TEXT AREA MARKING

Stefan Renkema, 1297570, s.renkema@ai.rug.nl

Abstract: One of the difficulties facing handwriting recognition and writer identification software is dealing with non-text objects in their input. Stains and paper edges for example can be mistaken for actual text and thus lead to inaccurate results. I will propose a method for dealing with such side-effects by reducing the area in which recognition and identification will be performed to a rectangle slice, sized by the outer edges of the detected text. This greatly reduces the influence of non-text objects present in the original document, as well as objects added in the scanning process.

1. Introduction

Systems which allow users to automatically identify the writer of a given handwritten text, such as letter bombs, anthrax mail or death threads, are of great use for law enforcement agencies. Also there is a need for a system allowing query based digital searching through handwritten, and perhaps historical documents, because the traditional manner of manual search is too much time consuming.

Writer identification systems rely on a very large volume of handwriting samples, accessible through a database. These samples are acquired from known perpetrators and suspects and are then stored digitally. In order to successfully extract writer specific features from these documents some sort of pre-processing has to be performed beforehand. A possible way to go about this is by doing these pre-processing steps manually, but this would be immensely costly in manpower and would consume a large amount of time, since there are so many documents on file that require processing. This means the prerequisites are there for an automatic computer based solution.

The goal of such a system is to be as autonomous as possible. Preferably the user should not be bothered with supplying the system with one or perhaps several parameters before adding a new document to the database, a standardized text handwritten by a suspected perpetrator. And also posing an identification query, a handwritten message found at a crime scene, presumed to be from a perpetrator, should be completely automatic.

The same level of autonomous behaviour is required for systems allowing digital searching in handwritten documents, since there are also very many of them. Contrary to writer identification systems it is very likely that documents to be added to the database are of very poor quality, since unlike their forensic counterparts the historical documents are written on a less quality medium with usually a very aggressive ink that consumes it. Add to this the wear and tear over many years, as well as less than optimal storage conditions and you end up with documents sometimes barely readable even to the human eye.

Especially for this type of deprecated documents that text are marking is needed, since the medium will not only contain the desired text, but also an abundance of non-text objects, black edges, stains and the like. The non-text objects make the job of handwriting recognition and writer identification much more difficult. With handwriting recognition non-text objects are processed and falsely text is returned, producing unreadable output, which is of course an undesirable effect. Writer identification software relies on handwriting features which vary from person to person. But when non-text features are included in the input for the feature extractor, the results are negatively influenced, since the stains and other non-text objects do not carry information about the writer. The method I propose for dealing with these non-text objects is by drawing a rectangle around the actual text, allowing the recognition and identification programs to only consider the actual text by reducing their analyses to the boundaries of this rectangle.

The correct location for these boundaries are acquired from connected components analysis performed on the objects present in the input documents. From these objects various ratios are determined on which the judgement is made to consider them text or non-text objects. The outermost text objects define the location of the boundary. At this time this system does not function perfectly as there is still some development to be done, but even though it already does a good job at removing areas of the scanned image which do not contain text.

The system for which I developed the text marking tool utilizes evolutionary algorithms for optimising pre-processing. This means that for every parameter added to a pre-processing step there is another degree of freedom extending the time it takes for the evolution to converge to the correct values.

So the method proposed in this paper, uses no other inputs besides the input document.

2.Methods

This section describes the methods which are needed in order to mark the text area in a document.

These methods include the section selection mechanism, a mechanism which divides the image into smaller sections. After that two methods for pre-processing the image, an initial way to rid the image of non-text objects are described. Followed by two methods gathering information about the remaining candidate text objects, connected components labelling and boundary tracing. The section continues with describing how this gathered information is used to judge whether an object is text or not.

2.1 Section selection mechanism

The section selection mechanism allows for a localized approach to analysing the document. The section selection mechanism was initially in place to be able to deal with the somewhat limited recursion depth the used programming language provides. This depth was reached by the connected components method when processing large components, and was found to be at a little over 16.600. A way of dealing with this problem was to divide the image into sections which contain less than that critical amount of pixels. This provided the size of 129 by 129 pixels for the sections. 129 is a little under the square root of the critical recursion depth, resulting in a section that, even when filled completely by a component, does not reach the critical recursion depth. Later on another benefit of this local approach was discovered regarding thresholding, this is discussed in section 2.2.2.

The way in which the mechanism selects the sections guarantees that a section will always be of the desired size, 129 by 129 pixels, making sure that every section contains the same amount of information. This is necessary for the thresholding function, as discussed in section 2.2.2. A problem with dividing the document into sections is that components that lie on the edge of a sections get cut off, making misjudgement more likely. To get rid of this side effect the sections overlap. This is achieved by moving the analysed section along by shorter distance than the dimensions of the section. The amount of overlap that produces good results is 59 pixels. To realise this the 129 by 129 pixels section is moved along horizontally and vertically in steps of 80 pixels.

