Using Readers' Highlighting on Monochromatic Documents for Automatic Text Transcription and Summarization

Ricardo da Silva Barboza
DES – CTG – UFPE
Recife, PE, BRAZIL
rsbarboza@gmail.com

Rafael Dueire Lins
DES – CTG – UFPE
Recife, PE, BRAZIL
rdl@ufpe.br, rdl.ufpe@gmail.com

Victor Matheus de S. Pereira
CEC – EST – UEA
Manaus, AM, BRAZIL
victor.msp18@gmail.com

***Abstract* — Very often interested readers highlight documents with felt pens. Such marking may be seen as personal view of the most important aspects of the document, which are used to instantly draw the readers' attention at. This article addresses ways of using the highlighting made by the reader of a book or a paper documents to automatically generate its text summary.**

Keywords — highlighting, paper documents, text summarization.

I. INTRODUCTION

Over the centuries, the interested readers often underlined texts to somehow emphasize parts of a text for further reference. Yukio Horie invented the modern felt-tip pen in Japan in 1962. A *marker pen*, *marking pen*, *felt-tip pen*, or simply a *marker*, is a pen which has its own ink-source, and usually a tip made of a porous material, such as felt or nylon. Highlighters, such as the one used in this sentence, are permanent markers filled with transparent fluorescent ink used to cover texts, emphasizing such content. Many highlighters come in bright, often fluorescent colors, which glow under a black light. The most common color for highlighters is yellow, but they are also found in blue, green, orange, and magenta varieties. Red highlighters can be purchased along with a green translucent sheet used to hide the highlighted material. Some yellow highlighters may look greenish in color to the naked eye. Table 1 presents the most usual colors of highlighters and the components they affect of the original text.

Table 1. Component alteration due to highlighting

Highlight	Color	Components
	Yellow	Blue
	Blue	Red/Green
	Green	Red/Blue
	Orange	Green/Blue
	Cyan	Red/Green
	Magenta	Red/Green/Blue

Today, most readers highlight texts instead of underlining them. Highlighters are used to take notes in textbooks and every day becomes more popular. On the one hand, annotations, underlining or highlighting parts of a document may be perceived as "noise" physically damaging the document [1]. On the other hand, the highlighted text offers a summary of the most important parts of a document.

To the best of the authors' knowledge, reference [2] provides the first solution for highlighting removal in monochromatic document images. Real highlighting removal is far more complex than one may imagine at first glance, because the ink fades, sometimes non-uniformly, and interacts with the paper background.

This paper enhances the algorithm presented in [2] to automatically generate summaries of documents with the author highlighted parts, as well as providing the possibility of removing it from the text image. The new algorithm was tested with the different colors of markers available in the market (Yellow, Blue, Green, Orange and Cyan). Besides that, a study on the optimization of the input parameters to allow maximizing the correct transcription rate by OCRs is performed.

II. MAKING SUMMARIES WITH THE HIGHLIGHTED TEXTS

Highlighting affects at least one RGB component of the original (non-highlighted image) lowering their intensity. Algorithm 1 tests if the components of a (originally) monochromatic document which was highlighted and scanned as a color document are further apart from each other than a controlled value which is named *distance*. The pixels with whose value of the RGB component goes beyond the *distance* threshold are assigned to the value of the highest intensity RGB component. The suggested value for the parameter distance equals 10.

Algorithm 1 – Pseudo-code for the algorithm that removes highlighting that affects one or two primary RGB component in monochromatic documents.

```
for j <- 0 to image.height - 1 do begin
  for i <- 0 to image.width - 1 do begin
    red <- image.pixel[i,j].red;
    green <- image.pixel[i,j].green;
    blue <- image.pixel[i,j].blue;
    rg <- |red – green|;
    rb <- |red – blue|;
    gb <- |green – blue|;
    if ((rg > distance) or (rb > distance) or (gb > distance)) then
    begin
      if ((red >= blue) and (red >= green)) then
        image.pixel[i,j].color <- (red, red, red);
      if ((green >= red) and (green >= blue)) then
        image.pixel[i,j].color <- (green, green, green);
      if ((blue >= red) and (blue >= green)) then
        image.pixel[i,j].color <- (blue, blue, blue);
    end;
  end;
end;
```

