Efficient Solutions for OCR Text Remote Correction in Content Conversion Systems

Costin-Anton Boiangiu, Alexandru Topliceanu, Ion Bucur

University "Politehnica" of Bucharest, Splaiul Independenţei 313,
Bucharest, 060042, Romania
(e-mail: {costin.boiangiu,alexandru.topliceanu}@cti.pub.ro, ion.bucur@cs.pub.ro)

Abstract: This paper describes a collection of algorithms for detecting text areas in document images using morphological operators, text clustering using geometrical text measurements and efficient image coding for fast remote correction in automatic content conversion systems Text characteristics are automatically discovered and used to filter out all non-text areas in the image. All the algorithms were implemented and tested on a representative set of test images obtained by scanning newspapers, books and magazines. The document image page clustering uses a measure of normalized text font resemblance. The approach makes use solely of the geometrical characteristics of characters, ignoring information regarding context or character recognition.

Keywords: automatic content conversion, image morphology, text clustering, text detection, text extraction, text remote correction, font resemblance, OCR

1. INTRODUCTION

Content conversion systems are generally aimed at the extraction of the informational content from various types of media, usually printed form, and convert this information into a digital format, with numerous advantages. Content conversion processes generate easy-to-find, available information from an otherwise limited starting point. For example, a book in a library somewhere in the world is a lot more difficult to access that its digital version published on the Internet.

Content conversion processes involve complex operations with a high degree of flexibility, among many other requirements, such as computing power and large scale storage solutions. The need for high algorithm flexibility comes from the variability of the input, as no two input documents are the same. Some documents may require a special set of parameters for color conversion or other operations (Boiangiu et al. 2009) such as page layout detection, extraction of paragraphs, line detection etc. But however flexible these content conversion algorithms may be, errors will continue to appear, as only a small fraction of documents are available for testing, and new scenarios with specific issues appear constantly.

A complete content conversion system offers the benefit of Quality Assurance (QA) of the processed documents. The QA process comes in the form of a remote correction activity, consisting of documents that have been automatically processed being sent to another location, where human operators ensure the correctness of the results. The QA process however, comes with some added difficulties mainly regarding the size of the documents to be transferred (sometimes documents have to be sent to another continent for verification). In order to address the problem of document size and network bandwidth usage, the files to be sent have to undergo a number of preparation stages. This paper deals with the most important issues involved in the remote correction of documents resulting from a content conversion process, and proposes a number of solutions backed by results obtained from testing.

There is a very high demand for automated systems of detecting text areas in images (Harman et al. 1995), no matter the way these images have been obtained. There are multiple reasons for this, the one that drove our research was the prospect of using an optimized algorithm for compression in portions of the image that are text-only so that the OCR text can be send to correction alongside with the target image in a remote QA process. The challenge is to discern the text areas from images, symbols and other printing specific non-text glyphs. All text must be located, no matter the font type, size, and weight. We assume that the document images are not skewed, otherwise a deskew operation (Chen et al. 1994; Raducanu et al. 2011) is required as a preprocessing phase.

2. TEXT RECOGNITION ALGORITHM

The proposed algorithm is based on the observation that all the text in a document image has a limited number of reoccurring heights, due to the use of the same dominant fonts. As opposed to text, the composing connected elements of an image can have various unique heights and widths. Text characters and words fall within a certain range of heights, given by the text type, set height and font used. The algorithm discovers the most frequent heights in the elements of the image and states that those elements are words, whereas singular heights correspond to images (Chen et al. 1996; Das et Al. 2001). This gives the criteria by which to clear the image portions of the document leaving only the text areas where post-processing can be performed.

The algorithm works in 7 steps. An assumption is made that input images are high resolution (anywhere from 2K by 2K

up to 10K by 10K pixels or more) image scans of documents. Images are binarized: text and image pixels are classified as foreground and as a result marked as black whilst white pixels are considered as background. Adapters to this format of input data are easy to implement, whatever the initial file type.

Fig. 1. Initial test image from a newspaper.

The first step consists of using morphological transforms (Chen et al. 1994; Thanh et al. 2007) to merge black pixels – fig. 2. The aim is to create as many large connected objects as possible and to eliminate possible rebel pixels. An opening transform with a horizontal line structural element combined with a closing transform by a small disk-shaped structural element is used for that purpose. The opening is done with a $xDPI/10$ pixels width by 1 pixel height rectangle structural element which gives the result of binding letters of a word into a single element. Linking of words is sometimes obtained, due to small height fonts, but this does not change the result of the algorithm. The closing transform uses a $xDPI/100$ pixels disk-shaped structural element (where $xDPI$ represents the scanning resolution measured in Dots-Per-Inch along the X-axis) and is important because it splits the accidentally connected elements, especially when words are linked with images through narrow isthmuses. If a word was to be connected to an image after opening, the resulting element would be interpreted as an image and filtered out in our algorithm. In case of images, this step results in very large black objects. As for text areas, long parallel thick lines representing words and sentences are obtained. These long parallel lines of relative equal height that represent the text areas are the desired result and the rest of the algorithm identifies them.

