Text Area Detection in Handwritten Documents Scanned for Further Processing

Jakub Leszek Pach[1], Artur Krupa[2], Izabella Antoniuk[2]
[1] *National Library of Poland, Warsaw, Poland*
j.pach@bn.org.pl
[2] *Institute of Information Technology*
Warsaw University of Life Sciences – SGGW, Warsaw, Poland
artur_krupa@sggw.edu.pl, izabella_antoniuk@sggw.edu.pl

Abstract. In this paper we present an approach to text area detection using binary images, Constrained Run Length Algorithm and other noise reduction methods of removing the artefacts. Text processing includes various activities, most of which are related to preparing input data for further operations in the best possible way, that will not hinder the OCR algorithms. This is especially the case when handwritten manuscripts are considered, and even more so with very old documents. We present our methodology for text area detection problem, which is capable of removing most of irrelevant objects, including elements such as page edges, stains, folds etc. At the same time the presented method can handle multi-column texts or varying line thickness. The generated mask can accurately mark the actual text area, so that the output image can be easily used in further text processing steps.

Key words: text area detection, handwritten text, machine learning, optical character recognition, text recognition.

1. Introduction

Text recognition is a very demanding and varied field of research. Depending on the type of document containing text, i.e., whether it is handwritten or printed, and in the case of handwritten documents, what period is it from, in which region it originated, etc., the methods required to obtain processed text can differ significantly. Furthermore, even before recognizing the actual text, a series of different operations need to be performed, to first optimize the data, remove different types of noise (appearing during data acquisition or occurring in scanned text) and detect the actual text area. Especially in case of handwritten documents these first preprocessing steps are extremely important, since any errors made at this stage can later result in lower accuracy of algorithms used in optical character recognition (OCR).

Processing scanned documents is not a new problem, but its importance is rapidly increasing. When it comes to printed documents, applications such as processing business documents for further use, preparing captioning for hard-hearing persons or voice readings for blind persons come to mind. This problem is even more crucial in the case

Machine GRAPHICS & VISION 29(1/4):21–31, 2020. DOI: 10.22630/MGV.2020.29.1.2 .

of older documents, and untranslated text. Processing handwritten text, especially old or damaged manuscripts, can pose major problems. At the same time, storing the processed text in digital form can speed up its translation, and ease the document circulation between different units (scanned old texts tend to be stored in high resolution while not always containing the amount of information justifying their size). Due to these and other reasons the processing of various documents, especially the handwritten and old ones, have become the objective for researchers and scientists from different fields.

When the text recognition problem for handwritten documents is approached, one of key steps is outlining the area which contains the actual writing. Especially the old documents usually contain a number of irrelevant components, like comments placed on margins, different illustrations, folds, stains, initials, etc., which, if recognized as main text, can actually hinder the quality of OCR made in the following steps. In case of finding the text area, the methodologies used can be divided into three groups: *top-down*, *bottom-up* and *hybrid*. The first approach divides a single image view into smaller parts, to later exclude margins, initials and other elements from the main text. The bottom-up approach groups sets of pixels with a homogenous structure which can be defined using such properties as ink consistency, letter spacing, or similar. Later, single letters are grouped into words, building text lines from them. Hybrid methods use both methodologies, applying the machine learning methods as well as various other computational models to better delimit the text area, and as a result also to improve the text recognition quality.

