Optical Character Recognition on images with colorful background

Matteo Brisinello
RT-RK Institute for Computer Based Systems
Osijek, Croatia
Matteo.Brisinello@rt-rk.com

Ratko Grbić
University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology
Osijek, Croatia
ratko.grbic@ferit.hr

Dejan Stefanović, Robert Pečkai-Kovač
RT-RK Institute for Computer Based Systems
Novi Sad, Serbia
Dejan.Stefanovic@rt-rk.com, Robert.Peckai-Kovac@rt-rk.com

***Abstract*— In this paper, a preprocessing method is presented for improving Tesseract Optical Character Recognition (OCR) performance on images with colorful background. The proposed method consists of two steps. At first, a text segmentation method is performed which attempts to extract the text from the colorful background. This step is based on input image clustering into *k* images. In the second step, a classifier is used to identify the image containing text among *k* images resulting from the previous step. OCR is then performed on the identified image. The proposed preprocessing method improves Tesseract OCR performance by approximately 20%.**

Keywords— OCR; images with colorful background; image segmentation; image classification

I. INTRODUCTION

Optical character recognition (OCR) is a process of text recognition in digital images. This is usually performed by dedicated OCR software such as Tesseract, CuneiForm and ABBYY [1,2]. The practical OCR usage as well as the number of different OCR applications is constantly growing. Simple usage includes reading text from scanned black and white images, while more recent OCR applications include more complex tasks like reading text from ID cards, passports, vehicle registration plates, traffic signs, etc. Performing OCR on images in the mentioned situations usually requires additional image preprocessing in order to achieve correct OCR results due to several reasons. The images under inspection can have lower resolution than the minimal required for the specific OCR software. The inspected images can be of a low quality too. Low quality images usually contain some type of noise which can be removed with an appropriate filter or other image preprocessing methods. But one of the most challenging tasks when using OCR is getting correct results on images with colorful background which is the focus of this paper.

The motivation of this paper stems from the attempts of reading text by OCR software on images grabbed from set-top boxes (STBs). This is a usual procedure in STB development and testing where OCR is performed on specific part of the grabbed image, such as STB menu that is displayed to the user, and OCR result is compared with the text that should appear in that specific part of the grabbed image. Since modern STBs support displaying transparent menus, grabbed images can contain complex backgrounds which are in fact part of a live TV content. Therefore, those images are considered as images with colorful background. Straightforward application of the commonly used OCR software on those images usually ends up with the poor OCR accuracy.

To the author's knowledge, there is no OCR software specifically designed for reading text on images containing colorful background. Therefore, the main idea of this paper is to couple OCR software with the suitable preprocessing method in order to enhance OCR performance. There are many software solutions for OCR. Some of them are Tesseract, CuneiForm and ABBYY. In this work Tesseract was chosen mainly because it is free, but also it has been shown that it is one of the best open source OCR engine in general [2].

The process of reading a text with Tesseract consists of several steps. The first step is a connected component analyses in which outlines of the components are stored as blobs. Blobs are then organized into text lines which are broken into words. Recognition then proceeds as a two-pass process [1]. It is already reported that Tesseract has difficulties reading text from images with colorful background [3]. Basically, there are no specific steps included in Tesseract which could efficiently separate text from the background prior actual OCR. Therefore, in such cases Tesseract has usually very low OCR accuracy.

978-1-5386-6095-9/18/$31.00 ©2018 IEEE

In this paper, preprocessing method is proposed in order to increase Tesseract OCR performance on images with colorful background. The proposed preprocessing method is a segmentation method which is meant to be performed before using Tesseract. At first the original image is divided into *k* images of which one should contain only text, while the other images contain some parts of the colorful background. After that, an image classification is performed on the resulting *k* images in order to identify which image contains text and which should be further processed with OCR. OCR was performed using Tesseract 4.0 which was the last version at the time this paper has been written. The efficiency of the proposed preprocessing method is evaluated on a set of computer generated images with colorful background and which are very similar to those obtained during STB testing, i.e. each image contains only one line of text and the whole text is represented with a single color.

