Chapter 1
Document Image Segmentation through Clustering and Connectivity Analysis

Mihai Bogdan Ilie

Abstract. This chapter presents a new document image segmentation algorithm, called Cluster Variance Segmentation (CVSEG). The method is based on the analysis of the tiles suspected to be part of an image and filtering them subsequently. In the end, the results are enhanced through a reconstruction stage. I present the design of the algorithm as well as the test results on various document images. The experiments validate the efficacy and efficiency of the proposed approach when compared with other algorithms.

1.1 Introduction

In the context of the current stage of document generation and recording, there is very much interest shown in the Document Analysis and Retrieval (DAR) field, which is a viable solution to automatically process and classify the documents in a specific area. There are many problems that have been addressed by DAR researchers all over the world, like:

- extracting the text from documents written in different languages with different characters [1];
- document sorting according to keywords [2];
- document sorting according to the layout;
- web crawling [3];
- automatically processing official forms [4];
- signature and stamp detection [5];

Mihai Bogdan Ilie
"Dunarea de Jos" University of Galati,
Faculty of Automatic Control, Computers,
Electrical and Electronics Engineering,
Stiintei Street 2nd, Galati, Romania
e-mail: mihai.ilie@ugal.ro

A. Zgrzywa et al. (eds.), *New Research in Multimedia and Internet Systems*,
Advances in Intelligent Systems and Computing 314, DOI: 10.1007/978-3-319-10383-9_1

- distinguishing between two different languages [6];
- many others.

The sub-problems encountered in the document processing area are complex and are originated in different areas:

- scanning conditions;
- noise determined by the page curvature;
- poor page illumination;
- degraded physical support;
- stroke size;
- different languages and characters etc.

The implementations vary from solving the basic problems of document processing up to complete, sophisticated software solutions, completely connected. A common approach (especially in the areas which require a classifier, but not only) is to use artificial intelligence techniques. Among these, the most common during the classification stage are the neural networks and the support vector machines. Besides these, there are implementations that make use of genetic algorithms [7], unsupervised learning [8], swarm intelligence, Markov random fields [9], fuzzy modules [10] or self organizing feature maps [11].

One of the problems addressed by the DAR area is the automated extraction of images from documents. Next, this information can be used for multiple purposes, like plagiarism detection, document classification according to the image content or according to the logo.

1.2 Related Work

The main purpose of the DAR area is to recognize the text and the graphical elements in a document scan, as a human would. The DAR field includes multiple research directions, like:

- binarization,
- noise reduction,
- segmentation,
- OCR,
- skew estimation,
- many others.

From all of the above I am mostly interested in the segmentation area, especially in image segmentation.

There are many possibilities to classify the current DAR approaches but one of the most complete studies establishes the below algorithm taxonomy [12]:

- based on image characteristics – extracting local and global descriptors for colour, shape, texture, gradient etc.,
- based on the physical structure – establishing the document geometrical hierarchy,

- based on logical characteristics – establishing the document logical hierarchy,
- based on text characteristics – extracting keywords, based on an OCR algorithm.

In what regards the classification strategy, the approaches can be classified as below:

- bottom-up – which start by analysing pixels, regions or connected tiles; the resulted objects are then merged and classified as document areas;
- top-down – which start analysing the document from the encompassing scan image and then split it in unit regions.

Regardless of the techniques described above, the authors agree on the below stages involved in the process of obtaining the desired results:

- binarization,
- noise reduction,
- segmentation,
- thinning,
- chain coding and vectorization.

In the document segmentation area, most of the researchers target text segmentation in all its variations - block, paragraph, line, word and character segmentation. There are not many image segmentation algorithms; among them, probably the best known is the one implemented by Bloomberg [13], based on eroding the image in order to eliminate the text and then dilating it in order to enhance the image features. Subsequently, this algorithm got improved by Syed Saqib Bukharia [14], who introduced some additional steps, refining the two stages.

I propose a bottom-up image segmentation approach, based on image characteristics, targeting the document image segmentation.

1.3 New Approach

The algorithm (CVSEG) is based on decomposing the document in tiles, clustering and filtering them based on their inner variance and connectivity.