Among the existing methods, one of the bottom-up procedures divides the binary image (BI) of the processed manuscript by combining the data series encoding the background and ink, to later produce a descriptive rectangle containing only text, without margins or other additional elements [4]. With this procedure both single and double column texts can be processed. In [2] the authors find edges of tables and the margin space. The denoting of the text space is performed by a tracking script to create a curvilinear separation path between each pair of subsequent text lines, which in result leads to finding the separate text fragments. In [8] the text area is identified by first using image binarization and later separating the graph of connected components (CC) with segmentation methods. In a next step, Hough transform is applied to define each connected component and to calculate the distance vector for each graph component, resulting in designating the external edge blocks. Finally, the space found in this way is divided into single text lines by analyzing the CC centroids, which later are grouped into final text space. Another interesting method is applied to Arabic manuscripts analysis [1]. The authors use advanced feature extraction based on image fragments analysis with graph coherent components and a group of multilayer perceptrons, to achieve the highest possible accuracy in separating main text from margins. Finally, a method using Markov random fields (MRF) described in [10] is applied to the analysis of the French manuscript by Gustave Flaubert from XIXth century, to divide the text and the background [13,14].

Taking into account research in the field of finding actual text area in old manuscripts, it can be stated that there still is much space for improvement, especially when it comes to the accuracy line detection, as well as to the speed of the entire process. Our research is inspired by some of these shortcomings and strives to improve the overall text area detection accuracy, while minimizing the number of mistakes at the same time. Results of the presented procedure are meant to be used as an input in the subsequent text processing algorithms.

2. Text area detection

When it comes to text area detection there are many different methods, most of which have some common components. In our case we based our solution on the method described in [17]. The algorithm used in that work is shown in Fig. 1, while each of its key steps is described in subsections below.

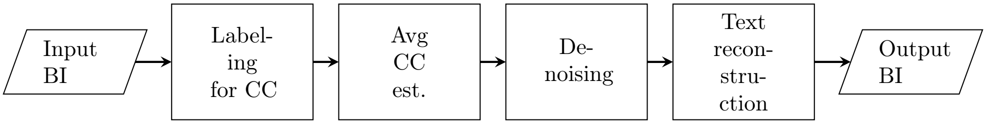


Fig. 1. Stages of the text area detection process.

2.1. Labeling

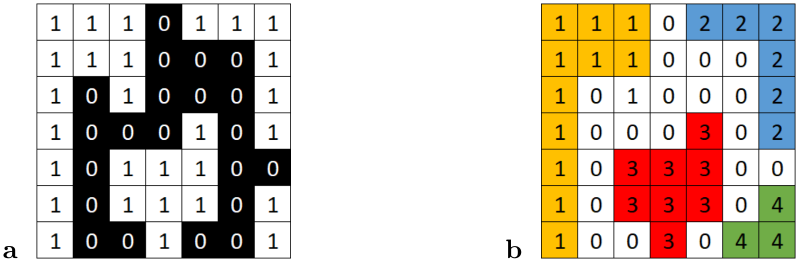
After we obtain the input binary image (see first stage in Fig. 1), we label each separate CC element in this image, since each of them might be important in the recognition process. In case of text separation we start from finding the text area, which is later divided into lines, words, and, at the final stage, into individual letters. For the examples of connected components please refer to Fig. 2.

There are three ways to describe the CC model (Fig. 3):

- based on the *bounding box* wrapped around the text [7],
- based on the *convex hull* related to the analyzed text [5],
- based on the *ellipse* wrapped around text [6].

The rectangle-based model is more than sufficient for testing purposes as far as performance and efficiency is considered, so this model was used. The binarization was implemented with the classic Otsu method [11]. Labeling the elements of the manuscript requires that the procedure makes it possible to distinguish between three types of CC:

- large elements – blocks of text, stains, folds, shadows, initials;
- medium elements – separated words, letters and fragments;
- small elements – language-specific marks like accents, diacritic elements, dots or unclassified noise.



1	1	1	0	1	1	1
1	1	1	0	0	0	1
1	0	1	0	0	0	1
1	0	0	0	1	0	1
1	0	1	1	1	0	0
1	0	1	1	1	0	1
1	0	0	1	0	0	1

1	1	1	0	2	2	2
1	1	1	0	0	0	2
1	0	1	0	0	0	2
1	0	0	0	3	0	2
1	0	3	3	3	0	0
1	0	3	3	3	0	4
1	0	0	3	0	4	4

Fig. 2. Example of CC labeling: (**a**) input binary image; (**b**) labeled output image with color labels [9].

