determining the parameter values of two character components.

Table 1. Rules and requirement.

No.	Rules	Requirement
1.	Horizontal distance between A and B	$\frac{H(A,B)}{\min(W_A, W_b)} < T_H$
2.	Vertical distance between A and B	$\frac{V(A,B)}{\min(W_A, W_b)} < T_V$
3.	Different size of A and B	$\frac{\lvert W_A - W_b \rvert}{\min(W_A, W_b)} < T_S$

3. Results and Discussion

The measurement of accuracy uses three test scenarios. Table 2 is a detailed measurement of the recall, precision, and f-score. Four parameters must be determined: the threshold of the confidence score, the threshold value of the vertical distance, the threshold value of horizontal distance, and the letter size's boundary-value. The choice of confidence score will affect the successful detection of character components. The system can detect more candidate characters. Then we make

changes in the confidence value ranging from 50 percent to 65 percent. The parameter of T_H and the parameter of T_V uses values from Tian's study [6]. For T_S values need to be adjusted again due to differences in image size and input image type.

Table 2. The accuracy results of each combination of threshold.

Scenario	Confidence Score, T_H, T_V dan T_S	Recall	Precision	F-score
1	(65, 2, 0.6, 0.2)	6	7	6.6
2	(60, 2, 0.6, 1)	22.4	30	25
3	(50, 2, 0.6, 2)	67	58	62

*Confidence Score, Recall, Precision, F-score(%)

Based on Table 2, the best parameter combination is the result of the 3rd test. The recall value is 67%, the precision value is 58%, and F-score is 62%. The combination of parameters used is the threshold value of 128 with a min confidence score of 50, threshold of horizontal distance (T_H) 2, vertical distance (T_V) 0.2, and the magnitude between letters (T_S) of 2. The effect of using a low confidence value produces a high recall value. These results are consistent with research by Chaithanya et al. [15], which states that using low confidence values will result in better text detection. However, decreasing the confidence value will impact the decrease in precision. The reason is that there will be more candidate components that must be selected. Increasing the number of character candidates will make Tesseract misclassify detect some candidates. We determined the confidence score threshold based on experimental results. Several other studies [16] used the Naïve Bayes model to obtain confidence values and select candidates based on this value. In addition, Lee et a. [17] and Zhu et al. [18] used Adaboost to get the confidence value. The use of supervised learning methods [19] requires training data to form a classification model. 20. Phangtriastu et al. [20] and Singh and Chaudhury [21] provided an overview of the features to detect characters or non-characters.

The size parameter between letters (T_S) in this study is different from the previous research, which is 0.2 [6]; we suspect the cause is due to differences in the type of letters used and the difference in size in a certificate and certificate size that is twice the size of the data used by Tian. Figure 8 is an example of the final output line of the text. The min-cos flow method can detect text lines well.

Figure 8 shows that none of the component text comes from different text lines. The findings obtained during testing are that this method can distinguish the signature and logo on the certificate.

Fig. 8. The result of text line detection.

There are some possible failures in detecting text. Some noises still present after the smoothing process, and Tesseract cannot filter them as non-text components. The other problem is that the system has difficulty detecting coincide writing so that the system will detect two words as one word. For letter objects consisting of two separate objects, the letter i (lowercase of I) and the letter j (lowercase of J) will not be recognized as one object but two objects. If there are letters not detected in the middle of the text line, then the text will be two different parts. In contrast to OCR research, which did not process negative images [2] to detect text on certificates, researchers also need to detect negative images so that white writing with black background images, as shown in Fig. 8, can be detected. Number plate detection [22] or text scene detection in natural images [23] are some examples that used the negative image to detect text.

4. Conclusion

Based on the results, we get the best accuracy for recall of 67%, precision of 58%, and f-score of 62%. The solution of min-cost flow is successfully detecting text lines, distinguishing logos, and separating signatures from the background. For further development, we are going to improve the stages of the preprocessing and character detection process. Adaptive thresholding process and make the machine learning system to detect component character into two choices for further research development. There is a difference in the type of letter and certificate size, so it is necessary to consider determining adaptive $T_H, T_V,$ and T_S values so that it is resistant to diverse certificate data input.