The image segmentation process is split into several phases. First of all, the algorithm assumes that the image has been previously binarized. In order to achieve its goal, the algorithm extracts the pixel grid, splits the image in tiles of a certain size and goes through the below stages:

1) text filtering, in order to facilitate processing the remaining areas. Currently I am using a simple algorithm, based on XY axis projection [15]. I looked for a simple method which would be computational fast and that would manage to eliminate the text up to some extent. This step works well on documents which are correctly aligned with the axis; the results are shown in the image below (Fig. 1.1). However, if the text is not completely filtered, it will be excluded from the final result due to its poor variance score.

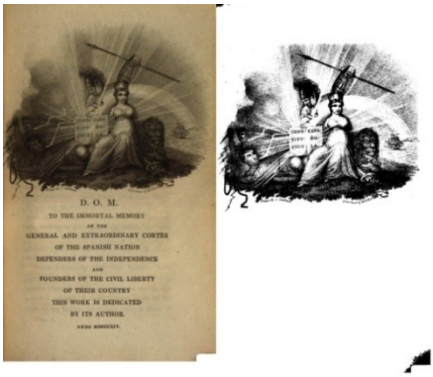

Fig. 1.1 Text filtering

2) calculating the average pixel intensity for each tile.

$$G_L = \frac{1}{w*h}\sum_{i,j} p_{i,j}\,, the\ average\ gray\ level \tag{1.1}$$

$$\Delta_G = \frac{1}{w*h}\sum_{i,j} |p_{i,j} - G_L|, w\ and\ h\ represent\ the\ tile\ width\ and\ height$$

$$\Delta_G = \frac{1}{w*h}\sum_{i,j} |p_{i,j} - G_L|, w\ and\ h\ represent\ the\ tile\ width\ and\ height$$

3) calculating the average variance in each tile. Based on this score, I am establishing whether a certain tile may be part of an image or not ($\Delta_G > T_G$ and $\Delta_C > T_C$). The decision criteria is that in a particular tile, an image tends to be more uniform than the text. The thresholds have been set through experiments at $T_G = 0.1$ and $T_C = 0.37$.

$$C_L(p_1, p_2) = \begin{cases} 0, if\ p1 = p2 \\ 1, if\ p1! = p2 \end{cases} \tag{1.2}$$

$$\Delta_C = \frac{1}{w*h}\sum_{i,j}\frac{1}{N_{i,j}}\sum_{m,n} C_L\left(p_{i,j}, p_{m,n}\right), m! = i, n! = j$$

$$N_{i,j} - \ total\ neighbours\ of\ the\ p_{i,j}\ pixel$$

4) eliminating singular tiles, which are not connected to any other validated sub-windows. At this stage I am calculating a score based on the neighbours and filtering out the isolated tiles. Generally, at this step I am discarding the tiles which include noise or any remaining text areas. The results are shown in the image below (Fig. 1.2).

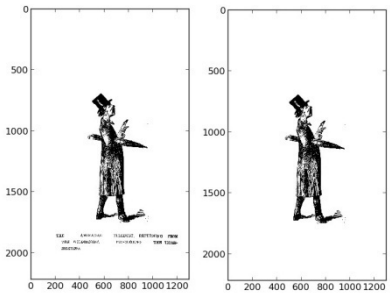

Fig. 1.2 Filtering singular tiles

5) on the resulted tile set I am applying a K-Means clustering algorithm, which uses as a decision metric the Euclidean distance between the elements. At this step I am building the tile sets which represent the images from the document, or parts of a larger image.
6) for each of the above clusters I am computing a score based on their scarcity and connectivity coefficients. The thresholds for these 2 scores have been determined experimentally as T_{sparse}=0.33 and T_{con}=0.5.

$$sparsity_{\mathrm{k}} = \frac{\mathrm{w}*\mathrm{h}*\mathrm{S_k}}{\mathrm{A_k}} \tag{3.3}$$

w and h represent the tile size,
S_k represents the number of tiles in the k cluster,
A_k represents the smallest rectangular window which includes all the tiles

$$conn_k = \frac{K_c}{S_k} \tag{3.4}$$

K_c represents the tile count with at least 2 valid neighbours,
S_k represents the number of tiles in the k cluster

7) the clusters are then filtered in order to eliminate tiles containing text areas with different fonts, affected by noise/poor illumination or by page curvature. The example in the figure below (Fig. 1.3) shows the effect of cluster filtering.
8) the resulting tiles are then merged and exposed to a reconstruction stage, which adds connected pixels, up to the distance D, where D is the diagonal of the tile. Sometimes, the image boundaries may be mistaken as text areas, which means that the final result may not include them. In order to avoid this, I am adding the pixels which are connected to the ones identified as being part of the image.