Fig. 2. Morphological transformations applied to the image in step 1.

Steps two and three consist of labeling each connected black element in the image – Fig. 3, then calculating widths and heights for each element. These metrics are rounded so as to account for the slightly variable element widths resulted in the previous step. It is important to note that the width and height of the enclosing ellipse is measured for each element, not the actual width and height of that element. This allows for a better clustering of similar elements.

Step four is where the frequency by which element heights appear is counted. For each individual height we count all the elements that have that approximate height. This is very important in our algorithm as we will filter the elements with heights of a lower rate of appearance and keep only the elements that have a common (high rate of appearance) height. In our research we have obtained the optimal results when using a threshold of 5. If we found more that 5 elements with the same height we state that they correspond to words and keep them in the resulting image. This is done in phase 5. Phase 6 finds the connected elements that haven't passed our predefined threshold and erases them (makes them part of the background), leaving only the text areas colored in black. We call this a bit mask; an example of this is in fig. 4.

Fig. 3. Labeling of connected elements in step 2. Each element has a different color.

Fig. 4. Bit mask obtained in step 6. All images have been eliminated.

Before applying the previously computed mask to the initial image so as to return an image containing only text, we have observed that isolated small elements have been preserved by the filtering phase. This is due to the fact that scanned articles are tainted with Gaussian noise. Noisy elements of equal height can appear and might pass the previous filters. These could be filtered out by increasing the threshold in the previous step, but that would risk eliminating words with special low-occurring font like titles, subtitles and footers. The decision was to keep the threshold low and to eliminate these unwanted elements through the use of morphological transforms. An extra step in the algorithm has been added for this. It was found that the best combination to eliminate them and to preserve the form of the wanted elements (e.g. the long thick lines representing text) is a sequence of close and erodes transforms using the same square structural element of $xDPI/30$ pixels in height.

The seventh and final phase consists in applying the mask to the initial image. Given that both the mask and the initial image are represented as arrays of Boolean values (binary images), a simple method of applying the mask by using the AND logical operator between corresponding pixels in the two images was used.

2.1. Text Recognition Results

In the testing phase, it was used a relatively small set of 100 test images covering various types of scanned documents: newspaper front-pages, newspaper articles where text is dominant, magazine pages with high concentration of images and small areas of text, receipts, etc. These image documents are binary, have small degrees of noise and negligible skew.

The algorithm was capable of locating columns of text and removing large images – Fig. 5. The algorithm's many parameters described in the above section can be easily adapted for various cases. However, the aim of the research was to find optimal values for these parameters that offer acceptable results in all test cases. For that purpose, a scoring technique was used to evaluate the results and the parameter set with the best average score was used in the following processing.

Due to the nature of the algorithm, words printed in fonts and sizes that are not used everywhere else in the image are removed. Sadly this includes parts of article's titles and headers. Moreover when morphological transforms blend text areas with images, the result is filtered by the algorithm and the corresponding text does not appear in the finished image.

This behavior can be dimmed by modifying the closing transform's structural element from the first phase, lowering its width. This improves the results for some images but worsens results for other images. Tests have revealed that the optimal dimensions for the structural element are $xDPI/10$ pixels width by 1 pixels height.

Fig. 5. Bit mask applied to the initial image in step 7. Image contains only text areas.

Image elements are successfully removed by the presented solution, yet small debris from these images remain after the morphological operators in the first stage. This happens because the morphological opening transform does not manage to blend all the pixels in an image into one connected block. Some of these elements are eliminated by closing the image in the first step, others are filtered in the sixth phase, but further filtering is needed. Thus, an extra morphological close transform was introduced on the image after the sixth phase.

Another issue that appeared was the poor quality of the resulted text. The text areas are located correctly, but their width and height does not take into account that some letters might have accents, or some letters like f, t, p and q might be larger than the element they have been blended into. A simple solution to this problem is to morphologically dilate the connected elements in the bit mask resulted in phase 6. A rounded structural element with a 10 pixels diameter was found to be optimal, as it had to keep neighboring elements separated.

For document images where pictures are dominant, such as magazine front-page scans, results are generally poor. Because of the fact that images are blended with the text in these scans, our algorithm merges the text areas with picture areas and then filters them all. Some text zones are discovered but they are unintelligible and sparse. Also due to the fact that magazines use color photos in their front pages, the result of the binarization process is that some text may not appear black with white background, but white with black background. In the presented algorithm we have assumed that all text and images are black and the background is white.

To minimize the presented inconveniences in the text detection and extraction algorithm a text clustering scheme is employed using the freshly obtained results (Yu 2000; Hull 1997). It is up to this clustering approach to recover text information in a consistent manner so that different text lines and paragraphs are segmented correctly and that the entire text regions are extracted and/or marked in the image.

3. TEXT NORMALIZATION AND CLUSTERING ALGORITHM

Text normalization is a process by which text is transformed in some way to make it consistent in a way which it might not have been before. Text normalization is often performed before text is processed in some way, such as generating synthesized speech, automated language translation, storage in a database, or comparison. Several algorithms were used to achieve this: binary image transformation (Boiangiu et al. 2011), connected components labeling, merge filter, width filter, inside filter (Boiangiu et al. 2008a, b).