The section selection mechanism has four different modes of operation, one for each of the sides of the rectangle being established to mark the text area. They all have in common that the search for the text is started from the edge of the image, and then works its way inwards, per row and column until contact is made with the text. Once an edge of the text area is detected no sections beyond that edge will be analysed, this reduces the amount of calculations that have to be performed and thus decreases the time required to process a document, see figure 1 for a graphical representation.

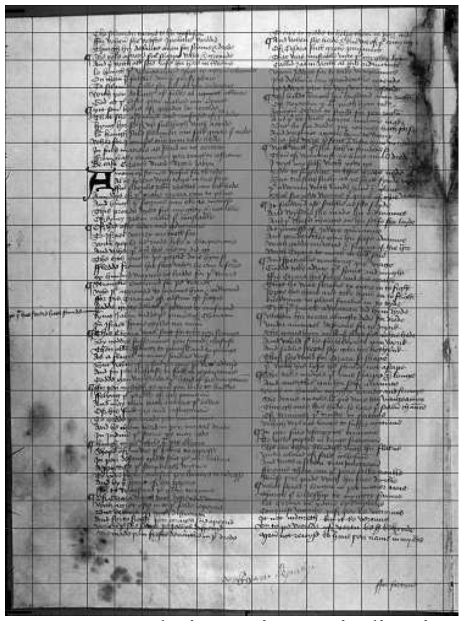


Figure 1, a sample input image indicating the step size, indicated by the grid lines, of the section selection mechanism, as well as the area, marked red, that can be excluded from being analysed, since a wider edge was already found.

2.2 Pre-Processing

The pre-processing stage is the first stage at which a separation is made amongst text and non-text objects. This is done by smoothing the background which consists of non-text objects, this greatly reduces the amount of objects that remain to be analysed. Also at this stage the image is binarized, this means that the fore- and background are each assigned a different brightness level, the background is turned white, and the foreground is turned black. The advantage of binarizing the image is discussed in section 2.2.2.

This binary image is then fed to the connected components labelling method which will identify individual objects and determine their size. The individual objects are then traced, determining the size of their boundary. With this size and boundary information various calculations are performed. The results of these calculations are then used to separate text from non-text objects.

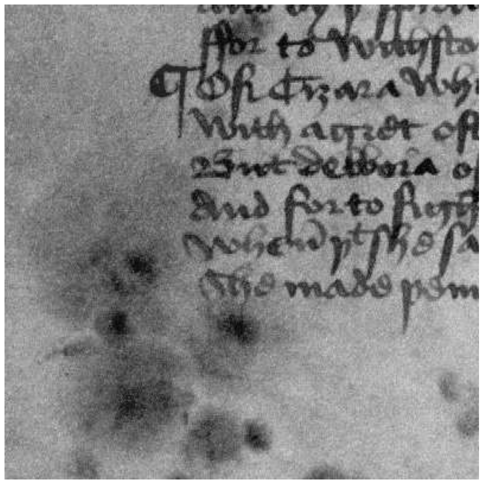


Figure 2a: Greyscale version of the original full colour input image.

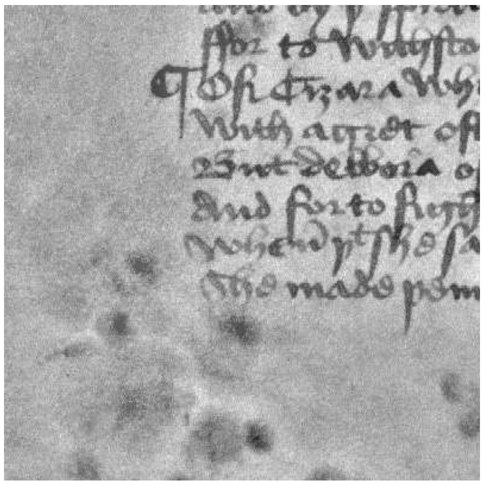


Figure 2b: The high-pass filter applied to the greyscale image, this is the type of input from which sections are cut to analyse.

2.2.1 Filtering

The first pre-processing step is filtering. The filtering method used is a high-pass filter. It involves subtracting a blurred version of the image from its original and then adding the average greyscale level. This results in an image in which small stains are blended into the background. See figures 2a and 2b for an example.

2.2.2 Thresholding

After filtering we are left with an image in which the background is of a higher brightness than the foreground. The foreground consists of candidate text objects. In order to get an image containing only these candidate text objects a threshold is applied. This means that all pixels lighter than a certain brightness level are turned into a full white, and pixels under that level are turned black. This rises the question at what level the threshold should be.

One particular thresholding method developed by Otsu [Otsu, 1979] uses a statistical analyses of the histogram of an image to determine the level at which to put the threshold and is parameter free.

