OCR QUALITY IMPROVEMENT USING IMAGE PREPROCESSING

Vlad Badoiu[1]*
Andrei-Constantin Ciobanu[2]
Sergiu Craitoiu[3]

ABSTRACT:

Optical character recognition (OCR) remains a difficult problem for noisy documents or documents scanned at low resolution. Many current approaches rely on stored font models that are vulnerable to cases in which the document is noisy or is written in a font dissimilar to the stored fonts.

In this paper we test two approaches for preprocessing, or correcting the input images. The focus is on noise reduction, lightness correction and binarization, all relative to found letters with a slow but more accurate method and a fast and less accurate method. We then compare the results and see if the extra time spent in developing more complex letter deduction technique offers significant improvements.

KEYWORDS: OCR preprocessing, image correction, noise reduction, binarization

1. INTRODUCTION

One important problem in the field of image processing is data extraction from documents. We are trying to convert as much information as possible from paper to digital form for preservation purposes. Due to the quantity of data to be converted, manual processing is not an option for cost and time issues. OCR [4] helps with the automation of the process. However especially important historical documents are degraded and they can't be detected by the OCR without prior processing. Here are some of the more common techniques applied before actual interpretation of the text:

- Equipment Calibration and Data Acquisition [7]
- Noise Reduction
- Binarization [8]
- Deskew [9]
- De-speckle

[1]* *corresponding author,* Engineer, vlad.badoiu@cti.pub.ro, "Politehnica" University of Bucharest, 060042 Bucharest, Romania
[2] Engineer, andrei.ciobanu@cti.pub.ro, "Politehnica" University of Bucharest, 060042 Bucharest, Romania
[3] Engineer, sergiu.craitoiu@cti.pub.ro, "Politehnica" University of Bucharest, 060042 Bucharest, Romania

- Classification
- Layout analysis
- OCR preprocessing (the step we're interested in this paper)
- OCR [10]
- Optional step: OCR postprocessing [11]

In this paper we test two approaches at preprocessing and compare the results.

Related work

Noise reduction is an important problem today with a very wide field of applications, from artistic photography to medical diagnosis. In the domain of optical text recognition, noise can affect profoundly the performance of the text recognition and reconstruction software, having a cascade effect on the processing pipeline.

Various methods were employed to be able to reduce the noise without damaging the useful signal in the image. In their work on medical X-ray images enhancement, Hensel et al [1] use a transform called Laplace pyramids. Multiple layers of pyramidal transformations of the image are generated and the processed image is reconstructed by combining these layers in a different way that gives a bigger weight to the useful signal and a smaller weight to the noise. This work is the main inspiration for the presented application.

Usually referred to as "salt and pepper" noise, or impulse noise, the type of noise given by extreme values for some of the pixels is the subject of Chan et al. [2]. This type of noise can appear in binary images after different other noise cleaning methods are applied. In their work, they apply adaptive median filtering to neutralize this kind of noise from images.

In 1998, Winkler et al [3] did an extensive work on edge preserving noise removal methods. They analyze some of the current methods and approach M-estimators and nonlinear filters to obtain similar performance.

Aside the correction of the image, a slightly modified version of the Adaptive thresholding using integral images [6] will be used, in order to work with degraded images.

2. PROPOSED APPROACH

We will compare two different approaches, both very similar, but with different implementation approaches for achieving the purposes (ex: the first approach actually encapsulates letters in bounding boxes, while the second considers letters the regions of

high contrast in the image). At the end we will see if the extra time spent on precision processing offers significant improvements or none at all.

3. THE FIRST PREPROCESSING ALGORITHM

- Grayscale transformation of the image
- Histogram equalization
- Adaptive binarization with window dimension 12
- Identification of the potential letters using the WordHeight function on the binarized image
- After the letters are saved, the noise level is measured by counting the potential letters with height less than a certain value
- If the noise level is above a threshold, we apply noise reduction using a Gaussian pyramid on the post-histogram equalization image from above. Otherwise, we skip to the next step
- We apply adaptive binarization using a size 12 window to determine the average letter height (a potential letter is a set of interconnected black pixels)
- We re-apply the adaptive binarization using the average letter height as the window size. The letters are saved in an array
- Each rectangle containing what is present in the original image inside its limits is saved in an OpenCV matrix which in turn is saved in an array
- On each of these matrices we apply histogram equalization and adaptive thresholding. We create a new white image the same size of the original. The new image is populated by the rectangles. We apply a simple median filter
- We save the final image

