Extended Character Segmentation Approach for Compressed Machine-Typed Documents

Vladan Vučković, Boban Arizanović, and Simon Le Blond

Abstract - This paper presents an analysis of threshold parameters used in a new character segmentation approach for machine-typed documents, together with efficient new image compression and decompression methods used in the real time OCR system. Provided results show that the character segmentation technique is robust to machine-typed documents from different typewriters, giving far superior results than state-of-the-art approaches.

Keywords - Image processing, OCR, Character segmentation, Machine-typed documents, Image compression.

I. Introduction

Character segmentation is a very important pre-processing stage in Optical Character Recognition (OCR) systems [1,2], and together with character recognition [3,4] has been an important subject of research for many years [5]. It should be emphasized that the difficulty of character segmentation is usually underestimated compared to the process of character recognition [6,7]. Previous work that deals with character segmentation in document images can be divided into machine-printed documents [6,8,9], where the document structure and the shape of its elements is regular, and handwritten documents where character segmentation is challenged due to irregular document structure [4,7,10]. Old machine-typed documents are of particular significance because important historical documents are often in this form [4,11,12].

Recent research of character segmentation includes all levels of this process. Analyses on image binarization parameters, used in document image pre-processing, showed that the Otsu method and other Otsu-based methods give the best results on average [12]. As a pre-processing stage of the character segmentation system, image compression and decompression are required to efficiently store the document images. Genetic algorithm based on discrete wavelet transformation information for fractal image compression was presented in [13]. A lossy image compression technique which uses singular value decomposition (SVD) and wavelet difference reduction (WDR) was proposed in [14]. Many methods for character segmentation have been proposed. A technique based on searching for connected regions in the spatial domain performed on a binary image was proposed in [15]. Another process uses the Bayes theorem for segmentation by exploiting prior knowledge, and is adapted for real time tasks [16]. Diverse methods for segmentation of handwritten documents are proposed. One technique exploits clustering in the process of segmentation [17]. To solve the problem of touching characters in handwritten documents, self-organizing maps, SVM classifiers, and Multi-Layer Perceptron are used [10,18].

This paper presents further improvements and analyses of the author's character segmentation approach, which forms part of a real time OCR system for the needs of the "Nikola Tesla Museum" in Belgrade [19-22]. The first contribution presented in this paper is evaluating pre-processing methods for document image compression and decompression, which take place after the image binarization, while the second contribution is a detailed analysis of the threshold parameters used in the character segmentation system. The results show that proposed image compression and decompression methods are better than JPEG and JPEG2000 image compression standards, giving up to 4-fold improved compression ratio. Results also show that the extended character segmentation presented is robust to variation in the typewriter, outperforming other methods in this respect also.

This paper is organized as follows: Section II offers the complete description of the proposed image compression and decompression methods, as well as analysis of threshold parameters. In Section III, a set of experimental results for image compression methods and segmentation accuracy are provided. Finally, discussion of the extended real time character segmentation method, results, and future work is given in Section IV.

II. Extended character segmentation approach

This section proposes new image compression and decompression methods used in the pre-processing stage of the character segmentation system and provides detailed analyses of threshold parameters used in the segmentation stage to control the segmentation process.

A. Image compression and decompression

Image compression/decompression allows decoupling of the character segmentation system into two independent stages. The first is a pre-processing stage with the document image compression and decompression as a final process, and the second is the document image segmentation. The most evident gain here is the ability to execute two independent system parts at different times. In this way, the document image compression and document image segmentation can be executed on different machines. Also, the previously compressed/decompressed document

Vladan Vučković and Boban Arizanović are with the Faculty of Electronic Engineering, University of Nis, Aleksandra Medvedeva 14, 18000 Nis, Serbia, E-mails: vladanvuckovic24gmail.com, bobanarizanovichotmail.com.

Simon Le Blond is with University of Bath, Department of Electronic & Electrical Engineering, Bath, United Kingdom, E-mail: S.P.leBlond@bath.ac.uk.

images can be processed using different versions of the segmentation engine. This is very important since it allows efficient testing of the segmentation engine. Finally, image compression allows efficient storing of document images which can save significant memory space.

A.I Image compression and decompression using RLE

The first proposed image compression and decompression methods employ the RLE algorithm for data compression. This approach is general and can be used for all types of binarized images, but RLE algorithm gives better compression results in the case of document images than other classes e.g. natural images. Illustration of the RLE algorithm including the coding format is given in Fig. 1.