Applying the threshold to sections, generated by the section selection mechanism, has the advantage that faint text objects are more accurately defined compared to applying the threshold to the entire image. This difference is indicated by figure 2a and 2b, which feature the same section as show in figure 2c with the only difference being the global versus local thresholding approach. The text objects, particularly in the bottom left corner, are more clearly defined when the threshold is applied locally. This in turn allows for more accurate separation between text and non-text objects since the letters now consist of one complete component, and not separate tiny components which would be mistaken for noise.

2.3 Gathering numerical data

With the image properly pre-processed it is now possible to gather numerical data, this data is retrieved from connected components labelling, which will determine the

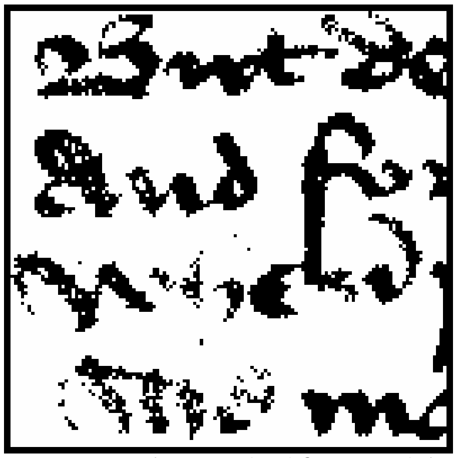


Figure 3a: a section resulted from applying Otsu threshold to the whole image.

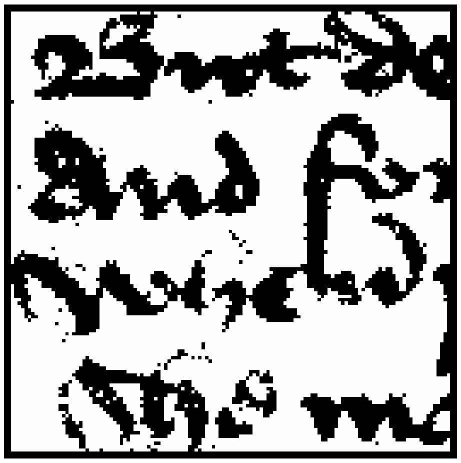


Figure 3b: a section with Otsu thresholding applied

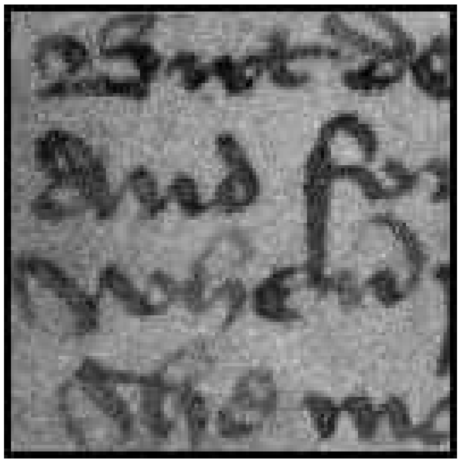


Figure 3c: the original section.

size, the number of pixels defining the object, and from boundary tracing, which returns the length of the outline of an object. Once this information is retrieved it can be used to separate text from non-text objects, as is discussed in section 2.4.

2.3.1 Connected Components Labelling

In order to be able to decide from object to object whether or not it is text, it is first required to identify individual objects. A common method for achieving this is connected components labelling. What it does is assigning a unique label, in this case an integer value, to all pixels of an individual component. A modification was made to also count the number of pixels the components is defined by, while labelling it.

There are several variations available of the connected components labelling method. The variation used here is of the four-connectivity recursive kind. Four-connectivity means that only pixels directly above, under and to the left and right are accessible. With Eight-connectivity, another common variation, the adjacent diagonals are also accessible. The reason why I settled for four-connectivity was quite simple. The input images are of a high resolution, so that weak links, diagonal connections (figure 4a) do not occur in text objects. Instead at high resolutions strong links, connections one or multiple pixels wide (figure 4b) will be far more abundant. The inherent reduction in if-then rules benefits the time required to process an image. The connected components method returns an image, or actually a two dimensional array, with all the objects uniquely labelled by integer numbers. This array provides the input for the boundary tracing method, how it is processed it described in the following subsection.

2.3.2 Boundary tracing

Provided with a two dimensional array containing the individual objects, each labelled by a unique integer number, it is now time to trace the boundary of each object, retrieving the length of the objects outline. This is done with a method described in an image processing textbook [Sonka et al, 1998], using the eight-connectivity variant. The chosen variant has the benefit that it reduces the amount of memory the method requires, thus optimising the speed at which the images are processed.

The lengths of the boundaries are acquired by tracing the objects with the unique label number as its target. Once an object is done, the next label number is located and the object traced, until all objects are traced.