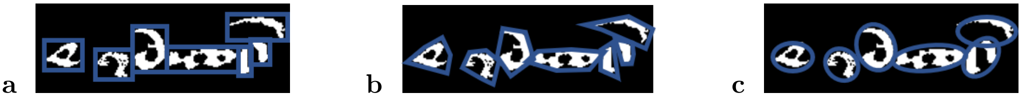


Fig. 3. CC labeling: (**a**) bounding box, after [7]; (**b**) convex hull, after [7]; (**c**) ellipse, according to [16].

2.2. The modified Constrained Run Length Algorithm for noise reduction

Binarization of an image is a process in which pixels are assigned only one of two values – 0 or 1. Within the frames of the text recognition process the value 1 (white) will represent the regions where ink is visible and 0 (black) where it is not. In the process of conversion of a text from a color image to a binary image the white color is assigned to text as well as to all irrelevant elements, like image noise, stains, folds and other unnecessary objects. To reduce such noise, we used the Constrained Run Length Algorithm (CRLA) [3, 17] which was modified to better fit our input data.

In the CRLA the Run-Length Encoding (RLE) method is used for information coding. With this method the string in the form "abbcccddddeeeee" can be defined as a counted string which contains the number of occurrences and the occurring character (e.g., ASCII character). In the case of the presented example string, after encoding it with RLE it might look like "1a2b3c4d5e". Shortening of input strings can cause less data to be transferred (here: 10 characters instead of 15). This algorithm was used for the first time in data compression methods [12, 15].

Coming back to noise removal, three important steps need to be performed:

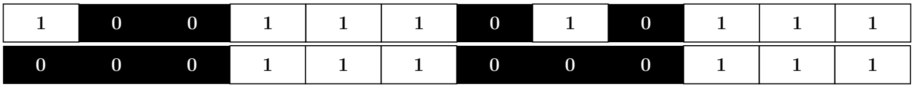
- replacing strings of bits with pairs containing a value and a correlated number of occurrences,
- defining which of the values should be replaced (setting the threshold for replacement),
- reverting the shortened string description to the original form.

The algorithm is run twice: along rows and along columns of the image, giving two images. In each direction, the strings of ones having the length less than the threshold, are replaced with zeros. In passing the image along rows, as the threshold the average width $\bar{w}$ is used, and along the columns the threshold value is the average height $\bar{h}$. The RLE is helpful, as the lengths of the strings are explicitly contained in the code. In this way, small white gaps are removed. Results for an example row of an image are presented in Fig. 4.

The two resulting images – the effect of CRLA filtering along rows and columns – are composed into one output image by performing the AND operation pixel-wise, which is equivalent to pixel-wise arithmetic product.

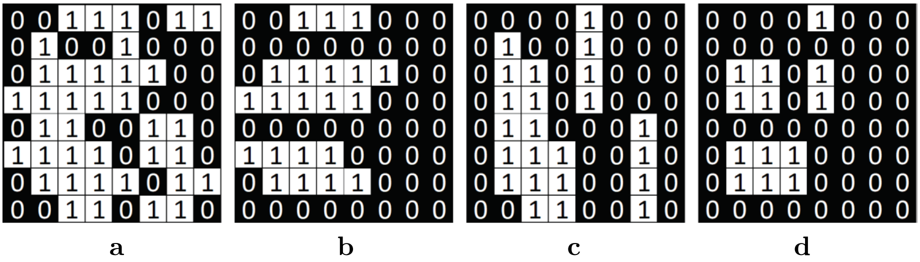
The effect of the application of the CRLA method to an input image horizontally and vertically is presented in Fig. 5. The application of CRLA to a scan of a manuscript is shown in Fig. 6, where additionally the vertical and horizontal projections of image intensities are shown.