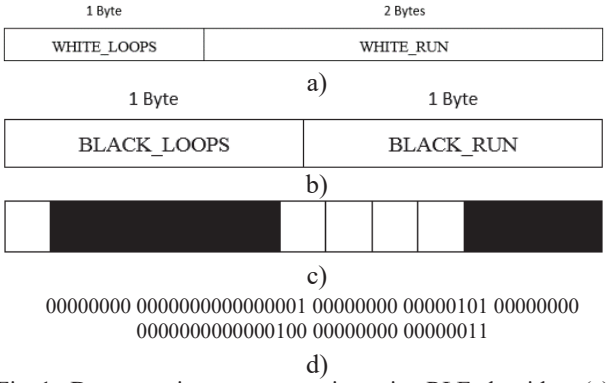

Fig. 1. Document image compression using RLE algorithm: (a) Compression format for white pixel runs, (b) Compression format for black pixel runs, (c) Example of pixel scanline, (d) Compressed pixel scanline

The RLE algorithm counts white and black pixels and stores the information about pixel runs in a compressed file. Storing is achieved using 3 bytes for all information about the white pixels and 2 bytes for information about the black pixels. In both cases the pixels are counted until the maximal value is reached. Since 2 bytes are used for white pixels (WHITE_RUN), this value is $2^{16} = 65536$, while in case of black pixels (BLACK_RUN) this value is $2^8 = 256$. When these values are reached, the WHITE_LOOPS or BLACK_LOOPS byte is incremented and WHITE_RUN and BLACK_RUN values are set to 0. The whole process of counting is then repeated. Since white pixels are a part of the background and dominate in document images, it is expected that 1 byte is not enough for storing the information about the number of consecutive white pixels. On the other hand, black pixel runs are expected to be short since they represent document characters and some spaces between characters are expected, thus only 1 byte is used for storing this information. Document image decompression is straightforward. The first byte is always multiplied by 256 for black pixels or 65536 for white pixels. This value is then incremented by the value of the next byte or 2 bytes. The obtained value represents the number of consecutive pixels of the same color in the current run of pixels. This process is repeated until the end of the compressed file.

A.II Image compression and decompression using document character contour extraction and scanline fill algorithm

The second proposed method employs the combination of the algorithm based on document character contour extraction and the scanline fill algorithm. The document image is processed in the horizontal and vertical direction and distances between the black pixels which represent starting and ending pixels of the black runs are stored in the compressed file. For this purpose, 2 bytes can be used to store the distance between two black pixels. It should be mentioned that in both compression methods the number of bytes used for storing the information about the white and black pixel runs is dependent primarily on the image dimensions. Small images are expected to have short runs, while large images are expected to have long runs of pixels of the same color. Therefore, 1 byte can be used for both white and black pixels in the case of small images, while in case of large images 2 bytes are necessary. Another important factor is a structure of a document image. If textual content dominates in a document image, background areas are less significant and thus even with large images, 1 byte can be used for storing the information about pixel runs. On the other side, if the background area dominates, even with medium images, 2 bytes are not enough to store the information about pixel runs.

After obtaining the offsets of black pixels which represent the contours of the characters, in the first step of the decompression method contours are drawn to the output image. The second step uses the iterative scanline fill algorithm. The main idea here is to scan the whole output image and fill the contours which represent the background with background color, while character contours will be filled with black color. This is achieved by repeating execution of the scanline fill algorithm. The background color of a document image is then replaced with white color and the original binarized document image is obtained.

B. Analyses of threshold parameters

This subsection provides detailed analyses of threshold parameters used in the character segmentation system. Analyses of document image binarization and line, word, and character segmentation are taken into consideration.

B.I Binarization

The binary image is obtained using a thresholding function T which is a gray-level transformation function of the form:

$$T(r) = \begin{cases} 0, & 0 \le r \le T_{hb} \\ 1, & T_{hb} < r \le r_{\max} \end{cases} \quad (1)$$

In the concrete case, value r_{max} is equal to 255. The threshold value T_{hb} that gives the best results is influenced primarily by the quality of the document image. If the document image is of low quality, a higher threshold value is required. Experimental results showed that in case of machine-typed documents the best results are achieved when T_{hb} takes values in the range 160-190. This threshold parameter affects line and word segmentation in some aspects, but character segmentation the most, since the spaces between the characters are small and the modified projection profiles technique can fail to separate the words into characters properly. In the case of images of high quality, T_{hb} can take values closer to 160, while in case of images of low quality this value must be closer to 190. The graphs which show a dependency of segmentation accuracy from the binarization threshold parameter T_{hb} are shown in Fig. 2.

