A VOTING APPROACH FOR IMAGE BINARIZATION OF TEXT-BASED DOCUMENTS

Costin-Anton BOIANGIU[1], Giorgiana Violeta VLĂSCEANU[2], Alexandru Marian ATANASIU[3], Petrișor Alin DAMIAN[4], Cristian PANAITESCU[5]

We live in a digital era and the need to access everything from everywhere is the new normal. In order to digitalize documents, it is mandatory to have a processing step for enhancing the image document. This step ensures high accuracy for text recognition. Current paper collects appropriate techniques of thresholding on text content images and selects the best result using voting-based mechanisms.

Keywords: image thresholding, voting technologies, Otsu, Niblack, Sauvola, Wolf, Nick, Bradley-Roth

1. Introduction

The focus of the current paper is proposing methods of qualitative evaluation for existing implementations of thresholding algorithms and to assess how we can improve choosing the most suitable output given a fixed set of algorithms.

1.1. Thresholding

Thresholding is a class of segmentation algorithms which aims to split the information in an image into 2 classes: a foreground class and a background class (as opposed to general segmentation which aims to generate *k* classes representing *k* distinct entities in the image) like in Fig. 1.

[1] Professor, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, email: costin.boiangiu@cs.pub.ro
[2] Teaching Assistant, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, email: giorgiana.vlasceanu@cs.pub.ro
[3] Engineer, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, email: alexandru.atanasiu@outlook.com
[4] Engineer, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, email: alin.damian94@gmail.com
[5] Engineer, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, email: cristian.panaitescu24@gmail.com

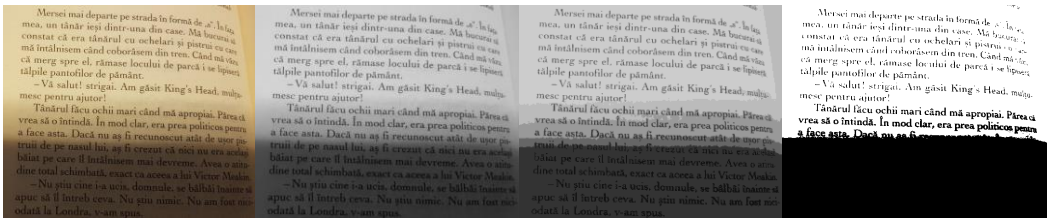

Fig. 1. From left to right: original image document, grayscale conversion using CIE-Y luminance component [1], segmentation with 6 classes using a histogram-based segmentation [2], and with 2 classes (the latter is also referred to as binarization or thresholding).

For an image I, defined as a two-variable function $I:\{0,1,\dots w\} \times \{0,1,\dots h\} \rightarrow [0,1]$, where w is the width and h is the height, we can define the thresholding operation as $T:\{0,1,\dots w\} \times \{0,1,\dots h\} \rightarrow \{0,1\}$, with 0 and 1 being the two distinct classes that we want to separate. Usually, 0 is the foreground class and it is represented by a fully black pixel and 1 is the background class and is represented by a fully white pixel.

1.2. Thresholding algorithm types

Thresholding algorithms can be classified into the following categories:

- Histogram-shape algorithms;
- Clustering-based algorithms;
- Entropy-based algorithms;
- Local algorithms.

The Histogram-shape algorithms [3] choose to analyze the histogram of one image (e.g. peaks, valleys, and curvatures) and extract meaningful statistics in order to, finally, detect the most appropriate threshold value. The histogram can be defined as the distribution of discrete values (usually pixels have a discrete representation e.g.: 0-255). Mathematically, histograms can be described as a function $\boldsymbol{h}:\{\mathbf{0},\mathbf{1},\dots\mathbf{255}\} \rightarrow \mathbb{N}$ with the property of $\boldsymbol{h(v) = c}$ if there are exactly $\boldsymbol{c}$ pixels in the image that have the value $\boldsymbol{v}$.