After performing these operations, the output image contains significantly less noise. At the same time, the shapes of underlying pages and small inscriptions in the margin areas are mostly filled with zeros. Furthermore, the bottom area of the document, where some notes written with different handwriting are placed, is visible equally clearly as the main text. Also, at this point only small amount of noise still remains (i.e., remnants of the edges of underlying pages) and resulting images can be further processed.



1	0	0	1	1	1	0	1	0	1	1	1
0	0	0	1	1	1	0	0	0	1	1	1

Fig. 4. Input (top) and output (bottom) of CRLA method for an example row.



0	0	1	1	1	0	1	1
0	1	0	0	1	0	0	0
0	1	1	1	1	1	0	0
1	1	1	1	1	0	0	0
0	1	1	0	0	1	1	0
1	1	1	1	0	1	1	0
0	1	1	1	1	0	1	1
0	0	1	1	0	1	1	0

a

0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	1	1	1	1	1	0	0
1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0
0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0

b

0	0	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	0	1	0	0	0
0	1	1	0	1	0	0	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	0	1	1	0	0	1	0

c

0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	1	1	0	1	0	0	0
0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0

d

Fig. 5. Results of CRLA filtering: (**a**) input image; (**b**) filtered horizontally; (**c**) filtered vertically; (**d**) output image: pixel-wise product of images **b** and **c**.

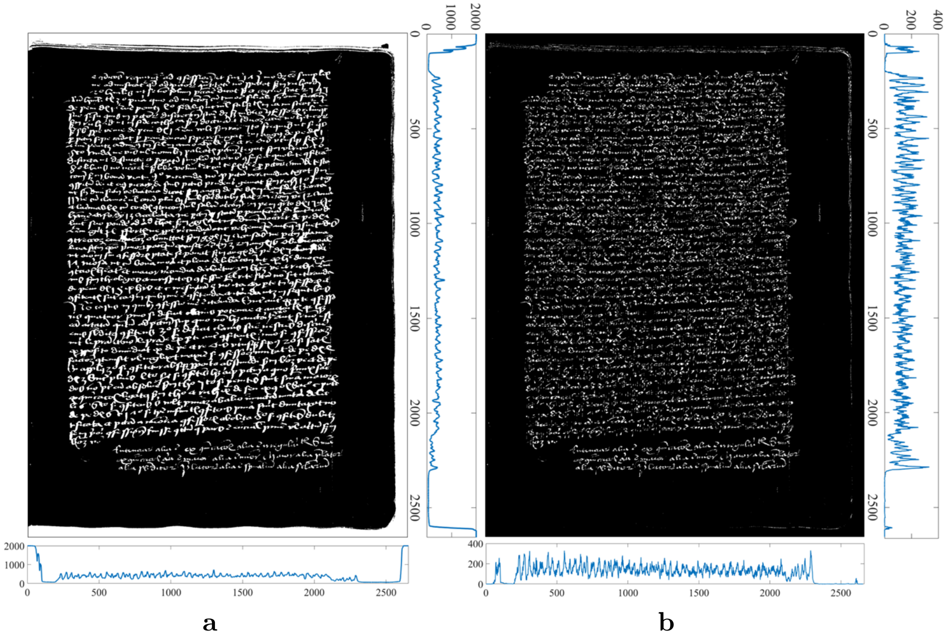


Fig. 6. Noise cancellation in a manuscript: (**a**) before and (**b**) after the CRLA filtering.

2.3. Other noise reduction methods

With the above method, the most part of high-density noise, like stains, folds and similar elements, have been removed. The next step addresses low-density noise elements, as well as unusual objects that can pose some difficulties for the OCR algorithm. Now, the masks of text regions will be generated by classifying image rows and columns as belonging to the text area or not.

To find the threshold for this classification, the average numbers of ones are calculated for each row and column in the binary image. Statistically, when it comes to historical manuscripts, ink would take up to 20% of total page area (in present day documents it would take up to 10%, since nowadays the handwriting is thinner). Therefore, every row (or column) of the image can be safely considered as belonging to text region if it contains more than $\frac{1}{3}$ of white pixels.