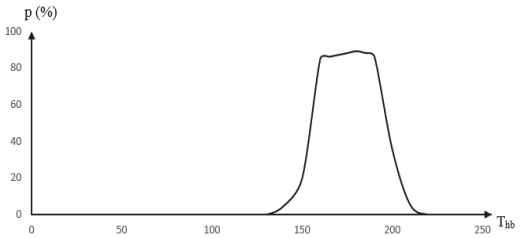

Fig. 2. Dependency of the segmentation accuracy from the binarization threshold parameter

The character segmentation system is intended for processing of document classes which contain documents typed on the same typewriter. Although it does not guarantee that documents which belong to the same class will be of the same quality, this is nevertheless likely. This means that a constant value for binarization threshold parameter can be used for different documents classes.

B.II Segmentation logic

Segmentation logic is based on a modified projection profiles technique, which uses histogram processing. A sliding window is used for calculation of the concentration of black pixels in the area of interest. Document line segmentation exploits the sliding window based method with vertical sliding. Suppose that the sliding window is size of N x H, where N is the width of the binary image f and H is the height of the sliding window. The concentration of black pixels in the sliding window is calculated as:

$$s_n = \sum_{s=0}^{H-1} \sum_{y=0}^{N-1} f(x+s, y) \tag{2}$$

Values s_n represent the values in the array of all sliding window concentrations of black pixels, S. The next condition is used for making a decision about which offsets will be taken as potential delimiters between lines:

$$d_x = \begin{cases} x + \left\lfloor \frac{H}{2} \right\rfloor, & s_n < T_{hswl} \\ -1, & otherwise \end{cases} \tag{3}$$

where d_x represents the offset relative to the top of the image and $\lfloor H/2 \rfloor$ represents integer division. Values d_x represent offsets of the middle lines of the windows which slide vertically along the area of interest. Only the offsets which belong to the sliding window with the concentration of black pixels lower than the chosen threshold value are considered. Using the offset analysis and considering the fact that the document line height is greater than d pixels, the closest offsets on distance greater than d are taken. Value H is the height of the sliding window which slides from top to bottom of the image. In general, the goal is to find local minima which represent spaces between the document lines. If value greater than 1 is chosen, it will lead to worse results, since multiple pixel scanlines are considered. In that case, the bigger value for threshold parameter T_{hswl} is necessary due to higher average concentration of black pixels in the sliding window. This would lead to unpredictable behavior and would be impossible to separate document lines that are very close to each other. Therefore, the sliding window height equal to 1 is the best choice and makes the process of document line segmentation easier to control. The value of the threshold parameter T_{hswl} depends on the document image structure and document image quality. The best choice is the smallest possible value which will separate correctly and completely the document image in lines. Even if this is achieved, it is possible and expected that other separators which cut the document lines will be found, too. For this reason, the previously mentioned value d is used to eliminate the wrong separators. This value is constant for documents inside the same document class and approximately equal to the document image character height.

The word segmentation process uses the sliding window based method with horizontal sliding. Suppose that a sliding window size of W x H is taken, where W is the width of the sliding window and $H = L_{cl} - L_{cu}$ represents the height of the current line. The concentration of black pixels in the sliding window is calculated as:

$$h_n = \sum_{x=L_{cu}}^{L_{cl}} \sum_{t=0}^{W-1} f(x, y+t) \tag{4}$$

where L_{cl} is the offset of the lower border and L_{cu} is the offset of the upper border of the given document line. The histogram is computed for each document line using the values h_n. The essence of the histogram analysis is in determination of the histogram valleys which represent the minima of the concentration of black pixels in the given line. The local minima less than the chosen threshold value are taken and these values represent delimiters between the words. Taking the word segmentation into consideration,

the sliding window width W and the threshold value T_{hsww} should be analyzed. Since the sliding window height is fixed and determined by the previously obtained document line height, the sliding window width controls the width of the spaces that are searched inside the document line. The value used for sliding window width ideally should be around the average width of spaces between the words. An alternative solution is to choose the value higher than average space between the characters inside the word. This will raise the possibility that only the spaces between the words are detected. This value also depends on document image structure, quality, and dimensions. The threshold parameter T_{hsww} has a role to eliminate the potentially wrongly detected spaces between the characters. Choosing the low value for T_{hsww} will ensure only the spaces between the words are detected, thus the latter choice is more suitable. The graph in Fig. 3 shows the dependency of the word segmentation accuracy from the sliding window width.