This paper is structured as follows. In section II, papers which deal with similar OCR problems are described. In Section III, an image segmentation method is presented. Section IV presents an image classification method which identifies the image containing text after the clustering method is performed. The efficiency of the proposed method is evaluated in Section V. In the end, conclusions are drawn.

II. Related Work

As mentioned in the Introduction, OCR engines have difficulty reading text from images with colorful background and none of them is designed to achieve high recognition rate on that specific task. However, there are many approaches to solve this issue. Most of them are focused on image preprocessing, resulting in an image more suitable for performing OCR on.

In [4] a method which removes background images from documents is presented. This is achieved by first enhancing the document images utilizing the brightness and chromaticity as contrast parameters. Then color images are converted to gray and approximately thresholded. Doing this, background images can be removed without affecting the quality of text characters. The problem with this method is that it works on relatively simple backgrounds.

Before performing OCR on images with colorful background, it can be useful to first identify and crop only regions of interest from original images which in this case are regions containing text. This approach reduces colorful background regions which may contain random edges in different size, thickness and orientation which are often recognized as diacritic characters. In [5] a text identification method in complex background using Support Vector Machine (SVM) is presented. The algorithm first extracts the candidate text line on the basis of edge analysis, baseline location and heuristic constraints. SVM is then used to identify text lines from the candidates in edge-based distance map feature space. A text segmentation method which relies on a rule saying that a text pixel always lies between an "edge couple" is presented in [6]. First, a rule-based sampling method is proposed to get portion of the text pixels. Then, the sampled pixels are used for training Gaussian Mixture Models of intensity and hue components in HSI color space. Finally, the trained GMMs together with the spatial connectivity information are used to extract all text pixels from their background.

Another method for separating foreground text from complex background in color document images is presented in [7]. It is achieved with a hybrid approach which combines connected component analysis and an unsupervised thresholding for separation of text from the complex background. The down side of this method is that the complexity of the backgrounds is still relatively simple compared to backgrounds on images used in this paper.

III. Image Segmentation

As mentioned in the Introduction, motivation for this work is OCR on images that are grabbed from STB devices and cropped such that they contain only one line of text presented with single color as shown in Fig. 1. By taking this into account, it can be concluded that pixels representing text occupy a significant part of the image and they are approximately of the same color. Thus, pixels representing text can be extracted using a clustering method applied to the pixel color values. In the RGB color space color information is stored in all three components. Separating pixels into clusters is more efficient using two dimensions like in the L*a*b color space than using three dimensions like in the RGB color space. Therefore, the image is converted from the RGB to the L*a*b before clustering method is performed.

In this paper, clustering is achieved using the *k*-means algorithm. In this algorithm Euclidean distance metric is used to group pixels with similar color values. When using *k*-means, the number of clusters *k* must be set before running the algorithm. There are several algorithms for calculating the optimal number of clusters. Those algorithms are usually time-consuming. Since this text segmentation method tends to be simple and time efficient, no special algorithm for computing the number of clusters was used. Considering the dataset used in this paper where the text has always unique color and occupy significant part of the image, three clusters should be the minimum number of clusters in *k*-means algorithm to clearly extract the text from the background. After applying the algorithm, one of the clusters contains pixels that represent text as shown in Fig. 2 (b). Other clusters, Fig. 2 (c) and (d) contain pixels which are part of the colorful background. Obviously, OCR should be performed only on the image generated from the cluster which contains pixels representing text.

Fig. 1. Typical image obtained in the process of STB testing.