3.1. Histogram equalization

This method is mostly used for contrast adjustment using the image's histogram to modify the pixel intensities. [5]

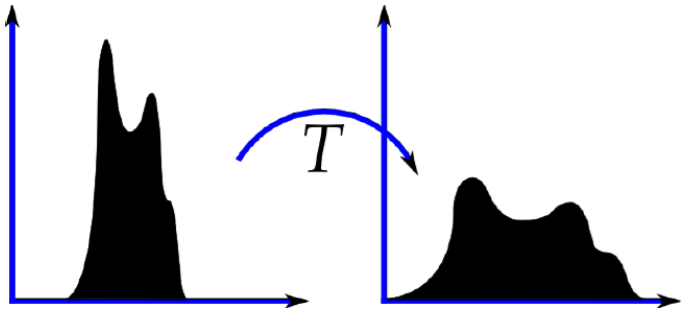


Figure 1. An example of histogram equalization

In this implementation, the color space used is YCbCr. For each intensity, the probability that a pixel has that intensity is calculated and added to the sum of the probabilities of all the intensities smaller than the current one. Having arrived at this value, the algorithm tries to find an i for which i/255 is closest to the current value. After finding it, all pixels with the current intensity will be converted to intensity i.

3.2. The WordHeight function

The idea is that after a basic binarization with a small window, most letters will be present more or less correctly. The crucial aspect is their heights. Making the presumption that the binarization will only ruin a small number of letters, we will be left with a meaningful amount of letters from which we can approximate their average height.

The algorithm works by visiting each fairground pixel: for each unvisited black pixel, the following steps are executed:

- The pixel is marked as visited in the visited pixels array and a new letter instance is created
- The immediate neighbors of the pixel are checked and added to the letter
- This iterations ends when reaching the last pixel in the array
- After forming a letter, it is added to the letter array and its height is added to the height array.
- After finding all of the letters, the heights are sorted in a separate array and eliminate the ones below a certain value.
- The median height is considered as the height used in the future steps of the algorithm.
- Finally, letters with height lower than the average will be eliminated.
- During these operations, each letter structure will also contain the encompassing rectangles.

Figure 2. An encompassing rectangle around a binarized letter b

3.3. Noise reduction with Gaussian pyramid

This method consists of reducing the image to half of its dimensions and then expanding it to its original size. In doing this, a sizeable amount of noise will be removed. The reduce operation consists of isolating a 5x5 matrix with the current pixel as its center and calculating 4 values by multiplying the middle row, middle column and diagonals with vectors and then performing a mean average. These vectors are:

- [0.1, 0.25, 0.3, 0.25, 0.1] for the column and row
- [0.075, 0.225, 0.4, 0.225, 0.075] for the diagonals; the pixels are chosen intermittently on the columns and rows. The expand operation is the inverse of the first one, interpolating the pixel values to find the missing pixel values.

3.4. Final steps

The image is re-binarized according to the medium computed letter height. Thus, we can establish an encompassing rectangle for each letter and local operations can be performed, ignoring the potential large contrast differences from other parts of the image. Also, to eliminate the noise which will be saved as letters, only letters over certain heights will be passed along to the next step.

4. SECOND APPROACH FOR IMAGE PREPROCESSING

- Substract Laplacian Operator
- Non Local Means Denoising
- Noise removal from non-text regions
- Automatically choose binarization parameters based on noise measurements
- Automatically choose binarization parameters based on result observations

4.1 Discrete Laplacian Operator

This is the first important algorithm applied over the image, of course after the image was transformed to grayscale. This step is done to detect the edges of letter. Of course to see the result, scaling to absolute is required.