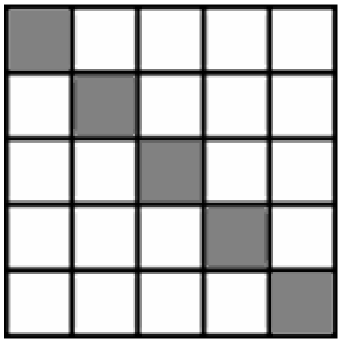


Figure 4a: A weak link, at four connectivity separate components will be labelled.

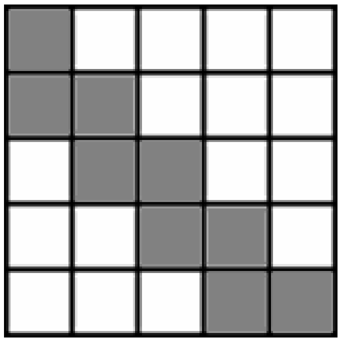


Figure 4b: A strong link, now the component will be seen as a whole.

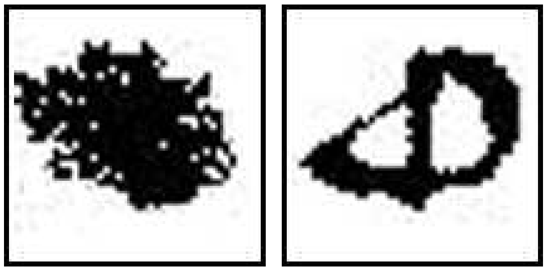
2.4 Separation among objects

Now having gathered information on the size of the surface of an object, the total number of pixels, and of the size of the boundary, the number of pixels in its outline, the next step is to use this information to distinguish between text and non-text objects.

For this a series of fairly simple ratios turn out to be surprisingly effective to discriminate between objects, as the results section will demonstrate. One has to note that these ratios are based upon connected components that are present within a section. By no means this guarantees that a large object, a line of connected handwriting for example, completely lies within a section. However, the piece of the object that does fit in the section is able to produce ratios that fit within the range of text objects.

Firstly there is the width versus height ratio. The width of a connected component is defined by the distance between the leftmost and rightmost pixels. To efficiently determine the width, these two extreme pixels are searched for in the list containing the pixels that define the objects boundary. This list, especially for larger objects, will contain far less items than a data structure holding all pixels of an object. This is done in a similar fashion for the topmost and bottommost pixels. I found that the width versus height ratio lies between 4.0 and 1.0 for text objects, present in a section. Testing with these values indicated that they work with different types of documents, from a range of different hand-writings.

Another ground for discriminating amongst objects is the ratio between the surface, the number of pixels defining an object (as returned by the connected components function) and the boundary (the number of edge pixels found by the tracing function). Figures 5a and 5b give an example of the difference between ratios in stains and letters. Here the difference between height and width ratios aren't strong enough to discriminate amongst the two, however there is enough difference between the surface to boundary ratio. Since the ratio between the height and width of a small object, as well as the surface to boundary ratio, tends to be close to 1.0, a minimal size requirement is placed upon candidate text objects. A minimal width and height of 7 pixels is required, as well as a minimum of 60 pixels defining the boundary. These values have proved to be effective for a large range of image resolutions.



Figures 5a and 5b: A stain and a letter respectively. The stain on the left has a surface to boundary ratio of 4.54 and a vertical to horizontal ratio of 0.788. The letter has a surface to boundary ratio of 3.14 and a vertical to horizontal ratio of 0.735. This sort of ratios are used to separate the two.

Dark edges, of the sort which may be produced in the scanning of a document may still sometimes be mistaken for text. To get rid off these, long components that extend to the full width or height of the section are ignored, as they will most likely not be text objects. If a text object does extend for such a length, it will at one end of it be judged to be text, since portions of text objects can also be identified as text when only a piece of it is contained in the currently selected section.

Also when a section that has been processed by the Otsu thresholding function contains more than 40% black pixels, it is very unlikely that it contains text objects. This is of course not necessarily true; it is possible that the section still holds some text objects as well as large stains or a piece of black border for example. The overlap in the sections mentioned earlier allows the text to get another chance at being evaluated, so the current, too dark, section is discarded.

When the mechanism reaches a section filled with text, it will look something like figure 6.

Even with a rather thick handwriting the percentage of black pixels does not exceed 30%. So it is very unlikely that the 40% threshold will misclassify actual text holding sections.

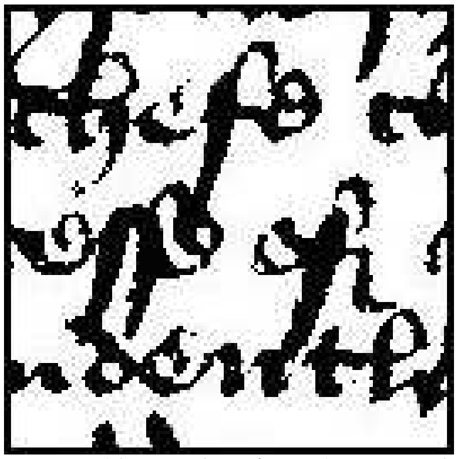


Figure 6: This section (129 by 129 pixels) contains a relatively large amount of black pixels, and only text-objects. The percentage of black pixels here equals 29.51%.