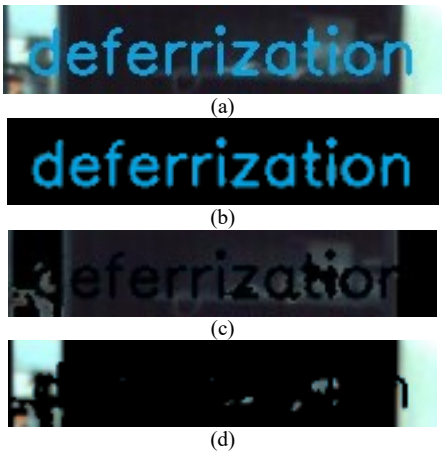

Fig. 2. Resulting images after original image clustering with k=3. Original image (a), first cluster (b), second cluster (c), third cluster (d).

IV. Image Classification

After image clustering into k clusters, i.e. original image segmentation into k resulting images, identification of the image that contains text is performed. For this task an image classification model is developed and used to classify each of k generated images as an image containing text or as an image not containing text.

The first step when building a classifier is choosing the appropriate input variables. In this case appropriate image features are extracted and used as classifier inputs. Several assumptions have been considered before defining those features. The first assumption is that the text is unicolor. This assures that after clustering, one cluster contains pixels representing text. The second assumption is that the original image contains only one line of text. The third and final assumption is that the whole text is only one word without space characters between other characters. Taking into account those three assumptions, it is assumed that features which should discriminate the image containing text from the image not containing text are extracted as follows. The first step of extracting features from an image is finding contours on the image itself. On an image containing only text, contours are in fact characters. The second step is defining bounding rectangles surrounding all contours which gives a better overview of the positions of each contour and gives the possibility of calculating the height, width and area as well as the possibility of getting coordinates of its boundaries. Bounding rectangles on an image containing text should be rectangles bounding only characters. Based on the assumption that the image contains one line of text, it can be said that the rectangles are approximately horizontally aligned on images containing text. In other words, the lower y coordinates of bounding rectangles on images containing text are "more aligned" than the lower y coordinates of bounding rectangles on images containing parts of background. The mentioned alignment can be calculated as the standard deviation of all lower y coordinate according to (1) where x_i represents each lower y coordinate, $\bar{x}$ the mean of all lower y coordinates, while N is the total number of rectangles.

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})}{N-1}} \tag{1}$$

According to equation (1), the standard deviation is expressed as a number of pixels. Images can be of different sizes, so the standard deviation expressed in this way is not the most appropriate. In order to achieve image size independence, the relative standard deviation (RSD) is calculated according to (2) where s is the standard deviation calculated according to (1) and $\bar{x}$ the mean of all lower y coordinates.

$$RSD[\%] = \frac{s}{x} * 100 \tag{2}$$

RSD expresses the deviation of all lower y coordinates expressed as percentage which assures image size independence. As this value is closer to zero, it means that lower y coordinates of bounding rectangles are less dissipated. This value is the first feature used for classifier building. The second and third features are based on the assumption that the rectangles surrounding the letters are approximately equal in area and approximately equal in height. This is also expressed by the RSD calculated according to (1) and (2) where x_i represents each rectangle area and rectangle height, $\bar{x}$ the mean of all areas and all heights, while N is the total number of rectangles. The fourth feature is based on the assumption that the image contains only one word. Based on this assumption it can be concluded that the rectangles surrounding the letters are repeated in some regular intervals on the x axis. This feature is calculated as the RSD of all distances between the rectangles. The distance is calculated as the difference between central x coordinates of adjacent rectangles.

On images which contain text, first feature should be closer to zero that on the images which contain background. In other words, the lower y coordinates of bounding rectangles surrounding characters are approximately equal. There are some exceptions such as rectangles surrounding a dot that is part of a lowercase letter "i" or "j" and rectangles surrounding lowercase letters "q", "g", "j", "y" and "p" whose lower y coordinates are lower than the other letters. Regardless of these exceptions, the RSD of the lower y coordinates should be closer to zero than those in images that contain background contours whose lower y coordinates are much more irregular.

