Normalized Text Font Resemblance Method Aimed at Document Image Page Clustering

Costin-Anton Boiangiu, Andrei-Cristian Spataru, Andrei-Iulian Dvornic, Dan-Cristian Cananau
Computer Science Department
"Politehnica" University of Bucharest
Splaiul Independentei 313, Bucharest
ROMANIA
Costin@cs.pub.ro, Andrei.Spataru@yahoo.com, Andrei.Dvornic@yahoo.co.uk,
Dan_Cananau@yahoo.com

Abstract: This paper describes an approach towards obtaining the normalized measure of text resemblance in scanned images. The technique, aimed at automatic content conversion, is relying on the detection of standard character features and uses a sequence of procedures and algorithms applied sequentially on the input document. The approach makes use solely of the geometrical characteristics of characters, ignoring information regarding context or the character-recognition.

Key-Words: - automatic content conversion, text characteristics, font size, boldness, italic, texture measurements

1 Introduction

Automatic document content conversion has been one of the most interesting areas of development in the last years [10][11], also determined by the rapid expansion of digital libraries [16]. The OCR software has been developping and has greatly improved during this time, but errors still occur in detection, due to several factors [12][13]. Also new scanners and image processing methods have appeared. The next step in this domain is the detection of the logical structure of the document, and, in order to accomplish this, there are a number of measurements that need to be made.

This paper provides several algorithms that retrieve the relevant information in the document such as font size or the degree of boldness, and others, without any actual character recognition. Several such algorithms already exist, but what is different in this approach is the fact that it takes into consideration all the computed characteristics and combines them in order to present a structural view of the document. By doing so, scanned images will be easier to interpret, structural characteristics for text areas will be available, and the clustering of text will be possible.

2 The Need for Text Classification

Content conversion on documents strongly relies on the extraction of character parameters from input images. The information obtained from these measurements is then used in the creation of the logical hierarchy of content, so detection methods have to be exact and controllable.

As the variety of input images is extensive, a method must be found to perform the comparison of text characteristics on a wide array of input scenarios. Given an input image, the final goal is to obtain a normalized measure of text resemblance, and decide if text areas within the image are logically connected, so they can be categorized and structured hierarchically (e.g. titles, subtitles, paragraphs, footnotes and page numbers).

In the following part, the main issues of the task ahead will be presented and addressed, in order to obtain a generally applicable method.

The input image has to undergo a preparation stage, comprising a conversion from any color space into black & white, an extraction of connected pixels in the foreground (which will be referred to in following as *"Entities"*), and a filtering of Entities as to separate text (characters) from images or other components that are irrelevant to this purpose.

2.1 Applying Measurements

In order to obtain relevant results, a number of measurements that concern various geometrical aspects of the Entities will be taken into consideration. In following, these methods will be enumerated along with a short reasoning to their usage. All measurements will be thoroughly explained in the next section.

- **Texture**: refers to the application of texture-specific measurements to the Entities, such as Mean, Variance, Skewness and Kurtosis. The output is a statistical result of black pixel appearance and has the advantage of speed of execution over the output accuracy.
- **Font size**: a measurement of the high caps (height of the capital letters) and low caps (height of non-capital letters).
- **Font boldness**: also referred to as the "pen width", is the thickness of the Entity.
- **Font italics**: represents the slant angle of the Entity.
- **Line spacing**: refers to the distance between two consecutive text rows.

2.2 Interpreting Results

The final step is the interpretation of the results obtained from the above measurements. The importance of the detected text features comes into question here, as well as the running time of the algorithms for each feature.

3 The Text Measurements

3.1 Image Preparation Stage

Because the actual black and white conversion of the input image and the extraction of Entities are beyond the scope of this paper, further attention will not be paid in this direction. A suitable algorithm for image binarization should be used, depending on the quality of the input image. In this case, color conversion algorithms from [4] were used.

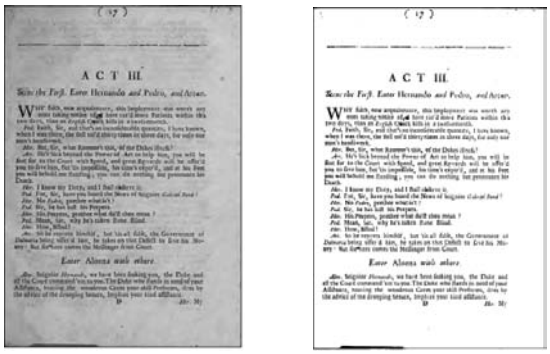


Fig.1: Conversion from Color or Grayscale to Black and White

For the extraction of Entities, horizontal run-length sequences of black pixels (called "segments") are found in the input black & white image. The segments, which have as main characteristics their "*row*", the "*start column"* and "*stop column",* values that represent the leftmost and rightmost X-axis coordinates, are then grouped into clusters of connected black pixels, called "*Entities*".