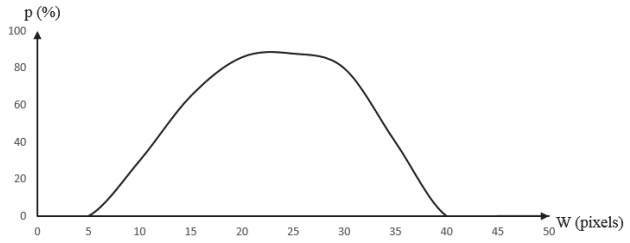

Fig. 3. Dependency of the word segmentation accuracy from the sliding window width

The approach used for character segmentation consists of the word alignment process, the already described histogram based method, and decision-making logic. The word alignment process represents the correction of dislocated words that appear due to the use of old typing machines. Sliding window height is again equal to the previously obtained document line height, while sliding window width depends on various factors. In case that a document image is of good quality, it is possible to perform quality image filtering and remove undesirable noise. But, in case of document images of low quality, even the best possible choice of the value for the binarization threshold parameter would not be helpful. The reason for this lies in sensitivity of the projection profiles technique used in the first part of the character segmentation process. Document images of low quality will have missing pixels in some part of the word and this area will be detected as a space between the characters. Also, the additional pixels in the areas which represent spaces between the characters will prevent the algorithm to detect them as spaces. A sliding window of generally smaller width than that used for word segmentation should be chosen for the first part of character segmentation. The value for sliding window width should approximately correspond to the average space between the characters. It depends primarily on document image dimensions. The threshold value T_{hswc} used to choose the potential delimiters within a word should be low enough to avoid detecting the wrong delimiters, and is strongly dependent on document image structure, quality, and dimensions.

The second part of the character segmentation is more specific and represents a decision-making logic. After determination of the local minima, which represent potential delimiters between characters, the threshold value for document character width T_{hcw} is used for determining the word length. The average character width is calculated as follows:

$$C_n = \left\lfloor \frac{W_w}{T_{hcw}} \right\rfloor \quad (5)$$

$$C_{wavg} = \left\lfloor \frac{W_w}{C_n} \right\rfloor \quad (6)$$

where C_n is the assumed number of characters in a word calculated using the threshold value T_{hcw}, W_w is the width of the given word in pixels, and C_{wavg} represents the average character width used in further processing. The word length is used to determine the number of delimiters in a word. Afterwards, the average character width can be calculated using the assumed word length, which is calculated using the threshold value T_{hcw}. The threshold value T_{hcw} is dependent on document image dimensions. This value ideally should be equal to the average character width, but even the slightly lower or higher value would be suitable. This deviation also depends on document image dimensions. The larger document images allow the bigger deviation and smaller document images allow smaller deviation values.

The segmentation of the given word starts from the left border, and delimiters at distance equal to the determined average character width are taken as the referent delimiters. The crucial part in character segmentation is the choice of the correct delimiter from the potential delimiters. The algorithm goal is to find the potential delimiter that is closest to the referent delimiter, where the maximal distance between the referent and potential delimiter is defined by the maximal offset allowed. In the case where there are no potential delimiters within the allowed distance, the referent delimiter will be chosen to be the real delimiter and the next referent delimiter will be set on a distance equal to the average character width from the chosen delimiter. Suppose that d_1, d_2,…,d_n is the sequence of potential delimiter offsets, where each of them is calculated as:

$$d_i = h_v + \left\lfloor \frac{W}{2} \right\rfloor \quad (7)$$

where h_v represents the given histogram valley and W is the width of the sliding window. The offset of the chosen delimiter is determined as follows:

$$j = \underset{i}{argmin}\left(\left|d_i - d_{ref}\right|\right) \quad (8)$$

$$d = \begin{cases} d_j, & \left|d_j - d_{ref}\right| \le T_{hoffset} \\ d_{ref}, & \left|d_j - d_{ref}\right| > T_{hoffset} \end{cases} \quad (9)$$

where j is the index of the chosen delimiter in the set of all potential delimiters, d_{ref} is the referent delimiter, and $T_{hoffset}$ is the threshold value for the maximal allowed distance between the closest potential delimiter and the referent delimiter. The threshold value $T_{hoffset}$ also primarily depends on document image dimensions. This value represents a deviation from the assumed delimiter position d_{ref}. In any case, this deviation is smaller than determined average character width C_{wavg}.