3. Results

In this section the results produced by the proposed method for text area marking is discussed. I will discuss some successes and of course also some shortcomings of this method, as there is still some work to do to complete it. For this I have some suggestions which are discussed in the next section. In the first subsection I will describe the results qualitatively, in the subsection after that I will describe a way of quantifying them.

3.1 Visual Results

In this subsection some results are

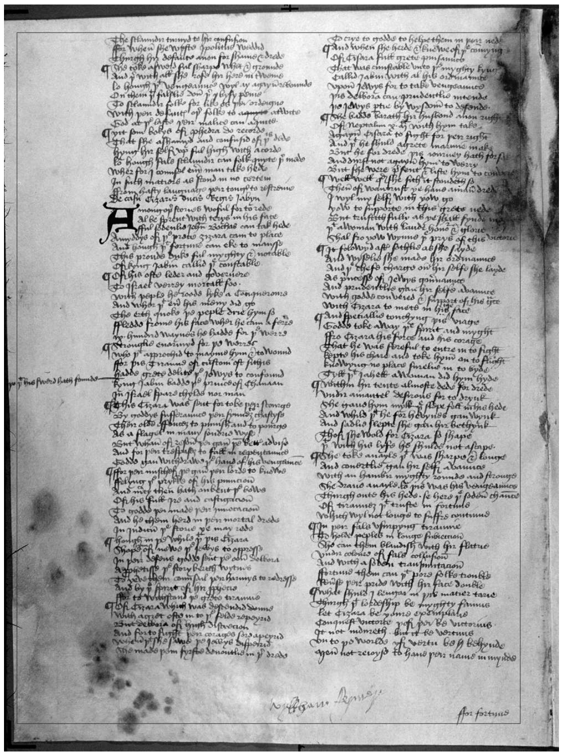


Figure 7a: A page with medieval handwriting on a deprecated medium.

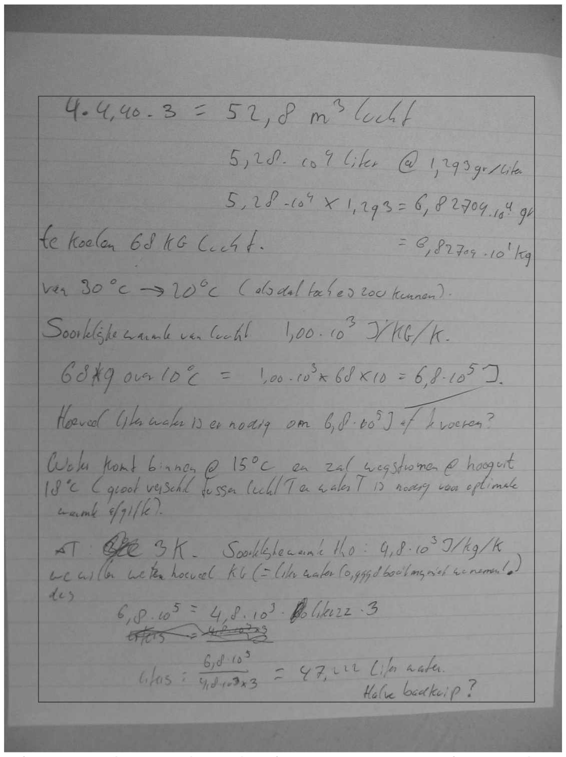


Figure 7b: A handwritten text, written by myself on lineated paper.

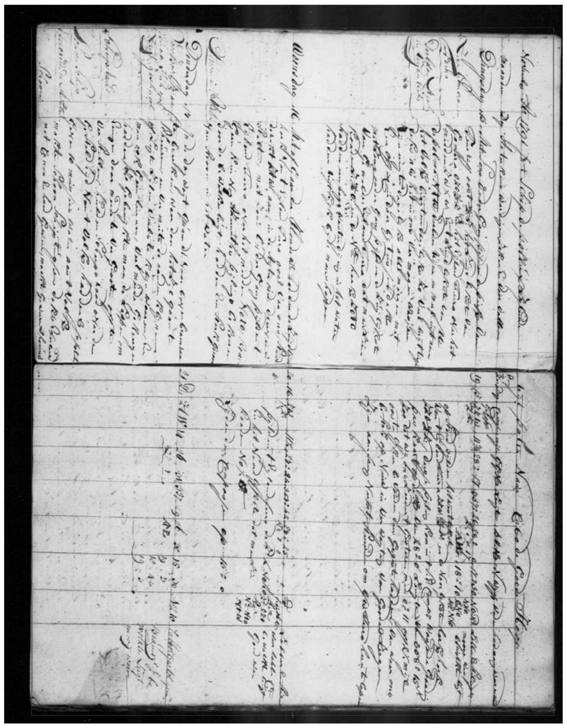


Figure 7c: Rotated 90 degrees clockwise for a better fit of a scanned VOC logbook entry with the text region too widely marked on the top, right and bottom side.

discussed illustrated by actual output from the text area marking method. This gives an intuitive insight in the performance the method displays.