After the text normalization process the similarity detection begins. The points of resemblance targeted are the following: font size, font boldness, line spacing, font italics.

Image preparation is necessary in order to label the connected components. In the end, each component is colored differently for better a view. In fig. 6 is represented a test image and the connected components with their initial bounding boxes.

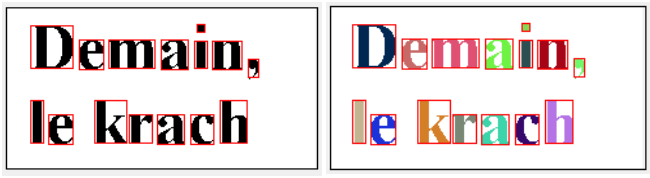


Fig. 6. Initial image and connected components.

3.1 Inside Filter

Scanned documents usually have noise problems, lower quality and so on. This can generate a series of unconnected letters, spots of color and so on. To be able to properly analyze the text all letters must be correctly identified and as much as possible the noise must be eliminated.

The "inside filter" checks whether a component is inside another. In fig. 7 are presented some results of this filter execution applied onto a common set of "non-ideal" scanned characters.

All filters make use of the component's bounding box (the enclosing rectangle). The edges are tested to check for coverage.

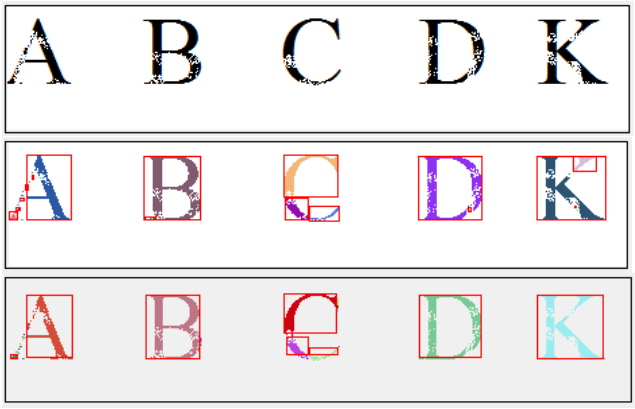


Fig. 7. Corrupted letters, initial connected components and connected components after inside filter.

3.2. Merge Filter

An improvement was made by applying the inside filter, but the letters still aren't completely defined. Merge filter checks if two or more components are connected vertically. In our example, the letter C is split in 3 components divided vertically.

The algorithm makes use of the average letter height. A margin of 10% of that average is used to merge the components together.

Fig. 8. Components after "merge filter".

3.3 Width Filter

Noise cancellation is another important step in normalization. The bounding box of each component is computed. If the width to height ratio is higher than 80% the component is removed as noise. Punctuation marks are often removed as well as a result. Once all bounding boxes have been computed, components extracted text analysis can begin.

3.4 Similarity Measure between Texts: Font Size

In order to determine the font size in a text, a histogram of the heights of connected components within the document is created. For the resulting histogram, is counted the number of peaks. According to this count, some conclusions can be drawn:

- If the histogram contains less than two peaks, it follows that the text is written in capital letters (font capital letters are usually all the same height).
- Otherwise, the text is written in mixed capital and lower case letters. Histogram peaks represent: capital letters, lower case letters and punctuation. Normally, a mixed-case text will contain proportionally more lower case letters than higher case letters, thus, the histogram's highest peak will belong to lower case letters.

3.5. Similarity Measure between Texts: Font Boldness

Two font boldness detection methods have been implemented, both generating robust results.

The first of these computes the ratio between the number of pixels that belong to the contour of a connected component and the total number of pixels belonging to the connected component. Using these percentages, a histogram is created and the most frequent item is selected.

The second implemented method that test for font boldness is the "crosshair" method. This method actually measures the width of each letter. For each black pixel inside the bounding box for each connected component, is measured the vertical and horizontal segments that pass through (and intersect each other in) this pixel. In order to measure the length of the vertical segment for the pixel, we check the color of the pixel directly above it. If black, we increment the length counter and check the next pixel above. This continues until we encounter a white pixel. The horizontal segment is measured in a similar manner. Of the two resulting segments, the shorter one is chosen. With these values a histogram is built for each connected component, finally obtaining the width for this letter (the most frequent item in the histogram). Finally, we create a histogram of pen widths for each letter in the document and select the one with the highest frequency.

After applying this algorithm for two documents, the higher resulting value marks the document with the bolder text.

3.6. Similarity Measure between Texts: Font Italics

Determining the italics characteristics of font can be made using two main algorithms: width and chain. The choice was the width method due to the importance of the bounding box.

Typically italic font has an inclination degree of 16. For this reason we measure the width of the letter after applying a rotation of 16 and -16 degrees. Italic characters are wider than non-italic one, so those letters that are thinner after the rotation can be categorized as italic. The chain algorithm compares the longest vertical line in the original character and in the rotated one. If the length is smaller after the rotation, then the character was italic.