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.51

For cropping the highlighted text areas, Algorithm 1 must be suitably modified. Readers often highlight several pages in the text, thus the algorithm may need to process several page images of highlighted text simultaneously. The current line of the resulting image is stored in an integer variable called "*aux*". Decision making to check if a pixel was affected by highlighting follows the same logic presented in Algorithm 1: if the value of the difference between the RGB components exceeds the threshold *distance*, then the logical variable *flag* is set as *true*. The variable *flag* keeps the information if the pixel which is under analysis is after a highlighted pixel in the same line. The decision if a pixel is text or background is stored in the variable called *threshold*. Pixel values below the variable *threshold* are considered text, while above it is considered background. If a pixel of text follows a highlighted pixel then the pixel in the same column and line controlled by variable y is set to black (text) in the final (summary) image.

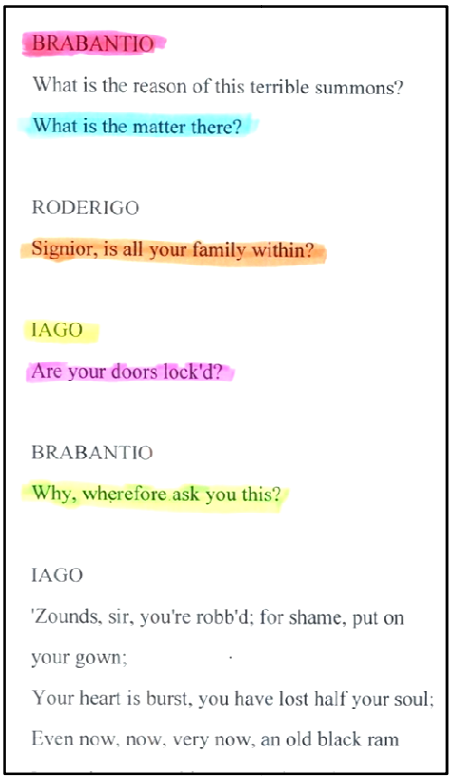
Algorithm 2 – Pseudo code of the algorithm for creating the summary image with the highlighted parts of the text.

```
aux  <- -1;
flag <- false;
for all images to be processed do begin
  for j <- 0 to image_processed.height - 1 do begin
    for i <- 0 to image_processed.width - 1 do begin
      red <- image_processed.pixel[i,j].red;
      green <- image_processed.pixel[i,j].green;
      blue <- image_processed.pixel[i,j].blue;
      if ((red + green + blue) div 3) > threshold) then
        rg <- |red – green|;
        rb <- |red – blue|;
        gb <- |green – blue|;
        if ((rg > distance) or (rb > distance) or (gb > distance)) then
          begin
          if aux <> j then begin
            aux <- j;
            y <- y + 1;
            brief_image.new_line;
          end;
          flag <- true;
        end
        else begin
          if flag then flag <- false;
        end;
      end else begin
        if flag then
          brief_image.pixel[i,y] <- black;
      end;
    end;
  end;
end;
```

Figure 1 (a) presents a highlighted text and (b) shows the result of automatically cropping the highlighted image under processing with Algorithm 2.

The tests performed showed that the value of the parameter *distance* close to 35 yielded the best final results.

If the color of the marker is known a priori, Algorithm 2 may be optimized in the checking between color intensity of the RGB components. For instance, if a yellow highlighter is used, according to reference [2] one knows that only the blue component is affected. Then one may need only to check the variation between the red-blue and green-blue components.



(a)

BRABANTIO
What is the matter there?
Signior, is all your family within?
IAGO
Are your doors lock'd?
Why, wherefore ask you this?

(b)

Figure 1 – (a) Original image. (b) Image processed with Algorithm 2. Parameters: *threshold* = 170; *distance* = 35.