The two images, one resulting from classifying the rows, and one from classifying the columns, are combined into the output image by applying the pixel-wise product, as it was done in the previous algorithm. A binary image of a Latin manuscript processed with this method, containing the preliminary mask of text areas, is presented in Fig. 7.

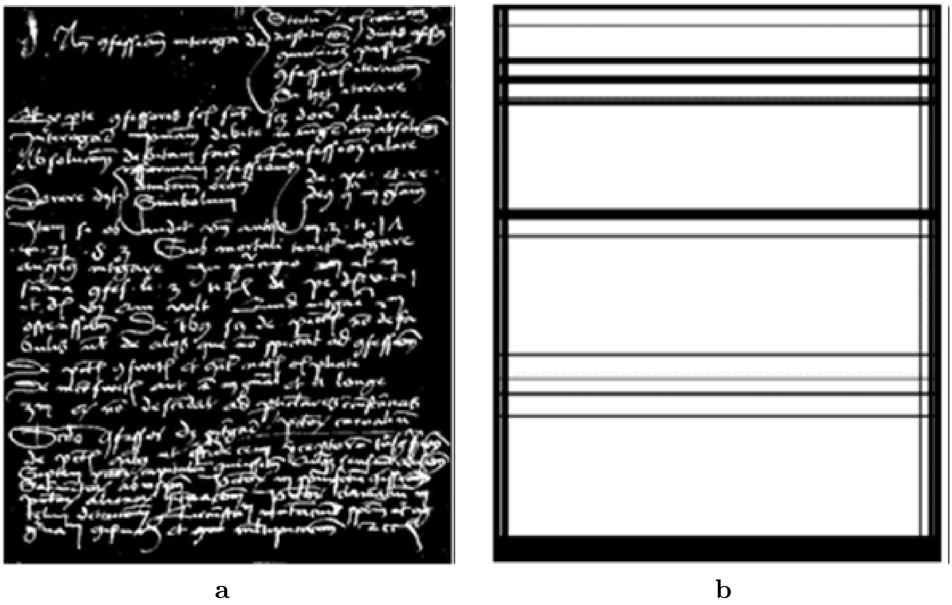


Fig. 7. Preliminary text area in a Latin manuscript: (**a**) source and (**b**) result.

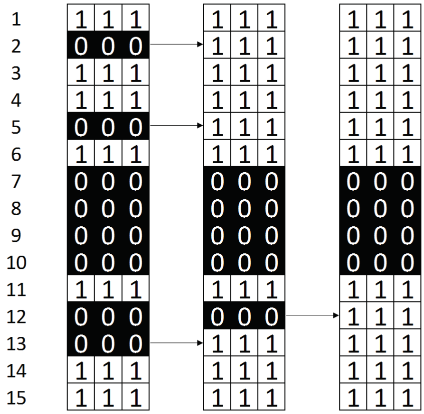
3. Image reconstruction

As mentioned before, the input data was a binary image of a manuscript that required preparation for the text recognition process. After performing the above operations, the output is a binary image without noise and without additional objects that could pose problems in further processing steps. Initials, stains, weak ink or punctures present in the original image were filtered out correctly. The final stage of processing is the reconstruction of the handwriting area. This can be compared to typical morphological opening operation. The application of this method to the processing of historical documents was described in [4]. As it was the case with the previous algorithms, as a result we get two images, each being a set of vectors: one with rows and one with columns.