To start off lets begin with a seemingly tricky image, demonstrated by figure 6a, with which the program deals pretty well. It is a page from a book written in a medieval style with a (by today's standards at least) very decorative handwriting. Notice how the rectangle appears to be too far to the left, this is due to the fact that there is some writing which is exceptionally far to the left side, but since it may contain valuable information, it is included.

Another image, figure 7b, which shows good results, is a page of my own handwriting. It isn't a scanned image, but actually a photograph.

These results seem promising, however this does not apply to every type of input. Some documents are polluted with stains that, even when thresholded and processed by the high-pass filter resemble text-objects. Unfortunately there are some types of documents, for which the historical document program is intended, that do contain this type of non-text objects. An example of this is illustrated in figure 7c, a very old logbook entry from a VOC-captain.

Objects in the paper edges are misiden-

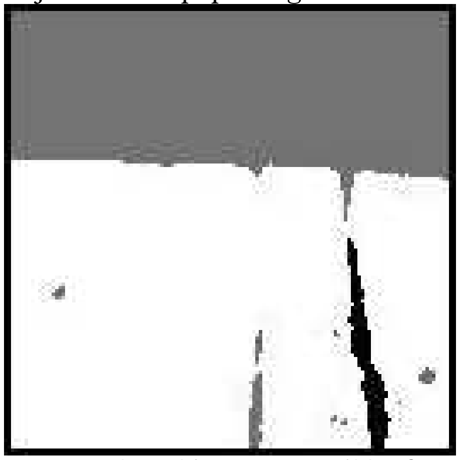


Figure 8a: The object responsible for misjudging the top edge.

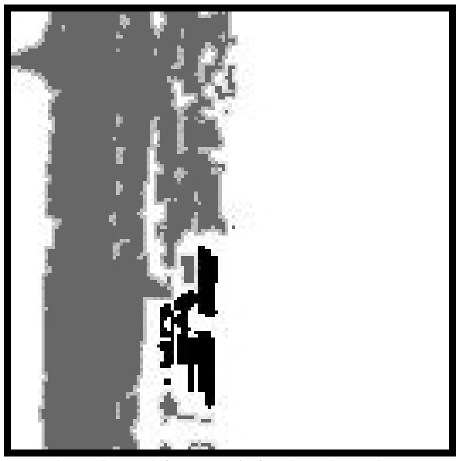


Figure 8b: The object misjudged to be a text object on the right side.

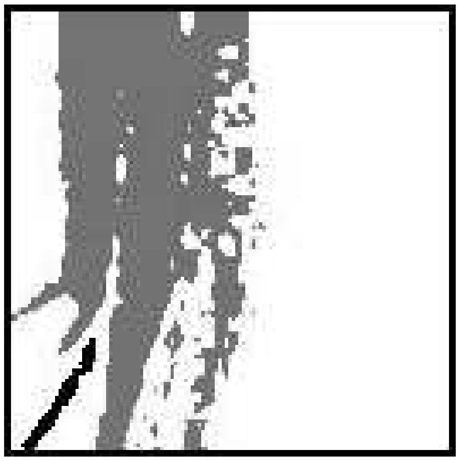


Figure 8c: The diagonally oriented object is mistaken for a text object on the bottom.

tified as text objects, for the top, right and bottom edges. A closer look at the actual sections presented to the program will shed some light as to why this happened. Figures 8a to 8c show where things went wrong for the top, right and bottom side respectively. The objects responsible are shown black. Firstly the top and bottom objects. Ideally the ratio between the width and the height of the object would disqualify these as text, but these objects are somewhat tilted, so the way in which the height and width are defined, the distance between the most far apart pixels, is unsuitable, even for the slightest rotation. The discussion section describes an alternative method to cope with this problem.

The object mistaken for a text object on the right side is by far the toughest to distinguish. With some imagination it could very well have been an actual letter. A profile, as discussed in the next section, of the component still would not be sufficient to base a judgement upon. However there are more techniques that can be applied, these are described in the discussion section.

The current discriminating features in place are apparently not enough to successfully mark the text region for all scanned documents Therefore the program displays a bias. This bias is a false positive bias, since at no time during trials text was wrongly cut off by the drawn rectangle. Instead the method tends to place the rectangle too widely around the text, including undesired non-text objects within it.