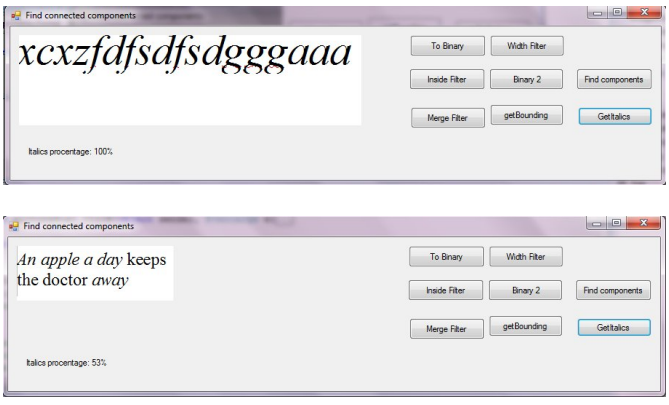


Fig. 9. Results of italic measurements: computed percentages of 100% (upper) and respectively 53% (lower).

3.7. Line Spacing

Spacing between two lines was measured by averaging the distance between entities found on consecutive lines and approximately on the same column (one beneath the other).

Initially, the average height of connected components within the document was computed, and using this average, the line to which each component belongs was determined.

Next, the connected components were sorted in increasing order of the size of their bounding box's lower side. If the vertical distance between two bounding box bottom segments is greater than the computed average height, we have passed on to another line of text.

Connected component coordinates on each line of text are kept in the lines of a matrix. Components are tested if they are approximately on the same column by parsing the matrix lines two by two and checking the x axis coordinates of their bounding boxes. If these are not approximately on the same column, the distance is discounted when computing the average distance.

3.8. The Font Resemblance Algorithm

The bounding box coordinates are computed for each connected component in every image.

Next, for each image is performed a count of the number of peaks in the height histogram and decide whether the text is written in all upper case or mixed upper case and lower case, determine the height of a lower case letter and/or of a upper case letter and line spacing.

A similarity ratio is computed between the two images according to results computed thus far. If this ratio is greater than 50%, then the two boldness tests are applied: based on the contour of each connected component and crosshair; otherwise it is determined that the two fonts are too different and no further tests are applied.

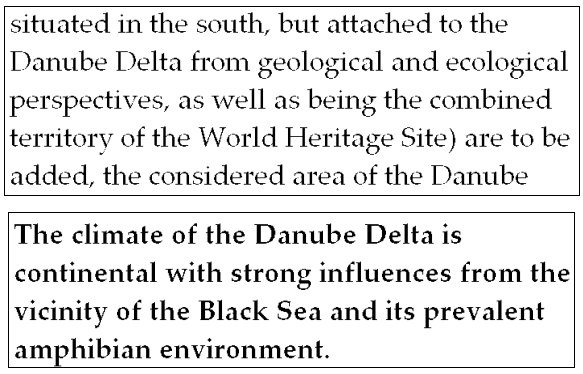


Fig. 10. Input texts for comparison (Test 1).

Table 1. The algorithm applied on an image with bold text and a image with normal text.

	Text Up	Text Down	Conclusions	
Number of peaks	3	4	Low caps & high caps	Low caps & high caps
Low Caps (pixels)	21	24		
High Caps (pixels)	28	30		
Line Spacing (pixels)	33	30		
Font size match	90.58%			
Boldness contour (%)	100	70	Image 2 has bold text	
Boldness crosshair (pixels)	2	2		

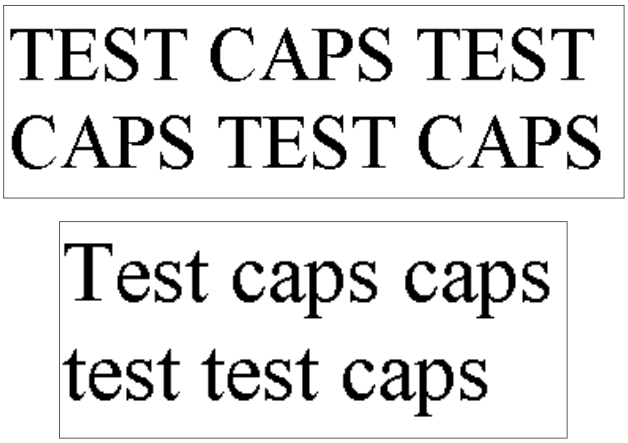


Fig. 11. Input texts for comparison (Test 2).

Table 2. The algorithm applied on an image written only with high caps and an image with low and high caps.

	Text Up	Text Down	Conclusions	
Number of peaks	1	2	High caps only	Low caps & high caps
Low Caps (pixels)	0	27		
High Caps (pixels)	36	33		
Line Spacing (pixels)	23	30		
Font size match	56.11%			
Boldness contour (%)	-	-		
Boldness crosshair (pixels)	-	-		

4. ONLY-TEXT IMAGE COMPRESSION METHODS

A highly effective solution to reduce the size of image data for remote correction purposes is to use a well-established format that has a compression profile suited for the task at hand. In the following section, the most relevant image compression formats will be detailed, compared and also combined with some image configurations that will improve compression.