Detailed methods for run-length connected pixels extraction can be found in [14], while conversion algorithms are also available in [15].

Fig.2: Entities extracted from an input image, represented as bounding rectangles

A set of Entities extracted from the converted (black & white) image will be considered as input.

3.2 Filtering of Entities

As already stated in Section 2, the set of input Entities contains both significant and insignificant characters and symbols, so a number of filters have to be applied in order to ensure the relevance of the dataset for the measurements.

Methods for the improvement of OCR input quality already exist [9], but their complexity makes them unsuited for the purpose at hand.

In this approach, three filters are used, and are applied in the following order:

1. "Inside" filter
2. "Merge" filter
3. "Width" filter

The issues raised by erroneous binarization, also due to the low quality of some scanned documents and the presence of noise, are partially or totally rectified by these filters. Also, a number of image de-noising algorithms are available in [2].

3.2.1 "Inside" Filter

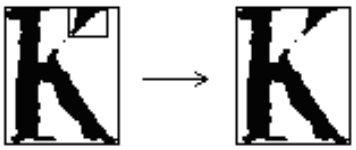


Fig.3

The "inside" filter algorithm checks whether an entity is included entirely into another one, by considering

the top-left and bottom-right coordinates of their bounding boxes. If this condition is fulfilled, the entity with the smaller bounding box area is added to the bigger one. The purpose of the procedure is to unite letters that are fragmented, for example adding the top right part of the K to the other part of the letter.

3.2.2 "Merge" Filter

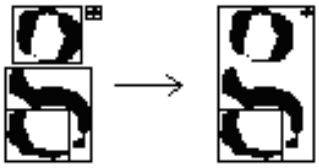


Fig.4

The "merge" filter checks if an entity can be connected with another one vertically, in order to rebuild a letter. If one entity has the left or right bounds inside the left or right bounds of the other (if they are one above the other) and if the distance between them is smaller than a chosen threshold, they are connected.

3.2.3 "Width" Filter

This method first takes the height and width of each entity's bounding rectangle and, if the width is greater or equal than the height, the fill ratio of black pixels in the bounding rectangle is computed. If this fill ratio is above 80%, the current entity is considered noise or punctuation mark, and is removed from the input array.

3.3 Applying measurements

At this point, the input is a set of filtered Entities, and the algorithm can be continued with the extraction of text characteristics, as stated in [3].

3.3.1 Texture measurements

The following formulas were applied, considering the black pixels as input.

Mean: $$m = \frac{1}{N}\sum_{i=1}^{N} \mathrm{x_i} \quad (1)$$

Variance: $$s = \frac{N\sum_{i=1}^{N} x_i^2 - m^2}{n(n-1)} \quad (2)$$

Variance coefficient: $$cv = \frac{s}{m} \quad (3)$$

Skewness:

$$\tau = \frac{N}{(N-1)(N-2)}\sum_{i=1}^{N}\left(\frac{x_i - m}{s}\right)^3 \quad (4)$$

Kurtosis: (5)

$$k = \frac{N(N+1)}{(N-1)(N-2)(N-3)}\sum_{i=1}^{N}\left(\frac{x_i - m}{s}\right)^4 - \frac{3(N-1)^2}{(N-2)(N-3)}$$

where N represents the number of black pixels, and x the coordinate on the horizontal axis of the pixel.

The most interesting result was obtained using the skewness measurement. By applying the measurement on a regular newspaper page, the titles, subtitles, and paragraph text returned values with a difference of 1 order of magnitude between them. For example the title of the newspaper returned a value in units, the subtitle and paragraph titles in 10's and normal text in 100's.

3.3.2 Font Size

The main idea of the method is to select the peaks in the histogram of entity heights. The peaks are the small caps, big caps and various noise or punctuation signs that were not eliminated by the filters. If two or more peaks are found, the highest one is the small caps, but also the other peaks have to be inside a predefined range in order to consider them as big caps and punctuation signs.

If only one peak is found, or the entities are not in the range, then all letters are considered in high caps, and the low caps are 0. In order to obtain better results, a triangle filter is applied to the histogram, and is presented below.

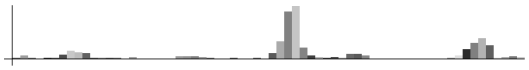


Fig.5: Histogram before filtering

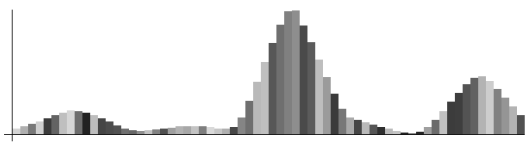


Fig.6: Histogram after filtering

The filter is a weighted average using a triangular window of width 10% from the total histogram length (ex: first element has the weight 0.1, ascends until the middle element and then descends to the last element). In this way the highest value in a group will be emphasized, in order to point out the difference of the peaks with the other values.