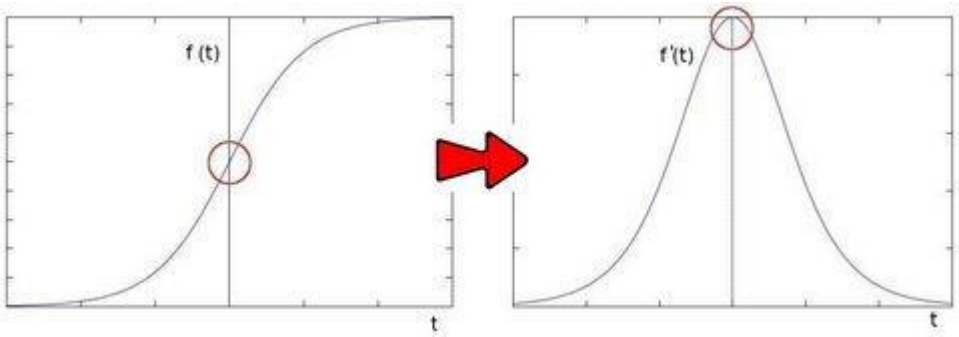


Figure 3. Edge is characterized by a maximum. First Derivative[1]

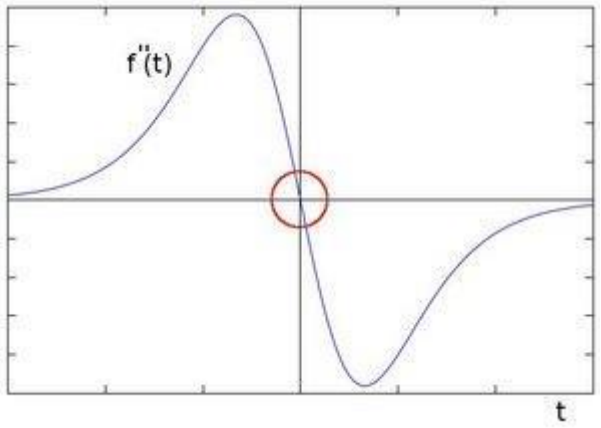


Figure 4. Second derivative is 0. Used in edge detecion[1]

4.2 Store difference between Laplace and normal image

Now subtracting the result obtained from the Laplace transform from the original image, the algorithm will increase letters thickness. Parameters can be change by increasing the contrast of the Laplace image.

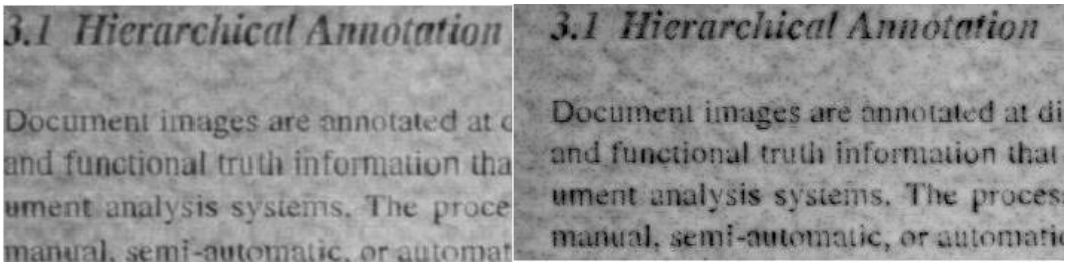


Figure 5. Applying Laplace and subtraction from original: left=original, right=processed.

4.3 Denoising

Denoising is applied over the image obtained in the second step. There are a few algorithms used in image processing. For this experiment Non Local Means Denoising was used. The algorithm searches for similar pixels in the image and can classify a pixel as a good value or noise. To compute similarities in a window a weight function can be applied.

There are a few factors that should be considered when implementing this algorithm. The size of the window (gives the processing time), the similarity window which must be smaller than the process window (usually it's somewhere about 7X7 pixels and the weight function. This is the most time consuming part of the algorithm.

4.4 Speckle removal

The noise removal stage affects only the binary noise that occurs in non-text areas.

The noise removal algorithm is obtained by applying the following steps:

- Compute the binary gradient map by applying the Sobel operator (Figure 6);
- Binarize the gradient map (Figure 7);
- Apply a morphological dilation to the binary gradient map. This step is necessary in order to make sure that we don't remove any of the text regions (Figure 8).
- Remove small contours from the image obtained in the previous step (Figure 9).
- Apply a bitwise and operator between the image obtained in step 4 and the binary image selected by the adaptive thresholding algorithm (Figure 10).