At this point in the presented research using the text extraction algorithm that was presented, some text areas can be extracted. In order to fully format the text and reject the unwanted data, the text extracted is clustered using the algorithm presented in the second stage.

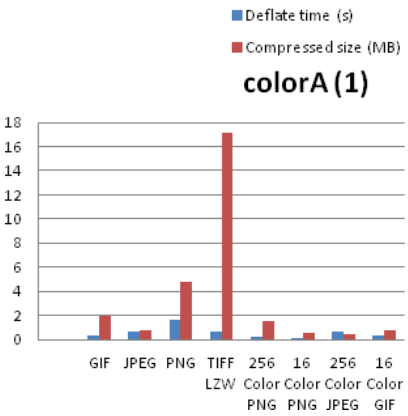
As a result, after the two presented approaches are run, valid cluster of texts are found on the image. An OCR engine is run using the selected areas and in order to reach our initial goal (i.e. to remote correct the OCR-engine output) is needed that the content conversion remote operator to receive fast and accurate both the OCR result and the original text found in the image. The image text should be readable and should be transferred very fast because the remote correction is a time-critical process. This is why the entire document (which may contain huge collections of scanned pages at 10K x 10K points or more per each page) cannot be sent remote, instead, a lower resolution image which contains only the text information is sent instead. The problem that arises now is how to compress (eventually preprocess) the image so that it is still readable remote, it transfers fast and decompresses almost instantaneously so that the remote operator can be productive in his/her text correction task.

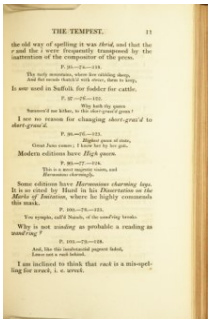
In order to compare results obtained by different compression approaches, there have been considered the size of the image after compression ("compressed size"), and the time the image takes to load/decompress ("load time").

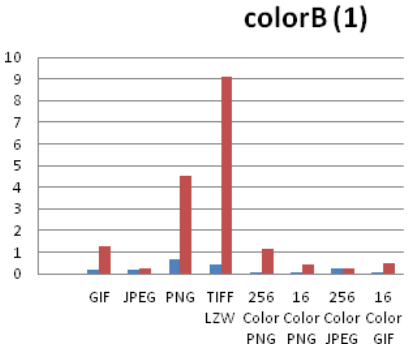
In order to determine best compression settings, the input images have been chosen to be of 3 types: true-color 24-BPP, grayscale 8-BPP and black & white 2-BPP. The image content is significant to our purpose, extracted from various British Library books and newspapers. Several tests were performed with the original images cropped to the union of the bounding rectangles of the text areas and with the background erased to a dominant color:

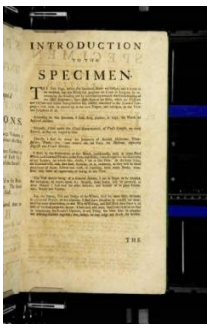
- Color images:
  - CompuServe GIF : no extra settings
  - JPEG : at "normal" compression
  - PNG : at "best" compression
  - TIFF : LZW
  - 256 and 16 Color PNG : reduce number of colors to 256 and 16 and compress using PNG
  - 256 Color JPEG Lossless: reduce number of colors to 256 and compress using JPEG Lossless
  - 16 Color GIF : reduce number of colors to 16 and compress using GIF
- Grayscale images:
  - GIF
  - JPEG
  - TIFF LZW
  - 16 Color PNG
  - 16 Color GIF
  - 16 Color JPEG
- Black & White images:
  - GIF
  - PNG/TIFF Group Fax 4
  - TIFF LZW
  - JBIG2 (just as a compression ratio reference)

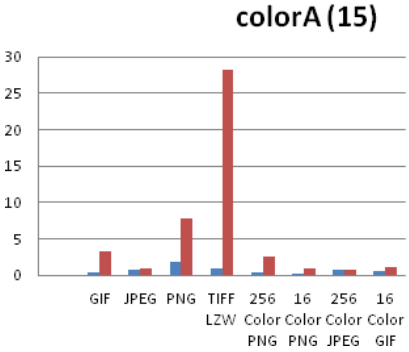
Unfortunately, the JPEG 2000 format (which for sure would have proved to be the most space efficient for a desired quality by all of the image compression formats) has not been included between candidates. The reason for this is that JPEG 2000 despite the fact that it is a very modern and extremely strong image compression algorithm; it is also a very slow one. The compression is very time consuming but what bothers most in this case is that the decompression time is unacceptable high for a system that should ensure fast OCR remote correction.