III. THE PARAMETER *THRESHOLD* AND ITS INFLUENCE IN THE PERFORMANCE OF OCRS

If the summary image is to be latter processed via OCR a few items need to be analyzed to yield a good transcription rate [6][7]. They are: quality of the original text, skew correction, quality of image binarization. As the binarization CUT-off point is controlled by variable *threshold*, this section analyses its effect in the quality of the summary image. Figure 2 presents three results obtained by the variation of this parameter. One may observe that the lower the value of *threshold* the fader the text will appear in the summary image, losing text pixels even. A high value for *threshold* implies in leaving parts of the highlight as noise in

the summary image heavily degrading the OCR performance.

For a better analysis of the most suitable value for the *threshold* parameter its value was varied between 0 and 255 in 10 different page images, totaling 1,250 characters affected by highlighting. Figure 3 presents the correct transcription rate for character recognition of the summary image. The OCR used was Cuneiform PRO OCR 6.0 [3].

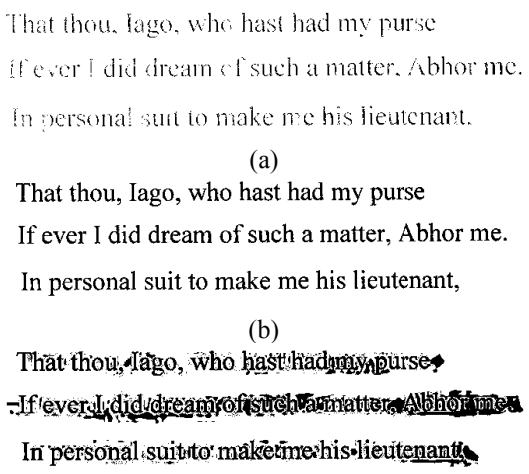


Figure 2 – Influence of the parameter *threshold* in a summary image. (a) *threshold* = 70. (b) *threshold* = 170. (c) *threshold* = 210.

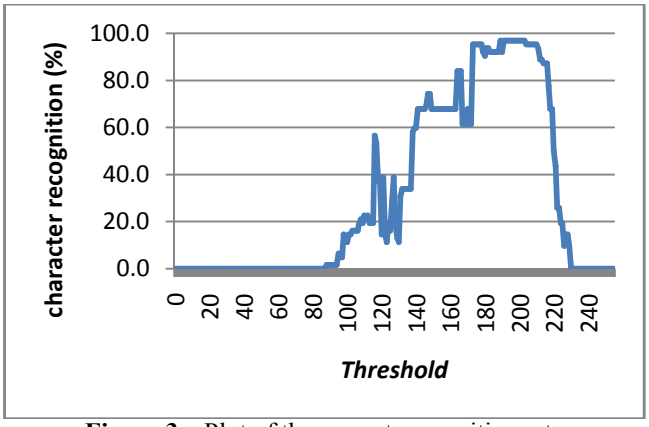


Figure 3 – Plot of the correct recognition rate of characters with the variation of the parameter *threshold.*

The best transcription results were obtained by setting the value of *threshold* between 190 and 200.

IV. Uneven Highlighting and Pre-processing

Readers whose mother tongue is a language that use the Latin alphabet, and texts written in the same alphabet, highlighting tends is performed in a similar way to cursive writing: horizontally from left to right. Not always readers are careful enough during highlighting to generate a uniformly marked text. Sometimes the marker ink is low causing a faded highlighted area or the reader does not hold the marker pen in complete contact with the paper surface. These factors may cause non-uniform marking, which sometimes may not be perceptible to the naked eye. Figure 4 presents a zoom at a part of an unevenly highlighted document, for which it is difficult to observe that some parts of the "S" are not painted by the yellow marker.

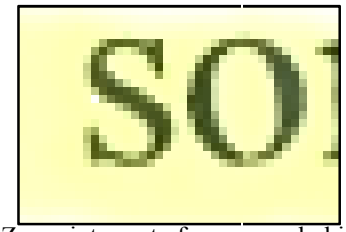


Figure 4 – Zoom into part of an unevenly highlighted text.