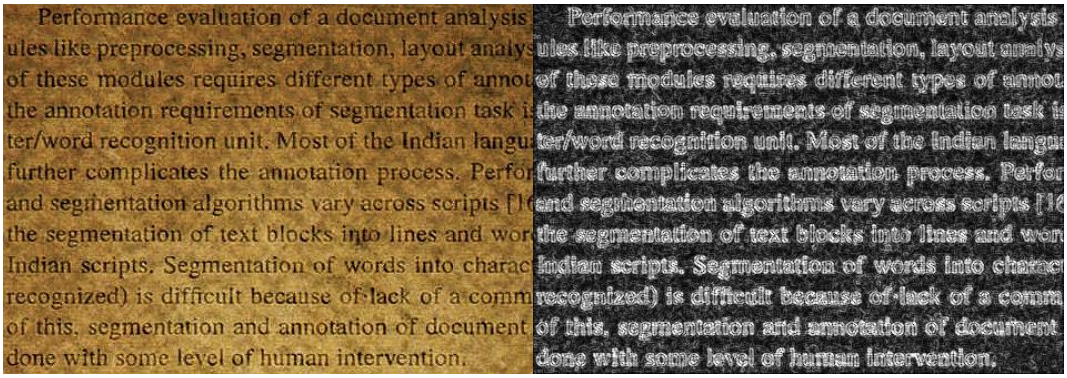


Figure 6. The results after applying the Sobel operator: left - edge enhanced image, right - gradient map

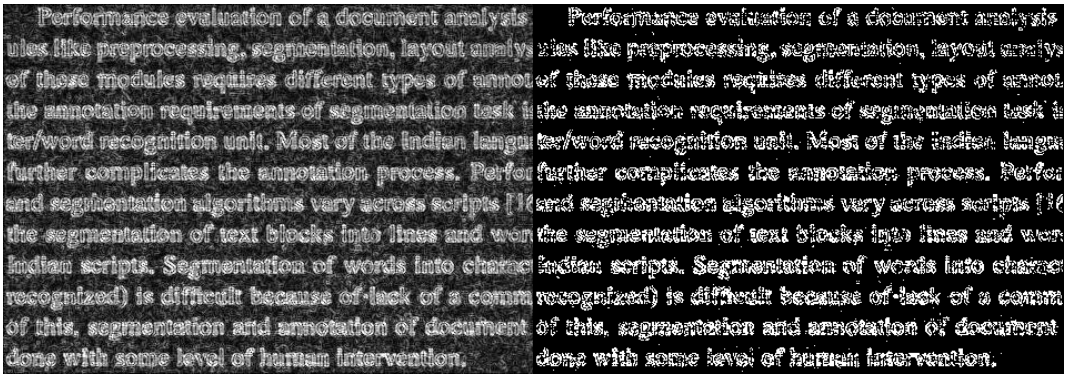


Figure 7. The results after applying a threshold binarization to the gradient map: left - gradient map, right - binary gradient map

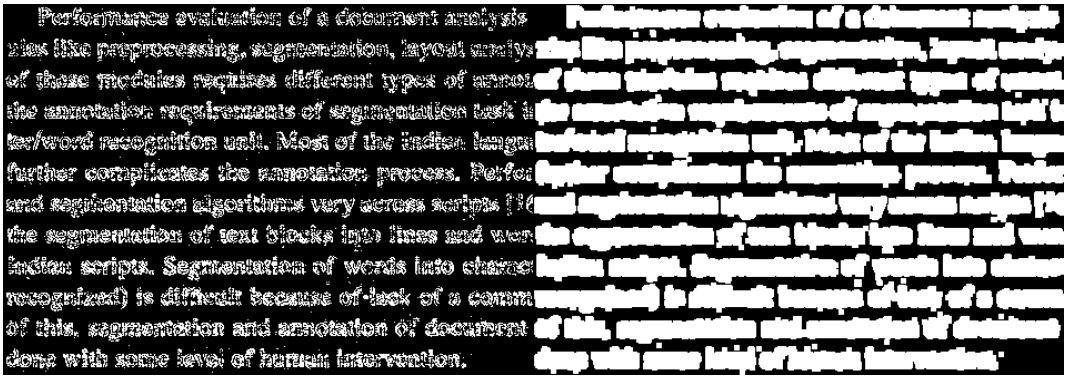


Figure 8. The results after applying the dilation morphological operator: left - binary gradient map, right - dilated binary gradient map