IV. Experiments

The proposed image compression and decompression methods, as a part of a character segmentation system, are tested on several PC machines. These methods are evaluated from the perspective of the image compression ratio and time complexity, to the perspective of the segmentation accuracy when specific compression methods are used. The second batch of experiments show the adaptability of the character segmentation approach for different classes of machine-typed documents. Documents typed on different typewriters require different threshold values, and by choosing the correct values for the thresholds, it is possible to achieve the high segmentation accuracy for all classes of documents. In order to obtain comparative results, JPEG and JPEG2000 image compression standards are used. Image compression using JPEG and JPEG2000 is performed using low, medium, and high quality compression, including lossless compression in the case of JPEG2000 compression.

The most important metric for evaluating the compression methods is compression ratio. Since the proposed image compression and decompression methods will be used in character segmentation system, their performances specifically on document images should be analyzed. To perform this analysis, image compression methods are tested using two document images. These document images are machine-printed documents since the second method is limited to machine-printed documents which have regular structure, thus character contour extraction is easier. Compression ratio results for these document images and different image compression methods are shown in Table I.

TABLE I

COMPARISON OF THE IMAGE COMPRESSION RATIO FOR MACHINE-PRINTED DOCUMENT IMAGES FOR DIFFERENT IMAGE COMPRESSION METHODS

Image Dimensions	Image File Size (KB)	Compression Ratio								
		JPEG			JPEG2000				**RLE**	**Contour Extraction/Scanline Fill**
		Low	Medium	High	Low	Medium	High	Lossless		
719x328	692	19.771	15.727	9.479	24.714	16.878	11.533	14.417	**67.184**	**46.443**
1266x924	3429	14.169	9.605	5.164	36.095	13.138	6.671	9.741	**27.878**	**19.373**

The proposed methods perform very well on document images based on results from Table I. This conclusion is expected since the presented algorithms perform better when the image contains huge areas of the same color, as in the case of document images. The RLE based method gives the best results, while the contour extraction method gives the second best results on average. To justify the usage of the second image compression method, the segmentation results for document images previously compressed and decompressed using different methods are given in Table II.

TABLE II

COMPARISON OF THE SEGMENTATION ACCURACY RESULTS FOR DIFFERENT IMAGE COMPRESSION METHODS USED IN THE PRE-PROCESSING STAGE

	Segmentation Accuracy (%)			
	JPEG	JPEG2000	**RLE**	**Contour Extraction/Scanline Fill**
Line Segmentation	81.54	81.54	**81.54**	**80.32**
Word Segmentation	78.28	78.28	**78.28**	**78.14**
Character Segmentation	87.08	87.08	**87.08**	**86.92**

The medium or high quality JPEG and JPEG2000 compression is shown since it behaves the same way as lossless compression after the additional binarization. Therefore, the results for JPEG, JPEG2000, and RLE based compression are identical. The most important conclusion here is that contour extraction based compression in combination with scanline fill decompression gives slightly worse results than previous compression methods. The reason for this lies in sensitivity of the evaluation metrics and also in the specificity of the character segmentation technique. In general, this technique is not sensitive to small changes in document image structure and therefore the segmentation accuracy results are similar to those obtained using the lossless compression methods.

Finally, a very important aspect of the image compression methods is time complexity since they are intended for a real time character segmentation system. In order to provide reliable results, the proposed image compression and decompression methods are tested on several PC machines and results are shown in Tables IV and V.

TABLE IV

PROCESSING TIME FOR RLE BASED COMPRESSION AND DECOMPRESSION METHOD (AMD ATHLON™ X4 840 QUAD CORE PROCESSOR 3.1 GHZ)

Image dimensions (pixels)	White Pixels/Black Pixels (%)	Processing Time (ms)			
		RLE Compression		RLE Decompression	
719x328	93.09:6.91	0.26639	0.30149	0.23231	0.27278
1266x924	83.51:16.49	2.04907	2.59600	1.94450	3.05122
2632x3575	98.06:1.94	10.61898	15.07239	13.35769	21.82402
2640x3612	98.69:1.31	10.79404	16.99354	13.31302	21.95595