Uneven highlighting yields images not suitable to be processed by Algorithm 2. The irregular highlighted surface becomes perceptible in the resulting image. For instance, the image in Figure 4 processed with Algorithm 2 yields the image shown in Figure 5, in which it is easy to see that the "S" became split into two regions.

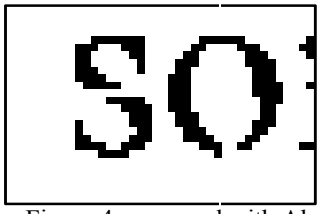


Figure 5 – Figure 4 processed with Algorithm 2.

Figure 6 shows an example of the result of uneven highlighting in a line of a summary image, in which one may observe that the "artifact" often spans through the whole line, making some works completely unreadable even for humans. The effect of the automatic transcription of such part of the image is disastrous, as anyone can imagine.

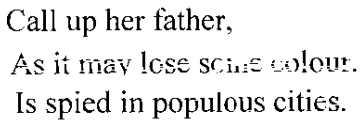


Figure 6 – Summary image from uneven highlighted text processed with Algorithm 2.

Analyzing the "dynamics" of Algorithm 2 one may observe that such artifact appears in the summary image when the highlighted text has "holes" in the marker painting, because the pixels to the right of the hole are not copied into the summary image. One possible solution is to "correct" the highlighting in the original document, if it is still available.

Algorithm 3 – Pseudo code of the algorithm for pre-processing the image.

```
y <- -1; aux <- -1;
for j <- 0 to image_processed.height - 1 do begin
  copy <- false;
  for i <- 0 to image_processed.width - 1 do begin
```

```
    if not(copy) then begin
      red <- image_processed.pixel[i,j].red;
      green <- image_processed.pixel[i,j].green;
      blue <- image_processed.pixel[i,j].blue;
    end;
    if copy or (abs(r - b) > distance) then begin
      if aux <> j then begin
        aux <- j; y <- y + 1;
      end;
      copy <- true;
      img_aux.pixels[i,y] <- image_processed.pixels[i,j];
    end; end; end;
image_processed <- img_aux;
for j <- 1 to image_processed.height - 2 do begin
  for i <- 1 to image_processed.width - 2 do begin
    red <- image_processed.pixel[i,j].red;
    green <- image_processed.pixel[i,j].green;
    blue <- image_processed.pixel[i,j].blue;
    rg <- |red – green|;
    rb <- |red – blue|;
    gb <- |green – blue|;
     if ((red + green + blue) div 3) > threshold) and
       ((rg > distance) or (rb > distance) or (gb > distance)) then
begin
      red <- image_processed.pixel[i,j-1].red;
      green <- image_processed.pixel[i,j-1].green;
      blue <- image_processed.pixel[i,j-1].blue;
      if ((red + green + blue) div 3) > threshold
        image_aux.pixel[i,j-1] <- RGB(red, green, 0);
      red <- image_processed.pixel[i,j+1].red;
      green <- image_processed.pixel[i,j+1].green;
      blue <- image_processed.pixel[i,j+1].blue;
      if ((red + green + blue) div 3) > threshold
        image_aux.pixel[i,j+1] <- RGB(red, green, 0);
      red <- image_processed.pixel[i-1,j].red;
      green <- image_processed.pixel[i-1,j].green;
      blue <- image_processed.pixel[i-1,j].blue;
      if ((red + green + blue) div 3) > threshold
        image_aux.pixel[i-1,j] <- RGB(red, green, 0);
      red <- image_processed.pixel[i+1,j].red;
      green <- image_processed.pixel[i+1,j].green;
      blue <- image_processed.pixel[i+1,j].blue;
      if ((red + green + blue) div 3) > threshold
        image_aux.pixel[i+1,j] <- RGB(red, green, 0);
    end; end; end;
  image_processed <- image_aux;
```

A neater solution is provided by pre-processing the document finding the neighboring of each pixel. Algorithm 3 scans the original document and generates a color summary image with all highlighted blocks in the document. Then the second step is to correct the unevenness of the marker highlighting by making it uniform, thus making it suitable to de processed by Algorithm 2.