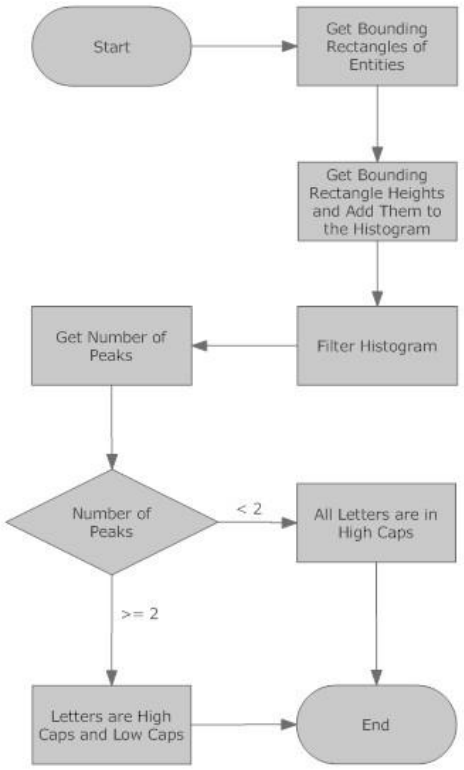


Fig.7: Flowchart of the Font Size Algorithm

3.3.3 Font Boldness

For the extraction of the font boldness characteristic, two distinct algorithms have been used: "contour length" and "crosshair".

a. b.

Fig.8: The difference between a normal letter (a) and a bold letter (b) regarding the ratio between the number of outline pixels and the total number of pixels.

"Contour length" computes the number of pixels comprising the contour of the entity and the total number of pixels of the entity.

The percentage result is the ratio between these two.

$$ratio = \frac{outline}{total} \tag{6}$$

Consequently, this ratio will have a lower value in bold letters.

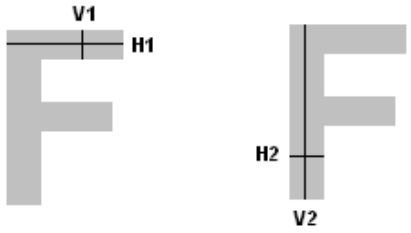


Fig.9: For the pixel at the intersection of V1 and H1, the dominant will be vertical (V1), as for the pixel at the intersection of V2 and H2 the dominant will be horizontal (H2).

The "Crosshair" method calculates the width of the "pen" that wrote the entity. This is done by iterating each black pixel in the entity and searching for the length of the segments of black pixels up, down, left and right, returning the vertical and horizontal "crosshair" segments. Only the smaller segment is considered, called "dominant", representing the pen width.

After obtaining dominants (segment lengths) for all black pixels in the entities, these segments are added to an array, and a histogram of frequency is created. The most frequent dominant represents the value of interest. As an addition to this, the algorithm checks if 2 peaks are close in size (on certain texts, the value oscillates by 1 pixel) and returns the average between the two, making the method more accurate.

3.3.4 Font Italics

For the extraction of the italic characteristic, again two distinct algorithms have been used: "width" and "chain".

The "Width" method computes the width of the bounding rectangle of the entity and then applies a rotation of the black pixels inside.

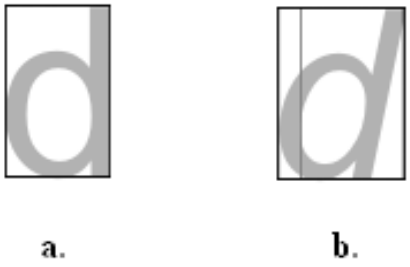


Fig.10: (a) The bounding rectangle of a normal letter (b) The same bounding rectangle over the italic letter. Observe the difference in width.

The rotation is done by -16 degrees (considered as a common value for italic characters), and by +16 degrees, in the real space, in order to keep accuracy. The algorithm then computes the maximum width of the set of rotated points. The idea is that a rotated italic character has a smaller width value than it had initially, while if the character was not italic, after rotation the width grows. The rotation by +16 degrees is done as a check in order to eliminate errors coming from the geometry of the entity, or so-called "naturally italic" characters like "w".

The "Chain" algorithm rotates the entity by -16 and then by +16 degrees and computes the longest vertical black pixel line in each case. The idea is that an italic character has a lower value for the longest vertical line.

Fig.11: (a) the longest vertical line in a normal letter (b) the longest vertical line in an italic letter. Observe the shorter line in case (b).

After rotation, the pixels tend to align vertically. The decision is taken according to the rotated -16 and rotated +16 longest vertical lines. (if after the rotation by -16 the line increases, the character is italic).

3.3.5 Line Spacing

Line spacing measures the distance between two rows of characters, or entities in this case. The algorithm takes into consideration only the letters that are above each other. It checks if the entities are in consecutive lines by comparing the distance between them with a mean measure of their heights, and then checks if the entities are above each other, or approximately above each other, by verifying that the x coordinates of an entity is inside a bound given by the x coordinate of the other entity. Finally the algorithm returns a floating-point value, representing the average distance, in pixels, between the base lines of characters in the considered text area.