The second and the third feature are also closer to zero as their values are less dissipated. The area of the bounding rectangle surrounding the lowercase letter "i" and the area of the bounding rectangle surrounding a dot that is a part of lowercase letters "i" and "j" are smaller than other areas, but areas of bounding rectangles on images with background are

much more irregular, so their RSD should be greater than the RSD on images with text. This also applies to the height of the rectangles. An example of the image containing text is shown in Fig. 3 where values from which features are calculated are marked in yellow color. After the features have been defined, it is necessary to choose a classifier type which will be used for image classification. In this paper, several standard machine learning classifiers were examined. In the end, naïve Bayes classifier was chosen due to its simplicity and satisfactory accuracy. The built classifier predicts a class for each image that is obtained in the process of image segmentation by a clustering technique, i.e. outputs whether image contains text or not. Apart from that, classifier outputs the probability that image contains text. After obtaining classes and its probabilities for each image, only the image with the highest probability of containing text is passed to Tesseract for OCR as shown in Fig. 4.

V. Results

In order to evaluate the proposed method, the set of images were generated. The generated images should be very similar to the images that are grabbed from STB during testing. Therefore, these images were generated from a set of 10000 photos of different content which were cropped at random positions. These cropped images were used as colorful background. To each cropped image a random text with a random color is added. Since the text color is randomly generated, there are situations where the generated color is approximately the same as the background color resulting in the image where the text is not readable even with a human eye.

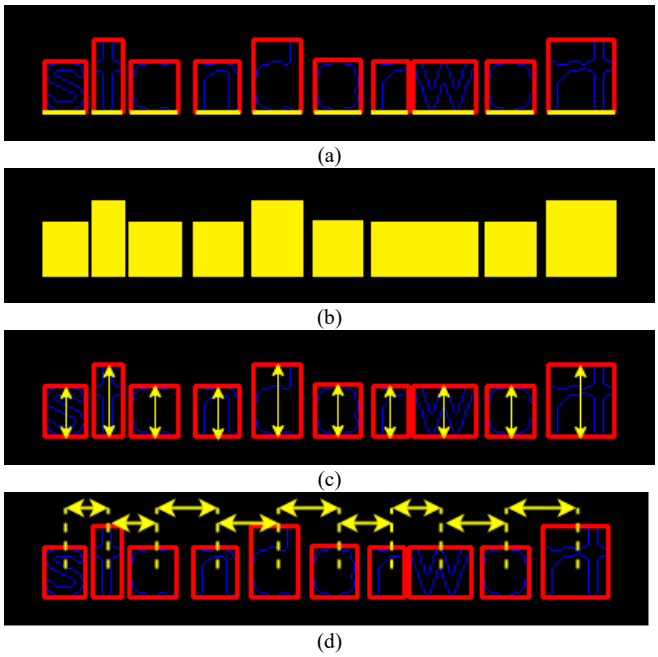

Fig. 3. Values from which classifier features are calculated. Rectangles lower y coordinates (a), rectangles areas (b), rectangles heights (c), rectangles distances (d).

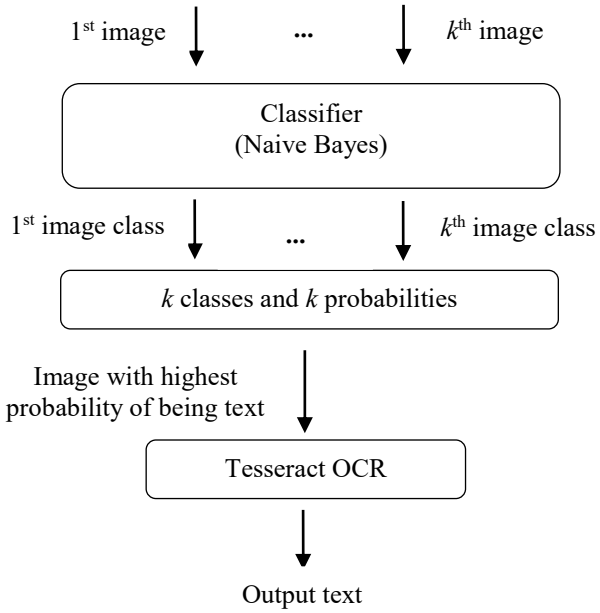

Fig. 4. Classifying k images after original image segmentation and subsequent Tesseract OCR.