Pré-processing the image in Figure 4 with Algorithm 3 and cascading the resulting image in Algorithm 2 one obtains the image shown in Figure, in which the segmentation of the “S” disappears.

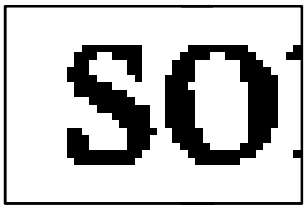


Figure 7 – Figure 4 pre-processed with Algorithm 3 and filtered with Algorithm 2

The unevenness and “holes” in the text of the summary image of Figure 6 were removed and the text in the image of Figure 8 looks as if it were in boldface.

Call up her father,
As it may lose some colour.
Is spied in populous cities.

Figure 8 – Summary image of Figure 6 with pre-processing.

The pre-processing step performed by Algorithm 3 which removed the unevenness of the highlighting also yields effects in the transcription rate by OCRs, as shown in Figure 9. The best recognition rates are shifted left wise in the graph for values between 127 and 156 of threshold. One may also observe that pre-processing increased the overall recognition rate reaching almost 100% correct character recognition.

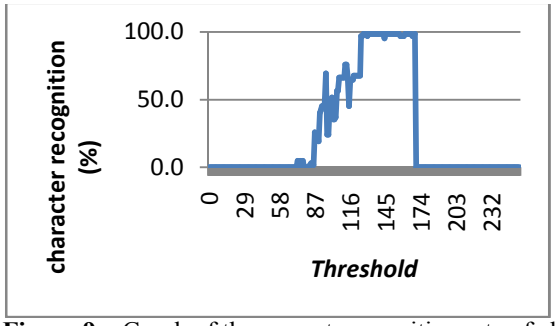


Figure 9 – Graph of the correct recognition rate of characters for the parameter *threshold* after pre-processing.

V. Performance

Algorithm 2 has linear asymptotic complexity O(n) with the number of pixels to be processed in the input image. For images with 0.3 Mpixels the processing time for each input image in a computer with c.p.u. AMD Turion Mobile Technology MK-36 1.99 GHz with 1.75 GB of RAM, was 2.2 seconds. Processing images with 3.8 Mpixels the elapsed time was 23.1 seconds per image in the same platform.

The graph of Figure 10 exhibits the average processing time for 20 runs of Algorithm 2 in images of 0.3 Mpixels. It shows that adding more images to the processing batch the rise in processing time is linear and in the hardware platform used it takes around 5.9 seconds for each Mpixel of input.

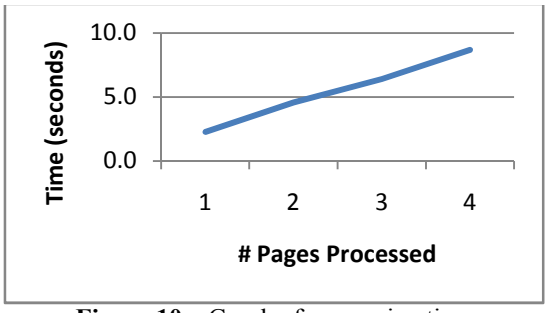


Figure 10 – Graph of processing time elapsed for images of 0.3 Mpixels.

Pre-processing the image with Algorithm 3 implies in a processing overhead of 1.6 times the time elapsed in using Algorithm 2 only, on average. If unevenly highlighted areas appear frequently, this overhead is worthwhile as the correct character transcription rate by OCR largely increased for those areas.

VI. Conclusions and Lines for Further Work

This paper presents an efficient algorithm for the automatic removal of highlighting besides making possible the generation of images that are formed only with the highlighted texts as a summary of the original input. The input document must be originally monochromatic and highlighted with a felt-pen marker of any color commercially available, either fluorescent or not. The input document should be scanned in 200 d.p.i. true color. The summary image may be automatically transcribed using a commercial OCR with good results.