TABLE V

PROCESSING TIME FOR CONTOUR EXTRACTION BASED COMPRESSION METHOD AND SCANLINE FILL DECOMPRESSION METHOD (DOCUMENT IMAGE SIZE OF 719X328)

PC Machine Specification	Processing Time (ms)			
	Contour Compression		Scanline Fill Decompression	
AMD Athlon™ X4 840 Quad Core Processor 3.1 GHz	0.26672	0.29824	1.59406	2.06575
Intel® Core™ i3-4150 CPU @ 3.50GHz	0.50409	0.52009	1.16008	1.18575
Intel® Core™ i5-750 CPU @ 2.67GHz	1.05348	1.06509	1.94235	1.95962

Time complexity results are obtained after 10000 executions of algorithms implementations. The first method also achieves excellent results in the case of document images with more black pixels which is a characteristic of documents with greater textual content. The second method processing time is primarily affected by the number of closed contours which represent character borders and must be filled using the scanline fill algorithm. Each time the closed contour must be filled, the scanline fill algorithm for region filling must be executed. Since the document image used for obtaining the results in Table V has huge areas of the same color and a small number of characters, the second method proved to be very efficient.

The second part of the experimental section is focused on the adaptability of the character segmentation system. In order to evidence this property, documents are divided into 12 classes, where each class represents a set of documents typed on the same typewriter. Therefore, 12 combinations of threshold values that give the best results for corresponding classes are chosen, and segmentation accuracy results when these combinations are used for processing are provided. The results for all levels of segmentation are shown in Table VI. Results rarely go below 80%. Text line segmentation results are above 90% in cases when spaces between document lines are bigger and when it is easy to determine the position of the lines. It is expected that text line segmentation results are good for all document classes, where spaces between lines are clear,

otherwise threshold values should be chosen carefully. Another important feature are threshold values for word segmentation and the determination of the average character width.

TABLE VI

SEGMENTATION RESULTS FOR CHARACTER SEGMENTATION APPROACH FOR DIFFERENT CLASSES OF MACHINE-TYPED DOCUMENTS

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
1	**88.62**	85.23	85.84	85.76	82.85	86.61	86.63	88.42	85.98	88.27	87.92	86.59
	87.16	87.34	86.97	84.67	84.62	85.43	87.24	85.21	85.14	87.32	86.42	87.43
	85.43	84.28	82.34	81.35	83.22	81.74	84.38	82.77	81.93	82.46	82.74	81.22
2	83.47	**87.51**	85.38	83.21	84.39	85.10	86.12	89.86	86.60	87.64	86.24	88.50
	85.36	**88.35**	84.69	84.18	85.13	86.42	85.46	85.39	84.38	87.05	85.30	87.39
	83.82	**86.79**	83.17	80.54	82.68	82.38	81.18	81.74	80.27	83.41	83.97	85.21
3	84.12	85.19	**87.15**	85.63	81.18	84.17	87.79	87.31	85.23	86.36	88.73	88.18
	85.31	83.25	**88.68**	82.78	82.44	85.68	86.54	86.12	85.83	87.39	88.65	86.35
	82.08	83.10	**85.27**	80.34	80.97	83.71	83.68	84.67	82.59	81.59	85.27	81.46
4	87.14	86.13	85.19	**88.30**	85.53	83.36	85.91	88.55	87.16	85.58	87.34	87.23
	86.29	85.67	85.23	**89.17**	85.49	82.54	86.33	86.34	86.57	86.33	88.28	86.54
	84.05	83.76	84.66	**86.14**	82.69	79.66	83.14	83.46	83.24	83.42	82.73	84.42
5	82.25	83.16	84.38	83.36	**88.59**	83.23	86.71	88.33	86.79	86.18	88.14	86.55
	85.63	84.80	84.57	80.46	**86.72**	85.74	86.93	87.24	85.47	88.27	86.79	85.04
	81.18	82.22	82.31	78.27	**86.71**	80.48	85.42	82.63	82.39	82.75	83.20	80.63
6	85.15	86.88	86.20	82.42	82.83	**92.68**	86.57	87.94	87.76	89.29	86.47	87.64
	81.31	83.07	84.62	84.79	82.57	**89.73**	85.12	85.24	87.52	87.66	85.32	86.32
	82.45	82.67	81.18	79.03	78.46	**87.64**	81.14	81.72	82.40	83.63	80.95	84.45
7	86.65	85.14	85.67	84.87	82.34	84.78	**91.05**	86.70	87.11	87.75	88.74	88.24
	86.25	85.31	85.34	84.49	80.65	85.57	**92.23**	87.23	86.25	86.49	87.51	87.93
	84.12	83.45	82.85	81.43	79.38	82.41	**88.42**	82.11	83.09	81.78	84.73	85.13
8	87.54	85.12	85.49	83.22	82.84	85.64	85.25	**93.18**	86.59	86.46	87.12	87.74
	82.37	84.35	83.06	84.53	83.76	85.72	87.64	**89.62**	85.17	86.23	85.44	86.53
	81.68	82.38	82.40	79.61	80.45	83.59	81.37	**88.14**	81.26	84.62	82.56	82.26
9	86.13	86.63	84.49	83.38	84.41	84.06	87.82	86.27	**92.43**	87.33	85.27	86.79
	84.34	85.54	85.16	82.76	82.70	86.37	86.52	85.37	**89.53**	86.29	87.41	87.43
	82.17	81.33	83.76	80.92	83.27	85.21	84.13	83.28	**88.77**	84.37	80.76	84.33
10	87.63	85.22	84.73	84.24	83.31	85.32	88.40	87.26	88.61	**92.75**	88.53	87.38
	84.41	86.15	83.35	84.55	84.75	87.45	87.25	85.34	87.69	**90.55**	86.22	86.42
	78.62	80.27	81.36	81.34	81.67	86.89	84.61	82.46	84.55	**85.64**	81.94	83.77
11	85.28	84.52	83.86	82.74	82.69	87.36	87.72	88.29	85.69	89.64	**89.11**	89.34
	83.43	83.37	82.93	80.46	84.35	85.28	88.25	87.33	86.23	86.76	**90.19**	87.29
	80.29	81.20	80.32	78.17	83.91	83.77	83.64	85.75	84.36	81.23	**87.49**	84.71
12	87.52	86.91	86.48	85.44	81.11	88.52	86.57	88.54	88.18	88.14	86.31	**91.48**
	84.93	85.13	84.37	83.68	83.67	86.44	87.26	86.30	86.22	86.67	86.47	**90.45**
	81.35	82.78	82.67	81.74	79.43	85.39	83.61	82.68	82.38	82.53	82.74	**88.92**