The algorithm's logical schema is presented in the below figure (Fig. 1.4).

Fig. 1.3 Filtering clusters

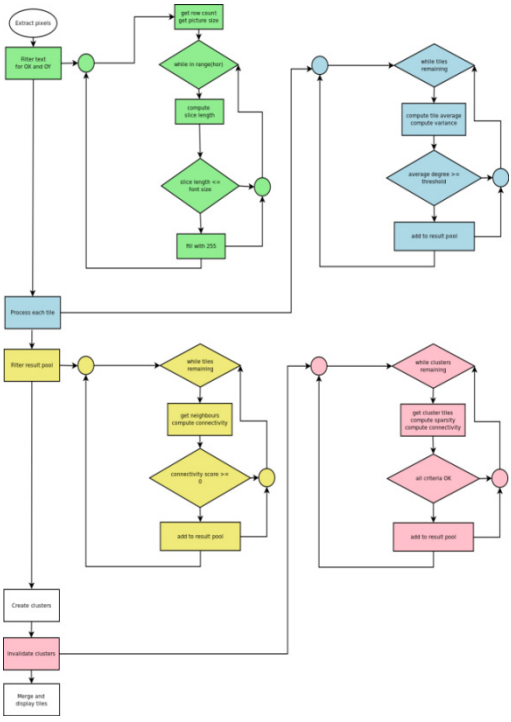

Fig. 1.4 CVSEG logical schema

1.4 Experimental Setup

The experiments have been conducted on a set of 1380 images, obtained from 2 sources:

- scans of old, degraded documents, used as a benchmark in the ICDAR 2007 conference [16];

- high quality copies, containing mostly manuals and documentation for the Ubuntu 12.04 operating system. In order to be able to use them, I have previously converted them from the pdf format to the jpeg one.

Since the CVSEG algorithm requires a pre-binarized document, I have used the below binarization algorithms:

- average binarization, obtained by splitting the image in multiple tiles and applying a normalized threshold on each of them;
- Sauvolabinarization, developed by [17];
- NLBIN, a non linearbinarization algorithm, recommended by the authors of the Ocropus library [18], the default document processing tool used on the Android operating system.

The chosen programming language was python, especially due to the facilities provided for the clustering algorithms and for the matrix handling API. I have used a 32 bit Ubuntu 12.04 operating system, running on a machine based on an Intel I5 dual core processor and 4GB of RAM. Due to the software's modular structure I could easily switch between different tile sizes and binarization algorithms.

The test images have been tagged with bounding boxes, saved in corresponding text files, containing the upper left and lower right coordinates.

The best results have been obtained when using a tile size of 20 pixels. If the tile size is increased too much, it is hard to distinguish between text and images based on the variance, as the analyzed surface would be too big and this indicator will always be high - this translates into disregarding most of the sub-windows. Choosing a tile which is too small translates into a very small variance score, which basically means that the algorithm will validate much more sub-windows.

The results are described in the Table 1.1. The total computing time is basically the sum of the CVSEG algorithm and the binarization execution times.

Table 1.1 CVSEG segmentation results

Segmentation / Binarization	Accuracy	Total computing time
CVSEG avebin	81.2 %	4.614 s
CVSEG Sauvola	84.0 %	14.500 s
CVSEG NLBIN	84.1 %	28.762 s

The figure below (Fig. 1.5) shows an example of how the algorithm behaved on the sample images, originating in both areas - poor quality scans and high quality images.

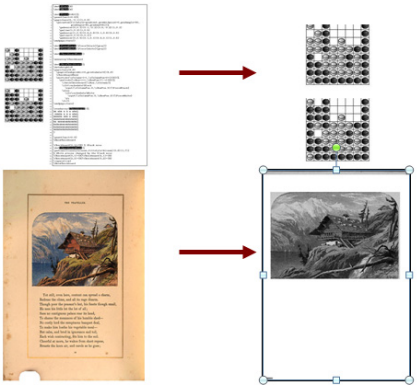

Fig. 1.5 CVSEG results

I have used for the test environment Bloomberg's document image segmentation algorithm, implemented by the author in the Leptonica library [19], available in the Ubuntu 12.04 repositories. The library contains a large set of utilities and is used by many DAR applications and OCR engines, including tesseract [20], an engine maintained and sponsored currently by Google. As this implementation requires a specific type of input images, I have used a GIMP CLI script in order to convert them to the indexed mode.