To set the threshold ε for the algorithm, the average height $\bar{h}$ of a connected component in a page will be used:

$$\bar{h} = \frac{1}{n}\sum_{i=1}^{n} h_i \ , \quad \varepsilon = \frac{\bar{h}}{2} \ ,$$

where h_i is the average height of the i-th CC in a page of text, and n is number of all CCs in this page. The rationale for setting the threshold to half of the average height



1	1 1 1	1 1 1	1 1 1
2	0 0 0	1 1 1	1 1 1
3	1 1 1	1 1 1	1 1 1
4	1 1 1	1 1 1	1 1 1
5	0 0 0	1 1 1	1 1 1
6	1 1 1	1 1 1	1 1 1
7	0 0 0	0 0 0	0 0 0
8	0 0 0	0 0 0	0 0 0
9	0 0 0	0 0 0	0 0 0
10	0 0 0	0 0 0	0 0 0
11	1 1 1	1 1 1	1 1 1
12	0 0 0	0 0 0	1 1 1
13	0 0 0	1 1 1	1 1 1
14	1 1 1	1 1 1	1 1 1
15	1 1 1	1 1 1	1 1 1

Fig. 8. Three stages of image reconstruction (numbers on the left denote row indexes).

of the connected component is the observation that the height, and also the width, of a typical small letter (like letter ‘a’, for example) is equal to ε.

The reconstruction goes on according to the following steps, along rows (or columns):

1. Set the `changed` flag to `false`.
2. Set the current element i of the row (or the column) to its first element.
3. It the element i is zero, then count the elements with values one in the neighborhood $[i-\varepsilon, i-1] \cup [i+1, i+\varepsilon]$ of the i-th element (its closed ϵ-neighborhood without the element itself). If their number is greater or equal to ε then change the i-th element to one, and set the `changed` flag to `true`. Go to next element.
4. If not end of row (or column), then proceed from step 3. Otherwise, go to next row (or column).
5. If the row (or column) was the last one, then check the `changed` flag. If `false`, then stop. Otherwise, set the `change` flag to `false`, return to the first row (or column) and proceed from step 2.

The two images, one resulting from performing the above algorithm by rows, and one by columns, are combined into the output image by applying the pixel-wise product, as it was done in the previous algorithms.

Let us consider an example three-column image shown in Fig. 8, where the columns are processed. Assume that $\bar{h} = 4$, so the threshold $\varepsilon = 2$. Let us consider the first column. The first i with a zero is row $i = 2$. Its neighborhood is composed of rows 1, 3 and 4, with three values equal to one; $3 \geq \varepsilon$ so the condition for a change from zero to one is true. The condition is also true for $i = 5$ and 13, so these elements are changed to ones. The same will be done in the remaining two columns. In the next iteration