The Clustering techniques [4] investigate the splitting into two clusters of gray pixel collections which are both normally (Gaussian-like) distributed in order to separate the background and the foreground class.

The Entropy algorithms [5] compute the entropies of foreground and background images and compare the cross-entropy of the resulting image in order to compute the variation.

The Local algorithms [6] either split the image into tiled windows or operates using pixel-centered sliding ones and computes several computationally inexpensive statistics inside the aforementioned window in order to compute a local threshold. Usually, this is relevant if there are regions that differ drastically

in the thresholding point like in Fig. 2. It can be denoted that no unique threshold may be found on this type of images so that all the text is recovered. In that case, a local algorithm is required since there can't be a single threshold value that separates the foreground from the background, but one for each region of the input image document.

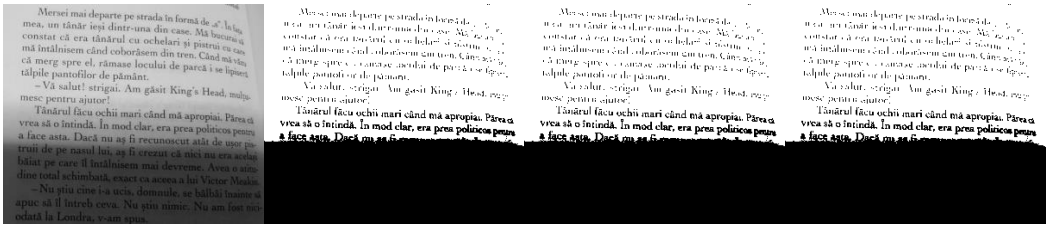

Fig. 2. Example of results for trying a global thresholding algorithm on a document acquired in non-uniform lighting conditions.

A test using local thresholding operations may be seen in Fig. 3. It can be denoted that there should be a fine-tuning operation performed between the window size, on one hand, and the other algorithm-specific parameters, on the other hand, in order to balance the text contrast with the amount of background noise.

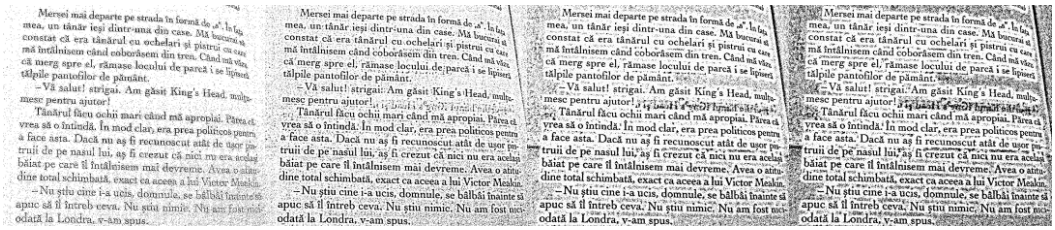

Fig. 3. A test with local thresholding using the same input image that was subjected to a global thresholding operation in Fig. 2.

2. The Employed Algorithms

For the demonstrator application, we have studied and tweaked several thresholding algorithms, from very simple and basic techniques to complex heuristics and formulas in order to detect the thresholding of the image. Important to note is that the current article employs thresholding on text-based images that do not contain images or another kind of media content.

2.1. Global Thresholding Algorithms

The algorithms in the presented approach tend to search in 256 levels greyscale images, using different methods, a global threshold level (let it be called $t \in \{0,1 \dots 255\}$) which will separate the classes as follows: pixels with values below t will be part of the foreground (and will have the value 0 in the resulting image) and the pixels with the value above t will be part of the background (and will have the value 1 in the resulting image).

The proposed approach begins with Otsu's method [7] as the starting point for searching a threshold in initial value. This method calculates the optimal threshold by dividing the foreground and background classes such that their combined expansion is maximal (minimizing the inner-class spread).