Besides the above configuration, the software and hardware environment has been the same as the one used for testing the CVSEG algorithm.

The results are described in the Table 1.2.

Table 1.2 Bloomberg segmentation results

Segmentation / Binarization	Accuracy	Total computing time
Bloomberg GIMP	72.2 %	5.205 s
Bloomberg GIMP ave bin	72.3 %	8.012 s
Bloomberg GIMP Sauvola	74.1 %	13.205 s
Bloomberg GIMP NLBIN	74.5 %	15.429 s

1.5 Conclusions

The CVSEG algorithm obtained results up to 9% better than the Bloomberg one. The most frequent errors in the Bloomberg algorithm have been:

- no image in the result - this was caused by documents containing images with very thin lines; the images have been filtered out during the eroding stage;
- the complete document in its negative form is included provided a result - this was caused by documents affected by noise due to the page transparency;
- incomplete results - usually caused by low contrast or by images being tightly connected to the text blocks (Fig. 1.6).

Fig. 1.6 Bloomberg incomplete segmentation

In what regards the CVSEG algorithm, most of the problems have been caused by the below:

- the result included additional areas which were not part of the image - this was caused by the text filtering stage, malfunctioning for text written in different font types (Fig. 1.7);

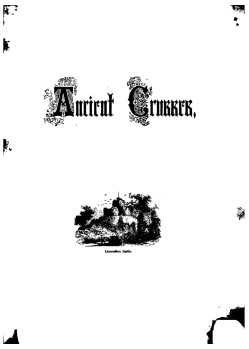

Fig. 1.7 CVSEG erroneous segmentation

- the result includes areas adjacent to the image - this is caused by two factors: the text filtering algorithm and the clustering algorithm have not succeeded to eliminate the text located next to the image (Fig. 1.8);

Fig. 1.8 CVSEG erroneous segmentation – clustering

- when using the Sauvola algorithm, the binarized output included an image rotated at a certain angle, which affects to some extent the segmentation algorithm, at the text filtering stage (Fig. 1.9);

Fig. 1.9 Sauvolabinarization output

- the result included black areas - this was caused by the binarization algorithms which failed to filter out the noise, especially in the images scanned under poor illumination conditions (Fig. 1.10 Binarization affected by noise;
- the high quality documents have been mostly segmented correctly. The only problems I have met have been related to incomplete diagrams and schemes, especially when using thin lines, as can be seen below.

The CVSEG execution times are higher than the Bloomberg ones, mostly due to the chosen programming language, which introduces a significant overhead when compared to native compiled C code.

We can notice that the binarization algorithms do not affect with much the accuracy. There is also a large increase in the computing time when switching to the NLBIN binarization (two times bigger than the Sauvola measurements), which produces just 0.1% accuracy improvement.

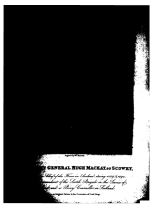

Fig. 1.10 Binarization affected by noise

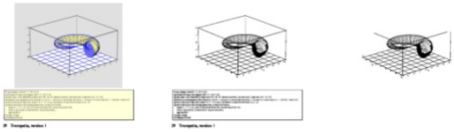

Fig. 1.11 CVSEG affected by thin lines

1.6 Future Research

In the future I would like to continue the research conducted so far. The CVSEG performances can be improved by refining the text filtering algorithm and the clustering algorithm.

Also, I want to add an OCR module based on a neural network combined with a variable size sliding window technique, in order to achieve 2 goals:

- detecting the character scale and automatically establishing the tile size;
- ensuring the absence of the text in the final result.

Another problem I have ran into is the result comparison stage, due to the lack of document image segmentation algorithms. Therefore, I intend to develop a classifier which assigns a specific score to any segmentation algorithm (image or text), according to its results. This will allow me to compare my results with text segmentation algorithms as well.