3.4 Interpreting the measurements

Because two alternative algorithms were created for the boldness and italics measurements, tests were performed on a large number of input images, to find the algorithm that yields the best results in both cases.

For the boldness measurement, the "Crosshair" method was more exact, because the result is returned in pixels, while the "Contour Length" method returns the result in percent. Also, a result in pixels makes the comparison and thresholding easier to perform.

For the italics measurement, the "Width" method was considered more reliable, because the "Chain" method relies more on the quality of the input (if the letter is fragmented, the longest vertical line will no longer be correctly found).

As was stated in Section 3.2.1, the Skewness value for the Texture measurements was used in the tests, where the other text characteristics were found similar.

The first step is to compare the small caps of the first text area with the small caps of the second one; if one of them is zero then that text has only big caps and the algorithm passes to the next step; Else, a ratio between the two low caps values is computed.

The same goes for high caps, and so the final ratio is given by the mean between the small caps ratio and big caps ratio.

If the ratios are significantly different, there is no need for boldness or italics computation. Otherwise the algorithm goes to the next step, computing the boldness.

First, the higher boldness value between the two text areas is found. If the difference in boldness is above 1 pixel / entity, the two texts differ and there is no need for italics computation. Otherwise, the italics comparison is made, checking only if the texts are italic or not.

If none of the above measurements yield a result that clearly joins or separates the texts, the texture measurements are applied, but only if the input is considerable in size.

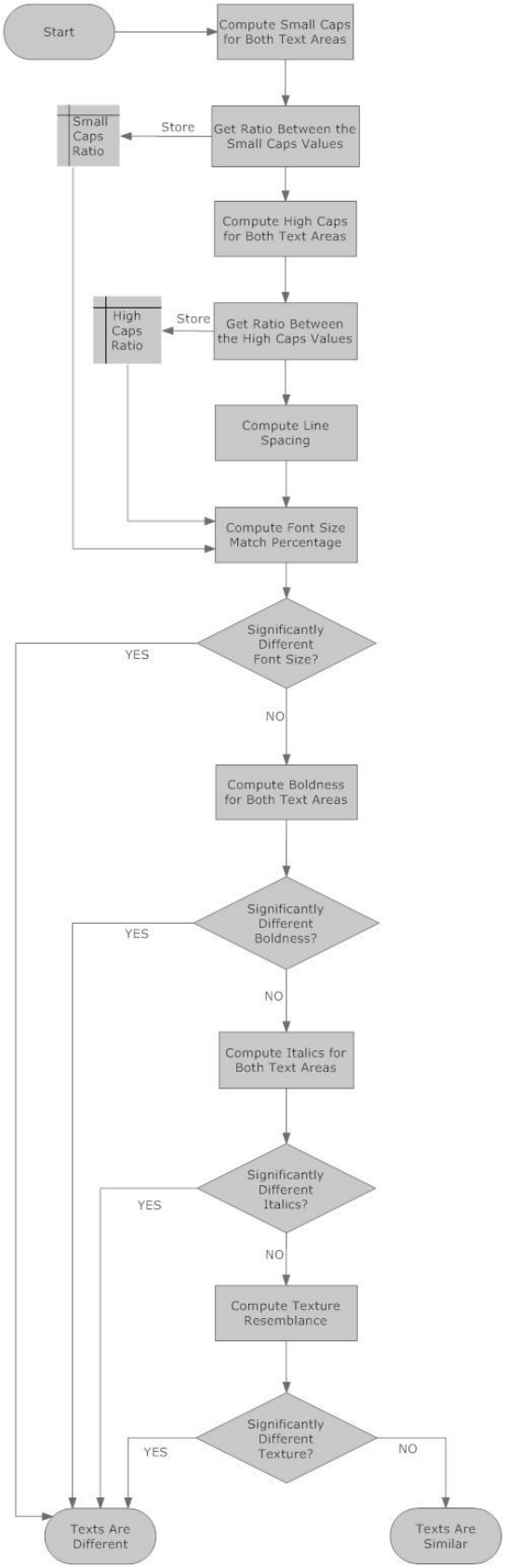


Fig.12: Flowchart of the Measurements Interpretation

3.5 Experimental Results

In this section, results using the described algorithm are presented.

The sample images have been chosen from scanned newspaper pages as to emphasize various scenario outcomes.

3.5.1 Significantly Different Text Areas

The first test is performed on two text areas that are significantly different in content. The first one (*Fig.13*) is a newspaper title, and has to be compared to a paragraph (*Fig.14*) from the same page. The results of the algorithm in this case are presented in Table 1 and *Fig.15*.