Let I be an image, we define ${\sigma_0}^2(t), {\sigma_1}^2(t)$ the variances of the two classes for a given t threshold value, $\omega_0(t), \omega_1(t)$ the probabilities of the two classes (pixel count of each class divided by the total number of pixels), $\mu_0(t), \mu_1(t)$ the means of the two classes, and $p(v)$ the probability for an image pixel to have the value v.

Otsu defines the intra-class variance as:

${\sigma_\omega}^2(t) = \omega_0(t){\sigma_0}^2(t) + \omega_1(t){\sigma_1}^2(t)$, where $\omega_0(t) = \sum_{i=0}^{t-1} p(i)$ and

$$\omega_1(t) = \sum_{i=t}^{255} p(i).$$

The purpose of this algorithm's global approach is to search a threshold in the entire $\{0,1 \dots 255\}$ set of valid values which minimizes ${\sigma_\omega}^2(t)$.

Otsu's observation is that minimizing the intra-class variance means, in fact, maximizing the inter-class variance

$$ {\sigma_b}^2(t) = \sigma^2 - {\sigma_w}^2 = \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2 $$

where $\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$ and $\mu_1(t) = \frac{\sum_{i=t}^{255} ip(i)}{\omega_1(t)}$.

The Otsu algorithm implies the following computing steps, and performs very fast in just 256 cycles using just simple, incrementally built statistics:

1. Compute $h(v)$ and $p(v) = \frac{h(v)}{\sum_{i=0}^{255} h(i)}$
2. Set initial $\omega_i(0)$ and $\mu_i(0)$
3. For each i in available values (0…255)
    a. Update ω_i and μ_i
    b. Compute ${\sigma_b}^2(i)$
4. Select the threshold where ${\sigma_b}^2(t)$ is maximum

At this stage, the threshold found using Otsu approach is considered to be the most-probable global one. Around its value, iterating through the space of

valid thresholds, and through a set of small windows for median filtering-based smoothing of the image, a significant number of noise-removal operations accompanied by global thresholding operations will be performed on the initial image, thus resulting in a large set of globally binarization solutions.

2.1. Local Thresholding Algorithms

Fig. 2 proves that, sometimes, in the case of non-uniform illumination, a global thresholding operation cannot be successful, no matter how clever the determination of the threshold is. Fig. 3 illustrates the impact achieved when local thresholding is used on this category of images.

It is, therefore, the duty of the local thresholding to enter the stage. In the presented research a number of 3 different local approaches were selected:

- Local mean-based thresholding, employed from [8] where the local threshold value is computed by adding a constant value around a local mean retrieved on a sliding local window, centered in each pixel, one after another;
- The same as above but using a local Gaussian-weighted sum computed in the same manner, in the neighborhood of each pixel. The thresholding operation was employed from [8];
- An adaptive threshold using the integral image technique and proposed by Bradley and Roth [9].

For each of all the aforementioned techniques, we performed a multi-space iteration for every of the relevant input parameters selected from their meaningful value range, thus resulting in a large set of locally binarization solutions.

In the end, we join the local and global set of binarization solutions into a large pool of potentially available candidates, from which to select from in the next stages of voting-based processing.

It is not a subject of this paper to recommend iterators for every one of the parameters involved in the global and local thresholding operations, the basic idea is, that generating more viable candidates will increase the quality of the end result, but also will increase the execution time for both the generation and the voting processes. In the end, selecting the proper balance point will depend on how much time an application may be allowed to consume.

2.3. Voting System and Heuristics

In order to be able to choose a fit threshold, we studied several heuristics to be able to vote out the most appropriate thresholding method that our algorithms can perform. The proposed voting system, similar to the one used in [10][11], takes into consideration **multiple elections** in order to come up with a certain set of proper thresholds on the given image.

The first election of the voting algorithm is based on eliminating trivial candidates that don't represent a viable threshold. Since we started with the assumption that our input images are photos taken from books/documents with text-only information, we can impose that whatever image we will apply our algorithms on, the ratio between foreground and background pixels cannot be beyond above a certain ratio, thus eliminating images like the one in Fig. 4.