In both cases, for determining larger spaces between words and characters, higher threshold values are required. Also, for document images with greater character width, it is necessary to use higher threshold values in the process of determination of the average character width. Experiments also showed that the threshold value used for word segmentation could even be fixed, since it is usually clear which spaces represent the spaces between words. In contrast, selection of threshold value for character width is crucial, since it is the entry point to the decision-making algorithm. The threshold value for the maximal allowed offset between the referent and the potential delimiter is also important. Based on experiments, this value is usually between 5 and 10 pixels.

IV. Conclusion

This paper presents an image compression/decompression stage of the author's existing character segmentation approach, together with analysis of the threshold parameters used in the character segmentation stage. In Section II the image compression and decompression methods are presented and analysis of threshold parameters is provided. The presented methods use the RLE data compression algorithm and document character contour extraction for image compression, and the scanline fill algorithm for document image decompression. The character segmentation method is adaptive, since it can be used for character segmentation of the documents with different characteristics. In Section III a set of experimental results is provided for image compression methods and adaptability of the character segmentation system is proved using documents from different classes. The proposed methods perform up to 4 times better than JPEG and JPEG2000 image compression standards. Results show that choosing the correct threshold values leads to high segmentation accuracy for documents of different classes. Future work will be focused on algorithm improvement and its integration into the

complete OCR system.

Acknowledgement

This paper is supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia (Project III44006-10), Mathematical Institute of Serbian Academy of Science and Arts (SANU), and The "Nikola Tesla Museum" (providing original typewritten documents of Nikola Tesla).

REFERENCES

[1] Bourbakis, N., Pereira, N., Mertoguno, S., *"Hardware design of a letter-driven OCR and document processing system"*, Journal of Network and Computer Applications, Vol. 19, No. 3, 1996, pp. 275-294.

[2] Mao, J., Mohiuddin, K. M., *"Improving OCR performance using character degradation models and boosting algorithm"*, Pattern Recognition Letters, Vol. 18, No. 11-13, 1997, pp. 1415-1419.

[3] Namane, A., Guessoum, A., Soubari, E. H., Meyrueis, P., *"CSM neural network for degraded printed character optical recognition"*, Journal of Visual Communication and Image Representation, Vol. 25, No. 5, 2014, pp. 1171-1186.