Fig.13: Title

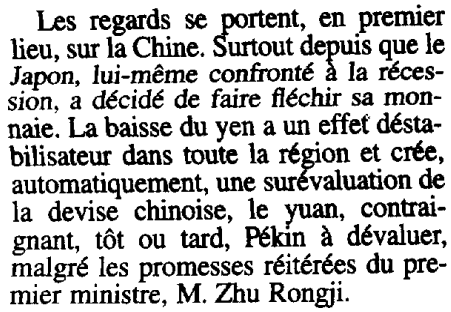


Fig.14: Paragraph

Image	Fig.13	Fig.14
High Caps (pixels)	114	26
Low Caps (pixels)	82	19
Line Spacing (pixels)	70	11
Font Size Match (%)	0	
Crosshair (pixels)	Not applied	Not applied
Width (%)	Not applied	Not applied
Skewness	Not applied	Not applied
Conclusion	The texts differ significantly.	

Table 1

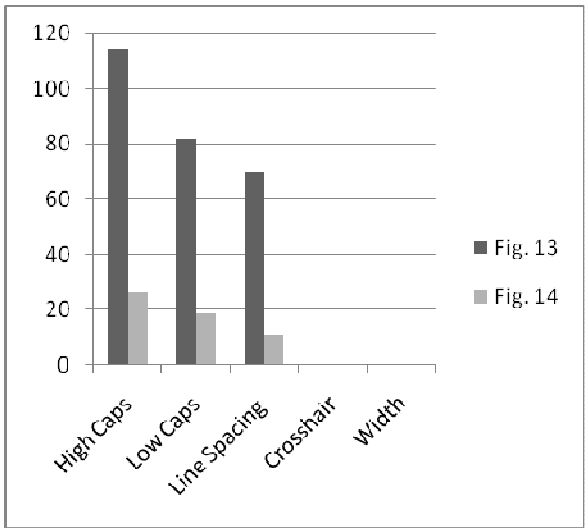


Fig.15: Graph showing the results from the algorithm for significantly different text areas

As it can be observed in *Fig.15*, the differences found in the High Caps, Low Caps and Line Spacing determine the algorithm to stop the following measurements and return a "no match" between the text areas.

3.5.2 Normal and Bold Text Areas

The second test is performed on two text areas containing bold text lines (*Fig.16*) and normal text lines (*Fig.17*). The texts are chosen from the same paragraph, in order to have the same font size and line spacing.

tiert, wird vom Militär müh-sam in Schach gehalten. Die Studenten verbrennen Bilder des neuen, alten Staatschefs.

Fig.16: Bold text lines

rungsfonds IWF, Feuerwehr der Feuerwehren im Finanzbreich, wollte mit 43 Milliarden Dollar helfen. Aber Suharto, Habibie

Fig.17: Normal text lines

Image	Fig.16	Fig.17
High Caps (pixels)	28	28
Low Caps (pixels)	20	20
Line Spacing (pixels)	19	19
Font Size Match (%)	100	
Crosshair (pixels)	6.0	3.5
Width (%)	Not applied	Not applied
Skewness	Not applied	Not applied
Conclusion	The texts have the same font size, but they differ in boldness by 2-3 pixels.	

Table 2

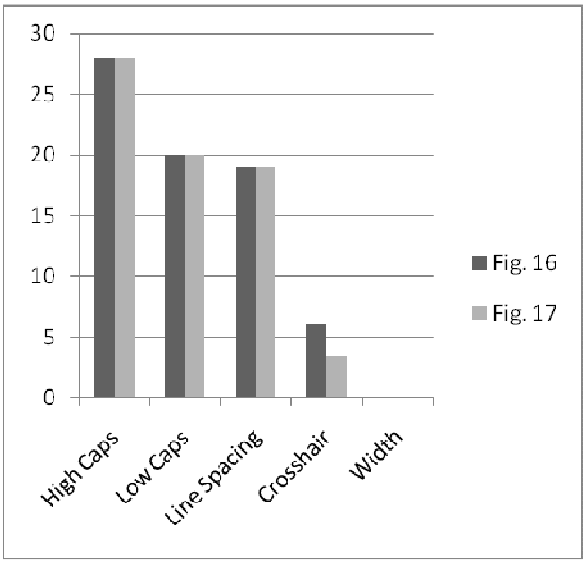


Fig.18: Graph showing the results from the algorithm for text areas that differ in boldness

As expected, the difference of 2.5 pixels in thickness returned by the Crosshair algorithm causes the algorithm to return a "no match" result, and stopping before the Italics measurement.

3.5.3 Normal and Italic Text Areas

The third test is performed on two text areas, one containing mostly italic characters (*Fig.19*), and another containing text in the same font size and line spacing, but normal characters (*Fig.20*).