This bias is preferable to a false negative bias, which will result in discarding text, and thus throwing away information.

3.2 Quantifying the results

After having reviewed the actual output of the method the next step is to analyse this output in a quantitative manner.

An obvious way of quantifying the results would be to run the text area marking method on a wide range of scanned documents and then measure how far off the rectangle is per document. But what lies at the base of accurately drawing the rectangle is correctly separating text from non-text objects, per object. A way to analyse this is by running the method on a batch of documents and then inspect the sections the method judges to contain text on actually containing text.

A test run with documents from a variety of handwritings a total of 726 sections were judged to contain text, after manual inspection it was found that in 82 of those sections there was no text present. This means that the proposed method exhibits a 11.29 % false positive bias. The false negative bias, text mistaken for non-text, would be another good measure to express the performance of the method, but since in none of the test cases the text area was too narrowly drawn the only drawback of significance here is the false positive bias.

4. Discussion

In this section I will point out some improvements that could be done to the text area marking program. Firstly I will discuss some improvements that could be applied to parts already in place, and secondly I will discuss how other research that has been done on the subject of text detection may be used to extend the method I propose in this paper, to further increase the accuracy, and reduce the false positive bias.

4.1 Internal improvements

This subsection discusses a number of possible improvements on elements already in place in the proposed method, which due to time restraints are not yet implemented.

The current manner in which the section selection mechanism works is probably not the most efficient one. Because currently for every column, horizontally, and row, vertically, contact has to be made with the text, starting from the edge of the document. This means that a lot of sections, which most likely will not contain text have to be analysed. An optimisation would be to only let the first contact with the text be made in this way, and then do a trace around the text, in a similar way as the boundary tracing method, but on a larger scale. This would significantly reduce the amount of sections that the system would have to analyse. For an example see figure 9.

As mentioned in the previous section, fairly straight non-text objects tend to be identified as text objects because of the way the width and height are determined. A more intelligent approach would take the orientation of the objects into account. This could be done by finding the two contour pixels that are farthest apart and determining the orientation. The object could then be placed in a buffer and rotated so that it would be up right and then the traditional calculation could be performed. Or it could be left in its current orientation and the width be determined by finding the pixels that lie farthest apart from each other, perpendicular to the orientation line, as illustrated in figure 9. The latter method would be preferable if the profiling method discussed in the next subsection would also be implemented. This should take care the

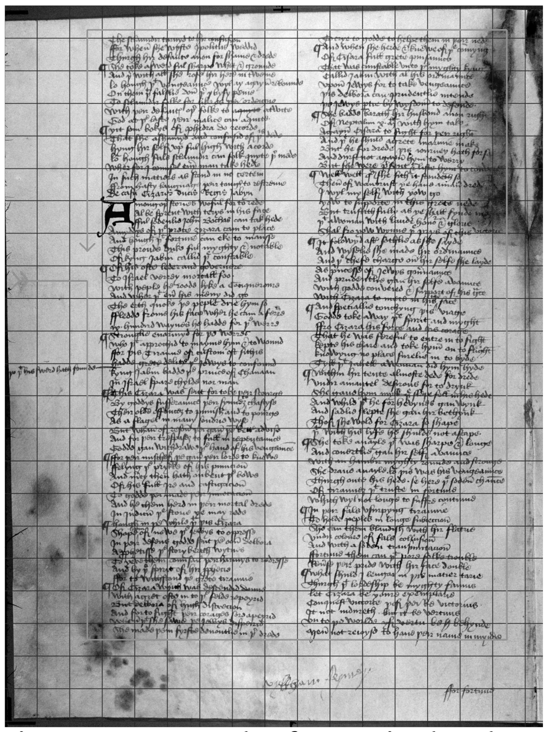


Figure 9: An example of an optimal pathway the section selection mechanism could follow by tracing the text area.

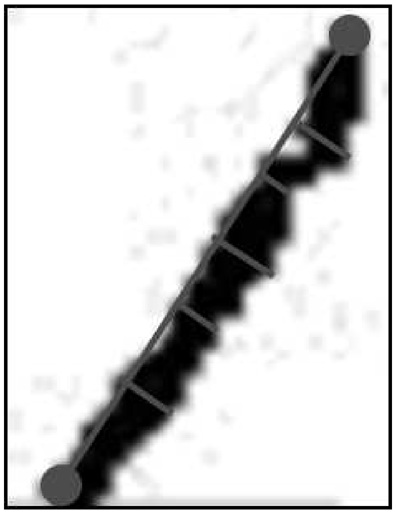


Figure 10: The object responsible for misjudging the bottom edge, enlarged and with marked orientation and perpendicular width measurement.

objects pointed out in figures 8a and 8c.

A way of dealing with objects the type of 8b remains to be found.