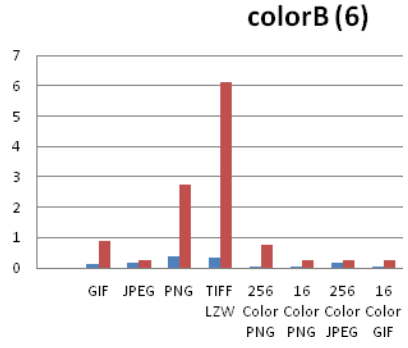


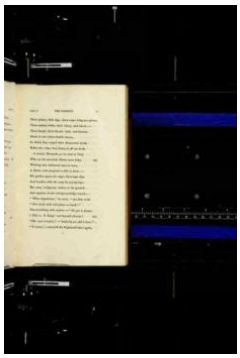


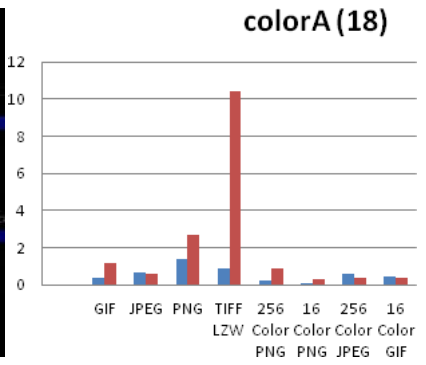


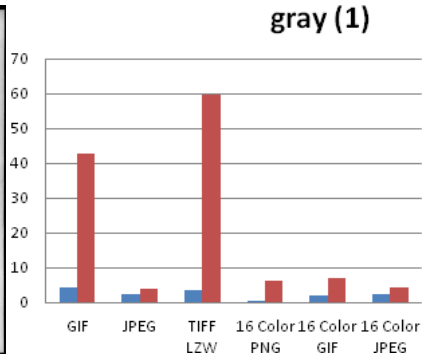


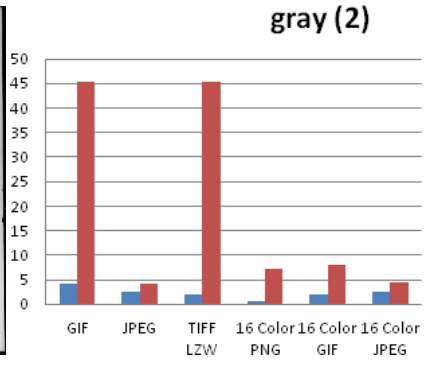


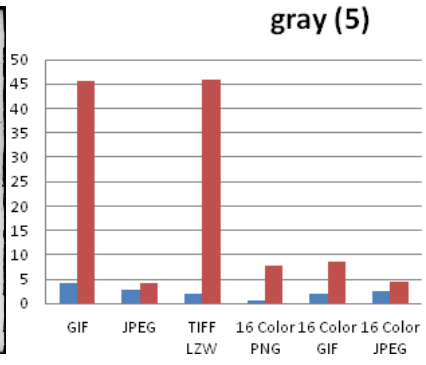


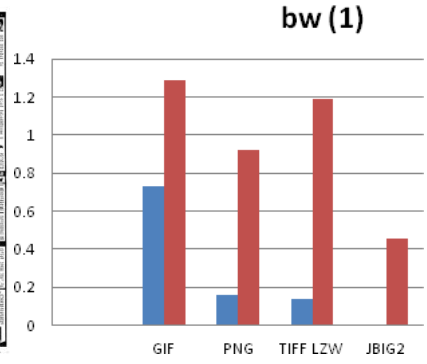


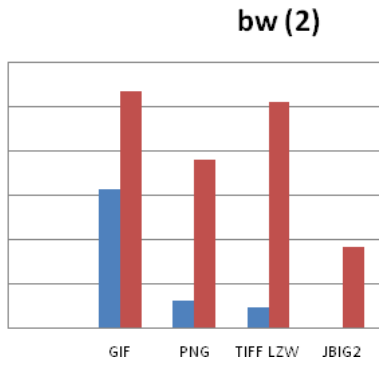


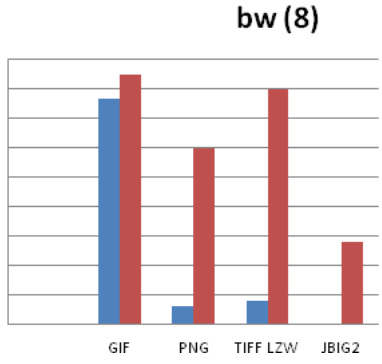












Below are presented results for significant images of the 3 types and the corresponding compressions.

4.1. True-Color Remarks

- The compression time for PNG is the largest of all methods.
- PNG and GIF work a lot better on images with a reduced number of colors, but still easily readable.
- The worst results are obtained using TIFF-LZW.
- The best results, both in loading time and disk space came from the preprocessed images, using color reduction. As it can be seen in the graphs, the results from 16-Color PNG and 256-Color JPEG Lossless are the best for Color images, but images in 256-Colors are of slightly higher quality. The 16-Color PNG has the lowest deflate time of all methods, and can be used if speed is more important than quality.
- Depending on the need, a choice can be made between **16-Color PNG, 256-Color JPEG lossless and 16-Color GIF**, according to the graphs.

4.2. Grayscale Remarks:

- In the case of grayscale images, the color reduction does not improve the compression and speed capabilities of other formats beyond those of the standard JPEG.
- The PNG compression time for files of size similar to the tested images (around 60-70 MPixels) is very high, and so the unprocessed PNG compression method has not been included in this comparison.
- The worst results are those of TIFF-LZW and GIF formats.
- The best overall results are obtained using standard JPEG compression. Again, PNG has the fastest load time, but occupies more disk space. Reducing colors to 16 and compressing by JPEG does not improve

performance, and so the best options are standard **JPEG and 16-Color PNG**.