Dans son éditorial, Ignacio Ramonet constate que *« la moitié de l'économie mondiale se trouve frappée par une crise systémique »*. Et se demande : *« L'autre moitié (dont l'Union européenne) peut-elle éviter la contamination ? »*

Fig.19: Italic text

Avant d'atteindre la Chine, la vague de la crise financière a déferlé sur le Japon et sur le Sud-Est asiatique, puis sur la Russie et, plus récemment, sur le Brésil. Rien, là, d'une catastrophe naturelle : au cœur de l'effondrement de ces pays se trouve une véri-

Fig.20: Normal text from the same page

Image	Fig.19	Fig.20
High Caps (pixels)	25	26
Low Caps (pixels)	18	18
Line Spacing (pixels)	20	19
Font Size Match (%)	93.65	
Crosshair (pixels)	3.0	3.0
Width (%)	88.1	11.5
Skewness	Not applied	Not applied
Conclusion	The texts have the same font size, the same boldness, but the characters in *Fig.19* are italic.	

Table 3

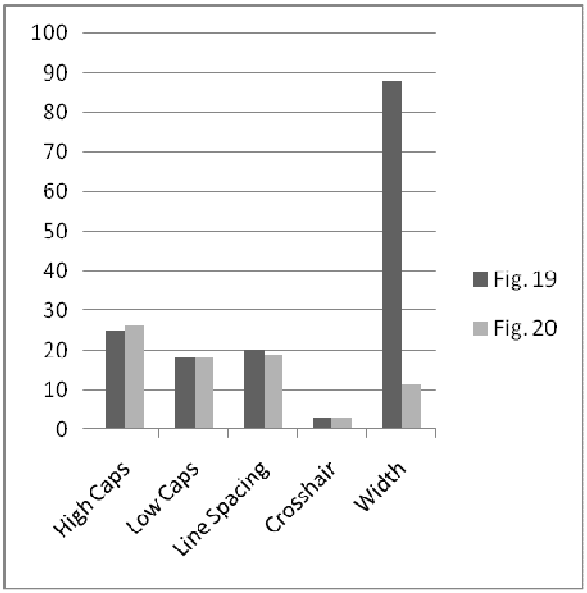
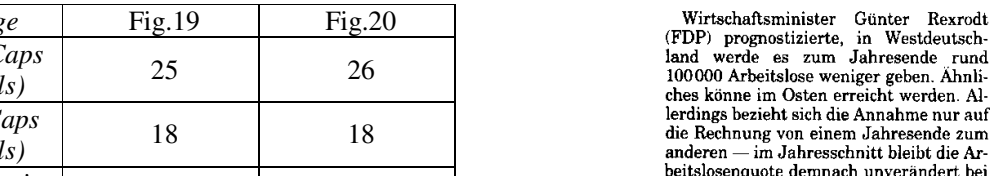


Fig.21: Graph showing the results from the algorithm for text areas that differ in italics

In this test, the percentage of italic characters returned by the Width method for *Fig.19* clearly separate it from the text area containing normal characters. The algorithm stops and returns a "no-match" result.

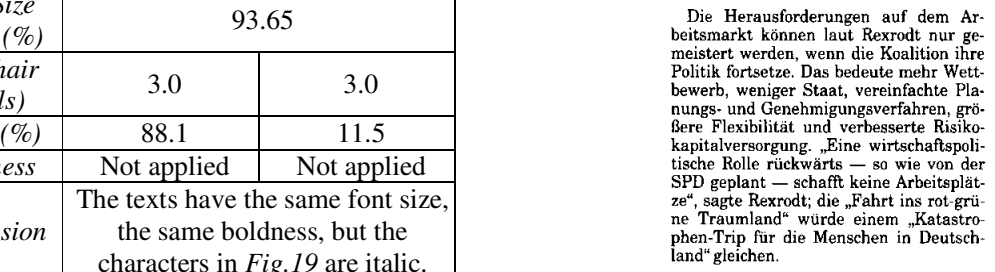
3.5.4 Similar Text Areas

For the final test, two text areas containing similar text were used (*Fig.22, Fig.23*). The input images were chosen to show the result of the algorithm being applied on texts that are visually identical in geometry, and to demonstrate the application of the skewness texture measurement.



Wirtschaftsminister Günter Rexrodt (FDP) prognostizierte, in Westdeutschland werde es zum Jahresende rund 100000 Arbeitslose weniger geben. Ähnliches könne im Osten erreicht werden. Allerdings bezieht sich die Annahme nur auf die Rechnung von einem Jahresende zum anderen — im Jahresschnitt bleibt die Arbeitslosenquote demnach unverändert bei

Fig.22



Die Herausforderungen auf dem Arbeitsmarkt können laut Rexrodt nur gemeistert werden, wenn die Koalition ihre Politik fortsetze. Das bedeute mehr Wettbewerb, weniger Staat, vereinfachte Planungs- und Genehmigungsverfahren, größere Flexibilität und verbesserte Risikokapitalversorgung. „Eine wirtschaftspolitische Rolle rückwärts — so wie von der SPD geplant — schafft keine Arbeitsplätze", sagte Rexrodt; die „Fahrt ins rot-grüne Traumland" würde einem „Katastrophen-Trip für die Menschen in Deutschland" gleichen.