Arguably the width and height ratio will misclassify individual letters such as ‘l’ and ‘i’, since those letters also consist of fairly straight lines. Two properties of hand-written text help the proposed system to be accurate. Firstly, handwritten letters, which are supposed to be straight, tend to be written somewhat curved, this is due to the fact that most people rest their hand on the paper they write on, thus creating an articulation point much like a pair of compasses used in math class. The second property is the fact that most handwritten texts are written in columns. This means that unless all of the sentences start, or end, with the letters ‘i’ or ‘l’ and are misclassified, another, different letter at the same vertical position will ensure the correct position of the edge of the text area marking rectangle.

The current implementation of the program is written in Python, an easy to learn and widely applicable programming language, but unfortunately for some purposes extremely slow. A solution would be to write an implementation in the C programming language. A typical image, 20 mega pixels, takes around five minutes to process, typically C programs are at least a hundred times faster. Fortunately Python offers good C integrating capabilities.

Right now the system demands the resolution of the input to lie within a certain interval, full size images between three and twenty mega pixels have proven to be suitable, as well as portions of these images. Ideally no such demands would be laid upon the user with regard to the input. It is however imaginable that for a system a standard scanner would be used, resulting in the input being of the desired resolution, and thus supply the system with images of the desired resolution.

The current system only works on greyscale images. Support for colour images supplies a wealth of extra information, on which judgements can be based. It is not hard to imagine that in most cases stains and actual ink are of a different colour. A tool that would first search for a definite text object, and then set its colour as the target colour for the rest of the system would be a very valuable asset.

4.2 Improvements based on other research.

This section discusses some of the research performed in the area of text detection which may be adapted to be beneficial for the method proposed in this paper to increase the performance.

A lot of research has been done in the text detection area for the benefit of detecting machine written text. Some of the ideas this research produced could prove to be useful for detecting handwritten text as well.

Text has at least two distincting characteristics [Wu, Manmatha, Riseman 1997]. In this research the characteristics were discovered for machine written text, but to some extend the principles also apply to handwritten text. Firstly text has a certain frequency and orientation. This means that text could be regarded as a texture, and so a texture segmentation algorithm is used to find the location of the text. To deal with the variety of font sizes it can come across in a single image, it is processed at several resolutions. The results from this are called chips, areas containing text. These chips are then assimilated to hold text lines on the same strip. The authors admit that their method only works for documents as clean as newspapers, but perhaps it would be possible to combine ideas from the method I propose in this paper and the finding text in images method and create a version that can also cope with deprecated handwritten documents.

An application similar to the historical document searching program mentioned in the introduction uses profile features of complete words [Toni, Rath, Manmatha, Lavrenko 2004]. Figure 10 shows a graphical representation of the profile of the word "Alexandria" as was demonstrated in their paper. The authors propose a method for

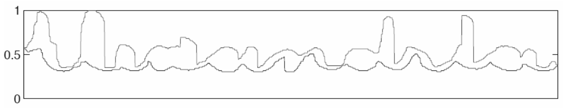


Figure 10: The upper and lower profile features of the word "Alexandria"

searching in historical documents without using a handwriting recogniser, instead profiles from a transcribed set of pages are matched with profiles found in the very large amount of untranscribed pages. I think that this profiling could also be used for determining whether or not an object is text or not. A function that collects profiles from a large set of handwritten texts without noise could be used to learn what sort of profiles belong to text. Then for example a neural network could be used to go about classifying detected objects in an input image.

5. Conclusion

The method proposed in this paper has proven that by benefiting from local analysis and simple features it is possible to do a fairly good job on detecting text. However, there remain documents that the current program cannot completely satisfactory process. The proposed improvements could help deal with the majority of the remaining documents that for now are too much of a challenge.

Greatly reducing the number of non-text objects in scanned documents the proposed method can still fulfil a niche in handwriting recognition and writer identification programs. Even though it is not yet perfect, it helps these programs concentrate their efforts on the actual text and thus increase their overall effectiveness.

Because of the consistent false positive bias the program exhibits it is also imaginable to tighten the current conditions placed upon candidate text objects and so increase the accuracy with which the text area is marked. However bear in mind that changing the ratios and other values means running extensive tests on a wide variety of input images as to assure that the text area marking program remains free of user provided parameters.

6. References

Sonka, M, Hlavac, V, Boyle, R (1998). *Image Processing, Analyses, and machine vision.*

Wu, V., Manmatha, R.,Riseman, E. *Finding Text in Images.* (1997) 20th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval, *pp. 3–12, 1997.*

Otsu, N. (1979) *A threshold selection method from grey level histograms.* IEEE Trans. Systems, Man and Cybernetics. Volume 9 p62-66.

Toni, M, Rath, R, Manmatha, Lavrenko, V (2004). *A search engine for historical manuscript images.* Proceedings of the 27th annual international ACM SIGIR conference on Research and

development in information retrieval,
July 25-29, 2004 Sheffield, UK