Acknowledgements. The author would like to thank the Project SOP HRD /107/1.5/S/76822 – TOP ACADEMIC, of University "Dunarea de Jos" of Galati, Romania.

References

1. Pujari, A.K., Dhanunjaya Naidu, C., Sreenivasa Rao, M., Jinaga, B.C.: An intelligent character recognizer for Telugu scripts using multiresolution analysis and associative memory. Image and Vision Computing 22(14), 1221–1227 (2004)
2. Cai, K., Bu, J., Chen, C., Huang, P.: An automatic approach for efficient text segmentation. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4251, pp. 417–424. Springer, Heidelberg (2006)
3. Yang, J., Kang, J., Choi, J.: A focused crawler with document segmentation. In: Gallagher, M., Hogan, J.P., Maire, F. (eds.) IDEAL 2005. LNCS, vol. 3578, pp. 94–101. Springer, Heidelberg (2005)
4. Zhanga, X., Lyu, M.R., Dai, G.-Z.: Extraction and segmentation of tables from Chinese ink documents based on a matrix model. Pattern Recognition 40(7), 1855–1867 (2007)
5. Roy, P.P., Pal, U., Lladós, J.: Document seal detection using GHT and character proximity graphs. Pattern Recognition 44(6), 1282–1295 (2011)
6. Xia, Y., Xiao, B.H., Wang, C.H., Li, Y.D.: Segmentation of mixed Chinese/English documents based on Chinese Radicals recognition and complexity analysis in local segment pattern. In: Huang, D.-S., Li, K., Irwin, G.W. (eds.) Intelligent Computing in Signal Processing and Pattern Recognition. LNCIS, vol. 345, pp. 497–506. Springer, Heidelberg (2006)

7. Sas, J., Markowska-Kaczmar, U.: Similarity-based training set acquisition for continuous handwriting recognition. Information Sciences 191, 226–244 (2012)
8. Tsai, C.M.: Intelligent region-based thresholding for color document images with highlighted regions. Pattern Recognition 45(4), 1341–1362 (2012)
9. Tonazzini, A., Vezzosi, S., Bedini, L.: Analysis and recognition of highly degraded printed characters. Document Analysis and Recognition 6(4), 236–247 (2003)
10. Fonseca, M.J., Pimentel, C., Jorge, J.A.: CALI: An online scribble recognizer for calligraphic interfaces. In: AAAI Spring Symposium on Sketch Understanding, pp. 51–58 (2002)
11. Papamarkos, N.: A neuro-fuzzy technique for document binarisation. Neural Computing & Applications 12(3-4), 190–199 (2003)
12. Chen, N., Blostein, D.: A survey of document image classification: problem statement, classifier architecture and performance evaluation. International Journal of Document Analysis and Recognition (IJDAR) 10(1), 1–16 (2007)
13. Bloomberg, D.S.: Multiresolution morphological approach to document image analysis. In: Proc. of the International Conference on Document Analysis and Recognition, Saint-Malo, France (1991)
14. Bukhari, S.S., Shafait, F., Breuel, T.M.: Improved document image segmentation algorithm using multiresolution morphology. In: IS&T/SPIE Electronic Imaging, pp. 78740D-78740D. International Society for Optics and Photonics (2011)
15. Ha, J., Haralick, R., Phillips, I.T.: Recursive XY cut using bounding boxes of connected components. In: Proceedings of the Third International Conference on Document Analysis and Recognition, vol. 2, pp. 952–955. IEEE (1995)
16. Antonacopoulos, A., Bridson, D., Papadopoulos, C.: Page Segmentation Competition. In: ICDAR 2007 (2007), http://www.primaresearch.org/ICDAR2007_competition/
17. Sauvola, J., Seppanen, T., Haapakoski, S., Pietikainen, M.: Adaptive document binarization. In: Proceedings of the Fourth International Conference on Document Analysis and Recognition, vol. 1, pp. 147–152. IEEE (1997)
18. Google. Ocropus, http://code.google.com/p/ocropus (April 21, 2013)
19. Bloomberg, D.S.: libleptonica, http://www.ubuntuupdates.org/package/core/precise/universe/base/libleptonica (March 6, 2012)
20. Google, Inc. Tesseract (2013), http://en.wikipedia.org/wiki/Tesseract_%28software%29