4.3. Black & White Remarks

- Because JBIG2 is quite an uncommon format, the computation of the loading time is not available for this format.
- The worst results are obtained using GIF.
- If the compatibility of the JBIG2 is not a problem, this format is the most space-effective. Unfortunately, the compression and decompression time is much higher than any of the other candidates, fact that is making JBIG2 only a reference of compression ratio but unusable in fast remote correction in content conversion projects. The alternative is to use PNG, as it has a fast loading speed and also the lowest size from the remaining candidates. So, the best results for B&W are **JBIG2 and PNG**.

Overall, on our small test benchmark comprised of about 100 images from all kind of publications a gain of about 65% in compressed size when comparing the proposed approach with the full image coding using JPEG standard with average quality (current digitization industry standard). This may depend in a real-life scenario of the average document area occupied by illustrations, text compactness (compact text may result in smaller areas to be sent) and contrast (the better the contrast, the fewer colors may be used in palletized conversions).

5. CONCLUSIONS AND FUTURE WORK

The algorithms presented in this paper offers a solution to isolate text from scanned document images, a measure for classification and clustering of text lines and a proposal for a simple and efficient compression scheme that enables fast and reliable remote correction of OCR-resulted text. After all the steps of the algorithm are executed it may result in a significant reduction (almost two thirds in our benchmark test) of the data size to be sent in the remote OCR correction, with virtually indistinguishable processing time penalty.

The primary intent of the text selection algorithm is to eliminate images and other non-text areas in the printed document. This specification is accomplished but at the cost of sometimes eliminating some areas of text, especially headers, titles and footers. Isolated words are also sometimes filtered out. Some of this problems and misclassifications may be solved in the future by employing an improved font resemblance and clustering scheme.

ACKNOWLEDGMENTS

The work presented in this paper was funded by the Sectorial Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the financial agreement POSDRU/89/1.5/S/62557.

REFERENCES

Boiangiu, C.A., Spataru A.C., Dvornic, A.I. and Cananau D.C. (2008). Normalized Text Font Resemblance Method Aimed at Document Image Page Clustering". *WSEAS Transactions on Computers*, Issue 7, Volume 7, pp. 1091–1100, ISSN: 1109-2750.

Boiangiu, C.A., Spataru, A.C., Dvornic, A.I., and Cananau, D.C. (2008). Automatic Text Clustering and Classification Based on Font Geometrical Characteristics, *Proceedings of the 9th WSEAS International Conference on Automation and Information*, WSEAS Press, pp. 468 – 473, Bucharest, Romania, ISBN 978-960-6766-77-0, ISSN 1790-5117

Boiangiu, C.A. and Cananau, D.C. (2009). Combined Approaches in Automatic Page Clustering for Content Conversion, chapter in *DAAAM International Scientific Book 2009*, pp. 289-304, ISSN 1726-9687, ISBN 978-3-901509-69-8 Editor: Brako Katalinic, Vienna.

Boiangiu, C.A., Olteanu, A., Stefanescu, A., Rosner, D., Tapus, N. and Andreica, M. (2011), Local Thresholding Algorithm Based on Variable Window Size Statistics, *Proceedings CSCS-18, The 18-th International Conference on Control Systems and Computer Science*, Bucharest, Romania, Volume 2, pp. 647-652, Politehnica Press, ISSN: 2066-4451.

Chen, F.R., Bloomberg, D.S. and Wilcox, L.D. (1996). Detection and Location of Multi-Character Sequences in Lines of Imaged Text. *Journal of Electronic Imaging*, Vol. 5, pp. 37-49.

Chen, S. and Haralick R.M. (1994), An Automatic Algorithm for Text Skew Estimation in Document Images Using Recursive Morphological Transforms, *Image Processing, 1994. Proceedings ICIP-94., IEEE International Conference*, Vol. 1, pp. 139 – 143.

Das, A.K. and Chanda B. (2001). A fast algorithm for skew detection of document images using morphology, *The International Journal on Document Analysis and Recognition*, Springer-Verlag.

Harman, D., Buckley, C., Callan, J., Dumais, S., Lewis, D., Robertson, S., Smeaton, A., Jones, K.S., Tong, R., Salton, G. and Damashek, M. (1995). Performance of Text Retrieval Systems. *Science*, pp. 1417-1420.

Hull, J.J. and Cullen, J.F. (1997). Document Image Similarity and Equivalence Detection. *Proceedings 4th International Conference on Document Analysis and Recognition, ICDAR'97*, Ulm, Germany, Vol. 1, pp. 308-312.

Raducanu, B., Boiangiu, C.A., Olteanu, A., Ştefănescu, A., Pop, F. and Bucur, I. (2011) Skew Detection Using the Radon Transform. *Proceedings CSCS-18, The 18-th International Conference on Control Systems and Computer Science*, Bucharest, Romania, Volume 2, pp. 653-657, Politehnica Press, ISSN: 2066-4451.