To avoid this situation, a black transparent image is added to the cropped images before the text is added to the image. In this way the generated images have colorful background while the text is always readable with a human eye. Text size was also randomly generated within limits suitable for Tesseract. Examples of generated images are shown in Fig. 5. The generated images were randomly divided into training and testing dataset such that each dataset contains 1000 images.

Training dataset was used to train the classifier. Features used for training the classifier were calculated from the bounding rectangles surrounding all contours on images after running the k-means algorithm as described in Section III and IV. To recall, the used features are: RSD of lower y coordinates, RSD of areas, RSD of heights and RSD of distances. Fig. 6 shows the rectangles surrounding all contours on each of the images produced during clustering of the image Fig. 5 (d) with $k = 7$. As can be seen from Table I, the image that is containing text has significantly lower values of the used features in comparison with the rest of the clusters. To examine the aforementioned features further, the distribution of the features values from the training images are shown in Fig. 7.

Fig. 5. Examples of generated images.

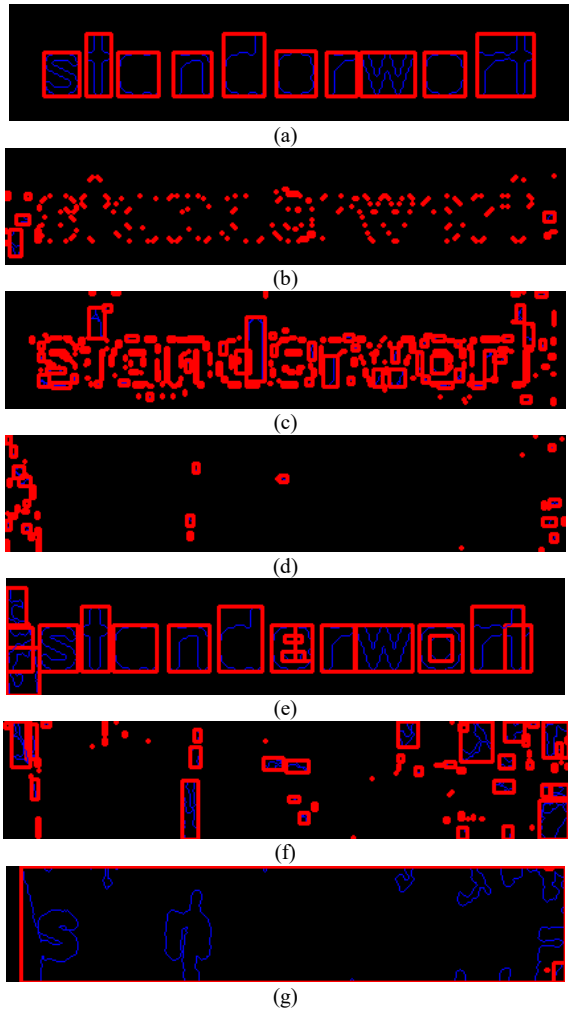

Fig. 6. Rectangles surrounding all contours on each of the images produced during clustering of the image Fig. 5 (d) with k = 7 (a) – (g)

Hereby, the values of the features calculated from images containing text are marked in red color, while the values of the features calculated from images containing background are marked in blue color. It can be concluded that there is no feature that can completely divide the two classes.

Several standard types of machine learning classifiers were developed based on the training dataset. In the end, Naïve Bayesian classifier was chosen since it obtained minimal classification error calculated by the cross-validation technique on the train dataset. The evaluation of the proposed method on the test dataset was performed in two ways. At first the accuracy of the built classifier is examined on the test dataset. The accuracy of the classifier is defined as the number of correct text image classifications after clustering. Basically, a test image is clustered with k set to 10 and then resulting image with the highest probability is inspected whether or not it contains text. Since the model will never get as an input an image without text, the accuracy is the only metric used to evaluate the model, while other metrics such as sensitivity or specificity are not used.