Fig.23

Image	Fig.22	Fig.23
High Caps (pixels)	27	27
Low Caps (pixels)	18	18
Line Spacing (pixels)	19	21
Font Size Match (%)	94.3	
Crosshair (pixels)	3.0	3.5
Width (%)	1	0
Skewness	121	102
Conclusion	The texts are similar.	

Table 4

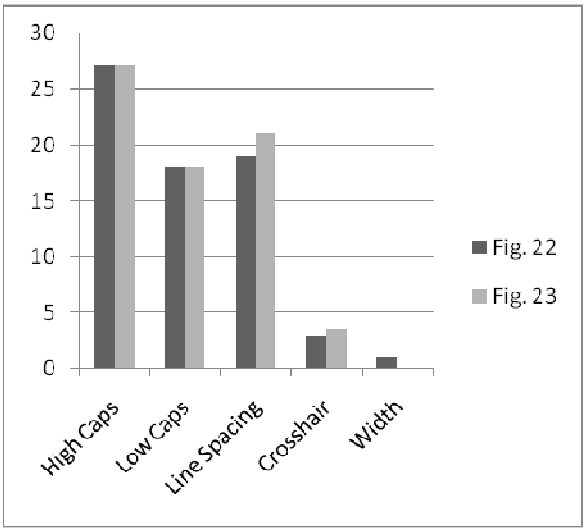


Fig.24: Graph showing the results from the algorithm for similar text areas

The difference in Skewness of the two text areas is not large enough to be counted as a major dissimilarity, so the algorithm returns a "match".

4 Page Segmentation

In order to accomplish the task of page segmentation and extraction of logically homogeneous elements, suitable text areas should be input to the algorithm for comparison. The most important issue when trying to run the algorithm for separation purposes, on a whole newspaper page (for example) comes from the step of the iteration, or the way the comparison operands are chosen. In this approach, the comparison operands are chosen to be text lines, extracted using a geometrical algorithm. Alternative page segmentation approaches are available in [1].

4.1 The Text Line Detection Algorithm

The Text Line Detection Algorithm comprises a custom data structure and a routine that iterates through the input array of Entities and builds them into geometrically connected text lines.

The custom data structure is designed as follows:

- an index, indicating the first Entity in the text line
- an array of indexes, containing the rest of the Entities found to be on the same line, based on a decision rule.
- coordinates of the bounding box of the text line (*topleft* and *bottomright*)
- the number of Entities the line contains

The text line composition routine iterates through the input array of Entities and either adds new components to an existing line (to the array of indexes) or creates a new line with the current Entity as first index.

Based on the decision rule, Entities are on the same text line if two conditions are fulfilled:

- The bounding rectangles of the Entities are overlapping on the vertical axis
- The horizontal (x-axis) distance between two consecutive Entities is smaller than a chosen threshold value. This threshold value is chosen so that lines in neighboring text columns will not be merged.

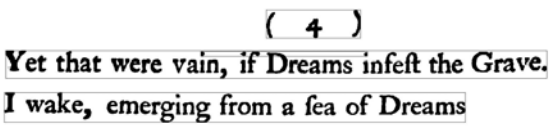


Fig.25: Detected text lines

Because of the punctuation marks or noise in the page, the detection of text lines will not always be exact. In *Fig.25*, a separate text line was found inside another and also, in *Fig.26*, the skewed input page makes the line detection erroneous.

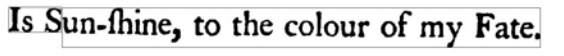


Fig.26: Skewed page causing an error in text line detection

To address these issues, a filtering function was applied, joining text lines that are either one completely inside another or horizontally adjacent.

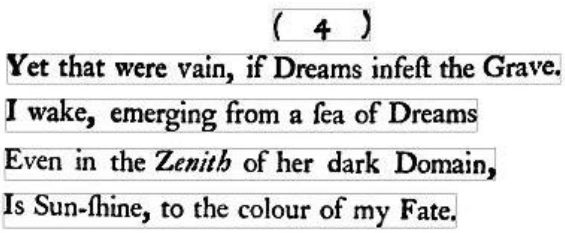


Fig.27: Final result of the text line detection

4.2 Application on a whole page

By applying the Text Line Detection algorithm on an entire page, text rows are extracted to serve as operands for the Text Characteristics algorithm. Thus, the text lines of the input page are compared, and a connection is made between ones with similar characteristics (when a "match" is returned by the algorithm). The result (*Fig.29*) is that headlines, subtitles and paragraphs are separated and viewed as homogeneous page elements.