References

1. Widiastuti, N.I.; and Dewi, K.E. (2020). Document image extraction system design. *IOP Conference Series: Materials Science and Engineering*, Bandung, Indonesia, 879(1), 1-7.
2. Fernández-Caballero, A.; López Bonal, M. T.; and Castillo, J. C. (2012). Display text segmentation after learning best-fitted OCR binarization parameters. *Expert Systems with Applications*, 39(4), 4032-4043.
3. Yin, X. C.; Pei, W. Y.; Zhang, J.; and Hao, H. W. (2015). Multi-orientation scene text detection with adaptive clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9), 1930-1937.
4. Yin, F.; and Liu, C. L. (2009). Handwritten Chinese text line segmentation by clustering with distance metric learning. *Pattern Recognition*, 42(12), 3146-3157.
5. Garz, A.; Fischer, A.; Sablatnig, R.; and Bunke, H. (2012). Binarization-free text line segmentation for historical documents based on interest point clustering. *Proceeding of 10th IAPR International Workshop on Document Analysis Systems*, Queensland, Australia, 95-99.
6. Tian, S.; Pan, Y.; Huang, C.; Lu, S.; Yu, K.; and Tan, C.L. (2015). Text flow: aunified text detection system in natural scene images. *Proceeding of IEEE International Conference on Computer Vision*, Santiago, Chile, 4651-4659.
7. Ahuja, R.K.; Orlin, J.B.; and Magnanti, T.L. (1993). *Network flows: theory, algorithms, and applications.* New Jersey: Prentice-Hall.
8. Goldberg, A.V. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms,* 22(1), 1-29.

9. Gonzalez, R.C.; and Woods R.E. (2018). *Digital image processing, vol. 4th edition*. New York: Pearson.
10. Singh, T.; Roy, S.; Singh,O.; Sinam, T.; and Singh, K. (2011). A new local adaptive thresholding technique in binarization. *International Journal of Computer Science,* 8(6)2, 271-277.
11. Szeliski, R.(2010). *Computer vision*. London: Springer.
12. Moallem, P.; and Razmjooy, N. (2012). Optimal threshold computing in automatic image thresholding using adaptive particle swarm optimization. *Journal of Applied Research and Technology*, 10(5), 703-712.
13. Fu, K.S.; and Mui, J.K. (1981). A survey on image segmentation. *Pattern Recognition,* 13(1), 3-16.
14. Smith, R. (2007). An overview of the tesseract ocr engine. *Proceeding of Ninth International Conference on Document Analysis and Recognition (ICDAR)*, Parana, Argentina, 629-633.
15. Chaithany, C.P.; Manohar, N.; and Issac, A. B. (2019) Automatic text detection and classification in natural images. *International Journal of Recent Technology and Engineering,* 7(5S3), 176-180.
16. Rainarli, E. (2020). Maximally stable extremal regions and naïve bayes to detect scene text. *IOP Conference Series: Materials Science and Engineering*, Bandung, Indonesia, 879(1), 1-6.
17. Lee, J. J.; Lee, P. H.; Lee, S. W.; Yuille, A.; and Koch, C. (2011). Adaboost for text detection in natural scene. *International Conference on Document Analysis and Recognition*, Beijing, China, 429-434.
18. Zhu, K.; Qi, F.; Jiang, R.; Xu L,; Kimachi, M.; Wu, Y.; and Aizawa, T. (2005). Using Adaboost to detect and segment characters from natural scenes. *Proceedings of Camera-Based Document Analysis and Recognition (CBDAR), ICDAR Workshop*, Seoul, Korea, 52-59.
19. Bissacco, A.; Cummins, M.; Netxer, Y.; and Neven, H. (2003). PhotoOCR: Reading Text in Uncontrolled Conditions. *IEEE International Conference on Computer Vision (ICCV),* Sydney, Australia, 785-792.
20. Phangtriastu, M. R.; Harefa, J.; and Tanoto, D. F. (2017). Comparison between neural network and support vector machine in optical character recognition. *Procedia Computer Science*, 116, 351-357.
21. Singh, K.R.; and Chaudhury, S. (2020) Classification of rice grain using wavelet decomposition: a comparative study. *Journal of Engineering Science and Technology* (*JESTEC*)*,* 15(1), 108-127.
22. Zheng, L.; and He, X. (2006). Number plate recognition based on support vector machines. *IEEE International Conference on Video and Signal Based Surveillance*, Sydney, Australia, 13.
23. Sun, L.; Huo, Q.; Jia, W.; and Chen, K. (2014). Robust text detection in natural scene images by generalized colour-enhanced contrasting extremal region and neural networks. *22nd International Conference on Pattern Recognition*, Stockholm, Sweden, 2715-2720.