If the marker paint is uneven for some reason (low ink level, incorrectly holding the pen, etc.) the resulting summary image exhibits "holes" that degrade the character recognition rate. A pre-processing scheme introduced herein besides correcting such "artifact" yielding a better summary image highly increases the OCR performance, reaching almost 100% correct character recognition.

Another optimization already implemented, which was omitted here due to space restrictions, is to allow a different treatment for the different colors of the marker. This allows the reader to have the option of generating different texts according to the color used. For instance, some readers highlight parts of the text he agrees with in yellow and in red the ones he does not agree. The Tool would give the option of either generating one single text or separating them, adding new flexibility.

Several lines for further work are already in progress. The first of them is related to correctly reformatting two column documents in the generation of the summary image. Another research path also being currently pursued is mapping the color of the marker used for highlighting in the color of the font of the transcribed text. This can be done by introducing a post-processing phase in which the color of the original highlighting would be mapped into a different color in the transcribed text. A more ambitious, by far a more complex research line, is addressing the case of color background highlighted documents for marker removal and summarization. In general, separating text and background in degraded documents is a complex task [4], highlighting makes it even more complex.

A different research path that may also be explored is the use of a noise classifier, such as the one described in [5], to select the highlighted areas for different colors of markers. The highlighted pixels would be processed by a simplified version of Algorithm 2 and Algorithm 3.

One of the referees in ICDAR 2011 to this paper drew the authors' attention to the existence of two patents that address the same problem [7][8]. The solution proposed here seems to be more general (as different colors of markers are allowed) than the one in the patents [7][8], that seems to focus only in yellow highlighting. Besides that, for the yellow marker the solution in this paper seems to be more time efficient, whose effectiveness in is still to be examined as those documents do not present any real example.

The code for the algorithms and test images are available by requesting to the first author of this paper.

Acknowledgments

V.M.S.P. acknowledges the research student grants received from Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM, BRAZIL.

References

[1] Lins, R.D. A Taxonomy for Noise Detection in Images of Paper Documents - The Physical Noises. ICIAR 2009. LNCS v. 5627. p. 844-854, Springer Verlag, 2009.

[2] Barboza, R.S., Lins, R.D., Mattos, V. S. *Removing Highlighting in Paper Documents*. VII IEEE International Telecommunications Symposium, ITS2010. Manaus-AM, Brazil, September 2010.

[3] Cuneiform PRO OCR 6.0. http://cuneiform-pro-ocr.softwareandgames.com/ visited 21/02/ 2011.

[4] Leedham, G., Varma, S., Patankar, A., Govindaraju, V., *Separating text and background in degraded document images—a comparison of global thresholding techniques for multi-stage thresholding*, Proceedings of the Eighth International Workshop on Frontiers in Handwritten Recognition, pp. 244–249, 2002.

[5] Lins, R.D, Silva, G.F.P., Banergee, S., Kuchibhotla, A. and Thielo, M. *Automatically Detecting and Classifying Noises in Document Images*, ACM-SAC`2010, ACM Press, v.1. p.33 – 39, March 2010.

[6] Gatos, B., Papamarkos, N., and Chamzas, C. Skew detection and text line position in digitized documents. Pattern Recognition, V 30(9): 1505-1519, 1997.

[7] Mello, C. A. B. de, Lins, R. D. A Comparative Study on Commercial OCR Tools In: Vision Interface'99, 1999, Québec. v.1. p.224 – 232.

[8] R. Nagarajan, et al. Automated Method for Extracting Highlighted Regions in Scanned Souce. U.S. Patent 2007/0253620, Nov. 1, 2007.

[9] R. Nagarajan. Automated Method and System for Retrieving Documents Based on Highlighted Text from a Scanned Source. U.S. Patent 2007/0253643, No. 1, 2007.