TABLE I. FEATURES FROM CLUSTER IN FIG. 6.

Image from Fig. 6	*RSD of lower y coordinates*	*RSD of areas*	*RSD of heights*	*RSD of distances*
(a)	0.000000	32.90119	17.96446	16.07143
(b)	27.4166	462.9296	126.3277	120.7123
(c)	31.64519	258.2908	123.0358	90.99444
(d)	51.05530	99.98101	62.14392	267.1192
(e)	19.51131	63.56925	44.64616	79.55530
(f)	50.54622	260.8566	126.2880	191.7235
(g)	62.72304	194.6598	116.1731	141.8037

The build classifier achieves accuracy of 95.5% on the test dataset. The second way of the proposed method evaluation is based on comparison of Tesseract OCR accuracy with and without proposed preprocessing method. The OCR accuracy on original test images using only Tesseract 4.0 is 42.6%. After applying proposed preprocessing method, OCR accuracy of Tesseract 4.0 is 62.47% on the test dataset which is the improvement of 19.87%. An example of images with incorrect OCR after applying proposed preprocessing method is shown in Fig. 8. The OCR result obtained on original image is “iAItc.”. After applying proposed preprocessing method, the OCR result on image classified as image containing text is “Alto.”. Even though the result is still incorrect, in this specific case it can be said that image clustering and image classification provides some OCR improvement compared to the original OCR result. An example of correct OCR after applying proposed preprocessing method is shown in Fig. 9. The OCR result obtained on original image is “fannas”. After applying image proposed preprocessing method, OCR result on the image classified as image containing text is “anergias” which is correct.

VI. CONCLUSION

Performing OCR with Tesseract on images with colorful background usually end up with wrong results. The most common approach to solve this problem is image preprocessing before running OCR. In this paper, a preprocessing method is proposed which consists of the segmentation step and image classification step. In the first step an image is clustered into k clusters with the intention to extract the text from the background. Hereby, one image is supposed to contain text while the rest of the k-1 images contain image background. In the second step the image that contains text is identified with the Naïve Bayes classifier from the set of k images. The identified image is then processed with Tesseract OCR. However, since the proposed method is a preprocessing method, it can be applied when using other OCR software solutions as well.

The accuracy of the built classifier is 95.5%. The overall Tesseract 4.0 performance is improved by approximately 20% with the proposed preprocessing technique.

While the presented improvement is significant, it has to be pointed out that there are some prerequisite for using the proposed method: the pixels representing text should occupy a