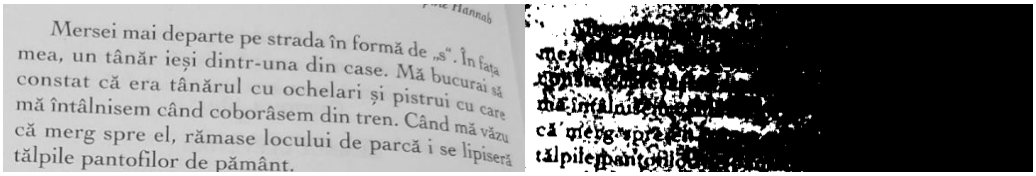

Fig. 4. The input image and a trivial case of elimination where the chosen threshold was too high and the ratio was too high.

The second election of the voting algorithm takes a similar approach, but it applies it locally on windows of the approximate size of three text rows' height ensuring that, although global thresholding is successful, locally there have been detected some abnormalities. Although at this moment, we may use fixed thresholds like in the first step, this might most likely fail if we have for example page margins (where we don't have both foreground and background elements) like in Fig. 5.

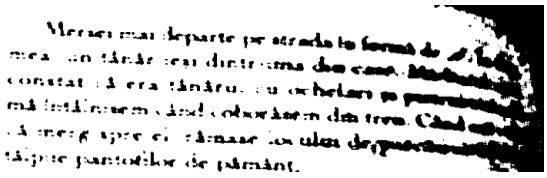

Fig. 5. Although this binarization passes the first step, there are certain windows where we can detect that the threshold is not properly computed.

In order to consider this, we compute the histogram of the input image (which for convenience will have 256 possible values) on the current window and we execute two eliminatory tests:

- The average test;
- The entropy tests.

The average test computes the weighted average value of the histogram for the initial window of the image: $A_i = \frac{\sum I(i,j)}{w^2}$ and the average between the threshold values $A_t = \frac{h_t(0)+h_t(1)}{w^2}$, where w is the window's size.

The test implies that the ratio between the two averages should be in the same range as the ratios set in the first step.

The entropy test computes the entropy of the entry window $S_i = -\sum p(i) log\,[p(i)]$ and the entropy of the generated by the thresholding $S_t = -p(0)\log(p(0)) - p(1)\log(p(1))$ and we enforce that the ratio between entropies should be more than 2 (meaning that once we group more values into a single value, we can consider that the entropy may in the worst case halve).

Although this step is computationally more expensive than the first, it is easy to parallelize on SIMD architectures (using, for instance, GPGPU), since we have fixed size of chunks, and we compute the same steps on different chunks of data, improving this way the overall performance of the thresholding system.

The third election is a tournament-based step designed in 2 steps: once on all binarizations and once on groups of binarizations (e.g. 16 players in a batch); where intend to eliminate a ratio of the players (e.g. 25% of the oddest players).

In order to do that, we compute an auxiliary structure: a probability matrix, the same size as the image itself, in which for every pixel, the computed value will be the ratio between how many times that pixel was classified as foreground and the total number of images in a batch. This auxiliary probability matrix will be processed as follows: corresponding pixels that are below the 50% threshold will be assigned to class 0, and the ones above will be assigned to class 1. Now, we can evaluate each image to see how many pixels match this processed probability image. The worst 25% are dropped from the tournament.

Doing one such step on all binarizations and two such steps on groups, and again one step an all, assures keeping around only 30% of the candidate thresholding operations which succeeded into this tournament. After each of these steps, we need to reshuffle the data batches in order not to propagate the same erroneous qualifiers. The process is illustrated in Fig. 6.