Thanh, N.D., Binh, V.D., Mi, N.T.T., and Giang, N.T. (2007). A Robust Document Skew Estimation Algorithm Using Mathematical Morphology. *The 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI '07*, pp. 496-503.

Yu, Z. (2000). Similarity measure of text images, *M.Sc Thesis*, School of Computing, National University of Singapore.

Appendix A. True-Color Image Compression Results.

Image Name	Color Space	Format	Uncompressed size	Compression method	Deflate time (s)	Compressed size (MB)	Compression ratio
colorA (1)	Color 24BPP	3328 x 4992	47.53 MB	GIF	0.328	2	4%
				JPEG	0.609	0.736	2%
				PNG	1.641	4.78	10%
				TIFF LZW	0.672	17.09	35%
				256 Color PNG	0.172	1.49	3%
				16 Color PNG	0.125	0.587	1%
				256 Color JPEG	0.625	0.486	1%
				16 Color GIF	0.297	0.748	2%
colorB (1)	Color 24BPP	1505 x 2500	10.77 MB	GIF	0.188	1.26	10%
				JPEG	0.156	0.222	2%
				PNG	0.656	4.52	50%
				TIFF LZW	0.406	9.11	84%
				256 Color PNG	0.063	1.12	10%
				16 Color PNG	0.046	0.405	3%
				256 Color JPEG	0.25	0.232	2%
				16 Color GIF	0.078	0.461	4%
colorA (15)	Color 24BPP	3328 x 4992	47.53 MB	GIF	0.438	3.31	6%
				JPEG	0.766	1	2%
				PNG	1.907	7.8	16%
				TIFF LZW	0.891	28.3	59%
				256 Color PNG	0.313	2.5	5%
				16 Color PNG	0.094	0.939	2%
				256 Color JPEG	0.703	0.737	1.50%
				16 Color GIF	0.485	1.11	2%
colorB (6)	Color 24BPP	1691 x 2555	12.37 MB	GIF	0.109	0.88	20%
				JPEG	0.187	0.258	2%
				PNG	0.391	2.75	22%
				TIFF LZW	0.328	6.11	49%
				256 Color PNG	0.047	0.757	6%
				16 Color PNG	0.031	0.244	2%
				256 Color JPEG	0.188	0.259	2.00%
				16 Color GIF	0.062	0.252	2%
colorA (18)	Color 24BPP	3328 x 4992	47.53 MB	GIF	0.344	1.15	2%
				JPEG	0.688	0.591	1%
				PNG	1.39	2.71	5%
				TIFF LZW	0.875	10.42	21%
				256 Color PNG	0.235	0.889	2%
				16 Color PNG	0.109	0.274	< 1%

Image Name	Color Space	Format	Uncompressed size	Compression method	Deflate time (s)	Compressed size (MB)	Compression ratio
				256 Color JPEG	0.609	0.353	< 1%
				16 Color GIF	0.406	0.401	1%

Appendix B. Grayscale Image Compression Results.

Image Name	Color Space	Format	Uncompressed size	Compression method	Deflate time (s)	Compressed size (MB)	Compression ratio
gray (1)	Grayscale 8BPP	7130 x 8765	59.6 MB	GIF	4.343	42.77	70%
				JPEG	2.531	4.04	6%
				TIFF LZW	3.422	59.6	100%
				16 Color PNG	0.625	6.44	10%
				16 Color GIF	1.938	7.16	12%
				16 Color JPEG	2.391	4.2	7%
gray (2)	Grayscale 8BPP	7204 x 8744	60 MB	GIF	4.297	45.29	75%
				JPEG	2.656	4.25	7%
				TIFF LZW	2.109	45.34	75%
				16 Color PNG	0.672	7.27	12%
				16 Color GIF	2.016	8.03	13%
				16 Color JPEG	2.656	4.55	7%
gray (5)	Grayscale 8BPP	7164 x 8797	60 MB	GIF	4.265	45.73	76%
				JPEG	2.703	4.25	7%
				TIFF LZW	1.906	45.82	76%
				16 Color PNG	0.75	7.75	13%
				16 Color GIF	2.109	8.54	14%
				16 Color JPEG	2.547	4.59	7%

Appendix C. Binary Image Compression Results.

Image Name	Color Space	Format	Uncompressed size	Compression method	Deflate time (s)	Compressed size (MB)	Compression ratio
bw (1)	BW 2BPP	4634 x 6461	3.57	GIF	0.734	1.29	36%
				PNG	0.156	0.919	25%
				TIFF LZW	0.141	1.19	33%
				JBIG2	N/A	0.456	12%
bw (2)	BW 2BPP	3504 x 4960	2.08	GIF	0.313	0.534	25%
				PNG	0.063	0.381	18%
				TIFF LZW	0.047	0.511	24%
				JBIG2	N/A	0.183	8%
bw (8)	BW 2BPP	4944 x 6249	3.69	GIF	0.766	0.844	22%
				PNG	0.062	0.597	16%
				TIFF LZW	0.078	0.796	20%
				JBIG2	N/A	0.281	7%