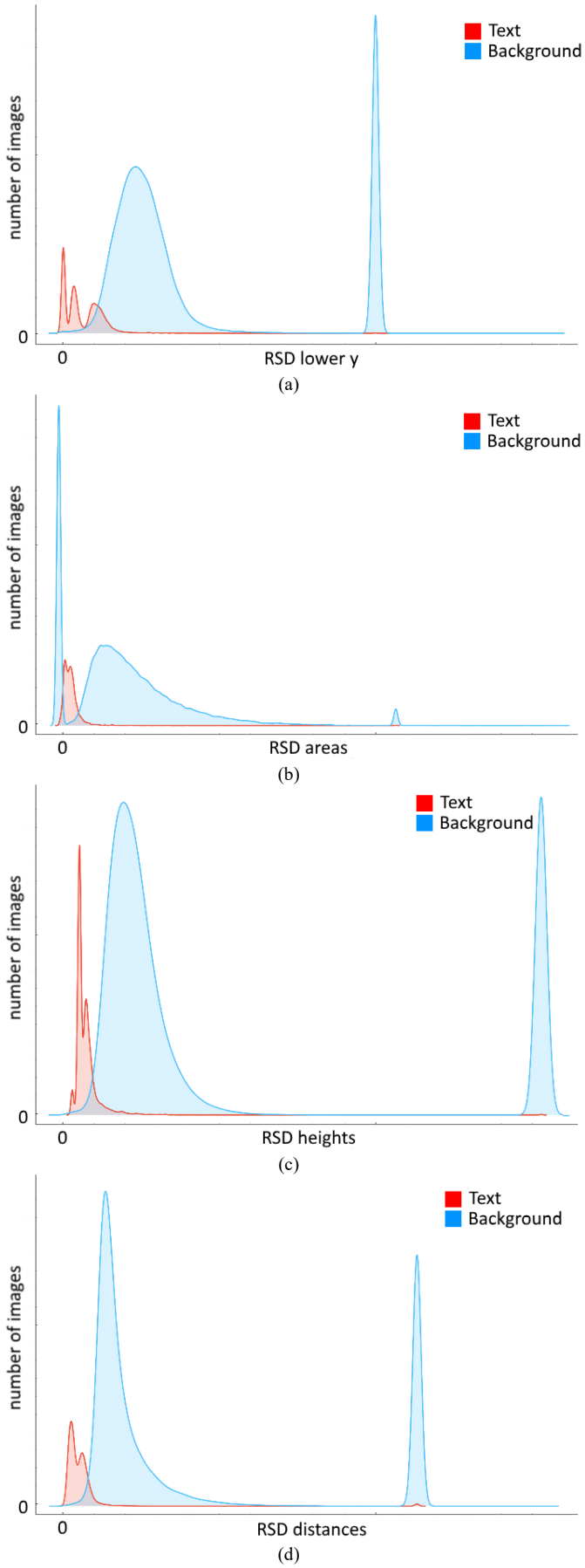

Fig. 7. Distribution of RDS of lower y coordinates (a), RSD of areas (b), RSD of heights (c), RSD distances.

Fig. 8. Example of image with incorrect OCR after applying image clustering and image classification (b) on original image (a).

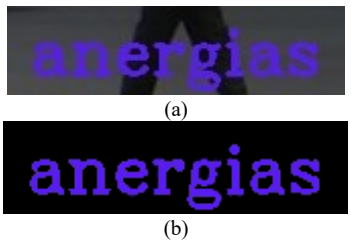

Fig. 9. Example of image with correct OCR after applying image clustering and image classification (b) on original image (a).

significant part of the image and they are approximately of the same color. Additionally, the analyzed image contains only one line of text. This usually requires manual cropping from larger images. Future work will include another step of automation which will find the text regions from larger image, crop these regions and apply the method presented in this paper.

ACKNOWLEDGMENT

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, under grant number: III044009_6.

REFERENCES

[1] Angelica Gabasio, “Comparison of optical character recognition (OCR) software.” Jun-2013.

[2] R. Smith, “An Overview of the Tesseract OCR Engine,” in Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2007, vol. 2, pp. 629–633.

[3] M. Brisinello, R. Grbić, M. Pul, and T. Anđelić, “Improving optical character recognition performance for low quality images,” in 2017 International Symposium ELMAR, 2017, pp. 167–171.

[4] M. Shen and H. Lei, “Improving OCR performance with background image elimination,” in 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2015, pp. 1566–1570.

[5] Datong Chen, H. Bourlard, and J.-P. Thiran, “Text identification in complex background using SVM,” 2001, vol. 2, p. II-621-II-626.

[6] Q. Ye, W. Gao, and Q. Huang, “Automatic text segmentation from complex background,” in 2004 International Conference on Image Processing, 2004. ICIP ’04, 2004, vol. 5, p. 2905–2908 Vol. 5.

[7] N. Shivananda and P. Nagabhushan, “Separation of Foreground Text from Complex Background in Color Document Images,” in 2009 Seventh International Conference on Advances in Pattern Recognition, 2009, pp. 306–309.