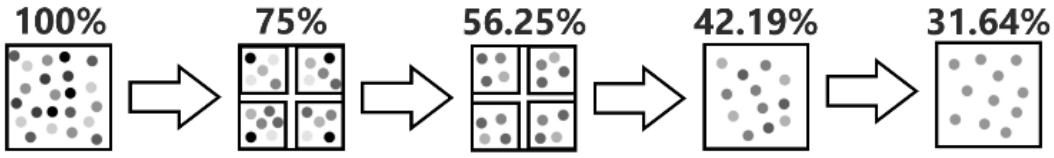

Fig. 6. The tournament election that removes around 70% of unfit candidates.

The last step, which is not an election per se, takes the remaining images, builds a probability matrix, as in the previous step, consider it as a grayscale image where the probability in the [0%-100%] range stretched to [0-255] gray-shades range, and constructs the final binarized image by thresholding in the middle (127).

3. The implementation and results

The demonstrator system that was implemented for this article was written in Python and GNU Linux Bash. There is a central bash script which tests on multiple images the aforementioned algorithms.

As Python implementation, there is a script that implements all the thresholding variation to create the pool tests and one script used for evaluation. For manipulation of images, basic processing, and thresholding, it was used the OpenCV library [8].

In order to evaluate the presented approach, a thorough comparison was performed against some of the most popular thresholding techniques: Otsu [7] for the globally-based approach and Niblack [12], Sauvola [13], Wolf [14], Nick [15] for the local-based approaches. A basic local average thresholding was included in the batch test as a reference for the most brutal foreground and background local separation, ensuring the best edge consistency but with the most amount of noise located in continuous tones.

For the evaluation process, a dataset comprised of a selection containing text-only document images (with binarization ground-truth maps) was employed. The selection was performed from a pool containing all the 101 images from the following well-known image binarization databases: ICDAR Document Image Database Competition (DIBCO-2009, DIBCO-2010, DIBCO-2011, DIBCO-2012, DIBCO-2013, DIBCO-2014, DIBCO-2016) [16] and PHIBD-2012 [17]. Since the proposed voting approach is designed to work only on textual information, a total of 12 documents were rejected for not having text-like statistics throughout their entire content. In the end, 89 of the most complex, text-based document images along with their binarized ground-truth maps were employed in the evaluation of the proposed technique's performance.

In Fig. 7 are illustrated some classic binarization approaches applied to the input document presented in Fig. 1. This document exhibits several types of degradation, uneven lighting, uneven contrast, acquisition noise. All the locally-based approaches were tested using a sliding window of 33x33 pixels. For all the other parameters, they were set according to their general-purpose recommended values.

In general, the tests revealed that the presented voting approach always ranks amongst the best methods if the input image document is text-based. For documents that contain unusual writing, illustrations, decorators, or diagrams, the presented research may not offer optimal results since the voting selection is based on rejection tests using the average font-filling statistics. Moreover, the local window size is chosen in relation to the average text height.

The voting influence consists in selecting, from the pool of the available binarization candidates, of only the ones that do not have defects in terms of

statistics at both local and global levels. The elimination of candidates, in both randomly-chosen batches and full-size tournaments, improves the chance that the per-pixel decision inside the group of finally-remaining candidates may result in a binarization with the least damage in the most areas of the input image document.