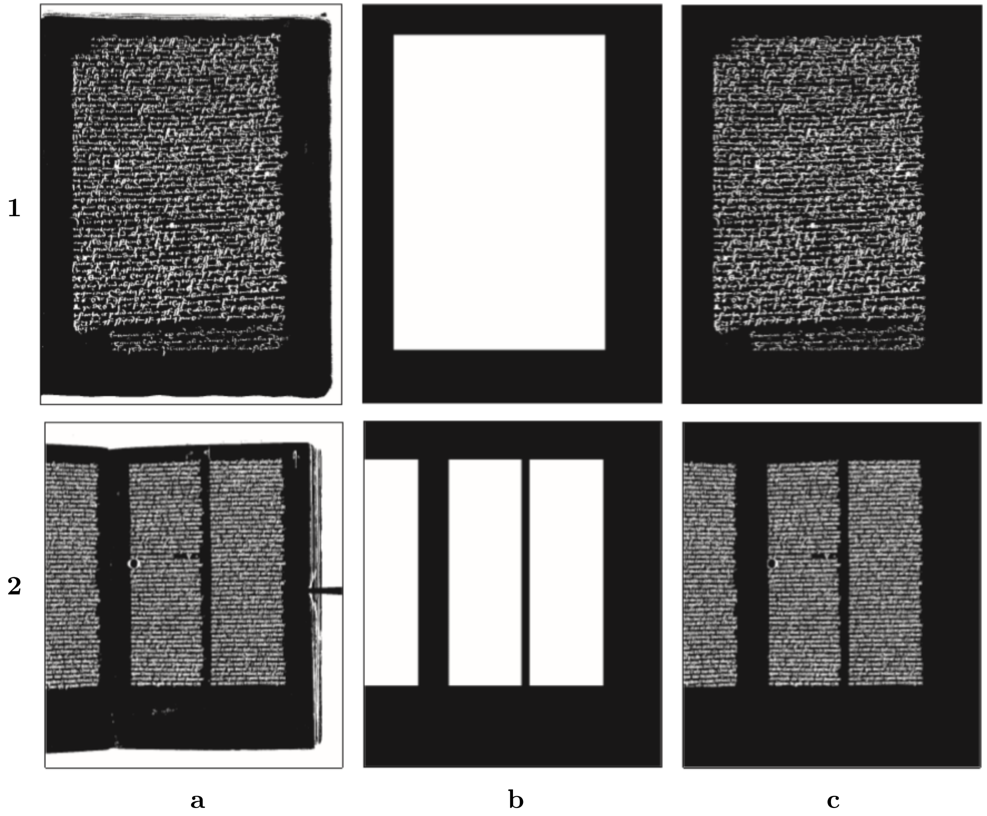


Fig. 9. Intermediate and final results of text area segmentation for two non-trivial manuscripts: (**1**) for a manuscript with differing text thickness and style, and (**2**) for text with multiple columns, uneven spaces and noise produced by page edges and other elements. Stages of processing: (**a**) image after binarization; (**b**) text mask; (**c**) final result after reconstruction.

through this image, the row 12 is changed to one. The image is analyzed one more time and there are no more rows to modify, so the algorithm stops.

Examples of results of the whole text area detection method described in this paper are shown in Fig. 9. Two non-trivial manuscripts are considered: a manuscript with differing text thickness and style, and a manuscript with text in multiple columns, with uneven spaces and noise produced by page edges and other elements. In both cases the unwanted artefacts are properly removed.

4. Conclusion

In this paper we presented a method for text area segmentation for handwritten manuscripts, which can be used as one of the preprocessing steps for further text recognition algorithms. Text recognition is a complex problem, with many difficulties, most of which depend on the type of manuscript and on the final application of the obtained results. The objectives of processing the text and of trying to understand its meaning can vary greatly, from simply storing the data in a most efficient way, up to adjusting the final content to very specific, individual needs.

Our method focuses on the first stage of this process, which is text area detection. Since the algorithms used for text recognition and analysis can be very sensitive to any noise present in the input data, it is crucial to achieve the best possible results at this step. Our method is able to accurately outline the text area, while omitting most of irrelevant elements, such as page edges, stains, folds, etc. At the same time, the presented approach can handle a large level of variety in single manuscripts. Text area can be accurately pointed out in documents with multiple columns, uneven text width, and with different objects not related to actual text. At the same time it also does not cut out the elements such as fragments of text that differ in line thickness. The obtained images are free from most of such elements like margins, spaces between columns, etc., which are irrelevant to the subsequent analysis steps. The obtained final images can be further used in text recognition algorithms.

References

[1] S. S. Bukhari, T. M. Breuel, A. Asi, and J. El-Sana. Layout analysis for Arabic historical document images using machine learning. In *Proc. 2012 Int. Conf. on Frontiers in Handwriting Recognition*, pages 639–644, Bari, Italy, 18-20 Sept. 2012. IEEE. doi:10.1109/ICFHR.2012.227.

[2] M. Bulacu, R. Van Koert, L. Schomaker, and T. van der Zant. Layout analysis of handwritten historical documents for searching the archive of the cabinet of the Dutch queen. In *Proc. 9th Int. Conf. on Document Analysis and Recognition ICDAR 2007*, volume 1, pages 357–361, Parana, Brazil, 23-26 Sept. 2007. IEEE. doi:10.1109/ICDAR.2007.4378732.