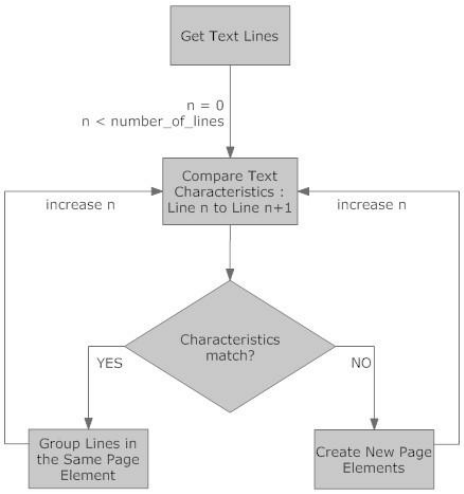


Fig.28: Flowchart of the Page Segmentation algorithm

CDU beschwört „Jobmaschine“

Bundesregierung prophezeit 200 000 neue Stellen / Opposition: Schönfärberei

Türkische Polizisten ohne Strafe

Gericht in Manisa nennt Vorwurf der Folter nicht bewiesen

Öffentlicher Dienst

Schlichter zeigen sich „gemäßigt optimistisch“

Fig.29: Segmentation results obtained on a newspaper page

5 Conclusion

In this paper, a number of algorithms for extracting character information were presented, and used in conjunction, in order to obtain a normalized measure of text resemblance, between two input text areas. The comparison was done using the geometry of the text alone, without knowledge of the font type.

A practical application for the algorithm was presented in Section 4. By using the Text Characteristics extraction algorithm, together with a Text Line detection algorithm, a regular newspaper page was successfully segmented.

As another application, the extracted page elements can be used to obtain a page hierarchy, by comparing the general characteristics of every element to standard values for titles, subtitles and paragraphs.

References:

[1] G. N. Srinivasan, G. Shobha, “An Overview of Segmentation Techniques for Target Detection in Visual Images”, *Proceedings of the 9th WSEAS International Conference on Automation and Information (ICAI ’08)*, WSEAS Press, June 2008, ISBN 978-960-6766-77-0, ISSN 1790-5117

[2] P. Bojarczak, S. Osowski, “Denoising of Images – a Comparison of Different Filtering Approaches”, *WSEAS Transactions on Computers*, Issue 3, Volume 3, July 2004, ISSN 1109-2750

[3] C. A. Boiangiu, A. C. Spataru, D. C. Cananau, A. I. Dvornic, “Automatic Text Clustering and Classification Based on Font Geometrical Characteristics”, *Proceedings of the 9th WSEAS International Conference on Automation and Information (ICAI ’08)*, WSEAS Press, June 2008, ISBN 978-960-6766-77-0, ISSN 1790-5117

[4] C. A. Boiangiu, A. I. Dvornic, “Bitonal Image Creation for Automatic Content Conversion”, *Proceedings of the 9th WSEAS International Conference on Automation and Information (ICAI ’08)*, WSEAS Press, June 2008, ISBN 978-960-6766-77-0, ISSN 1790-5117

[5] Costin-Anton Boiangiu, ”Multimedia Techniques”, Macarie 2002.

[6] B. Chen, and L. He, “Fuzzy template matching for printing character inspection”, *WSEAS Transactions on Circuits and Systems*, Issue 3, Vol. 3, 2004.

[7] L. M. Sheikh, I. Hassan, N. Z. Sheikh, R. A. Bashir, S. A. Khan, and S. S. Khan, “An Adaptive Multi-Thresholding Technique for Binarization of Color Images”, W*SEAS Transactions on Information Science and Applications*, Issue 8, Vol. 2, 2005.

[8] M. I. Rajab., “Feature Extraction of Epiluminescence Microscopic Images by Iterative Segmentation Algorithm”, *WSEAS Transactions on Information Science and Applications*, Issue 8, Vol. 2, 2005.

[9] Prateek Sarkar, Henry S. Baird, Xiaohu Zhang, “Training on Severely Degraded Text-Line Images”, *ICDAR*, Volume 1, 2003.

[10] Steve Man, “*Intelligent Image Processing*”, John Wiley & Sons, 2002.

[11] William K. Pratt, “*Digital Image Processing*”, John Wiley & Sons, 2001.

[12] S. V. Rice, G. Nagy, and T. A. Nartker, “*OCR: An Illustrated Guide to the Frontier*”, Kluwer Academic Publishers, 1999.

[13] S. V. Rice, F. R. Jenkins, and T. A. Nartker, “*The Fifth Annual Test of OCR Accuracy*”, ISRI TR-96-01, Univ. of Nevada, Las Vegas, 1996.

[14] S. Di Zenzo, L. Cinque, S.Levialdi, “Run-, Based Algorithms for Binary Image Analysis and Processing”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.18, No.1, January 1996.

[15] A. Bovik, “*Handbook of Video and Image Processing*”, Academic Press, 2000.

[16] H. S. Baird, “Digital Libraries and Document Image Analysis”, *ICDAR,* Volume 1, 2003.