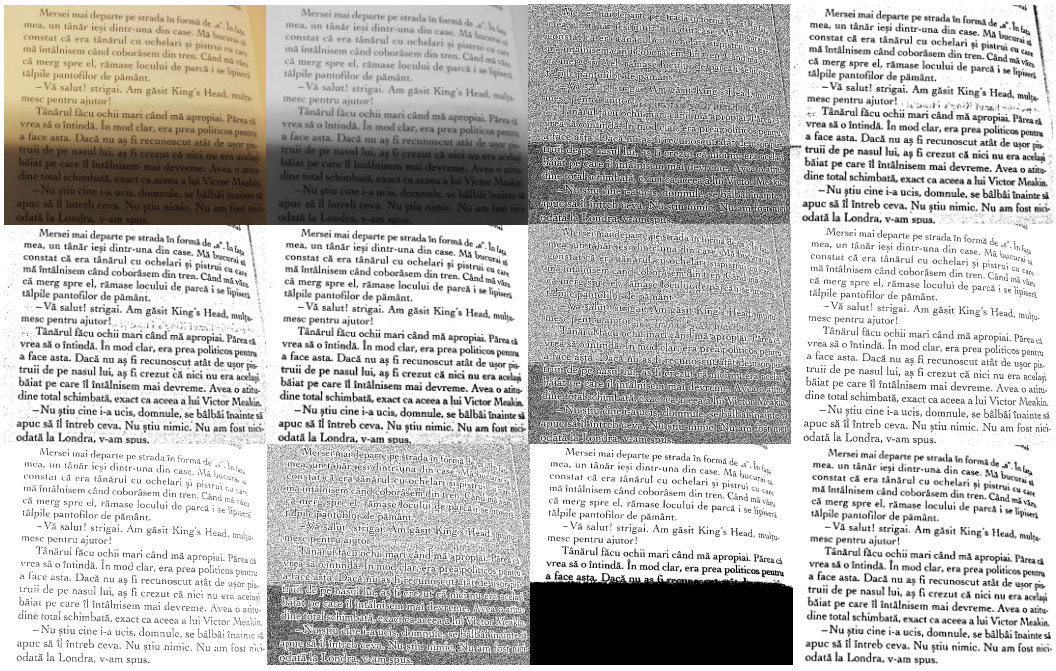

Fig. 7. From left to right, top to bottom: original image document, grayscale conversion using CIE-Y luminance component [1], local average (reference for best separation/worst noise), adaptive based on offset average [8], adaptive based on offset Gaussian [8], Bradley-Roth [9], Niblack [12], Nick [15], Sauvola [13], Wolf [14], Otsu [7], Proposed voting approach.

Table 1

The proposed approach compared with some of the most representative methods

Binarization Method	Average F-Measure	STDEV(F-Measure)
Average	38.10	14.41
Niblack	43.61	15.32
Nick	79.39	14.70
Sauvola	61.14	27.70
Wolf	59.72	17.47
Otsu	80.55	17.35
Proposed method	**82.72**	**11.32**

Table 1 presents the aggregated statistics on the images from the aforementioned datasets when running the proposed method against some of the most representative (both global and local) approaches in the scientific literature. It can be noticed that the method gets the first place when compared to all the other candidates for F-Measure. Moreover, the proposed methods have the smallest overall standard deviation for F-Measure. This proves not only that the method is most suitable when compared to other approaches but also that it behaves the most stable in terms of results' consistency.

5. Conclusions

This paper presented a series of techniques for image thresholding in order to separate the foreground from the background. The proposed approach presents a binarization system that runs several methods to generate viable candidates for the problem at hand, then performs a series of validations tests and voting-based approaches in a tournament–like selections in order to generate the most suitable candidate.

The presented solution is performant when compared to other classical solutions, stable in providing excellent results, robust to the errors inherently added by the binarization algorithms candidates and contains only fast, computationally inexpensive statistics necessary to perform the basic decisions. It is also very useful when not a single threshold value can be the solution to the binarization problem, like in the case of variable lighting conditions across the same page, and, as a result, can be safely used as an unsupervised binarization stage in a large-scale mass-digitization project.

Future work may be oriented toward better rejection statistics at both individual level and tournament level and a more educated shuffle operation, in order to better propagate the best-fit candidates at upper tournaments' levels.

Acknowledgement

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI - UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0689 / „Lib2Life- Revitalizarea bibliotecilor si a patrimoniului cultural prin tehnologii avansate" / "Revitalizing Libraries and Cultural Heritage through Advanced Technologies", within PNCDI III.

R E F E R E N C E S

[1] *Sareesh Sudhakaran*, "What is the difference between CIE LAB, CIE RGB, CIE xyY and CIE XYZ?", January 3rd, 2013, Retrieved from https://wolfcrow.com/what-is-the-difference-between-cie-lab-cie-rgb-cie-xyy-and-cie-xyz/ on April 5th, 2019.