Figure 9. The results after removing small contours from the dilated gradient map: left - original image, right - processed image

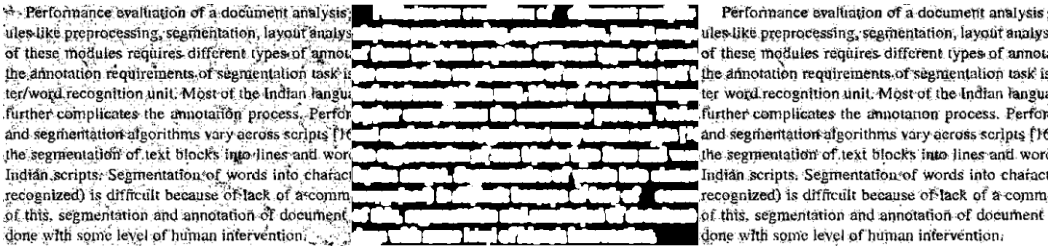


Figure 10. The final result (right) obtained after combining the binary image (left) with the dilated gradient map (center).

4.5 Compute Noise Ratio

The noise ratio can be estimated based on the difference between the denoised image and the original image. The noise estimation helps in deciding upon which threshold algorithm to choose from. The adaptive thresholding is applied only if the noise ratio is above the threshold. The final image is binarized based on an adaptive thresholding method. The method is comparing if a pixel has a smaller or bigger value than the average pixels. If the pixel value is smaller only with a percent then the value is turned to black otherwise is white. This algorithm is super robust and can perform well on lighting variations. The only problem to be consider it the dimension of the window which can be chose based on the width of the image.

4.6 Binarization algorithm improvements

For this project we have used a binarization algorithm described in the article "Adaptive thresholding using the integral image" [6]. This is a simple and easy to implement algorithm, and provides good results in cases where the illumination is not uniform.

The adaptive thresholding algorithm depends on two parameters: S - a fixed window size and T - a percentage threshold.

In short terms, the main steps of the binarization algorithm are:

- For every pixel I(x; y) in the input image I, compute the mean intensity of the pixels in the window S centered in (x; y);

- If the intensity of the pixel in the center of S is lower with T percent than the mean intensity set its correspondent pixel in the binary image to 0, else set the correspondent pixel to 1.

For speed optimization, the integral image is computed at the beginning. Every pixel in this image is represented by the sum of all pixels with lower position indexes (in a top to bottom, left to right order).

In order to determine the optimal size of the window (parameter S) we have applied the following steps (results are shown in Figure 11):

- Compute multiple binary images with various sizes (for example from 2 to 20)

- Select the image that has the biggest difference in terms of the number of white pixels compared to the previous size

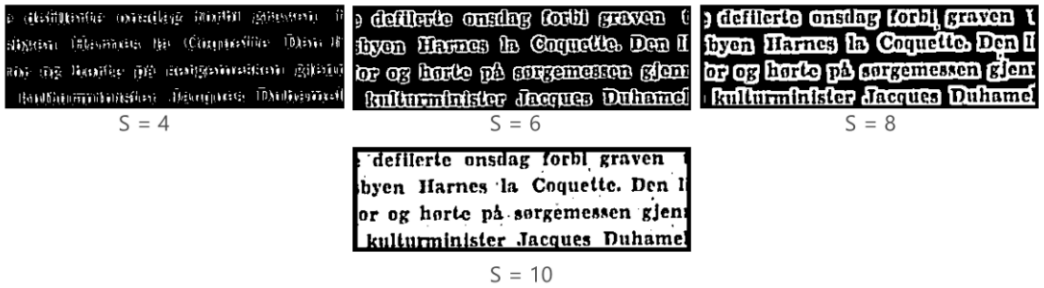
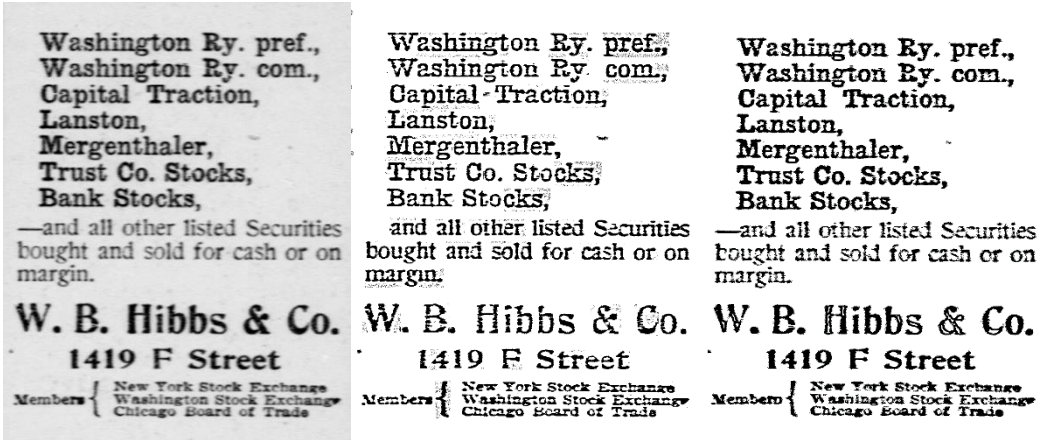