[3] B.-S. Chien, B.-S. Jeng, S.-W. Sun, G.-H. Chang, K.-H. Shyu, and C.-H. Shih. Novel block segmentation and processing for Chinese-English document. In *Proc. Visual Communications and Image Processing'91: Image Processing*, volume 1606 of *Proc. SPIE*, pages 588–598, 1 Nov. 1991. doi:10.1117/12.50377.

[4] B. Gatos, G. Louloudis, and N. Stamatopoulos. Segmentation of historical handwritten documents into text zones and text lines. In *Proc. 2014 14th Int. Conf. on Frontiers in Handwriting Recognition*, pages 464–469, Heraklion, Greece, 1-4 Sept. 2014. IEEE. doi:10.1109/ICFHR.2014.84.

[5] S. H. Kim, S. Jeong, G. S. Lee, and C. Y. Suen. Gap metrics for handwritten Korean word segmentation. *Electronics Letters*, 37(14):892–893, 2001. doi:10.1049/el:20010596.

[6] H. I. Koo and N. I. Cho. Text-line extraction in handwritten Chinese documents based on

an energy minimization framework. *IEEE Trans. on Image Processing*, 21(3):1169–1175, 2011. doi:10.1109/TIP.2011.2166972.

[7] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis. Text line and word segmentation of handwritten documents. *Pattern Recognition*, 42(12):3169–3183, 2009. doi:10.1016/j.patcog.2008.12.016.

[8] V. Malleron, V. Eglin, H. Emptoz, S. Dord-Crouslé, and P. Régnier. Text lines and snippets extraction for 19th century handwriting documents layout analysis. In *Proc. 10th Int. Conf. on Document Analysis and Recognition ICDAR 2009*, pages 1001–1005, Barcelona, Spain, 26-29 Jul. 2009. IEEE. doi:10.1109/ICDAR.2009.199.

[9] K. Mirul. Object counting using connected component labelling. In *It's Science – Blog.*, 2020. [Online; accessed 16 Jan. 2020]. `http://k-sience.blogspot.com/2017/06/object-counting-using-connected.html`.

[10] S. Nicolas, T. Paquet, and L. Heutte. Complex handwritten page segmentation using contextual models. In *Proc. 2nd Int. Conf. on Document Image Analysis for Libraries DIAL'06*, pages 46–59, Lyon, France, 27-28 Apr. 2006. IEEE. doi:10.1109/DIAL.2006.8.

[11] M. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Systems, Man and Cybernetics*, 9(1):62–66, 1979. doi:10.1109/TSMC.1979.4310076.

[12] J. L. Pach. Analysis of lossless data compression methods (in Polish). Technical report, Warsaw University of Life Sciences – SGGW, Faculty of Applied Informatics and Mathematics – WZIM, Warsaw, 2011.

[13] J. L. Pach. *Identification of the author of Latin manuscripts with the use of image processing methods* (in Polish). PhD thesis, Warsaw University of Technology, Faculty of Electronics and Information Technology, Warsaw, Poland, 2019.

[14] J. L. Pach and P. Bilski. Robust method for the text line detection and splitting of overlapping text in the Latin manuscripts. *Machine Graphics & Vision*, 23(3/4):11–22, 2014. `http://mgv.wzim.sggw.pl/MGV23.html#3-11`.

[15] D. Pountain. Run-length encoding. *Byte*, 12(6):317–319, 1987. `https://archive.org/details/byte-magazine-1987-06`.

[16] J. Ryu, H. I. Koo, and N. I. Cho. Language-independent text-line extraction algorithm for handwritten documents. *IEEE Signal Processing Letters*, 21(9):1115–1119, 2014. doi:10.1109/LSP.2014.2325940.

[17] F. M. Wahl, K. Y. Wong, and R. G. Casey. Block segmentation and text extraction in mixed text/image documents. *Computer Graphics and Image Processing*, 20(4):375–390, 1982. doi:10.1016/0146-664X(82)90059-4.