[2] *Julie Delon, Agnès Desolneux, José-Luis Lisani, Ana-Belén Petro*, "A nonparametric approach for histogram segmentation", IEEE Trans. Image Process. 16 (1), 2007, pp. 253-261.

[3] *Sumeyya Ilkin, Fatma Selin, Suhap Sahin, "*Comparison of Global Histogram-based Thresholding Methods that Applied on Wound Images", International Journal of Computer Applications, Vol. 165, doi: 10.5120/ijca2017914002, 2017, pp. 23-28.

[4] *L. Benrais, N. Baha*, "Image Segmentation Using Clustering Methods", Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016, ISBN 978-3-319-56991-8, 2018.

[5] *L. Mahmoudi, A. El Zaart*, "A survey of entropy image thresholding techniques", 2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), Beirut, doi: 10.1109/ICTEA.2012.6462867, 2012, pp. 204-209.

[6] *J. Soumela, "*Survey of local algorithms", ACM Computing Surveys (CSUR), Vol. 45 Issue 2, February 2013, Article No. 24, 2013.

[7] *H.J. Vala, Astha Baxi*, "A review on Otsu image segmentation algorithm". International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 2 (2), 2013.

[8] *** OpenCV (Open Computer Vision) Library Thresholding Tutorial, Retrieved from https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html on March 21st, 2019.

[9] *Derek Bradley, Gerhard Roth, "*Adaptive Thresholding using Integral Image", Journal of Graphics Tools, Vol. 12, No. 2, doi: 10.1080/2151237X.2007.10129236, 2007, pp. 13-21.

[10] *Costin-Anton Boiangiu, Radu Ioanitescu, "*Voting-Based Image Segmentation", The Proceedings of Journal ISOM, Vol. 7 No. 2, ISSN 1843-4711, pp. 211-220, 2013.

[11] *B. Parhami, "*Voting algorithms", IEEE Transactions on Reliability, vol. 43, no. 4, doi: 10.1109/24.370218, 1994, pp. 617-629.

[12] *O. A. Samorodova, Andrey Samorodov*, "Fast implementation of the Niblack binarization algorithm for microscope image segmentation", Pattern Recognition and Image Analysis. 26, doi: 10.1134/S1054661816030020, 2016, pp. 548-551.

[13] *J. Sauvola, M Pietikainen*, "Adaptive document image binarization, Pattern Recognition", Volume 33, Issue 2, ISSN 0031-3203, doi: 10.1016/S0031-3203(99)00055-2, 2000, pp. 225-236.

[14] *Christian Wolf, Jean-Michel Jolion*, "Extraction and Recognition of Artificial Text in Multimedia Documents", Formal Pattern Analysis & Applications, Vol. 6, doi: 10.1007/s10044-003-0197-7, 2004, pp. 309-326.

[15] *Khurram Khurshid, Imran Siddiqi, Claudie Faure, Nicole Vincent*, "Comparison of Niblack inspired binarization methods for ancient documents", Proc. SPIE 7247, Document Recognition and Retrieval XVI, 72470U; doi:10.1117/12.805827, 2009.

[16] *Ioannis Pratikakis, Konstantinos Zagoris, Georgios Barlas, Basilis Gatos*, "DIBCO 2017 / ICDAR 2017 Document Image Binarization Competition", Retrieved from https://vc.ee.duth.gr/dibco2017/ on April 5th, 2019.

[17] *Hossein Ziaie Nafchi, Seyed Morteza Ayatollahi, Reza Farrahi Moghaddam, Mohamed Cheriet*, "Persian Heritage Image Binarization Dataset (PHIBD 2012)", April 30th, 2013, Retrieved from http://www.iapr-tc11.org/mediawiki/index.php/Persian_Heritage_Image_Binarization_Dataset_(PHIBD_2012) on April 5th, 2019.