Figure 11. Binary images computed by varying the window size. In this case, the last image is selected as the best candidate.

5. RESULTS



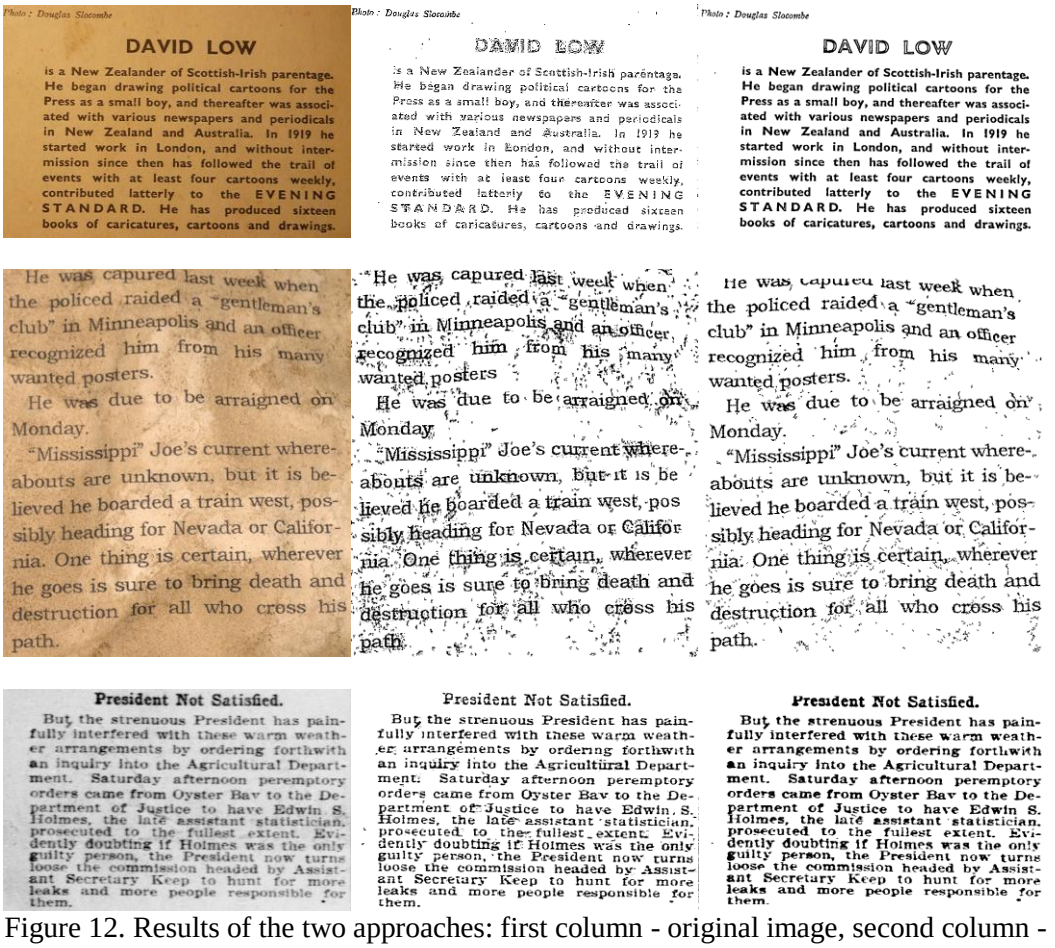


Figure 12. Results of the two approaches: first column - original image, second column - first approach, third column - second approach

From the results of different images we can see that even if the ideas are similar, the actual implementation approaches offer slightly different results. The first algorithm tends to induce noise around letters because of the binarization around each letter, but the letters are connected, whilst the second approach tends to find edges around large letters because of the edge detection approach in finding letters.