[4] Razzak, M. I., Anwar, F., Husain, S. A., Belaid, A., Sher, M., *"HMM and fuzzy logic: A hybrid approach for online Urdu script-based languages' character recognition"*, Knowledge-Based Systems, Vol. 23, No. 8, 2010, pp. 914-923.

[5] Fujisawa, H., *"Forty years of research in character and document recognition-and industrial perspective"*, Pattern Recognition, Vol. 41, No. 8, 2008, pp. 2435-2446.

[6] Lu, Y., *"Machine Printed Character Segmentation - An Overview"*, Pattern Recognition, Vol. 28, No. 1, 1995, pp. 67-80.

[7] Lu, Y., Shridhar, M., *"Character segmentation in handwritten words - An overview"*, Pattern Recognition, Vol. 29, No. 1, 1996, pp. 77-96.

[8] Min-Chul, J., Yong-Chul, S., Srihari, S. N., *"Machine Printed Character Segmentation Method Using Side Profiles"*, Proceedings of IEEE SMC '99 Conference on Systems, Man and Cybernetics, 1999.

[9] Park, H. C., Ok, S. Y., Yu, Y. J., Cho, H. G., *"A word extraction algorithm for machine-printed documents using a 3D neighborhood graph model"*, International Journal on Document Analysis and Recognition, Vol. 4, No. 2, 2001, pp. 115-130.

[10] Lacerda, E. B., Mello, C. A. B., *"Segmentation of connected handwritten digits using Self-Organizing Maps"*, Expert Systems with Applications, Vol. 40, No. 15, 2013, pp. 5867-5877.

[11] Younes, M., Abdellah, Y., *"Segmentation of Arabic Handwritten Text to Lines"*, Procedia Computer Science, International Conference on Advanced Wireless Information and Communication Technologies (AWICT 2015), Vol. 73, 2015, pp. 115-121.

[12] Gupta, M. R., Jacobson, N. P., Garcia, E. K., *"OCR binarization and image pre-processing for searching historical documents"*, Pattern Recognition, Vol. 40, No. 2, 2007, pp. 389-397.

[13] Wu, M., *"Genetic algorithm based on discrete wavelet transformation for fractal image compression"*, Journal of Visual Communication and Image Representation, Vol. 25, No. 8, 2014, pp. 1835-1841.

[14] Rufai, A. M., Anbarjafari, G., Demirel, H., *"Lossy image compression using singular value decomposition and wavelet difference reduction"*, Digital Signal Processing, Vol. 24, 2014, pp. 117-123.

[15] Zheng, Z., Zhao, J., Guo, H., Yang, L., Yu, X., Fang, W., *"Character Segmentation System Based on C# Design and Implementation"*, Procedia Engineering, International Workshop on Information and Electronics Engineering, Vol. 29, 2012, pp. 4073-4078.

[16] Grafmüller, M., Beyerer, J., *"Performance improvement of character recognition in industrial applications using prior knowledge for more reliable segmentation"*, Expert Systems with Applications, Vol. 40, No. 17, 2013, pp. 6955-6963.

[17] Venkateswarlu, N. B., Boyle, R. D., *"New segmentation techniques for document image analysis"*, Image and Vision Computing, Vol. 13, No. 7, 1995, pp. 573-583.

[18] Bae, J. H., Jung, K. C., Kim, J. W., Kim, H. J., *"Segmentation of touching characters using an MLP"*, Pattern Recognition Letters, Vol. 19, No. 8, 1998, pp. 701-709.

[19] Vučković, V., Arizanović, B., *"Efficient character segmentation approach for machine-typed documents"*, Expert Systems with Applications, Vol. 80, 2017a, pp. 210-231.

[20] Vučković, V., Arizanović, B., *"Automatic document skew pre-processor for character segmentation algorithm"*, Facta Universitatis: Electronics and Energetics, Vol. 30, No. 4, 2017b, pp. 611-625.

[21] Vučković, V., Arizanović, B., Le Blond, S., *"Ultra-fast basic geometrical transformations on linear image data structure"*, Expert Systems with Applications, Vol. 91, 2018a, pp. 322-346.

[22] Vučković, V., Arizanović, B., Le Blond, S., *" Generalized N-way iterative scanline fill algorithm for real-time applications"*, Journal of Real-Time Image Processing, Vol. 13, No. 4, 2018b, pp. 1-19.