1. CONCLUSION

Because the range of the possible input images in the task of text recognition is very wide it is important to be able to bring different types of document scans to a common format that is easy to process by an OCR software. This involves an adaptive preprocessing that can correct the deteriorated document images without breaking the clean ones.

The list of dimensions of variability is very long, with just a few mentions being: contrast, skew, warping, text color, size and type-face, page layout and formatting, shadows /

specular highlights, background color, noise level, blur level. The current work aims to add invariance for contrast and noise differences.

Noise is particularly important, because it is almost omnipresent in the domain of scanned documents and it very often interferes with the textual content. A full range of factors can add noise to an image, from camera sensor aberrations, print bleeding through the page, to dirt and printer defects.

By using an adaptive contrast adjustment followed by a noise reduction algorithm, this work aims to improve the readability of documents, both for human readers and for OCR software. Noise reduction is obtained by decomposing the image in a Laplacian pyramid and recomposing this back into a gray-scale image with giving less visibility to the noisy layers. This algorithm chaining can give bad result on images with different type and sizes of fonts and good results on images with few variations for size.

The second algorithm is based on a series of smaller algorithms put together to create a good image which in the end is passed to an OCR algorithm. On a medium PC, the algorithm can take up to a few minutes, depending on the size of the input image. For future work parallelization can be implemented to speed up the whole process. The denoising algorithm is already parallelized using OMP.

REFERENCES

[1] Marc Hensel, Ulf Brummund, Thomas Pralow, Rolf-Rainer Grigat, "Noise Reduction with Edge Preservation by Multiscale Analysis of Medical X-Ray Image Sequences", Bildverarbeitung für die Medizin 2005, Informatik aktuell, pp 55-59

[2] Chan, R.H., Chung-Wa Ho, Nikolova M., "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization", IEEE Transactions on Image Processing. November 2005; 14(10), pp.1479 - 1485.

[3] Winkler, Aurich, Hahn, Martin, Rodenacker, "Noise Reduction in Images: Some Recent Edge-Preserving Methods", Sonderforschungsbereich 386, Paper 138 (1998)

[4] http:// en.wikipedia.org/ wiki/ Optical_character_recognition, Accessed: January 2015

[5] http:// en.wikipedia.org/ wiki/ Histogram_equalization, Accessed: January 2015

[6] D. Bradley and G. Roth, "Adaptive thresholding using the integral image," Journal of Graphics Tools 12(2), pp. 13–21, 2007

[7] Costin-Anton Boiangiu, Alexandru Victor Stefănescu, "Target Validation and Image Color Calibration", International Journal of Circuits, Systems and Signal Processing, Volume 8, 2014, pp. 195-202

[8] Costin-Anton Boiangiu, Andrei Iulian Dvornic. "Methods of Bitonal Image Conversion for Modern and Classic Documents". WSEAS Transactions on Computers, Issue 7, Volume 7, pp. 1081 – 1090, July 2008.

[9] Costin-Anton Boiangiu, Bogdan Raducanu, Andrei Cristian Spataru, „High-Precision Orientation and Skew Detection for Texts in Scanned Documents", Proceedings of the 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing", Cluj-Napoca, Editor Ioan Alfred Letia, August 27-29, pp.145-148, 2009.

[10] Costin-Anton Boiangiu, Alexandru Topliceanu, Ion Bucur - "Efficient Solutions for OCR Text Remote Correction in Content Conversion Systems" , Journal of Control Engineering and Applied Informatics, Volume 15, No. 1, pp 22-32, 2013

[11] Costin-Anton Boiangiu, Dan-Cristian Cananau, Serban Petrescu, Alin Moldoveanu, "OCR Post Processing Based on Character Pattern Matching", 20th DAAAM World Symposium, pp. 141-144 Austria Center Vienna (ACV), 25-28th of November 2009.